"ГОСТ Р 59453.4-2025. Национальный стандарт Российской Федерации. 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"
(утв. и введен в действие Приказом Росстандарта от 10.03.2025 N 112-ст)

Утвержден и введен в действие
Приказом Федерального
агентства по техническому
регулированию и метрологии
от 10 марта 2025 г. N 112-ст

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

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

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

ГОСТ Р 59453.4-2025

ОКС 35.030

Дата введения
31 марта 2025 года

Предисловие

1 РАЗРАБОТАН 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 "РусБИТехАстра"), Федеральным автономным учреждением "Государственный научно-исследовательский испытательный институт проблем технической защиты информации Федеральной службы по техническому и экспортному контролю" (ФАУ "ГНИИИ ПТЗИ ФСТЭК России")

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 362 "Защита информации"

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 10 марта 2025 г. N 112-ст

4 ВВЕДЕН ВПЕРВЫЕ

Правила применения настоящего стандарта установлены в статье 26 Федерального

закона от 29 июня 2015 г. N 162-ФЗ "О стандартизации в Российской Федерации". Информация об изменениях к настоящему стандарту публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе "Национальные стандарты", а официальный текст изменений и поправок - в ежемесячном информационном указателе "Национальные стандарты". В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ближайшем выпуске ежемесячного информационного указателя "Национальные стандарты". Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования - на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (www.rst.gov.ru)

Введение

Проектирование и разработка средств защиты информации является сложной инженерно-технической задачей. По этой причине для обеспечения надежности и корректности функционирования средств защиты информации, так же как в других областях конструкторской деятельности, необходимо привлекать техники моделирования и исследования как самих моделей, так и продуктов, программных решений.

В ГОСТ Р 59453.1-2021, ГОСТ Р 59453.2-2021 и ГОСТ Р 59453.3-2025 приводятся рекомендации по разработке моделей управления доступом и по процессам верификации этих моделей. Корректная модель управления доступом является первым звеном в цепи работ по верификации средств защиты информации в целом. В связи с этим настоящий стандарт устанавливает рекомендации по организации работ по верификации средств защиты информации, реализующих политики управления доступом, а также по доказательству или демонстрации того, что исследуемое средство защиты информации в действительности реализует формальную модель управления доступом.

Поскольку схемы реализации средств защиты информации в различных системах существенно различаются и не могут быть сведены в единый архитектурный шаблон, в данном стандарте рекомендации даются в общем виде. Способов конкретизации требований стандарта может быть несколько. В каждом конкретном случае разработчик должен выбрать наиболее подходящие техники моделирования и верификации средства защиты информации. Необходимым требованием является лишь выполнение общих рекомендаций, которые содержит данный стандарт.

1 Область применения

Настоящий стандарт устанавливает рекомендации по верификации средств защиты информации, реализующих политики управления доступом, на основе формализованного описания модели управления доступом.

Настоящий стандарт предназначен для разработчиков средств защиты информации, реализующих политики управления доступом, а также для органов по сертификации и испытательных лабораторий при проведении сертификации средств защиты информации, реализующих политики управления доступом.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты:

ГОСТ Р 59453.1 Защита информации. Формальная модель управления доступом. Часть 1. Общие положения

ГОСТ Р 59453.2 Защита информации. Формальная модель управления доступом. Часть 2. Рекомендации по верификации формальной модели управления доступом

ГОСТ Р 59453.3 Защита информации. Формальная модель управления доступом. Часть 3. Рекомендации по разработке

Примечание - При пользовании настоящим стандартом целесообразно проверить действие ссылочных стандартов в информационной системе общего пользования - на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет или по ежегодному информационному указателю "Национальные стандарты", который опубликован по состоянию на 1 января текущего года, и по выпускам ежемесячного информационного указателя "Национальные стандарты" за текущий год. Если заменен ссылочный стандарт, на который дана недатированная ссылка, то рекомендуется использовать действующую версию этого стандарта с учетом всех внесенных в данную версию изменений. Если заменен ссылочный стандарт, на который дана датированная ссылка, то рекомендуется использовать версию этого стандарта с указанным выше годом утверждения (принятия). Если после утверждения настоящего стандарта в ссылочный стандарт, на который дана датированная ссылка, внесено изменение, затрагивающее положение, на которое дана ссылка, то это положение рекомендуется применять без учета данного изменения. Если ссылочный стандарт отменен без замены, то положение, в котором дана ссылка на него, рекомендуется применять в части, не затрагивающей эту ссылку.

3 Термины и определения

В настоящем стандарте применены следующие термины с соответствующими определениями:

3.1

верификация формальной модели управления доступом: Подтверждение посредством представления объективных свидетельств непротиворечивости формальной модели управления доступом и выполнения заданных в ее рамках условий безопасности.

[ГОСТ Р 59453.2-2021, пункт 3.1]

3.2 **динамическая верификация:** Вид тестирования на основе формальных моделей поведения систем.

Примечание - Результатом динамической верификации в случае верификации средства защиты информации являются данные, полученные на основе сопоставления входов, выходов, наблюдаемого поведения средства защиты информации в целом с ее формальной моделью, имеющей на входе соответствующие воздействия. Автоматический анализ поведения средства защиты информации, таким образом, сводится к построению отображения ее входов и выходов в интерфейсы модели и последующему анализу (или анимации) модели с полученными входными воздействиями и получению вердикта о том, нарушают ли полученные данные о поведении средства защиты информации требования модели или нет.

3.3 **модульная верификация;** *верификация на модульном уровне*: Верификация программного модуля или группы модулей методом погружения их в тестовое или модельное окружение, которое моделирует воздействия и реакции на обращения верифицируемого модуля.

3.4 **модуль (обеспечения) безопасности:** Программный модуль на некотором языке программирования, реализующий политики управления доступом в средстве защиты информации и имеющий явно заданный программный интерфейс.

Примечание - Часть функций по реализации политик управления доступом в средстве защиты информации может быть не локализована в виде программного модуля. В этом случае модуль безопасности выполняет не все функции по реализации таких политик, а только часть.

3.5 **статическая верификация:** Техники анализа программ, при которых проверка корректности не требует исполнения программы.

3.6

формализованное (машиночитаемое) описание: Описание формальной модели на формальном языке со строгой и однозначно определенной семантикой, позволяющее использовать инструментальные средства верификации.
[ГОСТ Р 59453.2-2021, пункт 3.3]

3.7

формальная модель управления доступом: Математическое или формализованное (машиночитаемое, пригодное для автоматизированной обработки) описание средства защиты информации и компонентов среды его функционирования, предоставление доступов между которыми регламентируется политиками управления доступом, реализуемыми этим средством защиты информации.
[ГОСТ Р 59453.1-2021, пункт 3.21]

3.8

формальный метод: Основанный на математике и логике метод, а также поддерживаемые ими языки, для верификации и разработки формальных моделей.
[ГОСТ Р 59453.2-2021, пункт 3.4]

3.9 **функциональная спецификация средства защиты информации:** Описание, детализирующее внешний интерфейс средства защиты информации.

4 Общие положения

4.1 Рекомендации настоящего стандарта по верификации средств защиты информации, реализующих политики управления доступом на основе формализованного описания модели управления доступом, направлены на обеспечение соответствия функционирования средства защиты информации формальной модели управления доступом, разработанной и

верифицированной в соответствии с критериями и рекомендациями ГОСТ Р 59453.1, ГОСТ Р 59453.2 и ГОСТ Р 59453.3.

4.2 Для верификации средств защиты информации, реализующих политики управления доступом, на основе формализованного описания модели управления доступом в качестве основной техники верификации предлагается динамическая верификация, частный случай тестирования на основе формальных моделей.

4.3 Статические методы верификации могут применяться для верификации средств защиты информации в случае, когда размер и сложность программного обеспечения средства защиты информации позволяют применить указанные методы верификации.

4.4 Динамическую верификацию средства защиты информации можно проводить на системном и на модульном уровне. Верификация на системном уровне обязательна. Верификация на модульном уровне может выполняться при наличии в средстве защиты информации выделенного модуля безопасности и технической возможности проведения модульного тестирования.

4.5 Рекомендуются следующие этапы верификации средства защиты информации, реализующего политики управления доступом, на основе формализованного описания модели управления доступом:

- проведение архитектурного анализа средства защиты информации и исследование наличия модуля безопасности или группы функций, которые могут рассматриваться как модуль безопасности (возможность определения интерфейсов взаимодействия такого модуля с его окружением);

- принятие решения о проведении модульной верификации или отказе от нее;

- выбор языков разработки формальных спецификации и инструментов разработки и верификации;

- выбор критериев и методики оценки полноты верификации (оценки верификационного покрытия).

Примечания

1 Критерии и методики оценки полноты верификации (оценки верификационного покрытия) разрабатываются и обосновываются разработчиками и специалистами по верификации с учетом технической возможности выбранных средств моделирования и верификации и с учетом общепринятой практики, описанной в приложении А.

2 Важно обращать внимание на способы обеспечения интеграции работ в ходе верификации, то есть способы организации переиспользования промежуточных результатов, созданных или полученных в ходе использования инструментов верификации. При выборе инструментов верификации нужно обращать внимание на наличие средств сбора и обработки информации о полноте покрытия;

- разработка формальной спецификации интерфейсов средства защиты информации, реализующих политики управления доступом, и ее верификация.

Примечание - Формальная спецификация разрабатывается с учетом требований формальной модели управления доступом, которая разрабатывается и верифицируется до данного этапа;

- верификация средства защиты информации, проверка соответствия поведения средства защиты информации формальной модели управления доступом.

Примечание - Вместо демонстрации соответствия поведения средства защиты информации модели управления доступом можно демонстрировать соответствие формальной спецификации средства защиты информации, если предварительно было доказано, что все требования безопасности, представленные в формальной модели управления доступом, аналогичным образом представлены и в формальной спецификации средства защиты информации;

- разработка формальной спецификации интерфейсов модуля безопасности (опционально, если ставится задача верификации модуля безопасности) и ее верификация (опционально);

- верификация модуля безопасности (опционально, если ставится задача верификации модуля безопасности).

4.6 В описании результатов верификации должны быть представлены:

- анализ архитектуры средства защиты информации, обоснование вывода о наличии или отсутствии возможности модульной верификации модуля безопасности;

- описание формальной спецификации интерфейсов средства защиты информации;

- сопоставление структуры формальной модели управления доступом и формальной спецификации средства защиты информации;

- обоснование выбора способа демонстрации того, что формальная спецификация соответствует формальной модели управления доступом;

- результаты проведения верификации в соответствии с рекомендациями разделов 6 и 7 настоящего стандарта.

5 Выбор инструментов верификации

При выборе инструментов для проведения верификации средства защиты информации, реализующего политики управления доступом, на основе формализованных описаний модели управления доступом следует руководствоваться следующими рекомендациями:

а) инструменты должны поддерживать статическую верификацию или динамическую верификацию;

б) в инструментах должна быть предусмотрена возможность использовать выбранные языки моделирования для описания формальной модели управления доступом и/или формальной спецификации интерфейсов средства защиты информации;

в) инструментами должна поддерживаться техника уточнения для перевода реализационного представления данных (определенного в терминах используемого языка

программирования) в их модельное представление. В случае динамической верификации должен быть выделен слой адаптеров/медиаторов для конвертации данных;

г) в инструментах статической верификации должны быть средства анализа верификационного покрытия;

д) в инструментах динамической верификации должны быть средства для сбора и анализа тестового (верификационного) покрытия:

1) на основе структур формальной модели управления доступом и/или формальной спецификации средства защиты информации;

2) на основе структуры реализации средства защиты информации.

Примечания

1 Формальные модели управления доступом, формальные спецификации средства защиты информации и формальные спецификации модулей обеспечения безопасности могут создаваться при помощи различных инструментов, языков моделирования, языков формальных спецификаций или других формальных нотаций.

2 Для целей моделирования и верификации моделей управления доступом большое распространение получили языки (нотации) Event-B и TLA+. Первая поддерживается несколькими инструментами, наиболее известные Rodin и ProB. Наиболее известным набором инструментов для TLA+ является набор TLA+ Toolbox. Перечисленные инструменты распространяются под открытыми лицензиями.

3 Распространенных инструментов, которые разрабатывались специально для тестирования на основе моделей, мало. Большая часть из них плохо интегрируется с языками программирования, на которых разрабатываются средства защиты информации, и с языками, на которых описываются модели и спецификации. Однако для указанных выше нотаций такие инструменты есть. Пример для нотации Event-B приводится в приложении Б.

6 Рекомендации по верификации средства защиты информации, реализующего политики управления доступом, на основе формализованных описаний модели управления доступом на системном уровне

6.1 Для верификации средства защиты информации рекомендуется проведение динамической верификации на соответствие формальной модели управления доступом или формальной спецификации этого средства защиты информации.

Примечание - Верификация на системном уровне состоит в проверке соответствия поведения средства защиты информации требованиям модели управления доступом на уровне его внешних интерфейсов, то есть при этом средство защиты информации анализируется как система, и не анализируются его внутренние процессы и функциональность на уровне межмодульных связей.

6.2 Проведение верификации средства защиты информации начинается с анализа соответствия интерфейсов модели управления доступом и интерфейсов средства защиты информации. В случае, когда интерфейсы по структуре совпадают или близки, верификацию

средства защиты информации можно проводить, проверяя соответствие поведения средства защиты информации формальной модели управления доступом. В случае, когда структуры интерфейсов различны, необходимо построить формальную спецификацию средства защиты информации и доказать (или продемонстрировать другим способом), что формальная спецификация средства защиты информации соответствует формальной модели управления доступом.

6.3 В случае наличия прямого соответствия между структурами интерфейсов формальной модели управления доступом и формальной спецификации средства защиты информации последняя может быть разработана как прямое уточнение формальной модели управления доступом.

6.4 Если интерфейс средства защиты информации не находится в прямом соответствии с интерфейсом формальной модели управления доступом, для установления соответствия между ними следует применять техники, доступные для разработчиков, включая формальную верификацию (верификацию, выполняемую при помощи формальных методов) или анализ вручную.

Примечание - Наиболее высокий уровень доверия к верификации соответствия формальной модели управления доступом и формальной спецификации средства защиты информации дает формальная верификация, которую можно выполнять при помощи процедуры установления уточнения по состояниям. Эта процедура включает построение так называемого абстрагирующего отображения состояний формальной спецификации средства защиты информации (СЗИ) в состояния формальной модели управления доступом. При этом выполнение условий безопасности в формальной спецификации СЗИ в некотором ее состоянии влечет выполнение условий безопасности формальной модели управления доступом в состоянии, полученном в результате применения абстрагирующего отображения к этому состоянию формальной спецификации СЗИ.

6.5 При проведении динамической верификации средства защиты информации, реализующего политики управления доступом, необходимо оценивать полноту верификации на соответствие формальной спецификации средства защиты информации с отслеживанием покрытия модели (см. приложение А). Если формальная спецификация средства защиты информации не является прямым уточнением формальной модели управления доступом, покрытие формальной модели управления доступом должно отслеживаться отдельно при помощи отображения отдельных событий или цепочек событий из формальной спецификации средства защиты информации в события формальной модели управления доступом.

6.6 В описании результатов верификации должны быть представлены:

- описание формальной спецификации средства защиты информации;

- обоснование того, что эта формальная спецификация соответствует формальной модели управления доступом;

- описание тестового (модельного) окружения и процесса верификации и обоснование выбора этого окружения;

- оценка полноты верификации на основе структуры формальной модели управления доступом и формальной спецификации средства защиты информации.

7 Рекомендации по верификации средства защиты информации, реализующего политики управления доступом, на основе формализованных описаний модели управления доступом на уровне интерфейсов модулей безопасности

7.1 Рекомендации данного раздела относятся к случаю, когда в средстве защиты информации явно выделен модуль безопасности и должна быть проведена его верификация. Такая верификация может проводиться, если есть техническая возможность отделить модуль безопасности от остальных составляющих средства защиты информации и организовать его модульное тестирование.

7.2 Верификация модуля безопасности в средстве защиты, реализующем политики управления доступа, должна основываться на формальной модели управления доступом. Для верификации модуля безопасности сначала необходимо разработать формальную спецификацию интерфейсов модуля.

Примечания

1 Разработка такой спецификации и проверка ее соответствия формальной модели управления доступом и формальной спецификации средства защиты информации в целом во многих случаях является сложной научно-технической задачей. В ряде случаев может быть использован подход на основе технической экспертизы, предполагающей последовательное построение формальной спецификации модуля с многократным и итеративным проведением ее обзоров (инспекций) несколькими специалистами.

2 Ситуация усложняется в тех случаях, когда модуль безопасности строится в виде обобщенного интерпретатора возможных политик, описываемых на некотором структурированном языке и поставляемых в виде отдельных конфигурационных файлов системы защиты информации. Примером такой реализации является модуль LSM SELinux. В этом случае нет какого-либо прямого соответствия между функциональностью модуля самого по себе и моделью управления доступом в целом. В этой ситуации верификация проводится для каждой политики безопасности или для каждого класса политик безопасности.

7.3 Оценку полноты верификации модуля безопасности следует проводить при помощи анализа покрытия модели (см. приложение А).

7.4 Кроме того, необходимо отслеживание покрытия кода модуля получаемыми тестами (не менее уровня покрытия всех ветвлений в программе). Непокрытые ветвления в программе должны анализироваться на предмет возможного существенного влияния на работу модуля в целом, при подтверждении этого влияния должны создаваться дополнительные тесты для покрытия таких ветвлений.

7.5 При проведении верификации конфигурируемого модуля безопасности необходимо использовать различные конфигурации и достигать покрытия структурных элементов языка описания конфигураций (правил и отдельных альтернатив грамматики, отдельных возможных операторов, используемых при описании правил политик, а также возможных сочетаний пар альтернатив в одном правиле).

Примечание - В случае сложных средств защиты информации (например, операционных

систем или систем управления базами данных) возможно проведение частичной верификации конфигурируемого модуля, при которой покрытие языка описания конфигураций не достигается, но обеспечивается достаточный анализ ситуаций, возникающих при изменении лишь небольшой части атрибутов конфигурации. В этом случае корректное применение средства защиты информации будет верифицировано только для тех случаев, когда изменения конфигурации/политик безопасности остаются в рамках покрытого тестами множества наборов значений ее атрибутов (в пределе, только при использовании ровно той же конфигурации, для которой была проведена верификация).

7.6 В описании результатов верификации модуля безопасности должны быть представлены:

- описание формальной спецификации модуля безопасности, обоснование того, что она соответствует формальной модели управления доступом и формальной спецификации средства защиты информации;

- описание тестового (модельного) окружения и процесса верификации и обоснование выбора этого окружения;

- оценка полноты верификации на основе структуры формальной модели управления доступом, формальной спецификации средства защиты информации и структуры реализации модуля безопасности.

Приложение А
(справочное)

РЕКОМЕНДАЦИИ ПО ОЦЕНКЕ ПОКРЫТИЯ ФОРМАЛЬНЫХ МОДЕЛЕЙ

Критерии покрытия формальных моделей, используемые при тестировании на соответствие им, должны использовать структурные элементы спецификации операций/событий модели в качестве основы.

Обычно операция/событие имеет некоторый набор условий ее успешного выполнения, называемый предусловием или набором охранных условий. При этом некоторые условия из этого набора обеспечивают саму возможность исполнения операции, а другие - обеспечивают успешность ее исполнения при соблюдении правил и условий безопасности. При нарушении условий первого типа операция не может быть исполнена вообще. Операция может быть исполнена при нарушении условий второго типа, но ее исполнение приводит к возвращению специализированного кода ошибки или созданию ситуации, в которой фиксируется нарушение правил.

Для исполнения операции условия первого типа всегда должны быть выполнены, поэтому они не учитываются при определении полноты тестирования. Условия второго типа могут быть нарушены, при этом нарушение каждого из них должно приводить к неуспешному завершению операции. Поэтому для полноты тестирования необходимо обеспечить в тестах ситуацию успешного исполнения, в которой все условия второго типа выполнены, а также ситуации, в

которых эти условия нарушены. Рекомендуется создавать, как минимум, набор ситуаций, в которых только одно из этих условий нарушено, а остальные - выполнены. Это обеспечит при тестировании верификацию того, что рассматриваемые условия влияют на успешность выполнения операции независимо.

Иногда ситуация, в которой одно из условий второго типа нарушено, а остальные выполнены, невозможна. В этих случаях рекомендуется создавать возможные ситуации, в которых нарушено минимальное множество условий второго типа, включающее рассматриваемое условие.

Отдельный подход необходим, если условие 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.

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).

1	*subj ∈ Subjects /* subj является субъектом доступа*/*
2	*∧ obj ∈ Objects /* obj является объектом доступа*/*
3	*∧ akind ∈ AccessKind /* akind является видом доступа */*
4	*∧ subj ∈ ActiveSubjects /* пусть в системе есть активные субъекты, которые могут получать доступ к объектам, и есть неактивные субъекты, которые доступ получать не могут, но могут выполнять какие-то другие операции */*
5	*/* получение доступа успешно, либо если субъект является привилегированным (администратором), либо если у него есть право на указанный вид доступа к указанному объекту */*
5.1	*(subj = Admin*

5.2	*∧ (obj, akind) ∈ AccessRights(subj))*

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 являются условиями первого типа для данного примера и во всех ситуациях должны иметь значение TRUE.

Условия 4 и 5 являются в этом примере условиями второго типа и 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*/

Приложение Б
(справочное)

ПРИМЕР ПРОВЕДЕНИЯ ВЕРИФИКАЦИИ СРЕДСТВА ЗАЩИТЫ ИНФОРМАЦИИ, РЕАЛИЗУЮЩЕГО ПОЛИТИКИ УПРАВЛЕНИЯ ДОСТУПОМ, НА ОСНОВЕ ФОРМАЛИЗОВАННЫХ ОПИСАНИЙ МОДЕЛИ УПРАВЛЕНИЯ ДОСТУПОМ

Рассматривается пример верификации средства защиты информации файловой системы в рамках типовой операционной системы. В ней есть операция open. В реальных файловых системах open выполняет две функции: открывает файл, если он уже создан; создает и открывает файл, если в момент вызова open файла с указанным именем нет.

При верификации средства защиты информации файловой системы в рамках типовой операционной системы с операцией open, которая открывает файл, если он уже создан, в качестве нотации моделирования используется Event-B. В формальной модели управления доступом операция open представляется как GetAccessForObjOperation, ее вид следующий:

event GetAccessForObjOperation
any
proc # *Процесс (субъект)*
obj # *Объект*
op # *Операция*
dir # *Родительский каталог*
where
@grd1 proc ∈ Processes
@grd2 obj ∈ Objects
@grd3 op ∈ ObjOperations
@grd4 dir ∈ Dirs ∩ Objects
@grd5 # *Режим администрирования*
Mode = UserMode ∨ (Mode = AdmMode ∧ EffProcUser(proc) = RootUser)
@grd6 # *dir — является родительским каталогом для obj*
obj ∈ Files ⇒ obj↦dir ∈ FileDirs
@grd7 # *У процесса proc есть доступ ExecObj для каталога dir*
obj ∈ Files ⇒ dir↦ExecObj ∈ ProcObjAccess(proc)
@grd8 # *И для всех родительских каталогов есть ExecObj*
obj ∈ Files ⇒ (∀d·dir↦d∈ContainingDirs ⇒ d↦ExecObj ∈ ProcObjAccess(proc))
@grd9 # *Если процесс proc запущен от администратора, то для доступа на выполнение должно быть выставлено право на исполнение в одной из трех групп прав доступа*
obj ∈ Files ∧ op=ExecObj ∧ EffProcUser(proc)=RootUser
⇒ UserAC↦op∈DACPermissions(obj) ∨
GroupAC↦op∈DACPermissions(obj) ∨
OthersAC↦op∈DACPermissions(obj)
@grd10 # *Проверка прав доступа пользователя*
obj ∈ Files ∧ EffProcUser(proc) ≠ RootUser
∧ EffProcUser(proc) = OwnerUser(obj)
⇒ (UserAC↦op)∈DACPermissions(obj)
@grd11 # *Проверка списков управления доступа пользователя*
obj ∈ Files ∧ EffProcUser(proc) ≠ RootUser
∧ EffProcUser(proc) ≠ OwnerUser(obj)
∧ obj∈dom(UserACL)
∧ EffProcUser(proc)∈dom(UserACL(obj))
⇒ EffProcUser(proc)↦op ∈ UserACL(obj) ∧ op∈ACLMask(obj)
@grd12 # *Проверка списков управления доступа группы*
obj ∈ Files ∧ EffProcUser(proc) ≠ RootUser
∧ EffProcUser(proc) ≠ OwnerUser(obj)
∧ (obj∉dom(UserACL) ∨
(obj∈dom(UserACL) ∧ EffProcUser(proc)∉dom(UserACL(obj))))
∧ (obj∈dom(GroupACL)
∧ (∃g·g∈dom(GroupACL(obj))
∧(g=EffProcGroup(proc)∨EffProcUser(proc)↦g∈UserGroups)))
⇒ op∈ACLMask(obj)

```
      ∧(∃g·g↦op∈GroupACL(obj)
      ∧ (g=EffProcGroup(proc) ∨ EffProcUser(proc)↦g∈UserGroups))
  @grd13 # Проверка прав доступа группы
    obj ∈ Files ∧ EffProcUser(proc) ≠ RootUser
    ∧ EffProcUser(proc) ≠ OwnerUser(obj)
    ∧ (obj∉dom(UserACL) ∨
      (obj∈dom(UserACL) ∧ EffProcUser(proc)∉dom(UserACL(obj))))
    ∧ (obj∉dom(GroupACL) ∨
        (obj∈dom(GroupACL) ∧ (∀g·g∈dom(GroupACL(obj))
          ⇒ (g≠EffProcGroup(proc) ∧ EffProcUser(proc)↦g∉UserGroups))))
    ∧ (OwnerGroup(obj)=EffProcGroup(proc) ∨
        EffProcUser(proc)↦OwnerGroup(obj)∈UserGroups)
  ⇒ GroupAC↦op ∈ DACPermissions(obj)
    ∧ ((obj∈dom(ACLMask)∧op∈ACLMask(obj)) ∨
        obj∉dom(ACLMask))
  @grd14 # Проверка прав доступа для остальных пользователей
    obj ∈ Files
  ∧ EffProcUser(proc) ≠ RootUser
  ∧ EffProcUser(proc) ≠ OwnerUser(obj)
  ∧ (obj∉dom(UserACL) ∨
      (obj∈dom(UserACL) ∧ EffProcUser(proc)∉dom(UserACL(obj))))
  ∧ (obj∉dom(GroupACL) ∨
     (obj∈dom(GroupACL)∧(∀g·g∈dom(GroupACL(obj))
       ⇒ (g≠EffProcGroup(proc) ∧ EffProcUser(proc)↦g∉UserGroups))))
  ∧ (OwnerGroup(obj)≠EffProcGroup(proc)
    ∧ EffProcUser(proc)↦OwnerGroup(obj)∉UserGroups)
  ⇒ OthersAC↦op ∈ DACPermissions(obj)
then # Обновление множества доступов процесса
  @act1 ProcObjAccess(proc) □ ProcObjAccess(proc) ∪ {obj↦op}
end
```

В формальной спецификации системного вызова соответствующая операция open выглядит так:

event open_exists
any
proc # *Процесс*
parent # *Родительский каталог*
file # *Файл*
name # *Имя файла*
flags # *Флаги операции*
fd # *Псевдо-параметр результата успешной операции, файловый дескриптор*
fdNumber # *Номер файлового дескриптора*
where
@grd1 proc ∈ Procs
@grd2 parent ∈ Folders
@grd3 file ∈ Files
@grd4 # В каталоге parent существует файл с именем name
file ↦ (parent ↦ name) ∈ FileParents
@grd5 flags ⊆ OPEN_FLAGS
@grd6 fd ∈ FILE_DESCRIPTORS \ FDs
@grd7 fdNumber ∈ N
@grd8 # *Файлового дескриптора fdNumber еще нет у proc*
∀pfd · proc ↦ pfd ∈ ProcFDs ⇒ FDNumber(pfd) ≠ fdNumber
@grd9 # Только открытие существующих файлов
O_PATH ∉ flags ⇒ ¬(O_CREAT ∈ flags ∧ O_EXCL ∈ flags)
@grd10 # *Режим доступа чтения, записи или чтения и записи*
O_RDONLY ∈ flags ∨ O_WRONLY ∈ flags ∨ O_RDWR ∈ flags
@grd11 # Исключающее или для доступов
¬(O_RDONLY ∈ flags ∧ O_WRONLY ∈ flags)
∧ ¬(O_RDONLY ∈ flags ∧ O_RDWR ∈ flags)
∧ ¬(O_WRONLY ∈ flags ∧ O_RDWR ∈ flags)

@grd12 # *Без O_PATH только чтение для каталогов*
file ∈ Folders ∧ O_PATH ∉ flags
⇒ O_WRONLY ∉ flags ∧ O_RDWR ∉ flags
@grd13 # *Ограничение на открытые файлы процесса*
card(ProcFDs[{proc}]) < PROC_FILE_LIMIT
@grd14 # *Ограничение на открытые файлы в системе*
card(ran(ProcFDs)) < FILE_LIMIT
@grd15 # *C O_DIRECTORY открывать только каталоги*
O_DIRECTORY ∈ flags ⇒ file ∈ Folders
@grd16 # *Следствие из grd10, grd11, grd12 и grd15*
O_DIRECTORY ∈ flags ∧ O_PATH ∉ flags ⇒ O_RDONLY ∈ flags
@grd17 # *Проверка пути до файла*
∀f · f ∈ PathToRoot(parent) ∪ {parent}
∧ ProcUser(proc) ≠ ROOT_USER
⇒ ((ProcUser(proc) = FileUser(f)
∧ UEXECUTE ∈ DACPermissions(f))
∨ (ProcUser(proc) ≠ FileUser(f)
∧ (f ↦ ProcUser(proc)) ↦ UEXECUTE ∈ UserACL
∧ f ↦ GEXECUTE ∈ MaskACL)
∨ (ProcUser(proc) ≠ FileUser(f)
∧ (f ↦ ProcUser(proc)) ∉ dom(UserACL)
∧ f ↦ GEXECUTE ∈ MaskACL
∧ (∃g · (f ↦ g) ↦ GEXECUTE ∈ GroupACL ∧ (g = ProcGroup(proc) ∨ ProcUser(proc) ↦ g ∈ UserGroups)))
∨ (ProcUser(proc) ≠ FileUser(f)
∧ (f ↦ ProcUser(proc)) ∉ dom(UserACL)
∧ (∀g · (f ↦ g) ∈ dom(GroupACL) ⇒ g ≠ ProcGroup(proc) ∧ ProcUser(proc) ↦ g ∉ UserGroups)
∧ (FileGroup(f) = ProcGroup(proc) ∨ ProcUser(proc) ↦ FileGroup(f) ∈ UserGroups)
∧ GEXECUTE ∈ DACPermissions(f)
∧ (f ∈ dom(MaskACL) ⇒ f ↦ GEXECUTE ∈ MaskACL))
∨ (ProcUser(proc) ≠ FileUser(f)
∧ (f ↦ ProcUser(proc)) ∉ dom(UserACL)
∧ (∀g · (f ↦ g) ∈ dom(GroupACL) ⇒ g ≠ ProcGroup(proc) ∧ ProcUser(proc) ↦ g ∉ UserGroups)
∧ FileGroup(f) ≠ ProcGroup(proc) ∧ ProcUser(proc) ↦ FileGroup(f) ∉ UserGroups
∧ OEXECUTE ∈ DACPermissions(f)))
@grd18 # *Проверка доступа на чтение*
ProcUser(proc) ≠ ROOT_USER
∧ O_RDONLY ∈ flags
∧ O_PATH ∉ flags
⇒ ((ProcUser(proc) = FileUser(file)
∧ UREAD ∈ DACPermissions(file))
∨ (ProcUser(proc) ≠ FileUser(file)
∧ (file ↦ ProcUser(proc)) ↦ UREAD ∈ UserACL
∧ file ↦ GREAD ∈ MaskACL)
∨ (ProcUser(proc) ≠ FileUser(file)
∧ (file ↦ ProcUser(proc)) ∉ dom(UserACL)
∧ file ↦ GREAD ∈ MaskACL
∧ (∃g · (file ↦ g) ↦ GREAD ∈ GroupACL ∧ (g = ProcGroup(proc) ∨ ProcUser(proc) ↦ g ∈ UserGroups)))
∨ (ProcUser(proc) ≠ FileUser(file)
∧ (file ↦ ProcUser(proc)) ∉ dom(UserACL)
∧ (∀g · (file ↦ g) ∈ dom(GroupACL) ⇒ g ≠ ProcGroup(proc) ∧ ProcUser(proc) ↦ g ∉ UserGroups)
∧ (FileGroup(file) = ProcGroup(proc) ∨ ProcUser(proc) ↦ FileGroup(file) ∈ UserGroups)
∧ GREAD ∈ DACPermissions(file)
∧ (file ∈ dom(MaskACL) ⇒ file ↦ GREAD ∈ MaskACL))
∨ (ProcUser(proc) ≠ FileUser(file)
∧ (file ↦ ProcUser(proc)) ∉ dom(UserACL)
∧ (∀g · (file ↦ g) ∈ dom(GroupACL) ⇒ g ≠ ProcGroup(proc) ∧ ProcUser(proc) ↦ g ∉ UserGroups)
∧ FileGroup(file) ≠ ProcGroup(proc) ∧ ProcUser(proc) ↦ FileGroup(file) ∉ UserGroups
∧ OREAD ∈ DACPermissions(file)))
@grd19 # *Проверка доступа на запись*
ProcUser(proc) ≠ ROOT_USER

```
        ∧ O_WRONLY ∈ flags
        ∧ O_PATH ∉ flags
          ⇒ ((ProcUser(proc) = FileUser(file)
                ∧ UWRITE ∈ DACPermissions(file))
              ∨ (ProcUser(proc) ≠ FileUser(file)
                ∧ (file ↦ ProcUser(proc)) ↦ UWRITE ∈ UserACL
                ∧ file ↦ GWRITE ∈ MaskACL)
              ∨ (ProcUser(proc) ≠ FileUser(file)
                ∧ (file ↦ ProcUser(proc)) ∉ dom(UserACL)
                ∧ file ↦ GWRITE ∈ MaskACL
                ∧ (∃g · (file ↦ g) ↦ GWRITE ∈ GroupACL ∧ (g = ProcGroup(proc) ∨ ProcUser(proc) ↦ g ∈ UserGroups)))
              ∨ (ProcUser(proc) ≠ FileUser(file)
                ∧ (file ↦ ProcUser(proc)) ∉ dom(UserACL)
                ∧ (∀g · (file ↦ g) ∈ dom(GroupACL) ⇒ g ≠ ProcGroup(proc) ∧ ProcUser(proc) ↦ g ∉ UserGroups)
                ∧ (FileGroup(file) = ProcGroup(proc) ∨ ProcUser(proc) ↦ FileGroup(file) ∈ UserGroups)
                ∧ GWRITE ∈ DACPermissions(file)
                ∧ (file ∈ dom(MaskACL) ⇒ file ↦ GWRITE ∈ MaskACL))
              ∨ (ProcUser(proc) ≠ FileUser(file)
                ∧ (file ↦ ProcUser(proc)) ∉ dom(UserACL)
                ∧ (∀g · (file ↦ g) ∈ dom(GroupACL) ⇒ g ≠ ProcGroup(proc) ∧ ProcUser(proc) ↦ g ∉ UserGroups)
                ∧ FileGroup(file) ≠ ProcGroup(proc) ∧ ProcUser(proc) ↦ FileGroup(file) ∉ UserGroups
                ∧ OWRITE ∈ DACPermissions(file)))
      @grd20 # Проверка доступа на чтение и запись
        ProcUser(proc) ≠ ROOT_USER
        ∧ O_RDWR ∈ flags
        ∧ O_PATH ∉ flags
          ⇒ ((ProcUser(proc) = FileUser(file)
                ∧ {UREAD, UWRITE} ⊆ DACPermissions(file))
              ∨ (ProcUser(proc) ≠ FileUser(file)
                ∧ {UREAD, UWRITE} ⊆ UserACL[{file ↦ ProcUser(proc)}]
                ∧ {GREAD, GWRITE} ⊆ MaskACL[{file}])
              ∨ (ProcUser(proc) ≠ FileUser(file)
                ∧ (file ↦ ProcUser(proc)) ∉ dom(UserACL)
                ∧ {GREAD, GWRITE} ⊆ MaskACL[{file}]
                ∧ (∃g · {GREAD, GWRITE} ⊆ GroupACL[{file ↦ g}] ∧ (g = ProcGroup(proc) ∨ ProcUser(proc) ↦ g ∈ User-
Groups)))
              ∨ (ProcUser(proc) ≠ FileUser(file)
                ∧ (file ↦ ProcUser(proc)) ∉ dom(UserACL)
                ∧ (∀g · (file ↦ g) ∈ dom(GroupACL) ⇒ g ≠ ProcGroup(proc) ∧ ProcUser(proc) ↦ g ∉ UserGroups)
                ∧ (FileGroup(file) = ProcGroup(proc) ∨ ProcUser(proc) ↦ FileGroup(file) ∈ UserGroups)
                ∧ {GREAD, GWRITE} ⊆ DACPermissions(file)
                ∧ (file ∈ dom(MaskACL) ⇒ {GREAD, GWRITE} ⊆ MaskACL[{file}]))
              ∨ (ProcUser(proc) ≠ FileUser(file)
                ∧ (file ↦ ProcUser(proc)) ∉ dom(UserACL)
                ∧ (∀g · (file ↦ g) ∈ dom(GroupACL) ⇒ g ≠ ProcGroup(proc) ∧ ProcUser(proc) ↦ g ∉ UserGroups)
                ∧ FileGroup(file) ≠ ProcGroup(proc) ∧ ProcUser(proc) ↦ FileGroup(file) ∉ UserGroups
                ∧ {OREAD, OWRITE} ⊆ DACPermissions(file)))
    then
      @act1 FDs := FDs ∪ {fd}
      @act2 FDNumber(fd) := fdNumber
      @act3 ProcFDs := ProcFDs ∪ {proc ↦ fd}
      @act4 FDFlags(fd) := flags
      @act5 FDFile(fd) := file
    end
```

Для организации динамической верификации необходимо построить отображение реализационных сущностей (типов данных, переменных, областей памяти и других элементов программы) и вызовов операций в модельные. Реализационными вызовами здесь является

подмножество системных вызовов open при условии открытия существующих в системе файлов.

Пример

int open(const char *pathname, int flags);

Где:

- ***pathname - имя файла;***
- ***flags - один из режимов доступа (чтение, запись, чтение и запись) вместе со специальными флагами открытия файла.***

Существуют и другие системные вызовы, например openat, описываемые данной моделью, но их рассмотрение выходит за рамки примера.

В качестве отдельного теста может служить простая программа, выполняющая один системный вызов.

Пример

```
int
main(int argc, char *argv[])
{
  syscall(SYS_open, argv[1], argv[2]);
  return 0;
}
```

Тестирование сводится к выполнению этой программы от лица разных пользователей, на файлах с разными разрешениями доступа и разными режимами открытия файла.

Отчет о результатах тестирования может выглядеть как таблица с информацией о покрытых условиях, входящих в охранные условия (см. приложение А) модельной операции (см. таблицу Б.1). К примеру, 141 тест на открытие существующего файла покрыл модельные условия следующим образом: T - количество раз, когда условие получило истинное значение, F - когда условие получило ложное значение, U - когда оно не могло быть вычислено, I - протестировано ли условие независимо от остальных.

Таблица Б.1

Информация о покрытых условиях, входящих в охранные условия модельной операции

Условие	T	F	U	I	Комментарий	Предикат
grd5	141	0	0	Нет	no False tests	flags ⊆ OPEN_FLAGS
grd9_c00	0	141	0	Нет	no True tests	O_PATH ∈ flags
grd9_c01	0	141	0	Нет	no True tests	O_CREAT ∈ flags
grd9_c02	0	141	0	Нет	no True tests	O_EXCL ∈ flags
grd10_c00	110	31	0	Нет	no independent	O_RDONLY ∈ flags
grd10_c01	31	110	0	Нет	no independent	O_WRONLY ∈ flags
grd10_c02	0	141	0	Нет	no True tests	O_RDWR ∈ flags
grd12_c00	55	86	0	Нет	no independent	file ∈ Folders
grd13	141	0	0	Нет	no False tests	card(ProcFDs[{proc}])<PROC_FILE_ (...)
grd14	141	0	0	Нет	no False tests	card(ran(ProcFDs))<FILE_LIMIT
grd15_c00	55	86	0	Нет	no independent	O_DIRECTORY ∈ flags
grd17	129	12	0	**Да**		∀ f·f ∈ PathToRoot(parent) ⋃ {parent} (...)
grd18_c00	30	111	0	Нет	no independent	ProcUser(proc)=ROOT_USER
grd18_c01	54	87	0	Нет	no independent	FileUser(file)=ProcUser(proc)

grd18_c02	135	6	0	Нет	no independent	UREAD ∈ DACPermissions(file)
grd18_c03	9	132	0	Нет	no independent	file ↦ ProcUser(proc) ∈ dom(UserAC (...)
grd18_c04	5	4	132	Нет	no independent	UREAD ∈ UserACL(file ↦ ProcUser(pr (...)
grd18_c05	42	27	72	Нет	no independent	GREAD ∈ MaskACL(file)
grd18_c06	32	109	0	**Да**		∃ g·(g=ProcGroup(proc) ∨ ProcUser(p (...)
grd18_c07	69	72	0	Нет	no independent	file ∈ dom(MaskACL)
grd18_c08	69	72	0	Нет	no independent	FileGroup(file)=ProcGroup(proc)
grd18_c09	90	51	0	Нет	no independent	ProcUser(proc) ↦ FileGroup(file) (...)
grd18_c10	48	93	0	Нет	no independent	GREAD ∈ DACPermissions(file)
grd 18_c11	108	33	0	Нет	no independent	∀ g·g=ProcGroup(proc) ∨ ProcUser(pr (...)
grd18_c12	3	138	0	Нет	no independent	OREAD ∈ DACPermissions(file)
grd19_c00	135	6	0	Нет	no independent	UWRITE ∈ DACPermissions(file)
grd19_c01	4	5	132	Нет	no independent	UWRITE ∈ UserACL(file ↦ ProcUser(p (...)

grd19_c02	33	36	72	Нет	no independent	GWRITE ∈ MaskACL(file)
grd19_c03	34	107	0	**Да**		∃ g·(g=ProcGroup(proc) ∨ ProcUser(p (…)
grd19_c04	39	102	0	Нет	no independent	GWRITE ∈ DACPermissions(file)
grd19_c05	3	138	0	Нет	no independent	OWRITE ∈ DACPermissions(file)
grd20_c00	0	141	0	Нет	no True tests	∃ g·(g=ProcGroup(proc) ∨ ProcUser(p (…)

Условия grd1 - grd4, grd6 - grd8, grd11 и grd16 опущены, так как они представляют собой типовые ограничения на параметры и всегда должны выполняться.

Условие grd5 в данном примере означает вызов операции с согласованными значениями флагов, что также всегда должно выполняться.

Условие grd9 также означает согласованность флагов при вызове операции для существующего файла и тоже не может быть нарушено.

Условие grd10 представляет собой ограничение на возможные флаги, позволяющие открыть файл только на чтение, только на запись и на чтение и запись одновременно. В рамках тестов реализовывались только первые две ситуации, открытие на чтение и запись одновременно не выполнялось.

Условия grd12 и grd15 означают ограничения на согласованность флагов при открытии каталога. Сами ограничения не нарушались, но в ходе тестов открывались как каталоги, так и обычные файлы.

Условия grd13 и grd 14 означают ограничения на количество файлов, открытых в рамках одного процесса и в системе в целом. Они в рамках тестов всегда были выполнены, попыток открыть слишком большое число файлов не предпринималось.

Условие grd17 означает ограничение на доступность всех директорий на пути до открываемого файла. Это условие в тестах принимало значение как TRUE, так и FALSE.

Условия grd18, grd19 и grd20 описывают специфические ограничения, которые должны выполняться в системе при корректном доступе к файлу только на чтение (grd18), только на запись (grd19) и на чтение и запись (grd20). Эти условия были разбиты на элементарные логические формулы, которые в большинстве случаев (кроме формулы grd20_c00) в тестах принимали значения как TRUE, так и FALSE. При этом сами условия grd18 и grd19 также принимали оба значения, а условие grd20 всегда было выполнено.

Соответственно, рекомендации по покрытию ситуаций, где остальные изменяемые охранные условия принимают оба возможных значения, выполнены, за исключением ситуаций открытия файлов на чтение и запись одновременно.

Проверка набора тестовых ситуаций по независимому выполнению отдельных условий в этом примере не полностью выполнена. На практике полный перебор бывает сложен и анализ зависимостей весьма трудоемок. Известно, что полный перебор тестовых ситуаций с анализом независимого выполнения отдельных условий позволяет повысить степень уверенности в корректности тестируемой системы, такой перебор предписывается, например, в Квалификационных требованиях [1].

БИБЛИОГРАФИЯ

[1]	Квалификационные требования КТ-178С	Требования к программному обеспечению бортовой аппаратуры и систем при сертификации авиационной техники

УДК 004.056:006.354 ОКС 35.030

Ключевые слова: защита информации, формальная модель управления доступом, средство защиты информации, политика управления доступом, верификация модулей средства защиты информации