DOI: 10.15514/ISPRAS-2025-37(3)-19

Методы динамической верификации промышленных средств защиты информации на основе формальных моделей управления доступом.

[1,2,3] *А.К. Петренко, ORCID: 0000-0001-7411-3831 <petrenko@ispras.ru>*
[4] *П.Н. Девянин, ORCID: 0000-0003-2561-794X <pdevyanin@astralinux.ru>*
[1] *Д.В. Ефремов, ORCID: 0000-0002-9916-056X <efremov@ispras.ru>*
[1,3] *А.А. Карнов, ORCID: 0000-0002-2066-9946 <karnov@ispras.ru>*
[1,2] *Е.В. Корныхин, ORCID: 0000-0001-9303-3132 <kornevgen@ispras.ru>*
[1,2,3] *В.В. Кулямин, ORCID: 0000-0003-3439-9534 <kuliamin@ispras.ru*
[1,2,3,5] *А.В. Хорошилов, ORCID: 0000-0002-6512-4632 <khoroshilov@ispras.ru>*

[1] *Институт системного программирования имени В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25.*

[2] *Московский государственный университет имени М.В. Ломоносова, 119991, Россия, Москва, Ленинские горы, д. 1.*

[3] *НИУ Высшая школа экономики, 101978, Россия, г. Москва, ул. Мясницкая, д. 20.*

[4] *ООО «РусБИТех-Астра» 117105, Россия, Москва, Варшавское шоссе, д. 26, стр. 11.*

[5] *Московский физико-технический институт, 141701, Россия, Московская область, г. Долгопрудный, Институтский пер., 9.*

Аннотация. В статье рассматриваются методы динамической верификации программных систем, представляющих собой средства защиты информации (СЗИ) или включающих такие средства в свой состав. Для обеспечения высокого уровня доверия и защищенности программных систем необходимо применять разные методы и технологии верификации, при этом важны не только потенциальная мощность метода, но возможность использовать его в реальных условиях промышленной разработки крупных и сложных программных систем. Строгость и точность верификации обеспечивают формальные методы, однако использование классических формальных методов диктует особые, крайне высокие требования к персоналу и влечёт за собой дополнительные трудозатраты. Статья предлагает технологию динамической верификации СЗИ, которая, с одной стороны, близка к техникам тестирования, поэтому проще для освоения инженерами-тестировщиками, и, с другой стороны, в качестве базы использует формальные модели управления доступом и спецификации внешних интерфейсов СЗИ, которые уже появляются у разработчиков ОС и СУБД, чья продукция должна соответствовать требованиям нового национального стандарта ГОСТ Р 59453.4-2025 «Защита информации. Формальная модель управления доступом. Часть 4. Рекомендации по верификации средства защиты информации, реализующего политики управления доступом, на основе формализованных описаний модели управления доступом». Данный стандарт также представлен в статье.

Ключевые слова: динамическая верификация; мониторинг; тестирование на основе моделей; формальная модель управления доступом; функциональная спецификация.

Для цитирования: ПетренкоА.К., Девянин П.Н., Ефремов Д.В., Карнов А.А., Корныхин Е.В., Кулямин В.В., Хорошилов А.В. Методы динамической верификации промышленных средств защиты информации на основе формальных моделей управления доступом. Труды ИСП РАН, том 37, вып. 3, 2025 г., стр. 277–290. DOI: 10.15514/ISPRAS–2025–37(3)–19.

Methods of Runtime Verification of Industrial Information Security Tools Based on Formal Access Control Models.

[1,2,3] *A.K. Petrenko, ORCID: 0000-0001-7411-3831 <petrenko@ispras.ru>*
[4] *P.N. Devyanin, ORCID: 0000-0003-2561-794X <pdevyanin@astralinux.ru>*
[1] *D.V. Efremov, ORCID: 0000-0002-9916-056X <efremov@ispras.ru>*
[1,3] *A.A. Karnov, ORCID: 0000-0002-2066-9946 <karnov@ispras.ru>*
[1,2] *E.V. Kornykhin, ORCID: 0000-0001-9303-3132 <kornevgen@ispras.ru>*
[1,2,3] *V.V. Kuliamin, ORCID: 0000-0003-3439-9534 <kuliamin@ispras.ru>*
[1,2,3,5] *A.V. Khoroshilov, ORCID: 0000-0002-6512-4632 <khoroshilov@ispras.ru>*

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

[2] *Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russia.*

[3] *National Research University, Higher School of Economics 20, Myasnitskaya ulitsa, Moscow, 101978, Russia.*

[4] *RusBITech-Astra, 26, Varshavskoe, Moscow, 117105, Russia.*

[5] *Moscow Institute of Physics and Technology (MIPT), 9, Campus per., Dolgoprudny, Moscow region, 114701, Russia.*

Abstract. The paper discusses methods of runtime verification of software systems that are security protection mechanisms (PM) or include such mechanisms in their design. To ensure a high level of trust and security of software systems, it is necessary to use different verification methods and technologies. In this case, not only the potential power of the method is important, but also the possibility of using it in real conditions of industrial development of large and complex software systems. The rigor and accuracy of verification are ensured by formal methods; however, the use of classical formal methods dictates special, extremely high requirements for personnel and entails additional labor costs. The article proposes a technology for runtime verification of PM, which, on the one hand, is close to testing techniques, therefore it is easier for test engineers to master, and, on the other hand, uses formal access control models and specifications of external PM interfaces as a basis, which are already appearing among OS and DBMS developers, whose products must meet the requirements of the new national standard GOST R 59453.4-2025 "Information Security. Formal access control model. Part 4. Recommendations for verification of information security tools implementing access control policies based on formalized descriptions of the access control model. This standard is also presented in the article.

Keywords: runtime verification; monitoring; Model Based Testing; formal access control model; functional specification.

For citation: Petrenko A.K., Devyanin P.N., Efremov D.V., Karnov A.A., Kornykhin E.V., Kuliamin V.V., Khoroshilov A.V. Methods of runtime verification of industrial information security tools based on formal access control models. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 3, 2025. pp. 277-290 (in Russian). DOI: 10.15514/ISPRAS-2025-37(3)-19.

1. Введение

Теме обеспечения защиты программных и программно-аппаратных систем уделяется все большее внимание. Особенно важно выполнение требований информационной безопасности к самим средствам защиты информации (СЗИ). Спектр СЗИ достаточно широк, но в данной статье мы фокусируемся на СЗИ, которые входят в состав и, по сути, даже выполняют часть функций по защите информации таких компонентов базового программного обеспечения (ПО) как операционные системы (ОС) и системы управления базами данных (СУБД). Такие СЗИ обеспечивают контроль за обеспечением корректности предоставления (или запрета) доступа тех или иных программ/процессов (субъектов доступа) к тем или иным программным и программно-аппаратным сущностям (объектам доступа, файлам, каналам ввода-вывода, процессам и т.д.). Роль ОС и СУБД является основополагающей в обеспечении информационной безопасности, этим и объясняется выбранный в данной статье фокус рассмотрения, хотя рассматриваемый метод и технологии подобного рода, конечно, могут применяться и для верификации СЗИ других видов.

Поводом к появлению данной публикации стали завершение разработки и утверждение нового национального стандарта ГОСТ Р 59453.4-2025 «Защита информации. Формальная модель управления доступом. Часть 4. Рекомендации по верификации средства защиты информации, реализующего политики управления доступом, на основе формализованных описаний модели управления доступом» [1]. Данный ГОСТ продолжает серию стандартов ГОСТ Р 59453:

- ГОСТ Р 59453.1-2021 «Защита информации. Формальная модель управления доступом. Часть 1. Общие положения» [2];
- ГОСТ Р 59453.2-2021 «Защита информации. Формальная модель управления доступом. Часть 2. Рекомендации по верификации формальной модели управления доступом» [3];
- ГОСТ Р 59453.3-2025 «Защита информации. Формальная модель управления доступом. Часть 3. Рекомендации по разработке» [4].

Стандарты были разработаны Федеральной службой по техническому и экспортному контролю (ФСТЭК России), Институтом системного программирования им. В.П. Иванникова Российской академии наук (ИСП РАН), Обществом с ограниченной ответственностью «РусБИТех-Астра» (ООО «РусБИТех-Астра») и Федеральным автономным учреждением «Государственный научно-исследовательский испытательный институт проблем технической защиты информации Федеральной службы по техническому и экспортному контролю» (ФАУ «ГНИИИ ПТЗИ ФСТЭК России»).

Первый из этих стандартов содержит общие положения и критерии, которым должны соответствовать формальные модели управления доступом (ФМД). Второй стандарт дает рекомендации по использованию методов верификации и определяет требования к инструментам верификации формальных моделей управления доступом. Третий стандарт описывает расширенные рекомендации по разработке формальных моделей в условиях создания и сертификации крупных программных систем, в частности, отечественных ОС.

Далее в статье представлены краткие сведения о стандарте, описывающем рекомендации по динамической верификации средства защиты информации, реализующего политики управления доступом [1], общая схема верификации СЗИ, методы и технологии, которые могут использоваться в рамках схемы, и требования к их выбору, обусловленные типовыми сценариями верификации и сертификации СЗИ.

2. ГОСТ Р 59453.4-2025. «Защита информации. Формальная модель управления доступом. Часть 4»

Стандарт представляет собой рекомендации по верификации средств защиты информации, реализующих политики управления доступом, на основе формализованного описания модели управления доступом. Он предназначен для разработчиков средств защиты информации, а также для органов по сертификации и испытательных лабораторий при проведении сертификации СЗИ.

В контексте развития и расширения использования перспективных технологий данный стандарт открывает возможности одного из современных подходов к задачам повышения надежности и защищенности программных и программно-аппаратных систем, который известен как динамическая верификации (Runtime Verification – RV) или тестирование на основе формальных моделей (Model Based Testing – MBT[1]). Наиболее известным международным стандартом, в котором RV и MBT представляется как один из рекомендуемых элементов технологической цепочки создания программных систем ответственного назначения, является DO-178C [5] и его дополнения DO-331 (Model-Based Development and Verification Supplement to DO-178C and DO-278A) и DO-333 (Formal Methods Supplement to DO-178C and DO-278A). В Российской Федерации действует аналог DO-178C, который известен как КТ-178C [6].

Семейство стандартов DO-178 широко используется в области аэрокосмической техники. Область распространения DO-331 и DO-333 пока еще не велика, но она активно расширяется. Главным фактором, который способствует применению формальных методов в разработке и сертификации программных систем ответственного назначения, является концентрация внимания основных участников программного проекта: аналитиков, архитекторов, инженеров по разработке требований, разработчиков и верификаторов - на задаче четкого определения функциональных требований к интерфейсам системы с внешним миром и к внутренним интерфейсам. Тщательная проработка интерфейсов и поддержка их спецификаций в актуальном и консистентном состоянии позволяет на строгой и систематической основе контролировать точность и корректность реализации проектных решений, на ранних стадиях проектирования и разработки выявлять отклонения от них и принимать меры по их устранению. То есть главный позитивный вклад MBT (тестирования на основе формальных моделей) состоит не столько в автоматизации собственно тестирования, сколько во внедрении в практику разработки ответственных программных систем (СЗИ очевидный пример таких систем) технологий и процессов построения строгих спецификаций функциональных требований как на уровне внешних интерфейсов, так и на уровне отдельных подсистем и модулей. Если спецификации записываются на однозначно понимаемом языке (а еще лучше на машиночитаемом языке), то открывается возможность реализовать единообразный и инструментально поддержанный подход к анализу полноты и корректности результатов разработки.

Первые три части стандарта [2-4] сосредотачиваются не на вопросах тестирования, а на вопросах разработки и верификации формальных моделей, которые являются основой для тестирования СЗИ. В них рассматриваются вопросы строгого описания требований к отдельным функциям (действиям) СЗИ и спецификации интегральных требований обеспечения информационной безопасности, так называемых условий безопасности. Область действия четвертой части стандарта ГОСТ Р 59453.4-2025 покрывает задачи проверки того, что все требования к СЗИ, которые представлены в формальной модели управления доступом и формальной функциональной спецификации программных интерфейсов (ФСП) СЗИ, выполнены в полной мере и корректно.

[1] Синонимом MBT является сокращение SBT – Specification Based Testing, если речь идет о формальных спецификациях.

В стандарте рассмотрено два сценария тестирования. Первый сценарий предлагает тестировать объект оценки на системном уровне, то есть не погружаться на уровень отдельных модулей и подсистем, которые образуют ОС или СУБД. Например, в случае ОС интерфейсом системного уровня являются системные вызовы ее ядра. Такое тестирование целесообразно проводить на основе формальных спецификаций системных вызовов, то есть ФСП. При этом предварительно надо доказать, что спецификация системных вызовов соответствует требованиям ФМД.

Второй сценарий предполагает наличие возможности выделения и определения интерфейсов модуля информационной безопасности (в ОС Linux такой модуль обычно называется Linux Security Module – LSM) и тестирования этого модуля либо по схеме модульного тестирования (то есть в изолированной среде), либо по некоторой комбинированной схеме. В последнем случае либо тестовые воздействия подаются на модуль непосредственно от тестовой системы, минуя некоторые службы ядра ОС, либо результаты работы модуля передаются непосредственно в тестирующую систему и анализируются ею.

Использование второй схемы не отменяет использование первой. Второй сценарий, как и любое модульное тестирование, позволяет добиться более полного тестового покрытия, но технически он существенно сложнее первого, так как требуется создание еще одного слоя формальных спецификаций интерфейсов модуля безопасности и тестового окружения этого модуля. В рамках данной статьи мы остановимся только на первом сценарии, опыт, полученный в эксперименте по реализации второго сценария, описан авторами и их коллегами в [7], где рассмотрен подход к тестированию механизмов управления доступом ОС Astra Linux [8] на основе описания на языке спецификаций (нотации) Event-B [9, 10] мандатной сущностно-ролевой ДП-модели управления доступом и информационными потоками в ОС семейства Linux (МРОСЛ ДП-модели) [11].

В приложениях к стандарту приведен пример верификации системного вызова открытия файла `open` и рекомендации по оценке тестового покрытия на основе формальной спецификации системного вызова.

Пример представляет сначала описание операции `open` на уровне ФМД, потом приводится формальная спецификация системного вызова `open`. Обе формальные спецификации написаны на языке Event-B. Далее приводится пример таблицы, представляющей описание полученного тестового покрытия

3. Обзор работ в исследуемой области

Данную работу можно рассматривать как один из подходов к тестированию на основе формальных моделей для задач оценки и повышения степени защищенности программных систем, то есть мы говорим о Model Based Testing (MBT) в контексте тестирования безопасности (security testing). Этой теме посвящено много публикаций. Для сужения области рассмотрения уточним, что в качестве формальных моделей, которые мы используем для извлечения тестов, оценки результата их выполнения и полноты тестового покрытия, мы используем формальные модели безопасности, а еще точнее формальные модели управления доступом.

Анализу работ в этой области посвящен фундаментальный обзор М. Фельдерера и др. [12]. Авторы различают три вида моделей безопасности и, соответственно, три группы целей тестирования на основе таких моделей:

- Общие модели безопасности. Цель – тестирование базовых составляющих информационной безопасности, а именно, обеспечение конфиденциальности, целостности, доступности, аутентификации, авторизации или неотказуемости.
- Модели уязвимостей. Цель – поиск уязвимостей, приводящих к определенным классам нарушений безопасности.

- Модели безопасности или моделей управления доступом. Цель – тестирование механизмов безопасности.

В свете такой классификации описанный в данной работе подход соответствует третьему виду моделей и целей тестирования, а именно тестированию функций СЗИ.

К первому виду моделей можно отнести модели сценариев воздействия на СЗИ. Широко известны работы группы Ины Шифердеккер (Schieferdecker, Ina) из Fraunhofer FOKUS по моделям тестовых сценариев и их использованию для тестирования [13]. В основе этого подхода лежат язык TTCN-3 и UML Testing Profile [14].

Ко второму виду моделей можно отнести различного рода статические и динамические паттерны уязвимостей, которые часто приводят к нарушениям информационной безопасности.

Работ по формализации спецификаций моделей безопасности и их верификации в свое время было достаточно много. Однако большая часть работ рассматривала верификацию моделей безопасности в отрыве от процесса разработки и верификации реализации СЗИ, поэтому в них не уделялось должного внимания интеграции различных процессов жизненного цикла и выбора технологий, удобных для применения не в обстановке академических исследований, а в промышленных проектах.

Ближе всего к задаче описания полного цикла работ, связанных с разработкой и верификацией моделей безопасности и с процессами создания/генерации тестов на построенной модели, подошли авторы работы [15]. В качестве языка описания модели безопасности ими был выбран B-метод [16]. Основой для генерации тестов были паттерны тестовых последовательностей, которые, по сути, описывали темпоральные свойства тестовых сценариев. Из соображений упрощения технологии авторы использовали не темпоральные логики и нотации на их основе, а регулярные выражения, которые хорошо известны в практике программирования. Роль модели безопасности фактически была сведена к роли тестового оракула, при помощи которого тест мог автоматически детектировать отклонения тестируемой системы от требований к управлению доступом. Основным фокусом данной публикации являлась задача генерации тестовых последовательностей, в частности методам мутации тестовых сценариев. Соответственно, задачам интеграции технологий поддержки моделей, совместного использования артефактов на разных языках моделирования и программирования, что важно в промышленном применении таких методов, внимания не уделялось.

Важный методологический результат описывается в работе европейской команды [17]. В этой работе предлагается общий подход на основе идей Model Driven Engineering (MDE). Рамки подхода охватывают разработку моделей безопасности, координацию работ по проектированию системы и встраиваемых в нее функций и компонентов СЗИ, развертывание системы и ее тестирование. В качестве базовой технологи для представления политик безопасности предлагается использовать XACML [18]. Интеграция функционального программного обеспечения (ПО) и компонентов СЗИ обеспечивается при помощи технологий аспектно-ориентированного программирования. Описание собственно предложенной методологии представлено в форме метамодели, которая уточняется при развертывании целевого ПО на конкретной программно-аппаратной конфигурации. Конкретных техник моделирования, верификации моделей и генерации тестов в данной работе не описано, но работа интересна целостным взглядом на проблему создания и верификации СЗИ в индустриальном окружении с привлечением перспективных методов проектирования и реализации программных систем ответственного назначения.

Подводя итог краткому анализу методов и технологий создания и верификации промышленных СЗИ на основе формальных моделей безопасности или управления доступом можно заключить, что в доступной литературе хорошо проработаны вопросы решения

отдельных задач жизненного цикла, но пока нет достаточно эффективного способа бесшовной интеграции таких решений в единую технологическую цепочку. При этом помимо собственно задач интеграции в промышленном контексте важно найти компромисс, который обеспечивает эффективное решение отдельных задач и не требует сверхвысокой квалификации разработчиков и инженеров-верификаторов или владения сложными в освоении инструментами и технологиями.

Предложенная ниже схема верификации СЗИ разрабатывалась с целью обеспечить такой компромисс и при этом достичь достаточно высокой степени интеграции процессов и технологий в общей цепочке работ жизненного цикла разработки безопасного ПО.

4. Схема верификации СЗИ на основе формальной модели управления доступом

Общая схема верификации СЗИ на основе формальной модели управления доступом описана в монографии [19]. Если не рассматривать задачи по спецификации и верификации модуля безопасности ОС Linux (Linux Security Module - LSM), а ограничиться задачами

- разработки и верификации ФМД,
- построения ФСП, отвечающей требованиям модели, и верификации реализации СЗИ,
- проверки выполнения требований ФСП,

то схема верификации будет выглядеть так, как показано на рис. 1.

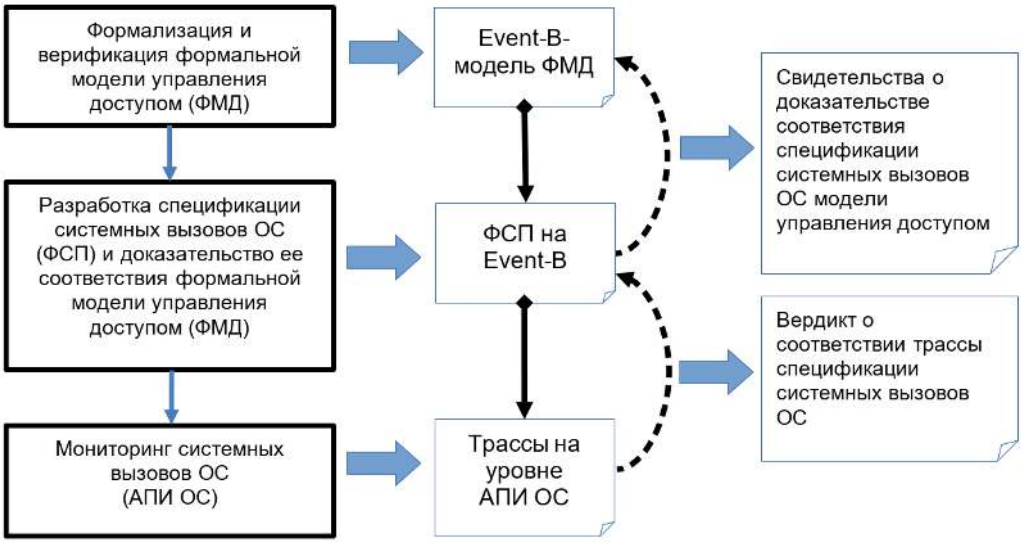

Рис. 1. Общая схема верификации СЗИ ОС на основе формальной модели управления доступом.
Fig. 1. General scheme of verification of OS security protection mechanism based on formal access control model.

Процесс начинается с разработки формальной модели управления доступом. В настоящее время известны две нотации, которые успешно используются для этой задачи: Event-B [9, 10] и TLA+ [20, 21]. В качестве примера спецификационной нотации во всех частях стандарта группы ГОСТ Р 59453 выбрана нотация Event-B и инструменты, которые с ней работают [22]. Это объясняется тем, что она базируется на привычной парадигме процедурного программирования и оснащена инструментами дедуктивной верификации в отличие от

TLA+, где парадигмой является темпоральная логика, а зрелые и удобные инструменты принадлежат к классу верификаторов моделей (model checker), что означает поддержку не строгого доказательства корректности, а лишь проверку перебором в рамках ограниченного числа экземпляров моделей поведения.

Вторая задача – это разработка и верификация формальной функциональной спецификации системных вызовов (ФСП). Здесь также используется нотация Event-B и инструменты дедуктивной верификации. В случае ФСП реальной альтернативой Event-B мог бы быть B-Method и инструменты, которые его поддерживают, но в России опыта их использования пока нет.

В данной статье мы не рассматриваем первую задачу верификации – как проверить, что формальная спецификация системных вызовов операционной системы отвечает требованиям формальной модели управления доступом. Хотя в монографии [19] приводится один из способов доказательства такого соответствия, применять его на практике затруднительно – создание эффективных методов и инструментов такой верификации пока еще не закончено. Тем не менее мы исходим из того, что ФСП соответствует модели и поэтому, если реализация СЗИ отвечает требованиям ФСП, то она соответствует и требованиям формальной модели управления доступом.

Третья задача – собственно динамическая верификация, которая выполняется либо в режиме пассивного тестирования, когда под реальной нагрузкой осуществляется мониторинг, либо в режиме выполнения специально подготовленного тестового набора с мониторингом. Мониторинг сводится к сбору трассы, в которой фиксируются данные о системных вызовах, связанных с работой СЗИ, и данные о результатах их выполнения. По завершении тестирования или в реальном времени выполняется анализ собранной трассы. Она должна соответствовать поведению СЗИ, представленному в спецификации ФСП, и в этом случае выносится положительный вердикт. В противном случае тест обнаруживает ошибку – отклонение реализации СЗИ от требований ФСП.

В ГОСТ Р 59453.4-2025 особое внимание уделяется контролю за полнотой тестового покрытия, где основной метрикой для оценки служат структуры формальной спецификации интерфейсов СЗИ. Рекомендации по оценке тестового покрытия приводятся в Приложении Б данного стандарта. Ниже приводится сокращенный вариант рекомендаций, который позволяет понять суть предлагаемого подхода.

> …Отдельный подход необходим, если условие *(в спецификации некоторой операции)* представляет собой дизъюнкцию из нескольких выражений-дизъюнктов. То же верно для импликации, при этом импликация может быть преобразована в дизъюнкцию по правилу $(A \Rightarrow B) \equiv (\neg A \vee B)$. В этом случае для создания ситуации, в которой условие принимает значение FALSE, необходимо, чтобы все входящие в дизъюнкцию выражения приняли значение FALSE. Для покрытия ситуаций, в которых полное условие принимает значение TRUE, рекомендуется создать, как минимум, набор ситуаций, в которых каждое отдельное входящее в дизъюнкцию выражение принимает значение TRUE, а остальные — FALSE. Если ситуация, в которой только одно из выражений выполнено, невозможна, рекомендуется создавать возможные ситуации, в которых выполнено минимальное множество выражений-дизъюнктов, включающее рассматриваемое. Эта рекомендация применяется, когда необходимо выполнение условия-дизъюнкции при выполнении остальных охранных условий. Если одно из других охранных условий нарушается, обеспечение значения TRUE для дизъюнкции не имеет особого значения. *(В приведенном ниже примере используется нотация Event-B. В квадратные скобки заключены метки-комментарии, которые нужны для ссылок на составные части условий, входящих в спецификации).*

Пример.

Допустим, мы пытаемся покрыть различные ситуации, связанные с работой операции получения доступа субъекта к объекту *GetAccess(subj, obj, akind)*. Пусть условие успешного выполнения этого события имеет следующий вид (числа в квадратных скобках нумеруют отдельные условия и дизъюнкты).

subj ∈ Subjects [1] / subj является субъектом доступа*/*

∧ obj ∈ Objects [2] / obj является объектом доступа*/*

∧ akind ∈ AccessKind [3] / akind является видом доступа */*

∧ subj ∈ ActiveSubjects [4] / пусть в системе есть субъекты, которые могут получать доступ к объектам, и есть неактивные, которые доступ получать не могут, но могут выполнять какие-то другие операции */*

∧ (subj = Admin [5.1] ∨ (obj, akind) ∈ AccessRights(subj) [5.2]) [5]

/ получение доступа успешно, либо если субъект является привилегированным (администратором), либо если у него есть право на указанный вид доступа к указанному объекту */*

Получение минимального покрывающего набора тестовых ситуаций в соответствии с представленными выше рекомендациями выполняется следующим образом.

Условия [1], [2], [3], по сути, представляют собой типовые ограничения на параметры и не могут быть нарушены при любой попытке исполнения данной операции. Они должны иметь значение TRUE.

Условия [4] и [5] могут быть нарушены. При этом условие [5] состоит из дизъюнктов [5.1] и [5.2], и его выполнение может быть обеспечено обращением в TRUE любого из этих дизъюнктов.

Рекомендуемый для создания в тестах минимальный набор ситуаций для данного примера такой:

1. Условия успешного исполнения операции выполнены, дизъюнкция [5] выполнена за счет дизъюнкта [5.1]

   subj ∈ Subjects
   ∧ obj ∈ Objects
   ∧ akind ∈ AccessKind
   ∧ subj ∈ ActiveSubjects
   ∧ (subj = Admin ∧ (obj, akind) ∉ AccessRights(subj) [нарушено 5.2])

2. Условия успешного исполнения операции выполнены, дизъюнкция [5] выполнена за счет дизъюнкта [5.2]

   subj ∈ Subjects
   ∧ obj ∈ Objects
   ∧ akind ∈ AccessKind
   ∧ subj ∈ ActiveSubjects
   ∧ (subj ≠ Admin [нарушено 5.1] ∧ (obj, akind) ∈ AccessRights(subj))

3. Условия успешного исполнения нарушены за счет нарушения условия [4]

   subj ∈ Subjects
   ∧ obj ∈ Objects
   ∧ akind ∈ AccessKind
   ∧ subj ∉ ActiveSubjects [нарушено 4]

$\wedge$ *(subj* $\neq$ *Admin [нарушено 5.1, выполнить его при нарушении 4 может быть невозможно]* $\wedge$ *(obj, akind)* $\in$ *AccessRights(subj) [выполнено 5.2])*

4. Условия успешного исполнения нарушены за счет нарушения условия [5]

   subj $\in$ *Subjects*
   $\wedge$ *obj* $\in$ *Objects*
   $\wedge$ *akind* $\in$ *AccessKind*
   $\wedge$ *subj* $\in$ *ActiveSubjects*
   $\wedge$ *(subj* $\neq$ *Admin* $\wedge$ *(obj, akind)* $\notin$ *AccessRights(subj)) [нарушены оба дизъюнкта 5.1 и 5.2]*

5. Особенности динамической верификации промышленных средств защиты информации

Опыт разработки инструмента динамической верификации СЗИ операционных систем АНИС [23] помог обнаружить некоторые подводные камни ее применения к такой сложной СЗИ, как ОС семейства Linux. Чтобы остановиться на вопросах, которые потребовали серьезного внимания, предварительно опишем общую схему работ по верификации СЗИ при помощи АНИС.

В соответствии со стандартами ГОСТ Р 59453 сначала разрабатывается формальная модель управления доступом – ФМД, потом спецификация СЗИ – ФСП. Разработка и верификация модели и спецификации может проводиться, например, при помощи инструмента дедуктивной верификации Rodin [22]. Далее инженер-верификатор переходит к собственно задачам тестирования, и, если модель и спецификация разрабатывается не с нуля, то инженеру-верификатору на этой фазе жизненного цикла удобнее работать не с инструментом дедуктивной верификации, а сразу с инструментом, который предназначен для разработки и отладки тестов. То есть функциональность инструмента для разработки тестов на основе формальной модели должна позволять редактировать, проводить легковесную проверку ФСП и, возможно, даже отлаживать ФСП.

Динамическая верификация на основе формальной модели накладывает некоторые ограничения на язык и стиль спецификации. Так использование кванторных выражений или имплицитных деклараций либо требует существенно больших ресурсов памяти и времени верификации, либо вообще делает невозможным точную интерпретацию модели в ходе тестирования. Это особенно важно, когда мониторинг выполняется под рабочей нагрузкой системы. По этой причине, во-первых, следует четко очертить подмножество языка, на котором разрабатывается спецификация, и, во-вторых, реализовать в инструменте функцию проверки того, что ФСП отвечает требованиям данного подмножества.

Далее важно отметить, что при тестировании промышленных систем на основе формальных спецификаций практически всегда приходится работать со спецификациями разного уровня абстракции (детализации), спецификация интерфейсов системы или модулей реализации также по уровню абстракции отличается от спецификаций, написанных на языках спецификации. В случае тестирования СЗИ в ОС для сопоставления требований из спецификации ФСП с результатами выполнения системных вызовов, которые представляются данными в формате, описанном на языке Си, необходимо создать слой адаптеров. Кроме того, в задачу сопоставления входит и анализ корректности последовательностей событий на уровне модели и событий на уровне системных вызовов, что тоже является нетривиальной задачей. Более подробная схема реализации и сценарии использования АНИС представлена в статье [23].

7. Заключение

В работе представлен новый стандарт, в котором описаны рекомендации по верификации СЗИ и метод тестирования на основе формальных моделей управления доступом. Данный подход позволяет одновременно решить несколько задач важных для повышения уровня доверия к СЗИ, в которых управление доступом является неотъемлемой частью функциональности объекта оценки. Важными примерами таких систем являются ОС и СУБД.

Тестирование на основе формальных моделей отличается от других подходов тем, что оно одновременно позволяет:

- опираться на строгие спецификации требований информационной безопасности;
- обеспечивать высокую степень масштабируемости вплоть до систем размером в десятки миллионов строк кода;
- обеспечивать объективную базу для оценки полноты тестирования как на основе спецификации требований, так и на основе структуры реализации;
- давать основу для трассируемости всех артефактов процессов разработки и тестирования на основе единой системы спецификационных сущностей;
- интегрировать на единой концептуальной и технологической базе различные техники верификации, включая дедуктивную верификацию, верификацию моделей (model checking) и тестирование, в том числе тестирование на основе традиционных, привычных инженерам методов.

Опыт разработки и общественного обсуждения нового стандарта позволили обобщить требования к инструментам для динамической верификации СЗИ промышленных программных систем. В частности, важной возможностью таких инструментов должны быть средства, которые позволяют не только генерировать и выполнять тесты, но и работать с формальными моделями и спецификациями разных уровней абстракции, поддерживать сбор и анализ трассовой информации и соотносить ее с требованиями спецификаций, уровень абстракции которых может существенно отличаться от собранной трассовой информации. Кроме того, изложенные в стандарте рекомендации прошли апробацию при тестировании на соответствии МРОСЛ ДП-модели механизмов управления доступом отечественной ОС Astra Linux.

Работа над группой стандартов ГОСТ Р 59453 будет продолжена, соответственно будут продолжены работы по расширению инструментальной поддержки процессов и технологий, которые рекомендуются данными стандартами. В направлении динамической верификации СЗИ инструмент АНИС будет развиваться для обеспечения возможности мониторинга работы ОС под рабочей нагрузкой. Сложной задачей является верификация ФСП на соответствие требованиям ФМД. Как уже отмечалось в монографии [19] приводится один из способов доказательства такого соответствия, но применять его на практике затруднительно. В настоящее время разрабатывается альтернативный вариант техники такой верификации. Запланирована разработка соответствующего стандарта ГОСТ Р 59453. Часть 5. Также ведутся работы по созданию инструментальной поддержки предлагаемой техники. Еще одним направлением работ по верификации СЗИ на основе формальных спецификаций является создание набора инструментов на основе широко известных фреймворков тестирования, например, PyTest. Это позволит построить удобную комбинацию привычных средств разработки тестов и системного подхода к анализу полноты тестов на основе строгих функциональных требований ФМД и ФСП.

Благодарности. Авторы выражают благодарность всем, принявшем участие в обсуждении и улучшении редакций разработанных стандартов. Особый вклад в работу внесла Ольга Валентиновна Кулагина ФАУ «ГНИИИ ПТЗИ ФСТЭК России».

Список литературы / References

[1]. ГОСТ Р 59453.4-2025 «Защита информации. Формальная модель управления доступом. Часть 4. Рекомендации по верификации средства защиты информации, реализующего политики управления доступом, на основе формализованных описаний модели управления доступом». М.: Стандартинформ, 2025. 20 с. / GOST R 59453.4-2025 «Information protection. Formal access control model. Part 4. Recommendations for verification of information security features that implement access control policies based on formal descriptions of the access control model», 2025 (in Russian).

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

[3]. ГОСТ Р 59453.2-2021 «Защита информации. Формальная модель управления доступом. Часть 2. Рекомендации по верификация формальной модели управления доступом». М.: Стандартинформ, 2021. 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).

[4]. ГОСТ Р 59453.3-2025 «Защита информации. Формальная модель управления доступом. Часть 3. Рекомендации по верификации средства защиты информации, реализующего политики управления доступом, на основе формализованных описаний модели управления доступом». М.: Стандартинформ, 2025. 20 с. / GOST R 59453.3-2025 «Information protection. Formal access control model. Part 3. Recommendations on development», 2025 (in Russian).

[5]. DO-178C - Software Considerations in Airborne Systems and Equipment Certification Software Considerations in Airborne Systems and Equipment Certification (RTCA DO178C), 2011 https://my.rtca.org/nc__store?search=do-178c

[6]. КТ-178С – Квалификационные требования. Часть 178 «Требования к программному обеспечению бортовой аппаратуры и систем при сертификации авиационной техники».

[7]. П.Н. Девянин, С.С. Жиляков, А.И. Смирнов. Тестирование подсистемы безопасности ОС Astra Linux на основе формализованного описания модели управления доступом, том 37, вып. 4, 2025. / Devyanin P.N., S.S. Zhiliakov, A.I. Smirnov. Testing the Astra Linux OS security subsystem based on a formalized description of the access control model. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 4, 2025 (in Russian).

[8]. Операционная система специального назначения Astra Linux Special Edition. Доступно по ссылке: https://astra.ru/software-services/os/, 17.04.2025. / Astra Linux Special Edition operating system. Available at: https://astra.ru/software-services/os/, accessed 17.04.2025.

[9]. Abrial, Jean-Raymond. Modeling in Event-B: system and software engineering.- Cambridge University Press, 2010.

[10]. 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).

[11]. 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, 2020, 352 p. (in Russian).

[12]. Michael Felderer, Philipp Zech, Ruth Breu, Matthias Büchler, Alexander Pretschner. Model-Based Security Testing: A Taxonomy and Systematic Classification.- Software Testing Verification and Reliability 00(2), July 2015:1-29, DOI:10.1002/stvr.1580.

[13]. Schieferdecker, Ina & Großmann, Jürgen & Schneider, Martin. (2012). Model-Based Security Testing. Electronic Proceedings in Theoretical Computer Science. A.K. Petrenko, H. Schlingloff (Eds.): Workshop on Model-Based Testing 2012 (MBT 2012), EPTCS 80, 2012, https://doi.org/10.4204/EPTCS.80

[14]. Krishnan, P., Pari-Salas, P. (2009). Model-Based Testing and the UML Testing Profile. In: Palsberg, J. (eds) Semantics and Algebraic Specification. Lecture Notes in Computer Science, vol 5700. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-04164-8_16/

[15]. Pierre-Alain Masson, Jacques Julliand, Jean-Chritophe Plessis, Eddie Jaffuel, Georges Debois. Automatic generation of model based tests for a class of security properties.- A-MOST '07: Proceedings of the 3rd

international workshop on Advances in model-based testing, Pages 12-22, https://doi.org/10.1145/1291535.1291537

[16]. The B-Book: Assigning Programs to Meanings, Jean-Raymond Abrial, Cambridge University Press, 1996. ISBN 0-521-49619-5.

[17]. Tejeddine Mouelhi, Franck Fleurey, Benoit Baudry, Yves Le Traon. A Model-Based Framework for Security Policy Specification, Deployment and Testing.- K. Czarnecki et al. (Eds.): MoDELS 2008, LNCS 5301, pp. 537–552

[18]. OASIS eXtensible Access Control Markup Language (XACML) TC | OASIS. www.oasis-open.org.

[19]. 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).

[20]. L. Lamport. Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley, 2002.

[21]. Козачок В.И., Козачок А.В., Кочетков Е.В. Многоуровневая модель политики безопасности управления доступом операционных систем семейства Windows.- Вопросы кибербезопасности, 2021, № 1(41), DOI: 10.21681/2311-3456-2021-1-41-56.

[22]. Event-B and the Rodin Platform, https://www.event-b.org/.

[23]. Е.В. Корныхин, Д.В. Ефремов, А.В. Карнов, А.К. Петренко. Инструмент АНИС - поддержка процессов верификации средств защиты информации на основе формальных моделей управления доступом.- Труды ИСП РАН, 2025, том 37, вып. 4.

Информация об авторах / Information about authors

Александр Константинович ПЕТРЕНКО – профессор, доктор физико-математических наук, заведующий отделом Технологий программирования ИСП РАН, профессор кафедр Системного программирования ВМК МГУ и ФКН НИУ ВШЭ. Его научные интересы включают формальные методы программной инженерии, языки спецификаций и моделирования, их применение для поддержки разработки и верификации программного обеспечения.

Alexander Konstantinovich PETRENKO – Dr. Sci. (Phys.-Math.), Prof., Head of Software Engineering Department of the Ivannikov Institute for System Programming, Russian Academy of Sciences, Professor of the Department of System Programming, Faculty of Computational Mathematics and Cybernetics, Moscow State University and the Faculty of Computer Science, National Research University Higher School of Economics. His research interests include formal methods of software engineering, specification and modeling languages, and their use in software development and verification.

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.

Денис Валентинович ЕФРЕМОВ – научный сотрудник. Сфера научных интересов: формальная верификация, статический и динамический анализ.

Denis Valentinovich EFREMOV – researcher. Research interests: formal verification, static and dynamic analysis.

Алексей Александрович КАРНОВ – научный сотрудник. Научные интересы: формальные спецификации, верификация и тестирование, статический и динамический анализ.

Aleksei Aleksandrovich KARNOV – researcher. Research interests: formal specifications, verification and testing, static and dynamic analysis.

Евгений Валерьевич КОРНЫХИН – кандидат физико-математических наук, доцент кафедры системного программирования МГУ, старший научный сотрудник ИСП РАН. Область интересов: формальная дедуктивная верификация моделей, тестирование на основе моделей.

Eugeny Valerievich KORNYKHIN – Cand. Sci. (Phys.-Math.), Associate Professor of system programming departments at MSU and Senior Researcher at ISP RAS. Fields of Interest: formal deductive verification, model-based testing.

Виктор Вячеславович КУЛЯМИН – кандидат физико-математических наук, ведущий научный сотрудник ИСП РАН, доцент кафедр системного программирования МГУ и ВШЭ. Область интересов: формальная дедуктивная верификация моделей, тестирование на основе моделей.

Viktor Vyacheslavovich KULIAMIN – Cand. Sci. (Phys.-Math.), leading researcher at ISP RAS, associate professor of system programming departments at MSU and the HSE. Fields of Interest: formal deductive model verification, model-based testing.

Алексей Владимирович ХОРОШИЛОВ – кандидат физико-математических наук, ведущий научный сотрудник, директор Центра верификации ОС Linux в ИСП РАН, доцент кафедр системного программирования МГУ, ВШЭ и МФТИ. Основные научные интересы: методы проектирования и разработки ответственных систем, формальные методы программной инженерии, методы верификации и валидации, тестирование на основе моделей, методы анализа требований, операционная система Linux.

Alexey Vladimirovich KHOROSHILOV – Cand. Sci. (Phys.-Math.), Leading Researcher, Director of the Linux OS Verification Center at ISP RAS, Associate Professor of System Programming Departments at MSU, HSE, and MIPT. Main research interests: design and development methods for critical systems, formal methods of software engineering, verification and validation methods, model-based testing, requirements analysis methods, Linux operating system.