**DOI: 10.15514/ISPRAS-2023-35(5)-1**

CC BY

# Результаты переработки уровней ролевого управления доступом и мандатного контроля целостности формальной модели управления доступом ОС Astra Linux

*П.Н. Девянин,* ORCID: 0000-0003-2561-794X <*pdevyanin@astralinux.ru*>
*ООО «РусБИТех-Астра»,*
*117105, г. Москва, Варшавское ш., д. 26, стр.11*

**Аннотация.** Механизм управления доступом является базовым для обеспечения безопасности системного программного обеспечения (ПО) такого, как операционная система (ОС) или система управления базами данных (СУБД). В качестве научной основы реализации такого механизма, а также в соответствии с требованиями нормативных документов отечественных регуляторов к сертифицированным средствам защиты информации должна разрабатываться соответствующая критериям ГОСТ Р 59453.1-2021 формальная модель управления доступом. Такой формальной моделью для сертифицированной по высшим классам защиты и уровням доверия ОС Astra Linux является мандатная сущностно-ролевая ДП-модель управления доступом и информационными потоками в ОС семейства Linux (МРОСЛ ДП-модель). С учетом внедрения в механизм управления доступом ОС Astra Linux новых элементов, с целью обеспечения более точного соответствия описания модели этому механизму, развитию научно обоснованных технологий и практик разработки и верификации формальных моделей МРОСЛ ДП-модель регулярно перерабатывается. В настоящее время завершена очередная такая переработка модели для двух уровней ее иерархического представления, соответствующих ролевому управлению доступом (представляющему традиционное для ОС семейства Linux дискреционное управление доступом) и мандатному контролю целостности, отражающая наиболее существенные изменения в ОС Astra Linux релиза 2023 года. В статье анализируются основные результаты этой переработки, в рамках которой: введены функции, задающие новые метки сущностей, изменены состав и описания де-юре правил преобразования состояний системы, административных и запрещающих ролей, скорректированы формулировки и заново доказаны несколько утверждений, а также внесены другие изменения в описание модели.

**Ключевые слова:** формальная модель управления доступом; МРОСЛ ДП-модель; ролевое управление доступом; мандатный контроль целостности; операционная система; Astra Linux.

**Для цитирования:** 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. DOI: 10.15514/ISPRAS–2023–35 (5)–1.

# 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

*P.N. Devyanin, ORCID: 0000-0003-2561-794X <pdevyanin@astralinux.ru>*
*RusBITech-Astra*
*26, Varshavskoe, Moscow, 117105, Russia.*

**Abstract.** The access control mechanism is the basis for ensuring the security of system software (OS or DBMS). In accordance with the requirements of regulatory documents of domestic regulators for certified information security tools, as a scientific basis for the implementation of such a mechanism, a formal access control model that meets the GOST R 59453.1-2021 criteria should be developed. Such a formal model for the Astra Linux operating system certified for the highest protection classes and assurance levels is the mandatory entity-role model of access and information flows security control in OS of Linux family (MROSL DP-model). Taking into account the introduction of new elements into the access control mechanism of the Astra Linux and in order to ensure a more accurate correspondence of the model description to this mechanism, the development of scientifically based technologies and practices for the development and verification of formal models, the MROSL DP-model is regularly revised. Another such revision of the model now has been completed for two levels of its hierarchical representation, corresponding to role-based access control (representing discretionary access control, traditional for the OS of Linux family) and mandatory integrity control, reflecting the most significant changes in the Astra Linux release 2023. The article analyzes the main results of this revision, within which: functions that define new entity labels are introduced, the composition and descriptions of the de-jure rules for transforming system states, administrative and negative roles are changed, the wording is corrected and several statements are re-proved, and other changes are made in the model description.

**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. 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). DOI: 10.15514/ISPRAS-2022-35(5)-1.

## *1. Введение*

Системное программное обеспечение (ПО) такое, как операционные системы (ОС) или системы управления базами данных (СУБД), часто является основой программных средств защиты информации, одна из ключевых функций которых – это организация безопасного управления доступом субъектов доступа (процессов) к объектам доступа (сущностям, каталогам, файлам, сокетам, базам данных, таблицам, записям и др.). Для этого в такие средства включают механизм управления доступом, реализующий политики управления доступом [1]. В большинстве случаев (например, во многих ОС) это политика дискреционного управления доступом, а также (как, например, в ОС семейства Microsoft Windows) мандатного контроля целостности, реже – мандатного управления доступом. В СУБД, как правило, используется политика ролевого управления доступом.

В качестве научной основы реализации механизма управления доступом в соответствии с требованиями нормативных документов отечественных регуляторов к сертифицированным средствам защиты информации [2], в том числе к разработке для них безопасного ПО [3] должна использоваться формальная модель управления доступом [4, 5]. При этом такая модель должна соответствовать критериям, изложенным в ГОСТ Р 59453.1-2021 «Защита информации. Формальная модель управления доступом. Часть 1. Общие положения» [1], и быть верифицирована с применением инструментальных средств согласно рекомендациям ГОСТ Р 59453.2-2021 «Защита информации. Формальная модель управления доступом. Часть 2. Рекомендации по верификация формальной модели управления доступом» [6].

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 (операционной системы специального назначения Astra Linux Special Edition) [7, 8]. В модели объединены мандатное управление доступом, мандатный контроль целостности и ролевое управление доступом, выражающее традиционное для ОС семейства Linux дискреционное управление доступом. При этом все три политики сочетаются в модели с учетом их свойств, не противореча друг другу.

Разработка, верификация и внедрение МРОСЛ ДП-модели в ОС Astra Linux ведется «Группой Астра» в рамках реализации единой методологии разработки безопасного системного ПО [9]. Для этого МРОСЛ ДП-модель формируется в двух нотациях: математической (аналогично классическим моделям) [5] и формализованной на языке формального метода Event-B [10]. Модель в первой нотации предоставляет возможность объективного анализа ее корректности любым специалистом в области информационной безопасности, владеющим языком математики. В формализованной нотации модель верифицируется с применением инструментальных средств дедуктивно (с помощью инструментального средства Rodin) и по методу проверки моделей (с помощью инструментального средства ProB) [11].

В обеих нотациях МРОСЛ ДП-модель имеет иерархическое представление, позволяющее описывать ее по уровням (слоям). При этом каждый нижний уровень модели представляет абстрактную систему, элементы которой не зависят от новых элементов, принадлежащих более высокому уровню модели, который, в свою очередь, наследует, а при необходимости корректирует или дополняет элементы нижнего уровня. В целом иерархическое представление модели состоит из восьми уровней – четырех уровней для моделирования управления доступом непосредственно в ОС (ролевого 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, что обеспечивает согласованность механизмов управления доступом в ОС и СУБД. Такое разделение МРОСЛ ДП-модели на уровни соответствует подходу по ее реализации непосредственно в программном коде подсистемы безопасности PARSEC ОС Astra Linux.

Поскольку подсистема безопасности PARSEC регулярно модифицируется, то с целью обеспечения более точного соответствия описания модели этому механизму, развития научно обоснованных технологий и практик разработки и верификации формальных моделей МРОСЛ ДП-модель также регулярно перерабатывается. В настоящее время завершена очередная такая переработка модели для первых двух очень важных для реализации в ОС уровней ее иерархического представления: ролевого управления доступом и мандатного контроля целостности. Под влиянием выполненных в рамках модели теоретических исследований именно эти два вида управления доступом претерпели наиболее существенные изменения в ОС Astra Linux релиза 2023 г. Кроме того, был накоплен опыт верификации реализации модели в подсистеме безопасности PARSEC, также нашедший отражение в описании модели. В связи с этим в статье анализируются основные результаты соответствующей переработки модели в ее математической нотации, которая также отвечает рекомендациям, изложенным в разрабатываемом при участии специалистов «Группы Астра» проекте нового национального стандарта ГОСТ Р «Защита информации. Формальная модель управления доступом. Часть 3. Рекомендации по разработке».

Статья организована следующим образом. В следующем разделе рассматриваются основные элементы МРОСЛ ДП-модели, используемые для описания моделируемой системы – ОС Astra Linux. В разд. 3 анализируются изменения в описании состояний системы, включая

корректировки множеств административных и запрещающих ролей, введенные функции, задающие новые метки сущностей (объектов доступа, файлов или каталогов). В разд. 4 излагаются основные изменения в составе де-юре правил преобразования состояний системы и результаты корректировки условий безопасности системы в смысле мандатного контроля целостности. Заключение завершает статью, в нем подводятся итоги текущей переработки модели, а также рассматриваются дальнейшие направления этой работы.

## *2. Иерархическое представление МРОСЛ ДП-модели*

Как уже было отмечено в предыдущем разделе, МРОСЛ ДП-модель имеет иерархическое представление, которое кроме удобства при переработке модели, обеспечения условий для ее развития и внедрения в механизмах защиты ОС Astra Linux, также позволяет использовать для верификации модели технику пошагового уточнения (refinement) Event-B [11-13].

Иерархическое представление модели, соответствующее механизму управления доступом непосредственно в ОС Astra Linux, состоит из следующих четырех уровней (рис. 1):

- Уровень 1.1 (базовый) – модель системы ролевого (дискреционного) управления доступом («корень» иерархического представления, не наследующий ни один из других уровней);
- Уровень 2.1 – модель системы ролевого управления доступом и мандатного контроля целостности;
- Уровень 3.1 – модель системы ролевого управления доступом, мандатного контроля целостности и мандатного управления доступом только с информационными потоками по памяти;
- Уровень 4.1 – модель системы ролевого управления доступом, мандатного контроля целостности и мандатного управления доступом с информационными потоками по памяти и по времени.

Для СУБД PostgreSQL описаны четыре аналогичных уровня (1.2, 2.2, 3.2 и 4.2), что кроме согласованности механизмов управления доступом в ОС и СУБД, также предоставляет возможность анализа безопасности информационных потоков (скрытых каналов) по памяти и по времени между сущностями ОС и СУБД [5]. При этом уровни СУБД наследуют и дополняют как предшествующие уровни для СУБД, так и соответствующие уровни для ОС (например, уровень 2.2 наследует и дополняет уровни 1.2 и 2.1).

Для описания состояний моделируемой системы на уровне 1.1 (ролевого управления доступом) используются следующие основные элементы:

- Множества: учетных записей пользователей, субъектов (процессов), сущностей – объектов (файлов) и контейнеров (каталогов), допустимых имен сущностей, ролей, запрещающих и административных ролей, видов прав доступа, видов доступа, прав доступа к сущностям или субъектам, административных прав доступа к ролям, запрещающим или административным ролям, доступов субъектов к сущностям, административных доступов субъектов к ролям, запрещающим или административным ролям;
- Функции: имен сущностей, ролей, запрещающих ролей или административных ролей, иерархии сущностей, ролей, запрещающих ролей или административных ролей, прав доступа ролей, запрещающих ролей или административных ролей к сущностям, административных прав доступа к ролям, запрещающим или административным ролям запрещающих или административных ролей, необходимого обладания запрещающей ролью, наличия права доступа у текущих ролей, запрещающих или административных ролей субъекта к сущности, роли, запрещающей роли, административной роли или субъекту, доступа субъекта к

сущностям в контейнерах (каталогах), потенциальной доступности авторизованной роли или административной роли учетной записи пользователя, вида метки сущности.

*Рис. 1. Схема иерархического представления модели.*
*Fig. 1. Schema of hierarchical view of model.*

Например, функция доступа субъекта к сущностям в контейнерах задается следующим образом (при этом используются обозначения: $S$ — множество субъектов, $E$ — множество сущностей, $C$ — множество сущностей-контейнеров, $H_E$ – функция иерархии сущностей, *check_right* – функция наличия права доступа у текущих ролей, запрещающих или административных ролей субъекта к сущности, роли, запрещающей роли, административной роли или субъекту):

*execute_container:* $S \times E \to \{true, false\}$ *– функция доступа субъекта к сущностям в контейнерах такая, что по определению для субъекта* $s \in S$ *и сущности* $e \in E$ *справедливо равенство execute_container (s, e) = true тогда и только тогда, когда существует последовательность сущностей* $e_1, \ldots, e_n \in E$*, где* $e = e_n$ *и либо* $n = 1$*, либо* $n \geq 2$ *и выполняются следующие условия:*

*Условие 1. Не существует сущности-контейнера* $e_0 \in C$ *такой, что* $e_1 \in H_E(e_0)$*;*

*Условие 2. Верно* $e_i \in H_E(e_{i-1})$*, где* $1 < i \leq n$*;*

*Условие 3. Верно check_right* $(s, e_i, execute_r) = true$*, где* $1 \leq i < n$*.*

На уровне 2.1 (мандатного контроля целостности) для описания состояний моделируемой системы дополнительно используются следующие основные элементы:

- Множества: видов информационных потоков, информационных потоков, уровней целостности, учетных записей доверенных пользователей, учетных записей недоверенных пользователей, доверенных субъектов, недоверенных субъектов, сущностей-объектов, параметрически ассоциированных с учетными записями пользователей, сущностей-объектов, параметрически ассоциированных с субъектами, сущностей-объектов, функционально ассоциированных с субъектами, сущностей-объектов, параметрически ассоциированных с ролями, запрещающими ролями или административными ролями, сущностей-объектов, доступы на запись или чтение к которым не могут быть использованы для реализации информационных потоков по памяти;

- Функции: уровней целостности учетных записей пользователей, текущих уровней целостности субъектов, уровней целостности ролей, запрещающих ролей или административных ролей, уровней целостности сущностей, способа получения доступа на запись к сущностям-контейнерам, ролям, запрещающим ролям и административным ролям с учетом или без учета их уровня целостности, порядка получения доступа на чтение или использования права доступа на выполнение к сущностям, ролям, запрещающим ролям и административным ролям с учетом или без учета их уровня целостности, порядка активизации субъекта из сущности-объекта с учетом уровней целостности этого объекта, способа задания уровня целостности при создании в сущностях-контейнерах, ролях, запрещающих ролях и административных ролях с наследованием или без наследования их уровня целостности, де-факто владения субъектами.

Например, так задается решетка уровней целостности и функции уровней целостности (где дополнительно к ранее приведенным обозначениям используются следующие обозначения: *U* — множество учетных записей пользователей, *R, NR, AR* — множества ролей, запрещающих ролей и административных ролей, соответственно):

*(LI, ≤) — решетка уровней целостности, при этом заданы минимальный i_low и максимальный i_high элементы решетки, соответственно;*
$i_u$*: U → LI — функция, задающая для каждой учетной записи пользователя ее уровень целостности;*
$i_e$*: E → LI — функция, задающая уровень целостности для каждой сущности;*
$i_r$*: R ∪ NR ∪ AR → LI — функция, задающая для каждой роли, запрещающей роли или административной роли ее уровень целостности;*
$i_s$*: S → LI — функция, задающая для каждого субъекта его текущий уровень целостности.*

В МРОСЛ ДП-модели на уровнях 1.1 и 2.1 ее иерархического представления определены 35 де-юре правил преобразования состояний системы, предназначенных для формального описания (спецификации) следующих основных функций механизма ролевого (дискреционного) управления доступом и мандатного контроля целостности ОС Astra Linux:

- Создание, удаление, переименование, 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 учетных записей пользователей, сущностей, ролей, запрещающих ролей или административных ролей;
- Задание объектов, параметрически ассоциированных с учетными записями пользователей, ролями или административными ролями.

При этом для каждого де-юре правила задаются его параметры, условия и результаты применения. Например, де-юре правило *set_entity_owner (x, r, r', y)*, используемое для назначения субъектом x вместо роли r новой роли-владельца r' сущности y на уровне 2.1 модели задается согласно табл. 1 (где дополнительно к ранее приведенным обозначениям используются следующие обозначения: $write_a$, $read_a$ – доступы на запись и чтение, соответственно, $own_r$ – право доступа владения, *AA* – множество административных доступов, *root_role* – специальная роль, соответствующая полномочиям суперпользователя *root* в ОС семейства Linux, *PA* – функция прав доступа ролей, $H_E$ – функция иерархии сущностей, *direct* – функция вида метки сущности, *check_i_right* – функция наличия права доступа у текущих ролей, запрещающих или административных ролей субъекта к сущности, роли, запрещающей роли, административной роли или субъекту с учетом мандатного контроля целостности), в которой первая строка соответствует условиям и результатам применения правила, оставшимся без изменения с уровня 1.1 модели:

*Табл. 1. Де-юре правило set_entity_owner(x, r, r', y)*
*Table. 1. De-jure rule set_entity_owner(x, r, r', y)*

| **Условия** | **Результаты применения** |
|---|---|
| $x \in S, r, r' \in R \cup AR, y \in E,$<br>$\{(x, r', write_a), (x, root_role, read_a)\} \subseteq AA,$<br>$direct(y) = true,$<br>$[((x, r, write_a) \in AA, (y, own_r) \in PA(r))$ *или (для всех* $r'' \in R \cup AR$ *выполняется* $(y, own_r) \notin PA(r''))]$ | $PA'(r) = PA(r) \setminus \{(y, own_r)\},$<br>$PA'(r') = PA(r') \cup \{(y, own_r)\},$<br>*если* $H_E(y) = \{y' \in H_E(y): direct(y') = false\},$<br>*то для всех* $y' < y$ *верно*<br>$PA'(r) = PA(r) \setminus \{(y', own_r)\},$<br>$PA'(r') = PA(r') \cup \{(y', own_r)\}$ |
| $check_i_right(x, y, own_r) = true,$<br>$execute_i_container(x, y) = true$ | – |

Для теоретического анализа условий безопасности в рамках МРОСЛ ДП-модели, начиная с уровня 2.1 ее иерархического представления, аналогично де-юре правилам определены 10 де-факто правил преобразования состояний системы. Они не требуют непосредственной реализации в ОС Astra Linux и предназначены для задания условий создания информационных потоков по памяти и по времени или получения одним субъектом возможности управления другим субъектом.

Кроме того, в МРОСЛ ДП-модели 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 определение трех смыслов нарушения безопасности системы:

- На уровне 2.1 – в смысле мандатного контроля целостности, позволяющее недоверенному субъекту с низким уровнем целостности захватить управление (де-факто владение) субъектом с более высоким уровнем целостности;
- На уровне 3.1 – в смысле Белла-ЛаПадулы, результатом которого является создание запрещенного информационного потока по памяти «сверху-вниз»;
- На уровне 4.1 – 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 достаточные условия безопасности системы во всех трех смыслах. Например, для уровне 2.1 модели дается следующее определение безопасного начального состояния (где дополнительно к ранее приведенным обозначениям используются следующие обозначения: $write_m$ – информационный поток по памяти, $N_S$ – множество недоверенных субъектов, $O$ — множество сущностей-объектов, $A$ – множество доступов субъектов к сущностям, $F$ – множество информационных потоков, $[y]$, $]y[$ – множества сущностей-объектов, соответственно, функционально или параметрически ассоциированных с субъектом $y$, $de_facto_own$ – функция де-факто владения субъектами):

**Определение 1.** Начальное состояние $G_0$ системы назовем безопасным, когда оно удовлетворяет следующим условиям:

*Условие 1. Для каждых субъектов $x, y \in S_0$ таких, что $y \in de_facto_own_0(x)$, верно $i_{s0}(y) \le i_{s0}(x)$ (текущий уровень целостности субъекта, де-факто владеющего другим субъектом, не меньше текущего уровня целостности этого субъекта).*

*Условие 2. Для каждых недоверенного субъекта $x \in N_{S0}$, субъекта $y \in S_0$ и сущности-объекта $o \in O_0$ таких, что либо $(o \in [y]$ и $(x, o, write_m) \in F_0)$, либо $(o \in ]y[$ и либо $(o, x, write_m) \in F_0$, либо $(x, o, read_a) \in A_0)$, верно неравенство $i_{s0}(y) \le i_{s0}(x)$ (недоверенный субъект может иметь информационные потоки по памяти с участием сущностей-объектов, функционально или параметрически ассоциированных с другим субъектом, или иметь к таким параметрически ассоциированным сущностям-объектам доступ на чтение, только если текущий уровень целостности первого субъекта не меньше текущего уровня целостности второго субъекта).*

*Условие 3. Для каждого информационного потока $(x, y, write_m) \in F_0$ справедливо $i_{x0}(x) \ge i_{y0}(y)$, где $i_{x0}$ и $i_{y0}$ соответствующие функции $i_{e0}$ или $i_{s0}$ (отсутствуют информационные потоки по памяти от субъектов или сущностей с меньшим уровнем целостности к субъектам или сущностям с большим или несравнимым уровнем целостности).*

Далее в статье анализируются результаты переработки уровня 1.1 (ролевого управления доступом) и уровня 2.1 (мандатного контроля целостности) иерархического представления МРОСЛ ДП-модели.

## *3. Переработка описания состояний системы*

Основное внимание при переработке описания состояний системы в рамках МРОСЛ ДП-модели было уделено определению функций, соответствующих в ОС Astra Linux новым меткам файлов и каталогов, позволяющим повысить удобство и гибкость применения мандатного контроля целостности. То есть эти функции были определены на уровне 2.1 модели, всего их четыре:

1. $IRELAX: C \cup R \cup NR \cup AR \to \{true, false\}$ — функция, задающая способ получения доступа на запись к сущностям-контейнерам, ролям, запрещающим ролям или административным ролям с учетом или без учета их уровня целостности;

2. *SSI:* $E \cup R \cup NR \cup AR \rightarrow \{true, false\}$ — функция, задающая порядок получения доступа на чтение или использования права доступа на выполнение к сущностям, ролям, запрещающим ролям или административным ролям с учетом или без учета их уровня целостности;
3. *SILEV:* $O \rightarrow \{true, false\}$ — функция, задающая порядок активизации субъекта из сущности-объекта с учетом уровня целостности этого объекта, уровня целостности родительского субъекта и уровня целостности учетной записи пользователя, от имени которой активизируется субъект;
4. *IINH:* $C \cup R \cup NR \cup AR \rightarrow \{true, false\}$ — функция, определяющая способ задания уровня целостности при создании в сущностях-контейнерах, ролях, запрещающих ролях или административных ролях с наследованием или без наследования их уровня целостности.

При этом в модели говорится, что если значение функции *IRELAX, SSI, SILEV* или *IINH* от сущности или какой-либо роли равно *true*, то эта сущность или роль помечена с помощью соответствующей функции.

Для всех видов ролей функция *IRELAX* задана только для обеспечения единообразия их атрибутов с сущностями-контейнерами (так как для ролей она по определению всегда равна *false*). При этом, если доступ на запись к сущности-контейнеру $c \in C$ разрешен субъекту только с текущим уровнем целостности, большим или равным уровня целостности этой сущности-контейнера, то по определению выполняется равенство *IRELAX(c) = false*, в противном случае выполняется равенство *IRELAX(c) = true*. В ОС такой помеченной с помощью функции *IRELAX* сущностью-контейнером является каталог «*/tmp*», доступ на запись к которому необходимо обеспечить процессам с любым текущим уровнем целостности.

При определении функции *SSI* задано, что если доступ на чтение или использование права доступа на выполнение к сущности или роли $e \in E \cup R \cup NR \cup AR$ разрешен субъекту только с текущим уровнем целостности, большим или равным уровня целостности этой сущности или роли, то по определению выполняется равенство *SSI(e) = true*, в противном случае выполняется равенство *SSI(e) = false*. Включение этой функции в модель связано с тем, что изначально мандатный контроль целостности накладывает ограничения на получение доступа на запись к сущностям, но не на чтение. Вместе с тем, в ОС существуют файлы или каталоги, доступ к которым на чтение (а к каталогам и на выполнение) необходимо разрешить только процессам с текущим уровнем целостности, не меньшим, чем у этих файлов или каталогов. Примерами этого являются некоторые файлы с высоким уровнем целостности из каталогов «*/etc*» и «*/dev*». Аналогично пометка некоторых ролей с помощью функции *SSI* позволяет запретить их получение в качестве текущих субъектами, обладающими текущим уровнем целостности, меньшим или несравнимым, чем у соответствующей роли (раньше такой запрет действовал на все роли). Таким образом, теперь даже обладающие высоким уровнем целостности административные роли можно делать (не помечая их с помощью функции *SSI*) текущими для недоверенных (с низким текущим уровнем целостности) субъектов, что позволит в будущем повысить гибкость ролевого управления доступом.

При активизации субъекта из сущности-объекта, помеченной с помощью функции *SILEV*, уровень целостности учетной записи пользователя, от имени которой он будет функционировать, должен быть не ниже уровня целостности объекта, и текущий уровень целостности субъекта устанавливается равным уровню целостности этого объекта. В ОС это позволяет процессам в сессии учетной записи пользователя с низким уровнем целостности запускать некоторые процессы, которым для выполнения их функций необходим высокий текущий уровень целостности. Примером использования такого помеченного с помощью функции *SILEV* объекта в ОС Astra Linux является исполняемый файл консольной утилиты

*passwd*, запускаемой для изменения пароля низкоцелостной учетной записи пользователя, образ которого хранится в обладающем высоким уровнем целостности файле «*/ets/shadow*».

Если при создании в сущности-контейнере или в роли любого вида $c \in C \cup R \cup NR \cup AR$ необходимо наследование их уровня целостности (с учетом возможной их пометки с помощью функции *IRELAX*), то по определению выполняется равенство *IINH(c) = true*, в противном случае, когда на создаваемые сущности или роли устанавливается минимальный уровень целостности (*i_low*), выполняется равенство *IINH(c) = false* (для ролей всех видов по определению задано, что все они помечены с помощью функции *IINH*). С использованием этой функции переопределен порядок наследования уровня целостности создаваемой сущности от уровня целостности родительской сущности-контейнера. По умолчанию наследования нет, то есть целостность создаваемой сущности устанавливается на минимальном уровне (*i_low*). Наследование осуществляется, когда родительская сущность-контейнер помечена с помощью функции *IINH*. При этом с учетом опыта реализации мандатного контроля целостности в ОС Astra Linux и с использованием функции *IRELAX* требуется:

- При создании в сущности-контейнере, не помеченной с помощью функции *IRELAX*, новой сущности она наследует уровень целостности от сущности-контейнера;
- Когда сущность-контейнер также помечена с помощью функции *IRELAX*, уровень целостности новой сущности устанавливается равным наибольшей нижней границе (в решетке уровней целостности) значений уровней целостности сущности-контейнера и текущего уровня целостности осуществляющего создание субъекта.

Для удобства использования функции *IINH* в ОС пометка с помощью этой функции наследуется на создаваемые подкаталоги от родительских каталогов, что также отражено в модели.

Кроме включения в МРОСЛ ДП-модель при ее переработке новых функций на уровне 1.1 реализована возможность назначения запрещающим ролям не только прав доступа к сущностям или субъектам, а по сути, запрещающих административных прав доступа к другим ролям или административным ролям, что делает запрещающие роли по назначению близкими к административным ролям. При этом во избежание разного рода взаимных запретов запрещающие роли не могут иметь административных прав доступа к другим запрещающим ролям, специальным административным ролям и ряду других ролей. Кроме того, запрещающие роли могут иметь к ролям или административным ролям только административные права доступа на чтение или запись.

Также существенные изменения при переработке модели коснулись состава ролей. На уровне 1.1 модели для администрирования ролей, запрещающих ролей и административных ролей добавлены специальные административные роли, соответственно, *cap_role, admin_cap_role, negative_cap_role*, назначение которых близко к имеющейся в ОС привилегии *PARSEC_CAP_CAP* (она позволяет давать или отзывать другие привилегии). При этом добавлена специальная административная роль *root_role*, предназначенная для моделирования ситуации использования штатных для ОС семейства Linux полномочий суперпользователя *root*, функционирующим от имени которого процессам разрешено игнорировать основное для этих ОС дискреционное управление доступом. Этой роли на уровне 2.1 модели назначается минимальный уровень целостности (*i_low*), так как изменять дискреционные права доступа (получать к *root_role* доступ на запись) в ОС Astra Linux можно и на низком уровне целостности. Всем остальным специальным административным ролям назначается максимальный уровень целостности (*i_high*), что по-прежнему дает возможность их администрирования только доверенным (обладающим максимальным уровнем целостности) субъектам. Однако только предназначенная для администрирования административных ролей административная роль *admin_cap_role* помечается с помощью

функции *SSI*, что обеспечивает возможность ее получения как текущей только доверенными субъектами.

На уровне 2.1 МРОСЛ ДП-модели для придания большего соответствия технологиям, используемым в ОС Astra Linux, добавляются еще три специальные административные роли. Первая из них – *chmac_role* соответствует привилегии *PARSEC_CAP_CHMAC* и позволяет обладающему ею как текущей субъекту понижать уровни целостности сущностей, когда они не выше текущего уровня целостности этого субъекта. Вторая – *inherit_integrity_role* соответствует привилегии *PARSEC_CAP_INHERIT_INTEGRITY* и обеспечивает субъекту возможность создавать в сущности-контейнере (каталоге) новые сущности с наследованием уровня целостности от этой сущности-контейнера, как если бы он была помечена с помощью функции IINH. Третья – *setmac_role* соответствует привилегии *PARSEC_CAP_SETMAC* и разрешает субъекту активизировать нового субъекта с текущим уровнем целостности меньшим, чем у активизирующего субъекта (по умолчанию их текущие уровни целостности должны быть равными), что будет полезно в ОС Astra Linux для реализации механизма "песочниц" для функционирования недоверенного ПО на низких уровнях целостности.

С учетом опыта исследования ОС Astra Linux по сравнению с предыдущими представлениями модели при ее текущей доработке на уровне 2.1 сделано уточнение, что ассоциированными с учетными записями пользователей, субъектами или ролями могут быть только сущности-объекты (раньше это были любые сущности). Это связано с тем, что до сих пор не было выявлено случаев, когда именно сущности-контейнеры (каталоги) без содержащихся в них сущностей-объектов (файлов) являлись бы ассоциированными параметрически или функционально с учетными записями пользователей, субъектами или ролями. Такой подход позволил уточнить, а в ряде случаев упростить формулировки некоторых положений модели. При этом в определении абсолютной функциональной корректности удалось отказаться от информационных потоков по памяти (их сложно анализировать в реальной ОС) и свести к доступу субъекта на чтение к сущности, т.к. только такой доступ субъекта с большим текущим уровнем целостности к сущности с меньшим уровнем целостности создает потенциальную угрозу возникновения при участии этого субъекта информационного потока по памяти к обладающей высоким уровнем целостности сущности-объекту, функционально ассоциированной с другим субъектом. Однако для параметрической корректности аналогичный результат пока получить не удалось, т.к. для нее существенны в общем случае не запрещаемые мандатным контролем целостности информационные потоки по памяти от обладающих высоким уровнем целостности сущностей-объектов, параметрически ассоциированных с субъектами, к сущностям с низким уровнем целостности.

Внесенные изменения в описания состояний системы в рамках МРОСЛ ДП-модели были учтены при переработке описаний и изменении состава де-юре правил преобразования состояний системы, а также при корректировке условий безопасности системы в смысле мандатного контроля целостности и, как следствие, повторном доказательстве их достаточности. Все перечисленное анализируется в следующем разделе.

## *4. Изменения в составе де-юре правил преобразования состояний системы и корректировка условий безопасности*

В первую очередь изменения в составе и описаниях де-юре правил преобразования состояний системы были направлены на обеспечение их большего соответствия реализованным в подсистеме безопасности PARSEC ОС Astra Linux обрабатывающим системные вызовы функциям. Относящиеся к чисто ролевому управлению доступом де-юре правила менялись меньше, чтобы не увеличивать и так большой объем правок модели (это оставлено на будущее, когда ролевое управление доступом будет непосредственно реализовываться в ОС). В результате уже на уровне 1.1 модели были осуществлены следующие основные изменения:

- Правила *access_write(x, y)* и *access_read(x, y)*, дающие возможность субъекту *x* получить, соответственно, доступ на запись или чтение к сущности или роли *y*, заменены одним соответствующим системному вызову *open* правилом *take_access (x, y, {$\alpha_{oj}$: 1 ≤ j ≤ k})*, где $\alpha_{oj}$ – предоставляемые доступы на чтение или запись;
- Правила *grant_rights (x, r, y, {$\alpha_{rj}$: 1 ≤ j ≤ k})* и *remove_rights (x, r, y, {$\alpha_{rj}$: 1 ≤ j ≤ k})*, позволявшие субъекту *x* дать или забрать у роли *r* права доступа *{$\alpha_{rj}$: 1 ≤ j ≤ k}* к сущности *y*, заменены одним соответствующим системному вызову *chmod* правилом *set_rights (x, r, y, {$\alpha_{rj}$: 1 ≤ j ≤ k}, plus, t)*, в зависимости от булева параметра *plus*, добавляющим или удаляющим права доступа у роли *r*. Кроме того, в это правило добавлен булевский параметр *t*, который в случае, когда *y* является сущностью-контейнером (каталогом), указывает, является ли она разделяемой (в ОС такой каталог имеет атрибут *sticky bit*) или нет, что позволило отказаться от отдельного устанавливающего параметр *t* де-юре правила *set_container_attr (x, y, t)*;
- Правила *add_negative_owner (x, nr, y)* и *remove_negative_owner (x, nr, y)*, использованные ранее для назначения субъектом x запрещающей роли-владельца nr сущности y, заменены одним правилом *set_negative_owner (x, nr, y, plus)*, что позволяет моделировать планируемое в будущем развитие функционала системного вызова *chown*.

Также на уровне 1.1 модели путем добавления нового де-юре правила *apply_user_role(x, u, r)* реализована возможность применения субъектом *x* авторизованной роли или административной роли r учетной записи некоторого пользователя u. Такое допустимо в ОС, когда некоторому ее процессу (например, средству виртуализации Libvirt) поступает запрос (в модели этому соответствует использование роли), который надо выполнить от имени некоторой учетной записи пользователя. При этом в правиле проверяется, что субъект *x* может применить роль *r* только при существовании последовательности авторизованных административных (кроме, возможно, последней роли в последовательности) ролей учетной записи пользователя *u*, начинающейся с ее индивидуальной административной роли (для каждой учетной записи пользователя задается такая административная роль), и заканчивающейся *r*, где каждая авторизованная административная роль в последовательности обладает административным правом доступа на чтение к последующей роли, и во множествах запрещающих ролей для ролей последовательности нет ни одной запрещающей роли, обладающей административным правом доступа на чтение хотя бы к одной из ролей последовательности. При этом на уровне 2.1 дополнительно проверяется, во-первых, отсутствие запретов на получения доступов на чтение к ролям последовательности с учетом их пометки с помощью функции *SSI* и отсутствие у этих ролей параметрически ассоциированных объектов. Во-вторых, субъект *x* должен иметь текущий уровень целостности не ниже уровня целостности учетной записи пользователя *u* и доступ на чтение или запись ко всем объектам, параметрически с ней ассоциированных.

Следующей существенной правкой описаний де-юре правил преобразование состояний стало добавление на уровне 2.1 модели в позволяющее менять параметры ролей всех видов правило *set_role_labels (x, r, ri, ssi, ro)* возможности изменения их уровней целостности. Аналогично сущностям понижение уровня целостности роли, запрещающей роли или административной роли разрешено только субъекту, обладающему текущим уровнем целостности не ниже уровня целостности этой роли и обладающему текущей специальной административной ролью *root_role* и текущей специальной административной ролью *cap_role*, *negative_cap_role* или *admin_cap_role*, соответственно. При этом для повышения или назначения несравнимого уровня целостности этой роли или ее пометки с помощью функции *SSI* (кроме запрещающих ролей) дополнительно требуется, чтобы субъект был доверенным (с максимальным текущим уровнем целостности *i_high*) и обладал текущей специальной административной ролью *admin_cap_role*.

Кроме того, на уровне 2.1 описания условий применения де-юре правил, определяющих порядок администрирования уровней целостности сущностей и значений от них функций *IRELAX, SSI, SILEV* и *IINH*, приближены к тому, что для этого сделано или планируется сделать в ОС Astra Linux. Так изменение значений функций *IINH*, *SSI* или *IRELAX* от сущности (функций *IINH* и *IRELAX* только для сущности-контейнера) разрешено субъекту с текущим уровнем целостности не ниже уровня целостности этой сущности. Понижение уровня целостности сущности разрешено только субъекту, обладающему текущим уровнем целостности не ниже уровня целостности этой сущности и обладающему текущими специальными административными ролями *root_role* и *chmac_role* (как аналогом привилегии *PARSEC_CAP_CHMAC*). При этом для повышения или назначения несравнимого уровня целостности этой сущности или изменения ее параметров, задаваемых функцией *SILEV,* дополнительно требуется, чтобы субъект был доверенным и обладал текущей специальной административной ролью *admin_cap_role*.

Приведенные в предыдущем и текущем разделах изменения в описаниях состояний и де-юре правил их преобразования потребовали на уровне 2.1 МРОСЛ ДП-модели скорректировать формулировки теоремы об условиях безопасности системы в смысле мандатного контроля целостности, дополнив их требованием пометки с помощью функции *SSI* сущностей, параметрически ассоциированных с субъектами (в этих формулировках условий дополнительно к ранее приведенным обозначениям используются следующие обозначения: $G_i \vdash_{opi} G_{i+1}$ – переход системы из состояния $G_i$ в состояние $G_{i+1}$ в результате применения де-юре или де-факто правила $op_i$, *af_correct* – функция, задающая для каждого субъекта множество пар вида субъект и сущность, относительно которых первый субъект абсолютно функционально корректен, *ap_correct* – функция, задающая для каждого субъекта множество пар вида субъект и сущность или субъект, относительно которых первый субъект абсолютно параметрически корректен).

**Определение 2.** Пусть $G_0$ — безопасное начальное состояние системы, и существует траектория без кооперации доверенных и недоверенных субъектов $G_0 \vdash_{op1} G_1 \vdash_{op2} \dots \vdash_{opN} G_N$ (траектория, на которой доверенные субъекты не инициируют выполнение де-юре правил преобразования состояний), где $N \geq 1$. Будем говорить, что в состоянии $G_N$ произошло нарушение безопасности системы в смысле мандатного контроля целостности, когда существуют недоверенный субъект $x \in N_{SN}$ и субъект $y \in de_facto_own_N(x)$ такие, что не верно неравенство $i_s(y) \leq i_s(x)$, и это условие не выполняется в состояниях $G_i$ траектории, где $0 \leq i < N$. Назовём систему безопасной в смысле мандатного контроля целостности, когда в ней невозможно соответствующее нарушение безопасности.

***Теорема 1.*** *Пусть $G_0$ — безопасное начальное состояние системы (в смысле определения 1). Пусть на всех траекториях системы без кооперации доверенных или недоверенных субъектов $G_0 \vdash_{op1} G_1 \vdash_{op2} \dots \vdash_{opN} G_N$, где $N \geq 0$, и в каждом состоянии $G_N$ для каждого субъекта $s \in S_N$ и сущности $e \in E_N$ выполняются следующие условия:*

*<u>Условие 1</u>. Если $o \in [s]$, то выполняется условие $i_{sN}(s) \leq i_{eN}(o)$ (уровень целостности объекта не ниже текущего уровня целостности субъекта, с которым он функционально ассоциирован).*

*<u>Условие 2</u>. Если $o \in ]s[$, то $i_{sN}(s) \leq i_{eN}(o)$ и либо $SSI_N(o) = true$, либо для каждой роли или административной роли $r \in R_N \cup AR_N$ такой, что $(o, read_r) \in PA_N(r)$, выполняется условие $i_{eN}(o) \leq i_{rN}(r)$ и $SSI_N(r) = true$ (уровень целостности объекта не ниже текущего уровня целостности субъекта, с которым он параметрически ассоциирован, и, кроме того, объект либо помечен с помощью функции SSI, либо любая обладающая к нему правом доступа на чтение роль или административная роль имеет уровень целостности не ниже, чем у объекта, и помечена с помощью функции SSI).*

*<u>Условие 3</u>. Для всех субъектов $s \in S_N$ таких, что $i_low < i_{sN}(s)$, выполняются условия $\{s' \in S_N \mid i_low < i_{sN}(s') \leq i_{sN}(s)\} \times E_N \subset af_correct_N(s)$, $\{s' \in S_N \mid i_low < i_{sN}(s') \leq i_{sN}(s)\} \times (E_N \cup S_N) \subset$*

*$ap_correct_N$ (s) (каждый субъект с текущим уровнем целостности выше минимального абсолютно функционально и параметрически корректен относительно субъектов с не большим, чем у него, и большим минимального текущим уровнем целостности, и любых сущностей).*

*Тогда на этих траекториях система безопасна в смысле мандатного контроля целостности (в смысле определения 2).*

Следует отметить, что в определении 2 рассматриваются только траектории без кооперации доверенных и недоверенных субъектов, на которых доверенные субъекты не инициируют выполнение де-юре правил (на них они могут инициировать выполнение только некоторых де-факто правил). То есть, как это принято при разработке формальных моделей управления доступом, предполагается, что проверка безопасности системы должна осуществляться после того, как доверенные субъекты выполнили свои задачи по администрированию системы.

С практической точки зрения реализация требований пометки сущностей с помощью функции *SSI* (условие 2 теоремы 1) гораздо удобнее при администрировании ОС Astra Linux, чем только использованные ранее ограничения на предоставление к таким сущностям прав доступа на чтение ролям или административным ролям с уровнем целостности не ниже, чем у этих сущностей. При этом было осуществлено повторное доказательство данной теоремы, с учетом всех внесенных в модель корректировок.

## *5. Заключение и направления дальнейшей переработки модели*

В настоящей статье изложены и проанализированы основные результаты очередной переработки описания МРОСЛ ДП-модели, а, именно, двух уровней ее иерархического представления, соответствующих ролевому управлению доступом и мандатному контролю целостности. При этом акцентировалось внимание на мотивированность этой переработки модели необходимостью отражения в ее описании изменений, планируемых или уже внесенных в подсистему безопасности PARSEC ОС Astra Linux релиза 2023 года.

Осуществляемая переработка модели является частью реализуемой «Группой Астра» методологии разработки безопасного системного ПО. Кроме того, поскольку с участием специалистов «Группы Астра» в настоящее время готовится проект нового национального стандарта ГОСТ Р «Защита информации. Формальная модель управления доступом. Часть 3. Рекомендации по разработке», которому, как и его предыдущим частям ГОСТ Р 59453.1-2021 и ГОСТ Р 59453.2-2021, полностью соответствует МРОСЛ ДП-модель, то настоящую ее переработку можно также считать частью процесса апробации этого проекта национального стандарта.

В будущем планируется завершить переработку других уровней иерархического представления МРОСЛ ДП-модели, соответствующих мандатному управлению доступом и штатной для ОС Astra Linux СУБД PostgreSQL.

## 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]. 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, 07.12.2023 / 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 07.12.2023. (in Russian).

[3]. 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, 07.12.2023 / 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 07.12.2023. (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]. 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).

[7]. Операционная система специального назначения Astra Linux Special Edition. Доступно по ссылке: https://astralinux.ru/software-services/os/, 07.12.2023. / Astra Linux Special Edition operating system. Available at: https://astralinux.ru/software-services/os/, accessed 07.12.2023.

[8]. 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).

[9]. 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).

[10]. Abrial J.-R., Butler M., Hallerstede S. et al. Rodin: An open toolset for modelling and reasoning in Event-B // International Journal on Software Tools for Technology Transfer. 2010. Vol. 12, no. 6, pp. 447-466.

[11]. 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).

[12]. Abrial J.-R., Hallerstede S. Refinement, decomposition, and instantiation of discrete models: Application to Event-B // Fundamenta Informaticae, Volume 77, Issue 1-2, 2007, pp. 1-28.

[13]. 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).

## 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 – Dr. Sci. (Tech.), 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.