МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И ПРОГРАММИРОВАНИЯ

Г.М. ЧЕРНОКНИЖНЫЙ
И.Н. ВАСИЛЬЕВА

БЕЗОПАСНОСТЬ ОПЕРАЦИОННЫХ СИСТЕМ

Учебное пособие

ИЗДАТЕЛЬСТВО
САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО
ЭКОНОМИЧЕСКОГО УНИВЕРСИТЕТА
2019

ББК 32.972.11
Ч49

Чернокнижный Г.М.

Ч49 Безопасность операционных систем : учебное пособие / Г.М. Чернокнижный, И.Н. Васильева. – СПб. : Изд-во СПбГЭУ, 2019. – 105 с.

ISBN 978-5-7310-4781-4

В учебном пособии рассмотрены требования к безопасности операционных систем и способы ее достижения: организационные меры, методы аутентификации и управления доступом, обеспечение целостности, аудит, криптографическая защита, гарантированное уничтожение информации. Рассмотрены механизмы безопасности операционных систем Windows и UNIX.

Предназначено для бакалавров направления подготовки 10.03.01 «Информационная безопасность», профиль «Безопасность компьютерных систем (в экономике и управлении)», а также может представлять интерес для всех изучающих данную и смежные дисциплины.

Chernoknijny G.M.

Operating System Security : a tutorial / G.M. Chernoknijny, I.N. Vasileva. – Saint-Petersburg : Publishing house of Saint Petersburg State University of Economics, 2019. – 105 p.

The tutorial discusses the requirements for security of operating systems and ways to achieve it: organizational measures, methods of authentication and access control, ensuring integrity, audit, cryptographic protection, guaranteed destruction of information. Safety mechanisms of Windows and UNIX operating systems are considered.

The tutorial is intended for bachelors of the direction of training 10.03.01 «Information security», profile «Security of computer systems (in economics and management)», and can also be of interest to all students of this and related disciplines.

LBC 32.972.11

Рецензенты: канд. экон. наук, доц. **И.Г. Гниденко**
канд. экон. наук, доц. **О.Д. Мердина**

ISBN 978-5-7310-4781-4

© СПбГЭУ, 2019

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 4

ГЛАВА 1. ТРЕБОВАНИЯ БЕЗОПАСНОСТИ ИНФОРМАЦИИ, ПРЕДЪЯВЛЯЕМЫЕ К ОС 5
1.1. Необходимые организационные меры 5
1.2. Нормативные документы регуляторов 7
1.3. Сертифицированные ОС 9

ГЛАВА 2. СПОСОБЫ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ ОС 16
2.1. Аутентификация и управление доступом 16
2.2. Диагностика и восстановление 34
2.3. Журналирование файловых систем 39
2.4. Протоколирование и аудит 41
2.5. Криптографическая защита 42
2.6. Гарантированное уничтожение информации 49

ГЛАВА 3. ОСОБЕННОСТИ ЗАЩИТЫ В РАЗЛИЧНЫХ ОС 55
3.1. Система безопасности ОС Windows 55
3.2. Защита в ОС UNUX 75
3.3. Безопасность виртуальных ОС 95
3.4. Безопасность ОС мобильных устройств 98

БИБЛИОГРАФИЧЕСКИЙ СПИСОК 105

ВВЕДЕНИЕ

Дисциплина «Безопасность операционных систем» является важнейшей в подготовке бакалавров направления «Информационная безопасность», профиль «Безопасность компьютерных систем (в экономике и управлении)». Подавляющее большинство угроз конфиденциальности, целостности и доступности информации, хранимой на компьютерах, происходит, в конечном итоге, при атаках на- или через операционные системы (ОС). Изучение данной дисциплины основывается на материале смежных дисциплин: программно-аппаратная защита информации; операционные системы; контроль безопасности в компьютерных сетях.

Цель и задачи дисциплины «Безопасность операционных систем» – предоставить студентам материал, содержащий знания по мерам противодействия злоумышленникам в различной операционной среде в том числе:

- необходимые организационные меры защиты;
- нормативные документы регуляторов;
- диагностика и восстановление ОС;
- криптографическая защита;
- механизмы защиты в различных ОС и др.

Владение современными способами и средствами защиты системного программного обеспечения (ПО) в локальной, сетевой, виртуальной и облачной средах позволит студентам стать специалистами, подготовленными к самостоятельной работе в качестве системных администраторов и администраторов безопасности на предприятиях различного масштаба в разных предметных областях.

ГЛАВА 1. ТРЕБОВАНИЯ БЕЗОПАСНОСТИ ИНФОРМАЦИИ, ПРЕДЪЯВЛЯЕМЫЕ К ОС

Защищенная ОС создает механизмы безопасности, которые обеспечивают соблюдение целей безопасности информационной системы (ИС), несмотря на угрозы, с которыми ИС сталкивается. Эти механизмы безопасности предназначены для обеспечения такой гарантии в контекст ресурсов и механизмов планирования. Цели безопасности определяют требования безопасной работы системы для любых процессов, которые она может выполнять. Механизмы безопасности должны обеспечивать эти цели независимо от возможных способов, которыми система может быть неправильно использована злоумышленниками.

Справедливо также утверждение, что ни одна ОС современной сложности не является в полной мере безопасной. Трудность предотвращения ошибок в процессе программирования и проблемы, связанные с попыткой удалить такие ошибки, означает, что ни одна достаточно сложная система, такая, как ОС, не может быть полностью защищена.

В пособии не рассматриваются сбои и неисправности технического обеспечения и предполагается, что таких ошибок нет.

1.1. Необходимые организационные меры

Сущность информационной безопасности (ИБ) – защита объектов информации и бизнес-процессов организации, в соответствии с требованиями конфиденциальности (информация доступна только тем, кто авторизован для доступа), целостности (обеспечивается точность и полнота информации и методов обработки) и доступности (информация доступна по требованию авторизованных пользователей).

Операционные системы являются базовым ПО, составляющим основу любой современной ИС. Поэтому обеспечение безопасности компьютерных систем невозможно без обеспечения надлежащего уровня защищенности ОС, которая должна иметь возможность защитить файлы, память и настройки конфигурации от нежелательного просмотра и изменения.

Несмотря на многообразие современных средств защиты информации в различных операционных средах, первоочередными являются организационные меры, на основе которых строятся все защитные механизмы и от которых зависит эффективность их использования.

Первоочередной организационной мерой, обеспечивающей, в том числе, безопасность ОС, является разработка и исполнение политики информационной безопасности (далее – политики безопасности, ПБ) ИС.

Политика безопасности (информации в организации) – совокупность документированных правил, процедур, практических приемов или руководящих принципов в области безопасности информации, которыми руководствуется организация в своей деятельности [5]. Политика ИБ, являясь документом организационного уровня, описывает безопасность в обобщенных терминах без привязки к конкретным техническим решениям и поставщикам. Разработка политики безопасности предполагает проведение оценки рисков информационной безопасности ИС и базируется на модели угроз и модели нарушителя.

Модель угроз – документ, содержащий в себе список угроз, которым может быть подвержена та или иная система, входящая в состав ИС. В качестве основы для ее разработки может быть рекомендована модель угроз, предложенная ФСТЭК РФ (fstec.ru/component/attachments/download/ 289). Оценка рисков предполагает определение вероятности реализации угроз с учетом существующих средств защиты информации. Далее нужно выделить те объекты инфраструктуры ИС, на которые могут быть направлены угрозы, и определить возможные негативные последствия (например, рассмотреть совокупность всех компонентов ОС на предмет возможности осуществления взлома, получения доступа к данным и нарушения работоспособности). Риск представляет собой сочетание вероятности реализации угрозы безопасности и тяжести возможных негативных последствий [3].

В составе модели угроз следует отдельно выделить *модель действий нарушителя*. Разработчик подсистем безопасности ставит себя на место потенциального нарушителя и представляет, какими способами он мог бы попытаться получить доступ к ресурсам ОС.

Документы, составляющие политику ИБ организации, образуют иерархическую структуру, уровни которой, начиная с верхнего, определяют, соответственно стратегию, тактику и методику обеспечения ИБ организации [2]. Нижние уровни политики затрагивают отдельные аспекты обеспечения ИБ, описываемые частными политиками, связанными с ними документированными процедурами и техническими документами (регламентами, инструкциями, руководствами, технологическими профилями). Частные политики определяют вопросы, затрагивающие, в том числе, и безопасность работы с ОС. Среди прочих, могут быть рассмотрены:

– авторизация пользователей;
– управление доступом пользователей;
– управление паролями и другой аутентификационной информацией;
– антивирусная защита;
– контроль обновлений ОС;
– установка и контроль сетевых соединений;
– работа с внешними носителями;

– резервное копирование и восстановление;
– информирование об инцидентах и др.

Внедрение политики безопасности сопровождается принятием других традиционных мер организационного уровня, таких как распределение полномочий и обязанностей, проведение обучения сотрудников, информирование и повышение осведомленности по вопросам ИБ.

Требования политик могут изменяться и дополняться, во-первых, в зависимости от конкретной ситуации в ИС, во-вторых, по мере развития самой организации, появления новых технологий, систем и проектов. Считается, что период пересмотра частных политик безопасности составляет около одного года, но может проводиться и чаще, в зависимости от оценки их эффективности.

Выбор тех или иных защитных мер определяется как действующей в организации политикой безопасности, модели угроз и действий нарушителя, так и детальным анализом рисков ИБ с оценкой экономической эффективности различных вариантов построения системы ЗИ. Такой подход должен обеспечить, с одной стороны, необходимую защиту ОС в конкретных условиях, с другой стороны, сохранить пользователям комфортные условия работы, не усложняя систему излишними ограничениями.

1.2. Нормативные документы регуляторов

В Российской Федерации регуляторами в области информационной безопасности выступают Федеральная служба по техническому и экспортному контролю России (ФСТЭК РФ), ФСБ России, Роскомнадзор (в составе Министерства цифрового развития, связи и массовых коммуникаций РФ) и, в особых случаях, Министерство Обороны России. При этом вопросы сертификации средств технической защиты информации, как и аттестации объектов информатизации по требованиям безопасности информации (исключая ИС, обрабатывающие сведения, составляющие государственную тайну), находятся в ведении ФСТЭК России. Базовыми документами для использования в работе с ПО являются Приказ ФСТЭК России от 31 августа 2010 г. № 489 (https://fstec.ru/ tekhnicheskaya-zashchita-informatsii/ dokumenty/110-prikazy/370) и информационные сообщения ФСТЭК России с требованиями безопасности ОС.

Приказ № 489 «Требования о защите информации, содержащейся в информационных системах общего пользования» регламентирует общие требования к мероприятиям по обеспечению защиты информации в ИС общего пользования.

Информационное сообщение ФСТЭК России от 18 октября 2016 г. № 240/24/4893 «Об утверждении требований безопасности информации к операционным системам» (https://fstec.ru/normotvorcheskaya/informatsion-

nye-i-analiticheskie-materialy/1206-informatsionnoe-soobshchenie-fstek-rossii-ot-18-oktyabrya-2016-g-n-240-24-4893) определяет сферу применения «Требований» как «ОС, используемые в целях обеспечения защиты (не криптографическими методами) информации, содержащей сведения, составляющие государственную тайну, и иной информации ограниченного доступа при ее обработке в ИС (автоматизированных системах управления)».

В соответствии с Требованиями выделяются следующие типы ОС:

– ОС общего назначения (тип «А») – ОС, предназначенная для функционирования на средствах вычислительной техники общего назначения (автоматизированные рабочие места, серверы, смартфоны, планшеты, телефоны и иные);

– встраиваемая ОС (тип «Б») – ОС, встроенная (прошитая) в специализированные технические устройства, предназначенные для решения заранее определенного набора задач;

– ОС реального времени (тип «В») – ОС, предназначенная для обеспечения реагирования на события в рамках заданных временных ограничений при заданном уровне функциональности.

Для дифференциации требований к функциям безопасности ОС выделяются шесть классов защиты: от самого низкого – шестого к самому высокому классу – первому:

– ОС, соответствующие 6 классу защиты, применяются в государственных информационных системах (ГИС) 3 и 4 классов защищенности, в автоматизированных системах управления производственными и технологическими процессами 3 класса защищенности, в информационных системах персональных данных (ИСПДн) при необходимости обеспечения 3 и 4 уровней защищенности персональных данных;

– ОС, соответствующие 5 классу защиты, применяются в ГИС 2 класса защищенности, в автоматизированных системах управления производственными и технологическими процессами 2 класса защищенности, в ИСПДн при необходимости обеспечения 2 уровня защищенности персональных данных;

– ОС, соответствующие 4 классу защиты, применяются в ГИС 1 класса защищенности, в автоматизированных системах управления производственными и технологическими процессами 1 класса защищенности, в ИСПДн при необходимости обеспечения 1 уровня защищенности персональных данных, в информационных системах общего пользования II класса;

– ОС, соответствующие 1, 2 и 3 классам защиты, применяются в информационных (автоматизированных) системах, в которых обрабатывается информация, содержащая сведения, составляющие государственную тайну.

Таким образом, всего может быть создано 18 профилей защиты ОС. Методические документы «Профили защиты операционных систем» для ОС 4, 5 и 6 классов опубликованы на сайте ФСТЭК России в разделе Специальные нормативные документы (https://fstec.ru/tekhnicheskaya-zashchita-informatsii/dokumenty-po-sertifikatsii/120-normativnye-dokumenty). Профили защиты формулируют функциональные возможности ОС, проблемы безопасности (угрозы, политику и предположения безопасности), цели безопасности, функциональные требования безопасности и требования доверия к безопасности в соответствии с ГОСТ Р ИСО/МЭК 15408 «Информационная технология (ИТ). Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий».

Например, в ОС общего назначения (типа «А») 4 и 5 классов должны быть реализованы следующие функции безопасности:

– идентификация и аутентификация;
– управление доступом;
– регистрация событий безопасности;
– ограничение программной среды;
– изоляция процессов;
– защита памяти;
– контроль целостности;
– обеспечение надежного функционирования;
– фильтрация сетевого потока.

В среде, в которой функционирует ОС, должны быть реализованы следующие функции безопасности:

– физическая защита;
– доверенная загрузка ОС;
– обеспечение условий безопасного функционирования ОС;
– обеспечение доверенного маршрута;
– обеспечение доверенного канала.

1.3. Сертифицированные ОС

Регуляторами сертифицирован ряд ОС, которые отвечают определенным классам защиты.

ОС общего назначения

ОС Windows 7 (SP1) сертифицирована в редакциях «Профессиональная», «Корпоративная» и «Максимальная» и имеет сертификат № 2180/1, дата внесения в реестр: 04.10.2011, срок действия сертификата: 04.10.2020 (https://www.microsoft.com/ru-ru/securitycertification/products.aspx).

Краткая характеристика ОС: программное средство общего назначения со встроенными средствами защиты от несанкционированного доступа к информации, не содержащей сведения, составляющие государственную тайну. Соответствуют требованиям руководящего документа (РД) «Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации» (Гостехкомиссия России, 1992)[1] – по 5 классу защищенности и могут использоваться при создании автоматизированных систем до класса защищенности 1Г включительно и при создании информационных систем персональных данных до 2 класса включительно.

Ограничения по применению:

– настройки и контроль механизмов защиты безопасности должны производиться в соответствии с «Руководством по настройке системы»;

– программный комплекс должен пройти процедуру контроля соответствия (верификации) сертифицированному эталону;

– на программный комплекс должны быть установлены все актуальные обязательные сертифицированные обновления безопасности.

ОС Microsoft Windows 8.1 сертифицирована в редакциях: «Профессиональная», «Корпоративная» и имеет сертификат № 3263, дата внесения в реестр: 07.11.2014, срок действия сертификата: 07.11.2020 (https://www.microsoft.com/ru-ru/securitycertification/products.aspx).

Соответствуют требованиям РД «СВТ. Защита от НСД к информации. Показатели защищенности от НСД к информации» – по 5 классу защищенности при условии установки всех актуальных обязательных сертифицированных обновлений безопасности и выполнения указаний по эксплуатации, приведенных в формулярах 501110-001-82487552-2014 03 ФО и 501110-001-82487552-2014 02 ФО. Имеет ограничения по применению, аналогичные Windows 7.

ОС Microsoft Windows 10 Pro 32/64-bit со встроенными и дополнительно интегрируемыми механизмами обеспечения безопасности, реализуемыми средством защиты информации «Secure Pack Rus версия 3.0 (исполнения 5, 6). Соответствует требованиям ФСБ России по защите информации от несанкционированного доступа в автоматизированных информационных системах, расположенных на территории Российской Федерации, класса АК3 (для исполнения 5), класса АК2 (для исполнения 6) и может использоваться для защиты информации, не содержащей сведений, составляющих государственную тайну.

---

[1] Далее – РД «СВТ. Защита от НСД к информации. Показатели защищенности от НСД к информации»

Альт Линукс СПТ 7.0 (https://www.basealt.ru/products/alt-spt/spt-workstation/) – унифицированный дистрибутив для серверов, рабочих станций и тонких клиентов со встроенными программными средствами защиты информации, сертифицированный ФСТЭК России, который может быть использован для построения автоматизированных систем по класс 1В включительно и ИСПДн по класс 1К включительно. ОС позволяет одновременно хранить и обрабатывать на одном персональном компьютере или сервере конфиденциальные данные, обеспечивать многопользовательскую работу с разграничением доступа к информации, работать с виртуальными машинами, а также использовать средства централизованной авторизации. Выданный ФСТЭК России сертификат подтверждает соответствие продукта требованиям следующих РД: РД «СВТ. Защита от НСД к информации. Показатели защищенности от НСД к информации» – по 4 классу защищенности; «Защита от несанкционированного доступа к информации. Часть 1. ПО средств защиты информации. Классификация по уровню отсутствия недекларированных возможностей» – по 3-му уровню контроля и технических условий. Техническая поддержка пользователей «Альт Линукс СПТ» осуществляется компанией «Свободные программы и технологии» через партнера-разработчика «Базальт СПО».

Red Hat Enterprise Linux Server 6.x (https://www.redhat.com/en/ technologies/linux-platforms/enterprise-linux) соответствует РД «СВТ. Защита от НСД к информации. Показатели защищенности от НСД к информации» – по 5 классу защищенности. Основными отличительными возможностями RHEL 6 названы всесторонняя поддержка виртуализации, переход на использование современного Linux-ядра, улучшенная поддержка технологий энергосбережения и доступ к самым свежим версиям ПО.

ОС специального назначения

ОС Astra Linux (https://astralinux.ru/) сертифицирована в редакции ОС специального назначения Astra Linux Special Edition (релиз «Смоленск», версия 1.6) по требованиям безопасности информации к ОС типа «А» 1-го класса защиты в системе сертификации СЗИ ФСТЭК России. Astra Linux Special Edition, единственная ОС в России, может применяться в ИС тех организаций, которые работают с государственной тайной самой высокой степени секретности – «особой важности» включительно.

Помимо сертификата ФСТЭК России по 1-ому классу защиты релиз «Смоленск» получил сертификат соответствия требованиям ФСБ России к средствам защиты информации, содержащей государственную тайну. Данные сертификаты подтверждают, что разработанная доверенная аппаратно-программная платформа на основе ОС специального назначения Astra Linux Special Edition может использоваться в ИС органов государственного и военного управления, включая высшие органы государственной власти, для обработки любой информации ограниченного доступа.

ОС «СинтезМ» (https://fintech.ru/index.php?id=2). Защищенная ОС «СинтезМ» (ЗОС «СинтезМ») является собственной разработкой компании АО «ФИНТЕХ» на основе продуктов с открытым кодом. ЗОС «СинтезМ» представляет собой комплекс программ, предназначенных для создания автоматизированных систем в защищенном исполнении (АСЗИ) в части обеспечения функционирования серверных группировок и автоматизированных рабочих мест пользователей, а также обеспечения выполнения требований по защите информации, обрабатываемой в АСЗИ, от НСД и реализации защищенного вычислительного процесса. Соответствует требованиям ФСБ России по защите информации от несанкционированного доступа с использованием средств криптографической защиты информации в автоматизированных информационных системах, расположенных на территории Российской Федерации, 1 класса, и может использоваться для обработки информации, содержащей сведения, составляющие государственную тайну.

В состав ЗОС «СинтезМ» входят следующие компоненты:

– ПК «СинтезМ-Сервер» ТАСП.62.01.11.000.002;
– ПК «СинтезМ-Клиент» ТАСП.62.01.11.000.003;
– КП «ПСЗИ «СинтезМ» ТАСП.62.01.11.000.004 в составе:

- ПК «СинтезМ-СБ» ТАСП.62.01.11.000.005;
- ПС «СинтезМ-АБ» ТАСП.62.01.11.000.006;
- ПК «СинтезМ-ИПА» ТАСП.62.01.11.000.007;
- ПС «СинтезМ-УК» ТАСП.62.01.11.000.008;
- ПС «СинтезМ-Репо» ТАСП.62.01.11.000.009.

«**Защищенная операционная система для суперЭВМ**» (https://www.rosatom.ru/production/supercomputer-and-software/), разработка «Росатом» на базе открытой программной платформы Linux. Эта ЗОС прошла сертификацию ФСТЭК РФ по требованиям к обработке информации, составляющей государственную тайну, и может применяться на высокопроизводительных комплексах для выполнения полного цикла решения параллельных задач имитационного моделирования. Данная ОС содержит программные компоненты, необходимые для организации распределенных высокопроизводительных вычислений, причем накладные расходы на общесистемные нужды при ее работе не превышают 1%. ЗОС позволяет строить конфигурации среды выполнения параллельных приложений для бездисковых вычислительных узлов.

Входящие в эту отечественную ОС программные компоненты обеспечивают полнофункциональную работу суперкомпьютера любой производительности, что позволяет использовать ее для создания автоматизированных высокопроизводительных систем управления в защищенном испол-

нении; сбора и обработки больших объемов структурированных и неструктурированных данных; создания систем поддержки принятия решений, требующих интенсивных вычислений, и др. Средства защиты информации в ЗОС отвечают требованиям технических условий, что позволяет применять системное ПО в автоматизированных системах класса защищенности 1Б-С. В 2014 году были проведены сертификационные испытания ЗОС и получен сертификат соответствия. Встроенные средства защиты информации ОС отвечают требованиям РД ФСТЭК России по второму уровню контроля на отсутствие недекларируемых возможностей, а также техническим условиям.

Средства защиты ЗОС обеспечивают выполнение следующих функций: идентификация и аутентификация; разграничение доступа; регистрация событий безопасности и сигнализация о них; организация и контроль доменной структуры разграничения потоков информации; контроль целостности средств защиты информации. В основе средств защиты информации ЗОС лежат программные компоненты Scientific Linux 6.4 для платформы x86, а недавно были завершены работы по переходу на версию Scientific Linux 7x.

ОС «Заря» и «Рассвет» (http://cniieisu.ru/projects/basic-technologies) – семейство ОС специального назначения, созданное для использования в Вооруженных Силах РФ. Разработаны в Объединенной приборостроительной корпорации по заказу Министерства обороны РФ. ОС «Заря» основана на дистрибутиве Linux корпоративного уровня Red Hat Enterprise Linux. Она соответствует 3 классу защиты от несанкционированного доступа и 2 уровню контроля отсутствия не декларированных возможностей. ОС «Рассвет» предназначена для реализации среды работы программ на серверах и рабочих станциях, построенных на платформе x86_64 и работы со сведениями «для служебного пользования».

ОС «Циркон 10С» (http://www.zirkon-platform.ru/zirkon) создавалась с использованием исходного кода ОС Solaris 10 и технологии Sun Ray. Актуальная версия – ОС «Циркон 36С» использует ядро CentOS GNU/Linux 6.5. ОС «Циркон 36С» и ПО АСТД 36С (в составе ОС «Циркон 36С» и ПО терминального доступа «Циркон 36Т») соответствуют:

– «Временным требованиям к программному обеспечению, используемому в автоматизированных системах ИТКС специального назначения по уровню, определенному для АИС, предназначенных для обработки информации, содержащей сведения, составляющие государственную тайну с грифом секретности не выше «совершенно секретно» и могут быть использованы для встраивания средств криптографической защиты информации;

– «Требованиям по защите информации, содержащей сведения, составляющие государственную тайну, от несанкционированного доступа с

использованием криптографических средств в автоматизированных информационных системах, расположенных на территории Российской Федерации» для АИС 1 класса.

«Нейтрино» КПДА.10964-01. Защищенная операционная система реального времени (ЗОСРВ) (http://www.kpda.ru/products/kpda10964/) предназначена для отказоустойчивого и предсказуемого управления ресурсами многопроцессорных/ многоядерных и многомашинных вычислительных комплексов реального времени.

ЗОСРВ «Нейтрино» удовлетворяет требованиям к СВТ по 3 классу защиты информации от НСД, 2 уровню контроля отсутствия НДВ, 2 классу защищенности для межсетевых экранов, что позволяет использовать изделие КПДА.10964-01 в АС класса защищенности до 1Б включительно. Сертификаты соответствия: МО РФ № 3385 от 27.10.2016 (действителен до 27.10.2021); ФСТЭК России № 3795 от 29.08.2017 (действителен до 29.08.2020).

Особенности области применения ЗОСРВ «Нейтрино»:

– высокие требования к надежности, продиктованные потенциально катастрофическими результатами в случае возникновения сбоя САУ или АСУ;

– жесткие требования функционирования в режиме реального времени, обусловленные тесной взаимосвязью вычислительного процесса с процессами, протекающими в электрическом и механическом оборудовании, химическими реакциями, распространением излучений и сигналов различной физической природы;

– жесткие требования технологической независимости, исключающие возможность негативного влияния зарубежных организаций или частных лиц на проектирование, разработку, производство, поставку и сопровождение изделия.

Mobile OS RUS (http://omprussia.ru/security.html). Защищенная мобильная ОС общего назначения на базе Sailfish является единственной мобильной ОС, включенной в реестр Отечественного ПО и прошедшей сертификацию ФСБ по классу АК1/КС1. Соответствует «Требованиям по защите конфиденциальной информации от несанкционированного доступа в автоматизированных ИС, расположенных на территории Российской Федерации», класса АК1, и может использоваться для защиты информации, не содержащей сведений, составляющих государственную тайну.

ОС Sailfish Mobile OS RUS со встроенным программным средством криптографической защиты информации PathfinderSSL реализует отечественные криптографические алгоритмы и соответствует «Требованиям по защите конфиденциальной информации от несанкционированного доступа в автоматизированных ИС, расположенных на территории Российской Федерации» – класса АК1, «Требованиям к средствам криптографической за-

щиты информации, предназначенным для защиты информации, не содержащей сведений, составляющих государственную тайну» – класса КС1, и может использоваться для защиты информации, не содержащей сведений, составляющих государственную тайну.

Sailfish Mobile OS RUS является передовой ЗОС, механизмы безопасности которой встроены не только в ядро ОС, но и во все основные компоненты:

– контроль целостности файловой системы (ФС);

– контроль целостности загрузчика;

– полная блокировка устройства, без возможности разблокирования, при нарушении целостности ФС или загрузчика;

– ограничения прав пользователя в соответствии с корпоративными политиками безопасности как на уровне ОС, так и с помощью систем управления MDM;

– запрет на установку ПО пользователем из недоверенных источников.

ГЛАВА 2. СПОСОБЫ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ ОС

2.1. Аутентификация и управление доступом

Аутентификация пользователей

Аутентификация пользователей является важной частью безопасности ИС, наряду с конфиденциальностью и целостностью. Особенно это важно для систем удаленного доступа, которые устанавливают соединения в общедоступной инфраструктуре недоверенных сетей, таких, как Интернет.

Идентификация – предоставление пользователем своих логина и пароля (так называемых идентификационных данных) при входе в ИС.

Аутентификация – процедура проверки идентификационных данных, предоставленных пользователем при входе в ИС, то есть, фактически, проверка принадлежности пользователю предъявленного им идентификатора или проверка (подтверждение) подлинности пользователя.

Авторизация – проверка наличия у пользователя прав на выполнение некоторых действий, например, на доступ к ресурсам ИС, и предоставление соответствующих полномочий для выполнения этих действий. Авторизация происходит после идентификации и аутентификации.

В качестве фактора аутентификации может рассматриваться:

– нечто, что знает пользователь (пароль, PIN);

– нечто, чем обладает пользователь (USB-ключ, OTP-токен, смарт-карта);

– нечто, что есть часть самого пользователя – биометрические характеристики пользователя: отпечатки пальцев, голос, сетчатка глаза и др.

Рассмотрим каждый из перечисленных факторов [10].

Парольный способ аутентификации в настоящее время является самым простым, дешевым и популярным несмотря на то, что недостатки этого способа неоднократно упоминались в разных источниках. Механизмы противодействия угрозам компрометации паролей пользователей также хорошо известны как системным администраторам, так и специалистам в области защиты информации. Однако, халатное отношение к вопросам безопасности со стороны администраторов, как привилегированных пользователей, позволяет злоумышленнику получить фактически неограниченный доступ ко всем типам данных. Опыт многих лет показывает, что слабая парольная политика или вообще ее несоблюдение приводит к возможности подбора или перебора паролей, что позволяет осуществить несанкционированный доступ к информации различного уровня конфиденциальности.

В большинстве случаев пользователи используют простые и легко подбираемые пароли для входа в систему. Если же пароль отвечает пароль-

ной политике, то он зачастую записывается на бумагу и хранится на мониторе или под клавиатурой. Поэтому возникает трудность в сохранении баланса между удобством пароля для пользователя и его надежностью.

В результате многие аналитики рекомендуют увеличивать количество методов проверки подлинности за счет использования нескольких методов аутентификации, чтобы обеспечить лучшую проверку подлинности. На рынке ИБ существует довольно много универсальных продуктов, которые способны решить проблему слабых паролей или вовсе заменить парольную аутентификацию.

Сейчас существуют *аппаратные решения* в виде USB-ключей, USB-ключей с генератором одноразовых паролей, OTP-токенов и смарт-карт. В общем случае пароли хранит или генерирует устройство, что значительно облегчает жизнь пользователю, а также исключает риск подсматривания их злоумышленником. Однако возникает проблема потери или кражи устройства, кроме того, стоимость аппаратных решений существенно выше обычных парольных систем, а их применение в ряде случаев требует дополнительной технической модификации системы (например, установка считывателей смарт-карт).

Биометрия развивается и становится все более востребованной на рынке. Обусловлено это тем, что биометрические устройства запоминают определенные уникальные особенности человека, свойственные только для конкретной личности, например, сетчатка глаза. Следовательно, аутентификация на основе биометрии – это надежный способ доступа в систему. Так же к плюсам можно отнести удобство для пользователя: не нужно запоминать длинные пароли или носить с собой ключи или смарт-карты.

Однако биометрия имеет ряд недостатков, помимо того, что отличительные особенности человека могут быть скомпрометированы. Возникает вопрос вмешательства в частную жизнь, личные, культурные и религиозные аспекты, так же вопросы гигиены и травмоопасности.

Действительно, пользователям не безразличен факт хранения и распространения их биометрических данных. Они не имеют возможности контролировать распространение таких данных, поэтому опасаются злоупотреблений. Еще один немаловажный факт: будет ли прибор для считывания, например, геометрии руки, обрабатываться раствором после каждого использования. Прибор может нанести пользователю травму, например, система сканирования сетчатки, в которой свет направляется в глаза. Так же пользователи подвергаются риску причинения вреда со стороны преступников – потеря части тела. Непригодность для всех пользователей – еще один минус биометрической аутентификации. Существует вероятность, что пользователь может не иметь необходимых частей тела для внесения в систему биометрического параметра.

Несмотря на указанные проблемы и недостатки, аутентификация на основе биометрии является перспективным направлением, так как значительно упрощает процесс и дает возможность выполнять сложные задачи.

Решением рассматриваемой проблемы является построение *многофакторной аутентификации*. В большинстве случаев для обеспечения безопасного доступа в систему одного проверочного фактора мало, поскольку любой способ аутентификации имеет свои недостатки. Однако, используя преимущества нескольких разных факторов аутентификации в комплексе, можно существенно снизить риски.

Классическими вариантами двухфакторной аутентификации является сочетание аппаратной или биометрической аутентификации с введением пароля или PIN-кода для разблокировки устройства аутентификации (например, смарт карты). Вместе с тем, считывающее устройство может быть не всегда доступно, эта ситуация характерна скорее не для входа в ОС, а при аутентификации к удаленным приложениям или сервисам. Для решения подобных проблем были разработаны новые подходы к аутентификации, позволяющие, наряду с вводом пароля, подтвердить владение доверенным устройством и получение информации по дополнительному каналу, зарегистрированным ранее надежным способом.

К таким методам аутентификации относятся:

1. Сочетание пароля и одноразового кода подтверждения, получаемого по SMS на зарегистрированный ранее (доверенный) номер[2]. При этом предполагается, что канал передачи данных по SMS можно считать доверенным. Несмотря на то, что в настоящее время существует техническая возможность перехвата SMS сообщений, а также может быть сделан дубликат SIM-карты, данная атака сложна для реализации, и нивелируется путем существенного ограничения срока действия кода подтверждения.

2. OTP-токен + PIN-код. OTP (One Time Password)-токен – устройство для генерации одноразовых паролей для аутентификации, принадлежащее определенному пользователю. OTP-устройство как правило предварительно регистрируется и синхронизируется с выполняющим аутентификации сервером. PIN-код может использоваться в качестве дополнительной информации, используемой при генерации OTP, а так же для активации OTP-токена.

3. QR-код + пароль или PIN-код. В последнее время все популярнее становится аутентификация, в основе которой лежит сканирование QR (quick response)-кода. QR-код является индивидуальным графическим идентификатором пользователя в мобильном приложении, привязанном к

---

[2] Следует отметить, что согласно действующему российскому законодательству, номер абонента мобильной связи может быть зарегистрирован только при предъявлении документа, удостоверяющего личность (паспорта).

телефонному номеру или доверенному мобильному устройству абонента. Считается, что QR-код может быть отображен только на доверенном устройстве и, таким образом, подтверждает владение им. При прохождении процедуры аутентификации QR-код считывается штатным оборудованием компьютера, например, web-камерой и отсылается проверяющей стороне. Таким образом, в процедуре аутентификации задействовано дополнительное устройство – смартфон пользователя, отображающий QR-код.

Учитывая, что смартфоны являются широко распространенными личными устройствами, подобные методы могут быть применены для аутентификации пользователя при входе в ОС.

Так, современные версии ОС Windows поддерживают, наряду с парольной, аппаратную аутентификацию с использованием смарт карт и USB-токенов, двухфакторную аутентификацию с дополнительным вводом PIN-кода, биометрическую аутентификацию с помощью сканера отпечатка пальцев. Платформа Windows Hello (https://www.microsoft.com/ru-ru/windows/windows-hello) ОС Windows 10 позволяет проходить аутентификацию по отпечатку пальца, лицу или радужной оболочке глаза, либо использовать для аутентификации личное устройство-компаньон, такое как цифровой браслет, смарт-часы, смартфон и др. (при этом может потребоваться связать компьютер и устройства-компаньоны в Azure Active Directory или Active Directory, а также посредством Bluetooth). В Windows 10 может быть настроена двухшаговая аутентификация с использованием смартфона. При настройке этого типа аутентификации пользователю будет предложено скачать мобильное приложение, которое должно быть синхронизировано с компьютером посредством QR-кода. Затем при каждом входе в ОС приложение на смартфоне будет генерировать одноразовый проверочный код, который подтверждает участие личного устройства в процедуре аутентификации.

Подобные технологии реализованы и в других развитых ОС, например, при доступе к учетной записи Apple ID на компьютерах Mac с участием доверенных устройств iPhone, iPad или iPod touch. В ОС Linux может быть настроена аутентификация с помощью стороннего мобильного приложения (например, Google Authenticator или Authy) с использованием QR-кода.

Протоколы аутентификации (ОС Windows)

NTLM (NT LAN Manager) является набором протоколов аутентификации, используемым в различных реализациях сетевых протоколов Microsoft и поддерживается NTLM Security Support Provider (NTLMSSP). Проверка подлинности NTLM поддерживается для компьютеров, настроенных как члены рабочих групп, а также используется для проверки подлинности входа в систему на автономных системах (для проверки подлинности и создания безопасного канала для DCE/ RPC, NTLM также используется во

всех ОС Microsoft в качестве интегрированной механизма единого входа SSO (Single Sign-On). Он также используется как «Встроенная проверка подлинности Windows» для HTTP, SMTP, POP3, IMAP, CIFS / SMB, Telnet, SIP и др. [12].

Single Sign-On (SSO, технология единого входа) широко используется в Windows-сетях, особенно в сетях с использованием домена, и позволяет пользователю получать доступ к различным сервисам и приложениям без повторной аутентификации, используя единый идентификатор. Например, если в сети существует множество различных сервисов то, пройдя процедуру аутентификации в одном из них, пользователь автоматически получает доступ ко всем остальным, что избавляет его от многократного ввода данных своей учетной записи. Начиная с Windows 2000, механизм единого входа базируется на протоколе Kerberos, который является предпочтительным методом проверки подлинности входа для сред Active Directory. Существуют два стандарта, в которых описывается информация о протоколах NTLM: RFC 2433 – для версии 1 и RFC 2759 – для версии 2.

NTLM является протоколом аутентификации типа запрос-ответ, который использует три сообщения для проверки подлинности клиента при соединении и четвертое дополнительное сообщение для обеспечения целостности:

1. клиент устанавливает сетевой путь к серверу и посылает NEGOTIATE_MESSAGE, уведомляя сервер о своих возможностях;
2. сервер отвечает CHALLENGE_MESSAGE, который используется для установления личности клиента;
3. клиент отвечает на вызов с AUTHENTICATE_MESSAGE.

Протокол NTLM использует один или оба из двух значений хэшированного пароля (LM и NT хэши), которые хранятся на сервере (или контроллере домена), и которые являются эквивалентом пароля. Это означает, что сервер может проверить пароль пользователя без фактической передачи его значения. При процессе аутентификации предоставляется имя пользователя, домен/ рабочая группа и хэш пароля [12].

В настройках Windows можно выставить различные уровни безопасности, который зависит от типа используемого протокола.

Уровень 0 – Отправление LM и NTLM ответа; запрет на использование протокола NTLMv2 (session security). Клиенты используют LM и NTLM аутентификацию, и никогда не используют протокол NTLM v2 (session security); контроллеры домена допускают LM, NTLM и NTLMv2 аутентификацию.

Уровень 1 – Использовать протокол NTLMv2 (session security) при согласовании. Клиенты используют LM и NTLM аутентификацию и протокол NTLMv2 (session security), если сервер поддерживает его; контроллеры домена допускают LM, NTLM и NTLMv2.

Уровень 2 – Отправка только NTLM ответа. Клиенты используют только проверку подлинности NTLM и NTLMv2 (session security), если сервер поддерживает ее; контроллеры домена допускают LM, NTLM и NTLMv2 аутентификацию.

Уровень 3 – Отправка только ответа протокола NTLMv2. Клиенты используют аутентификацию NTLMv2 и NTLMv2 (session security), если сервер поддерживает ее; контроллеры домена допускают LM, NTLM и NTLMv2 аутентификацию.

Уровень 4 – Контроллеры домена отказываются от ответов типа LM. Клиенты используют аутентификацию NTLMv2 и NTLMv2 (session security), если сервер поддерживает ее; контроллеры домена принимают аутентификацию только NTLM и NTLMv2.

Уровень 5 – Контроллеры домена не принимают аутентификацию LM и NTLM (только NTLMv2). Клиенты используют аутентификацию NTLMv2, а также используют аутентификацию NTLMv2 (session security), если сервер поддерживает ее; контроллеры домена не принимают аутентификацию LM и NTLM (они принимают только NTLMv2). В современных версиях ОС Windows этот уровень установлен по умолчанию.

Компьютеры на базе Windows используют два метода для хэширования паролей пользователей, имеющие совершенно разные последствия для безопасности, – это LAN Manager (LM) и NT LAN Manager (NTLM).

LM хэш был одним из первых алгоритмов хэширования пароля, примененным в ОС Windows. Современные системы используют NTLM хэш, но поддерживают и LM-хэши для целей обратной совместимости. LM-хэш основан на устаревшем алгоритме симметричного шифрования DES. Вычисление LM хэша пароля включает в 6 шагов:

1. Символы введенного пользователем пароля преобразуется в верхний регистр и кодируются System OEM Code.
2. К строке пароля добавляются нулевые символы, пока его длина не достигнет 14 символов (14 байт).
3. Полученная таким образом строка пароля делится на две половинки по 7 символов (7 байт).
4. Эти значения используются для создания двух ключей шифрования DES, по одному от каждого половины. Байты преобразуются в поток битов, к которому добавляются биты четности (вставляются после каждых 7-ми значащих битов). Так создаются 64-битные ключи DES.
5. Каждый ключ DES используется для шифрования предустановленной строки ASCII (KGS! @ # $%), что дает два 8-байтовых шифрованных значения.
6. Два 8-байтовых значения зашифрованного текста объединяются, чтобы сформировать 16 байтовое значение, и являющееся LM-хэшем.

Алгоритм хэширования LM имеет следующие недостатки.

– Пароли ограничены 14-ю символами, максимальное теоретическое пространство ключей равно $95^{14} \approx 2^{92}$ с использованием 95 клавиатурных ASCII символов (при условии, что сами пользовательские пароли являются случайным набором символов, однако на практике, как правило, это не так).

– Пароли длиннее 7 символов делятся на две части, и каждая часть хэшируется отдельно; эта слабость позволяет реализовать атаку по подбору каждой половины пароля по отдельности. При этом достаточно проверить только $95^7 \approx 2^{46}$ комбинаций. В настоящее время перебор такого количества комбинаций практически осуществим.

– Преобразование букв к верхнему регистру уменьшает пространство ключей для каждой половины до $69^7 \approx 2^{43}$.

– Любой пароль короче 7 символов приведет к добавлению 7-ми нулевых байтов, и вторая часть хэша в этом случае будет известной постоянной величиной (0xAAD3B435B51404EE). Этот факт позволяет легко определять короткие пароли.

– Алгоритм хэширования не использует добавление случайных последовательностей (соли), что делает его восприимчивым к атакам человек по середине (MitM) и Pass-The-Hash. Добавление пусть и известных случайных значений к исходной информации до начала хэширующего преобразования является стандартной техникой, применяемой для предотвращения атак подбора по словарю, таких как атака с использованием радужных таблиц.

– Окончательный слабость LM-хэшей заключается в их реализации – так как они изменяются только при изменении пользователем своего пароля, они могут быть использованы для проведения атаки Pass-the-Hash.

Указанные недостатки делают LM-хэши весьма уязвимыми. На практике подбор неслучайного пароля с использованием радужных таблиц происходит за считанные секунды, а при использовании метода грубой силы (в случае если пароль был задан случайным образом) потребуется до нескольких часов. Радужные таблицы – таблицы, содержащие вычисленные хэш-значения для каждого возможного пароля с учетом количества символов. Атака с использованием радужных таблиц выглядит как поиск хэш-значения и его обратное сопоставление паролю.

Поэтому LM-хэши, начиная с Windows Server 2008 и Windows Vista, отключены по умолчанию, вместо них используются NTLM-хэши, однако LM-хэши могут создаваться и поддерживаться для обратной совместимости со старыми системами. Так же для некоторых систем (старше, чем Windows Server 2008) вышли обновления, которые отключают LM-хэш как основной для взаимодействия в сети.

Алгоритм вычисление NTLM-хэша (далее хэш NT) гораздо проще, чем создание LM-хэша, потому что использует алгоритм бесключевой хэш-функции MD4 с выходным значением длиной 16 байт:

1. Сначала символы пароля преобразуются в Unicode кодировку.
2. Далее алгоритм MD4 используется для получения хэша NT.

Например, пароль «password123», после преобразования, будет представлен как «67A54E1C9058FCA16498061B96863248».

Недостатки NT-хэшей:

– Хэш-функция MD4 в настоящее время не считается криптографически сильной, так как допускает коллизии и не рекомендована к применению (может быть найден другой пароль с тем же самым хэшем).

– Как и для LM-хэшей, процедура генерации хэшей NT не использует добавление случайных значений (соли). Поэтому NT-хеши подвержены атакам с использованием радужных таблиц (для осмысленных паролей).

NTLMv2-хэш также имеет длину 16 байт, но при его вычислении дополнительно используется подмешивание случайной информации (соли) и использование ключевой конструкции HMAC-MD5, что обеспечивает приемлемую по современным меркам стойкость результирующего хэш-значения. Использование NTLMv2 позволяет подобрать пароль только в случае простых неслучайных значений и перехвата SMB пакетов аутентификации.

Kerberos – современный протокол удаленной аутентификации. Он используется по умолчанию в доменах на базе Windows Active Directory и принудительно используется в Server 2016. Сетевой протокол аутентификации, позволяющий передавать данные через незащищенные сети. Он ориентирован на клиент-серверную модель и обеспечивает взаимную аутентификацию сторон с использованием третьей доверенной стороны. Данная модель является одним из вариантов протокола Нидхема-Шредера с использованием симметричной криптографии [1]. В настоящее время используется 5 версия протокола. Требования реализации протокола изложены в документе RFC 1510, кроме того, в спецификации RFC 1964 описывается механизм и формат передачи жетонов безопасности в сообщениях Kerberos.

В протоколе Kerberos участвуют две взаимодействующие стороны и доверенный сервер KS (не обязательно выделенный), выполняющий роль Центра распределения ключей (KDC, Key Distribution Center). Протокол успешно работает при условии, что часы каждого участника синхронизированы с часами сервера. Следует отметить, что в этом протоколе необходим обмен с сервером для получения сеансового ключа каждый раз. Протокол обеспечивает надежное соединение объектов при условии, что ни один из ключей не скомпрометирован и сервер защищен.

Сервер KS (Kerberos Server) выполняет две основные роли: сервер аутентификации AS (Authentication Server) и сервер службы выдачи манда-

тов TGS (Ticket Granting Service). Физически эти серверы могут быть совмещены. Информационными ресурсами, необходимыми клиентам, управляет сервер информационных ресурсов RS. Предполагается, что серверы службы Kerberos надежно защищены от физического доступа нарушителей.

Сетевые службы, требующие проверки подлинности, и клиенты, которые хотят использовать эти службы, регистрируют в Kerberos свои секретные ключи. Kerberos хранит БД о клиентах и их секретных ключах. «Знание» сервером Kerberos секретных ключей каждого пользователя и ресурсов сети, поддерживающих этот протокол, позволяет создавать зашифрованные сообщения, направляемые клиенту или серверу; успешное же расшифрование этих сообщений является гарантией прохождения аутентификации всеми участниками протокола. Kerberos также создает сеансовые ключи (session key), которые выдаются клиенту и серверу (или двум клиентам) и никому больше. Сеансовый ключ используется для шифрования сообщений, которыми обмениваются две стороны, и уничтожается после окончания сеанса.

Область действия системы Kerberos распространяется на тот участок сети, все пользователи которого зарегистрированы под своими именами и паролями в БД сервера Kerberos.

Укрупненно процесс идентификации и аутентификации пользователя в системе Kerberos версии 5 можно описать следующим образом (отображен на рис. 2.1).

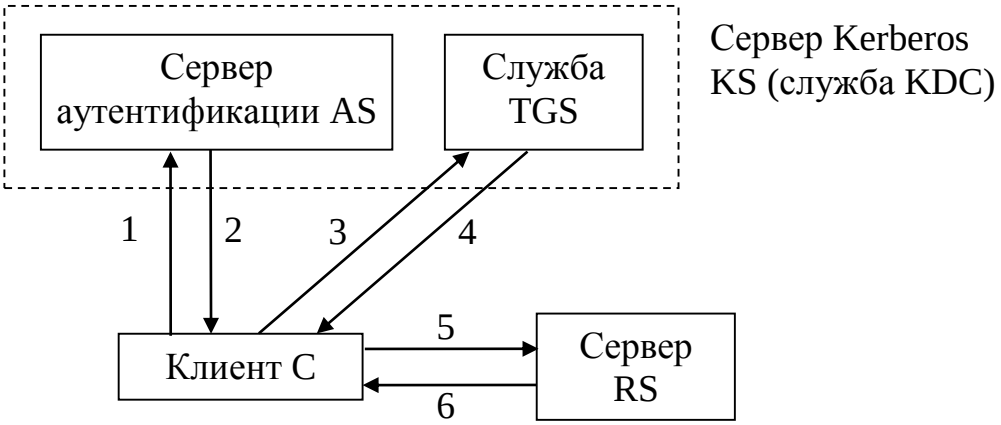

Рис. 2.1. Схема работы протокола Kerberos

Клиент C, желая получить доступ к ресурсу сети, направляет запрос серверу аутентификации AS. Сервер AS идентифицирует пользователя с помощью его имени и пароля и высылает клиенту мандат (ticket) на доступ к серверу службы выделения мандатов TGS.

Для использования конкретного целевого сервера информационных ресурсов RS клиент C запрашивает у TGS мандат на обращение к целевому

серверу RS. Если все в порядке, TGS разрешает использование необходимых ресурсов сети и посылает соответствующий мандат клиенту C.

Основные шаги работы системы Kerberos:

1. C → AS – запрос клиента C к серверу *AS* за разрешением обратиться к службе TGS.

2. AS → C – разрешение (мандат) от сервера AS клиенту *C* для обращения к службе TGS.

3. C → TGS – запрос клиента C к службе TGS на получение допуска (мандата) к серверу ресурсов RS.

4. TGS → C – разрешение (мандат) от службы TGS клиенту C для обращения к серверу ресурсов RS.

5. C → RS – запрос информационного ресурса (услуги) у сервера RS.

6. RS → C – подтверждение подлинности сервера RS и предоставление информационного ресурса (услуги) клиенту C.

Данная модель взаимодействия клиента с серверами может функционировать только при условии обеспечения конфиденциальности и целостности передаваемой управляющей информации. Без строгого обеспечения информационной безопасности клиент C не сможет отправлять серверам AS, TGS и RS свои запросы и получать разрешения на доступ к обслуживанию в сети.

Чтобы избежать возможности перехвата и несанкционированного использования информации, Kerberos применяет при передаче любой управляющей информации в сети систему многократного шифрования с использованием комплекса секретных ключей (секретный ключ клиента, секретный ключ сервера, секретные сеансовые ключи пары клиент-сервер). Kerberos может использовать различные алгоритмы симметричного шифрования и хэш-функции.

Расширение PKINIT протокола Kerberos предоставляет возможность использовать цифровые сертификаты (то есть криптографию с открытым ключом) на этапе предварительной аутентификации клиента при входе в систему с помощью смарт-карты или USB-токена. Процедура аутентификации в этом случае протекает следующим образом:

1. Пользователь идентифицируется в системе и предъявляет свой закрытый ключ.

2. Клиентская машина формирует запрос к AS (AS_REQ), в котором указывает, что будет использоваться асимметричное шифрование. Отличие данного запроса заключается в том, что он подписывается (с помощью закрытого ключа клиента) и, кроме стандартной информации, содержит также сертификат открытого ключа пользователя.

3. Получив запрос, TGS сначала проверяет достоверность сертификата пользователя, а затем электронную подпись (используя полученный

открытый ключ пользователя). После этого TGS проверяет локальное время, присланное в запросе (для защиты от атаки повторного воспроизведения).

4. Удостоверившись в подлинности клиента, TGS формирует ответ (AS_REP), в котором, в отличие от стандартной версии, сеансовый ключ зашифровывается открытым ключом пользователя. Кроме того, ответ содержит сертификат TGS и подписывается его закрытым ключом (аналогично запросу клиента).

5. Получив ответ, пользователь проверяет подпись TGS и расшифровывает сеансовый ключ (используя свой закрытый ключ).

В настоящее время протокол Kerberos поддерживают все ОС Windows и множество Unix- и Linux-подобных ОС.

RADIUS (Remote Authentication in Dial-In User Service) – служба удаленной аутентификации дозванивающихся пользователей**)** – сетевой протокол, предназначенный для обеспечения централизованной аутентификации, авторизации и учета (Authentication, Authorization, and Accounting, AAA) пользователей, подключающихся к различным сетевым службам. RADIUS используется, например, при аутентификации пользователей WiFi, VPN, dial-up подключений, и в других подобных случаях. Описан в стандартах RFC 2865 и RFC 2866.

Сервер RADIUS выполняет централизованную аутентификацию с помощью одного из следующих протоколов [4]:

– PAP (Password Authentication Protocol) – простой протокол аутентификации, в котором имя пользователя и пароль отправляются на сервер удаленного доступа в незашифрованном виде. Использование PAP не рекомендуется;

– CHAP (Challenge Handshake Authentication Protocol) и MS-CHAP – широко поддерживаемый метод аутентификации, в котором используется передача не самого пароля, а косвенных сведений о нем;

– EAP (Extensible Authentication Protocol) – протокол аутентификации, используемый в беспроводных сетях и соединениях точка-точка;

– DNIS (Dialed Number Identification Service) – служба идентификации набранного номера);

– ANI (Automatic Number Identification) – служба автоматической идентификации номера.

Работа DNIS основана на номере, набранном пользователем, а ANI – на номере, с которого звонит пользователь.

Пользователь посылает свои данные для аутентификации и авторизации серверу (FTP, POP, WWW, PROXY и т.д.), такие серверы называются Network Access Server (NAS). Сервер NAS и сервер RADIUS используют общий секретный ключ (для аутентификации друг друга, и хэширование открытых паролей пользователей).

NAS (клиент) формирует запрос серверу RADIUS: *Access Request*. Ниже приведены возможные ответы сервера RADIUS (рис. 2.2).

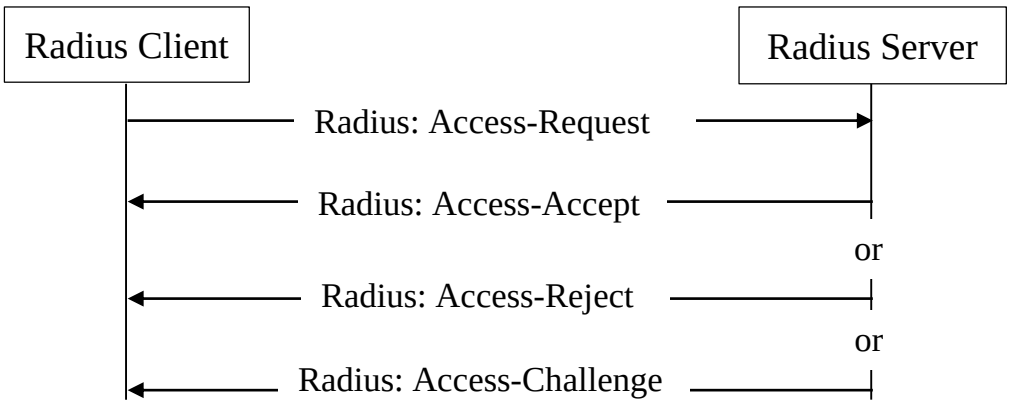

Рис. 2.2. Логическая схема взаимодействия клиента и сервера RADIUS

– *Access-Reject* – доступ запрещен;
– *Access-Challenge* – запрос дополнительной информации от пользователя, например, второй пароль, PIN-код, номер карты и т.п.;
– *Access Accept* – доступ разрешен.

Ответ *Access-Challenge* может использоваться для посылки случайного числа пользователю, для дальнейшего хэширования его с паролем.

При выполнении всех условий в отклик *Access-Accept* включается список всех конфигурационных параметров для данного пользователя.

Всего насчитывается свыше трех десятков реализаций RADIUS-серверов, для всего многообразия ОС и платформ. Наиболее популярными современными серверами RADIUS являются FreeRADIUS, OpenRADIUS, GNU RADIUS, ClearBox Enterprise RADIUS Server.

В развитие протокола Radius разработан протокол DIAMETER (текущие версии RFC 3588 и RFC 3589), который призван заменить RADIUS, оставшись обратно совместимым с ним. В основе протокола DIAMETER лежит концепция в создании базового протокола с возможностью его расширения для предоставления сервисов AAA при появлении новых технологий доступа. Основные отличия протокола DIAMETER:

– поддержка TCP, кроме UDP;

– защита данных на сетевом и транспортном уровнях (IPsec или TLS) (эти функции для RADIUS еще в процессе стандартизации IETF);

– поддержка механизмов надежной доставки на уровне приложений;

– сообщения об ошибках;

– увеличенное адресное пространство для пар атрибут-значение (AVP) и идентификаторов (32 bit вместо 8 bit);

– улучшенная поддержка мобильности;

– динамическое обнаружение узлов;

– улучшенная расширяемость: возможность использования пользовательских атрибутов и команд.

Модели управления доступом

Средства управления доступом – система разграничения доступа – позволяют определить и контролировать действия, которые субъекты информационной системы могут выполнять над ее объектами [11].

Речь идет о логическом управлении доступом, которое, в отличие от физического, реализуется программными средствами. Логическое управление доступом – основной механизм безопасности многопользовательских систем, призванный обеспечить конфиденциальность и целостность объектов.

Субъект – активный компонент системы, который может стать причиной потока информации от объекта к субъекту или изменения состояния системы (например, пользователи, программы или процессы).

Объект – пассивный компонент системы, хранящий, принимающий или передающий информацию (например, папки и файлы, объекты базы данных). Доступ к объекту означает доступ к содержащейся в нем информации.

Предполагается, что существует безошибочный способ различения объектов и субъектов.

С теоретической точки зрения, *доступ* – взаимодействие между субъектом и объектом, в результате которого производится перенос информации между ними. Два фундаментальных типа доступа: *чтение* – операция, результатом которой является перенос информации от объекта к субъекту; *запись* – операция, результатом которой является перенос информации от субъекта к объекту.

Санкционированный доступ к информации – доступ, не нарушающий установленные правила разграничения доступа.

Правила разграничения доступа – правила, определяющие права (полномочия) субъектов на доступ к объектам информационной системы. Правила разграничения доступа задаются установкой разрешений и запретов на выполнение тех или иных действий с объектами информационной системы.

Понятия объекта и видов доступа меняется от сервиса к сервису. С точки зрения пользователя доступ означает ознакомление (чтение) с информацией, ее обработку (запись, модификация, уничтожение). Отдельным правом может быть возможность передачи полномочий доступа другим субъектам (так называемое право владения, полный доступ). Например, для ОС к объектам относятся файлы и папки, устройства и процессы. Применительно к файлам, папкам и устройствам обычно рассматриваются права:

чтение (просмотр) и выполнение (для программных файлов), запись (в том числе и создание нового объекта), модификация, удаление. Процессы можно создавать и уничтожать. Современные ОС могут поддерживать и другие объекты.

Уровни доступа обычно образуют иерархическую систему: это значит, что разрешение более высокого уровня требует наличия разрешений более низких уровней. Например, невозможно изменять содержимое документа, не имея права его просмотра и записи изменений. Полный доступ дает разрешения на все остальные уровни доступа.

При принятии решения о предоставлении доступа обычно анализируется следующая информация:

– идентификатор субъекта (идентификатор пользователя, сетевой адрес компьютера и т.п.). Подобные идентификаторы являются основой произвольного (или дискреционного) управления доступом;

– атрибуты субъекта (группа пользователя, метка безопасности и т.п.). Метки безопасности – основа принудительного (мандатного) управления доступом.

Дискреционное управление доступом (Discretionary Access Control) характеризуется следующими свойствами:

– все субъекты и объекты должны быть идентифицированы;

– права доступа субъекта к объекту системы определяются на основании некоторого внешнего по отношению к системе правила.

Правила дискреционного управления доступом часто задаются матрицей доступа Лэмпсона. В подобной матрице строки соответствуют субъектам системы, столбцы – объектам, элементы матрицы описывают права доступа для соответствующей пары «субъект – объект».

Матрица доступа, обладающая большой разреженностью (большинство клеток – пустые), обычно хранится не в виде двухмерного массива, а по столбцам – в виде *списков управления доступом* (ACL – Access Control List). Список связан с защищаемым объектом и хранит перечень «допущенных» субъектов и их прав на данный объект. Посредством списков управления доступом легко добавить права или явным образом запретить доступ.

В качестве примера можно привести использование списков управления доступом для описания прав пользователей и групп к файлам и папкам в файловой системе NTFS в ОС семейства Windows.

В дискреционных системах управления доступом субъект с определенными правами доступа (администратор, создатель-владелец объекта) может передать это право любому другому субъекту.

Подавляющее большинство операционных систем и систем управления базами данных (СУБД) реализуют именно дискреционное управление доступом, что вызвано, прежде всего, простотой реализации и гибкостью этой модели доступа.

Удобной надстройкой над средствами логического управления доступом является *ограничивающий интерфейс*, когда пользователя лишают самой возможности попытаться совершить несанкционированные действия, включив в число видимых ему объектов только те, к которым он имеет доступ. Подобный подход обычно реализуют в рамках системы меню (пользователю показывают лишь допустимые варианты выбора) или посредством ограничивающих оболочек, таких как restricted shell в ОС Unix.

Разнообразие объектов и применимых к ним операций приводит к принципиальной *децентрализации* логического управления доступом (каждый сервис сам решает, позволить ли конкретному субъекту ту или иную операцию). На практике это приводит к значительным трудностям. Главная проблема заключается в том, что ко многим объектам можно получить доступ с помощью разных сервисов. Права доступа, определенные для одного сервиса, могут не иметь смысла для другого. Трудность возникает и при организации обмена данными между различными сервисами (экспорте/импорте данных), когда информация о правах доступа, как правило, теряется.

Рассредоточенность управления доступом ведет к тому, что доверенными должны быть многие пользователи, а не только системные администраторы, что повышает вероятность утечек информации из-за некомпетентности. Средства дискреционной защиты не решают задачу контроля передачи информации. Права доступа существуют отдельно от данных и проверяются только при первом обращении к объекту. При этом возникает опасность переноса информации из защищенного объекта в общедоступный. Наличие дискреционной защиты не может помешать легальному пользователю законным образом получить секретную информацию и затем сделать ее доступной для других. Например, можно создать новый объект, в который будет перенесена информация, создатель-владелец имеет полномочия для передачи прав доступа к этому объекту. Доступ можно ограничить только к именованным объектам (объектам ФС, объектам СУБД), а не к самим данным. Нельзя защитить часть данных, хранящихся в таком объекте.

Для решения этих проблем была предложена модель *мандатного* управления доступом, которая явилась развитием дискреционной модели.

Мандатное управление доступом (Mandatory Access Control) – это разграничение доступа субъектов к объектам данных, основанное на характеризуемом меткой уровне секретности информации (метка содержится в самом объекте) и на официальном разрешении (допуске) субъектов.

Мандатное управление доступом подразумевает, что:

– все субъекты и объекты должны быть идентифицированы;

– задан линейно упорядоченный набор меток секретности;

– каждому объекту системы присвоена метка секретности, определяющая ценность содержащейся в нем информации – его уровень секретности;

– каждому субъекту системы присвоена метка секретности, определяющая уровень доверия к нему – его уровень доступа;

– решение о разрешении доступа субъекта к объекту принимается исходя из типа доступа и сравнения метки субъекта и объекта.

Мандатное управление доступом называют также физической защитой, поскольку метки секретности хранятся вместе с данными, и их невозможно проигнорировать при получении доступа к информации. Метки секретности неизменны на всем протяжении существования данных, неотделимы от информации и переносятся (копируются) вместе с ней.

Использование мандатной защиты позволяет ограничить доступ собственно к данным, а не к именованным объектам (в СУБД – вплоть до отдельных атрибутов записи).

Два интуитивно понятных правила безопасности, реализованных в мандатной модели управления доступом, позволяют контролировать утечки конфиденциальной информации:

– «нет чтения вверх» – субъект не может прочитать информацию из объекта, метка секретности которого выше, чем его уровень допуска;

– «нет записи вниз» – субъект не может перенести (скопировать) информацию из объекта с большим грифом секретности в объект с меньшим грифом секретности.

Мандатная модель значительно повышает уровень защищенности информационных систем, однако у данного подхода есть один очень серьезный недостаток – сложность реализации. В мандатной модели необходимо классифицировать не только пользователей, но и все файлы, программы и носители информации. Матрица доступа в такой модели была бы очень громоздкой, поэтому для ее представления используется функциональный способ, когда матрицу не хранят в явном виде, а каждый раз вычисляют содержимое соответствующих клеток (производя сравнение меток безопасности субъекта и объекта).

Мандатная модель управления доступом ориентирована на предотвращение одной угрозы – нарушения конфиденциальности. Но информация и информационная система обладают и другими важными качествами безопасности – целостностью и доступностью. Существует модель управления доступом, аналогичная мандатной и ориентированная на обеспечение целостности, однако попытка скомбинировать их приводит к противоречиям и делает их совместную работу невозможной.

При большом количестве пользователей и объектов традиционные дискреционные системы управления доступом слишком сложны для адми-

нистрирования. Для упрощения администрирования крупных информационных систем, например, компьютерных сетей, используется *ролевое управление доступом* (Role-Based Access Control).

Суть *ролевого управления доступом* заключается в том, что между пользователями и назначаемыми им правами появляются промежуточные слои – роли. В ролевой модели понятие «субъект» замещается понятиями «пользователь» и «роль». *Пользователь* – человек, работающий с информационной системой и выполняющий определенные служебные обязанности. *Роль* – это активно действующая в системе абстрактная сущность, с которой связан набор полномочий, необходимых для выполнения определенной деятельности.

Подобное разделение хорошо отражает особенности деятельности различных организаций, что привело к распространению ролевых политик безопасности. При этом, как один пользователь может быть авторизован администратором на выполнение одной или нескольких ролей, так и одна роль может быть сопоставлена одному или нескольким пользователям.

При использовании ролевой политики, управление доступом осуществляется в две стадии:

– для каждой роли указывается набор полномочий (разрешений на доступ к различным объектам системы);

– каждому пользователю сопоставляется список доступных ему ролей.

В самом простом варианте права доступа назначаются не лично пользователям, а ролевым группам; для получения необходимых прав пользователей включают в соответствующие группы (рис. 2.3). Для любого пользователя может быть одновременно активно несколько ролей, каждая из которых дает определенные права. Работа пользователя разбивается на сеансы, в каждом из которых он авторизуется с одной или несколькими разрешенными для него ролями.

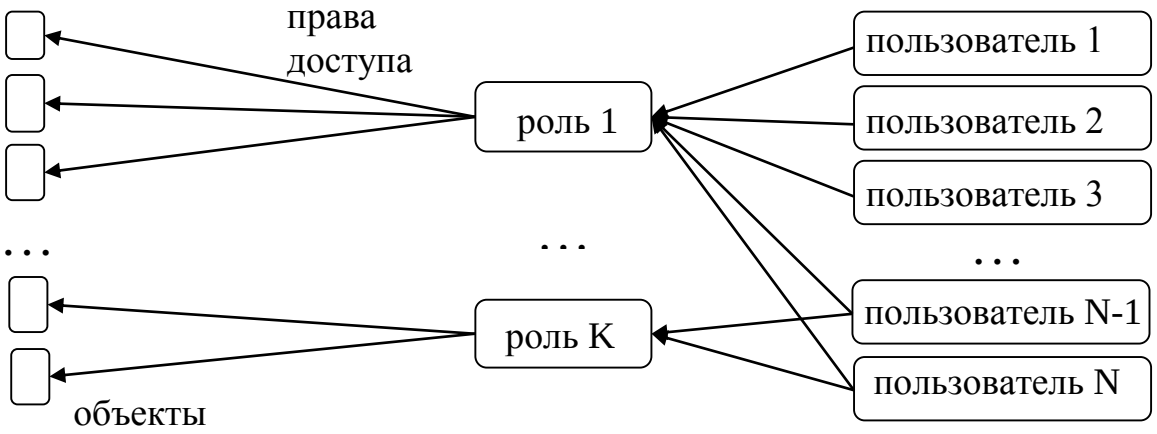

Рис. 2.3. Ролевая модель доступа

При ролевом управлении доступом пользовательские группы не реализуются в виде отдельных и самостоятельно действующих субъектов доступа – действуют только субъекты пользователей.

Для ролей может быть определено отношения наследования, когда роль наследует все права от своей предшественницы. Таким образом, роли могут составлять иерархию, начиная с минимума прав, с постепенным добавлением новых прав. Технологически последовательное расширение прав достигается членством пользователя в нескольких ролевых группах.

Ролевая модель управления доступом обеспечивает хорошую управляемость в сложных системах с большим количеством пользователей и объектов. В настоящее время широко распространена *двухуровневая* схема назначения прав доступа (в сетевых ОС, СУБД), являющаяся развитием идеи ролевого управления (рис. 2.4).

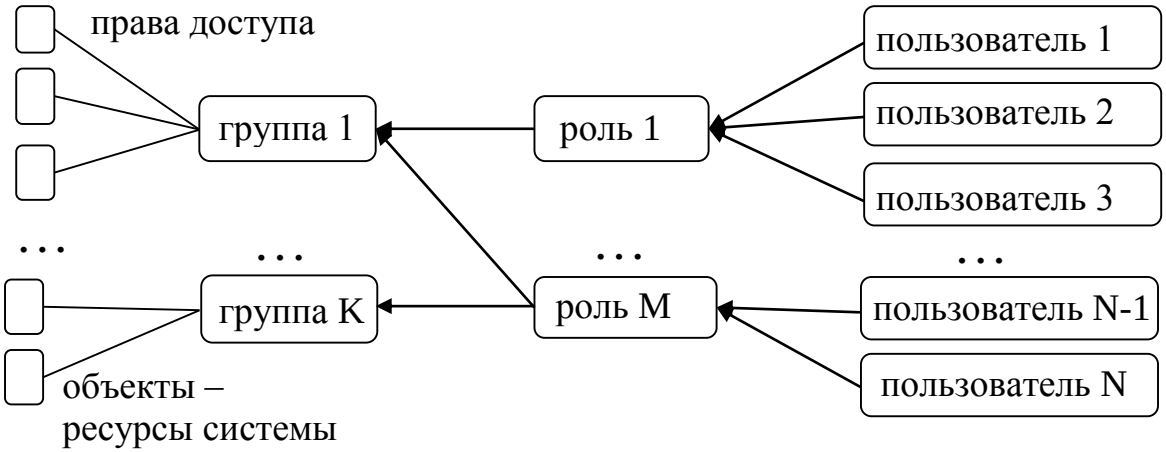

Рис. 2.4. Двухуровневое назначение прав доступа

На втором уровне пользователей объединяют в функциональные группы (например, сотрудники могут группироваться по занимаемой должности или по подразделению в зависимости от выполняемых функций). Назначение прав функциональным группам производится включением их в ролевые группы.

Например, в сетях под управлением Windows Server косвенная группировка ресурсов достигается назначением *локальным группам* прав одного уровня на доступ к однотипным ресурсам (внутри доменов). Пользователей объединяют в *глобальные группы*, которые затем могут быть включены в соответствующие локальные группы, что обеспечивает применение разрешений на доступ.

Если пользователи и ресурсы расположены в нескольких разных доменах, может быть использован промежуточный уровень группировки с помощью *универсальных групп*.

Вложенная структура групп (рис. 2.5) используется, если многим пользователям требуется доступ к ресурсам, владельцы которых не могут эффективно контролировать членство пользователей в группах.

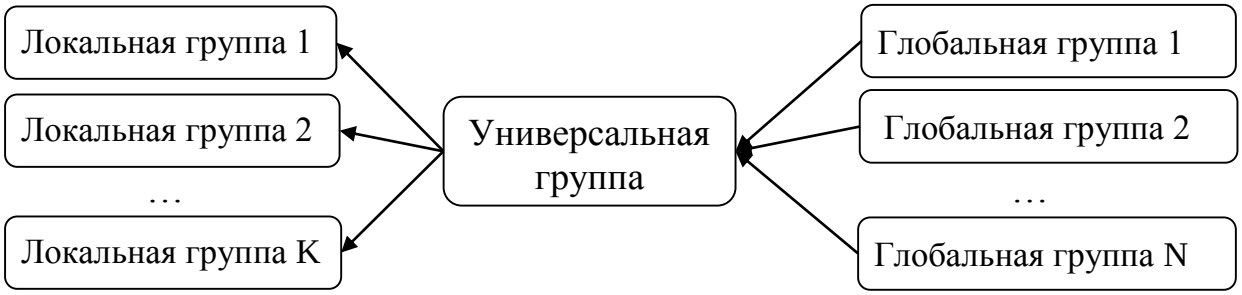

Рис. 2.5. Вложенность групп в Microsoft Active Directory

Вместе с тем, трехуровневое назначение прав доступа может существенно затруднить понимание механизма получения доступа, что снижает эффективность администрирования.

Безопасность модели ролевого управления доступом не может быть формально доказана, поэтому системы ролевого управления доступом реализуются средствами мандатных и дискреционных моделей. Модель ролевого управления доступом сочетает преимущества обеих этих моделей, обеспечивая, с одной стороны, невозможность передачи прав доступа и достаточную гибкость, с другой.

2.2. Диагностика и восстановление

Обеспечение целостности системных ресурсов

Поддержание целостности системных ресурсов крайне важно, как для устойчивой и надежной работы самой системы, так и для обеспечения доверия к ней. Подмена исполняемых системных файлов или файлов конфигураций позволяет нарушителям выполнять произвольные несанкционированные действия, а также скрывать следы своего присутствия в системе и активно противодействовать средствам защиты информации. Например, руткиты, поставляют ложную информацию – как правило, они скрывают некоторые файлы на диске или работающие процессы. Чаще всего речь идет о файлах, установленных нарушителем после взлома системы. Также существует возможность модификации ОС таким образом, чтобы она автоматически подменяла данные в некоторых секторах диска (например, при попытках антивирусного сканирования).

Поэтому с точки зрения безопасности ОС крайне важно обеспечить возможность проверки целостности системных компонент и, в случае необходимости, их восстановления. Методические документы ФСТЭК России

«Профили защиты операционных систем» (https://fstec.ru/ tekhnicheskaya-zashchita-informatsii/dokumenty-po-sertifikatsii/120-normativnye-dokumenty) включают (в зависимости от типа и класса ОС) следующие функциональные требования безопасности:

– контроль целостности компонентов ПО, разрешенного для запуска, и реагирование на попытки запуска компонентов ПО, целостность которых была нарушена;

– возможность тестирования (самотестирования) функций безопасности ОС, проверки целостности ПО и данных (параметров) ОС;

– возможность восстановления штатного режима функционирования ОС;

– возможность возврата ОС при сбоях и отказах к безопасному состоянию в автоматизированном режиме;

– возможность восстановления объектов ОС из резервных копий, созданных с использованием ОС, и использования ассоциированных с ними атрибутов безопасности.

Подсистема целостности ОС может использовать разные концепции. Один из подходов рассматривает целостность как принципиальную неизменяемость файлов, так что в файл не могут быть внесены какие-либо изменения. Концепция неизменяемых файлов реализована, например, в ОС BSD, такие файлы могут быть изменены только в административном однопользовательском режиме. ОС Android хранит все свои системные файлы в разделе, который монтируется только для чтения во время нормальной работы, изменения же могут быть внесены только программой восстановления во время загрузки. Неизменные системы хотя и обеспечивают большую целостность, однако затрудняют нормальную установку или исправление системных компонентов. На сходных принципах базируется и система защиты ресурсов Windows (WRP), однако здесь доступ к критическим системным файлам оставлен для системы обновления и восстановления.

Другой подход заключается в расчете и хранении эталонных значений (значений бесключевых хэш-функций, цифровых подписей или кодов аутентификации MAC, HMAC), позволяющих проводить проверку целостности файлов. При этом надежность такого подхода во многом определяется степенью защиты хранения эталонов.

Большинство развитых ОС сочетают оба подхода для обеспечения достаточного уровня контроля целостности системы наряду с гибкостью ее обновления и исправления. Существует три основных направления обеспечения целостности критических системных объектов:

– защита системных файлов и данных (параметров, конфигураций) во время работы ОС;

– защита от автономных атак (когда ОС не активна);

– контроль целостности системных объектов и их восстановление в случае обнаружения несанкционированного изменения.

Защита критически важных системных файлов во время работы ОС решается путем *ограничения доступа* к ним. В ОС Linux используется стандартная модель безопасности, в которой для изменения системных файлов могут быть произведены только от лица root. Дополнительно могут быть включены и настроены списки доступа ACL наподобие разрешений дискреционной защиты Windows, однако по умолчанию эта возможность в Linux системах отключена.

Однако, если компьютер был загружен с другого экземпляра ОС, например, с Live CD, или при непосредственном физическом доступе к носителю информации, ограничения доступа будут не активны и защищенные файлы ОС могут быть изменены. Для защиты от автономных атак может использоваться *полнодисковое шифрование*. Полнодисковое шифрование не позволяет нарушителю целенаправленно менять хранимую на диске информацию и эффективно контролировать изменения, однако сохраняется возможность изменения «вслепую» отдельных секторов и даже байтов. Таким образом, полнодисковое шифрование не может гарантировать отсутствия неконтролируемых изменений на диске, в том числе и в результате сбоев и отказов оборудования. При этом эффективно локализовать местоположение модифицированных данных не всегда представляется возможным.

Требование одновременного обеспечения конфиденциальности и целостности защищаемой информации приводит к необходимости использования *аутентифицированного шифрования* (Authenticated Encryption, AE). В настоящее время возможность полнодискового аутентифицированного шифрования или только контроль целостности содержимого диска реализована в ОС Linux (LUKS2) в качестве экспериментальной. Поэтому возникает необходимость в проверке и восстановлении критически важных для работы системы файлов.

Безопасная и доверенная загрузка ОС

Безопасная загрузка (UEFI Secure Boot) – это механизм проверки, обеспечивающий надежность кода, запускаемого микропрограммой. использование UEFI Secure Boot требует, чтобы каждый двоичный файл, запущенный при загрузке, проверялся по известным ключам доверенных поставщиков для идентификации проверенных аппаратных компонентов, микропрограмм и кода загрузчика ОС. Безопасная загрузка позволяет предотвратить выполнение неподписанного кода на этапе загрузки и выполнения ОС (например, как в ОС Windows, так и в ОС Linux проверяются подписи драйверов/модулей ядра, таким образом, безопасная загрузка не

позволяет системе выполнять случайно или преднамеренно измененный код в режиме ядра).

Когда какая-нибудь программа или ОС пытается загрузиться в память компьютера, Secure Boot сравнивает сигнатуру загрузочного кода с ключами, зашитыми непосредственно в BIOS и, если загрузочный код не проходит проверку подписи, загрузка тут же блокируется.

Однако при получении нарушителем физического доступа к компьютеру существует возможность подмены предустановленных ключей. Эта проблема решается при использовании аппаратно защищенного ключевого хранилища, например, TPM.

Доверенный платформенный модуль TPM (Trusted Platform Module) – криптографический сопроцессор, включающий аппаратный генератор случайных чисел и защищенное хранилище ключевой информации. Криптографические ключи никогда не покидают микросхему TPM, поэтому хост-процессор и ОС не могут напрямую обращаться к ним. TPM также участвует в формировании цепочек доверия при загрузке ОС. В этом случае цель использования TPM состоит в обеспечении целостности платформы, то есть в контроле того, что оборудование и ПО, участвующее в процессе загрузки, выполняет строго предопределенные действия до полной загрузки ОС. TPM является пассивным устройством, поэтому все действия по обеспечению целостности выполняются встроенным ПО аппаратных средств или ОС. Подсистема ввода/вывода UEFI может использовать TPM для защищенного хранения определенных метрик, описывающих изменения в конфигурации аппаратных компонентов компьютера, на основе которых принимается решение о целесообразности дальнейшей загрузки системы.

В настоящее время общепризнано, что для гарантированной защиты данных от несанкционированного доступа (НСД) необходимо использовать, принцип *пошагового контроля целостности*. Данный принцип заключается в выполнении взаимосвязанных последовательных шагов по контролю целостности аппаратных, а затем и программных компонентов компьютерной системы. При этом на раннем этапе активизации компьютера (включение питания, работа базовой подсистемы ввода/вывода) в системе должен присутствовать резидентный компонент безопасности, целостность встроенного ПО которого обеспечивается технологически. Данный модуль позволяет проводить аппаратный контроль целостности на ранних стадиях и, тем самым, обеспечивает неизменность и дальнейшую активизацию программных средств статического/динамического контроля целостности и разграничения доступа. Таким образом формируются цепочки доверия.

Статическая цепочка доверия SRTM (Static Root of Trust for Measurements) формируется при загрузке системы (рис. 2.6), а динамическая цепочка доверия DRTM (Dynamic Root of Trust for Measurements) – во время ее выполнения.

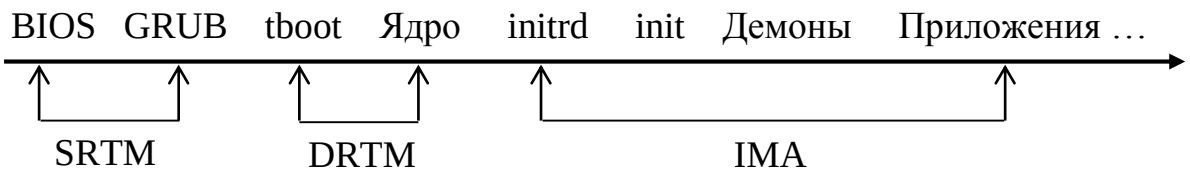

Рис. 2.6. Доверенная загрузка Linux системы с использованием TPM и IMA

Статическая цепочка доверия начинается, когда компьютер включается (или перезагружается), первое измерение выполняется аппаратным обеспечением (то есть процессором) для измерения модуля с цифровой подписью, предоставляемого производителем чипсета. Процессор проверяет подпись и целостность подписанного модуля перед его выполнением. Затем этот модуль измеряет первый модуль кода BIOS/UEFI, который может выполнять следующие измерения.

Таким образом, каждый компонент измеряет следующий компонент в цепочке. Пройдя всю цепочку от корня доверия, можно гарантировать неизменность текущего компонента (например, ОС), предоставляющего доверенную среду для более высокоуровневого компонента (например, приложений). Динамическая цепочка доверия формируется с некоторого известного состояния во время загрузки.

Таким образом, *доверенная загрузка* – это технология, которая обеспечивает цепочку доверия для всех компонентов во время загрузки.

Следует отметить, что существуют и отечественные решения, реализующие концепцию пошагового контроля целостности, которые получили название аппаратных модулей доверенной загрузки (АМДЗ). В отличие от TPM это узкоспециализированные активные аппаратные компоненты, которые принудительно перехватывают загрузку для проведения контрольных процедур. Как правило, АМДЗ входят в состав программно-аппаратных комплексов средств защиты информации от НСД.

Восстановление системы

К наиболее частым причинам, требующим восстановления системы, относятся:

– вирусное заражение, если системные файлы невозможно восстановить или невозможно удалить компоненты вредоносного ПО;

– неверные системные настройки;

– сбои после установки обновлений или стороннего ПО;

– утрата системных файлов;

– невозможность запуска системы.

ОС обладают различными возможностями и степенью автоматизации в зависимости от типа, версии и производителя, но, как минимум, существуют следующие возможности:

– сброс параметров ОС до заводских настроек по умолчанию;

– создание полной копии диска для последующего восстановления системного тома.

В первом случае может использоваться созданный в процессе установки диск восстановления или загрузочный установочный диск ОС, а в результате восстановления будет утрачено все установленное ПО и, возможно, пользовательские данные. Во втором случае все данные на диске, как относящиеся к системе, так и приложения и пользовательские данные будут восстановлены на момент создания образа диска. Однако хранение такого образа может потребовать достаточно большого объема дискового пространства. Образ носителя информации (физического диска, системного раздела) может быть создан с помощью стороннего специализированного ПО, в том числе, и свободно распространяемого (например, AccessData FTK Imager для Windows систем и MacOS, встроенной команды dd или утилит dc3dd, dcfldd, Clonezilla для Linux). Для создания образа диска, восстановления или установки с его использованием, могут использоваться решения резервного копирования и восстановления, такие как Acronis Backup/ True Image, Paragon Backup & Recovery и др.

Кроме двух указанных базовых возможностей, доступных практически для любой ОС, могут поддерживаться и другие механизмы восстановления, такие как восстановление из архивного образа системы или точек восстановления.

2.3. Журналирование файловых систем

Журналированием (journaling) называют механизм восстанавливаемой файловой системы (recoverable file system). Такие системы гарантируют целостность тома за счет приемов протоколирования, изначально рассчитанных на обработку транзакций. В случае сбоя ОС восстанавливаемая ФС возвращает свою целостность, выполняя процедуру восстановления путем обращения к хранящейся в файле журнала информации.

ФС снижает издержки протоколирования за счет объединения записей файла журнала в пакеты. То есть, за одну операцию ввода-вывода в журнал добавляется несколько записей. Журналируемая ФС может применять так называемую отложенную запись, увеличив интервалы между сбросами данных из кэша на диск, так как метаданные ФС поддаются восстановлению, если сбой происходит до того, как изменения в кэше переписаны на диск. Производительность кэширования в ФС с отложенной записью часто компенсирует издержки, связанные с протоколированием транзакций. Однако ни одна из этих ФС не гарантирует защиты пользовательских данных.

Протоколирование метаданных

Восстанавливаемость файловой системы NTFS, например, обеспечивается за счет записи в журнал всех операций, меняющих метаданные ФС. В NTFS применяется внутренняя реализация протоколирования, так называемая служба файла журнала (Log File Service, LFS). Кроме того, NTFS записывает все изменения метаданных, чтобы обеспечить целостность файловой системы в случае системного сбоя.

Служба файла журнала

Служба файла журнала представляет собой набор процедур режима ядра, локализованных внутри NTFS-драйвера, которые NTFS использует для доступа к файлу журнала. NTFS передает LFS указатель на открытый файловый объект, который соответствует нужному файлу журнала. LFS либо инициализирует новый журнал, либо вызывает диспетчер кэша для доступа к существующему журналу через кэш.

Порядок восстановления тома:

1. Сначала NTFS вызывает LFS для записи в кэшируемый журнал любых транзакций, меняющих структуру тома.

2. NTFS модифицирует том (в кэше тоже).

3. Диспетчер кэша предлагает LFS сбросить на диск файл журнала. Этот сброс реализуется путем обратного вызова диспетчера кэша с указанием страниц памяти, подлежащих выводу на диск.

4. После сброса журнала транзакций на диск диспетчер кэша записывает туда изменения тома (собственно, результаты операций над метаданными).

Восстановление ФС начинается автоматически при первом обращении к дисковому тому после перезагрузки.

Типы записей журнала

Механизм восстановления NTFS использует следующие типы записей журнала:

– записи обновления (update records);

– записи контрольных точек (checkpoint records).

Каждая запись представляет одну подоперацию в составе транзакции, создавая новый файл. Запись повторения в каждой записи обновления объясняет NTFS, каким образом следует повторно применить подоперацию к тому, а запись отмены – как отменить эту подоперацию.

NTFS является не единственной журналируемой ФС. Журналирование поддерживается многими другими ФС, например, ReFS (ОС Windows), ZTS (Sun Solaris, FreeBSD), HFS+ (MacOS), APFS (iOS, MacOS), ReiserFS, Btrfs, ext3/4 (ОС Linux) и другие. Многие из этих систем используют дополнительные технологии обеспечения целостности данных, такие как создание снимков ФС (snapshot) и поддержка технологии «копирование при записи» (copy-on-write), когда обновляются только те данные, которые были

изменены. При этом изменения в файловой системе и записи в журнале остаются в синхронизированном виде, даже если во время записи произошел сбой (например, было отключено электропитание).

2.4. Протоколирование и аудит

Процедура аудита применительно к защищенным ОС заключается в регистрации в специальном журнале, называемом *журналом аудита* или *журналом безопасности*, событий, которые могут представлять опасность для системы. Пользователи системы, обладающие правом чтения этого журнала, называются *аудиторами.*

Аудит необходим по следующим причинам:

– подсистема защиты компьютерной системы, не обладая интеллектом, неспособна отличить случайные ошибки пользователей от злонамеренных действий. Например, отличить случайную ошибку при вводе пароля от попытки подбора. Но при просмотре журнала администратор сможет установить, что же имело место на самом деле.

– если обнаружено, что против защищаемой системы проведена успешная атака, с помощью журнала аудита можно выяснить, когда и как она осуществлялась.

Специалисты по компьютерной безопасности считают, что привилегия работать с подсистемой аудита не должна предоставляться администраторам ОС, что повышает защищенность системы от несанкционированных действий администраторов. При этом отделение аудиторов от администраторов должно быть подкреплено организационно-административными мерами. На практике отделение аудиторов от администраторов практикуется только в крупных организациях (корпорациях, банках и т.п.). При этом обязанности аудитора часто берет на себя должностное лицо, которому непосредственно подчиняются системные администраторы, либо его заместитель.

Подсистема аудита ОС должна удовлетворять следующим требованиям:

– добавлять записи в журнал аудита могут только псевдопользователи, от имени которых выполняются системные процессы. Если предоставить эту возможность какому-то физическому пользователю, данный пользователь получит возможность компрометировать других пользователей, добавляя в журнал аудита соответствующие записи;

– ни один субъект доступа, в том числе и сама ОС, не имеет возможности редактировать или удалять отдельные записи в журнале аудита;

– для ограничения доступа к журналу аудита желательно применять дополнительные средства защиты, например, криптографические;

– в журнале аудита должны обязательно регистрироваться следующие события: попытки входа/выхода пользователей из системы; попытки изменения списка пользователей; попытки изменения политики безопасности, в том числе и политики аудита.

Окончательный выбор того, какие события должны регистрироваться в журнале аудита, определяется спецификой информации, хранимой и обрабатываемой в ОС. Политику аудита не следует рассматривать как нечто неизменное. Она должна оперативно реагировать на изменения в конфигурации ОС, в характере хранимой и обрабатываемой информации, и, особенно, на выявленные попытки атаковать защищаемую операционную систему.

В некоторых ОС подсистема аудита предусматривает возможность интерактивного оповещения аудиторов о регистрируемых событиях. Данная дополнительная функция позволяет аудиторам более оперативно реагировать на попытки преодоления злоумышленниками защиты ОС и тем самым повышает общую защищенность системы.

2.5. Криптографическая защита

Криптография является традиционным средством обеспечения конфиденциальности как хранимой, так и передаваемой информации. Однако круг задач, решаемых с использованием криптосистем, гораздо шире: контроль целостности, аутентификация сторон коммуникации, формирование общего секрета, обеспечение невозможности отказа сторон от авторства. Поэтому не удивительно, что большинство современных развитых систем, таких как большинство ОС универсального назначения, обладают встроенными криптографическими функциями.

В корпоративной среде аутентификация пользователей и устройств, работа с цифровыми сертификатами и защищенный обмен информации по сети, могут выполняться средствами базового ПО, например, ОС Windows и поддерживаемыми ею серверными службами [6]. ОС семейства Windows имеют ряд встроенных криптографических средств, поддерживающих локальное шифрование файлов (файловая шифрующая система EFS) и дисков (BitLocker), защищенную сетевую передачу информации (поддержка протоколов TLS и SSL для HTTPS, IPSec), сетевую аутентификацию с использованием криптографического протокола Kerberos (при использовании аппаратных ключей для входа в систему), управление сертификатами пользователей, устройств и служб (служба Certification Authority). *Служба сертификации* CA является основой развертывания корпоративной *инфраструктуры открытых ключей* PKI (Public Key Infrastructure) и осуществляет выпуск и управление сертификатами пользователей, компьютеров, служб и

устройств в локальной сети предприятия, что позволяет на ее основе осуществлять аутентификацию, локальное шифрование и защищенный сетевой обмен данными. Поэтому важным этапом является планирование инфраструктуры PKI, что предполагает, в частности получение ответа на вопрос о необходимости внедрения в корпоративной среде:

– защищенных механизмов сетевой аутентификации на основе сертификатов, в том числе с использованием смарт-карт;

– защищенных сетевых протоколов (SSL/TLS, IPsec);

– защищенных запросов чтения и записи данных в Active Directory с помощью Secure LDAP;

– защищенной электронной почты с возможностью шифрования и/или подписания сообщений;

– подписание программного кода приложений или документов;

– управление шифрующей файловой системой EFS.

Планирование самой службы CA предполагает, исходя из анализа существующей сетевой инфраструктуры и политики ИБ организации, определение:

– иерархической структуры корневых и подчиненных центров сертификации в корпоративной сети;

– используемых криптографических алгоритмов;

– сроков действия сертификатов;

– возможности централизованной архивации и восстановления закрытых ключей;

– возможности автоматической регистрации и обновления сертификатов и т.п.

Еще одна важная задача, решаемая подсистемой криптографической защиты информации (КЗИ) ОС, – обеспечение безопасности жесткого диска и других цифровых носителей информации. При этом средства криптографической защиты информации (СКЗИ) могут использоваться для обеспечения:

– конфиденциальности хранимых данных;

– контроля целостности хранимой информации;

– гарантированного уничтожения информации, в том числе и экстренного.

В зависимости от решаемых задач и рассматриваемых векторов атак может быть использован один из трех следующих подходов (рис. 2.7):

– шифрование всего носителя на уровне ядра ОС (полнодисковое шифрование);

– шифрование на уровне файловой системы;

– высокоуровневое шифрование (поверх уровня файловой системы).

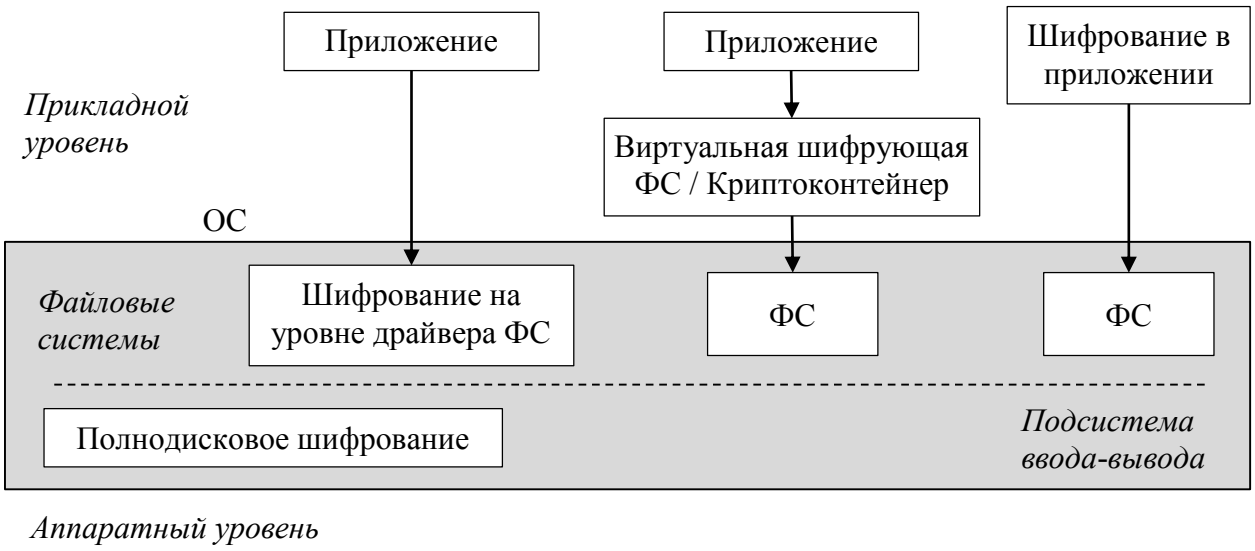

Рис. 2.7. Уровни программного шифрования

Каждый из этих уровней характеризуется рядом возможностей, таких как объем и тип защищаемой информации, эффективность (скорость) выполнения операций шифрования и последующего доступа к зашифрованной информации, возможность полного расшифрования (снятия шифрования) информации, возможность избирательного шифрования (шифрования отдельных объектов файловой системы – папок и файлов), возможность применения сторонних криптографических алгоритмов (например, определенных отечественными ГОСТ).

Поддержка отечественных криптографических стандартов (ГОСТ 34.10–2018, ГОСТ 34.11–2018, ГОСТ 34.12–2018, ГОСТ 34.13–2018) является важным требованием при построении защищенных ИС. Криптографические методы и средства защиты информации традиционно являются объектом правовых ограничений. Так, СКЗИ, используемые в ГИС, а также для обеспечения конфиденциальность информации, доступ к которой ограничен федеральными законами РФ, должны быть сертифицированы по соответствующему классу безопасности. Требования регулятора (ФСБ России) предполагают реализацию в таких средствах защиты российских криптоалгоритмов.

Использование отечественных криптографических стандартов рекомендовано и в некоторых других случаях, например, в защищенных информационных системах финансово-кредитных учреждений РФ, для формирования квалифицированной цифровой подписи, работы удостоверяющих центров и т.д.

Некоторые из встроенных в ОС СКЗИ допускают настройку, предполагающую выбор криптографических алгоритмов, режимов их работы и длин ключей. Набор доступных криптоалгоритмов зависит от конкретной

реализации. К сожалению, популярные криптографические функции, встроенные в системное ПО, равно как и встроенные реализации сетевых защищенных протоколов по умолчанию не поддерживают выбор отечественных криптоалгоритмов.

Высокоуровневое шифрование (на прикладном уровне) не зависит от существующей файловой системы и ядра ОС. Высокоуровневые реализации обеспечивают гибкий выбор объектов шифрования, вместе с тем, на этом уровне может зашифровываться лишь содержимое файлов, а вся метаинформация, включая имя, местоположение, тип и другие атрибуты файла остаются открытой. Кроме того, для шифрования каждого объекта должен быть использован свой секретный ключ (пароль), что не позволяет реализовать политику единого входа. Частичное решение этих проблем предлагается путем создания криптоконтейнера, представляющего, по сути, зашифрованный виртуальный диск, внутри которого создается своя ФС наподобие обычных виртуальных дисков. Криптоконтейнер может создаваться в пустом логическом разделе или в файле. После подключения криптоконтейнера ОС «видит» его как обычный логический том, а шифрование и расшифровывание объектов внутри криптоконтейнера осуществляется прозрачно.

Высокоуровневое шифрование обычно реализуется сторонними утилитами, которые, однако, могут быть включены в стандартную сборку ОС. В качестве примера можно привести включенное в старых версиях ОС Linux Ubuntu шифрование домашнего каталога пользователя с помощью виртуальной шифрующей файловой системы eCryptfs, реализуемое утилитой ecryptfs-utils. Широко известные утилиты создания криптоконтейнеров True Crypt/ Vera Crypt имеют реализации для различных ОС. Как криптоконтейнер реализовано и стандартное средство ОС Windows для полнодискового шифрования съемных носителей (с ФС, отличными от NTFS) BitLocker To Go.

Следует иметь ввиду, что высокоуровневое шифрование более ресурсоемко по сравнению с низкоуровневыми типами шифрования, однако оно может быть использовано как альтернатива в тех случаях, когда ФС или ядро ОС не имеют встроенных криптографических инструментов.

Шифрование на уровне файловой системы так же обеспечивает защиту лишь содержимого файла, оставляя открытой метаинформацию файловой системы. Основными преимуществами такого типа шифрования является возможность иметь в одной файловой системе как зашифрованные, так и открытые файлы, а также предоставить возможность шифрования разным пользователям, что позволяет криптографически изолировать данные пользователя от других, реализуя концепцию единого входа.

В ОС Windows шифрование на уровне драйвера ФС реализуется шифрующей файловой системой EFS (Encrypting File System), представляющей

собой надстройку над NTFS. Шифрование на уровне ФС ext4 добавлено в Linux ядро версии 4.1.

Шифрование всего носителя – *полнодисковое шифрование* (FDE, Full Disk Encryption) может быть реализовано как на аппаратном, так и на программном уровне. Полнодисковое шифрование предназначено для защиты от оффлайновых атак, когда доступ к носителю осуществляется при загрузке с другого экземпляра ОС или носитель информации физически подключается к другому компьютеру. Такая ситуация возникает, если нарушитель может получить физический доступ к выключенному устройству, например, при его краже или утере, а также в случае сдачи устройства для замены по гарантии, для ремонта, на таможенный досмотр и т.п. При этом традиционные методы управления доступом не работают. В то же время полнодисковое шифрование не позволит нарушителю извлечь ценную информацию с носителя.

Полнодисковое шифрование может быть реализовано программно на уровне ядра ОС, то есть уровнем ниже файловой системы, что позволяет файловым системам работать с зашифрованным диском абсолютно прозрачно. В то же время будут утрачены возможности оптимизации работы с SSD-накопителями за счет использования файловых систем, специально для этого предназначенных, таких как JFFS2, YAFFS и др.

После инициализации полнодискового шифрования, отказаться от него уже невозможно – диск будет шифроваться в любом случае, однако имеются различия в реализациях – в ОС Windows при снятии шифрования BitLocker ключи будут храниться в области метаинформации в открытом виде, а в Linux системах можно лишь использовать пустой пароль.

Другой особенностью полнодискового шифрования является полная доступность зашифрованной информации после разблокировки носителя. Как только зашифрованный диск был разблокирован, ключи шифрования помещаются в оперативную память компьютера, что делает их потенциально уязвимыми для компрометации. Так, снятие дампа оперативной памяти позволяет нарушителю получить ключи шифрования, а значит и доступ к всему содержимому зашифрованного диска.

Для защиты системы полнодискового шифрования и затруднения снятия дампа оперативной памяти следует:

– запретить загрузку ОС с внешнего носителя: USB-устройства и CD (или сделать первым загрузочным устройством жесткий диск), а также установить пароль на изменение настроек BIOS/UEFI;

– использовать режим безопасной загрузки UEFI;

– блокировать компьютер, если он временно не используется (с требованием повторной аутентификации для разблокировки),

– заблокировать FireWire и другие DMA-порты (если они установлены на компьютере) в BIOS/UEFI, с помощью групповой политики или специализированного программного обеспечения (например, Kaspersky Endpoint Security, DeviceLock и т.п.).

Программная реализация полнодискового шифрования для системных дисков требует прохождения предварительной аутентификации (Pre-Boot Authentication, PBA) для получения от пользователя информации (пароля, ключевого файла), позволяющей разблокировать носитель информации, то есть получить доступ к ключам шифрования данных, хранящимся в зашифрованном виде. Поэтому некоторая часть диска, содержащая модули начальной предаутентификационной загрузки и метаданные, остается незашифрованной. Подмена и модификация загрузочного кода, возможная при физическом доступе к диску, позволяет нарушителю контролировать процесс предаутентификации и, как следствие, получить доступ к зашифрованным данным.

В ОС Windows контроль целостности файлов загрузки обеспечивается использованием встраиваемого в материнскую плату компьютера аппаратного компонента – доверенного платформенного модуля TPM, что предполагает, кроме того, жесткую аппаратную привязку зашифрованного носителя. На территории Российской Федерации компьютеры с TPM недоступны из-за экспортных ограничений.

В ОС Linux открытый раздел с файлами загрузчика и конфигураций может быть перенесен на съемный носитель, с которого будет осуществляться загрузка. Кроме того, загрузчик нового поколения Grub2 поддерживает непосредственную загрузку из UEFI с активным Secure Boot. В этом случае можно настроить полное шифрование диска LUKS, включая и раздел /boot. Непосредственная загрузка из UEFI поддерживается в случае, если ядро ОС Linux было скомпилировано с параметром CONFIG_EFI_STUB.

Системы полнодискового шифрования обеспечивают конфиденциальность хранимых данных, но не их целостность. Это означает, что изменения, произведенные нарушителем, или случайная прямая запись на зашифрованный диск не могут быть эффективно обнаружены, хотя и могут повлечь сбои и невозможность расшифрования части хранимой информации.

Некоторые файловые системы (например, ZFS и BtrFS, ReFS), осуществляющие расчет контрольных сумм для каждого логического блока и их сверку при чтении данных. Сквозной контроль целостности делает эти файловые системы достаточно ресурсоемкими, что отчасти компенсируется рядом оптимизаций, таких как применение принципа «копирование при записи» (copy-on-write), дедупликация, сжатие и др. Восстановление данных достигается за счет возможностей зеркалирования, создания моментальных снимков файловой системы и инкрементальных архивов.

Следует, однако, отметить, что применяемый ZFS/ BtrFS механизм расчета контрольных сумм на основе некриптографических контрольных сумм CRC32 и бесключевых хэш-функций не обеспечивает достаточной защиты от преднамеренного изменения данных (в частности, в ходе оффлайновых атак), поскольку нарушитель может осуществить подмену как самих данных, так и значения соответствующих контрольных сумм. Для решения этой проблемы требуется использование криптографических контрольных сумм, значения которых невозможно рассчитать без знания секретной информации (ключа), – кодов аутентификации MAC, представляющих собой особые режимы работы блочных шифров, либо HMAC, совмещающих использование хэш-функций и ключевой информации.

Одновременное обеспечение конфиденциальности и целостности защищаемой информации возможно с использованием аутентифицированного шифрования (Authenticated Encryption, AE). Идея аутентифицирована шифрования реализована в настоящее время в ОС Linux. Так, в Linux ядро, начиная с версии 4.12, включена поддержка утилиты dm-integrity (обеспечивает контроль целостности и позволяет включить журналирование для восстановления после сбоев) и обновленного модуля dm-crypt с аутентифицированным шифрованием по протоколу LUKS2. Вторая версия утилиты cryptsetup по умолчанию использует LUKS1, но реализует и новые функции LUKS2 (аутентифицированное шифрование или отдельно контроль целостности на основе CRC32 или бесключевых хэш-функций), которые однако помечены как экспериментальные и не рекомендованы для промышленного использования.

Следует отметить, что контроль целостности блоков данных, как отдельная функция, так и в составе аутентифицированного шифрования, требует дополнительных накладных расходов на хранение присоединенных данных (например, используемых в проверках контрольных сумм), которые для носителей большого размера могут оказаться весьма существенными. Например, для пустого диска объемом 10 Гб, отформатированного в файловой системе ReFS, дисковое пространство, выделенное под контрольные данные, составит около 700 Мб.

Полнодисковое шифрование может быть рассмотрено и с точки зрения гарантированного уничтожения данных. После удалении ключей шифрования из области метаданных, чтение информации тома станет невозможным. В экстренных случаях такой подход может служить хорошей альтернативой традиционным методам удаления информации, поскольку позволяет пресечь доступ к информации на зашифрованном носителе практически мгновенно.

Здесь, однако, существует ряд оговорок, например, копии ключей могут быть депонированы в облако или сохранены на внешнем носителе,

откуда могут быть извлечены в дальнейшем. Однако, некоторые реализации полнодискового шифрования, например, в Kali Linux или в ОС FreeBSD (инструмент geli) подобную возможность предусматривают специально.

2.6. Гарантированное уничтожение информации

При удалении файла обычными средствами ОС данные файла остается на диске (в большинстве файловых систем). В таблице размещения файлов соответствующий файл помечается как удаленный, а занятые им сектора становятся «свободными» для дальнейшей записи. Но само тело файла в этих секторах остается. Если эти сектора ничем не занять, то удаленный файл вполне можно восстановить. Этот принцип лежит в основе работы различных утилит восстановления удаленных файлов.

Потенциальная возможность восстановления удаленных с носителя данных может порождать угрозы нарушения конфиденциальности. Потребность хранения и обработки все больших объемов информации приводит к тому, что носители периодически сменяются на новые, большего объема, работающие быстрее и имеющие больше функций. Старые устройства выбрасываются, продаются или утилизируются без дополнительных предосторожностей, что может привести к утечкам конфиденциальной информации. Шифрование данных может ослабить угрозу их неконтролируемого извлечения при условии, что криптографический ключ надежен и правильно контролируется (то есть, сам не хранится на диске и не является частью остаточной информации), в этом случае данные на устройстве могут оказаться неизвлекаемыми. Политикой безопасности организации должны быть предусмотрены процедуры для утилизации устройств хранения информации, подразумевающие их санацию (безвозвратное уничтожение информации).

Гарантированное уничтожение данных используется для защиты конфиденциальной информации от утечек, вызванных неправильной утилизацией носителей информации и их дальнейшим использованием нарушителем. Применение методов гарантированного уничтожения конфиденциальной информации на ее носителях актуально в случаях:

– продажи/дарения носителя;

– возможности потери или хищения неиспользуемого носителя;

– списание носителей или содержащих их устройств за ненадобностью;

– передача носителей или содержащих их устройств в другое подразделение или организацию.

При этом следует иметь ввиду, что многие устройства, обычно не относящиеся к компьютерной технике (мобильные телефоны, цифровые

фото- и видеокамеры, фоторамки, фитнес браслеты и пр.) также могут содержать носители информации, как встроенные, так и съемные.

Применяемые методы гарантированного удаления информации во многом определяются физической природой носителя информации и зависят от его типа (HDD, SDD, оптический диск CD/DVD). В общем, возможны *программные* способы уничтожения записанной информации, либо деструктивное воздействие на сам носитель, приводящее к невозможности его дальнейшего использования (*механические* и *физические* методы уничтожения). Выбор применяемых методов зависит, прежде всего, от намерений дальнейшего использования носителя. Как правило, для уничтожения действительно секретной информации применяют деструктивные методы, но для данных, не представляющих особой ценности, допустимо использовать и программные алгоритмы. Таких алгоритмов существует великое множество.

Программные алгоритмы гарантированного уничтожения данных на HDD носителях базируются на перезаписи (возможно, многократной) информации с помощью штатных средств записи информации на магнитных носителях. В случае применения программных методов уничтожения информации на диске, он может быть повторно использован в других ПК, после инсталляции новой ОС и приложений.

Традиционный способ удаления данных с HDD – полное форматирование логических разделов. В старых версиях ОС Windows (XP и более ранние) при полном форматировании диска система не записывала нули в каждый сектор. Поэтому в ОС Windows XP следовало форматировать диск в командной строке с помощью команды format с параметром /p, задающим число проходов перезаписи информации нулевыми значениями. Начиная с Windows Vista логика команды форматирования изменена – перезапись нулями выполняется по умолчанию, и параметр /p становится избыточным.

Очистка данных из свободного пространства диска в расположении папки или всего тома после форматирования может быть осуществлено с помощью команды cipher.exe (для файловой системы NTFS). Утилита cipher с параметром /w производит удаление данных из неиспользуемых блоков тома с файловой системой NTFS. На отформатированном диске утилита cipher стирает оставшиеся данные, перезаписывая весь диск сначала нулями, затем единицами и после этого – случайными значениями.

На сходных принципах уничтожения информации за счет ее многократной перезаписи основано большинство алгоритмов программного уничтожения данных и реализующих их утилит. Приведем лишь несколько возможных алгоритмов.

– ГОСТ Р 50739-95 использует два прохода – в первом цикле производится запись 0, во втором – псевдослучайных чисел.

– Немецкий стандарт BSI Verschlusssachen- IT- Richtlinien (VSITR) позволяет использовать от двух до шести проходов (в зависимости от классификации информации), записывающих поочередно псевдослучайную последовательность и ее XOR- комплимент. Последним проходом записывается последовательность 01010101.

– американский специалист в области криптографии Брюс Шнайер предлагает использовать метод, включающий 7 проходов перезаписи. 1-й проход – запись логических единиц, 2-й – нулей, с 3-го по 7-й – псевдослучайных последовательностей.

Следует отметить, что многократная перезапись данных может занимать достаточно длительное время для современных носителей большой емкости и не подходит для экстренного удаления информации.

Примерно с 2001 года в конструкции жестких дисков ATA IDE и SATA на уровне контроллера включена поддержка стандарта Secure Erase, предполагающая перезапись диска тремя разными шаблонами. Стандарт Secure Erase поддерживается также и твердотельными (SSD) накопителями, и реализован на уровне прошивки.

Большинство поставщиков дисков поставляют утилиту, запускающую диагностику S.M.A.R.T., которая позволяет контролировать состояние накопителя, обновлять его прошивку и вызывать процедуру Secure Erase. Но при этом пользователь должен принять соглашение, в соответствии с которым признает, что предлагаемый инструмент может необратимо повредить ваш диск. К утилитам подобного рода относятся Data Lifeguard (Western Digital), Drive Fitness Test (Hitachi), OCZ Toolbox, Samsung Magician (только для твердотельных накопителей) и SeaTools (Seagate). Кроме того, в ОС Linux (и других Unix-подобных ОС) процедура Secure Erase может быть выполнена с помощью стандартной утилиты командной строки hdparm.

Выполнение процедуры Secure Erase не должно прерываться, а диск в большинстве случаев должен быть подключен к внутреннему интерфейсу. Процедура не должна вызываться при подключении диска через USB-интерфейс (диск может перестать определяться). Также может потребоваться изменение некоторых настроек BIOS/UEFI. Для старых дисков Secure Erase имеет ряд потенциальных недостатков, обусловленных не вполне корректной реализацией ее алгоритма.

Следует отметить существенные отличия программных процедур гарантированного удаления информации для твердотельные SSD накопителей. Эти устройства имеют ряд особенностей физической организации, в частности, для наиболее распространенного типа флэш-памяти (NAND, 3D NAND) на уровне контроллера устройства обычно реализованы методы выравнивания изнашивания, заключающиеся в перемещении данных из наиболее изношенных участков памяти в менее изношенные. Контроллер

SSD выполняет подмену логических адресов, так что невозможно гарантировать, что данные физически будут записаны именно в ту ячейку памяти, к которой произошло обращение ОС. Поэтому выполнение многократной перезаписи данных для большинства SSD накопителей не имеет смысла.

Обнаружение контроллером флэш-накопителя свободных участков данных файловых систем для последующей очистки может производиться двумя способами.

1. С помощью ATA-команды Trim, работающей совместно с ОС. ОС, поддерживающая эту ATA-команду, автоматически при удалении файлов или при форматировании передает контроллеру SSD сведения о неиспользуемых участках данных, которые можно использовать для выравнивания изнашивания. Сама ОС не перезаписывает эти блоки и не стирает информацию. Контроллеру устройства всего лишь передается массив адресов ячеек, которые не содержат полезной информации. С этого момента контроллер может начать фоновый процесс удаления данных из ячеек.

2. Путем обработки структур файловых систем контроллером флэш-накопителя. Контроллер без участия ОС производит чтение хранимых данных, обрабатывает структуры наиболее распространенных файловых систем и определяет незанятые участки данных, которые можно очистить. При этом очистка производится достаточно быстро.

Вызов команды Trim в ОС Windows возможен с помощью утилиты оптимизации диска (в ОС Windows версий 8 и 10), которая реализована для SSD накопителей как вызов команды Trim. После поступления контроллеру SSD диска команды Trim дальнейшее никак не зависит от действий пользователя или ОС – контроллер начнет очистку ненужных ячеек по своим внутренним алгоритмам. При этом даже если очитка еще не завершена, при обращении к удаленным данным, контроллер гарантированно не будет их выдавать. С практической точки зрения, удаленные ранее любым способом данные станут недоступными для чтения как с компьютера, так и на специальном стенде. Это утверждение верно, если Trim функционирует корректно на системе, где используется накопитель. Последнее предполагает соблюдение ряда условий:

1. диск должен быть подключен напрямую (SATA, NVME);
2. том должен быть размечен в ФС NTFS или ReFS (для ОС Windows);
3. Trim должны поддерживать как драйверы, так и BIOS компьютера.

Проверить работоспособность Trim в Windows можно командой fsutil behavior query DisableDeleteNotify (командная строка должна быть запущена от имени администратора). Результат: 0 – Trim включен и работает корректно; 1 – Trim неактивен.

Для USB- накопителей (внешних SSD) Trim с большой вероятностью не будет активен, хотя и может поддерживаться на уровне контроллера.

Trim не работает с виртуальными дисками, хранящимися в виде файлов образов, что ограничивает использование команды в виртуальной среде. Trim не всегда поддерживается в RAID-массивах. Остановить процесс фоновой очистки Trim невозможно – если на SSD- накопитель подается питание, контроллер продолжит очистку.

Следует отметить, что заметная часть емкости SSD накопителя (в некоторых моделях – до 10%) отводится под резервный неадресуемый пул. В теории ячейки в этом пуле должны очищаться; на практике же из- за многочисленных особенностей реализации и ошибок в прошивках данные могут физически оставаться на диске даже после завершения очистки. Данные, попавшие в такие вторичные местоположения, никак не контролируются ОС и могут оставаться в значительной степени нетронутыми. С другой стороны, их извлечение недеструктивными способами не представляется возможным. Если же из SSD извлекут микросхемы, то физически оставшиеся в ячейках данные удастся считать. Более того, даже если процесс очистки, казалось бы, завершен, в неадресуемом резервном пуле могут остаться ячейки, в которых содержатся удаленные данные.

Таким образом, полное удаление информации на внутреннем SSD носителе возможно при вызове ATA-команды Secure Erase или форматировании с последующим вызовом команды Trim. Это гарантирует невозможность извлечения данных недеструктивными способами. Однако полной гарантии невозможности восстановления данных это не дает.

Поэтому организации, работающие с секретной информацией, не признают иных способов очистки SSD, кроме физического уничтожения носителя.

Деструктивные (разрушающие) – механические и физические – методы уничтожения информации на магнитных носителях (HDD, стримерных картриджей, видео- и аудио кассет, дискет и т.д.) приведены в таблице 2.1. Некоторые из этих методов (например, механический) могут применяться и к носителям других типов.

Таблица 2.1. Деструктивные методы гарантированного уничтожения

Метод	Тип разрушающего воздействия
Механический	Измельчение носителя, механическое разрушение
Термический	Нагревание носителя до температуры разрушения его основы (или до точки Кюри)
Пиротехнический	Разрушение носителя взрывом
Химический	Разрушение рабочего слоя или основы носителя химически агрессивными средами

Окончание табл. 2.1

Метод	Тип разрушающего воздействия
Размагничивание носителя	Помещение в медленно убывающее магнитное поле
Намагничивание носителя	Помещение в мощное магнитное поле
Радиационный	Разрушение ионизирующим излучением

Применение большинства деструктивных методов требует обеспечения безопасности оператора. В России существует большое количество разработок области аппаратного, в том числе экстренного, уничтожения данных, а точнее самих носителей информацией (например, утилизатор носителей «Прибой», комплексы «Цунами» и «Горгона», разработанные компанией КСУ).

Методики надежного уничтожения неперезаписываемых оптических дисков (CD-R, DVD-R) включают: отслоение хранящего информацию слоя, измельчение, разрушение электрической дугой (как при помещении в микроволновую печь) и помещение в растворитель поликарбоната (например, ацетон).

ГЛАВА 3. ОСОБЕННОСТИ ЗАЩИТЫ В РАЗЛИЧНЫХ ОС

3.1. Система безопасности ОС Windows

Компоненты безопасности

В реализацию системы безопасности Windows входят следующие компоненты и базы данных.

Монитор безопасности (Security Reference Monitor, SRM) – компонент, который находится в исполняющей системе Windows (%System Root%\ System32\Ntoskrnl.exe) и отвечает за определение структуры данных маркеров доступа для представления контекста безопасности, выполнение проверки безопасности доступа к объектам, работу с привилегиями (правами пользователей) и генерацию любых итоговых сообщений проверки безопасности.

Подсистема проверки подлинности локальной системы безопасности (Local Security Authority Subsystem, LSASS) – запускаемый процессом пользовательского режима образ %SystemRoot%\System32\Lsass.exe, отвечает за политику безопасности локальной системы, аутентификацию пользователя и отправку сообщений проверки безопасности журналу событий (Event Log).

База данных политики LSASS – содержит настройки политики безопасности локальной системы. Эта база данных хранится в реестре в ACL-защищенной области в разделе HKLM\SECURITY. Она включает информацию, которая определяет, каким доменам доверена аутентификация попыток входа в систему, у кого есть права на доступ к системе и каким образом осуществляется этот доступ (интерактивные, сетевые или служебные регистрации), кому и какие привилегии назначены и какие виды проверки безопасности следует выполнять. В базе данных политики LSASS также хранятся «секреты», включающие информацию о входе в систему, использующуюся для кэшированных входов в домены и входов в службы Windows под учетной записью пользователя.

Администратор учетных данных в системе защиты (Security Accounts Manager, SAM) – служба, отвечающая за управление базой данных, содержащей имена пользователей и определения групп на локальной машине. SAM-служба, реализованная в виде библиотеки %SystemRoot%\ System32\ Samsrv.dll, загружается в процесс LSASS.

База данных SAM – содержит определения локальных пользователей и групп наряду с их паролями и другими атрибутами. На контроллерах доменов служба SAM не хранит сведений о пользователях, определенных на домене, но хранит информацию определения учетной записи и пароля системного администратора, имеющего право на восстановление системы. Эта база данных хранится в реестре в разделе HKLM\SAM.

Active Directory – служба каталогов, которая содержит базу данных, хранящую информацию об объектах в домене. Домен является совокупностью компьютеров и связанных с ними группами безопасности, которые управляются как единое целое. Active Directory сохраняет информацию об объектах в домене, куда включаются пользователи, группы и компьютеры. В Active Directory хранится информация о паролях и привилегиях для пользователей и групп домена, которая тиражируется на компьютерах, назначаемых в качестве контроллеров домена. Сервер Active Directory, реализованный в виде библиотеки %SystemRoot%\System32\Ntdsa.dll, запускается в контексте процесса LSASS.

Пакеты аутентификации (Authentication packages) – включают динамически подключаемые библиотеки – dynamic-link libraries (DLL), которые запускаются как в контексте процесса LSASS, так и в контекстах клиентских процессов и реализуют имеющуюся в Windows политику аутентификации. DLL-библиотека аутентификации отвечает за аутентификацию пользователей, реализуя ее путем проверки соответствия введенного имени пользователя и пароля и возвращения процессу LSASS в случае такого соответствия информации, уточняющей детали пользовательской идентичности с точки зрения безопасности, которые используются LSASS для генерации маркера доступа.

Интерактивный диспетчер входа в систему (Winlogon) – процесс пользовательского режима, в котором выполняется образ %SystemRoot%\ System32\Winlogon.exe, отвечающий за реагирование на SAS и за управление интерактивными сеансами входа в систему. К примеру, Winlogon создает первый пользовательский процесс, когда пользователь входит в систему.

Пользовательский интерфейс входа в систему (Logon User Interface (LogonUI) – процесс пользовательского режима, выполняет образ %SystemRoot%\System32\LogonUI.exe, который предоставляет пользователям интерфейс, используемый ими для самоидентификации в системе.

Поставщики учетных данных (Credential Providers, CP) – находящиеся в процессе COM-объекты, запускают процесс LogonUI (стартующий по запросу со стороны Winlogon при выполнении SAS) и используются для получения имени пользователя и пароля, PIN-кода смарт-карты или биометрических данных (например, отпечатка пальца). Стандартными CP являются библиотеки %SystemRoot%\System32\authui.dN и %SystemRoot%\ System32\ SmartcardCredentialProvider.dll.

Служба входа в сеть (Network logon service, Netlogon) – служба Windows (%SystemRoot%\System32\Netlogon.dll), устанавливает безопасный канал к контроллеру домена, по которому отправляются запросы безопасности, например, интерактивный вход в систему (если на контроллере домена запущена Windows NT 4) или проверка аутентификации диспетчера

LAN Manager и диспетчера NT LAN Manager (v1 и v2). Netlogon также используется для входов в систему со стороны Active Directory.

Драйвер устройства безопасности ядра (Kernel Security Device Driver, KSecDD) – библиотека функций режима ядра, реализует интерфейсы усовершенствованной системы вызова локальных процедур – advanced local procedure call (ALPC), которые используются для обмена данными в пользовательском режиме с LSASS другими компонентами безопасности режима ядра, включая такой компонент, как шифрующая файловая система (Encrypting File System, EFS). KSecDD находится в файле %SystemRoot%\ System32\Drivers\Ksecdd.sys.

AppLocker – механизм, позволяющий администраторам определять, какие исполняемые файлы, DLL-библиотеки и сценарии могут использоваться конкретными пользователями и группами. AppLocker состоит из драйвера (%SystemRoot%\System32\Drivers\AppId.sys) и службы (%System Root%\ System32\AppIdSvc.dll), выполняемой в процессе SvcHost.

Защита объектов

В число защищаемых в Windows объектов входят файлы, устройства, почтовые ящики, каналы (именованные и безымянные), задания, процессы, потоки, события, ключевые события, пары событий, мьютексы, семафоры, общие разделы памяти, порты завершения ввода-вывода, LPC-порты, таймеры ожиданий, маркеры доступа, тома, станции окон, рабочие столы, сетевые ресурсы, службы, разделы реестра, принтеры, объекты Active Directory и т. д. – теоретически все, что управляется *диспетчером объектов* исполняющей системы. На практике те объекты, которые не открыты пользовательскому режиму (например, объекты драйверов), обычно не защищены.

Windows использует единую модель и механизм безопасности, как для файлов в ФС, так и для объектов исполняющей системы. Объекты исполняющей системы отличаются от файлов только в методах доступа, поддерживаемых каждым типом объектов. Windows требует аутентифицированного входа в систему перед получением доступа к любым системным ресурсам. Когда процесс запрашивает дескриптор объекта, диспетчер объектов и система безопасности используют идентификатор безопасности запрашивающего субъекта и дескриптор безопасности объекта, чтобы определить, может ли запрашивающему быть присвоен дескриптор, предоставляющий желаемый доступ к объекту.

Поток может получить другой контекст безопасности, нежели его процесс. Этот механизм называется заимствованием прав (impersonation), и когда поток заимствует чьи-либо права, механизм проверки безопасности использует контекст безопасности потока, а не того процесса, которому принадлежит поток. Если поток ничьих прав не заимствует, проверка

безопасности возвращается обратно к использованию контекста безопасности того процесса, который владеет потоком. Важно иметь в виду, что все потоки в процессе используют общую таблицу дескрипторов, поэтому, когда поток открывает объект, даже если он заимствует чьи-либо права, доступ к объекту имеют все потоки процесса.

Проверки прав доступа

Модель безопасности Windows требует, чтобы поток, прежде чем открыть объект, указал, какого типа действия он хочет совершить над объектом. Для выполнения проверок прав доступа, основанных на желаемом потоком доступе, диспетчер объектов вызывает SRM, и, если доступ предоставляется, процессу потока назначается дескриптор, с которым поток (или другие потоки процесса) может выполнять дальнейшие операции над объектом (диспетчер объектов записывает предоставленные права доступа в дескриптор в таблице дескрипторов процесса). На выходе получаем либо «да», либо «нет», что показывает, дает модель безопасности потоку запрошенный доступ или не дает.

Идентификаторы безопасности

Вместо использования имен Windows использует идентификаторы безопасности – Security Identifiers (SID). SID-идентификаторы имеются у пользователей, а также у локальных и доменных групп, у локальных компьютеров, доменов, участников доменных групп и служб. SID представляет собой числовое значение переменной длины, состоящее из номера версии SID-структуры, 48-разрядного значения идентификатора *полномочий* и переменного количества 32-разрядных кодов значений *подчиненных полномочий* или относительных идентификаторов – relative identifier (RID). Значение полномочий идентифицирует агента, выдавшего SID, и этим агентом обычно является локальная система Windows или домен. Значения подчиненных полномочий идентифицируют представителей, имеющих отношение к выдавшему полномочия, а RID-идентификаторы являются просто способом, применяемым в Windows для создания уникальных SID на основе общего базового SID. Из-за большой длины SID-идентификаторов Windows старается сгенерировать внутри каждого SID по-настоящему случайное значение. Практически невозможно, чтобы Windows выдала один и тот же SID на машине или домене, или где-либо еще дважды.

При текстуальном отображении каждый SID содержит префикс S, и его различные компоненты отделены друг от друга дефисами:

S-1-5-21-1463437245-1224812800-863842198-1128

В данном SID номером версии служит цифра 1, значением идентификатора полномочий служит цифра 5 (полномочия безопасности Windows), а затем следуют четыре значения подчиненных полномочий плюс один RID (1128), который составляет оставшуюся часть SID. Это SID домена, а у

локального компьютера домена будет SID с тем же номером версии, значением идентификатора полномочий и количеством значений подчиненных полномочий.

При установке Windows программа Windows Setup выдает компьютеру SID машины. Windows назначает SID-идентификаторы локальным учетным записям, имеющимся на компьютере. Каждый SID локальной учетной записи создается на основе исходного компьютерного SID и в конце имеет RID. RID-идентификатор для пользовательских учетных записей и групп начинается с 1000 и становится больше на 1 с каждым новым пользователем или группой. По аналогии с этим утилита Dcpromo.exe (Domain Controller Promote), используемая для создания нового домена Windows, повторно использует SID компьютера, повышаемого до контроллера домена в качестве SID домена, и она заново создает SID для компьютера, если он когда-либо будет понижен в должности. Windows выдает для нового домена SID-идентификаторы учетных записей, основанные на SID-идентификаторе домена с добавлением RID-идентификатора (который снова начинается с 1000 и повышается на 1 для каждого нового пользователя или группы). RID со значением 1028 показывает, что SID является двадцать девятым выпущенным в домене.

Windows выдает SID-идентификаторы, которые состоят из SID компьютера или домена с предопределенным RID-идентификатором для множества предопределенных учетных записей и групп. Например, RID для учетной записи администратора имеет значение 500, а RID для гостевой учетной записи имеет значение 501. Например, учетная запись локального администратора имеет в своей основе SID компьютера с добавленным к нему RID, который имеет значение 500:

S-1-5-21-13124455-12541255-61235125-500

Windows также определяет ряд встроенных локальных и доменных SID для представления широко известных групп. Например, SID, идентифицирующий любые учетные записи (за исключение анонимных пользователей), является всеобщим – Everyone SID: S-1-1-0. Другим примером группы, которая может быть представлена SID-идентификатором, является сетевая группа, то есть группа, представляющая пользователей, зарегистрировавшихся на машине по сети. SID сетевой группы имеет значение S-1-5-2. В отличие от пользовательских SID эти SID-идентификаторы являются предопределенными константами и имеют одинаковые значения на каждой системе Windows и домене во всем мире.

И наконец, Winlogon создает уникальный SID входа в систему для каждого интерактивного сеанса входа. Обычно SID входа в систему применяется в элементе управления доступом – Access Control Entry (ACE), который разрешает доступ во время сеанса входа клиента в систему.

Уровни целостности

Уровни целостности могут заменить разграничение доступа, чтобы провести различия между процессом и объектами, запущенными от имени одного и того же пользователя и находящимися в его владении, предоставляя возможность изоляции кода и данных в рамках учетной записи пользователя. Механизм мандатного контроля целостности (Mandatory Integrity Control, MIC) позволяет SRM располагать более подробной информацией о природе вызывающего процесса путем его ассоциации с уровнем целостности. Он также предоставляет информацию о доверии, необходимую для доступа к объекту путем определения для него уровня целостности.

Эти уровни целостности определяются с помощью SID. Хотя уровни целостности могут иметь произвольные значения, в системе используются пять основных уровней для отделения друг от друга уровней привилегий. Описание этих уровней дано в табл. 3.1 [9].

Таблица 3.1. SID-идентификаторы уровней целостности

sid	Имя (уровень)	Использование
S-1-16-0x0	Untrusted (0) (ненадежный)	Процессами, запущенными группой Anonymous, блокирует большинство доступов по записи
S-1-16-0x1000	Low (1) (низкий)	Защищенным режимом Internet Explorer, блокирует доступ по записи к большинству объектов системы (таких как файлы и разделы реестра)
S-1-16-0x2000	Medium (2) (средний)	Обычными приложениями, запущенными при включенной системе UAC
S-1-16-0x3000	High (3) (высокий)	Административными приложениями, запущенными через повышение уровня полномочий при включенной системе UAC, или обычными приложениями при выключенной системе UAC и при наличии у пользователя прав администратора
S-1-16-0x4000	System (4) (системный)	Службами и приложениями системного уровня (например, Wininit, Winlogon, Smss и т. д.)

Маркеры

Для идентификации контекста безопасности процесса или потока SRM использует объект под названием маркер (или маркер доступа). Контекст безопасности состоит из информации, описывающей учетную запись, группы и привилегии, связанные с процессом или потоком. Маркеры также включают такую информацию, как ID сеанса, уровень целостности и состояние виртуализации UAC.

При выполнении процесса входа в систему (рассматриваемого в конце данной главы) процесс LSASS создает исходный маркер для представления пользователя, входящего в систему. Затем он определяет, отно-

сится ли пользователь к группе, имеющей высокие полномочия, или обладает ли он высокими привилегиями. На данном этапе проверка принадлежности к группам ведется в следующем порядке [9]:

– Built-In Administrators (Встроенная группа администраторов);
– Certificate Administrators (Администраторы сертификатов);
– Domain Administrators (Администраторы доменов);
– Enterprise Administrators (Администраторы предприятий);
– Policy Administrators (Администраторы политик);
– Schema Administrators (Администраторы схем);
– Domain Controllers (Контроллеры доменов);
– Enterprise Read-Only Domain Controllers (Контролеры принадлежащих предприятию доменов, предназначенных только для чтения);
– Read-Only Domain Controllers (Контролеры доменов, предназначенных только для чтения);
– Account Operators (Операторы учетных записей);
– Backup Operators (Операторы по созданию резервных копий);
– Cryptographic Operators (Операторы криптографических систем);
– Network Configuration Operators (Операторы сетевой конфигурации);
– Print Operators (Операторы вывода на печать);
– System Operators (Системные операторы);
– RAS Servers (Серверы служб удаленного доступа – RAS);
– Power Users (Пользователи с повышенными привилегиями);
– Pre-Windows 2000 Compatible Access (Доступ в режиме совместимости с системами, предшествовавшими Windows 2000).

Проверяется также наличие следующих привилегий:

– SeBackupPrivilege (на создание резервной копии);
– SeCreateTokenPrivilege (на создание маркера);
– SeDebugPrivilege (на отладку);
– SeImpersonatePrivilege (на работу от имени других пользователей);
– SeLabelPrivilege (на создание меток);
– SeLoadDriverPrivilege (на загрузку драйверов);
– SeRestorePrivilege (на восстановление системы);
– SeTakeOwnershipPrivilege (на смену владельца);
– SeTcbPrivilege (на работу с блоком управления потоком).

Маркеры варьируются по длине, поскольку у разных учетных записей пользователей имеются разные наборы привилегий и связанные с ними учетные записи групп. Но все маркеры содержат одинаковые типы информации.

Дескрипторы безопасности и управление доступом

Структура данных для информации безопасности, связанной с объектом, которая определяет, кто и какие действия с объектом может выпол-

нять, называется дескриптором безопасности. В дескриптор безопасности входят следующие атрибуты:

– *Номер версии*. Версия модели безопасности SRM, использованная для создания дескриптора;

– *Флаги*. Дополнительные модификаторы, определяющие поведение или характеристики дескриптора;

– *SID владельца*. Идентификатор безопасности владельца;

– *SID группы*. Идентификатор безопасности основной группы объекта (используется только подсистемой POSIX);

– *Избирательный список управления доступом* (Discretionary Access Control List, DACL). Указывает, кто и какой доступ имеет к объекту.

– Системный список управления доступом (System Access Control List, SACL). Указывает, какие операции какими пользователями должны регистрироваться в журнале аудита безопасности и конкретный уровень целостности объекта.

Список управления доступом (ACL) состоит из заголовка и нескольких элементов управления доступом (ACE), которых может и не быть. Существует два типа ACL-списков: DACL и SACL. В DACL в каждом ACE-элементе содержится SID и маска доступа, которая обычно определяет права доступа (чтение, запись, удаление и т. д.), предоставленные или запрещенные держателю SID. Аккумулирование прав доступа, предоставленных отдельными ACE- элементами, формирует набор прав доступа, предоставленных ACL-списком. Если в дескрипторе безопасности отсутствует DACL (то есть имеется нулевой DACL), полный доступ к объекту предоставляется кому угодно. Если DACL пуст (то есть в нем нуль ACE-элементов), никто не имеет доступа к объекту.

Права доступа и привилегии

Для выполнения операций, связанных с безопасностью, различным учетным записям в Windows даются *привилегии* и *права доступа*.

Привилегии являются правом выполнять под той или иной учетной записью конкретную, связанную с системой операцию, например, выключение компьютера или изменение системного времени. *Право* учетной записи разрешает или запрещает той учетной записи, которой оно назначено, выполнять конкретный тип входа в систему, например, локальный или интерактивный.

Системный администратор назначает привилегии группам и учетным записям, используя такие инструментальные средства, как MMC-оснастка «Пользователи и группы Active Directory» (Active Directory Users and Groups) – для доменных учетных записей или редактор локальной политики безопасности (Local Security Policy Editor), находящийся в %System Root%\System32\secpol.msc. Доступ к этому редактору можно получить в папке Администрирование (Administrative Tools) Панели управления (Control Panel).

Права учетной записи не навязываются монитором безопасности и не хранятся в маркерах доступа. Функцией, отвечающей за вход в систему, является LsaLogonUser. Служба Winlogon, например, вызывает API-функцию LogonUser при интерактивном входе пользователя на компьютер, а функция LogonUser вызывает функцию LsaLogonUser. Функции LogonUser передается параметр, включающий тип выполняемого входа в систему, который может быть интерактивным, сетевым, пакетным, служебным и в качестве клиента терминального сервера.

В ответ на запросы входа в систему механизм проверки подлинности локальной системы безопасности (Local Security Authority, LSA) извлекает права учетной записи, назначенные пользователю, из базы данных политики LSA, сверяет тип входа в систему с правами учетной записи и отклоняет вход, если у учетной записи нет прав, разрешающих данный тип входа в систему.

Привилегии. В отличие от прав пользователя, которые задаются LSA в одном месте, различные привилегии определяются разными компонентами, которые их же и применяют. Например, привилегия отладки проверяется диспетчером процесса. API-функции, связанные с привилегиями, ничего не знают о существовании прав учетных записей, а вот API-функции, связанные с правами учетных записей, знают о наличии привилегий.

В отличие от прав учетных записей, привилегии могут быть включены или выключены. Чтобы проверка привилегии прошла успешно, она должна присутствовать в указанном маркере и быть включена.

Суперпривилегии. Пользователь, которому они назначены, фактически является «суперпользователем», имеющим полный контроль над компьютером. Эти привилегии могут использоваться различным способом для получения несанкционированного доступа к неограниченным в иной ситуации ресурсам и к выполнению несанкционированных операций. Например: *Проводить отладку программ* (Debug programs), *Приобретать права владения* (Take ownership), *Действовать в качестве части операционной системы* (Act as part of operating system) и др.

Использование повышенных привилегий за границы машины не выходит и на сеть не распространяется, поскольку любое взаимодействие с другим компьютером требует аутентификации в контроллере домена и проверки доменных паролей, которые не хранятся на компьютере ни в виде простого текста, ни в зашифрованном виде, а потому вредоносному коду не доступны.

Биометрическая среда для аутентификации пользователей

Windows предоставляет стандартизированный механизм для поддержки определенных типов биометрических устройств, в частности ска-

неров отпечатка пальца, чтобы обеспечивать идентификацию пользователей: Windows Biometric Framework (WBF).

Кроме особо оговоренных в следующем списке, все эти компоненты поставляются вместе с Windows.

– Биометрическая служба Windows (Windows Biometric Service, %SystemRoot%\System32\Wbiosrvc.dll). Предоставляет среду выполнения процесса, в которой может выполняться один или несколько поставщиков биометрических служб.

– Интерфейс биометрического драйвера Windows (Windows Biometric Driver Interface, WBDI). Это набор интерфейсных определений, которым должен соответствовать любой драйвер для устройства биометрического сканирования, чтобы быть совместимым с биометрической службой Windows.

– Биометрический API Windows (Windows Biometric API). Позволяет существующим компонентам Windows, например, WinLogon и LoginUI, получать доступ к биометрической службе. Приложения сторонних разработчиков имеют доступ к биометрическим API-функциям и могут использовать биометрический сканер для функций, отличающихся от входа в систему Windows. Биометрический API-интерфейс предоставляется библиотекой %SystemRoot%\System32\Winbio.dll.

– Поставщик биометрической службы отпечатков пальцев (Fingerprint Biometric Service Provider). Служит оболочкой для функций биометрических адаптеров того или иного типа с целью присутствия общего интерфейса, независимого от типа биометрии, с биометрической службой Windows Biometric Service.

– Технология Windows Hello, появившаяся в Windows 10, предоставляет новые средства аутентификации пользователей на основании биометрической информации: отпечаток пальца, лицо, сетчатка глаза.

Брандмауэр Windows Firewall

В отличие от сетевых брандмауэров, защищающих периметр, он работает на каждом компьютере под управлением ОС Windows и обеспечивает локальную защиту от сетевых атак, которые сумели обойти периметр сети или были инициированы изнутри. Он также обеспечивает безопасность на уровне отдельных соединений между компьютерами, что позволяет группе безопасности затребовать аутентификацию и защиту данных для всех сеансов связи.

Windows Firewall исследует и фильтрует весь трафик TCP/IP. Входящий трафик отбрасывается, если он не поступил в ответ на запрос со стороны данного компьютера (запрошенный трафик) или не был явно разрешен (не добавлен в список исключений).

Исходящий трафик по умолчанию тоже фильтруется, хотя многим распространенным приложениям доступ к сети разрешен. Группа безопас-

ности может определить, какой трафик следует добавить в список исключений, задавая имя приложения, имя службы, номер порта, сеть получателя, членство в домене и другие критерии. Все это достигается за счет настройки брандмауэра Windows Firewall.

Для разрешенного трафика Windows Firewall можно использовать, чтобы запросить или затребовать взаимную аутентификацию компьютеров перед началом сеанса связи, а также шифрование и обеспечение целостности данных на всем протяжении сеанса.

В современных ОС Windows брандмауэр Windows Firewall дополнен многими новыми возможностями, в том числе:

– интеграцией управления с протоколом IPSec;
– политиками IPSec на уровне приложения;
– поддержкой IPv6;
– фильтрацией исходящего трафика;
– повышением степени защищенности служб Windows, то есть ограничением списка действий, которые может выполнять служба, для снижения ущерба в случае компрометации системы;
– полной интеграцией с групповыми политиками;
– новыми критериями фильтрации;
– IP-аутентификацией (возможностью выполнять два раунда аутентификации с передачей дополнительных верительных грамот пользователя).

Брандмауэр Windows обеспечивает высокую степень защиты от червей, вирусов и других вредоносных программ, атакующих по сети. Однако поскольку брандмауэр Windows способен фильтровать весь обмен информацией по сети, он потенциально способен существенно нарушить работу нормальных программ.

Для предотвращения нарушения работы полезных программ с сохранением корпоративных требований в отношении фильтрации пакетов и аудита можно использовать одну из следующих категорий параметров настройки брандмауэра Windows: профили, исключения или ведение журнала брандмауэра.

Подсистема контроля целостности

В ОС Windows предусмотрена защита системных файлов WRP (Windows Resource Protection) для версий ОС, начиная с Vista, и WFP (Windows File Protection) для старых версий ОС. WRP предотвращает замену основных системных файлов, папок и разделов реестра, установленных как часть ОС. Windows Resource Protection работает, устанавливая дискретные списки доступа (DACLs) и ACL для защищаемых объектов. Владельцем таких объектов является субъект SYSTEM (Система), а разрешение на чтение-запись допускается лишь процессам, использующим

службу Установщик модулей Windows (Windows Modules Installer – TrustedInstaller.exe).

Ограничения WRP базируются на функциональных возможностях *Обязательного контроля целостности* (Mandatory Integrity Control, MIC), доступных начиная с Windows Vista. Когда пользователь входит в систему, ему присваивается идентификатор целостности (integrity SID). Он содержит метку целостности, которая определяет уровень доступа маркера пользователя (token), и, следовательно, указывает уровень полномочий, которых этот пользователь может достигнуть. Во время проверки прав доступа пользователю разрешается модифицировать или удалять объект, если присвоенный ему уровень целостности доминирует (то есть это число больше уровня целостности объекта или равно ему). В противном случае попытка доступа будет отклонена.

Как уже указывалось, MIC определяет пять уровней целостности: ненадежный (untrusted), низкий (low), средний (medium), высокий (high) и системный (system). Обычные пользователи имеют средний уровень, при повышении уровня полномочий до администратора они получают высокий. Настройки прав доступа для объектов, защищенных механизмом WRP, таковы, что доступ к ним имеет только сервис TrustedInstaller, а все остальные учетные записи, включая SYSTEM, имеют право только на чтение и исполнение. Поэтому даже администраторы не имеют прав полного доступа к системным файлам.

Система защиты ресурсов Windows (WRP) включает два *инструмента проверки целостности* защищенных системных файлов – SFC (System File Checker) и DISM (Deployment Image Servicing and Management), работающих в режиме командной строки, а также использующую DISM команду Repair-WindowsImage в Windows PowerShell.

Если есть основания полагать, что системные файлы ОС были повреждены или изменены, в командной строке, запущенной от имени администратора, следует выполнить sfc /scannow. Найденные ошибки целостности будут автоматически исправлены с выдачей подтверждающего сообщения. Также имеется возможность проверить целостность конкретного системного файла (sfc /scanfile=путь_к_файлу). SFC не может исправить ошибки целостности для активных системных файлов, которые используются в настоящий момент времени. В таком случае можно запустить SFC через командную строку в среде восстановления (в безопасном режиме). Существует и возможность запуска только проверки системных файлов без их исправления (sfc /verifyonly или /verifyfile).

WRP копирует файлы, необходимые для перезапуска Windows, в каталог кэша, расположенный по адресу %Windir%\winsxs\Backup. Размер каталога кэша и список кэшируемых файлов не может быть изменен. После установки ОС WFP копирует проверенные версии системных файлов (.dll,

.exe, .ocx и .sys и некоторые шрифты TrueType) в хранилище %systemroot%\system32\dllcache, местоположение и размер которого могут настраиваться. WFP будет добавлять файлы в хранилище до тех пор, пока его размер не достигнет заданного предела. Для восстановления защищенного файла, отсутствующего в хранилище, WFP обратится к источнику установки.

Восстановление системных файлов приводит к сбросу некоторых пользовательских настроек (например, замененных значков в Проводнике и пр.), изменения, заданные политиками, не сбрасываются.

Инструмент DISM позволяет выявить и исправить проблемы с хранилищем системных компонентов Windows (начиная с 8). Это может оказаться необходимым в случае, если SFC не может выполнить восстановление системных файлов, несмотря на найденные повреждения.

В командной строке, запущенной от имени администратора, следует выполнить:

– для получения информации о состоянии и наличии повреждений компонентов Windows: dism /Online /Cleanup-Image /CheckHealth;

– для проверки целостности и наличия повреждений хранилища компонентов: dism /Online /Cleanup-Image /ScanHealth;

– для проверки и автоматического восстановления системных файлов Windows: dism /Online /Cleanup-Image /RestoreHealth.

Результаты выполнения команд SFC и DISM протоколируются. Если восстановление системных файлов не удается выполнить в автоматическом режиме с помощью инструментов SFC и DISM, в файле журнала можно посмотреть, какие конкретно системные объекты были повреждены, а затем восстановить их вручную, скопировав из источника установки (предварительно потребуется сменить владельца файла/ папки и задать соответствующие разрешения доступа).

Еще одно средство проверки системных файлов – инструмент проверки подписи файла (File Signature Verification, Sigverif.exe). Все входящие в состав Windows драйверы и системные файлы (начиная с Vista) должны иметь цифровую подпись, свидетельствующую, что они протестированы в лаборатории WHQL (Windows Hardware Quality Labs) и могут быть использованы с текущей версией Windows. Это позволяет сохранить целостность и стабильность ОС и быстро найти поврежденные или измененные файлы.

В старых версиях ОС Windows (XP, 2000 и Server 2003) в зависимости от выбранных значений политики «Цифровая подпись драйверов устройств», ОС может либо игнорировать драйверы, не имеющие цифровой подписи, либо выводить предупреждение при их обнаружении, либо не допускать их установки. В более поздних версиях драйверы и сервисы, загрузка которых осуществляется загрузчиком ОС перед инициализацией ядра, должны иметь цифровую подпись. В 64-разрядных версиях Windows

для загрузки любого ПО, работающего на уровне ядра, наличие цифровой подписи обязательно.

Цифровые подписи драйверов, входящих в дистрибутив Windows, хранятся в файлах каталога (catalog files) в папке %SystemRoot%\System32\Catroot. Драйверы подключаемых устройств должны поставляться вместе с их собственными файлами каталога, которые Microsoft предоставляет фирме-разработчику после успешного тестирования на совместимость с Windows [7].

Верификация цифровой подписи файлов служит для поиска установленных на компьютере файлов, не имеющих цифровой подписи. Список подписанных и неподписанных драйверов, обнаруженных программой Sigverif, помещается в файл Sigverif.txt вместе с информацией о дате модификации, типе и номере его версии. Файл журнала может располагаться в каталоге Windows или \Users\Public\Documents\ (для Windows 10).

ОС семейства Windows обладают развитыми средствами архивации и восстановления, затрагивающие, в том числе, и состояние системы.

Серверные версии Windows включают встроенную систему архивации (Windows Server Backup), которая позволяет создавать образы системы и восстанавливать ОС сервера (состояние ОС и данные критических томов) или весь сервер, включая тома с данными, из выбранного архивного образа. При этом можно планировать расписание создания архивных копий и сохранять их локально на отдельном жестком диске или логическом томе, либо в сетевой папке. Создаваемый архивный образ может содержать: файлы и каталоги, тома файловой системы, приложения, состояние системы.

Компания Microsoft предлагает и более развитый инструмент System Center Data Protection Manager (DPM) для резервного копирования и восстановления данных в корпоративной среде. Данные резервных копий при этом могут храниться в пулах дисков, за пределами локальной сети в облаке Microsoft Azure (с использованием службы архивации Azure), на магнитной ленте стримера, которую можно хранить вне помещений организации. DPM позволяет осуществлять следующие виды резервного копирования:

– с поддержкой приложений Microsoft, включая SQL Server, Exchange и SharePoint;

– файлов, а также папок и томов для компьютеров под управлением серверных и клиентских версий ОС Windows;

– состояния системы либо исходного состояния системы для физических компьютеров под управлением серверных или клиентских версий ОС Windows;

– виртуальных машин Hyper-V под управлением Windows или Linux. Можно создать резервную копию всей виртуальной машины или резервные

копии с поддержкой приложений Майкрософт на виртуальных машинах Hyper-V под управлением Windows.

Резервное копирование состояния системы включает следующие компоненты.

– член домена: файлы загрузки, база данных регистрации классов COM+, реестр;

– контроллер домена: Active Directory (NTDS), файлы загрузки, база данных регистрации классов COM+, реестр, системный том (SYSVOL);

– службы кластера на компьютерах: дополнительно копируются метаданные кластерного сервера;

– службы сертификатов на компьютерах: дополнительно копируются данные сертификатов.

Резервирование исходного состояния системы создает резервные копии файлов ОС и всех файлов за исключением пользовательских данных на критических томах. Резервная копия исходного состояния системы по определению включает резервную копию состояния системы.

В десктопных версиях ОС Windows доступны следующие средства восстановления:

– восстановление системы (System restore, rstrui.exe) из точки восстановления;

– восстановление файлов из теневых копий;

– восстановление файлов пользователя из архивной копии, созданной с помощью инструмента архивации данных;

– восстановление ОС с помощью образа системы;

– возвращение ОС к первоначальным настройкам с помощью параметров восстановления, диска восстановления или установочного диска;

– удаление установленного обновления.

Восстановление системы – встроенный компонент ОС, который позволяет восстановить работоспособность ОС путем отката до состояния (системных файлов, ключей реестра, установленных приложений), записанного в определенный момент времени, – *точке восстановления* (restore point). Применение точки восстановления удалит все программы, которые были установлены после ее создания, и не восстановит удаленные раннее файлы. Данные пользователя восстановлением системы не затрагиваются.

Точки восстановления создаются автоматически при возникновении определенных системных событий, например, установка новых приложений или драйверов, а также установка критических обновлений. В случае выполнения операции восстановления также создается точка восстановления, что обеспечивает обратимость операций восстановления системы. Точки восстановления могут также создаваться периодически по заданному расписанию (которое настраивается через Планировщик заданий Task

Scheduler), или вручную. Для хранения точек восстановления выделяется определенная часть системного диска, размер которой можно настраивать. По мере заполнения предоставленного им дискового пространства более давние точки восстановления удаляются и на их место сохраняются новые.

Начиная с Windows 7 появилась возможность просмотреть, на какие приложения повлияет откат системы к выбранной точке восстановления. Поиск затрагиваемых программ позволяет определить, какие программы и драйверы придется в случае необходимости переустановить после отката.

Средство восстановления системы, начиная с Windows Vista, базируется на *теневом копировании* томов (Volume Shadow Copy). Теневые снимки тома представляют собой исходные копии тома, созданные на момент выполнения операции создания точки восстановления. Поэтому, если для логического раздела (даже несистемного) включена защита системы, доступно восстановления предыдущих версий файлов. Предыдущие версии файлов не создаются на дисках с файловой системой FAT и FAT32. Запуск восстановления файлов осуществляется командой Восстановить прежнюю версию (Restore previous versions) контекстного меню диска, папки или конкретного файла.

Центр архивации и восстановления Windows включает встроенные утилиты резервного копирования, позволяющие производить как архивирование пользовательский данных на выбранных томах, так и создать образ системы и диск аварийного восстановления.

Образ системы (system image) представляет собой «моментальный снимок» системного раздела, на который установлена ОС Windows, со всеми программами и файлами, присутствующими на нем. Кроме основного раздела Windows, можно включить в создаваемый образ и другие разделы. Утилита создания образа системы работает только с томами NTFS.

Диск восстановления можно использовать для загрузки среды восстановления Windows и получения доступа к набору диагностических и восстановительных утилит. Необходимость в нем может потребоваться в случае утраты или повреждения критических файлов ОС, когда загрузка системы невозможна. Диск аварийного восстановления позволяет восстановить ОС из точки восстановления системы или вернуть компьютер в исходное состояние (к параметрам по умолчанию). Те же варианты восстановления доступны при использовании загрузочного установочного диска.

Возврат ОС к первоначальным настройкам приводит к переустановке Windows, с возможностью сохранить или удалить личные файлы из профиля пользователя. Установленные приложения и драйверы будут удалены. Возврат компьютера в исходное состояние не требует предварительного создания диска восстановления или образа системы и может быть произведено через Панель управления Windows из раздела параметров восстановления системы.

В случае, если нестабильная работа ОС вызвана установкой обновления, недавно установленное обновление системы можно удалить через Центр обновления Windows, используя журнал просмотра обновлений.

Защита файловой системы

Главное отличие файловой системы NTFS по сравнению с традиционными файловыми системами – надежность. Во-первых, благодаря восстанавливаемости гарантируется, что структура NTFS-тома не будет повреждена и после системного сбоя доступ ко всем файлам сохранится. Во-вторых, несмотря на отсутствие гарантий сохранения пользовательских данных в случае системного сбоя, приложения могут использовать преимущества сквозной записи и сброса кэша для записи на диск изменений файлов через подходящие интервалы времени.

Как *сквозная запись кэша* (cache write-through), то есть немедленная запись данных на диск с их дублированием в кэше, так и *сброс кэша* (cache flushing), то есть принудительная запись на диск содержимого кэша – операции вполне эффективные. NTFS не требуется дополнительного ввода-вывода для сброса на диск изменений ряда структур файловой системы, так как изменения в этих структурах регистрируются в файле журнала в ходе единственной операции записи. Если в результате сбоя содержимое кэша оказывается утраченным, изменения файловой системы могут быть восстановлены при помощи журнала. Более того, в отличие от FAT, NTFS гарантирует, что сразу после сквозной записи или сброса кэша данные пользователей сохранят целостность и доступность даже при последующем сбое системы.

Новая файловая система ReFS, доступная начиная с Windows Server 2012 для дисковых пространств, а в Windows Server 2016 и Windows 10 – для любых внутренних дисков (кроме системных разделов), разработана на основе NTFS и предназначена для улучшенного контроля целостности данных. Она более устойчива к повреждению данных, лучше справляется с повышенной нагрузкой и легко масштабируется для очень больших файловых систем. Файловая система использует контрольные суммы для метаданных, а также может использовать контрольные суммы и для данных файла. Во время чтения или записи система проверяет контрольную сумму, чтобы убедиться в ее правильности. Таким образом, обнаружение искаженных данных осуществляется в режиме реального времени. Если файловая система обнаружит поврежденные данные, которые не имеют альтернативной копии для восстановления, то ReFS сразу удалить такие данные с диска. Это не потребует перезагрузки системы или отключения устройства хранения информации, как в случае с NTFS. Необходимость использования утилиты chkdsk полностью исчезает, так как файловая система автоматически корректируется сразу в момент возникновения ошибки.

Во время изменения метаданных ReFS создает новую копию данных и связывает данные с файлом, только после записи метаданных на диск. Это исключает возможность повреждения данных. Эта функция называется копированием на запись (copy-on-write), она присутствует и в других популярных ОС: Linux-системах: ZFS, BtrFS, а также файловой системе Apple APFS.

Функции криптографической защиты

Основным средством криптографической защиты файлов является шифрование EFS, доступное только на томах с ФС NTFS. Поддержка EFS встроена в NTFS-драйвер. При обнаружении зашифрованного файла NTFS выполняет встроенные EFS-функции, осуществляющие шифрование и расшифровывание данных по мере обращения к ним. NTFS отправляет запросы к EFS-службе через подсистему локальной аутентификации (Local Security Authority Subsystem, LSASS).

Работа с файлом для зашифровавшего его пользователя реализуется полностью прозрачно, другие же пользователи получат отказ в доступе при попытке открыть зашифрованный файл. Шифрование папки неспецифично и означает лишь, что любые помещенные в папку файлы будут шифроваться от лица текущего пользователя. Шифрование системных файлов недоступно.

При использовании PKI инфраструктуры существует возможность обеспечить совместный доступ к зашифрованному файлу для пользователей с известными сертификатами. Физическое удаление закрытого ключа приводит к невозможности получения доступа к зашифрованному файлу, даже если был получен доступ к аккаунту пользователя. Поэтому даже без использования смарт-карт или USB-токенов сотрудник, например, уезжая в отпуск, может экспортировать закрытый ключ на защищенный внешний носитель, одновременно удалив сертификат закрытого ключа из системы, сделав невозможным просмотр своих зашифрованных данных. После обратного импорта сертификата с закрытым ключом в систему доступ к зашифрованным файлам восстанавливается.

Доступ к данным, которые были зашифрованы пользователем, будет утрачен им и в случае сброса пароля пользователя. При плановой смене пароля (по заданному в групповых политиках сроку изменения пароля) или внеплановой смене с указанием предыдущего значения пароля, ключи шифрования обновляются (расшифровываются старым паролем и шифруются новым), и после смены пароля доступ к зашифрованным файлам не прекращается.

Доступ к зашифрованным EFS данным пропадает, если утрачена связь ОС с имеющимися физически на диске ключами. Существуют следующие типовые случаи, приводящие к потере доступа к зашифрованным файлам:

1. Система не загружается из-за смены или поломки комплектующих компьютера или нарушения работоспособности ОС (вышла из строя материнская плата, поврежден загрузочный сектор, испорчены системные файлы и т.п.).

2. Переустановка системы. Если имеется резервная копия системного диска, или профиля пользователя, доступ к зашифрованным данным может быть восстановлен с помощью специального программного обеспечения, но только в случае сохранности ключей.

3. Сброс пароля пользователя системным администратором или самим пользователем.

4. Удаление профиля пользователя.

5. Пользователь перенесен в другой домен (происходит его авторизация через другой сервер). Если при переносе сертификаты ключей хранились на сервере, то возможна потеря доступа к зашифрованным данным.

Существуют программные решения (например, утилиты компании Elcomsoft) позволяющие восстановить утраченный доступ к зашифрованным файлам в случае, если ключи не были физически удалены из системы или имеется сохраненный сертификат закрытого пользователя, с помощью которого было осуществлено шифрование (может потребоваться знание пароля пользователя для входа в ОС).

Выбор тех или иных криптографических алгоритмов в ОС Windows обеспечивается использованием низкоуровневых криптографических провайдеров CSP (Cryptography Service Provider) – поставщиков криптографических функций и прикладного интерфейса CNG API (Crypto API в старых версиях ОС). Архитектура криптографических интерфейсов ОС Windows представлена на рис. 3.1.

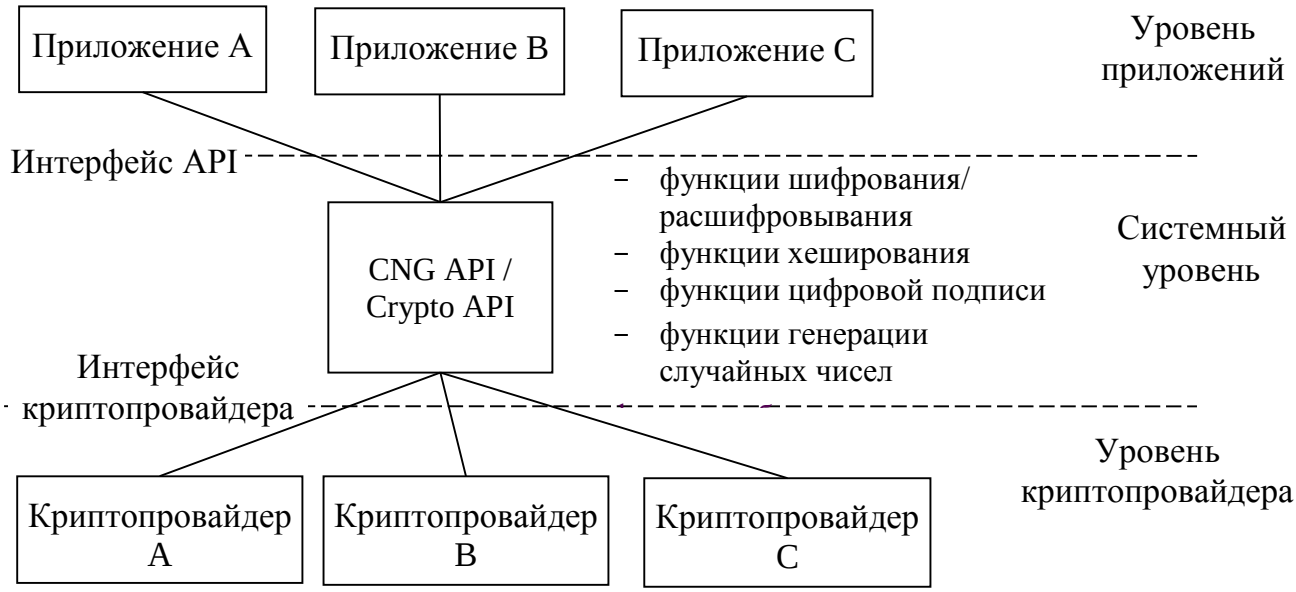

Рис. 3.1. Общая архитектура криптографических интерфейсов API

Начиная с Windows Vista, CNG API позволяет использовать не только встроенные, но и сторонние CSP, что допускает использование отечественной криптографии как при развертывании корпоративной PKI инфраструктуры с помощью службы Certification Authority, так и в EFS шифровании. Для этого в систему должен быть предварительно установлен криптопровайдер (например, КриптоПро CSP, VipNet CSP и т.п.). Криптопровайдеры де-факто стали стандартом СКЗИ, однако несовершенство предлагаемых ОС Windows механизмов расширения вынуждает разработчиков дополнительно модифицировать высокоуровневые криптобиблиотеки и приложения ОС Windows, что требует в некоторых случаях использования дополнительных утилит (например, КриптоПро IPSec, КриптоПро EFS и др.).

Таким образом, установив в ОС сертифицированный криптопровайдер отечественных производителей и сопутствующие утилиты, можно использовать стандартные средства ОС Windows для построения защищенных ИС с поддержкой российской криптографии.

Для защиты от автономных атак Microsoft предлагает использовать шифрование на уровне тома NTFS с помощью инструмента BitLocker (BitLocker Drive Encryption), доступного в версиях Windows Server, Windows 10 и в более ранних десктопных ОС версий Enterprise и Ultimate. Для съемных носителей (томов с файловой системой, отличной от NTFS) используется технология BitLocker To Go. BitLocker ориентирован, прежде всего, на совместное использование с аппаратным криптографическим сопроцессором – доверенным платформенным модулем TPM и глубоко привязан к конфигурации компьютера. Из-за импортных ограничений TPM на территории России недоступен, однако возможно использование шифрования BitLocker с ключом запуска, записанным на USB-накопитель.

BitLocker в стандартном режиме позволяет осуществлять полнодисковое шифрование тома NTFS (или ReFS), в том числе и системного тома с ОС Windows, а также осуществлять проверку целостности компонентов ранних стадий загрузки и конфигурационных загрузочных данных.

BitLocker автоматически шифрует все файлы, добавляемые на зашифрованный диск. Файлы будут зашифрованы только при хранении на зашифрованном диске, при их копировании на другой диск или компьютер они будут расшифрованы. Шифрование дисков BitLocker может использоваться совместно с шифрованием файловой системы EFS.

BitLocker оставляет незашифрованными:

– заголовок тома (совпадает с заголовком загрузочного сектора);

– поврежденные сектора, если они отмечены как нечитаемые;

– метаданные тома (зашифрованный том содержит три блока метаданных, для обеспечения избыточности).

BitLocker использует фильтрующий драйвер шифрования всего тома (Full-Volume Encryption, FVE), который автоматически видит все запросы к тому на ввод и вывод, зашифровывая блоки при записи и расшифровывая в процессе чтения. В отличие от EFS, функционирующей на уровне файловой системы, BitLocker работает на уровне подсистемы ввода-вывода, находящейся ниже NTFS. Поэтому NTFS «не знает» о включении BitLocker и работает с томом как обычно. Однако при чтении данных с тома при прямом физическом доступе к диску они окажутся зашифрованными. Это касается, в том числе, и системных файлов, если включено шифрование системного тома. Такая низкоуровневая реализация BitLocker не позволяет задавать для шифрования альтернативные криптографические алгоритмы, в том числе, российские.

3.2. Защита в ОС UNUX

Защита ОС семейства Unix в общем случае базируется на трех основных механизмах:

1. идентификации и аутентификация пользователя при входе в систему;
2. разграничении прав доступа к файловой системе, в основе которого лежит реализация дискреционной модели доступа;
3. аудит (регистрация событий).

При этом отметим, что для различных клонов ОС семейства Unix возможности механизмов защиты могут незначительно различаться

Аутентификация в UNIX

Список пользователей UNIX хранится в текстовом файле /etc/passwd, формат которого допускает его редактирование обычным текстовым редактором. В старых версиях UNIX зашифрованные образы паролей хранились в том же файле. В большинстве современных UNIX-систем образы паролей хранятся в отдельном файле, также расположенном в каталоге /etc. Имя этого файла различается в разных версиях UNIX, чаще всего применяется имя /etc/shadow. В некоторых UNIX-системах пароль каждого пользователя хранится в своем отдельном файле, например, в ALT Linux пароль пользователя username хранится в файле /etc/tcb/username/shadow.

Каждая строка файла /etc/passwd соответствует одному пользователю или псевдопользователю. Она включает в себя следующие семь полей, разделенных двоеточиями [8]:

– имя пользователя (login);
– хэш-образ пароля пользователя или признак того, что хеш-образ пароля хранится в другом файле, общем для всех пользователей (*) или признак того, что хэш-образ пароля хранится в отдельном файле (x);

– UID пользователя;

– GID первичной группы пользователя;

– комментарий к учетной записи пользователя (произвольная текстовая строка);

– домашняя директория пользователя;

– командный интерпретатор, который должен запускаться на терминалах, на которых работает данный пользователь (для псевдопользователей обычно указывается /dev/null).

Например, root:x:0:0:System Administrator:/root:/bin/bash.

Управление списком пользователей, как правило, осуществляется не путем ручного редактирования соответствующих файлов, а с использованием специальных утилит командной строки user add, userdel, usermod, groupadd, groupdel, groupmod, либо графических утилит, оснасток панели управления и т. п.

Некоторые загрузчики Linux позволяют установить пароль, запрашиваемый при загрузке системы. Так, при работе с LILO (Linux Loader) можно использовать параметры: «установить пароль для начальной загрузки» и «разрешение загрузки» после указания определенных опций в ответ на запрос LILO.

Хранение отпечатка пароля в файле /etc/passwd применялось в старых версиях UNIX систем, при этом оставалась возможность оффлайнового подбора пароля. Учитывая, что содержимое файла /etc/passwd имеет текстовый формат, а сам файл – общедоступен, такой способ хранения аутентификационной информации нельзя считать надежным. Поэтому для Linux были разработаны дополнительные механизмы защиты.

Механизм теневых паролей. Файл /etc/passwd больше не содержит отпечатка пароля, вместо него помещается значение, указывающее, что отпечаток пароля хранится в специальном парольном файле. Права на чтение парольного файла /etc/shadow принадлежат только суперпользователю. Хранимые в файле отпечатки пользовательских паролей получены с помощью функции crypt, допускающей как шифрование паролей с помощью любых методов, поддерживаемых криптографической библиотекой. Например, в Linux Debian/Ubuntu библиотека libcrypt поддерживает хэш, основанный на современном шифре Blowfish, или бесключевые хэш-функции с подмешиваннием случайной информации – соли, при этом используется многократное (порядка 1000 раз) применение алгоритма, что замедляет вычисление отпечатка, а значит и снижает вероятность успешного подбора паролей. Для реализации механизма теневых паролей в Linux используется набор программных средств Shadow Suite. В большинстве современных дистрибутивов Linux механизм теневых паролей задействован по умолчанию. Если же теневые пароли не используется, для хэширования паролей следует применять метод с использованием современных шифров.

Архитектура Pluggable Authentication Module (PAM) разработана Open Software Foundation (OSF) в качестве подсистемы аутентификации для современных UNIX/Linux-систем. Программные модули, входящие в подсистему аутентификации, делятся на две основные группы:

– клиенты (приложения и демоны), пользующиеся услугами PAM (login, passwd, rlogin, telnetd, ftpd и т.п.);

– программные модули, предоставляющие услуги PAM. Обычно эти модули представляют собой библиотеки, размещаемые в директории /lib/security.

С точки зрения клиента аутентификации его взаимодействие с PAM реализуется посредством так называемых примитивов – системных вызовов, передающих управление соответствующим модулям PAM. Поддерживаются шесть основных примитивов, сгруппированных в четыре подсистемы:

– pam_authenticate (подсистема auth) – аутентифицировать пользователя;

– pamjsetcred (подсистема auth) – авторизовать пользователя (установить UID, идентификаторы групп, квоты ресурсов и т.д.);

– pam_acct_mgmt (подсистема account) – проверить, доступна ли учетная запись пользователя для авторизации (не устарел ли пароль, не заблокирована ли учетная запись и т.п.);

– pam_openjsession (подсистема session) – начать сеанс работы пользователя с операционной системой;

– pam_closejsession (подсистема session) – завершить сеанс работы пользователя с операционной системой;

– pam_chauthtok (подсистема password) – назначить пользователю новые аутентификационные данные.

Вся техническая реализация процедуры аутентификации пользователя инкапсулирована внутри PAM, клиенту выдается лишь самая общая информация о результатах выполнения PAM того или иного примитива.

Каждый модуль PAM должен содержать функции-обработчики примитивов хотя бы одной подсистемы. Когда клиентская программа вызывает тот или иной примитив, PAM обращается к функциям- обработчикам данного примитива одного или нескольких своих модулей и принимает решение на основании возвращаемых ими результатов. Конкретный порядок того, какие функции каких модулей должны вызываться и как должны интерпретироваться результаты их вызова, определяется содержимым конфигурационных файлов PAM.

В ранних версиях PAM вся конфигурация PAM описывалась единственным файлом /etc/pam.conf. В более поздних версиях каждой клиентской программе сопоставляется индивидуальный файл конфигурации, расположенный в директории /etc/pam.d. Имя этого файла обычно совпадает с

именем клиента, так, например, файл конфигурации службы SSH обычно имеет имя /etc/pam.d/sshd. В каждом файле конфигурации содержится последовательность строк, определяющих, какие PAM-модули должны использоваться клиентом и каким образом это должно осуществляться. Конфигурация PAM, используемая клиентскими программами по умолчанию, описывается в файле /etc/pam.d/other.

Если, например, в некоторой конфигурации ОС необходимо полностью запретить выполнение команды su, достаточно вписать в начало файла /etc/pam.d/su строку: auth requisite pam_deny.so.

Модули PAM позволяют реализовать в ОС также аутентификацию с использованием внешних электронных носителей информации (например, pam_usb.so), а также биометрическую аутентификацию (например, pamJxprint.so). С помощью модуля pamJistfile.so можно реализовать в одной ОС различные схемы аутентификации для конкретных списков пользователей, групп пользователей, хостов и т.п., например, следующая строка конфигурационного файла:

auth sufficient pamJistfile.so item=user sense=deny file=/etc/users

deny запрещает вход в систему пользователям, имена которых перечислены в файле /etc/user.deny.

Таким образом, архитектура PAM позволяет администратору UNIX-системы гибко настраивать подсистему аутентификации в соответствии с реальными потребностями конкретной вычислительной сети и конкретного экземпляра ОС. Все, что нужно сделать администратору, – установить в системе необходимые модули PAM и обеспечить корректное взаимодействие между ними. В результате подсистема аутентификации может быть легко адаптирована к самым разным технологиям аутентификации, принятым в конкретной организации.

Сетевая защита

Сетевой защите в клонах ОС UNIX последних версий уделяется значительное внимание. Так, в ОС Linux для обеспечения сетевой безопасности применяется несколько эффективных средств:

– защищенная оболочка ssh для предотвращения атак, в которых для получения паролей используются анализаторы протоколов;

– программы tcp_wrappers для ограничения доступа к различным службам компьютера;

– межсетевой экран netfilter для ядер Linux, начиная с версии 2.4;

– сетевые сканеры для выявления уязвимых мест компьютера;

– демон tcpd для обнаружения попыток сканирования портов со стороны злоумышленников (в дополнение к этому средству полезно регулярно просматривать файлы системного журнала);

– система шифрования PGP (Pretty Good Privacy);

– программа stelnet (защищенная версия хорошо известной программы telnet);

– программа qmail (защищенная доставка электронной почты);

– программа ipfwadm для настройки межсетевых экранов (firewall);

– режим проверки паролей входных соединений для систем, разрешающих подключение по внешним коммутируемым линиям связи или локальной сети.

Управление доступом к данным

В ОС семейства UNIX к *объектам доступа* относятся файлы, директории, ссылки (links), устройства и именованные каналы (named pipes). В большинстве современных версий UNIX выполняющиеся процессы также рассматриваются как объекты доступа, размещенные в специальной директории /proc. Атрибуты защиты этих объектов могут учитываться при доступе отладчика к адресному пространству процесса, при отправке процессу сигналов и т. п.

К *субъектам доступа* в UNIX относятся пользователи (включая псевдопользователей) и группы пользователей. В ранних версиях UNIX действовало ограничение, заключающееся в том, что каждый пользователь операционной системы мог входить в одну и только одну группу, т. е. весь список пользователей операционной системы разбивался на некоторое количество непересекающихся групп. В современных UNIX-системах данное ограничение обходится теми или иными способами.

Среди пользователей UNIX выделяется предопределенный пользователь **root**, обладающий в ОС абсолютными полномочиями и имеющий возможность игнорировать любые принятые в системе правила управления доступом.

Пользователи и группы пользователей идентифицируются в UNIX числовыми идентификаторами, уникальными в пределах одного экземпляра ОС. Пользователь root всегда имеет идентификатор 0. Идентификаторы пользователей обозначаются стандартным обозначением UID (User ID), идентификаторы групп – GID (Group ID).

Все методы доступа ко всем объектам регулируются в UNIX тремя правами доступа:

– Read (R) – чтение;

– Write (W) – запись;

– Execute (X) – выполнение.

Для файлов смысл всех трех прав доступа вполне очевиден. Для каталогов право Read дает возможность получения списка вложенных файлов и подкаталогов, право Write – возможность создания, переименования и/или удаления вложенных файлов и подкаталогов, право Execute – переход в каталог (команда cd). Для устройств каждое право доступа дает возможность обраще-

ния к соответствующей функции драйвера, обслуживающего данное устройство. Для именованного канала право Read требуется для получения информации, право Write – для отправки, право Execute не определено.

В настоящее время стандартная для UNIX система управления доступом на основе векторов фиксированной длины оценивается большинством экспертов как устаревшая. Во многих современных версиях UNIX в дополнение к векторам доступа поддерживаются списки доступа переменной длины. Как правило, в реализации списков доступа в UNIX заметно влияние особенностей реализации аналогичного механизма в Windows. Так, например, в Mac OS типы, стандартный порядок расположения и флаги наследования элементов управления доступом точно соответствуют аналогичным элементам подсистемы управления доступом Windows. Кроме того, в Mac OS стандартные права read, write и execute дополнены следующими правами доступа:

– delete – удалить объект;

– append – дописать информацию в файл или создать поддиректорию в директории;

– read_attributes – получить атрибуты объекта;

– write.attributes – изменить атрибуты объекта;

– read.extended – получить расширенные атрибуты объекта;

– write.extended – изменить расширенные атрибуты объекта;

– read_permissions – получить список доступа объекта;

– write_permissions – изменить список доступа объекта;

– take_ownership – овладеть объектом (назначить себя новым владельцем объекта).

Данный набор прав доступа почти полностью совпадает с набором стандартных и специфичных прав доступа, определенных для файловых объектов Windows. Это сделано для обеспечения совместимости с Active Directory Windows. Mac OS, подобно Windows, поддерживает разрешающие и запрещающие элементы управления доступом, а также поддерживается назначение атрибутов защиты создаваемым объектам путем наследования.

Понятие привилегии субъекта доступа в ранних версиях UNIX систем отсутствовало. В наиболее распространенной на сегодняшний день ее реализации, применяемой, в частности, в Linux, привилегии называются *возможностями (capabilities*). Так, в ядре Linux 2.6.34 определены, например, следующие возможности субъектов доступа:

– CAP.CHOWN – позволяет модифицировать идентификаторы владельца или группы владельца любого объекта;

– CAP.DAC.OVERRIDE – позволяет игнорировать любые ограничения на доступ к объектам, задаваемые списками управления доступом;

– CAP.SETPCAP – позволяет включать или выключать любую привилегию у любого процесса;

– CAP_NET_BROADCAST – позволяет отправлять широковещательные сетевые пакеты;

– CAP_SYS_MODULE – позволяет загружать и выгружать модули расширения ядра ОС;

– CAP_SYS_RAWIO – позволяет выполнять низкоуровневые операции ввода-вывода (ioperm, iopl), взаимодействовать с USB- устройствами посредством виртуального файла /proc/bus/usb;

– CAP_SYS_BOOT – позволяет перезагружать ОС;

– CAP_SYS_TIME – позволяет управлять системными таймерами;

– CAP_SYS_TTY_CONFIG – позволяет управлять консольными терминалами.

Каждому процессу UNIX присваиваются четыре числовых идентификатора:

– UID – идентификатор пользователя, породившего данный процесс;

– GID – идентификатор группы пользователя, породившего данный процесс;

– EUID – эффективный идентификатор пользователя;

– EGID – эффективный идентификатор группы пользователя. Обычно EUID совпадает с UID, а EGID – с GID.

Каждый объект UNIX может иметь атрибуты защиты, включающие в себя следующие три элемента:

– идентификатор (UID) владельца объекта;

– идентификатор группы (GID) владельца объекта;

– вектор доступа.

Вектор доступа включат в себя следующие элементы:

– является ли объект каталогом – 1 бит (D);

– бит SUID;

– бит SGID;

– бит sticky;

– права доступа владельца – 3 бита (RWX);

– права доступа пользователей, входящих в группу владельца – 3 бита (RWX);

– права доступа всех остальных пользователей – 3 бита (RWX).

Проверка прав доступа субъекта к объекту осуществляется по следующему алгоритму.

1. Если EUID процесса равен нулю (т.е. процесс выполняется от имени суперпользователя), доступ к объекту предоставляется без каких бы то ни было дополнительных проверок.

2. Если EUID процесса совпадает с UID владельца объекта, права доступа, запрошенные процессом, сравниваются с правами, разрешенными владельцу данного объекта. Если все права, запрошенные процессом, разрешены владельцу объекта, доступ разрешен, в противном случае – запрещен.

3. Если EUID процесса отличается от UID владельца объекта, но EGID процесса совпадает с GID владельца объекта, права доступа, запрошенные процессом, сравниваются с правами, разрешенными пользователям, входящим в группу владельца объекта. Если все права, запрошенные процессом, разрешены группе владельца объекта, доступ разрешен, в противном случае – запрещен

4. Если EUID процесса отличается от UID владельца объекта и EGID процесса отличается от GID владельца объекта, права доступа, запрошенные процессом, сравниваются с правами, разрешенными пользователям, не входящим в группу владельца объекта (всем остальным пользователям). Если все права, запрошенные процессом, разрешены пользователям, не входящим в группу владельца объекта, доступ разрешен, в противном случае – запрещен.

Бит sticky в современных UNIX-системах применяется только в отношении директорий. Если в векторе доступа директории установлен данный бит, удаление файлов и поддиректорий разрешается только их владельцам.

Наличие в UNIX суперпользователя root с ничем не ограниченными полномочиями создает серьезные проблемы не только для безопасности, но и для надежности ОС. Любая ошибка, допущенная суперпользователем при работе с командной строкой, может стать фатальной. Если, например, в команде rm -rf /*.bak ошибочно поставить пробел между звездочкой и точкой, сама ОС и все хранящиеся в ней данные будут немедленно и необратимо уничтожены. Большинство администраторов UNIX в настоящее время избегают авторизоваться в ОС в качестве суперпользователя на протяжении всего сеанса работы с системой. Если администратору необходимо выполнить какие-то действия, требующие повышенных полномочий, он может воспользоваться утилитой командной строки su. Эта утилита получает имя и пароль пользователя (на практике чаще всего используется имя root, его можно не вводить, считается, что оно задано по умолчанию), и, если введены корректные данные, инициирует сеанс работы указанного пользователя на текущем терминале, запуская новую копию командного интерпретатора. Прекращение сеанса работы пользователя, инициированного командой su, происходит естественным образом при завершении командного интерпретатора, например, командой exit.

Во многих UNIX-системах пользоваться утилитой su по умолчанию (если администратор не менял настройки аутентификации для программы su) разрешено только пользователям, входящим в группу wheel. Иногда эту группу неформально сравнивают с группой администраторов Windows, а использование утилиты su – с запуском программы Windows на повышенном уровне мандатной целостности.

На практике вместо su чаще применяется утилита sudo, реализующая выполнение от имени суперпользователя (или другого пользователя, если

указан флаг -u) команды, являющейся параметром команды sudo. Например, команда sudo mc запускает от имени суперпользователя файловый менеджер Midnight Commander.

При первом вводе sudo пользователь должен повторно ввести свой пароль. После этого в течение некоторого времени (по умолчанию пять минут) пользователь может использовать команду sudo без пароля. Порядок использования команды sudo описывается конфигурационным файлом /etc/sudoers. Пример: суперпользователь root может выполнять через sudo любые команды, не вводя никаких паролей:

root ALL = NOPASSWD: ALL

Аудит в UNIX

В современных ОС семейства UNIX реализуются два принципиально различных подхода к организации аудита. Первый из них основан на применении традиционных для UNIX демонов регистрации событий syslogd и klogd. Второй основан на переносе в среду UNIX архитектуры и интерфейсов подсистемы аудита, изначально свойственных ОС семейства Windows. Если UNIX-система включена в состав гетерогенной сети, большинство узлов которой работают под управлением Windows, единообразное построение подсистем аудита всех ОС становится серьезным преимуществом, поскольку позволяет, например, распространять политики аудита на большие подмножества узлов сети независимо от того, под управлением какой ОС работает каждый конкретный узел того или иного подмножества. В состав большинства современных UNIX-систем могут включаться обе подсистемы аудита, при этом они могут работать как независимо одна от другой, так и во взаимодействии [8].

syslogd, klogd. Демон syslogd предоставляет услуги прикладным и системным программам, демон klogd реализует аудит операций, выполняемых ядром ОС. Основным назначением этих демонов является поддержка единой в масштабах ОС политики аудита, предписывающей выполнение определенных действий для каждого типа регистрируемых событий. Каждый раз, когда прикладная или системная программы выполняет потенциально аудируемое действие, информация об этом действии передается демону syslogd, тот сверяется с текущей политикой аудита и принимает решение о порядке регистрации данного события. Получение информации демоном syslogd обычно реализуется через сокет /dev/log (для локальных клиентов) или UDP-порт 514 (для удаленных клиентов). Демон klogd получает информацию через специальный файл /proc/kmsg или системный вызов ядра sys_syslog. Идентификаторы процессов демонов аудита хранятся в десятичном виде в текстовых файлах /var/run/syslogd.pid и /var/run/klogd.pid соответственно.

Действующая политика аудита описывается текстовым файлом /etc/syslog.conf. Каждая строка этого файла описывает одно элементарное правило политики аудита и включает в себя два основных поля:

– *селектор* – описывает условия применимости данного правила;

– *действие* – описывает действие, которое должно быть выполнено в результате применения данного правила.

Поле селектора в общем случае имеет следующий вид:

<источник>. <модификатор> <приоритет>

Подполе (источник) описывает подсистему ОС, которая обращается к демону syslogd с намерением зарегистрировать то или иное событие. Данное подполе может принимать следующие значения:

– auth – клиентская часть подсистемы аутентификации;

– authpriv – серверная часть подсистемы аутентификации;

– cron – демон автоматического запуска заданных процессов в заданное время;

– daemon – все другие демоны, кроме явно перечисленных в данном списке;

– ftp – FTP-сервер;

– kern – ядро ОС;

– lpr – сервер печати;

– mail – сервер электронной почты;

– mark – предназначено для внутреннего использования;

– news – NNTP-сервер;

– security – синоним для auth, считается устаревшим;

– syslog – сам демон аудита;

– user – любая прикладная программа;

– uucp – демон UUCP;

– 1oca10-1oca17 – зарезервированы;

– * – любой источник.

В одном селекторе можно указывать несколько источников через запятую.

Подполе (модификатор) может принимать следующие значения:

– (не заполнено) – данная строка описывает реакцию на события, имеющие приоритет, больший или равный указанному;

– = – данная строка описывает реакцию на события, имеющие приоритет, точно равный указанному;

– ! – данная строка описывает реакцию на события, имеющие приоритет, меньший указанного;

– != – данная строка описывает реакцию на события, имеющие приоритет, отличный от указанного.

Подполе (приоритет) может принимать следующие значения (перечислены в порядке убывания значимости):

– emerg, panic – ОС вышла из строя, после регистрации данного события произойдет аварийное завершение ее работы;

– alert – критический сбой, требующий немедленной реакции;
– crit – критический сбой;
– err, error – сбой;
– warning, warn – предупреждение;
– notice – важное информационное сообщение;
– info – информационное сообщение;
– debug – отладочное сообщение.

Также поддерживаются два специальных значения приоритета:

– * – любой приоритет;
– none – «никакой» приоритет, предназначено для описания отношений типа «все, кроме».

В одной строке файла syslogd.conf можно описывать несколько селекторов, они разделяются точкой с запятой (;).

Поле действия описывает конкретное действие, которое должен предпринять демон syslogd при регистрации события аудита, соответствующего указанному селектору. Поддерживаются следующие типы действий:

– запись информации о событии в файл – указывается имя файла;
– выдача информации о событии на терминал или консоль – указывается имя файла в директории /dev, соответствующее данному терминалу и консоли;
– запись информации о событии в указанный именованный канал – указывается имя канала, которому предшествует символ «|»;
– перенаправление информации о событии демону syslogd другого компьютера (514 порт LTDP) – указывается имя компьютера, которому предшествует символ «@»;
– интерактивное оповещение о событии одного или нескольких пользователей – указываются имена пользователя через запятую;
– интерактивное оповещение о событии всех пользователей, работающих в данный момент с операционной системой – указывается одиночный символ «*».

Пример описания политики аудита в файле syslog.conf:

```
# Все критические сообщения, кроме исходящих от ядра,
# записывать в файл /var/adm/critical
*.crit;kern.none /var/adm/critical
```

В большинстве UNIX-систем большая часть сообщений аудита традиционно записывается в файл /var/log/messages или /var/adm/log/messages. Для хранения сообщений о попытках аутентификации традиционно используется файл /var/log/auth.log или /var/adm/loginlog.

При обновлении файла syslog.conf новая политика аудита вступает в силу только после перезагрузки демона syslogd или после того, как этот демон получит сигнал SIGHUP. Прикладные программы UNIX интерпретируют данный сигнал как обрыв связи удаленного клиента с терминалом и

обычно, получив его, аварийно завершают работу, но демон syslogd воспринимает сигнал SIGHUP иначе – как требование обновить политику аудита.

Журналы аудита UNIX представляют собой обычные текстовые файлы. Их просмотр и анализ может проводиться как обычными утилитами просмотра текстов, так и более продвинутыми программными средствами, облегчающими работу с аудитом.

Удаление из файлов аудита устаревшей информации традиционно реализуется путем регулярного запуска демоном cron специальной утилиты logrotate, конфигурация которой описывается файлом /etc/logrotate.conf.

Серьезным недостатком подсистемы аудита UNIX является то, что доступ к демону syslogd в общем случае предоставляется любым программам, как системным, так и прикладным. При этом клиентская программа не только передает демону текст сообщения, но и самостоятельно указывает источник и приоритет регистрируемого события. Более того, существует специальная утилита командной строки logger, позволяющая обычному непривилегированному пользователю передать демону syslogd произвольную информацию от имени произвольной подсистемы ОС и присвоить этой информации произвольный уровень значимости, например, имитировать нарушение правил безопасности другим пользователем.

Auditd. Механизм регистрации событий, реализуемый демонами syslogd и klogd, в основном предназначен не столько для поддержания безопасности системы, сколько для выявления программных и аппаратных неисправностей, тестирования и отладки новых компонент ОС и т.п. Применять данный механизм для регистрации событий, связанных с безопасностью системы, не очень удобно, и в некоторых ОС, принадлежащих к семейству UNIX, для этой задачи предназначен отдельный, дополнительный демон аудита, которому чаще всего назначается имя auditd. Концептуально реализации данного демона в разных UNIX-системах обычно очень похожи на реализацию подсистемы аудита в Windows. Однако технические детали разных реализаций могут очень сильно отличаться одна от другой.

В операционной системе IBM AIX демон auditd предназначается главным образом для регистрации обращений пользователей к файлам. Конфигурация демона описывается в файлах config и objects, размещаемых в директории /etc/security/audit. Данные аудита записываются либо в псевдофайл /audit/trail, либо на логическое устройство /dev/audit, для считывания данных аудита уполномоченным пользователем используются специальные утилиты auditptr и auditstream.

В Mac OS X демон audit записывает регистрируемые данные в один или несколько файлов, обычно расположенных в директории /var/audit. Политика аудита для данного демона описывается конфигурационными файлами audit_class, audit_event, audit_control и audit _user. Для каждого польз-

ователя ОС порядок регистрации событий, связанных с деятельностью этого пользователя, задается индивидуально, кроме того, определен порядок регистрации событий по умолчанию. Механизм регистрации событий, реализуемый демоном audit в Mac OS, во многом похож на реализацию аудита в Windows. Аналогично категориям событий аудита в Windows, в Mac OS события аудита объединяются в так называемые классы, при этом порядок регистрации событий отдельно определяется для успешных и неуспешных событий каждого класса. Для каждого пользователя определяются так называемые флаги аудита – битовая маска, в которой каждый бит соответствует одному классу потенциально регистрируемых событий.

Для некоторых событий возможно интерактивное оповещение администратора с помощью автоматического запуска специального скрипта audit.warn, который в качестве параметра получает текстовую строку, содержащую описание зарегистрированного события. По умолчанию скрипт audit.warn просто записывает текущее время и переданный ему текст в конец файла /etc/security/audit_messages, но администратор ОС может менять код скрипта произвольным образом.

Похожим образом реализован дополнительный аудит в операционной системе Astra Linux. Здесь для каждого пользователя или группы пользователей может быть создан так называемый профиль аудита, описывающий порядок регистрации событий аудита для следующих категорий:

open – открытие объекта доступа;
create – создание объекта доступа;
exec – запуск процесса;
delete – удаление объекта доступа;
chmod – изменение вектора доступа объекта;
chown – изменение владельца объекта доступа;
mount – монтирование или размонтирование файловой системы;
module – загрузка или выгрузка модуля расширения ядра ОС;
uid – запуск процесса с использованием механизма SUID;
gid – запуск процесса с использованием механизма SGID;
audit – изменение профиля аудита;
xattr – изменение списка доступа объекта;
mac – изменение мандатных меток объектов доступа;
cap – изменение привилегий субъекта доступа;
chroot – смена корневого каталога файловой системы;
rename – переименование объекта доступа;
net – изменение сетевых настроек.

Большинство перечисленных категорий фактически представляют собой системные вызовы, при каждом выполнении которых регистрируется, либо не регистрируется соответствующее событие аудита.

Подсистема контроля целостности

Развитый механизм контроля целостности системных файлов ОС Linux – IMA (Integrity Measurement Architecture) [13], доступен для ядра версии 2.6.30 и выше. Технология IMA включена не во все дистрибутивы, поэтому для ее использования может потребоваться пересборка ядра. Однако в дистрибутивах с поддержкой IMA (например, Linux Ubuntu, начиная с версии 14.04) включение IMA требует лишь перепуска с указанием соответствующей политики в командной строке загрузки. По умолчанию технология IMA отключена, так как требует значительных накладных расходов, в частности, существенно увеличивает время загрузки системы.

Подсистема IMA отвечает за вычисление хэш-значений файлов и программ при их открытии до того, как их содержимое будет доступно для чтения или выполнения, а также поддерживает проверку их соответствия заранее заданному списку и создание отчетов.

Когда система впервые устанавливается в известном доверенном состоянии, записывается список «хороших» значений хеш-функции. Во время обычной работы системы IMA поддерживает список текущих файловых хэшей. Затем этот список можно сравнить с первоначально записанным списком «хороших» хэшей, чтобы обнаружить любые необоснованные изменения. Технология IMA изначально использовалась только для дальнейшей аттестации системы, то есть была пассивным средством защиты информации.

Начиная с версии ядра 3.7 технология IMA дополнена функцией оценки и компонентом EVM (Extended Verification Module), позволяющем блокировать загрузку бинарных данных в случае обнаружения нарушений их целостности. В настоящее время IMA/EVM предоставляют две основные функциональные возможности:

– *измерение* (хэширование) содержимого файла при обращении к нему и отслеживание этой информации в журнале аудита;

– *оценка* файлов, которая позволяет запретить доступ к файлу, если результат его измерения или цифровая подпись не соответствуют ожидаемому значению.

Функция измерения IMA используется только для обнаружения потенциально вредоносных изменений в системе, но не для их предотвращения. Функция оценки позволяет предотвратить выполнение или даже чтение ненадежных файлов. Это достигается путем подписания хэша содержимого файла и дальнейшей проверки подписи с использованием криптографии с открытым ключом (RSA).

Использование цифровых подписей подходит только для файлов, которые не должны изменяться во время выполнения, потому что в противном случае подпись необходимо будет пересчитать с использованием закрытого ключа, который не должен храниться в системе по соображениям

безопасности. Поэтому подписи, как правило, используют для двоичных файлов, а не для файлов конфигурации.

Если IMA использует только значение бесключевой хэш-функции, защита этого значения производится с помощью EVM. IMA/ EVM хранят хэш-значения или подписи файлов в расширенных атрибутах. Расширенный атрибут IMA хранит данные о фактическом содержимом файла. Расширенный атрибут EVM хранит данные о метаданных файла, такие как владение, биты режима, другие расширенные атрибуты и т. д. EVM создает подписанное хэш-значение или HMAC для указанных метаданных файлов. Использование EVM подписей целесообразно только для редко изменяемых файлов. Расчет HMAC требует использования ключа симметричного шифрования, который может вычисляться на основе вводимого пользователем пароля или храниться аппаратно (например, в доверенном платформенном модуле TPM).

EVM работает только в сочетании с инфраструктурами безопасности, то есть проверка метаданных EVM выполняется только тогда, когда выполняется оценка IMA. Если проверка метаданных не удалась, доступ к рассматриваемому файлу будет отклонен.

IMA/ EVM могут работать на платформах без аппаратного TPM, хотя и без аппаратной гарантии обнаружения компрометации. Преимущество привязки списка эталонных хэш-значений IMA или ключей к TPM состоит в том, что список измерений не может быть скомпрометирован какой-либо программной атакой без возможности обнаружения. Поэтому в доверенной загрузочной системе измерения IMA могут использоваться для подтверждения целостности системы во время выполнения. Оценка IMA при использовании совместно с TPM расширяет концепцию «безопасной загрузки» на запуск файлов доверенной ОС (рис. 2.6).

В Linux системах возможна проверка целостности установленных в системе пакетов. Это позволяет убедиться в том, что файлы, установленные из пакета, не были изменены после инсталляции. В Red Hat Linux и других дистрибутивах, использующих для установки пакетов RPM (RPM Package Manager) для этих целей используется опция --verify (-V) одноименной команды: rpm --verify --all (или rpm -Va). Можно выполнить также проверку файлов отдельного пакета или пакета, которому принадлежит указанный файл. При проверке пакета информация об установленных из пакета файлах сравнивается с информацией из оригинального пакета. Помимо прочего, при такой проверке проверяется размер, сумма MD5, разрешения, тип, владелец и группа владельца каждого файла. Результатом выполнения команды будет список кодов найденных ошибок (табл. 3.2) с указанием полного пути файла.

Таблица 3.2. Коды ошибок проверки целостности rpm пакетов

Код ошибки	Обнаружено изменение
S	размера файла
M	атрибутов доступа или типа файла
5	контрольной суммы MD5 файла
D	старшего и младшего номера файла устройства
L	пути ссылки
U	владельца файла
G	группы файла
T	времени создания или последней модификации файла
?	файл не удалось прочитать

Помимо этих флагов, через два пробела указывается тип файла: c – файл конфигурации, g – файл, изначально отсутствующий в пакете, d – файл документации, l – файл лицензии, r – readme файл.

Пакет, как правило, подписывается разработчиком и содержит контрольные суммы устанавливаемых файлов (в .deb пакетах подписываются файлы, содержащие список контрольных сумм). Для проверки целостности файла RPM-пакета используется опция --checksig (или -K) команды rpm. Для проверки цифровой подписи пакетов требуется открытый ключ. Открытые ключи для проверки подписи, как правило, загружаются автоматически при установке из дистрибутива.

В Linux Debian/ Ubuntu и подобных дистрибутивах с dpkg используется утилита debsums, которая должна быть предварительно установлена на компьютер. Debsums сравнивает хэш-коды md5 файлов с контрольными значениями из каталога /var/lib/dpkg/info. Опции debsums приведены в табл. 3.3.

Таблица 3.3. Основные опции утилиты debsums

Опция	Выполняемое действие
-a, --all	Проводить проверку также файлов конфигурации пакета (обычно не выполняется)
-e, --config	Проверять только файлы конфигурации
-c, --changed	Выводить только измененные файлы
-s, --silent	Выводить только ошибки
-l, --list-missing	Показать список пакетов (или архивов .deb), которые не содержат файла контрольных сумм md5
-r, --root=каталог	Задать проверяемый каталог (по умолчанию /)

Например, проверка целостности всех установленных файлов с выводом лишь измененных может быть произведена командой debsums --changed --all (debsums -ca).

Обе утилиты используют в качестве эталонов внутреннюю базу соответствующего установщика пакетов (rpm или dpkg), хранящуюся на компьютере. Поскольку в качестве контрольных сумм, как правило, используются значения бесключевых хэш-функций, эталоны, вообще говоря, могут быть изменены наряду с самими файлами. Однако наличие цифровой подписи пакета позволяет проверить неизменность эталонной информации.

При обнаружении изменений, пакеты следует переустановить, например, в Debian/ Ubuntu это можно сделать командой

```
apt-get install --reinstall $(dpkg -S $(debsums -c) | cut -d : -f 1 | sort -u)
```

Однако такой метод не подходит для системных файлов.

Контроль целостности избранных файлов, в том числе и системных, может производиться с помощью сторонних решений, которые доступны в большинстве дистрибутивов Linux.

Monit (https://mmonit.com/monit/) – небольшая утилита для управления и мониторинга процессов, программ, файлов, каталогов и файловых систем в *nux системах. Доступна для большинства Linux дистрибутивов, FreeBSD и OpenBSD, MacOS и Solaris, позволяет работать как в режиме командной строки, так и через веб-интерфейс. Monit записывает события в syslog или свой собственный файл журнала, а также может отправлять уведомления в случае обнаружения изменения контрольной суммы файла, либо добавления новых объектов файловой системы. Monit может выполнять автоматическое восстановление и проводить другие необходимые действия в случае возникновения ошибок.

AIDE (Advanced Intrusion Detection Environment) – утилита проверки целостности файлов (https://aide.github.io/). AIDE создает базу данных контрольных значений файлов в системе, а затем использует эту базу данных в качестве критерия для обеспечения целостности файлов и обнаружения системных вторжений. Для расчета контрольных значений могут использоваться контрольные суммы CRC32 или хэш-функции. Все обычные атрибуты файла (тип файла, разрешения, индексный дескриптор, Uid, Gid, имя ссылки, размер, количество блоков, количество ссылок, атрибуты времени) также можно проверить на несоответствия. Поддерживаются атрибуты файловой системы Posix ACL, SELinux, XAttrs и Extended. Позволяет использовать регулярные выражения для выборочного включения или исключения файлов и каталогов для мониторинга. AIDE доступен для большинства дистрибутивов Linux, FreeBSD и может работать на любой современной Unix-системе.

Tripwire (https://github.com/Tripwire/tripwire-open-source) – инструмент проверки целостности, который позволяет обнаруживать изменения системных файлов. Проект основан на коде, изначально предоставленном компанией Tripwire в 2000 году. Tripwire включает два основных компонента: политику и базу данных. Политика содержит список всех файлов и

каталогов, для которых средство проверки целостности должно сделать снимок, в дополнение к созданию правил для выявления нарушений изменений в каталогах и файлах. База данных состоит из снимка, сделанного Tripwire. Tripwire также имеет файл конфигурации, в котором указывается расположение базы данных, файла политики и исполняемого файла Tripwire. Для файлов политики и конфигурации, а также файлов базы данных и сгенерированных отчетов обеспечивается криптографическая защита (для этих целей предоставляются два криптографических ключа – ключ сайта и локальный ключ). Tripwire работает, периодически сравнивая каталоги и файлы со снимком в базе данных и сообщая о любых изменениях.

Для дистрибутивов Linux на основе RPM доступны централизованные механизмы, реализующие восстановление пакетов (в том числе и относящихся к ядру), то есть откат к предыдущим версиям. Для это используется менеджер пакетов YUM (Yellowdog Updater, Modified). С помощью команды yum downgrade можно откатиться к предыдущей версии пакета, а yum history позволяет просматривать историю обновлений пакета и отменять отдельные транзакции. Следует отметить, что при откате зависимости пакета не проверяются и не изменяются, так что их надо обрабатывать отдельно.

ОС Linux не имеет автоматизированных инструментов восстановления наподобие точек восстановления Windows-систем. Алгоритм предпринимаемых для восстановления работоспособности ОС действий зависит от причины возникновения проблем (сбой файловой системы, некорректное обновление дистрибутива, некорректные настройки и прочее). Вместе с тем существует ряд решений, позволяющих создавать снимки системы наподобие точек восстановления Windows.

TimeShift (https://github.com/teejee2008/timeshift) – инструмент восстановления Linux-системы с графическим интерфейсом. Создает снимки файловой системы с использованием rsync и жестких ссылок или снимков файловой системы BTRFS. Поддерживает создание снимков по заданному расписанию, несколько уровней резервного копирования и фильтры исключений. TimeShift предназначен только для резервирования и восстановления системных файлов, по умолчанию пользовательские данные не включены в снимок. Можно указать чтобы в резервные копии системы входили выбранные данные из домашней папки пользователя. Состояние системы можно восстановить со снимка во время работы системы или при загрузке с Live CD / USB. TimeShift доступен для Debian/Ubuntu/Linux Mint, последние релизы работают в Wayland-сеансах в Fedora и других дистрибутивах.

Systemback (https://sourceforge.net/projects/systemback/) – простая утилита с графическим интерфейсом для создания резервных копий и точек восстановления системы и конфигурационных файлов пользователя. Резервные копии создаются только по желанию пользователя. Доступны

некоторые дополнительные возможности, такие как создание Live-образа из существующей установки, установка системы с созданного образа, обновление системы, копирование системы из одного раздела в другой и прочее. Существует возможность восстановить или переустановить загрузчик GRUB 2. Разработка и поддержка Systemback прекращена автором в апреле 2017 года, однако утилита работает в современной версии Linux Ubuntu 18. Для установки Systemback в Ubuntu можно воспользоваться PPA-репозиторием.

Функции криптографической защиты

В ОС Linux поддержка шифрования на уровне файловых систем введена сравнительно недавно – начиная с версии ядра 4.1 и реализуется fscrypt − библиотекой, в которую могут подключаться файловые системы для поддержки прозрачного шифрования каталогов. С версии Linux ядра 4.1 fscrypt интегрирована в файловую систему ext4, F2FS и UBIFS поддерживают шифрование с версии ядра 4.2 и 4.10 соответственно. Полноценное шифрование ext4 поддерживается в ОС Linux Ubuntu с версии 18.04. Как правило, возможности шифрования используются для защиты домашней папки пользователя. Встроенное шифрование ext4 осуществляется с помощью утилиты e4crypt, входящей в пакет e2fsprogs.

fscrypt шифрует содержимое файлов, но оставляет открытыми метаданные файловой системы за исключением имен файлов (например, размеры файлов, права доступа, временные метки и расширенные атрибуты). Кроме того, наличие и расположение нераспределенных блоков, которые логически содержат все нули, в файлах не защищены. Как и в ОС Windows, шифрование системных объектов невозможно – запрещено шифрование каталога /root. Не существует методов снятия шифрования с зашифрованного каталога.

Реализация шифрования на уровне файловых систем в ОС Linux имеет ряд особенностей. Так, не поддерживается шифрование уже существующих файлов. Вместо этого доступна маркировка пустого каталога как зашифрованного. Затем, после того как пользовательское пространство предоставит ключ, все обычные файлы, каталоги и символические ссылки, созданные в этом каталоге и его подкаталогах, будут прозрачно зашифрованы. Имена файлов в зашифрованном каталоге шифруются, как и содержимое файлов. Остальная метаинформация файловой системы, включая наличие и расположение нераспределенных блоков, остается открытой.

Если файл из зашифрованной папки был открыт пользователем, его содержимое и имя кэшируется и становится доступным другим пользователям в той же системе, независимо от видимости ключа. Поэтому защита открытых файлов может быть обеспечена только разграничением прав доступа. Кроме того, если к зашифрованному файлу недавно обращались,

ключи остаются в памяти (а, следовательно, имена и открытое содержимое файлов можно восстановить) по крайней мере, до тех пор, пока файловая система не будет размонтирована или не будут удалены кэши файловой системы. Также в настоящее время не реализована защита от расшифрования файлов другого пользователя с неправильно указанным паролем (что может сделать их временно недоступными для законного владельца).

Для получения доступа к файлам из зашифрованного каталога в начале каждого сеанса работы ОС Linux пользователь должен сгенерировать ключи. При использовании стандартных параметров генерации ключи, получаемые разными пользователями на основе одного и того же пароля, совпадут.

В Linux ядре 2.6 и более поздних доступен криптографический модуль dm-crypt, обеспечивающий полнодисковое шифрование логических разделов. Этот модуль имеется во всех построенных на Linux ядре системах, включая ОС Android. При этом для мобильных устройств, как правило, используются предустановленные ключи шифрования, зашитые в защищенные области внутренней памяти устройства. Сами ОС Linux поддерживают стандарт (протокол) шифрования блочного устройства LUKS (Linux Unified Key Setup). LUKS позволяет зашифровать как внутренние диски компьютера, так и съемные носители, а также тома RAID, логические тома LVM и файлы криптоконтейнеров. LUKS также поддерживает отображение содержимого криптоконтейнеров TrueCrypt и VeraCrypt. LUKS реализует прозрачное шифрование, поддерживая шифрование и расшифровывание на лету.

LUKS-шифрование реализуется утилитой cryptsetup и использует модуль ядра dm-crypt. Фактически LUKS добавляет еще один слой, предназначенный для управления криптографическими ключами. При использовании dm-crypt криптографический ключ задается напрямую пользователем, который должен самостоятельно обеспечить его непредсказуемость. Кроме того, dm-crypt использует единый ключ шифрования, который не может быть изменен в дальнейшем. LUKS же самостоятельно генерирует криптографические ключи на основе пользовательского пароля или ключевого файла. Кроме того, LUKS, в отличие от dm-crypt, проводит проверку правильности введенного пароля и хранение настроек криптографии. Если при использовании dm-crypt будет введен неправильный пароль или указаны неверные настройки шифрования, данные раздела не будут корректно расшифрованы (что фактически означает потерю доступа к ним), кроме того, это может привести к нарушению работы ПО, в том числе, самой ОС.

При использовании LUKS в начало шифруемого раздела помещается LUKS заголовок (partition header), содержащий необходимую информацию о настройках шифрования, ключевая информация (KM, key material), далее располагаются сами зашифрованные мастер-ключом данные – рис. 3.2.

Всего может быть задано до 8 ключей шифрования, каждый из которых позволяет получить доступ к содержимому зашифрованного раздела.

LUKS заголовок	KM1	KM2	…	KM8	Зашифрованные данные

Рис. 3.2. Структура зашифрованного раздела LUKS

Таким образом, когда пользователь вводит пароль, на его основе система может вычислить ключ расшифрования мастер-ключа и разблокировать зашифрованный диск. Модуль dm-crypt и использующая его утилита LUKS-шифрования cryptsetup позволяют создавать в каталоге /dev/mapper виртуальное блочное устройство с шифрованием, прозрачным для файловой системы и пользователя. При этом фактически все данные находятся на зашифрованном физическом разделе. Если пользователь пытается записать данные на виртуальное устройство, они на лету шифруются и записываются на диск. При чтении с виртуального устройства, выполняется обратная операция − данные считываются с физического диска, расшифровываются и передаются в открытом виде через виртуальный диск пользователю. Имеется возможность шифрования не только разделов и дисков, но и файлов (например, содержащих образы дисков), при создании в них файловой системы и подключении как loop устройств.

В Linux системах можно шифровать только чистые диски, так как все данные при полнодисковом шифровании носителя уничтожаются. Использование российской криптографии возможно при реализации в виде модуля расширения ядра.

Для криптографической изоляции данных пользователя можно использовать шифрование на уровне драйвера ФС ext4, либо виртуальную шифрующую файловую систему EncFS, доступную с ядра версии 2.6.19 и реализованную поверх физических ФС. Для шифрования раздела он должен быть смонтирован с типом файловой системы ecryptfs, после этого шифрование осуществляется прозрачно для пользователя. Можно отметить и более ранние реализации виртуальных шифрующих файловых систем, обеспечивающих сквозное шифрование, такие как CFS (Cryptographic File System) и TCFS (Transparent Cryptographic File System), являющиеся расширением NFS.

3.3. Безопасность виртуальных ОС

Современные технологии виртуализации помогают получать максимальную отдачу от предоставленных ей вычислительных ресурсов, сокращают издержки на содержание парка программного и аппаратного обеспе-

чения за счет переноса его в виртуальное пространство. На сегодняшний день практически все компании, представляющие крупный или средний бизнес, стараются повысить управляемость информационных систем, сократив при этом расходы на эксплуатацию. Виртуализация серверов и рабочих станций позволяет программно или аппаратно эмулировать виртуальный сервер\компьютер, имеющий, как и все компьютеры, процессор, память, жесткий диск, и т.д. Данная эмуляция происходит благодаря монитору виртуальных машин (или гипервизору).

Основная проблема обеспечения безопасности виртуализированной среды связана с тем, что традиционные средства защиты информации не способны обеспечить защиту от новых угроз безопасности информации, специфичных именно для виртуальной инфраструктуры. Привычные решения зачастую несовместимы со средой виртуализации, так как изначально разрабатывались для использования в физической среде. Если нарушитель получает доступ к средствам управления виртуальной инфраструктурой, операционная среда традиционных средств защиты информации оказывается полностью скомпрометированной. Например, через гипервизор нарушитель может незаметно для традиционных средств защиты информации, работающих в виртуальных машинах, совершать следующие злоумышленные действия:

– копировать и блокировать поток данных, идущий на все устройства (HDD, принтер, USB, сеть, дискеты);

– читать и изменять данные на дисках виртуальных машин, даже когда они выключены или не работают, без участия ПО этих виртуальных машин.

Серьезной проблемой для виртуальных машин являются обновления ОС. Обновление необходимо для устранения текущих уязвимостей и установки патчей. Обновить виртуальную машину довольно сложно, так как не всегда известно, на каком сервере она находится (Live Migration), в случае неграмотного менеджмента есть вероятность запутаться. В связи с этим необходимо четко задать каталогизацию виртуальных машин. Snapshots / Suspend также затрудняют процесс управления обновлениями.

Snapshot – моментальный снимок, копия файлов и директорий файловой системы на определенный момент времени. Это довольно полезная функция виртуальных машин, так как, например, в случае падения сервера восстановить его из снапшота – дело нескольких десятков минут. С другой стороны, снапшоты занимают довольно большие объемы винчестеров, само создание снапшота – довольно длительный процесс. Чаще всего в момент создания снапшота виртуальная машина недоступна для конечных пользователей. Suspend – это пауза системы. Она может требоваться, например, для снятия нагрузки с энергосети на ночь. В случае если той же ночью необходимо сделать резервную копию это будет невозможно.

Здесь, как и на физической платформе, необходима установка средств защиты в гостевой ОС (антивирусы, сетевые экраны и прочее), а также правильная настройка виртуальных машин и виртуальных сетей.

В 2017 году введен в действие ГОСТ Р 56938-2016 «Защита информации. Защита информации при использовании технологий виртуализации. Общие положения» (https://fstec.ru/tk-362/standarty-tk362/1675-perechen-tk362), в котором уточнены основные термины и понятия, определены объекты защиты, рассмотрены угрозы при использовании технологий виртуализации, а также представлен перечень мер защиты от этих угроз.

Для решения задачи безопасной эксплуатации виртуальной ИС компанией «Код безопасности» разработано СЗИ от НСД vGate R2, которое обеспечивает безопасность самых распространенных платформ виртуализации – VMware vSphere и Microsoft Hyper-V – и при этом сертифицировано ФСТЭК России. Сертификат подтверждает соответствие требованиям руководящих документов:

– в части защиты от несанкционированного доступа по 5 классу защищенности (СВТ5);

– для контроля отсутствия недекларированных возможностей по 4 уровню контроля (НДВ4);

– по классу защищенности в автоматизированных системах (АС) до 1Г включительно;

– для защиты информации в ИСПДн до 1 класса включительно.

Основные возможности СЗИ НСД vGate R2 (securitycode.ru/products/vgate/):

– назначение политик безопасности напрямую на объекты виртуальной инфраструктуры;

– фильтрация трафика виртуальных машин на уровне гипервизора;

– противодействие ошибкам и злоупотреблениям при управлении виртуальной инфраструктурой;

– защита от специфических угроз, характерных для виртуальных сред;

– группировка объектов доступа и автодобавление виртуальных машин в группы;

– мандатное управление доступом к управлению виртуальной инфраструктурой;

– контроль целостности конфигурации виртуальных машин и доверенная загрузка;

– контроль доступа администраторов ИС к данным виртуальных машин;

– регистрация и аудит событий безопасности;

– доверенная загрузка и контроль целостности виртуальных машин;

– защита от несанкционированного копирования, клонирования, переноса и уничтожения виртуальных машин;

– централизованное управление и мониторинг.

С целью более эффективной защиты сервера виртуализации помимо vGate R2, необходимо применить программно-аппаратный комплекс (ПАК) «Соболь» версии 3.0 для реализации следующих защитных механизмов:

– идентификация и аутентификация пользователей на входе в систему (непосредственно при запуске сервера);

– ведение журнала безопасности;

– сигнализация попыток нарушения защиты;

– запрет загрузки с внешних носителей.

Перечисленный комплекс возможностей не позволит злоумышленнику (в случае если он получил физический доступ к серверу виртуализации) реализовать одну из наиболее распространенных угроз – перезагрузку сервера и загрузку с внешнего носителя для получения доступа ко всей информации, хранящейся на сервере.

Наличие у продуктов сертификатов ФСТЭК России позволяет использовать vGate R2, vGate-S и ПАК «Соболь» версии 3.0 для защиты информации, составляющей коммерческую или государственную тайну в автоматизированных системах с классом защищенности до 1Б включительно. На момент написания пособия проходит сертификацию последняя версия - vGate 4.1.

3.4. Безопасность ОС мобильных устройств

Одной из наиболее заметных тенденций развития компьютерной техники в последние годы является неуклонный рост вычислительной мощности мобильных электронных устройств: мобильных телефонов, планшетных компьютеров, электронных книг и т. п. Начиная примерно с 2007 года, граница между персональными компьютерами и мобильными электронными гаджетами стала заметно стираться. Сегодня почти не вызывает удивления ситуация, когда мобильный телефон превосходит некоторые персональные компьютеры по вычислительной мощности процессора, объему оперативной и долговременной памяти. Современные мобильные устройства оснащаются полноценными ОС, близкими по своим возможностям к универсальным ОС персональных компьютеров. Так, самая распространенная на сегодняшний день мобильная ОС Google Android использует то же ядро, что и универсальная ОС Linux, а вторая по популярности мобильная Apple iOS построена на основе прикладных интерфейсов универсальных ОС семейства BSD UNIX [8].

Типичные угрозы безопасности ОС мобильного устройства существенно отличаются от аналогичных угроз для ОС персонального компьютера или сетевого сервера. Некоторые угрозы, малозначительные для обычных компьютеров, становятся очень опасными для мобильных

устройств, и наоборот. Программная закладка, внедренная в ОС персонального компьютера и получившая доступ к электронным банковским счетам пользователя, обычно имеет весьма ограниченные возможности по несанкционированному переводу денежных средств с этих счетов. Но программная закладка, внедренная в ОС мобильного устройства, элементарно решает данную задачу путем несанкционированного заказа дорогостоящих SMS-услуг с телефонного номера, контролируемого нарушителем. С другой стороны, угрозы, связанные с одновременным доступом нескольких пользователей к одному экземпляру ОС, для мобильных ОС, как правило, неактуальны.

К наиболее актуальным угрозам безопасности мобильных ОС обычно относят следующие:

– раскрытие конфиденциальной информации в результате утери или кражи мобильного устройства;

– несанкционированный заказ дорогостоящих услуг программной закладкой, внедренной в ОС мобильного устройства;

– раскрытие конфиденциальной информации в результате перехвата беспроводного сетевого трафика, генерируемого мобильным устройством;

– несанкционированный сбор программной закладкой персональных данных пользователя мобильного устройства;

– потеря данных, хранящихся на мобильном устройстве.

В настоящее время для мобильных ОС разрабатывается огромное количество вредоносного ПО. По данным «Лаборатории Касперского» за 2018 год, вредоносное ПО, предназначенное для ОС Android, классифицируется следующим образом:

– 49 % – программные закладки, собирающие персональные данные пользователей для последующей рассылки спама;

– 25 % – программные закладки, несанкционированно заказывающие услуги путем отправки SMS;

– 18 % – многофункциональные боты, входящие в состав ботнетов;

– 2 % – программные закладки, специализирующиеся на сборе информации о банковских счетах;

– 6 % – другие вредоносные приложения.

Типичными симптомами заражения мобильного устройства вредоносными программами являются:

– необъяснимо быстрое уменьшение денежных средств на счету;

– появление новых ярлыков в списке установленных приложений;

– появление непонятных записей в папках полученных и отправленных SMS-сообщений или в списках, входящих и исходящих голосовых вызовов;

– скачкообразное увеличение расхода электроэнергии без видимых причин.

Не менее серьезной проблемой, чем заведомо вредоносное ПО, являются некорректно работающие мобильные приложения, причиняющие ущерб пользователю непреднамеренно в результате допущенных разработчиками программных ошибок. Пользователи часто не понимают, что причина некорректной работы мобильного устройства кроется не в устройстве как таковом, а в недостаточно отлаженном приложении. В результате недоработанные приложения могут оказывать заметное негативное влияние на репутацию разработчиков ОС мобильного устройства и самого мобильного устройства как такового.

Компания Apple решает эту проблему путем тщательного тестирования всех мобильных приложений сторонних производителей, разработанных для iPhone и iPad. В целом данная стратегия позволяет радикально сократить количество и разнообразие вредоносного ПО, разрабатываемого для iOS.

Перечислим некоторые правила, которым должно удовлетворять каждое приложение, доступное через Apple App Store:

– приложение не должно содержать недокументированных функций, как и использовать недокументированные функции операционной системы;

– приложение не должно выполнять загрузку больших файлов из мобильной сети без уведомления пользователя;

– приложение не должно устанавливать или запускать другие приложения;

– приложение не должно обращаться к данным других приложений;

– приложение не должно собирать персональные данные пользователя (включая географическое местоположение) без явно выраженного согласия пользователя.

Компания Google, в отличие от Apple, декларирует максимальную открытость своей ОС Android. Каждое приложение для Android должно быть подписано разработчиком, но заверение этой подписи каким бы то ни было удостоверяющим центром не требуется. В каталог Google Play и интернет-магазин Android Market допускаются все приложения, кроме заведомо вредоносных и заведомо неработоспособных. Однако, открытость ОС Android не следует преувеличивать. В отличие от большинства разработчиков универсальных ОС, компания Google сохраняет за собой контроль над всеми приложениями, работающими на всех экземплярах ОС Android. При необходимости компания Google, так же, как и Apple, может одновременно удалить все экземпляры любого заданного приложения, установленного на всех мобильных устройствах со своей ОС.

Долгое время обеспечение безопасности мобильных ОС рассматривалось их разработчиками как второстепенная, низкоприоритетная задача. Но ситуация постепенно меняется к лучшему, с каждой следующей версией

мобильные ОС становятся все более защищенными. При этом разработчики широко используют решения, апробированные на универсальных ОС. Так, в ОС Android начала поддерживаться функция доверенного подтверждения пользователем потенциально опасного действия, очень похожая на UAC в Windows. Эта функция используется по умолчанию при отправке SMS-сообщений – до тех пор, пока пользователь не подтвердит отправку сообщения, оно не будет отправлено, при этом прикладные программы не имеют технической возможности имитировать данное подтверждение. Для более ранних версий Android существуют приложения третьих фирм (например, LBE Security Master, LBE Privacy Guard), реализующие аналогичную функциональность.

В мобильных ОС широко применяется давно используемая в универсальных ОС идентификация программных файлов по цифровым подписям. Подобно тому, как пользователь Windows может запретить установку на свой компьютер драйверов, не имеющих цифровой подписи Microsoft, пользователь Android может запретить установку на свое устройство программных файлов, безопасность которых не подтверждена компанией Google.

Управление доступом в Android почти не использует унаследованный от Linux дискреционный механизм, основанный на субъектах, объектах, методах и правах доступа. Единственным инструментом управления доступом в Android являются привилегии, которые назначаются, в отличие от универсальных ОС, не пользователям, а приложениям. Фактически, в Android приложения рассматриваются в качестве пользователей. При установке приложения ему назначаются собственные UID и GID, добавляются все необходимые записи в системные файлы Linux, содержащие сведения о зарегистрированных пользователях. Исключением является ситуация, когда несколько приложений, изготовленных одним и тем же разработчиком, явно указывают операционной системе, что их следует выполнять от имени одной и той же учетной записи пользователя. Это может быть сделано посредством атрибута sharedUserld манифеста AndroidManifest.xml, включаемого в дистрибутив Android-приложения.

Поскольку каждое приложение Android в обязательном порядке снабжается цифровой подписью разработчика, теоретически возможно управление доступом приложений на основе информации об их разработчиках, например, разделение приложений на более доверенные и менее доверенные. Но на практике эта возможность пока не используется.

Перечислим некоторые привилегии Android, часто запрашиваемые приложениями:

– ACCESS.COARSE.LOCATION – получение доступа к информации о местоположении мобильного устройства, предоставляемой сетями GSM и WiFi;

– ACCESS_FINE_LOCATION – получение доступа к информации о местоположении мобильного устройства, предоставляемой модулем GPS;

– ACCESS_NETWORK.STATE – получение информации о GSM-сети, к которой подключено устройство;

– ACCESS_WIFLSTATE – получение информации о WiFi-сети, к которой подключено устройство;

– CALL_PHONE – инициация голосовых звонков, за исключением звонков на номера экстренной помощи;

– CAMERA – доступ к фото/видеокамере;

– INTERNET – доступ к Интернет;

– RECEIVE.SMS – возможность обработки входящих SMS-сообщений;

– RECORD_AUDIO – доступ к микрофону;

– WAKE_LOCK – предотвращение перехода устройства в режим сна;

– WRITE_EXTERNAL_STORAGE – запись данных на карту внешней памяти устройства.

ОС Android не поддерживает динамическое включение и выключение привилегий. Все привилегии, требуемые приложением, предоставляются ему сразу при инсталляции и затем автоматически включаются при каждом запуске. Перечень требуемых приложению привилегий включается в манифест приложения (файл AndroidManifest.xml).

Если в ходе выполнения той или иной операции обнаруживается, что привилегия, необходимая для завершения данной операции, не предоставлена приложению, в приложении возникает исключительная ситуация SecurityException. Почти все случаи отказа в доступе из-за отсутствия необходимых привилегий регистрируются в системном журнале. Кроме того, ОС может выдавать пользователю интерактивные сообщения об ошибках, связанных с попытками нарушения действующей политики безопасности теми или иными приложениями.

Пользователь может просматривать список привилегий, поддерживаемых в текущей конфигурации ОС, с помощью главного меню пользовательского интерфейса (пункт Settings/ Application) или в командной строке командой adb shell pm list permissions.

Временные данные приложений, записываемые во внутреннюю память устройства, хранятся внутри домашних директорий соответствующих пользователей Linux. Поскольку в Linux пользователи по умолчанию не имеют доступа к содержимому домашних директорий друг друга, данные (в том числе и cookie посещенных интернет- сайтов) одного приложения Android обычно недоступны другим приложениям. Исключением является ситуация, когда приложение явно предоставляет доступ другим приложениям к своим данным.

В iOS система управления доступом в целом похожа на аналогичную систему Android, но устроена намного проще и примитивнее. Привилегий поддерживается всего пять:

– обращаться к входящей электронной почте;
– обращаться к входящим SMS;
– отсылать SMS;
– инициировать телефонные звонки;
– обращаться к подсистеме GPS.

Перед тем, как воспользоваться любой из трех последних привилегий, приложение iOS должно получить явно выраженное согласие пользователя на выполнение соответствующего действия.

Как правило, на устройствах, работающих под управлением iOS, выполняются только доверенные приложения, тщательно проверенные и одобренные специалистами Apple. Поэтому к подсистеме управления доступом iOS не предъявляется высоких требований. Таким образом, несмотря на свою крайнюю примитивность, подсистема управления доступом iOS представляется вполне удовлетворительной.

Принцип минимизации полномочий соблюдается в базовых конфигурациях как Android, так и iOS – ни одно приложение не имеет технической возможности получить полномочия суперпользователя root. Однако многие пользователи Android и некоторые пользователи iOS отключают эту защиту. В результате приложения, в том числе и вредоносные, получают больше функциональных возможностей, в число которых входят возможности реализовывать стелс-технологии и активно противодействовать средствам администрирования и антивирусному ПО. Всякое приложение, выполняющееся телефоне с полномочиями суперпользователя, имеет ничем не ограниченный доступ ко всем объектам ОС телефона. Когда недостаточно отлаженный или заведомо вредоносный программный код получает такой доступ, последствия могут стать фатальными как для самой ОС, так и для хранимых и обрабатываемых в ней пользовательских данных.

Операционная система iOS хранит все данные в пользовательской части внешней памяти в зашифрованном виде. Наиболее конфиденциальные данные подвергаются дополнительному шифрованию. Используемые при этом ключи не предоставляются пользователем, а хранятся во внутренней памяти устройства, т. е. защита фактически реализуется только от нарушителя, пытающегося прочитать внешнюю память данного устройства, используя другое устройство. В Android, в отличие от iOS, шифрование данных практически не применяется, хотя и поддерживается.

Практически все производители антивирусного ПО уже разработали версии своих программ для ОС Android. По архитектуре и функциональности мобильные антивирусные комплексы не имеют принципиальных отличий от антивирусных комплексов, предназначенных для применения на персональных компьютерах или серверах. Многие эксперты отмечают, что эффективность мобильных антивирусных комплексов существенно уступает эффективности тех же антивирусных комплексов для универсальных ОС.

Начиная с версии Android 4.2, поддерживается режим, когда каждое приложение, независимо от источника, из которого оно установлено, должно автоматически проверяться облачным антивирусным сканером Google.

Антивирусные приложения для мобильных ОС часто дополняются так называемой «противоугонной» функциональностью, позволяющей легальному пользователю устройства найти свое устройство по сигналам GPS, удаленно заставить устройство издать громкий звуковой сигнал, привлекающий внимание окружающих к вору, удаленно уничтожить персональные данные, хранящиеся на устройстве и т. п. Эффективность такой защиты оставляет желать лучшего. После выключения и полной перепрошивки украденного устройства весь противоугонный функционал отключается. Кроме того, даже исправно работающая противоугонная программа не всегда позволяет вернуть украденное устройство.

Некоторые мобильные антивирусы (например, Avast) включают в себя дополнительную функциональность, позволяющую пользователю осуществлять фильтрацию входящих и исходящих голосовых звонков и SMS-сообщений, вводить белые и черные списки телефонных номеров.

Как и у любых других операционных систем, ПО мобильных ОС может иметь уязвимости, позволяющие вредоносным приложениям несанкционированно повышать свои полномочия. Так, вредоносные приложения Dream-Exploid, Wukong и Gongfu эксплуатировали уязвимости старых версий Android, чтобы несанкционированно получать полномочия суперпользователя root и устанавливать полный контроль над зараженной системой.

Отметим, что программный код приложений iOS выполняется непосредственно на процессоре мобильного устройства, в то время как программный код каждого приложения Android работает в виртуальной Java-машине. Однако, это различие не оказывает заметного влияния на сравнительную безопасность этих двух операционных систем.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Васильева И. Н. Криптографические методы защиты информации: учебник и практикум для академического бакалавриата. – М.: Юрайт, 2016. – 349 с.

2. Васильева И. Н. Управление информационной безопасностью: учебное пособие. – СПб.: Изд-во СПбГЭУ, 2014. – 82 с.

3. Васильева И. Н. Управление рисками информационной безопасности: учебное пособие. СПб.: Изд-во СПбГЭУ, 2016. – 177 с.

4. Гольдштейн Б. С. Протоколы AAA: RADIUS и Diameter/ Б.С. Гольдштейн, В.С. Елагин, Ю.Л. Сенченко// Серия «Телекоммуникационные протоколы». Книга 9 – СПб.: БХВ Петербург, 2014. – 352 с.: ил.

5. ГОСТ Р 50922-2006. Защита информации. Основные термины и определения. – М.: Стандартинформ, 2008. – 8 с.

6. Информационная безопасность цифрового пространства / под ред. Е. В. Стельмашонок, И. Н. Васильевой. – СПб.: Изд-воСПбГЭУ, 2019. – 155 с.

7. Кокорева О. Реестр Windows 7. – СПб.: БХВ-Петербург, 2010. – 704 с.

8. Проскурин В. Г. Защита в операционных системах. Учебное пособие.- М.: Горячая линия – Телеком, 2004. – 192с.

9. Руссинович М., Соломон Д. Внутреннее устройство Microsoft Windows. 6-е изд. Часть 1. – СПб.: Питер, 2013. – 800 с.

10. Чернокнижный Г. М. Аутентификация в информационных системах: как это есть и как это должно быть / Г.М. Чернокнижный, А. Г. Боховко// Актуальные вопросы безопасности современных информационных технологий/ СПб.: СПб ГЭУ, 2015. – С.4-14.

11. Чернокнижный Г. М. Вычислительные сети. Контроль безопасности в компьютерных сетях: учебное пособие. – СПб.: Изд-во СПбГЭУ, 2016. – 97с.

12. Microsoft NTLM [Электронный ресурс]. URL: https://msdn.microsoft.com/en-us/library/windows/desktop/ aa378749(v=vs.85).aspx (дата обращения: 10.09.19).

13. SDB: Ima evm [Электронный ресурс]. URL: https://en.opensuse.org/ SDB:Ima_evm (дата обращения: 25.08.2019).

Учебное издание

Ченокнижный Геннадий Михайлович
Васильева Ирина Николаевна

БЕЗОПАСНОСТЬ ОПЕРАЦИОННЫХ СИСТЕМ

Учебное пособие

Издано в авторской редакции

Подписано в печать 18.12.19. Формат 60×84 1/16.
Усл. печ. л. 6,75. Тираж 60 экз. Заказ 1465.

Издательство СПбГЭУ. 191023, Санкт-Петербург, Садовая ул., д. 21.

Отпечатано на полиграфической базе СПбГЭУ