МЕЖГОСУДАРСТВЕННЫЙ СОВЕТ ПО СТАНДАРТИЗАЦИИ, МЕТРОЛОГИИ И СЕРТИФИКАЦИИ
(МГС)
INTERSTATE COUNCIL FOR STANDARDIZATION, METROLOGY AND CERTIFICATION
(ISC)

МЕЖГОСУДАРСТВЕННЫЙ СТАНДАРТ

ГОСТ ISO/IEC TS 19249— 2021

Информационные технологии

МЕТОДЫ И СРЕДСТВА ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ

Каталог принципов построения архитектуры и проектирования безопасных продуктов, систем и приложений

(ISO/IEC TS 19249:2017, IDT)

Издание официальное

Москва
Стандартинформ
2021

Предисловие

Цели, основные принципы и общие правила проведения работ по межгосударственной стандартизации установлены ГОСТ 1.0 «Межгосударственная система стандартизации. Основные положения» и ГОСТ 1.2 «Межгосударственная система стандартизации. Стандарты межгосударственные, правила и рекомендации по межгосударственной стандартизации. Правила разработки, принятия, обновления и отмены»

Сведения о стандарте

1 ПОДГОТОВЛЕН Федеральным государственным учреждением «Федеральный исследовательский центр «Информатика и управление» Российской академии наук» (ФИЦ ИУ РАН), Обществом с ограниченной ответственностью «Центр безопасности информации» и Обществом с ограниченной ответственностью «Информационно-аналитический центр» (ООО ИАВЦ) на основе собственного перевода на русский язык англоязычной версии документа, указанного в пункте 5

2 ВНЕСЕН Межгосударственным техническим комитетом по стандартизации МТК 022 «Информационные технологии»

3 ПРИНЯТ Межгосударственным советом по стандартизации, метрологии и сертификации (протокол от 30 июня 2021 г. № 141-П)

За принятие проголосовали:

Краткое наименование страны по МК (ИСО 3166) 004—97	Код страны по МК (ИСО 3166) 004—97	Сокращенное наименование национального органа по стандартизации
Армения	AM	ЗАО «Национальный орган по стандартизации и метрологии» Республики Армения
Беларусь	BY	Госстандарт Республики Беларусь
Киргизия	KG	Кыргызстандарт
Россия	RU	Росстандарт
Узбекистан	UZ	Узстандарт

4 Приказом Федерального агентства по техническому регулированию и метрологии от 2 июля 2021 г. № 611-ст межгосударственный стандарт ГОСТ ISO/IEC TS 19249—2021 введен в действие в качестве национального стандарта Российской Федерации с 30 ноября 2021 г.

5 Настоящий стандарт идентичен международному документу ISO/IEC TS 19249:2017 «Информационные технологии. Методы и средства обеспечения безопасности. Каталог принципов построения архитектуры и проектирования безопасных продуктов, систем и приложений» («Information technology — Security techniques — Catalogue of architectural and design principles for secure products, systems and applications», IDT).

ISO/IEC TS 19249:2017 разработан подкомитетом SC 27 «Информационная безопасность, кибербезопасность и защита конфиденциальности» Совместного технического комитета JTC 1 «Информационные технологии» Международной организации по стандартизации (ISO) и Международной электротехнической комиссии (IEC).

При применении настоящего стандарта рекомендуется использовать вместо ссылочных международных стандартов соответствующие им межгосударственные стандарты, сведения о которых приведены в дополнительном приложении ДА.

Дополнительные сноски в тексте стандарта, выделенные курсивом, приведены для пояснения текста оригинала

6 ВВЕДЕН ВПЕРВЫЕ

Информация о введении в действие (прекращении действия) настоящего стандарта и изменений к нему на территории указанных выше государств публикуется в указателях национальных стандартов, издаваемых в этих государствах, а также в сети Интернет на сайтах соответствующих национальных органов по стандартизации.

В случае пересмотра, изменения или отмены настоящего стандарта соответствующая информация будет опубликована на официальном интернет-сайте Межгосударственного совета по стандартизации, метрологии и сертификации в каталоге «Межгосударственные стандарты».

Содержание

Введение

В настоящем стандарте описаны принципы построения архитектуры и проектирования, которые могут использоваться при разработке безопасных продуктов, систем и приложений.

Цель настоящего стандарта — описать эти принципы структурированным образом, охарактеризовать сам принцип, описать, как его можно использовать, как он может поддерживать безопасность и как это может помочь в оценке безопасности продукта, системы или приложения. Настоящий стандарт может применяться при проведении оценки информационной безопасности, выполняемой с использованием ISO/IEC 15408 и ISO/IEC 18045.

Поправка к ГОСТ ISO/IEC TS 19249—2021 Информационные технологии. Методы и средства обеспечения безопасности. Каталог принципов построения архитектуры и проектирования безопасных продуктов, систем и приложений

Дата введения — 03.09.2021

В каком месте	Напечатано	Должно быть		
Предисловие. Таблица согласования	—	Казахстан	KZ	Госстандарт Республики Казахстан

(ИУС № 11 2021 г.)

М Е Ж Г О С У Д А Р С Т В Е Н Н Ы Й С Т А Н Д А Р Т

Информационные технологии

МЕТОДЫ И СРЕДСТВА ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ

Каталог принципов построения архитектуры и проектирования безопасных продуктов, систем и приложений

Information technology. Security techniques. Catalogue of architectural and design principles for secure products, systems and applications

Дата введения —2021—11—30

1 Область применения

Настоящий стандарт представляет собой каталог принципов построения архитектуры и проектирования, которые допускается использовать при разработке безопасных продуктов, систем и приложений, а также руководство по эффективному использованию этих принципов. Каждый принцип построения архитектуры и проектирования описывается с использованием общей структуры, определяющей цель и преимущество принципа проектирования, то, каким образом он может способствовать разработке безопасного продукта, системы или приложения, его зависимость от других принципов, описанных в каталоге. Для каждого принципа представлены примеры того, как он может быть реализован, как он может способствовать свойствам и функциям безопасности и какие другие аспекты должны быть приняты во внимание в приведенном примере, чтобы также учитывать требования, не связанные с безопасностью, такие как удобство использования и производительность.

В настоящем стандарте приведены рекомендации по разработке безопасных продуктов, систем и приложений; он направлен на обеспечение более эффективной оценки свойств безопасности, которые должны реализовывать эти продукты, системы и приложения.

Настоящий стандарт относится к области применения ISO/IEC 15408 и ISO/IEC 18045 и адресован как разработчикам, так и специалистам по оценке безопасности продуктов, систем и приложений.

Настоящий стандарт не устанавливает никаких требований к оценке, процессу оценки или реализации.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты [для датированных ссылок применяют только указанное издание ссылочного стандарта, для недатированных — последнее издание (включая все изменения]:

ISO/IEC 15408-1, Information technology — Security techniques — Evaluation criteria for IT security — Part 1: Introduction and general model (Информационные технологии. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 1. Введение и общая модель)

ISO/IEC 15408-2, Information technology — Security techniques — Evaluation criteria for IT security — Part 2: Security functional components (Информационные технологии. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 2. Функциональные компоненты безопасности)

Издание официальное

ISO/IEC 15408-3, Information technology — Security techniques — Evaluation criteria for IT security — Part 3: Security assurance components (Информационные технологии. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 3. Компоненты доверия к безопасности)

ISO/IEC 18045, Information technology — Security techniques — Methodology for IT security evaluation (Информационные технологии. Методы и средства обеспечения безопасности. Методология оценки безопасности информационных технологий)

3 Термины и определения

В настоящем стандарте применены термины по ISO/IEC 15408 и ISO/IEC 18045, а также следующие термины с соответствующими определениями:

3.1 **свойство безопасности** (security property): Свойство системы или приложения, которое имеет решающее значение для достижения целей безопасности, определенных для системы или приложения.

3.2 **атрибут безопасности** (security relevant attribute): Атрибут, назначенный объекту, субъекту или пользователю системы или приложения, который используется для обеспечения соблюдения политики безопасности.

3.3 **привилегия** *(*privilege): Конкретный атрибут безопасности для субъекта или пользователя, который позволяет этому пользователю или субъекту выполнять специальные действия, связанные с этим атрибутом безопасности.

3.4 **служба прозрачного шифрования** (transparent encryption service): Услуга, предоставляемая системой, которая либо всегда включена, либо может быть включена администратором, автоматически зашифровывающая и дешифровывающая данные в постоянном хранилище или в каналах связи, о которой пользователь, хранящий или передающий данные, не знает, что данная служба шифрования работает.

Примечание — Примерами являются службы автоматического шифрования дисков (например, самошифрующиеся диски) или туннелирование канала связи через линию связи, защищенный IPSec или TLS.

4 Принципы построения архитектуры безопасных продуктов, систем и приложений

4.1 Общие положения

Создание безопасного продукта, системы или приложения требует не только реализации функциональных требований, но и архитектуры, которая позволяет эффективно осуществлять определенные свойства безопасности, которые продукт, система или приложение должны обеспечивать. Способность противостоять атакам, с которыми продукт, система или приложение могут столкнуться в предполагаемой операционной среде, во многом зависит от архитектуры, которая блокирует эти атаки или, если они не могут быть заблокированы, позволяет обнаруживать такие атаки и/или минимизировать ущерб, причиняемый такими атаками. Структурирование продукта, системы или приложения в выделенные домены, которые изолированы друг от друга и могут обмениваться данными только с использованием четко определенных каналов связи, где могут применяться определенные политики безопасности, — это и есть общий архитектурный принцип, который должен быть применен. Структура архитектуры, использующая такие домены, должна быть тщательно подобрана, чтобы обеспечить соблюдение свойств и требований безопасности, а также содействовать реализации требований, не связанных с безопасностью. Такая структура также не должна вызывать недопустимого снижения быстродействия продукта, системы или приложения. Поиск правильного баланса в структуре архитектуры, которая обеспечивала бы выполнение всех требований к продукту, системе или приложению, является основным предметом обсуждения архитектурных принципов, изложенных в этом разделе, которые могут использоваться совместно с соответствующими принципами проектирования.

Существует несколько фундаментальных принципов безопасности, которые необходимо соблюдать независимо от принципов построения архитектуры или проектирования. Они известны уже давно и впервые были задокументированы в исследовании, проведенном по поручению Правительства США в 1972 году [1]. Фундаментальные принципы:

- реализация функций безопасности должна быть такой, чтобы эти функции невозможно было обойти или подменить ненадежным кодом;

- функции безопасности должны быть вызваны каждый раз, когда необходимо осуществить выполнение политики безопасности;

- часть продукта, системы или приложения, реализующая функции безопасности, должна быть достаточно мала, чтобы ее можно было проанализировать на корректность и возможные побочные эффекты, критичные для безопасности.

В более поздних отчетах [2] части продукта, которые должны быть доверенными для выполнения политик безопасности и которые удовлетворяют вышеописанным принципам безопасности, называют «доверенной вычислительной базой» или ДВБ.

Ряд принципов проектирования, упомянутых в этой спецификации, уже был введен в [3] в 1974 г.

В настоящее время ИТ-продукты и системы значительно усложнились и теперь часто распространяются, полагаясь на внешних провайдеров услуг, и охватывают более широкий спектр функций безопасности. Это требует более сложных архитектур и конструкций, и настоящий стандарт предоставляет описание принципов построения архитектуры и проектирования с соответствующими аспектами безопасности.

4.2 Разделение на домены

4.2.1 Общие положения

Разделение на домены — это принцип инкапсуляции компонентов, данных и/или программ в отдельные объекты, которыми можно управлять независимо, включая назначение прав и других атрибутов безопасности каждому домену. Между доменами могут быть определены каналы связи, реализованные таким образом, чтобы контролировать взаимодействие между доменами, а также позволять домену, получающему запрос из другого домена по каналу связи, идентифицировать запрашивающий домен и соответствующе реагировать на полученный запрос.

Помимо безопасности, могут существовать различные причины для структурирования системы, приложения и т. д. Настоящий стандарт рассматривает только причины структурирования, связанные с безопасностью.

С точки зрения безопасности разделение продукта, системы или приложения на отдельные домены может быть эффективным методом для объединения функций и объектов, требующих выделенного набора привилегий, прав доступа, или для отделения критичных функций и объектов от менее критичных. Разделение доменов особенно важно и полезно, если определенные свойства безопасности могут быть сопоставлены с выделенными доменами, и доказательство того, что система или продукт удовлетворяет этим свойствам, может быть ограничено проверкой этих выделенных доменов.

Разделение доменов часто связано также с распределенной структурой системы или приложения, когда отдельные домены реализованы в физически разделенных частях продукта и обмениваются данными через сетевые соединения. В таких архитектурах один домен может реализовывать некоторый общий сервис, который используется сразу несколькими системами. Примерами служб безопасности, предоставляемых таким образом, являются службы каталогов, службы централизованной аутентификации, службы управления сертификатами или службы централизованного управления доступом.

4.2.2 Принципы определения структур доменов

При разделении приложения или системы на отдельные домены следует учитывать не только соображения безопасности. В настоящем стандарте перечислены только причины, связанные с безопасностью, которые следует учитывать при определении доменной структуры.

Первопричина для определения доменов с точки зрения безопасности заключается в том, чтобы отделить компоненты приложения или системы с общими атрибутами безопасности, такими как права, доступ к файлам и т. д., от других компонентов с различными атрибутами безопасности. Хорошая структура позволяет каждому домену работать с минимальным набором прав, необходимым для выполнения своей задачи. Кроме того, изолирование доменов с четко определенными и управляемыми интерфейсами и связями между ними позволяет легко обнаруживать ошибки, ограничивать их распространение и реализовывать стратегию глубокой защиты. Домены должны быть отделены друг от друга, чтобы взаимное влияние между доменами было ограничено и управляемо. Метод, используемый для изолирования, зависит от операционной среды. Отдельные домены могут быть реализованы в различных физических системах, связанных между собой некоторой сетью, они могут быть реализованы различными виртуальными машинами, которые обмениваются данными через механизмы, предоставля-

емые базовым гипервизором, или они могут быть реализованы как отдельные процессы, работающие в операционной системе. В этих случаях механизм разделения обеспечивается аппаратной частью, гипервизором или операционной системой. В остальных случаях приложение или система могут самостоятельно реализовать уровень, который представляет общий механизм разделения (часто с использованием вспомогательных функций основных аппаратных средств или программного обеспечения).

Часто хорошей стратегией является наделение доменов, подверженных атаке, минимальным набором прав, необходимых для их работы. Доверие к таким доменам со стороны других доменов также должно быть ограничено, что требует от других доменов выполнения дополнительных проверок данных, которые они получают от таких доменов с более низким уровнем доверия.

4.2.3 Принципы построения взаимодействия между доменами

Междоменные связи требуются для того, чтобы отдельные домены могли взаимодействовать. Поскольку домены имеют разные права и могут быть подвержены различным атакам, обмен данными между отдельными доменами должен контролироваться политикой, которая учитывает угрозы, связанные с различиями в привилегиях и покрытиях атак. Таким образом, такая политика должна определять:

- как один домен может доверять другому домену;
- как один домен может принимать данные, полученные из другого домена (например, если атрибутам данных можно доверять, если данные были проанализированы на предмет их синтаксиса и структуры, если данным обеспечена конфиденциальность);
- какие проверки безопасности необходимо выполнить;
- когда принимать или отклонять полученные данные.

4.2.4 Политики безопасности, которые могут быть реализованы с помощью разделения доменов

Разделение доменов можно использовать в качестве основы для определения политик информационных потоков или политик рабочих процессов путем определения способа взаимодействия доменов друг с другом. Политики могут быть определены на основе идентификации доменов или их атрибутов, таких как привилегии. Кроме того, уровень доверия к отдельным доменам может использоваться для обеспечения соблюдения политики.

4.2.5 Примеры

4.2.5.1 Структурирование сети на домены

Сети могут быть разделены на домены (сегменты), где связь между различными доменами регулируется шлюзами, такими как маршрутизаторы с функциями межсетевого экрана или более сложными устройствами с фильтрацией на уровне приложений. Маршрутизация, а также такие методы, как VPN[1)] или VLAN[2)], могут использоваться для определения способов связи между доменами, в то время как межсетевые экраны и шлюзы контролируют протокол и контент. Политики управления информационными потоками, основанные на метках данных, также могут быть эффективно реализованы путем разделения доменов, когда образцы «доверенных шлюзов» управляют интерфейсами между различными доменами «с одной меткой» и гарантируют, что данные передаются в домен только тогда, когда метка соответствует метке домена.

4.2.5.2 Разделение приложения по доменам

Подобно сетям отдельные приложения также могут быть разделены на домены, которые в данном случае представляют собой различные процессы. Эти процессы могут быть внутри одной физической системы либо распределены по нескольким системам в сети. В одной системе процессы обычно разделяются операционной системой или гипервизором, которые также обеспечивают возможность определения и использования способов связи между этими процессами.

4.2.5.3 Структурирование данных в домены

На домены могут быть разделены также данные в зависимости от важности данных и требований к их защите. Критические данные затем могут быть помещены в домен с высоким уровнем защиты, например в тот, к которому могут получить доступ только процессы с высокими привилегиями, или тот, который включает резервирование, чтобы избежать потери данных. Другой метод структурирования данных в различных доменах — зашифровать данные в каждом домене с использованием разных ключей шифрования. Тот, кто хочет получить доступ к информации, должен иметь доступ к ключу домена для расшифровки. Домены для данных могут быть реализованы с помощью различных файлов (кото-

---

1) *VPN — Virtual Private Network (виртуальная частая сеть).*

2) *VLAN — Virtual Local Area Network (виртуальная локальная сеть).*

рые позволяют использовать различные атрибуты управления доступом), различных дисков с разными свойствами надежности/целостности (например, уровнями RAID) или использовать СУБД[1)] в качестве хранилища данных, обеспечивающего дополнительную безопасность, такую как, например, безопасность транзакций.

4.2.6 Замечания по оценке аспектов безопасности

В вопросах оценки аспектами, которые необходимо рассмотреть в первую очередь, являются механизмы, используемые для разделения доменов, и механизмы, используемые для связи между доменами.

При оценке аспектов безопасности разделения доменов особенно важны механизмы, обеспечивающие разделение доменов, степень изоляции доменов друг от друга, управление доменами (если используется), такие как создание, удаление доменов или изменение привилегий доменов. Следует рассмотреть следующие аспекты:

- возможности механизмов разделения, которые изолируют домены друг от друга таким образом, что единственный способ, которым домены могут влиять друг на друга, — это через заданные механизмы связи;
- являются ли домены и их привилегии статическими или динамическими. Если они являются динамическими, необходимо создать и оценить более сложную модель, показывающую, что свойства безопасности системы не могут быть нарушены допустимой динамикой.

4.3 Многоуровневая архитектура

4.3.1 Общие положения

Многоуровневая архитектура — это иерархическая архитектура, в которой функции одного уровня используют функции другого, более низкого, уровня и предоставляют функции, которые используются на следующем, более высоком, уровне иерархии. Обычно в многоуровневой архитектуре самый нижний уровень предоставляет очень простые базовые функции, которые затем используются следующим, более высоким, уровнем для реализации более сложных функций, затем используются следующим уровнем для обеспечения еще более сложных функций и т. д. Каждый уровень абстрагируется от функций более низких уровней, позволяя использовать эти функции нижних уровней только тем функциям, которые он предоставляет. Это позволяет каждому уровню также реализовать свою собственную политику безопасности — при условии, что многоуровневая архитектура реализована таким образом, что не позволяет уровню обходить функции предыдущего нижнего уровня и напрямую использовать функции, предоставленные двумя уровнями ниже.

Многоуровневая архитектура широко используются при разработке сетевых протоколов, а также при разработке приложений. Но она не участвует непосредственно в обеспечении безопасности, поддерживая удобство эксплуатации и масштабируемость, являющиеся аспектами, поддерживающими безопасность и доверие.

4.3.2 Принципы определения уровней

С точки зрения безопасности уровни предоставляют механизм, позволяющий распределять функции безопасности по разным уровням, в зависимости от основных свойств функций и объектов, предоставляемых каждым уровнем, и защищать функции, реализованные на иерархически более низких уровнях, от вмешательства или обхода функциями, реализованными на иерархически более высоких уровнях.

Для того чтобы защитить иерархически более низкие уровни, каждый уровень (или набор смежных уровней) может быть реализован как свой собственный домен с иерархически более низкими уровнями, имеющий больше привилегий, чем более высокие уровни.

4.3.3 Принципы работы интерфейсов, предоставляемых уровнем

Уровень обычно реализует службы более высокого уровня, используя функции предыдущего, более низкого, уровня. Это означает, что все основные свойства, необходимые для реализации функциональности уровня, могут быть построены с использованием интерфейсов, предоставляемых предыдущим, более низким, уровнем. В некоторых случаях слою может быть разрешен доступ не только к функциям предыдущего, более низкого, уровня, но также и к функциям уровня ниже, чем предыдущий.

[1)] *СУБД — система управления базами данных.*

4.3.4 Политики безопасности, которые могут быть реализованы с помощью многоуровневой архитектуры

Многоуровневость не является основным архитектурным принципом, связанным с конкретными политиками безопасности, но ее можно эффективно использовать для реализации функций безопасности на уровне, доступном более высоким уровням. Службы прозрачного шифрования являются одним из примеров таких служб безопасности, которые могут быть реализованы на уровне. Это довольно распространенное явление для файловых систем, а также для криптографических протоколов. Подобным образом на выделенном уровне могут быть реализованы также другие службы безопасности, которые поддерживают службы обеспечения целостности, контроля доступа или избыточности.

4.3.5 Примеры

4.3.5.1 Многоуровневая архитектура файловой системы

Например, в многоуровневой реализации файловой системы самый нижний уровень может предоставлять доступ только к неразмеченным секторам диска, и его единственная функция безопасности может заключаться в обеспечении целостности секторов диска. Второй уровень может затем создавать виртуальные диски на основе уровня неразмеченных секторов диска и предоставлять дополнительные функции безопасности, такие как базовый контроль доступа к этим виртуальным дискам (например, позволяя назначать их разделам или приложениям), или даже реализовывать функциональность полного шифрования диска для этих виртуальных дисков или путем реализации улучшенных механизмов обеспечения целостности.

Третий уровень может затем реализовать файловую систему на тех виртуальных дисках, где файловая система обеспечивает сложную функциональность управления доступом (например, на основе списка контроля доступом и/или других атрибутов безопасности файлов) и возможно шифрование на основе файлов.

Четвертый уровень на вершине такой файловой системы может быть реализован приложением, когда данные файлов предоставляются отдельным пользователям приложения путем реализации собственного контроля доступа или других функций безопасности на основе отдельных записей в файлах.

4.3.5.2 Криптографические функции

Криптографические функции могут быть также реализованы многоуровневым образом. В качестве примера можно определить следующие уровни:

На самых нижних уровнях реализуются основные криптографические примитивы, такие как алгоритмы шифрования, хэш-алгоритмы и генерация случайных чисел.

На следующем более высоком уровне эти функции используются для создания и проверки цифровой подписи, кодов аутентификации сообщений, генерации ключей, обмена ключами, управления ключами и т. д.

Следующий уровень реализует такие функции, как проверка сертификата, создание сертификата и шифрование ключей.

4.3.6 Замечания по оценке

Многоуровневая архитектура может быть использована в пределах одного доверенного домена для структурирования функций в этом домене или ее можно использовать для реализации уровней как своих собственных доменов. В первом случае нет необходимости рассматривать какие-либо конкретные аспекты оценки. В случае, когда уровни реализованы как отдельные домены, при оценке необходимо убедиться, что соблюдены правила разделения доменов, и, кроме того, уровни корректно расположены относительно друг друга. Хотя не запрещено, чтобы уровень использовал функции не только предыдущего, но и более глубоких уровней, такая реализация может позволить обойти функциональность безопасности, реализованную на уровне. При оценке необходимо тщательно проанализировать такую реализацию архитектуры, чтобы убедиться, что обход функций безопасности в такой реализации невозможен.

4.4 Инкапсуляция

4.4.1 Общие положения

Инкапсуляция — это архитектурный принцип, при котором объекты «инкапсулируются» функциями, используемыми для доступа к объектам, манипулирования или управления ими. Совокупность таких функций, которые инкапсулируют объект, можно рассматривать как агента, управляющего объектом.

В такой концепции функции, которые инкапсулируют объект, могут определять политику безопасности для объекта. Они могут определять способ создания экземпляра объекта, потенциального изменения экземпляра объекта (например, если объект состоит из данных, изменять место и способ хранения данных), а также определять политику управления доступом для объекта (которая требует, чтобы функция инкапсуляции могла безопасно получать всю информацию, необходимую для принятия решения на основе политики безопасности, например гарантированную идентификацию средством доступа).

4.4.2 Принципы реализации инкапсуляции

Когда инкапсуляция используется для обеспечения безопасности, агенты, управляющие объектом, могут также реализовывать различные функции безопасности, защищающие объект. Эти функции безопасности могут, например, включать в себя управление доступом, аудит безопасности, защиту целостности или прозрачное шифрование.

Агенты, реализующие функции безопасности, необходимо отделить от недоверенных сущностей и гарантировать, что они не могут быть взломаны или проигнорированы.

4.4.3 Политики безопасности, которые могут быть реализованы с помощью инкапсуляции

Существует ряд функций безопасности, которые могут быть реализованы агентами, инкапсулирующими объект. К ним относятся контроль доступа к самому объекту, контроль того, кому разрешено использовать определенные функции, работающие с объектом, контроль управления объектом, контроль и управление атрибутами безопасности и привилегиями, которые агент связывает с объектом, аудит использования функций, обеспечение целостности и доступности объекта или шифрование содержимого объекта, когда он хранится в контейнерах, не полностью контролируемых агентом (например, если объекты являются записями, хранящимися в файле).

4.4.4 Примеры

В среде UNIX механизм setuid[1)] позволяет эффективно реализовать инкапсуляцию. Вместо предоставления пользователю доступа, например к файлу конфигурации, который содержит несколько параметров конфигурации, можно разработать набор программ, каждая из которых отвечает за управление только одним из этих параметров. Каждая из этих программ затем получает набор атрибутов setuid, в котором при вызове идентификатора пользователя переключается на тот, который предоставляет доступ для чтения и записи в файл конфигурации. Если нет других пользователей, имеющих право доступа к файлу, доступ к файлу конфигурации (для пользователей, не являющихся суперпользователями) осуществляется только через различные программы setuid. Набор программ setuid можно рассматривать в качестве агентов, инкапсулирующих файл конфигурации.

Сами программы setuid также могут управлять доступом к отдельному параметру конфигурации, контролировать значения этих параметров, контролировать их изменения, вести историю изменений с целью восстановления значений, обеспечивать совместимость с другими параметрами, хранить параметр только в зашифрованном виде и т. д. Кроме того, эта форма инкапсуляции позволяет делегировать определенную работу по администрированию безопасности и устанавливать конкретные административные роли, где каждая роль имеет доступ только к необходимому набору административных функций.

4.4.5 Замечания по оценке эффективности инкапсуляции

Оценка эффективности инкапсуляции требует от оценщика рассмотрения следующих аспектов:

- завершена ли инкапсуляция или есть ли аспекты инкапсулированного объекта, доступные без использования явно заданных функций;
- может ли механизм, реализующий саму инкапсуляцию, быть фальсифицирован;
- может ли инкапсуляция быть модифицирована несанкционированными сущностями (включая атрибуты безопасности, связанные с инкапсуляцией).

4.5 Резервирование

4.5.1 Общие положения

Резервирование — это архитектурный принцип, который позволяет восстанавливать такие элементы, как данные, функции, каналы связи, устройства и т. д., даже в случае ошибок. Если требуется непрерывная работа даже в случае ошибок или сбоев, резервирование часто используется для автома-

---

[1)] *Setuid является флагом прав доступа в операционной системе UNIX, который разрешают пользователям запускать исполняемые файлы с правами владельца или группы исполняемого файла.*

тического восстановления. Резервирование также может использоваться для повышения доступности элемента, особенно если ожидается высокая интенсивность его использования, и он может использоваться только последовательно или имеет ограничения по производительности.

Если резервирование используется для обеспечения восстановления работы или обеспечения непрерывной работы в случае ошибок или сбоев, важно иметь функциональные возможности, способные как можно быстрее обнаруживать потенциальные ошибки и сбои и определять, какой из элементов все еще работает правильно. Кроме того, неисправные элементы должны быть быстро заменены или отремонтированы для обеспечения должного уровня резервирования. В противном случае дополнительная ошибка или сбой в период сокращенного резервирования могут стать критическими.

В случае, если резервирование используется для восстановления, крайне важно обнаружить ошибку или сбой на этапе эксплуатации, когда восстановление еще возможно. Например, в системе транзакций обычно есть только фаза перед фиксацией транзакции, на которой будут работать процедуры восстановления.

Резервирование, используемое для повышения доступности, требует функций управления, которые бы распределяли работу между избыточными элементами таким образом, чтобы максимизировать пропускную способность или минимизировать максимальное время ожидания.

4.5.2 Принципы реализации избыточных элементов

При выборе избыточных элементов в архитектуре следует учитывать следующие аспекты:

- критичность/важность элемента;
- вероятность отказа элемента;
- требования к доступности элемента (обычно определяются требованиями к пропускной способности или максимальному времени ожидания/минимальному времени отклика).

4.5.3 Принципы обеспечения сопоставимости избыточных элементов

Основная проблема с резервированием заключается в обеспечении сопоставимости избыточных элементов. Когда данные хранятся избыточно, любое обновление этих данных должно отражаться во всех задействованных местах хранения. Ошибки или атаки, препятствующие корректному обновлению всех этих областей хранения, приводят к несоответствиям, которые противоречат цели резервирования и могут даже привести к нарушениям безопасности. Поэтому важно обеспечивать правильное отображение состояния обновлений, сохранение индикатора, пока обновление не будет завершено, и гарантировать, что обновления, выполненные только частично, можно либо отменить, либо вернуть соответствующие данные к состоянию до использования.

При реализации функций с избыточностью их выполнение необходимо синхронизировать для корректного использования ресурсов, к которым можно обратиться только последовательно.

4.5.4 Политики безопасности, которые могут быть реализованы с использованием резервирования

Избыточность может поддерживать доступность функций и данных так же хорошо, как и целостность данных. Резервное хранение данных может решить проблемы, связанные с ошибками хранения на отдельных секторах диска или всего диска, но оно также может быть частью плана аварийного восстановления, когда данные хранятся с избыточностью в разных местах. Резервирование также может способствовать доступности, сохраняя копию данных, которые часто используются или могут быть использованы в течение короткого периода времени, в хранилище, к которому можно получить доступ быстрее, чем к обычному хранилищу.

Избыточность также может использоваться для минимизации ущерба от атаки, если эта атака нацелена только на один из резервных механизмов. Это можно использовать в тех случаях, когда только один аспект функции механизма откажет в результате атаки. В таких случаях избыточность может использоваться как механизм определения того, что что-то подвергается атаке, обнаружения атакуемой функции или механизма и выполнения необходимых действий.

4.5.5 Примеры

RAID (избыточный массив независимых дисков) — хороший пример технологии, использующей резервирование для повышения надежности и доступности дискового хранилища. Другим примером является кластерное решение с высокой доступностью, в котором при обнаружении отказа элемента кластера задачи, выполняемые отказавшим элементом кластера, передаются другому элементу кластера.

В обоих примерах избыточность используется в сочетании с обнаружением ошибок/сбоев, приостановкой использования поврежденных элементов до тех пор, пока они не будут исправлены, и механизмом перезапуска, который позволяет автоматически синхронизировать исправленный элемент до

состояния, когда он снова полностью функционирует и система достигает исходного уровня избыточности.

Как упоминалось ранее, избыточность также может использоваться для обнаружения злонамеренного изменения функционала. Примером может быть, например, независимая реализация криптографического алгоритма в двух различных доменах, которые было бы трудно атаковать одновременно. Если одна реализация была злонамеренно изменена, и, если вычисление криптографической функции осуществляется в обоих местах, либо для каждого вызова, либо для каждого *n*-го вызова, такая атака будет обнаружена на ранней стадии и необходимые действия могут быть предприняты своевременно.

Важнейшие аспекты избыточности в системах:

- механизм обнаружения ошибок, который должен обнаруживать ошибку до того, как она приведет к потенциально критическим последствиям;
- механизмы, обеспечивающие непрерывную надлежащую работу системы даже при возникновении ошибки;
- установка нового компонента взамен неисправного.

Возможны ситуации, когда обнаружение злонамеренно измененного компонента практически невозможно. Примером является функция генерации случайных чисел, которой можно манипулировать таким образом, что злоумышленник знает генерируемые случайные числа, хотя они и проходят все возможные статистические тесты. В этом примере использование избыточных, но независимых генераторов случайных чисел может уменьшить уязвимость, которой подвергается манипулируемый генератор случайных чисел. В случаях, когда случайные числа требуются для использования только один раз, или в случае генерации криптографических ключей использование одного и того же генератора случайных чисел для обеих целей может привести к уязвимости, когда злоумышленник получает информацию одноразово. Поскольку одноразовые числа обычно пересылаются в открытом виде как часть криптографического протокола, злоумышленник может получить информацию о состоянии манипулируемого генератора случайных чисел. Эта информация может позволить злоумышленнику вычислить пул энтропии, используемый генератором случайных чисел, или значительно облегчить эту задачу. Использование разных независимых генераторов случайных чисел для разных целей, а также использование разных независимых источников энтропии для каждого генератора случайных чисел могут быть эффективными механизмами противодействия угрозе злонамеренного повреждения единственного источника энтропии или генератора случайных чисел.

4.5.6 Замечания по оценке системы резервирования

Системы резервирования часто предназначены для устранения ошибок и неполадок в компонентах, которые возникают не часто или случайно. Второе допущение состоит в том, что эти ошибки независимы друг от друга. Если оба эти условия не выполняются, резервирование не сможет решить проблему появления неполадок в компонентах.

Другой аспект избыточности — это защита от преднамеренных атак, направленных только на одну из избыточных функций или механизмов. Такое резервирование позволяет обнаружить потенциальное повреждение или злонамеренное управление и исправить его, либо отключив неисправную функцию, либо модифицировав ее, либо перезагрузив функцию, чтобы избавиться от манипуляции.

Поэтому при оценке систем и продуктов важно, чтобы оценщик определил тип ошибок, которые, как предполагается, устраняются избыточностью, и проверил:

- что эти ошибки могут возникать только на случайной основе;
- что эти ошибки возникают не часто при всех возможных допустимых эксплуатационных условиях;
- что возникновение одной такой ошибки не зависит от возникновения другой ошибки.

Проверка последнего условия требует тщательного анализа потенциальных зависимостей между избыточными компонентами, которые могут привести к ошибкам, возникающим одновременно в нескольких компонентах. Общий код в избыточных компонентах — это пример потенциально критических взаимозависимостей, когда один и тот же недостаток в коде может потенциально вызвать сбой более чем одного избыточного компонента почти одновременно, особенно когда один компонент заменяет вышедший из строя другой компонент и работает с теми же данными, которые вызвали отказ первого. Вероятность отказа второго компонента очень высока, если оба компонента работают с одним и тем же кодом.

С момента обнаружения ошибки до момента замены неисправного компонента система работает с пониженным уровнем избыточности, и ошибка, возникающая на этом этапе, может иметь критические последствия.

Такой режим функционирования дает возможность злоумышленникам попытаться вызывать ошибки достаточно часто для того, чтобы спровоцировать возникновение хотя бы одной ошибки в такой критической фазе, приводящей (в конечном итоге) к отказу в обслуживании, который должна была предотвратить реализованная избыточность.

Таким образом, специалисты по оценке должны проанализировать тип ошибок, которые должна предотвращать избыточная функциональность, и убедиться, что эти ошибки не могут возникать чаще, чем ожидается злоумышленником. В противном случае реализованная избыточность может оказаться не в состоянии обеспечить ожидаемый уровень защиты.

4.6 Виртуализация

4.6.1 Общие положения

Виртуализация — это архитектурный принцип, который эмулирует функциональность реального или гипотетического устройства, процессора, системы и т. д. на другом устройстве. Это позволяет ожидать, что программное обеспечение, разработанное ранее для невиртуализированного компонента, будет выполняться без изменений или с небольшими изменениями на уровне, обеспечивающем виртуализацию, или абстрагирование от сложной функциональности реальных компонентов.

Таким образом, виртуализация позволяет обеспечить общий программный интерфейс на двоичном уровне даже при использовании различных платформ или элементов платформы. Поэтому виртуализация является одной из ключевых технологий, когда программа во время исполнения должна быть независимой от реальной платформы, на которой она выполняется.

Виртуализация устройств распространена для дисков, видеоустройств и других устройств ввода—вывода. Она позволяет использовать некоторые упрощенные интерфейсы для работы с этими компонентами, оставляя обработку особенностей конкретных реальных устройств уровню виртуализации.

Виртуализация процессоров и платформ используется сегодня, чтобы позволить нескольким «виртуальным машинам» работать на одной платформе. Такая архитектура предоставляет более строгое разделение, чем то, которое обеспечивается разделением процессов и адресного пространства в операционных системах общего назначения. Она также позволяет таким операционным системам выполняться на новой платформе без необходимости использования новых или модифицированных функциональных возможностей, предоставляемых этой платформой (при условии, что уровень виртуализации эмулирует версию платформы, поддерживаемую операционной системой).

Виртуализация также позволяет пользователю на уровне виртуализации моделировать больше компонентов, чем доступно в реальной системе. Одним из первых приложений виртуализации была реализация виртуальной памяти, позволяющая приложениям использовать больше оперативной памяти, чем предоставлено реальной системой.

4.6.2 Принципы определения виртуализации

Виртуализация — это метод, который обеспечивает уровень абстракции, который может быть «вставлен» между двумя существующими уровнями, позволяя более высокому уровню работать без изменений или с незначительными изменениями на нижнем уровне. Уровень виртуализации часто абстрагируется от свойств нижнего уровня, что позволяет упростить верхний уровень. Виртуализация также часто используется для определения нескольких «виртуальных машин» или «виртуальных ресурсов/устройств» и сопоставления их с одной реальной машиной или одним ресурсом/ устройством.

4.6.3 Политики безопасности, которые могут быть реализованы с помощью виртуализации

С точки зрения безопасности виртуализация — это прежде всего механизм, позволяющий осуществлять разделение, дополнительный контроль доступа и прозрачную реализацию дополнительных функций безопасности, таких как службы шифрования. Поэтому в большинстве случаев виртуализация реализуется как уровень (или часть уровня), который работает между исполняемой программой и базовой платформой, который преобразует запросы приложения ресурсов виртуальной платформы в запросы ресурсов реальной платформы.

С точки зрения безопасности виртуализация — это архитектурный принцип, который может быть использован для реализации ряда различных функций и механизмов безопасности способом, понятным для приложения, выполняющегося на уровне виртуализации. Разделение различных приложений является наиболее очевидным, но на уровне виртуализации могут быть также реализованы контроль доступа, прозрачное шифрование, избыточность, функции аудита безопасности, а также аутентификация внешних объектов. Уровни виртуализации также могут предоставлять «классические» функции безопасности, такие как аутентификация пользователей, контроль доступа и аудит.

4.6.4 Примеры

Виртуализация устройств, например, позволяет имитировать несколько устройств, которые в действительности соответствуют одному устройству, и, таким образом, позволяет совместно использовать такое устройство управляемым образом. Один настоящий жесткий диск может, например, использоваться для реализации нескольких виртуальных жестких дисков, позволяя пользователю такого виртуального жесткого диска иметь полный контроль над этим виртуальным устройством, не вмешиваясь в работу других виртуальных жестких дисков, реализованных на этом же физическом диске.

Виртуальная память позволяет приложению работать так, как если бы ему был выделен большой фрагмент реальной памяти, фрагмент, который может быть даже больше, чем реальная память, доступная в базовой системе. Уровень виртуализации в этом случае использует резервное хранилище на жестких дисках или другом устройстве хранения, задействованном в качестве резервного хранилища, и использует стратегию подкачки, чтобы эти части виртуальной памяти поддерживались реальной памятью, которую приложение использует в настоящее время.

Примером использования виртуализации для эмуляции всей платформы может служить гипервизор. Такая технология позволяет программному обеспечению, написанному для этой платформы, выполняться на уровне виртуализации без изменений или с некоторыми простыми изменениями для адаптации к эмулируемой платформе. Платформа, на которой развернут гипервизор, может быть идентичной или очень похожей на платформу, которая эмулируется (самовиртуализация), а может и полностью отличаться (кроссплатформенная виртуализация).

4.6.5 Замечания по оценке уровня виртуализации

В зависимости от своего назначения уровень виртуализации может реализовывать ряд функций и свойств безопасности. Если несколько ресурсов сопоставлены с одним-единственным, отделение виртуальных ресурсов друг от друга имеет критическое значение, и применяются те же аспекты, что и для разделения на домены.

В большинстве случаев уровень виртуализации также реализует функцию эмуляции, которая сопоставляет виртуальные ресурсы с реальными. Например, гипервизору необходимо сопоставить виртуальные процессоры с реальными, виртуальную память с реальной и виртуальные устройства с реальными и гарантировать, что различные виртуальные машины отделены друг от друга (за исключением явного разрешенного совместного использования ресурсов). Таким образом, гипервизор очень похож на операционную систему, которая отделяет адресные пространства друг от друга. Поэтому оценка гипервизора очень похожа на оценку операционной системы в отношении свойств собственной защиты и отделения различных виртуальных машин друг от друга.

Тем не менее существует один аспект, специфичный для гипервизоров, которым часто пренебрегают: необходимо убедиться, что на виртуальной машине не произойдет повышения привилегий, если программное обеспечение, выполняемое на базе виртуальной машины, будет работать непосредственно на реальном оборудовании. Такое повышение привилегий может произойти, если эмулируемые ресурсы не имеют точно таких же свойств, как реальные.

5 Принципы проектирования

5.1 Общие положения

В этом разделе представлен список принципов проектирования, которые могут быть использованы вместе с принципами построения архитектуры, описанными в предыдущем разделе, для проектирования систем, продуктов или приложений. Их использование помогает удостовериться, что продукты, системы или приложения, реализованные с использованием соответствующей комбинации этих принципов проектирования, обеспечивают выполнение заявленных свойств безопасности и способны противостоять атакам. Некоторые из этих принципов проектирования тесно связаны с конкретными свойствами безопасности, в то время как другие могут применяться в целом. Для каждого элемента списка в соответствующем разделе описано, связан ли принцип проектирования с конкретными свойствами безопасности, что позволяет выбрать соответствующие принципы проектирования в отношении свойств безопасности, которым продукт, система или приложение должны удовлетворять.

5.2 Список принципов проектирования для обеспечения безопасности

5.2.1 Принцип наименьших привилегий

Принцип наименьших привилегий — это принцип назначения пользователю, приложению или компоненту системы только тех привилегий, которые необходимы для выполнения поставленной перед ними задачи. Применение этого принципа требует наличия набора привилегий, которые могут быть назначены объектам (статическим или динамическим), для ограничения доступа объектов к данным или функциям.

5.2.1.1 Описание принципа наименьших привилегий

Принцип наименьших привилегий требует, чтобы пользователям, приложениям или системным компонентам был назначен набор привилегий, который не может быть расширен без явных контролируемых административных действий или с помощью четко определенной и контролируемой политики. Автоматическое назначение привилегий может быть полезным, когда привилегии, в которых нуждается сущность, могут быть определены с помощью четко сформулированной политики на основе состояния системы или продукта.

5.2.1.2 Свойства безопасности, поддерживаемые принципом наименьших привилегий

Наличие точной детализации привилегий, которые могут быть назначены индивидуальным пользователям, приложениям или компонентам, является необходимым условием для использования принципа наименьших привилегий. Такой набор привилегий затем может использоваться для детализированных административных ролей, где каждый тип административного действия относится к выделенной привилегии, которая может быть назначена пользователям.

Кроме того, для приложений применение принципа наименьших привилегий может повысить безопасность. Ограничение доступа приложений только к необходимым файлам и сетевым службам препятствует тому, чтобы дефект или вредоносный код, внедренный в приложение, могли получить доступ, повредить другие файлы или использовать службы, не разрешенные для приложения. Это, конечно, требует, чтобы приложение было отделено от других приложений (например, как отдельные процессы) и чтобы оно выполнялось в среде, которая позволяет назначать индивидуальные права приложению (например, чтобы приложение выполнялось с выделенным идентификатором, которому могут быть назначены свои привилегии).

Ограничение набора привилегий до минимального уровня требует обнаружения ошибок, а также ситуаций, когда недоверенные пользователи или недоверенный код получают доступ к ресурсам, к которым эта сущность не имеет доступа. Поэтому защита критически важных ресурсов, а также раннее обнаружение ошибок или критического поведения являются важными свойствами безопасности, которые поддерживаются применением принципа наименьших привилегий.

5.2.1.3 Аспекты/зависимости, которые необходимо учитывать при использовании принципа наименьших привилегий

Существует компромисс между степенью детализации привилегий и сложностью управления ими. Небольшой набор привилегий может потребовать предоставить сущности больше возможностей, чем необходимо для выполнения ее задачи. В крайнем случае, когда все административные привилегии объединены в одну (например, если есть только суперпользователь), единственный вариант для сущности, которой необходимо выполнить какое-либо действие по администрированию, — это назначить эту самую «суперпривилегию».

Большой набор привилегий может снизить этот риск, но обычно такой набор значительно увеличивает сложность управления привилегиями. Модель привилегий может стать настолько сложной, что будет трудно обнаружить возможности по повышению привилегий, т.е. когда определенный набор привилегий, назначенных сущности, позволяет этой сущности получить привилегию, не назначенную ей, в обход средств управления.

Существуют архитектурные принципы, которые могут помочь в создании детализированных моделей привилегий, которыми также не слишком сложно управлять. Инкапсуляция объектов может использоваться как метод объединения привилегий, связанных с операциями над объектом, с самим объектом, а также для определения методов управления этими привилегиями как часть инкапсуляции. Этот метод позволяет использовать наборы гибких, расширяемых и детализированных привилегий, которые ограничены для конкретных объектов, которые назначаются и управляются с помощью объекта.

5.2.1.4 Примеры

Примерами применения принципа наименьших привилегий являются:

- ограничение доступа к сетевым сервисам только теми, которые необходимы для работы системы или приложения;

- ограничение доступа пользователя или приложения к файлам только теми, которые необходимы для работы, и ограничение режимов доступа (чтение/запись/выполнение) только одним необходимым;
- связывание отдельных операций администрирования с привилегиями и назначение этих привилегий административным пользователям в отношении административных задач, которые они должны выполнять;
- временное лишение привилегий при выполнении ненадежного или потенциально опасного кода;
- получение привилегий исходя из состояния системы или продукта и автоматическое назначение привилегий на основе четко определенной политики организации рабочего процесса.

5.2.1.5 Замечания по оценке привилегий

Модели назначения привилегий могут стать очень сложными при увеличении количества и степени детализации привилегий. Хотя большое количество привилегий может позволить ограничить сущность только теми, что ей необходимы для выполнения своей задачи, управление большими и сложными наборами привилегий также часто становится очень сложным. В таких сложных моделях возможно повышение привилегий в случаях, когда сущность может использовать набор назначенных ей привилегий для получения одной или нескольких дополнительных, не назначенных ей администратором или политикой системы. Простой пример — пользователь, получивший право сбрасывать пароли других пользователей (как своего рода функция службы поддержки). Если такой пользователь может сбросить пароль другого пользователя с дополнительными привилегиями, он может выдать себя за этого пользователя и, таким образом, использовать привилегии этого пользователя.

Повышение привилегий не всегда так просто и очевидно, как в примере с функцией службы поддержки. Атаки повышения привилегий могут иметь довольно сложные и неочевидные пути атаки. Следовательно, для оценки необходимо восстановить структуру привилегий в системе и способы управления ими, а также определить способы повышения привилегий, от которых система или продукт должны быть защищены. Важно отметить, что многие политики управления привилегиями не предназначены для запрета всех методов повышения привилегий. Вместо этого сущности с определенными привилегиями или наборами привилегий должны быть доверенными, чтобы не злоупотреблять ими. Для оценки важно определить привилегии, которые требуют безусловного доверия к сущностям, и либо убедиться, что это задокументировано как допущение (в случае, если сущность является внешней по отношению к системе или продукту), либо что это безусловное доверие принимается во внимание при оценке, если сущность является внутренней по отношению к системе или продукту.

В конечном итоге оценка должна показать, что либо повышение привилегий невозможно, либо возможно только для доверенных сущностей (которые, как предполагается или демонстрируется, не злоупотребляют своими привилегиями).

5.2.2 Минимизация поверхности атаки

5.2.2.1 Общие положения

Поверхность атаки системы или приложения характеризуется набором интерфейсов/служб, которые могут быть использованы недоверенными и потенциально опасными сущностями для запуска атаки. Сведение к минимуму этого набора важно для снижения вероятности успешного запуска атаки.

5.2.2.2 Описание принципа минимизации поверхности атаки

Разработчики склонны полагать, что большое количество интерфейсов предоставляет больше гибкости. С другой стороны, злоумышленники предпочитают большое количество интерфейсов, которые они могут использовать для потенциальной атаки, поскольку это увеличивает вероятность найти интерфейс, пригодный для нее. Поэтому, по соображениям безопасности, лучше минимизировать количество интерфейсов, к которым потенциальный злоумышленник может получить доступ, уменьшить сложность этих интерфейсов и защитить их использование для своевременного обнаружения потенциальных злоупотреблений или попыток атаки. Такой подход требует анализа того, какие интерфейсы являются частью поверхности атаки, какого типа злоумышленники могут эксплуатировать эти интерфейсы и какие навыки и мотивацию могут иметь эти злоумышленники. Другими словами, необходимо провести анализ рисков для интерфейсов, доступных потенциальному злоумышленнику, и впоследствии определить необходимые функции защиты для использования этих интерфейсов. Когда все выполнено правильно, включить интерфейс в поверхность атаки является более затратным по сравнению с проведением анализа и защитой механизма, выполняемыми в результате анализа риска, и поэтому уменьшение поверхности атаки обеспечивает также экономические преимущества.

5.2.2.3 Свойства безопасности, поддерживаемые принципом минимизации поверхности атаки

Минимизация поверхности атаки поддерживает свойство безопасности — собственную защиту. Небольшая поверхность атаки позволяет проводить более тщательную оценку дефектов и потенциаль-

ных атак, тем самым снижая риск появления нераспознанных уязвимостей, которые злоумышленник может использовать как часть своей атаки. В частности, можно значительно сократить дефекты, связанные с поверхностью атаки такие как некорректная или неполная проверка параметров.

Чтобы уменьшить размер поверхности атаки, необходимо проанализировать функции, реализованные в компонентах продукта или системы, которые являются критичными для безопасности, которые также могут быть реализованы вне критичных компонентов. Например, когда для доступа к объекту требуются сложные функции преобразования для того, чтобы объект был идентифицирован (например, потому, что требуется поддержать несколько путей, ведущих к объекту).

5.2.2.4 Аспекты/зависимости, которые необходимо учитывать при использовании принципа минимизации поверхности атаки

Уменьшение количества интерфейсов на поверхности атаки — это один аспект, снижение сложности этих интерфейсов — другой. Оба аспекта необходимо учитывать для минимизации поверхности атаки.

Метод, который также можно использовать для уменьшения размера поверхности атаки, заключается в деактивации интерфейсов, которые не требуются для конкретного экземпляра продукта или для конкретной системы. Поэтому разработчики продукта должны предусмотреть способ, позволяющий сделать интерфейсы временно или постоянно недоступными для определенной конфигурации продукта, тем самым позволяя пользователям продукта минимизировать поверхность атаки в соответствии с их эксплуатационными потребностями.

5.2.2.5 Примеры

Пример 1 — Сетевые интерфейсы

Существует два простых метода минимизации поверхности атаки для сетевых интерфейсов:

- уменьшение количества открытых сетевых портов до минимального, необходимого для работы;
- туннелирование как можно большего числа открытых портов с использованием криптографических протоколов, таких как IPSec или TLS.

Первый аспект уменьшает общее количество сетевых интерфейсов, второй — ограничивает набор возможных злоумышленников теми, кто владеет ключами для установления зашифрованного соединения.

Пример 2 — Интерфейсы операционной системы

Операционная система общего назначения обычно предлагает различные наборы интерфейсов, которые включают системные вызовы, сетевые интерфейсы, привилегированные программы, а также интерфейсы конфигурации и управления. Нередко возникают следующие проблемы:

- операционная система предлагает разные системные вызовы для очень похожих функций (часто, чтобы не нарушать совместимость со старыми версиями);
- операционная система предлагает привилегированные программы для функций, которые не требуют привилегий или когда набор привилегий программы значительно больше, чем требуется (нарушение принципа наименьших привилегий);
- операционная система недостаточно ограничивает доступ к интерфейсам конфигурации и управления, позволяя потенциальным злоумышленникам получить доступ к этим интерфейсам.

Разработчик операционной системы при внедрении нового интерфейса на поверхность атаки или изменении такого интерфейса должен учитывать следующие аспекты:

- необходим ли новый интерфейс, или эта функция также может быть вызвана с помощью существующих интерфейсов, возможно, с небольшими изменениями;
- требуются ли все привилегии, назначенные привилегированной программе, или набор привилегий может быть уменьшен без ущерба для функциональности;
- можно ли полностью удалить интерфейс с поверхности атаки, ограничив доступ к нему только доверенным сущностям.

5.2.2.6 Замечания при оценке поверхности атаки

Выявление поверхности атаки — один из важных шагов, который необходимо выполнить при оценке. Оценщику необходимо идентифицировать все интерфейсы и их параметры, к которым потенциальный злоумышленник может получить прямой доступ или, по крайней мере, повлиять на них. Это первый шаг в конструктивном анализе уязвимостей. Очевидно, что общий объем работ по анализу уязвимостей прямо пропорционален размеру поверхности атаки, и поэтому небольшая поверхность атаки уменьшит трудозатраты по оценке в области анализа уязвимости.

Само определение поверхности атаки часто бывает довольно трудным, поскольку поверхность атаки может сильно отличаться от программных или пользовательских интерфейсов, описанных разработчиком. Во многих случаях программные или пользовательские интерфейсы, описанные разработчиком, являются просто интерфейсами к функциям-оболочкам, которые затем сами вызывают интерфейсы на поверхности атаки. Часто эти оболочки можно легко обойти, что делает ограничения, налагаемые этими оболочками, неэффективными.

Поэтому оценщик должен определить поверхность атаки как самый низкий уровень интерфейса продукта или системы, к которому потенциальный злоумышленник может получить доступ. Зачастую это требует доступа к информации, которую разработчик не желает раскрывать. Тем не менее эта информация должна быть доступна в качестве внутренней проектной документации от разработчика, и любая конструктивная оценка должна основываться на этой документации, чтобы определить поверхность атаки и начать анализ потенциальных уязвимостей.

5.2.3 Централизованная проверка параметров

5.2.3.1 Общие положения

Большое количество уязвимостей, о которых сообщается в репозиториях уязвимостей, вызвано неполной или некорректной проверкой параметров. Атаки переполнения буфера или инъекции — это лишь два примера атак, вызванных неполной проверкой параметров. Поэтому очень важно обеспечить полную и точную валидацию параметров. Для достижения этой цели необходимо наличие общих функций, проверяющих параметры, которые позволяли бы проводить единый комплексный анализ этих функций на точность и полноту.

5.2.3.2 Описание принципа централизованной проверки параметров

Централизованная проверка параметров должна гарантировать, что все параметры критических функций проверяются с использованием общего набора функций проверки, которые были тщательно проанализированы на точность и полноту. Конечно, существуют особенности проверки параметров в зависимости от используемого языка, типа интерфейса и типа параметров. Поэтому единая реализация проверки параметров часто оказывается невозможной.

Тем не менее большинство систем и продуктов имеют наборы интерфейсов, в которых проверка параметров имеет достаточно общих аспектов, чтобы с ними можно было работать каким-то централизованным образом. Этот централизованный метод может быть единственным примером кода, но чаще всего это одна библиотека, к которой код обращается по мере необходимости.

Точная и полная проверка параметров особенно важна для интерфейсов, которые принадлежат поверхности атаки. На этих интерфейсах нельзя ожидать, что вызывающий их будет придерживаться правил, определенных для параметров. Вместо этого злоумышленник может намеренно вызвать эти интерфейсы с недопустимыми или перепутанными значениями параметров.

5.2.3.3 Свойства безопасности, поддерживаемые принципом централизованной проверки параметров

Централизованная проверка параметров — важная мера собственной защиты, поскольку при правильном применении она снижает вероятность появления одного из основных источников уязвимостей.

Довольно часто параметры интерфейса, принадлежащего поверхности атаки, целенаправленно создаются злоумышленником для тестирования или эксплуатации потенциальных уязвимостей.

5.2.3.4 Аспекты/зависимости, которые необходимо учитывать при использовании принципа централизованной проверки параметров

Как упоминалось выше, для централизованной проверки параметров требуются наборы интерфейсов, которые могут использовать общую методологию проверки параметров. Во многих случаях это связано с используемой техникой реализации (например, общий язык программирования) или используемой платформой. Часто требуется определение некоторых общих соглашений о разработке/внедрении, таких как максимальная длина строки, определенные форматы, или недопустимые символы и выражения в строках, или другие входные параметры. Эти соглашения должны быть определены глобально, и все разработчики должны быть осведомлены об этих ограничениях. Иногда может потребоваться изменение или снятие ограничений, но тогда эти изменения необходимо обосновать и принять, прежде чем в соответствии с ними будут определены функции централизованной проверки параметров.

5.2.3.5 Примеры

Межсетевые экраны, которые выполняют фильтрацию пакетов или проверку протоколов на уровне приложений, являются одним из примеров службы централизованной проверки параметров, обыч-

но выполняющей также некоторые дополнительные функции. Другой пример — специализированная библиотека, которую разработчики должны использовать для проверки параметров при реализации критических функций с использованием определенного языка программирования.

5.2.3.6 Замечания по оценке централизованной проверки параметров

Централизованная проверка параметров может помочь снизить затраты на оценку. Помимо анализа точности и полноты проверки параметров оценщику необходимо рассмотреть следующие аспекты:

- реализована ли проверка параметров как часть доверенного кода продукта или системы. В противном случае эту функцию можно легко обойти или нейтрализовать;
- можно ли обойти проверку параметров, например с помощью интерфейсов, которые не вызывают централизованную проверку параметров, например, из-за другого сетевого пути, который обходит централизованную проверку параметров, или из-за того, что функции проверки параметров реализованы с использованием другой технологии, в которой централизованная проверка параметров — как реализовано — не может быть выполнена;
- охватывает ли централизованная проверка параметров все параметры интерфейса;
- существуют ли ограничения в интерфейсе или параметры интерфейса, требующие дополнительных проверок, не входящих в централизованную проверку параметров. В этом случае оценщику необходимо убедиться, что эти дополнительные ограничения реализуются самой функцией.

5.2.4 Централизованные общие службы безопасности

5.2.4.1 Общие положения

Конкретные службы безопасности носят общий характер и используются несколькими компонентами продукта или системы. Предоставление этих служб базовым компонентом продукта или системы упрощает задачу и позволяет располагать обширными знаниями о различных подводных камнях при реализации этой функции.

Существует ряд служб безопасности, которые предоставляются либо базовой платформой (например, операционной системой), либо некоторой центральной службой в рамках инфраструктуры системы. Если такие службы уже обеспечиваются средой продукта или системы, то их следует использовать, вместо того чтобы реализовывать отдельно.

5.2.4.2 Описание принципа централизации общих служб безопасности

Централизованные службы безопасности обеспечивают значительные преимущества не только с точки зрения реализации (одна служба вместо множества различных), но и с точки зрения управления. Если такие службы, как аутентификация пользователей, контроль доступа или аудит, будут централизованными, ими можно будет управлять единообразно и их не нужно будет синхронизировать.

Хорошими кандидатами в централизованные службы безопасности являются:

- идентификация и аутентификация пользователей, управление привилегиями пользователей;
- контроль доступа;
- сбор и оценка записей аудита;
- криптографические службы;
- мониторинг и управление безопасностью.

5.2.4.3 Свойства безопасности, поддерживаемые принципом централизации общих служб безопасности

Централизованные службы безопасности могут гарантировать согласованную работу служб, предоставляемых в более крупных системах, и могут упростить управление сложными системами.

5.2.4.4 Аспекты/зависимости, которые необходимо учитывать при использовании принципа централизации общих служб безопасности

Использование централизованных служб безопасности подразумевает, что пользователи этих служб могут безопасно получать к ним доступ и использовать их. Такой подход требует, чтобы эти службы были защищены от несанкционированного доступа и гарантировали защиту критически важных данных от несанкционированного раскрытия и изменения. Также необходимо учитывать доступность этих служб, особенно когда нет безопасной резервной функциональности на случай, если служба безопасности недоступна.

5.2.4.5 Примеры

Существует ряд служб безопасности, которые часто реализуются как централизованные службы безопасности с использованием, например, службы каталогов, службы PKI (служба открытых ключей), хостов журналов, а также служб управления сетями и системами.

Предоставление конкретных функций, которые сложно реализовать безопасным образом, — очень важный аспект. Криптографические функции — оптимальный тому пример. Хотя кажется, что правиль-

но реализовать конкретный криптографический алгоритм достаточно просто, практика показала, что реализовать его таким образом, чтобы он не имел эксплуатируемых сторонних каналов, довольно трудно. Со временем были разработаны особые методы для эффективного противодействия таким атакам сторонних каналов. В то время как атаки по сторонним каналам, основанные на анализе возможностей или вызове неисправностей, могут быть опасными для систем, работающих в определенных средах с ограниченной физической безопасностью, сторонние каналы синхронизации представляют собой угрозу, которая существует почти во всех системах. Упрощенные реализации криптографических функций почти всегда приводят к утечке информации из-за разного времени синхронизации для разных ключей, разных сообщений или обработки ошибок. В крайнем случае такие сторонние каналы синхронизации пропускают критически важную информацию и позволяют злоумышленнику получить данные, которые должны были быть защищены с помощью криптографических функций.

Поэтому полезно предоставлять криптографические функции как сервис, который может использоваться различными «клиентами», и реализовать в них все функции защиты, необходимые для того, чтобы сделать атаки по сторонним каналам невозможными или настолько трудными для эксплуатации, что злоумышленник, скорее всего, не будет тратить на это свои ресурсы.

5.2.4.6 Замечания по оценке централизованных служб безопасности

При оценке централизованных служб безопасности оценщик должен убедиться, что эти сервисы предоставляют интерфейсы, которые могут быть безопасно использованы, т. е. которые обеспечивают требуемую защиту данных, передаваемых через эти интерфейсы, гарантируют, что служба может использоваться только авторизованными в службе сущностями и что специализированные функции и данные, которые они используют, защищены от злонамеренного использования этими авторизованными сущностями, и сама служба доступна, когда это необходимо.

При оценке также всегда следует иметь в виду, что такие централизованные службы безопасности сами являются объектом пристального внимания потенциального злоумышленника. Взлом такого централизованного компонента может представлять гораздо больший интерес для злоумышленника, чем взлом многих децентрализованных сервисов. Поэтому оценщик должен принять во внимание, что такая централизованная служба требует защиты от злоумышленников со значительно более высоким потенциалом нападения. Поэтому для таких централизованных служб может потребоваться применение других принципов архитектуры и проектирования, таких как избыточность и минимизация поверхности атаки. Кроме того, в большинстве случаев требуется более высокий уровень доверия.

5.2.5 Подготовка к обработке ошибок и исключительных ситуаций

5.2.5.1 Общие положения

Никакая система или продукты не могут гарантировать отсутствие ошибок. Кроме того, необходимо определить конкретные условия эксплуатации или сбои в среде функционирования продукта или системы, чтобы можно было принять корректирующие меры.

Следовательно, продукты и системы должны быть реализованы таким образом, чтобы позволять обнаруживать ошибки или исключительные ситуации, сообщать о них и предпринимать автоматические действия либо для устранения, либо для минимизации последствий ошибки.

5.2.5.2 Описание принципа обработки ошибок и исключительных ситуаций

Для подготовки к обработке ошибок и исключительных ситуаций необходимо выполнить следующие действия:

- идентификация возможных ошибок и исключительных ситуаций, требующих особой обработки (этот шаг необходимо выполнить на этапе определения требований к продукту/системе);
- спецификация функциональных возможностей, необходимых для обнаружения ошибок или исключительных ситуаций;
- спецификация функций обработки ошибок исключительных ситуаций;
- спецификация шагов для возврата из состояния обработки ошибок/исключительных ситуаций для возобновления нормальной работы или для корректного завершения работы выделенных функций или всей системы/продукта.

5.2.5.3 Свойства безопасности, поддерживаемые принципом обработки ошибок и исключительных ситуаций

Обработка ошибок и исключительных ситуаций должна гарантировать, что влияние ошибок или исключительных ситуаций на безопасность системы сведено к минимуму. В лучшем случае они должны гарантировать, что система или продукт всегда находится в «безопасном состоянии» (работает в соответствии с целями безопасности), даже в случае возникновения специфичных ошибок или исключительных ситуаций. Если это невозможно, система должна проанализировать влияние ошибки или

исключительные ситуации и предпринять наименее критичные действия, возможно, с помощью уполномоченных администраторов.

Избыточность часто используется как средство исправления ошибок. Во многих случаях избыточность используется для обнаружения ошибки, а также для определения корректирующих действий, если правильные значения могут быть извлечены из резервных данных.

5.2.5.4 Аспекты/зависимости, которые необходимо учитывать при использовании принципа обработки ошибок и исключительных ситуаций

В некоторых случаях система или продукт могут отслеживать определенные события, которые могут привести к состояниям, когда с большой вероятностью могут возникнуть ошибки или исключительные ситуации, и предпринимать корректирующие действия еще до их возникновения. Примерами могут служить ситуации, когда система или продукт постепенно достигают своих пределов эксплуатации, например когда температура поднимается выше некоторого порогового значения, нагрузка на систему становится слишком высокой, дисковое пространство переполняется или когда средство обнаружения сетевых вторжений сообщает о большом количестве подозрительной активности, которая сама по себе еще не может считаться вторжением. В этих случаях корректирующие действия могут быть предприняты до перехода в критическое состояние, когда возникнет ошибка или исключительная ситуация.

В случае возникновения ошибки или исключительной ситуации необходимо убедиться, что корректирующие действия по их исправлению могут быть выполнены без вызова другой ошибки или исключительной ситуации. Порождение такой же или подобной ошибки во время реагирования часто имеет критические последствия, например такие, как полное прекращение функционирования продукта или системы.

5.2.5.5 Примеры

Примеры обработки ошибок и исключительных ситуаций:

- действия по обнаружению и исправлению ошибок, определенные для протоколов связи;
- обнаружение ошибки программы, когда она пытается получить доступ к защищенной памяти или обращается к памяти в иной форме, чем ожидалось (например, попытка выполнить инструкцию в области памяти, обозначенной только как данные);
- обнаружение несоответствий в сетевых данных или параметрах конфигурации и информирование администратора.

5.2.5.6 Замечания по оценке обработки ошибок и исключительных ситуаций

Для эффективного использования принципа обработки ошибок и исключительных ситуаций требуется определение перечня ошибок и исключительных ситуаций, которые требуют обработки, способа обнаружения этих ошибок и исключительных ситуаций, а также действий при возникновении таких ошибок или исключительных ситуаций, и определение пути возврата к «безопасному состоянию».

Таким образом, оценщику необходимо проверить соответствие ошибок и исключительных ситуаций, которые должны быть обнаружены, заявленным целям безопасности, гарантируя, что полученный перечень ошибок и исключительных ситуаций достаточен для выполнения целей безопасности.

Что касается процесса обнаружения, оценщику необходимо удостовериться, что все ошибки и исключительные ситуации, требующие обработки, будут обнаружены (т. е. нет никаких экземпляров таких ошибок или исключительных ситуаций, которые не были бы охвачены заданным процессом обнаружения). Особенно следует учитывать случаи, когда ошибки или исключительные ситуации возникают во время обработки ранее возникших.

Как только это будет сделано, оценщик должен убедиться, что обработка ошибок достигает намеченного эффекта независимо от того, в каком состоянии возникает ошибка. Оценщику необходимо также удостовериться, что все действия, предназначенные для обработки ошибок, могут быть выполнены во время этого процесса. Для этого необходимо, чтобы были доступны все ресурсы, необходимые для обработки ошибок. Например, процесс обработки ошибок, который запускается в случае нехватки памяти или дискового пространства, сам по себе не требует выделения памяти или дискового пространства. Обработка ошибок, которая должна управлять перегрузкой системы, должна выполняться с достаточно высоким приоритетом, чтобы гарантировать, что на нее не повлияет сама перегрузка.

5.3 Использование принципов проектирования при проектировании безопасной системы или приложения

5.3.1 Общие положения

В этом разделе описано использование принципов проектирования на примере. Примером является система, которая позволяет пользователю подключаться к приложению через Интернет, просматривать каталог и заказывать товары из каталога.

Система разбита на следующие компоненты:

- фронтальная система, к которой подключается пользователь, которая предоставляет пользовательский интерфейс для просмотра каталога;
- внутренняя система, которая содержит каталог, принимает заказы, проверяет их и помещает заказы в базу данных заказов;
- набор других систем, предоставляющих централизованные службы безопасности.

При проектировании этой системы применяются нижеприведенные принципы проектирования.

5.3.2 Принцип наименьших привилегий

Поскольку фронтальной системе нет необходимости вести каталог или обращаться к базе данных заказов, она настроена таким образом, что у нее отсутствуют привилегии для этих действий, а также нет прав на подключение к любой другой системе во внутренней сети.

Каталог также содержит ограниченные разделы. Они хранятся в отдельной системе (сервер управления доступом), и фронтальная система не имеет прямого доступа к этим разделам.

Процесс во фронтальной системе, который предоставляет интерфейс для внешнего пользователя, сконфигурирован для работы без каких-либо прав на доступ к данным, кроме тех, которые связаны с каталогом и формой для размещения заказов.

Внутренняя система имеет запущенный процесс, который получает информацию о заказе от фронтального сервера. Этот процесс сконфигурирован так, чтобы не иметь никакого другого доступа, кроме доступа к базе данных заказов.

5.3.3 Минимизация поверхности атаки

Фронтальная система сконфигурирована так, чтобы не иметь никаких сетевых портов с выходом в Интернет, кроме портов, необходимых для связи с внешними пользователями. Канал связи с внутренней системой настроен так, чтобы не было никаких открытых портов, кроме тех, которые необходимы для связи с внутренней системой.

Внутренняя система, а также системы, обеспечивающие централизованные службы безопасности, упомянутые ниже, установлены таким образом, что они не имеют прямого подключения к Интернету или к какой-либо внутренней системе, с которой им не нужно взаимодействовать.

5.3.4 Централизованная проверка параметров

При размещении заказа его параметры передаются на централизованный сервер, который проверяет критические параметры (например, информацию о кредитной карте, информацию о пользователе, такую как его имя и адрес). Этот сервер используется несколькими системами, которые позволяют пользователям размещать заказы.

5.3.5 Централизованные службы безопасности

Фронтальная система предоставляет пользователям защищенное соединение TLS, в котором протокол TLS реализован общей библиотекой. Эта библиотека использует централизованного поставщика криптографии для функций асимметричного шифрования. Этот централизованный поставщик криптографии представляет собой выделенный сервер, использующий физически защищенный криптографический сопроцессор. Используемый закрытый ключ хранится только на этом сопроцессоре и поэтому никогда не попадет в память внешнего сервера. Таким образом, даже если внешний сервер будет скомпрометирован, закрытый ключ, используемый при настройке TLS-соединения, не будет скомпрометирован.

Когда фронтальная система также позволяет пользователям аутентифицироваться, она может отправлять учетные данные пользователя на центральный сервер аутентификации. Если пользователь успешно прошел аутентификацию, он может просматривать дополнительные части каталога и размещать заказы на товары, которые могут быть отправлены только аутентифицированным пользователям. Чтобы определить, какие дополнительные части каталога разрешено просматривать пользователю, фронтальная система отправляет запрос на просмотр закрытых частей каталога на централизованный сервер управления доступом, который управляет правами доступа пользователей. Этот сервер централизованного управления доступом обрабатывает запрос и отправляет результаты на внешний сервер.

Такой способ обработки данных запрещает использование взломанной фронтальной системы для доступа к закрытым частям каталога.

5.3.6 Подготовка к обработке ошибок и исключительных ситуаций

И фронтальный, и внутренний сервер используют функции, предоставляемые их операционной системой, чтобы для обработки ошибок и исключительных ситуаций передать управление функциям обратного вызова для любой такой ошибки или исключения, для которых операционная система позволяет устанавливать такие функции обратного вызова. Приложение в обеих системах написано с возможностью перезапуска в контрольных точках, которые приложение устанавливает через определенные промежутки времени. Когда возникает ошибка или исключительная ситуация, приложение проверяет ее возможную причину, а затем определяет:

a) если проблема незначительна и операцию можно перезапустить (после некоторого анализа ошибки или исключительной ситуации и некоторой незначительной очистки), приложение это делает. Это не должно влиять на работу;

b) если проблема более серьезна и возможны варианты: a) дальнейшая работа невозможна, операция может перезагрузиться с последней взятой контрольной точки. Это в некоторой степени влияет на работу (данные, собранные или измененные после прохождения контрольной точки, могут быть потеряны);

c) если проблема еще более серьезна, может потребоваться перезапустить все приложение. Это требует некоторого анализа для перезапуска с использованием безопасного и стабильного состояния. Воздействие на работу может быть более серьезным, и служба может не работать в течение времени, необходимого для перезапуска.

В нашем примере влияние ошибки или исключительной ситуации на внешнюю систему в целом менее критично, так как эта система не обновляет данные. Может быть потеряна только информация, которую пользователь ввел в форму заказа. Восстановление после ошибок или исключительных ситуаций во внутренней системе является более критичным, и процедуры, минимизирующие влияние ошибки или исключительных ситуаций на эту часть системы, должны быть разработаны более детально.

Этот пример показывает, как принципы проектирования могут быть использованы для разработки системы или приложения, обеспечивающего высокий уровень безопасности. Минимизация поверхности атаки затрудняет, но не исключает взлом системы. Принцип наименьших привилегий не позволяет злоумышленнику, успешно взломавшему внешнюю систему, получить прямой доступ к критически важным ресурсам. Функции подготовки к обработке ошибок и исключительных ситуаций могут помочь обнаружить взлом и справиться с ситуацией с минимальным влиянием на работу. Централизованные службы безопасности помогают снизить общую стоимость, но также помогают защитить критически важные ресурсы от злоумышленника. Централизованная проверка параметров также помогает снизить затраты, но еще может минимизировать ошибки, допущенные при реализации проверки параметров.

6 Мероприятия по оценке архитектурных принципов

6.1 Общие положения

В этом разделе объяснены аспекты принципов построения архитектуры, описанных в разделе 5, и семейства требований доверия ADV_ARC, описанного в ISO/IEC 15408-3. Раздел представляет собой руководство, как принципы архитектуры, определенные настоящим стандартом, соотносятся с описанием этого комплекса требований в ISO/IEC 15408-3 и соответствующих шагов оценивания в методологии оценки безопасности информационных технологий, определенной в ISO/IEC 18045. Поэтому в этом разделе используется терминология ISO/IEC 15408 и ISO/IEC 18045.

ISO/IEC 15408 определяет структуру оценки безопасности продуктов ИТ, где фактический продукт, подлежащий оценке, называется «объект оценки» (ОО)[1)]. В этом продукте ИТ-часть, отвечающая за реализацию функций безопасности и обеспечение соблюдения политики безопасности, определенной для продукта ИТ, называется «функциональными возможностями безопасности ОО» (ФБО)[2)].

Критерии и методология оценки, описанные в ISO/IEC 15408 и ISO/IEC 18045, не требуют четкого следования определенным принципам архитектуры или проектирования. Семейство требований доверия ADV_ARC требует описания архитектуры безопасности ФБО части ОО, которая необходима

---

1) *Далее по тексту используется сокращение ОО.*

2) *Далее по тексту используется сокращение ФБО.*

для описания того, как достигаются свойства собственной защиты, разделения на домены и невозможности обхода функций безопасности. Хотя следование определенным принципам проектирования и архитектуры помогает доказать, что эти принципы соблюдаются, они не требуются явно. Принимая во внимание, что ISO/IEC 15408 разработан для применения к широкому спектру различных продуктов, сложно предписывать использование определенных принципов проектирования и архитектуры, к тому же это накладывает ограничения, которые противоречат инновационным разработкам безопасных архитектур и проектированию. Кроме того, семейство ADV_INT требует, чтобы ФБО были «хорошо структурированными» и «не слишком сложными», без каких-либо дополнительных подробностей. В то время как ADV_ARC является частью каждого предопределенного уровня доверия в ISO/IEC 15408, ADV_INT требуется только для более высоких предопределенных уровней доверия.

Тем не менее очевидно, что соблюдение принципов проектирования и архитектуры может помочь сократить трудозатраты на оценку. В описании отдельных принципов проектирования и архитектуры уже описаны некоторые аспекты, которые следует учитывать при оценке. Этот раздел уточняет некоторые из этих аспектов и помещает их в контекст оценки, выполняемой в соответствии с ISO/IEC 15408 и ISO/IEC 18045.

Комитет по разработке общих критериев (CCDB), а также национальные системы сертификации могут выпускать разъяснения или вспомогательные документы, которые относятся к архитектурным и проектным аспектам в целом или для конкретных технологий. Эти разъяснения являются обязательными для систем сертификации, к которым они применяются, и имеют приоритет перед настоящей технической спецификацией.

Архитектурные решения, описанные в этой спецификации, могут (и должны) использоваться в сочетании друг с другом. Архитектурные принципы могут использоваться даже иерархически. Например, домен может состоять из набора поддоменов, может использовать разделение на уровни или инкапсуляцию для внутреннего структурирования, а может использовать методы виртуализации или избыточность. Адекватное использование этих архитектурных принципов в иерархическом порядке может помочь снизить общую сложность ОО и обеспечить более эффективную оценку.

Семейство ADV_TDS в ISO/IEC 15408 требует, чтобы ОО был разбит на подсистемы, а для более высоких уровней также требуется, чтобы эти подсистемы были разбиты на модули. Разработчик имеет некоторую свободу выбора подсистем и модулей до тех пор, пока структура помогает оценщику понять, как реализованы функции безопасности, и позволяет отследить это до самого нижнего уровня проектирования или даже до реализации. Таким образом, структура, представленная оценщику разработчиком, служит главным образом для того, чтобы помочь ему получить детальное представление о том, как были реализованы функции безопасности. Она может отличаться от проектной спецификации, которую должен иметь разработчик и которая больше ориентирована на особенности конкретного компонента, за который разработчик несет ответственность или который он должен поддерживать. Разработчику требуется конкретное локальное представление, тогда как оценщику необходимо иметь более глобальное представление об ОО с точки зрения обеспечения безопасности.

Это может привести к тому, что описание подсистем и модулей, содержащееся в общей проектной документации, представленной оценщику, не соответствует структуре в части деления на домены, уровни или инкапсулированные объекты. Это не является проблемой до тех пор, пока можно проследить соответствие между этими разными представлениями структуры ОО. Оценщику также потребуется описание структуры ОО в части разбиения на домены, уровни и/или инкапсулированные объекты для выполнения анализа уязвимостей.

6.2 Разделение на домены

Концепция разделения на домены упоминается в ISO/IEC 15408-3 в семействе требований доверия ADV_ARC. Из представленной документации оценщику необходимо определить количество и тип доменов, которые предоставляет ОО, назначение каждого домена, механизмы, которые отделяют домены друг от друга, и способ взаимодействия доменов для обеспечения функциональных возможностей ОО. ADV_ARC определяет, что уровень детализации в описании архитектуры должен быть таким же, что и описание, требуемое для семейства требований доверия ADV_TDS. Таким образом, оценщику необходимо ознакомиться с требованиями к этому семейству доверия, чтобы определить, какой уровень детализации и полноты требуется для описания интерфейсов и взаимодействия между доменами. В большинстве случаев структура доменов соответствует декомпозиции ФБО ОО на подсистемы. Это сопоставление не может быть взаимно однозначным, поскольку разработчик может разбить ОО на

подсистемы, состоящие из более чем одного домена, или домены, которые охватывают несколько подсистем. Когда дело доходит до декомпозиции на модули, один домен должен состоять из одного или нескольких модулей. Если разделение на подсистемы и модули трудно соотнести с разделением ФБО на домены, оценщику следует опираться на документацию, которая позволит ему сопоставить подсистемы и (где требуется) модули с доменами. Это важно для понимания того, как разделение на домены влияет на свойства архитектуры: разделение, собственная защита и невозможность обхода.

Различные домены обычно имеют различные привилегии, связанные с безопасностью, и поэтому аспект потенциального повышения привилегий при использовании интерфейсов между доменами и аспект получения или вывода информации, которую домен должен сохранять конфиденциальной, становятся критическими. Поэтому при проведении анализа уязвимостей оценщику необходимо учитывать следующие потенциальные проблемы безопасности:

- можно ли создать такой путь атаки, который использует интерфейсы между различными доменами для повышения привилегий злоумышленника;
- может ли злоумышленник использовать интерфейсы к домену таким образом, чтобы информация, конфиденциальность которой домен должен защищать, утекла к злоумышленнику.

6.3 Разделение на уровни

Разделение на уровни часто используется, чтобы удостовериться, что злоумышленник может обойти определенные функции, реализованные на уровне. Довольно часто функции безопасности, такие как контроль доступа или использование элементов управления, реализуются на определенном уровне, и нижние уровни предполагают, что эти элементы управления уже были применены. В этом случае важно знать, что эти уровни нельзя обойти, и очень важно иметь доказательства, почему. Невозможность обхода — это требование, выраженное в ISO/IEC 15408 в семействе ADV_ARC, и разработчик должен предоставить документацию об архитектуре, которая позволила бы оценщику понять, как принцип разделения на уровни был применен в ОО.

Подобно разделению на домены, разделение на уровни может не соответствовать напрямую разделению на подсистемы и модули. В этом случае оценщик должен опираться на документацию, которая позволяет понять взаимосвязь между подсистемами и модулями и многоуровневой структурой.

Требования к оценке многоуровневой архитектуры, используемой в ОО:

- оценщику необходимо иметь возможность сопоставить многоуровневую архитектуру с доказательствами невозможности обхода, представленными в документации для ADV_ARC, поскольку разделение на уровни используется для обеспечения невозможности обхода ФБО;
- оценщику необходимо понять из документации, представленной в качестве доказательства для ADV_ARC и ADV_TDS, как организовано разделение на уровни.

При проведении анализа уязвимостей оценщику необходимо рассмотреть возможность создания пути атаки, который обходит разделение на уровни (т. е. обходит по крайней мере один уровень), и определить, приводит ли это к нарушению хотя бы одной из целей безопасности, заявленных для ОО.

6.4 Инкапсуляция

Инкапсуляция часто используется как одна из конструкций языка реализации для защиты данных и функций от несанкционированного доступа и использования. Поэтому обеспечение разделения и невозможности обхода обычно выполняется исполняющей системой, которая поставляется с компилятором, используемым для обработки кода. Довольно часто оценщик не имеет доступа к данным об особенностях этой исполняющей системы и должен доверять ей.

Такая конструкция может стать проблематичной, если ФБО создаются с использованием нескольких языков или компиляторов, что приводит к параллельной работе нескольких исполняющих систем. В таких случаях может получиться так, что инкапсуляция, предоставленная одной исполняющей системой, может быть пропущена или даже деактивирована частями кода, сгенерированного с использованием языка программирования, который либо вообще не обеспечивает инкапсуляцию, либо не обеспечивает строгую инкапсуляцию, либо вообще допускает конструкции, которые способны подорвать инкапсуляцию, обеспечиваемую какой-либо исполняющей системой для других языков. В такой среде, где присутствует несколько языков программирования, инкапсуляция, обеспечиваемая конкретными исполняющими системами, всегда находится под угрозой.

Это затрудняет оценку разделения и невозможности обхода ФБО ОО, разработанного с использованием разных языков. Поэтому оценщику необходимо выполнить некоторое исследование в рамках

анализа уязвимости, которое позволит определить возможность такого подрыва используемых методов инкапсуляции, определить возможные кодовые конструкции и способы использования языков программирования, которые потенциально могут подорвать инкапсуляцию, обеспечиваемую другой исполняющей системой, и разработать тесты на проникновение, которые попытаются использовать эти способы для подрыва функций безопасности ОО.

ФБО, которые используют инкапсуляцию, базируются на функциях, предоставляемых исполняющей системой, поэтому оценщику необходимо получить информацию о принципах, как исполняющая система реализует инкапсуляцию и как она защищает инкапсулированные объекты от доступа в обход средств управления, реализованных исполняющей системой. Этот анализ должен выполняться как часть оценки ADV_ARC и должен быть дополнен тестами на проникновение в соответствии с выбранным уровнем AVA_VAN.

В тех случаях, когда инкапсуляция обеспечивается функциями защиты ФБО (например, в случае «setuid» в среде UNIX), оценка должна быть сосредоточена на аспекте получения недоверенного кода, исполняемого в инкапсулированной среде. Одним из примеров такого проникновения является программа, которая запускается в инкапсулированной среде, где правильность и доверенность библиотеки, из которой вызывается программа, не проверяется в достаточной степени.

6.5 Резервирование

Избыточные элементы в архитектуре используются для повышения надежности системы в случае ошибок или сбоев. При использовании таких элементов в целях безопасности также необходимо учитывать аспект ошибок и сбоев, вызванных преднамеренными атаками.

При проведении анализа оценщику необходимо получить убедительную информацию, позволяющую определить, являются ли избыточные элементы достаточно независимыми для защиты от последствий таких ошибок или сбоев. Поэтому в архитектуре, которая включает в себя избыточные элементы, необходимо перечислить типы ошибок и сбоев, от которых избыточность должна защищать, аргументы, почему избыточные элементы являются достаточно независимыми по отношению к этим ошибкам и сбоям, дать описание механизма, используемого для обнаружения ошибок или сбоев, а также описание функций, используемых для реагирования на обнаруженные ошибки или сбои. Эта информация должна содержаться в документации, содержащей требования к компонентам, выбранным из семейств ADV_ARC и ADV_TDS для оценки.

Затем оценщику необходимо проверить:

- что заявленный уровень независимости избыточных элементов верен даже в случае преднамеренной атаки;
- что механизм обнаружения ошибок или сбоев эффективен даже в случае преднамеренной атаки и что ошибки или сбои сами по себе не влияют на этот механизм;
- что реакция на ошибки или сбои не создает дополнительных проблем безопасности (например, упрощает перегрузку системы, тем самым увеличивая вероятность выхода исправного элемента из строя);
- что восстановление нормальной работы возможно в течение предполагаемого периода времени и что злоумышленник не может повлиять на время возврата к нормальной работе недопустимым образом.

Все это выполняется в сочетании с мероприятиями по оценке для ADV_ARC, ADV_TDS и AVA_VAN, где оценка, выполненная для ADV_ARC и ADV_TDS, используется для получения тестов на проникновение, выполняемых для AVA_VAN.

6.6 Виртуализация

Виртуализация всегда связана с эмуляцией виртуализированного объекта. Для оценки безопасности оценщику требуется:

- полная поведенческая спецификация виртуализированного объекта. Это необходимо для выявления любых несоответствий между этой спецификацией и поведением, демонстрируемым уровнем, реализующим эмуляцию;
- описание функций, реализующих функции эмуляции. Это покрывается требованиями к компонентам семейств ADV_ARC, ADV_FSP и ADV_TDS;

- описание какой-либо технической поддержки со стороны ИТ-среды, используемой для виртуализации. Эти поддерживающие функции должны быть описаны достаточно подробно, особенно если используется аппаратная или встроенная поддержка базовой платформы.

С помощью этой информации оценщик должен оценить и подтвердить:

- что спецификация виртуализированного объекта не определяет небезопасного поведения. Если это так, правильная виртуализация отразит это небезопасное поведение и приведет к ненадежной системе;
- что спецификация поддержки, обеспечиваемой ИТ-средой, допускает удобную поддержку эмуляции;
- что поведение виртуализированного объекта правильно отражает заданное поведение «реального» объекта;
- что реализация функций эмуляции безопасна;
- что реализация использует поддержку, обеспечиваемую ИТ-средой, как задумано, и без побочных эффектов, критичных для безопасности.

Когда это будет выполнено, анализ уязвимостей может быть сосредоточен на аспектах, не затронутых в поведенческой спецификации эмулируемого элемента, и на том, могут ли эти аспекты привнести уязвимости, которые могут не существовать при использовании «реального» элемента. Одним из примеров такого поведения, не рассматриваемого в поведенческой спецификации, является время. Например, когда несколько виртуализированных дисков сопоставлены с одним физическим диском, время, используемое для работы диска, может предоставить информацию о том, используются ли и как используются другие виртуализированные диски. Если подразумевается использование разных разделов, которые должны быть отделены друг от друга, виртуализированные диски могут создать скрытый канал с потенциально высокой пропускной способностью.

Приложение ДА
(справочное)

Сведения о соответствии ссылочных международных стандартов межгосударственным стандартам

Таблица ДА.1

Обозначение ссылочного международного стандарта	Степень соответствия	Обозначение и наименование ссылочного межгосударственного стандарта
ISO/IEC 15408-1	—	*, [1)]
ISO/IEC 15408-2	—	*, [2)]
ISO/IEC 15408-3	—	*, [3)]
ISO/IEC 18045	—	*, [4)]
* Соответствующий межгосударственный стандарт отсутствует. До его принятия рекомендуется использовать перевод на русский язык данного международного стандарта.		

[1)] В Российской Федерации действует ГОСТ Р ИСО/МЭК 15408-1—2012 «Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 1. Введение и общая модель».

[2)] В Российской Федерации действует ГОСТ Р ИСО/МЭК 15408-2—2013 «Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 2. Функциональные компоненты безопасности».

[3)] В Российской Федерации действует ГОСТ Р ИСО/МЭК 15408-3—2013 «Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 3. Компоненты доверия к безопасности».

[4)] В Российской Федерации действует ГОСТ Р ИСО/МЭК 18045—2013 «Информационная технология. Методы и средства обеспечения безопасности. Методология оценки безопасности информационных технологий».

Библиография

[1] Anderson James P., Computer Security Technology Planning Study, Deputy for Command and Management Systems, HQ Electronic Systems Division (AFSC), ESD-TR-73-51, Volume I and II, October 1972

[2] Nibaldi Grace H., Specification of a Trusted Computing Base (TCB), The Mitre Corporation, M79-228, 30 November 1979

[3] Saltzer J.H., Schroeder M.D., The Protection of Information in Computer Systems, Communications of the ACM, 17, 1974

[4] Schroeder C., Saltzer W., Final Report of the Multics Kernel Design Project, Massachusetts Institute of Technology, Laboratory for Computer Science, MIT/LCS/TR-196, 1977

[5] Anderson R., Security Engineering — A Guide to Building Dependable Distributed Systems, 2nd Edition, Wiley Publishing 2008, ISBN: 978-0-470-06852-6

[6] Gasser M., Building a Secure Computer System, Van Nostrand Reinhold, 1988

[7] Dougherty C., Secure Design Patterns, Technical Report CMU/SEI-2009-TR-010, 2009

[8] Levin Timothy E., Design Principles and Guidelines for Security, SecureCore Technical Report, NPS-CS-08-001

[9] Pfleeger C.P., Pfleeger S.L., Margulies J., Security in Computing, Prentice Hall, Inc, Fifth edition, 2015

[10] Neumann P.G., Principled assuredly trustworthy composable architectures, Technical Report CDRL A001 Final Report December 28, 2004, SRI, Menlo Park, CA, 2004

УДК 006.34:004.056:004.056.5:004.056.53:006.354 МКС 35.030

Ключевые слова: информационная безопасность, принцип построения архитектуры, оценка безопасности информационных технологий

Технический редактор *И.Е. Черепкова*
Корректор *Л.С. Лысенко*
Компьютерная верстка *Е.А. Кондрашовой*

Сдано в набор 05.07.2021. Подписано в печать 19.07.2021. Формат 60×84⅛. Гарнитура Ариал.
Усл. печ. л. 3,72. Уч.-изд. л. 3,34.

Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

Создано в единичном исполнении во ФГУП «СТАНДАРТИНФОРМ»
для комплектования Федерального информационного фонда стандартов,
117418 Москва, Нахимовский пр-т, д. 31, к. 2.
www.gostinfo.ru info@gostinfo.ru