Разработка тестового набора для верификации реализаций протокола безопасности IPsec v2

А.В. Никешин, Н.В. Пакулин, В.З. Шнитман

Аннотация. Статья посвящена разработке тестового набора для проверки соответствий реализаций узлов Интернет спецификациям нового протокола безопасности IPsec v2 [1-7]. Для построения тестового набора использовалась технология автоматического тестирования UniTESK [8] и программный пакет CTesK [9], реализующий эту технологию.
Работа выполнялась в Институте системного программирования РАН в рамках проекта «Верификация функций безопасности протокола нового поколения IPsec v2» при поддержке гранта РФФИ № 07-07-00243. В ходе ее выполнения были выделены требования к реализациям IPsec v2, разработаны формальные спецификации и прототип тестового набора для верификации реализаций IPsec v2, в том числе реализаций протокола автоматического создания контекстов безопасности IKEv2. В статье описаны метод формализации требований IPsec v2, процесс создания тестового набора, а также результаты тестирования существующих реализаций. Эти результаты показывают, что предложенный в данной работе метод верификации позволяет эффективно автоматизировать тестирование таких сложных протоколов, как протоколы безопасности.

1. Введение

Под семейством протоколов IPsec обычно понимается вся совокупность технических средств, обеспечивающих защиту передачи данных на уровне IP. В настоящее время эксплуатируются две несовместимые версии IPsec, которые обычно обозначаются как IPsec v1 и IPsec v2. В данной статье под IPsec мы будем понимать исключительно IPsec v2.

Семейство протоколов IPsec представляет собой инфраструктуру безопасности: спецификации IPsec v2 включают описание архитектуры обеспечения безопасности передачи данных в сетях IP, собственно протоколы защиты данных Authentication Header (AH) и Encapsulating Security Payload (ESP), протокол автоматического установления защищённого соединения Internet Key Exchange (IKEv2), а также криптографические алгоритмы аутентификации и шифрования [1-7].

Семейство IPsec предоставляет целую совокупность сервисов безопасности: контроль доступа, целостность в режиме без установления соединения (целостность дейтаграмм), аутентификацию источника данных, обнаружение и отклонение повторно воспроизводимых сообщений (вид частичной целостности последовательности сообщений), конфиденциальность данных и ограниченную конфиденциальность потоков трафика. Эти сервисы предоставляются на сетевом уровне и обеспечивают прозрачность для протоколов транспортного и верхнего уровней. IPsec предлагает единый стандартный метод защиты для всех протоколов, которые могут работать поверх IP (включая сам IP). Кроме того, IPsec v2 включает спецификацию минимальной функциональности межсетевых экранов, поскольку она является существенным аспектом организации контроля доступа на уровне IP.

Задачу тестирования соответствия можно условно разделить на две подзадачи: построение тестовых воздействий и оценка правильности наблюдаемых результатов. К первой задаче примыкает проблема оценки полноты покрытия – чем шире будет спектр тестовых воздействий, тем шире получится охват функций протокола при тестировании. Вторая задача заключается в вынесении вердикта о соответствии тестируемой системы спецификации протокола.

Разработанный метод верификации [10] основан на автоматизированном тестировании соответствия формальным спецификациям. Требования, представленные в тексте стандарта, изложены на английском языке и представляют собой неформальный текст, описывающий желаемое поведение системы на естественном языке. Для того, чтобы автоматизировано извлечь тесты для протокола, необходимо перевести его спецификацию в вид, пригодный для решений этой задачи. В разработанном подходе в этой роли выступают формальные спецификации, в которых требования задаются как логические выражения, записанные посредством математического формализма.

В основе подхода лежит представление протоколов как асинхронных автоматов, причем автомат задаётся неявно посредством контрактных спецификаций. А именно, формальное описание поведения протокола задаётся как контрактная спецификация: набор сообщений протокола рассматривается как некоторый формальный интерфейс между реализацией протокола и её окружением, поведение протокола описывается посредством пред- и постусловий. Такое задание протоколов позволяет описывать поведение сложных недетерминированных протоколов, таких как протоколы защиты передачи данных в сетях IP. Эти протоколы отличаются сложными структурами данных сообщений и состояния, недетерминированным поведением и неполными спецификациями – в спецификациях умышленно оставлены пробелы для облегчения реализации протоколов. Контрактные спецификации позволяют представлять требования к сложным протоколам в форме, пригодной для автоматизированного тестирования реализаций таких

протоколов. Для протоколов, формальная спецификация которых задана в виде контрактной спецификации, разработан метод автоматизированного построения тестовых последовательностей с полностью автоматическим вынесением вердиктов о соответствии наблюдаемого поведения реализации её спецификации. В совокупности разработанные методы позволяют автоматизировать тестирование соответствия реализаций сложных сетевых протоколов их спецификациям.

В рамках предложенного метода тест представляет собой конечный автомат. С каждым переходом автомата сопоставлено определённое тестовое воздействие. При выполнении перехода это воздействие подаётся на тестируемую реализацию, регистрируются реакции реализации и автоматически выносится вердикт о соответствии наблюдаемого поведения спецификации. Обход автомата теста совершается автоматически во время тестирования, алгоритм обхода не зависит от протокола, тестируемой реализации или конкретного теста. Последовательность переходов определяет тестовую последовательность. В силу недетерминизма протоколов и различий в поддержке необязательных функций в реализациях конкретные тестовые последовательности, получаемые при прогоне тестов на разных реализациях, могут не совпадать друг с другом.

Предложенный подход к верификации функций безопасности включает два метода: метод формализации стандартов протоколов и метод формального задания тестовых наборов [11]. Метод формализации стандартов протоколов включает анализ спецификации протокола и извлечение требований, определение формального интерфейса протокола, формализацию функциональных требований к реализации протокола, задание критериев покрытия и разработку функции реконструкции состояния. Метод формализации тестовых последовательностей состоит из определения целей тестирования, разработки проекта автомата теста для конкретной цели тестирования, задания переходов автомата теста, задания функции определения состояния автомата теста по модельному состоянию контрактной спецификации протокола, проектирования настроечной информации автомата теста и разработки формата для представления опций, а также включает прогон тестового сценария и анализ результатов тестирования. Для описания автоматов тестов в данном методе используется специальный вид задания автоматов тестов, который называется тестовым сценарием.

2. Обзор процесса формализации спецификаций IPsec

2.1. Извлечение требований

Процесс формализации стандартов начинается с извлечения требований – утверждений, определяющих наблюдаемое поведение реализации протокола в зависимости от внутреннего состояния реализации и входных воздействий, оказываемых на неё окружением. Требования к протоколам Интернета

изложены на английском языке в документах, называемых по историческим причинам «запросами комментариев» (Request for Comments, RFC)

Для построения формальной спецификации IPsec прежде всего необходимо из этих документов извлечь требования, регламентирующие поведение реализаций протоколов. Процесс извлечения требований связан с решением четырех задач: определение набора документов, регламентирующих протокол, идентификация функциональных требований в регламентирующих документах протокола, сведение требований в каталог и систематизация требований. Ниже этот процесс описан более подробно.

2.1.1. Определение набора документов, регламентирующих протокол

Регламентирующие документы IPsec разрабатываются Комитетом по стандартизации Интернета (Internet Engineering Task Force, IETF). Спецификации протоколов Интернета публикуются в формате IETF RFC [12] на английском языке. Анализ структуры документации начинается с изучения обзоров протоколов Интернета. Цель изучения – обнаружить связи между протоколами и составить список RFC для подробного изучения. В этот список должны быть занесены RFC, которые регламентируют сам протокол, его расширения и дополнения, а так же другие протоколы Интернета, с которыми протокол взаимодействует.

К настоящему моменту разработано более 50 документов, регламентирующих различные аспекты IPsec – применение криптографических алгоритмов, параметры настройки, базовые протоколы и т.д. К основным документам, регламентирующим IPsec v2 относятся спецификации [1-4].

2.1.2. Идентификация функциональных требований

Для обеспечения прослеживаемости требований, т.е для создания ссылок из разработанных формальных спецификаций в требования из регламентирующих документов, необходимо представить требования как уникальные именованные сущности. В данной работе эта задача решена составлением каталога требований.

Каталог требований – это список требований, налагаемых регламентирующими документами и общим контекстом предметной области. Каждое требование, извлечённое из документации, каталог привязывает к соответствующему месту в тексте. Одно такое требование обычно соответствует логически замкнутому фрагменту текста, выражающему одно ограничение на элемент функциональности или элемент данных. Каждое требование имеет уникальный идентификатор.

Каталог требований позволяет в дальнейшем оценивать полноту формализации и адекватность тестирования в терминах исходного текста стандарта и предметной области.

2.1.3. Систематизация требований

Систематизация требований призвана разделять требования на группы «похожих» требований. Поскольку конечной целью является разработка формальной спецификации требований, систематизация предназначена для выделения групп требований, которые формализуются сходным образом.

1. Разделение требований на **ограничения поведения и ограничения целостности**. Под ограничениями поведения мы понимаем требования к поведению реализации протоколов IPsec в определённых условиях или некоторой ситуации. Под ограничениями целостности мы понимаем требования к связям между элементами состояния протокола, между полями сообщений протокола и т.п., которые должны выполняться на протяжении всего времени функционирования системы или в большом количестве ситуаций, возникающих при работе реализации.

2. Разделение требований по **объектам требований**. Требования в спецификации предъявляются к поведению различных компонентов или объектов, входящих в архитектуру IPsec. Требования, описывающие поведение одного объекта, удобно выделять в одну группу, это облегчает использование каталога требований в последующих работах процесса.

3. Разделение по **степени обязательности**. Этот критерий отражает тот факт, что во многих протоколах выделяются функции протокола, которые должны быть представлены в каждой реализации, и необязательные для поддержки в реализации. В спецификациях протоколов Интернета выделяется три уровня обязательности [13]:
   - Обязательные – требования из этой категории должны быть представлены в каждой реализации протокола. Такие требования помечаются в тексте спецификации ключевым словом MUST (негативные требования помечаются фразой MUST NOT).
   - Рекомендуемые – реализации могут нарушить требования из этой категории, но только если есть веские причины так поступить, и авторы реализации отдают себе отчёт обо всех последствиях такого шага. Такие требования помечаются ключевыми словами SHOULD и RECOMMENDED (негативные требования помечаются фразами SHOULD NOT и NOT RECOMMENDED).
   - Необязательные – авторы реализации могут не включать поддержку требований из этой категории, если, скажем, это не требуется в тех условиях, в которых будет использоваться реализация. Впрочем, есть одно ограничение «свободы выбора» – реализации, в которых отключена поддержка некоторого требования из числа необязательных, должны (MUST) быть в состоянии взаимодействовать с реализациями, в которых поддержка такого требования включена. Аналогично, реализация, в которой включена поддержка некоторого необязательного требования, должна (MUST) быть в состоянии взаимодействовать с реализациями, в которых поддержка требования отключена.

4. Разделение требований на **позитивные и негативные**. Под позитивными требованиями здесь понимаются требования, предписывающие реализации совершить определённые действия, например, отправить сообщение в сеть или возвратить некоторый код ответа. Под негативными требованиями понимаются требования, которые запрещают реализации совершать определённые действия.
   Например, спецификация архитектуры IPsec ЗАПРЕЩАЕТ реализации использовать одновременно нулевой алгоритм шифрования и нулевой алгоритм цифровой подписи.
   Этот критерий важен при формализации, так как позитивные и негативные требования формализуются различным образом. Данный критерий делит требования на два класса.

Указание категории, к которой относится требование, включается в каталог требований. В рамках отдельного критерия требование принадлежит только к одной категории, но часто встречается ситуация, когда одно требование подпадает под несколько критериев. В таком случае в каталог заносятся все выделенные категории.

2.1.4. Классификация требований по объекту требований

Мы вводим следующее деление требований в зависимости от объектов требований.

1. Требования к связям между полями сообщениями протокола.
2. Ограничения целостности состояния протокола.
3. Требования к обработке входящих сообщений.
4. Требования к содержимому исходящих сообщений.
5. Требования к обработке последовательностей сообщений. Фактически, это требования к отдельным переходам в автомате протокола, но у многих протоколов Интернета требования к обработке входящих пакетов и заполнению исходящих пакетов представлены без какой-либо сопутствующей модели состояния и переходов. При выделении требований из этой группы необходимо реконструировать автомат протокола.
6. Требования к взаимодействию с протоколами нижнего уровня. Сюда входят
   - правила представления сообщения протокола в пакетах нижележащего уровня,
   - правила отображения адресов протокола в адреса нижележащего уровня,
   - правила заполнения полей в сообщениях протокола в зависимости от конфигурации протокола нижележащего уровня

(например, использование протоколов шифрования и цифровой подписи в протоколе IKE).

7. Требования к взаимодействию с протоколами верхнего уровня. Сюда входят
   - правила представления сообщений протокола верхнего уровня в сообщениях рассматриваемого протокола,
   - управление состоянием протокола со стороны верхнего уровня,
   - сервисы, которые протокол предоставляет верхнему уровню
8. Требования к программному интерфейсу реализации IPsec. В спецификации протокола есть указания на требования к реализации программного интерфейса, но нет описаний типов данных и сигнатур соответствующих функций.

В данной работе из основополагающих документов IPsec v2 [1-7] было выделено в общей сложности 850 функциональных требований, из них 60% обязательных (категории MUST), 21% рекомендуемых (категории SHOULD), и 19% опциональных требований (категории MAY).

2.2. Формализация требований спецификации

2.2.1. Определение формального интерфейса протокола

Перед формализацией требований необходимо определить формальный интерфейс IPsec v2. А именно, необходимо решить следующие задачи:

1. Представить взаимодействие реализации IPsec с окружением в виде набора входных и выходных событий.
2. Определить набор внутренних событий, существенных для поведения реализации IPsec.
3. Определить сигнатуры элементов формального интерфейса, соответствующих выделенным множествам событий.

При определении элементов формального интерфейса необходимо учитывать семантику параллельного возбуждения событий, соответствующих элементам контрактной спецификации: оно должно быть эквивалентно некоторой последовательности возбуждения событий (семантика чередования или interleaving).

При составлении формального интерфейса необходимо учитывать указанное ограничение. Набор выбранных элементов формального интерфейса должен быть достаточен для построения адекватной модели поведения системы.

2.2.2. Множества входных и выходных событий для различных функций IPsec

Наиболее просто устроен формальный интерфейс для взаимодействий через синхронный программный интерфейс. Примером такого программного интерфейса служат функции настройки реализации IPsec через программный интерфейс реализации или расширения интерфейса сокетов. Так как

синхронные функции выполняются последовательно, то для синхронных функций формальный интерфейс, как правило, включает ровно по одному элементу для каждой интерфейсной функции.

Состав формального интерфейса для протоколов AH и ESP устроен следующим образом. Для каждого сообщения из алфавита сообщений в формальный интерфейс вводятся два элемента:

Прием сообщения протокола. В этом элементе формального интерфейса специфицируется обработка входящего сообщения.

Отправка сообщения. Здесь специфицируется проверка корректности исходящего сообщения.

Протокол IKEv2 расположен на прикладном уровне стека TCP/IP и использует протокол UDP для обмена сообщениями. Особенность IKEv2 заключается в том, что сообщения этого протокола обрабатываются на транспортном уровне протоколами IPsec. Несмотря на это формальный интерфейс для протокола IKE v2 устроен аналогичным образом – он включает два элемента: прием сообщения IKEv2 и отправка сообщения IKEv2.

Задание сигнатур элементов формального интерфейса

При задании сигнатур элементов формального интерфейса необходимо определить типы для представления входных и выходных значений элементов формального интерфейса. Из описания состава формальных интерфейсов следует, что элементы формального интерфейса делятся на две категории:

Элементы, представляющие программный интерфейс.

Элементы, представляющие обмены сообщениями.

Различия между этими элементами заключаются в том, как они определены в спецификации протокола. Функции программного интерфейса, как правило, описываются явным заданием сигнатуры на языке Си, с определением типов входных и выходных значений. Сообщения протоколов, напротив, заданы как размеченные наборы битов в неформальном табличном задании.

Соответственно, задачи определения типов для элементов формального интерфейса из этих категорий различаются.

2.2.3. Определение типов для элементов, представляющих функции программного интерфейса

Для задания сигнатур элементов формального интерфейса следует использовать типы, определённые в спецификации протокола, а не те типы, которые определены в заголовочных файлах реализации. Это требование к формальной спецификации призвано упростить развёртывание и сборку тестового набора, в состав которого войдёт формальная спецификация. Дело в том, что тестирование, как правило, выполняется удалённо. Для функций из программного интерфейса это означает, что спецификация будет компилироваться не на целевом узле, на котором функционирует тестируемая

реализация, а на инструментальном узле. Так как на инструментальном узле может быть установлена, вообще говоря, совсем другая операционная система, чем на целевом узле, то нельзя гарантировать, что система типов реализации будет присутствовать на инструментальном узле.

2.2.4. Определение типов для представления сообщений протоколов

В рамках разработанного метода разработка типов данных для параметров функций из программного интерфейса протокола следует общему методу разработки формальных спецификаций средствами инструмента CTesK [9]. В данном разделе рассмотрим важную особенность предлагаемого метода – представление сообщений протоколов в формальной спецификации.

Рассмотрим примерные сигнатуры спецификационных функций receive_PDU и sent_PDU, которые формализуют требования к обработке входящего сообщения протокола и требования к содержимому исходящего сообщения протокола:

```
specification void receive_PDU(PDU_in * pdu);
reaction PDU_out * sent_PDU(void);
```

Входное событие receive_PDU принимает входящее сообщение в модельном представлении в качестве единственного параметра. Возвращаемое значение отсутствует.

Выходное событие sent_PDU не имеет входных параметров, как и положено реакции в нотации SeC [9], а в качестве формального результата возвращает вышедшее сообщение в модельном представлении.

Для представления входящих и исходящих сообщений используются разные типы. Тип для исходящих сообщений снабжён инвариантом, в котором формализованы ограничения целостности из спецификации протокола.

Для входящих сообщений инварианты не задаются, так как реализация протокола Интернета обязана принимать на вход любую последовательность байтов, включая некорректно сформированные (нарушающие инвариант) и обрезанные сообщения.

В рамках данного метода модельное представление сообщений протоколов задаётся в нотации SeC по следующей схеме:

```
struct _PDU base {
        Object * errorDesc;
        MemBuf * actualPDU;
        Type1    field1;
        // ...
        TypeN    fieldN;
        List *   options;
```



```
        TypePayload *    payload;
};
```

Структура начинается с двух служебных полей: поля errorDesc и actualPDU. В поле errorDesc заносится информация о нарушениях формата, обнаруженных в фактическом сообщении, доставленном в тестируемую реализацию или отправленном реализацией. Если в сообщении нет нарушений формата, то значение errorDesc равно NULL. В поле actualPDU хранится фактическое сообщение, переданное по сети. Это значение используется для отладки тестов и построения подробных отчётов о тестировании.

Каждое поле заголовка сообщения, включая битовые поля, представлено в структуре отдельным полем.

Если в формате сообщения предусмотрены списки опций переменной длины, то в модели они представляются списком – объектом типа List.

Полезная нагрузка сообщения представлена полем payload.

Согласно правилам CTesK необходимо объявить спецификационные типы для использования модельных сообщений в спецификациях.

```
specification typedef struct _base_PDU PDU_in;
invariant typedef PDU_in PDU_out;
invariant (PDU_out * pdu) {
        if (pdu->errorDesc != NULL) return false;
        // проверка ограничений целостности
}
```

Тип PDU_in реализован как простой спецификационный тип на основе _PDU_base. Это позволяет использовать имена полей и описание нарушений формата в спецификациях. Тип PDU_out реализуется как подтип PDU_in, ограничение подтипа задаётся логическим условием в инварианте типа. Инвариант проверяется автоматически до и после события, соответствующего элементу формального интерфейса. Инварианты для исходящих сообщений содержат больше ограничений, так как реализации IPsec могут принимать на вход любые сообщения, а отправлять должны только корректные.

2.2.5. Разработка пред- и постусловий для формального интерфейса протоколов безопасности

Построение формальной спецификации требований сродни программированию. В обоих случаях неформальный текст на естественном языке преобразуется в текст на строго определённом формальном языке.

В последнее время методология программирования сделала значительный шаг вперёд благодаря выделению приёмов, или паттернов, проектирования программ. Методология формализации пока разработана не столь глубоко,

поэтому далее в тексте мы представим приёмы формализации только для отдельных видов требований.

2.2.6. Разработка модельного состояния

Пред- и постусловия описывают ограничения на параметры стимулов и ограничения на наблюдаемое поведение в терминах изменений модельного состояния. Для того чтобы разработать общую спецификацию, не зависящую от конкретной реализации, необходимо, чтобы модельное состояние состояло из концептуальных объектов, которые имеют смысл для всех реализаций.

Естественным образом такие структуры данных строятся на основании концептуальных моделей протокола. В протоколах безопасности Интернета концептуальные структуры данных представлены преимущественно в двух документах: спецификации архитектуры IPsec [1] и спецификации информационной базы (MIB) политик безопасности [6].

Спецификация MIB представляет собой текст на диалекте языка ASN.1. Набор основных типов и информационных объектов определён в RFC 1213 [14] и RFC 2011 [15]. Преимущества использования MIB при разработке модельного состояния заключается в том, что в MIB описаны структуры данных, не зависящие от конкретных реализаций протокола. Фактически, MIB является формальным заданием наиболее детально разработанных концептуальных структур данных.

2.2.7. Формализация семантических связей в сообщениях протоколов

Спецификации протоколов определяют синтаксис сообщений, задавая побитное разделение сообщения на поля. Для каждого поля описывается семантика и, если есть, связь между полями.

Семантические связи между полями записываются в инварианте типа, представляющего исходящие сообщения. Для входящих сообщений инварианты типа не записываются, так как для протоколов Интернета допускается получение сообщений с нарушениями требований к формату сообщений.

Для протоколов IPsec прописываются правила по обработке нарушений формата и ограничений целостности в сообщениях. Соответственно, в постусловиях событий, соответствующих обработке входящих сообщений, добавляется проверка нарушений формата сообщений и специфицируется предписанная реакция на нарушения.

2.2.8. Формализация негативных требований

Как мы уже указывали, негативные требования запрещают реализации демонстрировать определённое поведение. Негативные требования формализуются в постусловии реакции.

Предположим, что запрещаемое поведение представляется некоторым логическим выражением expr, тогда в постусловии запрещение поведения выражается следующей инструкцией:

if (expr) { **return false**; }

Другими словами, если реализация демонстрирует запрещённое поведение, то постусловие выходного события возвращает негативный вердикт.

2.2.9. Формализация требований к ожидаемым реакциям

В большинстве формальных языков, поддерживающих контрактные спецификации, включая используемое в работе расширение языка Си, нет встроенной поддержки темпоральных спецификаций, то есть спецификаций свойств цепочки событий, упорядоченных во времени. Поэтому в этих формализмах требования к ожидаемой реакции формулируются в постусловии спецификационной функции, соответствующей реакции. Постусловие выносит вердикт на основании истории взаимодействий, которая хранится в модельном состоянии.

Рассмотрим виды выходных событий, которые возникают в протоколах безопасности:

Реакции на стимулы. Это реакции, которые, согласно спецификации протокола, должны быть выданы в ответ на определённое входное событие.

Реакции на внутренние события. К этой группе относятся внешне наблюдаемые действия, связанные с ненаблюдаемыми извне событиями. Наиболее распространённый вид внутренних событий – истечение таймеров.

2.2.10. Формализация требований к обязательности реакций

Предложенные выше методы позволяют проверять корректность наблюдаемых реакций. Тем не менее, этих приёмов недостаточно для решения следующей задачи: проверить, что реализация действительно демонстрирует некоторую реакцию.

В данной работе для решения этой задачи используется следующая методика. Предполагается, что тестируемая система после приёма любого стимула за некоторое ограниченное время переходит в стационарное состояние, в котором тестируемая система ожидает прибытие стимула, и выдача реакций без предшествующего им стимула невозможна. После выдачи тестового воздействия тестовая система собирает в течение некоторого времени реакции. Наблюдаемое поведение тестируемой системы считается корректным, если корректны все реакции и конечное состояние является стационарным. В противном случае поведение реализации считается некорректным.

В формальную спецификацию вводится функция проверки стационарности состояния, которая возвращает определённые значения в зависимости от того, является ли текущее состояние спецификации стационарным или нет.

Требования к обязательности реакций формализуются в функции проверки стационарности. Функция определяет состояние как нестационарное, когда в истории взаимодействия есть стимулы, для которых не наблюдались обязательные реакции, или активные таймеры. Благодаря этому тестовая система вынесет вердикт о некорректности поведения реализации, если реализация не выдала обязательную реакцию.

2.2.11. Формализация требований к реакции на входное событие

Рассмотрим, как представляются требования к реакции на входное событие в том случае, когда на него выдаётся не более одной реакции.

Информация о полученных стимулах сохраняется в состоянии формальной спецификации. Модельное состояние может хранить историю стимулов различными способами: в виде списка объектов, набора (мультимножество, multiset), отображения, множества.

Требования, устанавливающие связь между стимулом и реакцией, формализуются в функциях со следующей сигнатурой:
bool matchReaction(Object * s, ReactionType * r);

Первый аргумент соответствует входному событию, второй – реакции. Функция возвращает **true**, если реакция соответствует стимулу. Для каждой реакции необходимо разработать такую функцию.

В постусловие реакции добавляется проверка того, что в истории принятых стимулов есть стимул, для которого вызов stimulusMatchReaction, возвращает **true** для данной реакции.

```
reaction PDU_out * sent_Message() {
 post {
  // ...
  Predicate * p=new_Predicate(matchReaction, sent_Message)
  if (!exists_Iterator(@history_iter, p)) {
   return false;
  }
  // ...
 }
}
```

Если на один стимул должно быть выдано несколько реакций, то в историю добавляется не только стимул, но и счётчик ожидаемых реакций. С каждой реакцией, соответствующей стимулу, значение счётчика уменьшается на 1. В функцию проверки стационарности добавляется проверка того, что если есть

стимул с ненулевым значением счётчика, то состояние является нестационарным.

2.2.12. Формализация требований к реакциям на истечение таймеров

В модельном состоянии есть переменные, которые представляют состояние реализации протокола на концептуальном уровне. В частности, модельное состояние должно быть спроектировано таким образом, чтобы содержать информацию об активных таймерах.

В постусловии для реакции на истечение таймера проверяется, что соответствующий таймер был активен до возбуждения реакции. Если это не верно, то постусловие выносит негативный вердикт.

Для верификации демонстрируемых реакций необходимо, чтобы информация о стимулах и текущем состоянии сохранялась в модельном состоянии. Задачу изменения модельного состояния решают функции реконструкции состояния, поэтому модификация истории взаимодействия реализуется в них.

2.2.13. Формализация требований к необязательным функциям

Под необязательными функциями мы понимаем функции протокола, которые спецификация разрешает не включать в реализации.

В рамках данного метода используется следующий приём для формализации требований к таким функциям.

Для необязательных функций в формальную спецификацию включаются параметры, которые определяют включение требования в состав спецификации. В зависимости от значения параметра логические выражения, формализующие требование, игнорируются или вычисляются при использовании формальной спецификации.

Параметры представлены в формальной спецификации как глобальные переменные. Пользователь тестового набора задаёт значения параметров в конфигурационном файле. При запуске теста автоматически производится разбор конфигурационного файла, и присваиваются значения соответствующим глобальным переменным.

Совокупность фактических значений параметров спецификации служат формальной записью *описания соответствия реализации* (Implementation Conformance Statement). Значения параметров устанавливаются экспертами на основе текстового описания соответствия реализации, исследования исходных текстов реализации или изучения поведения реализации.

2.2.14. Формализация недоопределённых функций

Под недоопределёнными функциями мы понимаем функции протокола, для которых не все ветви поведения определены в спецификации. Такие недоопределённости вскрываются при анализе концептуальных моделей поведения.

Возможны две стратегии формализации недоопределённых функций, которые условно можно обозначить как «запрещающая» и «разрешающая». В запрещающей стратегии спецификация явно запрещает подачу стимулов, которые соответствуют неопределённым ветвям функциональности, и явно запрещает использование неопределённых элементов в реакциях. А именно:

В предусловиях входных событий задаются граничные условия, которые сужают область определения постусловия только теми событиями, обработка которых полностью определена в исходных текстовых спецификациях протокола.

Для выходных событий используется следующая интерпретация недоопределённого формата: реализации должны порождать только хорошо определённые реакции. Если наблюдаемая реакция содержит недоопределённые элементы, то постусловие события выносит негативный вердикт.

При следовании «разрешающей» стратегии разработчик спецификации вносит в спецификацию параметры, значения которых соответствуют различным возможным способам разрешить неопределённость. Значения параметров устанавливаются по результатам анализа документации реализации, исходных текстов или проведения экспериментов. Соответственно, в постусловии элементов формального интерфейса вердикт выносится на основании того, соответствует ли наблюдаемое поведение реализации протокола заданному набору параметров, описывающих, как разрешена неопределённость в тестируемой реализации.

3. Формальная спецификация IPsec v2

В соответствии с приведенной выше моделью тестирования формальная спецификация IPsec v2 состоит из нескольких компонентов:

- модельного состояния, которое содержит набор структур данных, моделирующих концептуальные структуры данных из стандартов IPsec v2, таких как база данных контекстов безопасности и база данных политик безопасности;
- формального интерфейса IPsec, включающего спецификационные стимулы, формализующие требования к изменению состояния реализации IPsec v2 при внешнем воздействии на систему, и спецификационные реакции, которые

формализуют требования к реакциям реализации IPsec v2 на внешние воздействия;

- критериев покрытия, идентифицирующих различные ветви функциональности IPsec.

3.1. Модельное состояние

Модельное состояние реализации IPsec представлено тремя структурами данных: множеством контекстов безопасности (Security Association Database - SAD), упорядоченным множеством политик безопасности (Security Policy Database – SPD), и упорядоченной базой данных авторизации партнеров (Peer Authorization Database - PAD). Полное описание этих структур приведено в RFC 4301 и RFC 4807.

Рассмотрим, как эти концептуальные структуры данных представлены в формальной спецификации.

Модель контекста безопасности представлена абстрактным типом **SA** со следующими полями:

selectors селекторы трафика, включающие удаленный IP-адрес, локальный IP-адрес, протокол следующего уровня, удаленный порт, локальный порт. Каждый селектор в соответствии со спецификацией может определять несколько значений. Для протоколов, не имеющих портов, селекторы портов могут задавать другие параметры, присущие этим протоколам (для ICMP - тип и код сообщения; для Mobile IP - тип сообщения). Подробное описание селекторов трафика приведено в RFC 4301;

spi индекс параметров безопасности, используется для поиска контекста безопасности для обработки входящего трафика;

proto идентификатор протокола IPsec, используемого в данном контексте безопасности - AH или ESP;

mode режим протокола IPsec: туннельный или транспортный;

direction Направление трафика, для которого применяется данный контекст безопасности: входящий или исходящий;

sequence_number счетчик порядковых номеров, используется при формировании исходящих пакетов;

sequence_window окно порядковых номеров полученных пакетов, используется для противодействия атакам повторного воспроизведения;

lifetime время жизни данного контекста безопасности. Оно может быть выражено временем или количеством байтов, или одновременным использованием обоих типов. Согласно

спецификации преимуществом обладает первое исчерпанное время жизни;

tunnel_data IP-адреса источника и места назначения туннельного заголовка;

Algorithm_Data идентификаторы криптографических алгоритмов, используемых в данном контексте безопасности, их параметры (ключи, размеры блоков и т.п.) Содержание этих структур зависит от используемого протокола IPsec.

Модель политики безопасности представлена абстрактным типом **SP**:

selectors селекторы трафика, определяют применимость данной политики к входящим или исходящим пакетам. Это поле аналогично полю selectors структуры SA за одним отличием – селекторы для политики безопасности могут включать несколько наборов селекторов;

name дополнительный селектор, задающий список идентификаторов. В RFC 4301 определены четыре типа идентификаторов: полное имя пользователя электронной почты, полное имя DNS, различительное имя X.500, строка байтов; данный селектор используется в том случае, когда возможно установить пользователя, с чьими привилегиями исполняется процесс, порождающий защищаемые данные;

pfp флаги PFP – по одному на каждый селектор трафика. При динамическом создании контекста безопасности для данной политики каждый флаг для соответствующего селектора трафика указывает, нужно ли брать значение селектора из соответствующего поля в пакете, который инициировал создание контекста безопасности, или из значения в соответствующей политике. Заданный флаг применяется к соответствующему селектору во всех наборах селекторов, содержащихся в данной политике.

action Действие, которое требуется применить к пакету: защитить, пропустить без защиты или отбросить. Одно действие выполняется для всех наборов селекторов.

direction Направление потоков данных, для которого предназначена данная политика: входящий или исходящий

SA_spec Если в качестве требуемой обработки указывается защита, то этот элемент содержит данные, в основном повторяющие структуру контекста безопасности SA. В элементе, определяющем алгоритмы, задаются только их идентификаторы (без ключей и т.п.). При этом могут задаваться несколько однотипных алгоритмов (например для шифрования), упорядоченные по предпочтительности

использования. При согласовании с помощью протокола IKE нового контекста безопасности удаленный узел выберет из этого набора приемлемый для него алгоритм.

Модель контекста безопасности IKE представлена абстрактным типом **IKE_SA** со следующими полями:

spi индекс параметров безопасности, используется для поиска контекста безопасности IKE;

rID идентификатор партнера;

i_messageID следующий идентификатор сообщения, который должен использоваться для инициируемого данным узлом запроса;

r_messageID идентификатор сообщения, который предполагается увидеть в следующем запросе от партнера;

request_window размер окна партнера для перекрывающихся запросов;

child_ipsec_sa список ссылок на дочерние контексты безопасности IPsec;

request_list список отправленных запросов, на которые еще не получен ответ, используется для повторной передачи, если в течение заданного времени ответ не получен;

response_list список отправленных ответов, используется для повторной передачи в случае потери пакетов;

dhg группа Диффи-Хеллмана;

auth алгоритм защиты целостности;

enc алгоритм шифрования;

prf псевдослучайная функция;

keys общие ключи контекста безопасности IKE. Правила вычисления этих ключей приведены в RFC 4306;

NATDetection блок данных для пересечения NAT.

Помимо моделей данных, необходимых для описания поведения протокола IKEv2, в спецификацию включены модели состояний служебных и транспортных протоколов: UDP и ICMPv6.

Модель блока данных авторизации партнера представлена абстрактным типом **PA** со следующими полями:

selectors идентификаторы партнера, которые согласуются с типами символических имен и IP-адресами, используемыми для идентификации элементов SPD;

auth_protocol протокол для аутентификации каждого партнера;

auth_method метод для аутентификации каждого партнера;

auth_data аутентификационные данные для каждого партнера;

peer_gateway информация о местоположении шлюза партнера (используется для партнеров, о которых известно, что они находятся "за" защитным шлюзом).

Для работы с модельными структурами данных SAD, SPD, PAD реализован набор вспомогательных функций: добавление, удаление, поиск записей и другие.

3.2. Модель пакета

Для модельного представления пакетов использовалась разработанная библиотека полиморфных типов. Иерархия типов отражает устройство пакетов IPv4/IPv6, в которых заголовки протоколов разного уровня организуются в односвязный список. Полиморфизм типов позволяет для моделирования различных пакетов использовать единую структуру. Разработаны типы для преставления кадров Ethernet, пакетов IPv4 и IPv6, дополнительных заголовков IPv6, заголовков IPsec v2, сообщений ICMPv6, пакетов UDP и IKEv2.

3.3. Формальный интерфейс управления состоянием IPsec

Добавление и удаление записей в базе контекстов безопасности представлены тремя элементами формального интерфейса: append_SAD_spec для добавления новых записей, remove_SA_SAD_spec для удаления контекстов безопасности, и flush_spec для очистки базы. Спецификационные функции append_SPD_spec и remove_SP_SPD_spec предназначены для добавления и удаления политик безопасности SP. Функция flush_spec моделирует сброс баз данных SAD и SPD.

Для данных спецификационных функций разработаны медиаторные функции, которые реализуют соответствующие воздействия для конкретной целевой системы, и осуществляют трансляцию содержимого SAD и SPD из реализационного вида в модельный. Медиаторные функции существенно зависят от реализации IPsec, поскольку реализация может поддерживать не все элементы определенные для SA и SP. В пост-условии этих функций проверяется соответствие полученного после воздействия состояния SAD, SPD с их ожидаемым состоянием.

3.4. Формальный интерфейс протоколов AH и ESP

Спецификационная функция receive_IpsecPacket моделирует обработку входящих пакетов в соответствии с требованиями, определенными в RFC 4301.

Краткое описание спецификации. Если пакет содержит IPsec заголовок, то ищется соответствующий контекст безопасности SA, сравниваются алгоритмы и ключи пакета и SA, сравниваются селекторы (заметим, что здесь нет проверки целостности пакета или расшифровки шифротекста). После прохождения всех проверок из пакета удаляется обработанный IPsec

заголовок. Если пакет не содержит IPsec заголовков, то ищется соответствующая политика безопасности SP, сравниваются селекторы. На данном этапе принимается решение пропустить пакет или отбросить. Если выносится решение его пропустить, то выбирается сокет, который может получить этот пакет (в соответствии с адресом и портом), и пакет помещается к нему в очередь. Если пакет на каком-то этапе отвергается, или нет подходящего для его приема сокета, то он не помещается ни в какую очередь. Таким образом предполагается, что если пакет должен быть принят системой, он будет сохранен в очереди какого-либо сокета, иначе - просто отброшен.

Входящий пакет представлен в модельном виде как параметр спецификационной функции.

Спецификационная функция send_IpsecPacket моделирует обработку исходящих данных в IPsec v2 в соответствии с требованиями, определенными в RFC 4301.

Краткое описание спецификации. Исходящее сообщение считается корректным, если существует политика безопасности и контекст безопасности такие, что в результате обработки поданного стимула, такого, как отправка сообщения по протоколу UDP, получится сообщение, соответствующее отправленному. Данная спецификация является недетерминированной, так как точное значение шифротекста предсказать нельзя в силу включения в защищаемые данные случайного вектора.

Построение модельного исходящего пакета выполняется следующим образом. Выбирается политика безопасности, соответствующая данному пакету. На данном этапе принимается решение пропустить пакет, защитить с помощью IPsec или отбросить. Если требуется защитить пакет с помощью IPsec, то выбирается нужный контекст безопасности, в случае необходимости используется спецификация IKE v2. Если контекст безопасности не найден, фиксируется ошибка. Согласно найденному контексту безопасности в пакет добавляется модельный IPsec заголовок, в котором вместо аутентификационных данных и криптотекста присутствуют идентификаторы алгоритмов и ключи. Далее по адресу и порту из пакета выбирается сокет, который может отправить этот пакет, и пакет помещается к нему в очередь. Если пакет на каком-то этапе отвергается, или нет подходящего для его отправления сокета, то он не помещается ни в какую очередь. Таким образом предполагается, что если пакет должен быть отправлен системой, он будет сохранен в очереди какого-либо сокета, иначе - просто отброшен. В пост-условии проверяется, что если пакет отправлен, то он добавлен только в одну очередь, если пакет отброшен, то размер всех очередей не изменился.

Отправленный пакет IPsec представлен как возвращаемое значение спецификационной функции.

3.5. Формальный интерфейс протокола IKE v2

Формальный интерфейс протокола IKE v2 включает два элемента: receive_IkeMessage и send_IkeMessage. Первая спецификационная функция моделирует отправку сообщения протокола IKEv2 удаленному агенту, а вторая функция моделирует получение реализацией сообщения этого протокола.

Не смотря на то, что протокол IKE v2 использует UDP для доставки сообщений, спецификация UDP не входит в формальный интерфейс. Подуровень UDP реализуется внутренними функциями реконструкции состояния формального интерфейса IPsec.

Краткое описание спецификации. Постусловия формального интерфейса формализует требования RFC 4306 к обменам сообщениями протокола IKEv2:

IKE_SA_INIT – согласование параметров криптографических протоколов и обмен данными для генерации временных ключей по схеме Диффи-Хеллмана. Поддерживается ограниченное число алгоритмов: цифровая подпись по алгоритмам MD5 и SHA, шифрование алгоритмом 3DES.

IKE_AUTH – взаимная аутентификация и создание управляющего контекста безопасности, защищающего последующие обмены сообщениями IKE. В тестовом наборе поддерживается только один вид аутентификации – по общему секрету.

CHILD_SA – обмен сообщениями для создания дочернего контекста безопасности, обеспечивающего защиту транспортных протоколов на сетевом уровне.

Сообщения IKE в формальном интерфейсе представлены в открытом виде. Шифрование и цифровая подпись обменов реализуется медиаторами при отправке сообщения.

Структуры данных для моделирования сообщений IKE v2 основываются на форматах сообщений из RFC 4306.

4. Тестирование реализаций IPsec

4.1. Устройство тестового стенда

В состав тестового стенда входят инструментальный узел и целевой узел. На инструментальном узле исполняется основной поток управления тестовой системы. На целевом узле функционирует тестируемая реализация. Для целей тестирования на целевой узел устанавливаются тестовые агенты, которые предоставляют средства удалённого доступа к служебным функциям и протоколам верхнего уровня (UDP). Инструментальный узел и целевой узел находятся в одном сегменте локальной сети. На рис. 1 представлено распределение компонентов тестовой системы по тестовому стенду.

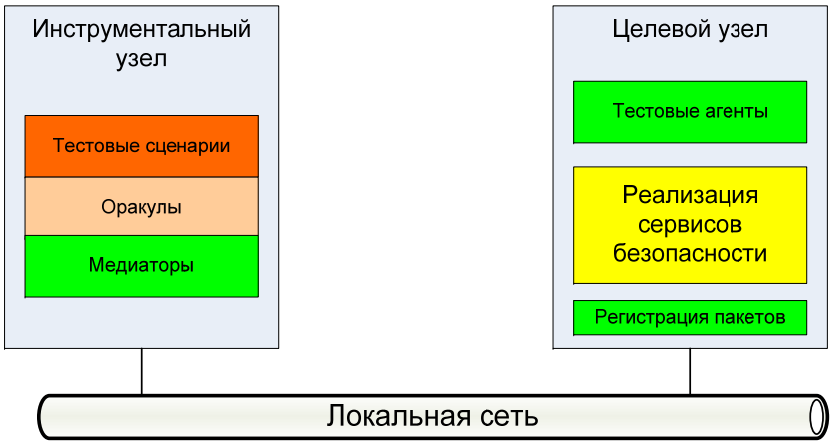

Рис. 1. Распределение компонентов тестовой системы в тестовом стенде.

В состав тестового набора входят следующие агенты:

- Агент для доступа к состоянию IPsec.
- Агент для доступа к состоянию IPv6.
- Агент для регистрации кадров канального уровня.
- Агент для управления сокетами UDP.
- Агент для управления реализацией протокола IKEv2.

Каждый агент предоставляет программный интерфейс для локальных вызовов. Для каждого агента разработан интерфейс удалённых вызовов посредством ONC RPC [16,17].

Агент для доступа к состоянию IPsec. Предоставляет тестовой системе средства для чтения и изменения настроек IPsec, прежде всего, БД политик безопасности и БД контекстов безопасности. Изменения заключаются в поддержке новых структур данных для представления контекстов безопасности и политик безопасности из IPsec v2.

Агент реализован для двух семейств операционных систем: FreeBSD 4.x и 5.x и OpenBSD. Реализации существенно зависят от платформы, так как нет стандартного программного интерфейса для операций с политиками безопасности и контекстами безопасности.

Реализации агента позволяют читать, добавлять и удалять политики безопасности, создавать, добавлять и удалять контексты безопасности.

Тестовый агент спроектирован таким образом, чтобы программный интерфейс агента не зависел от особенностей реализации. Такое решение обеспечивает переносимость удалённых вызовов агента на разных платформах.

Агент для доступа к состоянию IPv6. Предоставляет тестовой системе средства для чтения и изменения настроек протокола IPv6 на узле сети. На данный момент агент предоставляет следующие возможности:

- Получить список сетевых интерфейсов, установленных на узле,

Для любого сетевого интерфейса:

- Читать, добавлять, удалять адреса IPv6,
- Читать групповые (multicast) адреса IPv6,
- Читать адреса нижнего уровня (MAC адреса)
- Агент реализован для операционных систем FreeBSD 4.x и 5.x.

Реализация агента существенно зависит от платформы, так как нет стандартного программного интерфейса для операций с настройками IPv6.

Тестовый агент спроектирован таким образом, чтобы программный интерфейс агента не зависел от особенностей реализации. Такое решение обеспечивает переносимость удалённых вызовов агента на разных платформах.

Агент для регистрации кадров канального уровня. Обеспечивает регистрацию исходящих сообщений некоторого сетевого узла. Тестовая система использует агент для регистрации исходящих сообщений целевой реализации.

Агент реализован для операционных систем FreeBSD 4.x и 5.x, Linux с ядром версии 2.2 или 2.4. Существующая реализация агента использует фильтр пакетов BPF (Berkley Packet Filter). Реализации BPF есть на многих операционных системах, включая Microsoft Windows, поэтому предполагается, что агент может переноситься с минимальными изменениями на различные целевые платформы.

Также разработана реализация агента, использующая программный интерфейс PCAP.

Агент для управления сокетами UDP. Предоставляет тестовой системе средства для регистрации входящих сообщений UDP. Кроме того, интерфейс удалённого вызова агента предоставляет следующие средства:

- Удалённое создание, привязывание (bind) и удаление сокетов UDP.
- Удалённую отправка сообщений через сокет UDP.
- Удалённое чтение и изменение опций сокетов.

Агент реализован на операционных системах FreeBSD 4.x и 5.x, OpenBSD 3.x. Агент реализован в рамках POSIX, поэтому должен быть переносим на другие

операционные системы, поддерживающие программный интерфейс сокетов и потоков POSIX.

Агент для управления реализацией протокола IKEv2. Позволяет изменять настройки модуля IKEv2 и перегружать его.

Для корректного выполнения тестов необходимо, чтобы сетевое окружение, в котором выполняется тестирование, удовлетворяло ряду требований. В частности предполагается, что пакеты не теряются в сети (если пакет отправлен, он обязательно дойдет до адресата). Кроме того, предполагается, что пакеты не теряются внутри реализации. Для входящих пакетов это означает, что если пакет получен и прошел IPsec обработку, он будет доставлен в UDP сокет. Для исходящих пакетов: если политика безопасности разрешает такой трафик, пакет будет сформирован (возможно защищен IPsec) и отправлен в сеть.

Для обеспечения первого предположения в рамках данной работы был создан виртуальный тестовый стенд, так как исполнение тестов на виртуальных машинах имеет ряд преимуществ по сравнению с запуском тестов в реальном физическом окружении:

- Полный контроль над составом узлов в локальных сетях тестового стенда.
- Возможность создания идентичных копий тестовых стендов для проведения испытаний тестового набора разными участниками проекта.
- Возможность гибкой конфигурации состава тестового стенда путем добавления или удаления виртуальных машин.
- Экономия пространства, в частности, все средства ввода-вывода располагаются на одном физическом устройстве.

Виртуальный тестовый стенд включает в себя несколько виртуальных машин, объединенных в виртуальную локальную сеть.

Благодаря использованию виртуальных локальных сетей и виртуальных машин в тестовом стенде обеспечивается полный контроль над потоками данных. Все информационные потоки симулируются тестовой системой. В качестве виртуальной среды использовался продукт VMware Workstation 5.

4.2. Тестирование обработки исходящего трафика

В тестовом сценарии формируется модельное представление поступившего пакета, который в соответствии с ожидаемым поведением целевой системы, сохраняется (или не сохраняется) в очереди соответствующего модельного сокета. Модельный пакет преобразуется в сетевой пакет и отправляется тестируемой системе. Инструментальный узел запрашивает у тестового агента собранные пакеты, проверяет, есть ли они в модельных сокетах, и выносит

вердикт на основании логических предикатов, содержащихся в постусловиях спецификационных функций.

Первоначальный UDP-пакет с помощью дополнительного агента передается на целевой узел и отправляется через заданный сокет. При этом в сеть выходит пакет, уже прошедший IPsec обработку на целевой системе.

На целевом узле создаются UDP сокеты, привязанные к конкретным адресам имеющихся интерфейсов.

В тестовом сценарии формируется модельное представление исходящего UDP-пакета, который в соответствии с политиками безопасности проходит IPsec обработку, а затем или отбрасывается, или сохраняется в очереди соответствующего модельного сокета. Модельный UDP-пакет преобразуется в промежуточный формат и передается тестовому агенту, который через указанный сокет отправляет пакеты в сеть. При этом перед отправкой в сеть пакеты проходят IPsec обработку на целевом узле. На инструментальном узле кетчер собирает все пришедшие с целевого узла пакеты, переводит их в модельное представление с сохранением всех заголовков. После этого проверяется, есть ли такой пакет в очереди модельного сокета, и выносится вердикт.

4.3. Тестирование обработки входящего трафика

В сценарии формируется модельный пакет, который будет рассматриваться как входящий. IPsec заголовки этого пакета, если они присутствуют, содержат всю информацию, необходимую для обработки пакета (SPI, Sequence Number и т.д.), но не содержат криптографические данные (аутентификационные последовательности и шифротекст), вместо них включаются идентификаторы криптографических алгоритмов и ключи.

Затем с помощью генератора сетевых пакетов из модельного пакета создается сетевой пакет и он отправляется в сеть.

Открытые на тестируемом узле сокеты являются UDP-сокетами. Поэтому пакет, полученный таким сокетом, уже прошел IPsec обработку, т.е. для него либо был найден контекст безопасности, проверена целостность пакета, выполнена расшифровка пакета и другие проверки, либо была найдена политика безопасности с разрешением его пропустить. Таким образом вся IPsec обработка сетевого пакета перекладывается на тестируемую систему. Если какой-то сокет получил пакет, значит он был парвильно сформирован и есть соответствующая разрешающая политика, если ни один сокет неполучил пакет, значит он был отвергнут. Задача формирования сетевого пакета решается генератором сетевых пакетов, который по модельному представлению создает необходимые заголовки, а по заданным алгоритмам вычисляет аутентификационные последовательности и создает шифротекст.

Сборщик реакций тестовой системы периодически опрашивает тестового агента, и забирает все пакеты, полученные на данный момент сокетами.

Каждый полученный тестовым агентом пакет регистрируется как реакция целевой системы.

В ходе вынесения вердикта постусловие спецификационной функции реакции IpsecRecvInput вызывается для каждого такого пакета. По IP адресу и порту назначения пакета производится поиск соответствующего сокета. Если сокет не найден, фиксируется ошибка. Если сокет найден, данный пакет удаляется из его очереди. Если у сокета нет такого пакета, фиксируется ошибка.

В пост-условии проверяется, что пакет был удален из очереди только одного сокета, очереди других сокетов не изменились. После завершения всех тестовых воздействий, очереди всех модельных сокетов должны быть пусты.

4.4. Тестирование IKEv2

Стимулами в разработанном тестовом наборе являются сообщения от инструментального узла или сообщения, отправленные через UDP сокет целевого узла, с помощью агента управления сокетами, а реакциями – сообщения со стороны тестируемого узла. Основная часть требований спецификации IKEv2 проверяется в постусловиях реакций.

Разработанный тестовый набор покрывает большую часть требований спецификации [4], в том числе обеспечивает проверку правильности начальных обменов, обменов CREATE_CHILD_SA, информационных обменов, а также требований к удалению контекстов безопасности, размеру сообщений, использованию порядковых номеров, размеру окна для перекрывающихся запросов, номерам версий и совместимости, идентифицирующим цепочкам и общим требованиям к передаче сообщений IKE v2.

В тестовом стенде задействованы два узла – инструментальный узел, на котором исполняются тестовые сценарии, и целевой узел – узел, на котором работает тестируемая реализация IKE v2.

На целевом узле размещаются два тестовых агента. Один - по запросу тестовой системы отсылает сообщения через открытые UDP сокеты, привязанные к конкретным адресам имеющихся интерфейсов. Другой - регистрирует исходящие сообщения, и по запросу передает инструментальному узлу.

В начале каждого теста устанавливаются необходимые политики безопасности (в том числе разрешающие IKE трафик), и после каждого воздействия проверяется, что они не изменились.

В начале каждого теста агентом управления подсистемой IKE загружаются необходимые для данного теста настройки IKE.

4.4.1. Тестирование в режиме ответчика

В режиме ответчика реализация не генерирует запросы, а лишь поддерживает информационный обмен, инициированный другим узлом. Стимулами являются сообщения от инструментального узла.

В тестовом сценарии формируется запрос в модельном представлении, который передается спецификационной функции отправки пакетов. Если сообщение является запросом IKE_SA_INIT, создается новый контекст безопасности IKE SA. В предусловии этой функции проверяется правильность структуры тестового сообщения и его своевременность, и на основании этого делается вывод о том, должен ли на него быть ответ, или реализация должна его просто отбросить. Сообщение сохраняется в очереди запросов. В блоке медиатора из модельного представления тестового сообщения строится реализационное, которое и отправляется в сеть.

Сборщик реакций периодически опрашивает тестового агента, и забирает все пакеты, отправленные на данный момент целевой системой. Время ожидания реакции целевой системы должно устанавливаться для каждого сценария отдельно, так как некоторые ответные сообщения (например, запрос COOKIE) требуют реакции инструментального узла в течение ограниченного времени. Каждый полученный тестовым агентом пакет регистрируется как реакция целевой системы. Сборщик реакций строит из реализационного пакета его модельное представление.

В постусловии реакции сообщение проверяется на соответствие требованиям спецификации. Проверка разделена на несколько стадий. Сначала проверяется допустимость такого сообщения от реализации для текущего обмена, соответствие существующему IKE SA, соответствие ранее отправленному запросу. Далее проверяется структура самого сообщения (присутствующие блоки данных должны соответствовать данному обмену). Наконец разбирается структура каждого блока данных и проверяются значения всех его полей. После завершения проверки, соответствующий этому сообщению запрос удаляется из очереди запросов.

После проверки всех требований, сообщение передается тестовому сценарию, где в зависимости от плана сценария, принимается решение о продолжении или завершении информационного обмена. Если выявлено нарушение требований, то принимается решение, является ли это нарушение критичным для продолжения обмена. В случае продолжения обмена, формируется следующий запрос.

При отсутствии реакции реализации в течение установленного времени обмен считается завершенным.

После завершения всех воздействий, проверяется очередь запросов. Она должна быть пуста или содержать запросы, на которые не ожидался ответ.

4.4.2. Тестирование в режиме инициатора

В режиме инициатора реализация сама формирует запросы. Стимулами являются уже не сообщения от другого узла, а UDP пакеты, отправляемые агентом на тестируемом узле, через локальные UDP-сокеты. Эти пакеты перехватываются подсистемой обработки IPsec, которая передает реализации IKE запрос на создание контекстов безопасности IPsec.

В тестовом сценарии формируется модельное представление UDP пакета, который передается агенту на целевой узел. В очередь ожидаемых запросов добавляется тип запроса. Тестовый агент на целевом узле отправляет этот пакет через открытый сокет. При наличии политики безопасности, требующей IPsec обработки такого пакета, инициируется IKE обмен.

Сборщик реакций тестовой системы периодически опрашивает другого тестового агента, и забирает все пакеты, отправленные на данный момент целевой системой. Для данного режима тестирования следует учитывать время ожидания и сбора реакций, поскольку согласно спецификации, если на запрос не получен ответ в течение некоторого времени, реализация IKE должна повторно передать запрос или отказаться от соединения. Каждый полученный тестовым агентом пакет регистрируется как реакция целевой системы.

В постусловии реакции сообщение проверяется на соответствие требованиям спецификации. Проверка разделена на несколько стадий. Сначала проверяется допустимость такого сообщения от реализации для текущего обмена, соответствие существующему IKE SA (если сообщение не является запросом IKE_SA_INIT). Далее проверяется структура самого сообщения (присутствующие блоки данных должны соответствовать данному обмену). Наконец разбирается структура каждого блока данных и проверяются значения всех его полей.

Из очереди ожидаемых запросов удаляется соответствующий элемент.

После проверки всех требований, сообщение передается тестовому сценарию, где в зависимости от плана сценария, принимается решение о необходимости ответа, задержки ответа на некоторое время или завершении информационного обмена. Если выявлено нарушение требований, то принимается решение, является ли это нарушение критичным для продолжения обмена.

Если сообщение является запросом IKE_SA_INIT, создается новый контекст безопасности IKE SA.

Если принято решение отправить ответ, то формируется модельное представление сообщения. Сообщение передается спецификационной функции отправки пакетов. В предусловии этой функции проверяется правильность структуры тестового сообщения и его допустимость в текущем обмене. В очередь ожидаемых запросов добавляется тип запроса, если он

предполагается. В блоке медиатора из модельного представления тестового сообщения строится реализационное, которое отправляется в сеть.

При отсутствии реакции реализации в течение установленного времени обмен считается завершенным. После завершения всех воздействий очередь ожидаемых запросов должна быть пуста.

5. Результаты тестирования реализаций IPsec v2

5.1. Тестирование реализаций оконечных узлов и шлюзов безопасности

Тестирование реализаций оконечных узлов и шлюзов безопасности проводилось для операционной системы FreBSD 6.2. Эта реализация была выбрана по следующим причинам:

- Наличие стабильной поддержки IPsec v2. Для сравнения, на момент тестирования (2008-09 гг.) реализации IPsec для ОС Linux, Solaris не поддерживали вторую версию архитектуры IPsec. Реализация IPsec для остальных ОС семейства BSD имеют много общего с реализацией для FreeBSD, поэтому из всего семейства была выбрана наиболее разработанная и поддерживаемая ОС.

- Наличие исходных текстов утилит администрированием IPsec. Несмотря на отсутствие документации по программным интерфейсам реализации IPsec удалось реализовать удаленных агентов на основе штатных утилит управления IPsec в операционной системе. На проприетарных операционных системах, таких как MS Windows и Cisco IOS, такой возможности нет.

В ходе тестирования был обнаружен ряд отклонений от требований IPsec:

1. Неполная поддержка селекторов IPsec. Отсутствуют селекторы по портам TCP / UDP сокетов, именам процессов и некоторым другим.

2. Часть комбинаций туннелей не поддерживается. В частности, не поддерживаются вложения AH туннелей в ESP туннели.

На сообщения об обнаруженных ошибках разработчики IPsec для FreeBSD ответили, что в настоящее время IPsec в FreeBSD поддерживает только те функции, которые необходимы для функционирования IKEv2, поэтому обнаруженные ошибки не являются критичными.

5.2. Тестирование реализаций IKE v2

На момент проведения тестирования существовали четыре открытых реализации протокола IKEv2: OpenIKEv2, StrongSwan, IKEv2, Racoon2. Для

тестирования была выбрана реализация Racoon2-20090327c для операционной системы FreeBSD 6.2, что позволило использовать ряд компонентов, разработанных в рамках предыдущих проектов.

При выполнении тестового набора был выявлен ряд нарушений требований RFC 4306 и ошибок реализации:

- реализация не поддерживает диапазоны адресов и портов;
- реализация поддерживает префиксы для адресов, но не корректно с ними работает (например, когда на одном узле заданы адреса с префиксами, а на другом - точные);
- реализация игнорирует порядок заголовков в сообщениях;
- реализация не проверяет некоторые поля заголовков (во входящем запросе IKE_AUTH поле 'Exchange Type', во входящее ответе IKE_AUTH поле 'Responder's SPI');
- во входящем запросе REKEY IPsec SA не проверяется новый SPI (при совпадении с текущим, обмен успешно завершается, а IPsec SA в ядре ОС не создается);
- во входящих сообщениях обмена IKE_AUTH в блоке данных SA не проверяется ни количество Предложений, ни количество Преобразований. Если предложенный набор шире, чем требуется, он все равно может быть принят.
- при формировании запроса, содержащем в блоке данных SA несколько Предложений, начиная со второго Предложения поле SPI устанавливается 0;
- первый байт структур Предложение и Преобразование определен как не обязательный для обработки. Тем не менее, реализация опирается на него при разборе блока данных SA, и не правильные значения считает ошибкой.
- если в последней структуре Предложение первый байт ‘last’ выставлен не правильно (значение 2 вместо 0), реализация не проверяет, что блок данных SA закончился (исходя из поля ‘длина заголовка’ SA), и рассматривает следующую часть сообщения как еще одно Предложение, что приводит в лучшем к отбрасыванию пакета из-за не правильной структуры. При выполнении одного из тестов, произошло зацикливание реализации и потеря ее работоспособности, а также быстрое увеличение размера файла журнала.
- файл конфигурации racoon позволяет использовать опции esp_enc_alg и esp_auth_alg (предназначенные для ESP) для протокола AH. Причем эти опции установлены в настройках по умолчанию. При этом в обмене IKE_AUTH в структуре Предложение для протокола AH передаются преобразования Шифрование и Аутентификация. Если на другом узле такие же настройки, то обмен завершается созданием IPsec SA AH (правда только с алгоритмом аутентификации, алгоритм шифрования игнорируется ядром ОС).

Следует отметить, что в целом реализация соответствует спецификации с некоторыми ограничениями функциональности, такими как невозможность сужения селекторов трафика, не поддерживаемый протокол аутентификации EAP, невозможность одновременной обработки нескольких запросов.

Результаты тестирования доступны на сайте ipv6.ispras.ru.

6. Заключение

В ходе выполнения работы были выделены требования к реализациям IPsec v2, разработаны формальные спецификации и прототип тестового набора для верификации реализаций IPsec v2, в том числе реализаций протокола автоматического создания контекстов безопасности IKEv2. В статье описаны метод формализации требований IPsec v2, процесс создания тестового набора, а также результаты тестирования существующих реализаций.

Применение для решения задачи автоматизации тестирования таких традиционных для телекоммуникаций средств моделирования протоколов, как расширенные конечные автоматы и системы помеченных переходов (LTS), сталкивается с трудностями принципиального характера, обусловленными высокой сложностью протоколов Интернета.

Верификация функций безопасности протокола нового поколения IPsec v2 показала, что предложенный в данной работе метод верификации, основанный на контрактных спецификациях, позволяет эффективно автоматизировать тестирование таких сложных протоколов, как протоколы безопасности. При этом тестовые наборы обладают формально определенным и прослеживаемым покрытием требований, что в значительной степени улучшает качество тестирования.

Литература

[1] RFC4301 S. Kent, K. Seo. Security Architecture for the Internet Protocol December 2005

[2] RFC4302 S. Kent. IP Authentication Header S. Kent December

[3] RFC4303 S. Kent. IP Encapsulating Security Payload (ESP) December 2005

[4] RFC4306 C. Kaufman, Ed. Internet Key Exchange (IKEv2) Protocol December 2005

[5] RFC4307 J. Schiller. Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2) December 2005

[6] RFC4807 M. Baer, R. Charlet, W. Hardaker, R. Story, C. Wang. IPsec Security Policy Database Configuration MIB March 2007

[7] RFC4868 S. Kelly, S. Frankel. Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec May 2007

[8] Bourdonov, I., Kossatchev, A., Kuliamin, V., Petrenko, A. UniTesK Test Suite Architecture // Proceedings of FME, LNCS 2391. Springer-Verlag, 2002. P. 77-88.

[9] CTesK 2.1: SeC Language Reference. М.: ИСП РАН, 2005. 167 с.

[10] Н.В. Пакулин. Формализация стандартов и тестовых наборов протоколов Интернета. Автореферат диссертации на соискание учёной степени кандидата физико-математических наук. Москва, 2006.

[11] Н.В. Пакулин, А.В. Хорошилов "Разработка формальных моделей и тестирование соответствия для систем с асинхронными интерфейсами и телекоммуникационных протоколов", Журнал "Программирование" № 5, 2007 г., ISSN 0132-3474, с. 1-29.

[12] IETF RFC 2223. J. Postel, J. Reynolds. Instructions to RFC Authors. IETF, 1997. 20 с.

[13] IETF BCP 14 | IETF RFC 2119. S. Bradner. Key words for use in RFCs to Indicate Requirement Levels. IETF, 1997. 3 с.

[14] IETF RFC 1213. K. McCloghrie, M. T. Rose. Management Information Base for Network Management of TCP/IP-based internets: MIB-II. March 1991.

[15] IETF RFC 2011 K. McCloghrie, Ed. SNMPv2 Management Information Base for the Internet Protocol using SMIv2. November 1996.

[16] IETF RFC 1831. R. Srinivasan. RPC: Remote Procedure Call Protocol Specification Version 2, 1995.

[17] IETF RFC 1832. R. Srinivasan, XDR: External Data Representation Standard, 1995.