Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КАЛИНИНГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ»

В. В. Подтопельный

БЕЗОПАСНОСТЬ ОПЕРАЦИОННЫХ СИСТЕМ

Учебно-методическое пособие по изучению дисциплины
для студентов по специальности
10.05.03 «Информационная безопасность автоматизированных систем»,
специализация «Безопасность открытых информационных систем»

Калининград
Издательство ФГБОУ ВО «КГТУ»
2025

УДК 004.056(076)

Рецензент
кандидат физико-математических наук, доцент, доцент кафедры информационной безопасности ФГБОУ ВО «Калининградский государственный технический университет» Н. Я. Великите

Подтопельный, В. В.
Безопасность операционных систем: учебно-методическое пособие по изучению дисциплины для студентов специальности 10.05.03 «Информационная безопасность автоматизированных систем», специализация «Безопасность открытых информационных систем» / В. В. Подтопельный. – Калининград: Изд-во ФГБОУ ВО «КГТУ», 2025. – 90 с.

Учебно-методическое пособие является руководством по изучению дисциплины «Безопасность операционных систем». В учебно-методическом пособии приведен тематический план изучения дисциплины. Представлены методические указания по изучению дисциплины. Даны рекомендации по подготовке к промежуточной аттестации в форме зачёта и экзамена, по выполнению самостоятельной работы. Пособие подготовлено в соответствии с требованиями утвержденной рабочей программы модуля. Пособие предназначено для студентов 2, 3 курсов специальности 10.05.03 «Информационная безопасность автоматизированных систем».
Табл. 2, список лит. – 27 наименований

Учебно-методическое пособие по изучению дисциплины рекомендовано к использованию в качестве локального электронного методического материала в учебном процессе методической комиссией ИЦТ 26 мая 2025 г., протокол № 4

УДК 004.056(076)

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Данное учебно-методическое пособие предназначено для студентов специальности 10.05.03 "Информационная безопасность автоматизированных систем", специализация: «Безопасность открытых информационных систем», изучающих дисциплину «**Безопасность операционных систем».**

Наименование компетенции:

ОПК-12. Способен применять знания в области безопасности вычислительных сетей, операционных систем и баз данных при разработке автоматизированных систем

Цель освоения дисциплины является: формирование у студентов знаний о принципах построения операционных систем, защиты в операционных системах (ОС), навыки определения и внедрения комплекса мер (правил, процедур, практических приемов, руководящих принципов, методов, средств) для защиты операционных систем.

В результате освоения дисциплины обучающийся должен:

знать:

– способы реализации угроз безопасности в операционных системах;

– способы реализации угроз безопасности в автоматизированных системах;

уметь:

– формировать перечень мероприятий по предотвращению угроз безопасности операционных систем, информации в операционных системах;

владеть:

– навыками выявления уязвимости информационно-технологических ресурсов автоматизированных систем;

– навыками определения комплекса мер (правил, процедур, практических приемов, руководящих принципов, методов, средств) для защиты операционных систем.

Для успешного освоения дисциплины, в соответствии с учебным планом, ей предшествуют дисциплины **«Основы информационной безопасности», «Технологии и методы программирования»**.

Далее в пособии представлен тематический план, содержащий перечень изучаемых тем, обязательных лабораторных работ, мероприятий текущей аттестации и отводимое на них аудиторное время (занятия в соответствии с расписанием) и самостоятельную работу. При формировании личного образовательного плана на семестр следует оценивать рекомендуемое время на изучение дисциплины, возможно, вам потребует больше времени на выполнение отдельных заданий или проработку отдельных тем.

В разделе Содержание дисциплины приведены подробные сведения об изучаемых вопросах по которым вы можете ориентироваться в случае пропуска каких-то занятий, а также методические рекомендации преподавателя для

самостоятельной подготовки, каждая тема имеет ссылки на литературу (или иные информационные ресурсы), а также контрольные вопросы для самопроверки.

Раздел «Текущая аттестация» содержит описание обязательных мероприятий контроля самостоятельной работы и усвоения разделов или отдельных тем дисциплины. Далее изложены требования к завершающей аттестации – зачету и экзамену.

Помимо данного пособия, студентам следует использовать материалы, размещенные в соответствующем данной дисциплине разделу ЭИОС, в которые более оперативно вносятся изменения для адаптации дисциплины под конкретную группу.

Программное обеспечение:

1. Операционная система Astra Linux SE.
2. Операционная система Windows 10 (получаемая по программе Microsoft «Open Value Subscription»).
3. Офисное приложение LibreOffice.
4. Google Chrome (GNU) 4. Oracle VM VirtualBox (GNU/Linux, macOS и Windows).
5. Офисное приложение MS Office Standard 2016. (получаемое по программе Microsoft «Open Value Subscription»).
6. Oracle VirtualBox 7.1.6 и VirtualBox Extension Pack 7.1.6 for x86_64 hardware.
7. Google Chrome (GNU).

1. ТЕМАТИЧЕСКИЙ ПЛАН

Таблица 1 – Тематический план

	Раздел (модуль) дисциплины	Тема	Объем аудиторной работы, ч.	Объем самостоятельной работы, ч.
Лекции (4 семестр – 32 ч. ауд.)				
1.1	Современные операционные системы	Тема 1.1 Введение. Основные понятия. Классификация операционных систем. Концептуальные основы операционных систем.	4	
1.2		Тема 1.2 Архитектура операционной системы	4	10
1.3		Тема 1.3 Управление процессами	4	
1.4		Тема 1.4 Управление памятью	4	10
1.5		Тема 1.5 Прерывания	4	
1.6		Тема 1.6 Управление вводом-выводом	4	10
1.7		Тема 1.7 Файловая система	8	7,85
Лекции (5 семестр – 48 ч. ауд.)				
2.1	Подсистемы защиты операционной системы	Тема 2.1 Основные функции подсистемы защиты операционной системы	8	10
2.2		Тема 2.2 Управление доступом в операционных системах семейства UNIX	8	10
2.3		Тема 2.3 Управление доступом в операционных системах семейства Windows	8	10
2.4		Тема 2.4 Идентификация и аутентификация в ОС Linux	8	10
2.5		Тема 2.5 Идентификация и аутентификация в ОС Windows	8	10
2.6	Подсистемы защиты операционной системы	Тема 2.6 Аудит ОС	8	3
			80	**90,85**

	Раздел (модуль) дисциплины	Тема	Объем аудиторной работы, ч.	Объем самостоятельной работы, ч.
		лабораторные занятия (4 семестр)		
	Современные операционные системы	Лабораторная работа № 1. Работа с файлами и дисками в ОС Windows	4	-
		Лабораторная работа № 2. Организация пакетных файлов и сценариев в ОС Windows	4	-
		Лабораторная работа № 3. Организация консоли администрирования в ОС Windows	4	-
		Лабораторная работа № 4. Мониторинг, оптимизация и аудит ОС Windows	4	-
		Лабораторная работа № 5. Работа с Реестром ОС Windows	4	-
		Лабораторная работа № 6. Работа с подсистемой безопасности в ОС Windows	4	-
		Лабораторная работа № 7. Модель безопасности ОС Windows	4	-
		Лабораторная работа № 8. Создание и управление доменной политикой	4	-
		Всего за семестр:	32	
		лабораторные занятия (5 семестр)		
1.	Подсистемы защиты операционной системы	Лабораторная работа № 9. Конфигурирование доменной политики	4	-
2.		Лабораторная работа № 10. Конфигурирование и использование EFS. Восстановление данных	4	-
3.	Подсистемы защиты операционной системы	Лабораторная работа № 11. ОС семейства UNIX. Работа с файлами и каталогами. Управление пользователями. Защита	8	-

	Раздел (модуль) дисциплины	Тема	Объем аудиторной работы, ч.	Объем самостоятельной работы, ч.
		файлов. Резервное копирование данных		
4.		Лабораторная работа № 12. Работа с процессами в операционной системе LINUX	8	-
5.		Лабораторная работа № 13. Особенности ОС Linux	8	-
6.		Лабораторная работа № 14. Механизмы безопасности в Linux	32	-
		Всего за семестр:	**32**	
		Всего	**64**	
		Курсовая работа (проект)		
2.1	Название первого раздела	Контрольная точка 1. Раздел 1	-	-
3.1	Название третьего раздела	Контрольная точка 2. Раздел 2	-	-
		Оформление курсовой работы. Защита	-	-
			34,75	**0**
		Всего:	РЭ – 14 КА– 4,4	
		Рубежный (текущий) и итоговый контроль		
2.1	Название второго раздела	Контроль 1 (не предусмотрен)	-	-
3.1	Название третьего раздела	Контроль 2 (не предусмотрен)	-	-
		Итоговый контроль (зачет)		
		Итоговый контроль (экзамен)		
			0	**0**
		Всего	**197,15**	**90,85**
		ИТОГО 288		

2. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

Раздел 1. Методы защиты ПО

Тема 1.1 Введение. Основные понятия. Классификация операционных систем. Концептуальные основы операционных систем

Перечень изучаемых вопросов

1. Основные понятия;
2. Классификация операционных систем.
3. Концептуальные основы операционных систем: концепция процесса, концепция ресурса, концепция виртуальности, концепция прерывания.

Методические указания к изучению:

Операционная система (ОС) представляет собой фундаментальный слой программного обеспечения, который управляет аппаратными ресурсами компьютера, обеспечивает выполнение приложений и взаимодействие пользователя с вычислительной системой. Безопасность ОС – это совокупность механизмов, направленных на защиту системы от несанкционированного доступа, вредоносных программ.

Предварительно требуется определить понятие системного программного обеспечения. Рассмотреть классификации ОС по областям, по типам, поддержки многозадачности, поддержки многонитевости, особенностям методов построения, режимам обработки данных.

Требуется рассмотреть особенности реализации концепции процесса (процесс задача, программа, задание). Перечень основных состояний процесса. Обратить внимание на классификации процессов.

Требуется рассмотреть концепцию ресурса. Рассмотреть классификацию ресурсов

Требуется рассмотреть концепцию виртуальности.

Требуется рассмотреть концепцию прерывания, действия ОС при выполнении прерывания, классы прерываний.

Ключевыми аспектами безопасности ОС являются:

a. **Конфиденциальность** – предотвращение несанкционированного доступа к данным.

b. **Целостность** – защита информации от несанкционированного изменения.

c. **Доступность** – обеспечение бесперебойной работы системы для авторизованных пользователей.

Угрозы безопасности ОС включают вредоносное ПО (вирусы, трояны, руткиты), эксплуатацию уязвимостей (например, переполнение буфера или race condition), а также атаки на аутентификацию (перехват паролей, подбор учетных данных). Современные ОС используют комплексные механизмы защиты, такие как разграничение прав доступа, системы мониторинга и криптографические методы.

2. Классификация операционных систем

Операционные системы можно классифицировать по нескольким критериям, что помогает лучше понять их архитектуру и особенности обеспечения безопасности.

По типу ядра выделяют:

a. **Монолитные ОС** (например, Linux, Unix), где все компоненты работают в едином адресном пространстве, что обеспечивает высокую производительность, но усложняет защиту от ошибок.

b. **Микроядерные ОС** (QNX, MINIX), в которых критически важные функции вынесены в микроядро, а остальные работают в пользовательском режиме, что повышает отказоустойчивость.

c. **Гибридные ОС** (Windows NT, macOS), сочетающие элементы обоих подходов.

По числу пользователей различают:

a. **Однопользовательские ОС** (Windows 10 Home), предназначенные для персонального использования.

b. **Многопользовательские ОС** (Linux, Windows Server), поддерживающие одновременную работу нескольких пользователей с разграничением прав.

По числу задач ОС делятся на:

a. **Однозадачные** (например, MS-DOS), способные выполнять только одну программу в данный момент.

b. **Многозадачные** (все современные ОС), позволяющие параллельно исполнять несколько процессов.

По назначению ОС могут быть:

a. **Универсальными** (Windows, Linux), предназначенными для широкого круга задач.

b. **Специализированными** (ОС для банкоматов, медицинского оборудования, IoT-устройств), оптимизированными под конкретные задачи.

3. Концептуальные основы операционных систем

3.1. Концепция процесса

Процесс – это экземпляр выполняемой программы, обладающий собственным адресным пространством, регистрами процессора и системными ресурсами. Безопасность процессов обеспечивается за счет:

a. **Изоляции процессов** (каждый процесс работает в своем виртуальном адресном пространстве).

b. **Разграничения прав доступа** (DAC – дискреционное управление доступом, MAC – мандатное управление доступом).

c. **Контроля целостности** (проверка цифровых подписей исполняемых файлов).

3.2. Концепция ресурса

Ресурсы ОС включают процессорное время, оперативную память, устройства ввода-вывода и файлы. Управление ресурсами предполагает:

a. **Планирование доступа (алгоритмы планирования CPU, такие как Round Robin или приоритетное планирование).**

b. **Предотвращение конфликтов (например, deadlock – взаимоблокировка процессов).**
c. **Квотирование ресурсов (ограничение памяти или CPU для отдельных процессов).**

3.3. Концепция виртуальности

Виртуализация позволяет создавать абстракции физических ресурсов, таких как виртуальная память или виртуальные машины. Это повышает безопасность за счет:

a. **Изоляции сред (гостевые ОС в VMware или KVM не имеют прямого доступа к аппаратуре).**
b. **Эмуляции устройств (виртуальные сетевые адаптеры, диски).**
c. **Контейнеризации (Docker, LXC – изоляция процессов без полной виртуализации).**

3.4. Концепция прерывания

Прерывания – это сигналы, требующие немедленной реакции процессора. Они делятся на:

a. **Аппаратные** (например, нажатие клавиши на клавиатуре).

b. **Программные** (системные вызовы, исключения).

Безопасность обработки прерываний обеспечивается за счет:

a. **Привилегированных режимов работы CPU** (ring 0 – ядро, ring 3 – пользовательские процессы).

b. **Верификации источников прерываний** (защита от подделки).

Изучите базовые принципы работы ОС, механизмы управления процессами и ресурсами. Рекомендуемая литература: Таненбаум «Современные операционные системы», Вирт «Алгоритмы и структуры данных». Используйте эмуляторы (QEMU) и системы виртуализации (VirtualBox) для анализа работы ОС. Попробуйте настроить разграничение прав в Linux (chmod, SELinux).

При работе над курсовые работы обратите внимание на:

– описание уязвимостей, которые требуется приводить в первой (теоретической) главе.

Литература

1. Окороков, В. А. Безопасность операционных систем / В. А. Окороков. – Санкт-Петербург: Лань, 2024. – 228 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/367472 (дата обращения: 06.12.2024). – ISBN 978-5-507-48297-9. – Текст : электронный (гл. 1).

2. Киренберг, Г. А. Информационная безопасность современных операционных систем: учеб. пособие / Киренберг, Г. А. – Кемерово: КузГТУ им. Т. Ф. Горбачева, 2022. – 138 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/295736 (дата обращения: 06.12.2024). – ISBN 978-5-00137-320-9. – Текст : электронный (гл. 1).

3. Потерпеев, Г. Ю. Безопасность операционных систем: учеб. пособие / Г. Ю. Потерпеев, В. С. Нефедов, А. А. Криулин. – Москва: РТУ МИРЭА, 2021. – 93 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/182416 (дата обращения: 06.12.2024). – ISBN 978-5-7339-1393-3. – Текст : электронный (гл. 1).

4. Зверева, О. М. Операционные системы: учеб. пособие / О. М. Зверева; науч. ред. Л. Г. Доросинский; Уральский федеральный университет им. первого Президента России Б. Н. Ельцина. – Екатеринбург: Издательство Уральского университета, 2020. – 223 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=699030 (дата обращения: 03.11.2024). – ISBN 978-5-7996-3146-8. – Текст : электронный (гл. 1).

Контрольные вопросы

1. Приведите классификацию операционных систем.
2. Приведите классификации процессов.
3. Дайте определение процесса.
4. Охарактеризуйте классификацию ресурсов.
5. Приведите порядок действий ОС при выполнении прерывания.

Тема 1.2 Архитектура операционной системы

Перечень изучаемых вопросов

1. Совместимость и множественные прикладные среды.
2. Привилегированный режим работы ядра.
3. Многоуровневая организация операционных систем.
4. Аппаратная независимость и переносимость.
5. Микроядерная архитектура и безопасность.
6. Совместимость и множественные прикладные среды.

Методические указания к изучению

Современные операционные системы представляют собой сложные программные комплексы, архитектура которых определяет их функциональность, производительность и, что особенно важно, уровень безопасности. Центральным элементом любой ОС является ядро – основной компонент, обеспечивающий взаимодействие между аппаратным обеспечением и прикладными программами. Ядро операционной системы выполняет критически важные функции управления ресурсами, включая распределение процессорного времени, организацию работы с памятью и управление устройствами ввода-вывода.

Вспомогательные модули операционной системы расширяют базовую функциональность ядра, предоставляя дополнительные сервисы и интерфейсы для приложений. К таким модулям относятся драйверы устройств, подсистемы управления файлами, сетевые протоколы и другие компоненты. Важно отметить, что в современных ОС прослеживается четкая тенденция к выносу мак-

симального количества функциональности из пространства ядра в пользовательское пространство, что повышает стабильность и безопасность системы в целом.

Привилегированный режим работы ядра

Особенностью архитектуры современных процессоров является поддержка различных уровней привилегий выполнения кода. Ядро операционной системы работает в привилегированном режиме (часто называемом режимом ядра или Ring 0 в архитектуре x86), что предоставляет ему полный доступ ко всем ресурсам системы. Этот режим позволяет ядру выполнять привилегированные команды процессора, напрямую работать с аппаратурой и управлять виртуальной памятью.

Безопасность операционной системы во многом зависит от корректности работы кода, выполняемого в привилегированном режиме. Любая ошибка в этом коде может привести к серьезным последствиям, включая крах всей системы или появление уязвимостей безопасности. Поэтому современные ОС реализуют различные механизмы защиты, такие как проверка целостности драйверов перед их загрузкой, использование аппаратных средств защиты памяти и строгий контроль доступа к системным ресурсам.

Многоуровневая организация операционных систем

Современные операционные системы строятся по принципу многоуровневой организации, где каждый уровень предоставляет определенный набор абстракций и сервисов для вышележащих уровней. Такой подход позволяет добиться четкого разделения функциональности и упрощает разработку и поддержку системы. На нижнем уровне находится аппаратное обеспечение, выше располагается уровень абстракции оборудования, затем ядро ОС, системные библиотеки и, наконец, прикладные программы.

Многослойная архитектура играет важную роль в обеспечении безопасности операционной системы. Каждый уровень может реализовывать собственные механизмы защиты, а четкое разделение функциональности позволяет локализовать потенциальные уязвимости. Например, в системах с микроядерной архитектурой многие традиционно ядерные функции вынесены в пользовательское пространство, что значительно уменьшает возможную поверхность атаки.

Аппаратная независимость и переносимость

Одной из ключевых характеристик современных операционных систем является их способность работать на различном аппаратном обеспечении. Достигается это за счет реализации слоя аппаратных абстракций (HAL – Hardware Abstraction Layer), который скрывает специфику конкретного оборудования за унифицированными интерфейсами. Такой подход позволяет переносить ОС на новые платформы с минимальными изменениями в основном коде системы.

Проблема переносимости тесно связана с вопросами безопасности. Различные аппаратные платформы могут предоставлять разные механизмы защиты, и операционная система должна эффективно использовать их все.

Например, современные процессоры предлагают такие технологии безопасности, как NX-бит (запрет выполнения кода в определенных областях памяти) или аппаратная поддержка виртуализации, и ОС должна уметь работать с этими возможностями.

Микроядерная архитектура и безопасность

Альтернативой традиционной монолитной архитектуре операционных систем является микроядерный подход, при котором в привилегированном режиме выполняется минимальный набор функций, а остальные компоненты работают в пользовательском пространстве. Такая архитектура предлагает существенные преимущества с точки зрения безопасности, так как значительно уменьшает объем кода, выполняемого с максимальными привилегиями.

Микроядерные системы демонстрируют более высокую устойчивость к сбоям и атакам, поскольку ошибка в любом из компонентов пользовательского пространства не может напрямую повлиять на стабильность ядра. Однако такой подход требует тщательной проработки механизмов межпроцессного взаимодействия, которые могут стать потенциальной точкой уязвимости. Современные исследования в области безопасности операционных систем показывают, что микроядерная архитектура может обеспечить более высокий уровень защиты при правильной реализации.

Совместимость и множественные прикладные среды

Современные операционные системы сталкиваются с необходимостью поддержки приложений, разработанных для других платформ и сред. Решение этой задачи достигается различными способами: от полной эмуляции чужой среды до создания совместимых подсистем. Каждый из этих подходов имеет свои последствия для безопасности системы.

Например, подсистема Windows для Linux (WSL) реализует совместимость на уровне системных вызовов, тщательно контролируя взаимодействие между двумя средами. Аналогично, технологии виртуализации позволяют запускать целые операционные системы внутри защищенных сред, обеспечивая высокий уровень изоляции. Разработчики ОС должны тщательно проектировать такие механизмы, чтобы не создавать новых векторов атак при обеспечении совместимости.

Для глубокого понимания архитектуры операционных систем и связанных с ней аспектов безопасности рекомендуется сочетать теоретическое изучение с практической деятельностью. Особое внимание следует уделить анализу исходных кодов открытых операционных систем, таких как Linux или FreeBSD. Лабораторные задания могут включать исследование механизмов системных вызовов, анализ работы планировщика задач или изучение реализации подсистемы виртуальной памяти.

Полезным упражнением будет сравнение архитектурных решений в различных операционных системах. Например, можно проанализировать различия в реализации безопасности между монолитным ядром Linux и микроядерной архитектурой QNX. Также рекомендуется экспериментировать с настройкой параметров безопасности на разных уровнях операционной системы.

Предварительно требуется определить структурную организацию ОС на основе различных программных модулей. Рассмотреть типы ядер ОС.

Требуется рассмотреть **два режима работы ядра ОС:**

− пользовательский режим (user mode),

− привилегированный режим, который также называют режимом ядра (kernel mode), или режимом супервизора (supervisor mode).

Требуется рассмотреть Многослойную структуру ОС (**средства аппаратной поддержки ОС, машинно-зависимые компоненты ОС, менеджеры ресурсов, интерфейс системных вызовов (API)**).

Требуется рассмотреть аппаратную зависимость и переносимость ОС.

Требуется рассмотреть **микроядерную архитектуру**, набор функций микроядра.

При работе над курсовой работой обратите внимание на:

– описание уязвимостей ОС, которые требуется приводить в первой (теоретической) главе.

Литература

1. Окороков, В. А. Безопасность операционных систем / В. А. Окороков. – Санкт-Петербург: Лань, 2024. – 228 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/367472 (дата обращения: 06.12.2024). – ISBN 978-5-507-48297-9. — Текст : электронный (гл. 2).

2. Киренберг, Г. А. Информационная безопасность современных операционных систем: учеб. пособие / Киренберг, Г. А. – Кемерово: КузГТУ им. Т.Ф. Горбачева, 2022. – 138 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/295736 (дата обращения: 06.12.2024). – ISBN 978-5-00137-320-9. – Текст : электронный (гл. 1).

3. Потерпеев, Г. Ю. Безопасность операционных систем: учеб. пособие / Г. Ю. Потерпеев, В. С. Нефедов, А. А. Криулин. – Москва: РТУ МИРЭА, 2021. – 93 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/182416 (дата обращения: 06.12.2024). – ISBN 978-5-7339-1393-3. – Текст : электронный (гл. 2).

4. Зверева, О. М. Операционные системы: учеб. пособие / О. М. Зверева; науч. ред. Л. Г. Доросинский; Уральский федеральный университет им. первого Президента России Б. Н. Ельцина. – Екатеринбург: Издательство Уральского университета, 2020. – 223 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=699030 (дата обращения: 03.11.2024). – ISBN 978-5-7996-3146-8. – Текст : электронный (гл. 10).

Контрольные вопросы

1. Каковы основные функции ядра операционной системы и как они связаны с вопросами безопасности?
2. Почему привилегированный режим работы ядра является одновременно необходимым и потенциально опасным с точки зрения безопасности?
3. Как многоуровневая архитектура операционной системы способствует повышению её защищенности?
4. Какие механизмы используются в современных ОС для обеспечения аппаратной независимости?
5. В чем заключаются основные преимущества микроядерной архитектуры с точки зрения безопасности?
6. Каким образом операционные системы обеспечивают совместимость с приложениями, разработанными для других платформ?
7. Как реализована защита межпроцессного взаимодействия в микроядерных операционных системах?
8. Какие аппаратные технологии современных процессоров наиболее важны для обеспечения безопасности ОС?
9. Как влияет на безопасность системы вынос функциональности из ядра в пользовательское пространство?
10. Какие методы изоляции используются при реализации множественных прикладных сред в одной операционной системе?

Тема 1.3 Управление процессами

Перечень изучаемых вопросов

1. Понятие процесса и потока.
2. Управление процессами и потоками.
3. Алгоритмы планирования.
4. Синхронизация процессов и потоков.

Методические указания к изучению

Современные операционные системы обеспечивают выполнение множества задач через механизм процессов и потоков, играющий ключевую роль в распределении ресурсов и обеспечении безопасности. Процесс представляет собой экземпляр выполняемой программы, обладающий собственным виртуальным адресным пространством, набором регистров и системных ресурсов. Поток (нить выполнения) — это минимальная единица обработки, разделяющая с родительским процессом память и файловые дескрипторы. Управление этими сущностями требует сложных алгоритмов и механизмов синхронизации, от корректности которых зависит стабильность и защищенность системы.

Понятие процесса и потока

Процесс формируется при запуске программы и включает код, данные и состояние вычислительной среды. Каждый процесс изолирован от других средствами операционной системы, что предотвращает несанкционированный

доступ к его памяти. Потоки существуют внутри процесса и позволяют реализовать параллельное выполнение задач, однако их общий доступ к ресурсам создает риски состояния гонки (race condition). Например, два потока, одновременно изменяющие одну переменную, могут привести к неопределенному поведению программы.

С точки зрения безопасности, изоляция процессов критически важна для предотвращения атак типа **переполнения буфера** или **инъекции кода**. Операционные системы используют аппаратную поддержку виртуальной памяти (MMU) и механизмы разграничения прав (DAC/MAC), чтобы ограничить влияние скомпрометированного процесса на всю систему.

Управление процессами и потоками

Создание процесса включает выделение ресурсов, инициализацию структур данных (PCB — Process Control Block) и регистрацию в планировщике задач. Переключение между процессами (контекстный switch) требует сохранения состояния регистров и обновления указателей памяти, что влечет накладные расходы. Потоки, будучи легковесными, создаются быстрее, так как разделяют ресурсы процесса.

Операционная система управляет жизненным циклом процессов через системные вызовы: fork() в Unix-системах создает дочерний процесс, а CreateProcess() в Windows формирует новый независимый экземпляр. Для потоков используются API-интерфейсы вроде POSIX Threads (pthread) или Windows Threads.

Безопасность управления обеспечивается:

a. Проверкой прав доступа при создании процессов (например, через мандаты в SELinux).

b. Ограничением числа потоков для предотвращения исчерпания ресурсов (атаки типа fork bomb).

c. Механизмами контроля целостности исполняемых файлов (цифровые подписи).

Алгоритмы планирования

Планировщик задач определяет порядок выполнения процессов и потоков, основываясь на стратегиях, которые балансируют производительность и справедливость.

a. **First-Come, First-Served (FCFS)** – простейший алгоритм, уязвимый к "голоданию" длительных задач.

b. **Shortest Job Next (SJN)** – оптимизирует общее время выполнения, но требует знания длительности задач.

c. **Round Robin (RR)** – циклическое распределение квантов времени, обеспечивающее предсказуемость.

d. **Приоритетное планирование** – задачи выполняются в соответствии с назначенными приоритетами.

В многопоточных системах планирование усложняется необходимостью синхронизации. Например, в реальном времени (RTOS) алгоритмы должны га-

рантировать соблюдение временных ограничений, а ошибки планирования могут привести к отказам в критических системах.

Синхронизация процессов и потоков

Совместный доступ к ресурсам требует механизмов синхронизации для предотвращения конфликтов.

a. **Мьютексы** – блокируют доступ к ресурсу, пока один поток его использует.

b. **Семафоры** – счетчики, ограничивающие число одновременных обращений.

c. **Условные переменные** – позволяют потокам ожидать выполнения условий.

Неверная синхронизация ведет к **взаимным блокировкам** (deadlock), когда процессы ожидают ресурсы, занятые друг другом. Для предотвращения deadlock используются алгоритмы обнаружения (Banker's algorithm) или методы избегания (иерархия ресурсов).

Безопасность синхронизации зависит от корректной реализации примитивов. Например, атака на неблокирующий мьютекс может привести к утечке данных.

При работе над курсовой работой обратите внимание на:

– описание процессов ОС, которые требуется приводить в первой (теоретической) главе.

Литература

1. Окороков, В. А. Безопасность операционных систем / В. А. Окороков. – Санкт-Петербург: Лань, 2024. – 228 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/367472 (дата обращения: 06.12.2024). – ISBN 978-5-507-48297-9. – Текст : электронный (гл. 2, 3).

2. Киренберг, Г. А. Информационная безопасность современных операционных систем: учеб. пособие / Киренберг, Г. А. – Кемерово: КузГТУ им. Т.Ф. Горбачева, 2022. – 138 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/295736 (дата обращения: 06.12.2024). — ISBN 978-5-00137-320-9. — Текст : электронный (гл. 1).

3. Потерпеев, Г. Ю. Безопасность операционных систем: учеб. пособие / Г. Ю. Потерпеев, В. С. Нефедов, А. А. Криулин. – Москва: РТУ МИРЭА, 2021. – 93 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/182416 (дата обращения: 06.12.2024). – ISBN 978-5-7339-1393-3. – Текст : электронный (гл. 2).

4. Зверева, О. М. Операционные системы: учеб. пособие / О. М. Зверева; науч. ред. Л. Г. Доросинский; Уральский федеральный университет им. первого Президента России Б. Н. Ельцина. – Екатеринбург: Издательство

Уральского университета, 2020. – 223 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=699030 (дата обращения: 03.11.2024). – ISBN 978-5-7996-3146-8. – Текст : электронный (гл. 3).

Контрольные вопросы

1. Объясните различия между процессом и потоком с точки зрения ресурсов и безопасности.
2. Какие структуры данных использует ОС для управления процессами?
3. Как алгоритм Round Robin предотвращает «голодание» задач?
4. Опишите сценарий взаимной блокировки (deadlock) и методы её разрешения.
5. Почему приоритетное планирование может снижать безопасность системы?
6. Какие аппаратные механизмы процессора участвуют в изоляции процессов?
7. Как операционная система обнаруживает попытки несанкционированного создания процессов?
8. Объясните роль мьютексов в предотвращении состояния гонки.

Тема 1.4 Управление памятью

Перечень изучаемых вопросов

1. Иерархия памяти.
2. Управление памятью.
3. Типы адресации.
4. Виртуальная память и свопинг.
5. Алгоритмы управления памятью.

Методические указания к изучению:

Требуется обратить внимание на функции ОС по управлению памятью в мультипрограммной системе, типы адресации, способы структуризации виртуального адресного пространства, виртуальную память и свопинг, алгоритмы управления памятью, алгоритмы управления памятью. Требуется разобрать:

А. Алгоритмы управления памятью без использования механизма виртуальной памяти:

1. Распределение памяти фиксированными разделами.
2. Распределение памяти динамическими разделами.
3. Перемещаемые разделы.

Б. Алгоритмы управления памятью с использованием виртуальной памяти:

1. Страничное распределение.
2. Сегментное распределение.

3. Сегментно-страничное распределение.

Управление памятью – одна из ключевых функций операционных систем, определяющая не только производительность, но и безопасность вычислительных процессов. Эффективное распределение и защита памяти предотвращают утечки данных, атаки типа переполнения буфера и несанкционированный доступ к критическим ресурсам. Современные ОС реализуют сложные механизмы работы с памятью, сочетающие аппаратную поддержку и программные алгоритмы, что требует глубокого понимания их архитектуры и уязвимостей.

Иерархия памяти

Современные вычислительные системы используют многоуровневую иерархию памяти, где каждый уровень компенсирует ограничения предыдущего. На вершине иерархии находятся регистры процессора и кэш-память (L1, L2, L3), обеспечивающие минимальные задержки доступа. Далее следует оперативная память (ОЗУ), выступающая основным рабочим пространством для процессов. Завершают цепочку устройства долговременного хранения (HDD, SSD), используемые для хранения данных и расширения ОЗУ через механизм свопинга.

С точки зрения безопасности, иерархия памяти создает риски утечек данных через побочные каналы. Например, атаки **Spectre** и **Meltdown** эксплуатируют спекулятивное выполнение инструкций процессором и чтение данных из кэша. Это подчеркивает важность аппаратных и программных механизмов изоляции, таких как разделение кэша между процессами или использование технологий вроде Intel SGX.

Управление памятью

Операционная система отвечает за распределение памяти между процессами, отслеживание свободных и занятых областей, а также защиту данных. Каждый процесс получает собственное виртуальное адресное пространство, изолированное от других процессов. Это достигается через трансляцию виртуальных адресов в физические с помощью **Memory Management Unit (MMU)** – аппаратного компонента процессора.

Ключевые задачи управления памятью:

1. **Изоляция процессов** – предотвращение доступа к памяти других приложений.

2. **Динамическое выделение** – распределение памяти по запросу (например, через malloc() или new).

3. **Защита целостности** – обнаружение повреждений памяти (переполнение буфера, use-after-free).

Нарушение этих механизмов ведет к уязвимостям. Например, **переполнение буфера** позволяет злоумышленнику перезаписать соседние участки памяти и внедрить вредоносный код. Для противодействия используются технологии вроде **ASLR** (рандомизация адресного пространства) и **DEP** (запрет выполнения кода в областях данных).

Типы адресации

Адресация памяти делится на три типа:

a. **Физическая** – прямой доступ к ячейкам ОЗУ (используется ядром ОС).

b. **Виртуальная** – абстракция, предоставляемая процессам для изоляции.

c. **Логическая** – адреса внутри сегментов памяти (актуально для устаревших архитектур x86 в реальном режиме).

MMU преобразует виртуальные адреса в физические через таблицы страниц, хранящие права доступа (чтение, запись, выполнение). Например, в Linux каждая запись в таблице страниц содержит флаги RWX, которые запрещают выполнение кода в стеке или куче, предотвращая некоторые типы эксплойтов.

Уязвимости в трансляции адресов (например, **CVE-2018-18281** в ядре Linux) позволяют злоумышленникам обходить защиту и получать доступ к памяти ядра. Это требует своевременного обновления ОС и использования аппаратных расширений вроде **ARM Memory Tagging**.

Виртуальная память и свопинг

Виртуальная память расширяет доступное адресное пространство за счет использования диска. Когда физической памяти недостаточно, ОС перемещает неактивные страницы в **своп-файл** (pagefile.sys в Windows, swap в Linux). Этот процесс, называемый свопингом, критически важен для многозадачности, но создает риски: данные в свопе могут быть восстановлены после перезагрузки, что угрожает конфиденциальности.

Для защиты информации в свопе применяется:

- Шифрование всего диска (BitLocker, LUKS).
- Отключение свопинга в системах с жесткими требованиями безопасности.
- Использование энергонезависимой памяти (Persistent Memory), исключающей необходимость свопинга.

Атаки типа **Cold Boot** демонстрируют, что злоумышленники могут извлекать данные из ОЗУ даже после выключения системы. Это делает важным очистку чувствительных данных из памяти перед завершением работы.

Алгоритмы управления памятью

Алгоритмы управления памятью определяют, какие страницы остаются в ОЗУ, а какие выгружаются на диск. Наиболее распространены:

a. **FIFO (First-In, First-Out)** – вытеснение старейших страниц.

b. **LRU (Least Recently Used)** – удаление страниц, к которым дольше не обращались.

c. **Оптимальный алгоритм** – теоретическая модель, вытесняющая страницу, которая не понадобится дольше всего.

Эффективность алгоритмов влияет на устойчивость к атакам. Например, **атаки на кэш** могут искусственно увеличить количество промахов, замедлив систему. В реальных ОС (например, Linux) используется модифицированный LRU с учетом частоты обращений и «возраста» страниц.

Рекомендации

1. **Изучение архитектуры памяти:** Анализ структур данных ядра (например, struct mm_struct в Linux).
2. **Практика с инструментами:** Использование Valgrind для обнаружения утечек памяти, GDB для отладки сегментации.
3. **Эксперименты с свопингом:** Настройка размера своп-файла в Linux (swapon/swapoff) и анализ производительности.
4. **Исследование защитных механизмов:** Тестирование ASLR и DEP на примере эксплойтов (Metasploit Framework).

При работе над курсовой работой обратите внимание на:

– описание памяти ОС, которые требуется приводить в первой (теоретической) главе.

Литература

1. Окороков, В. А. Безопасность операционных систем / В. А. Окороков. – Санкт-Петербург: Лань, 2024. – 228 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/367472 (дата обращения: 06.12.2024). – ISBN 978-5-507-48297-9. – Текст : электронный (гл. 3–5).

2. Киренберг, Г. А. Информационная безопасность современных операционных систем: учебное пособие / Киренберг, Г. А. – Кемерово: КузГТУ имени Т. Ф. Горбачева, 2022. – 138 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/295736 (дата обращения: 06.12.2024). – ISBN 978-5-00137-320-9. – Текст : электронный (гл. 1).

3. Потерпеев, Г. Ю. Безопасность операционных систем: учеб. пособие / Г. Ю. Потерпеев, В. С. Нефедов, А. А. Криулин. – Москва: РТУ МИРЭА, 2021. – 93 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/182416 (дата обращения: 06.12.2024). – ISBN 978-5-7339-1393-3. – Текст : электронный (гл. 2, с. 31–39).

4. Зверева, О. М. Операционные системы: учеб. пособие / О. М. Зверева; науч. ред. Л. Г. Доросинский; Уральский федеральный университет им. первого Президента России Б. Н. Ельцина. – Екатеринбург: Издательство Уральского университета, 2020. – 223 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=699030 (дата обращения: 03.11.2024). – ISBN 978-5-7996-3146-8. – Текст : электронный (гл. 5).

Контрольные вопросы

1. Приведите классификацию распределения памяти.
2. Приведите классификации алгоритмов управления памятью без использования механизма виртуальной памяти.
3. Дайте определение синхронизации.
4. Приведите классификацию алгоритмов управления памятью с

использованием виртуальной памятью

5. Объясните роль MMU в обеспечении изоляции процессов.

6. Как ASLR предотвращает атаки переполнения буфера?

7. Какие риски безопасности связаны с использованием свопинга?

8. Как повреждение таблиц страниц может привести к уязвимостям?

9. Какие методы шифрования применяются для защиты данных в своп-файле?

Тема 1.5 Прерывания

Перечень изучаемых вопросов

1. Понятие прерывания.
2. Механизм прерываний.
3. Функции централизованного диспетчера прерываний.
4. Процедуры обработки прерываний, вызванные из текущего процесса.
5. Системные вызовы.

Методические указания к изучению

Требуется обратить внимание на**:** три больших класса прерываний, (Программные прерывания, Внешние прерывания, Внутренние прерывания), два основных способа выполнения прерывания (Опрашиваемый (polled), Векторный (vectored)), приоритезацию и маскирование прерываний, двухуровневый механизм планирования работ при управлении прерываниями, системные вызовы в синхронном или асинхронном режимах.

Прерывания представляют собой фундаментальный механизм взаимодействия между аппаратным обеспечением, программным обеспечением и операционной системой. Они позволяют процессору реагировать на внешние события (например, нажатие клавиши) или внутренние исключения (деление на ноль) без постоянного опроса устройств. Однако некорректная обработка прерываний может стать источником уязвимостей, таких как перехват управления системой или несанкционированный доступ к данным. Понимание архитектуры прерываний критически важно для обеспечения безопасности операционной системы.

Понятие прерывания

Прерывание – это сигнал, инициируемый аппаратурой или программным обеспечением, который требует немедленного внимания процессора. Аппаратные прерывания генерируются устройствами (клавиатура, таймер, сетевой адаптер) для уведомления ОС о завершении операции или возникновении ошибки. Программные прерывания вызываются исполняемым кодом для запроса сервисов ядра (системные вызовы) или обработки исключений (ошибка доступа к памяти).

С точки зрения безопасности, прерывания создают «окна уязвимости», когда контекст выполнения переключается с пользовательского режима на ре-

жим ядра. Злоумышленники могут эксплуатировать это, внедряя вредоносные обработчики прерываний или манипулируя векторами прерываний. Например, атака **IOMMU CVE-2017-2636** в ядре Linux позволяла перехватывать прерывания от периферийных устройств.

Механизм прерываний

Обработка прерываний начинается с сохранения текущего контекста выполнения (регистров, указателя инструкций) в стеке. Процессор переключается в привилегированный режим (Ring 0) и выполняет код обработчика прерывания, адрес которого хранится в таблице векторов прерываний (IDT — Interrupt Descriptor Table). Современные системы используют расширенные контроллеры прерываний (APIC), которые маршрутизируют сигналы между ядрами процессора.

Безопасность механизма прерываний обеспечивается:

1. **Защитой таблицы векторов** – только ядро ОС имеет право изменять IDT.

2. **Верификацией обработчиков** – цифровые подписи драйверов в Windows предотвращают внедрение неавторизованного кода.

3. **Изоляцией контекста** – обработчики выполняются в отдельном стеке ядра, чтобы избежать переполнения пользовательского стека.

Функции централизованного диспетчера прерываний

Централизованный диспетчер прерываний (например, **IRQ Manager** в Windows или **irqchip** в Linux) отвечает за распределение прерываний между процессорами, управление приоритетами и балансировку нагрузки. Он предотвращает конфликты при одновременных запросах от устройств и обеспечивает своевременное выполнение критических задач (например, обработки сетевых пакетов).

В многопроцессорных системах диспетчер использует алгоритмы вроде **IRQ affinity**, чтобы закрепить определенные прерывания за конкретными ядрами. Это снижает задержки, но создает риски неравномерной нагрузки и атак типа **DoS** через исчерпание ресурсов одного ядра. Для защиты применяется динамическое перераспределение прерываний и ограничение их частоты.

Процедуры обработки прерываний, вызванные из текущего процесса

Когда прерывание возникает во время выполнения процесса, процессор сохраняет его состояние (регистры, флаги) и передает управление обработчику. После завершения обработки восстанавливается контекст, и процесс продолжает работу. Однако если прерывание требует длительной обработки (например, загрузка данных с диска), ОС может перепланировать задачи, переключившись на другой процесс.

Опасность возникает, когда обработчик прерывания обращается к ресурсам, используемым текущим процессом. Например, **атака на двойное извлечение (double fetch)** эксплуатирует гонку между проверкой прав доступа в ядре и фактическим использованием данных. Для предотвращения таких уязвимостей используются атомарные операции и блокировки.

Системные вызовы

Системные вызовы – это программные прерывания, через которые приложения запрашивают сервисы ядра (открытие файла, создание процесса). В архитектуре x86 для этого используется инструкция int 0x80 или syscall, передающая управление заранее определенному обработчику в ядре.

Безопасность системных вызовов обеспечивается:

1. **Проверкой параметров** – ядро валидирует указатели и размеры буферов, переданные из пользовательского пространства.

2. **Изоляцией адресных пространств** – данные процесса не доступны ядру напрямую, копирование выполняется через специальные функции (copy_from_user).

3. **Аудитом вызовов** – системы вроде SELinux или AppArmor отслеживают подозрительные операции.

Уязвимости в системных вызовах (например, **CVE-2021-4034** в pkexec) позволяют повысить привилегии или выполнить произвольный код.

При работе над курсовой работой обратите внимание на:

– описание прерываний ОС, которые требуется приводить в первой (теоретической) главе.

Литература

1. Окороков, В. А. Безопасность операционных систем / В. А. Окороков. – Санкт-Петербург: Лань, 2024. – 228 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/367472 (дата обращения: 06.12.2024). – ISBN 978-5-507-48297-9. – Текст : электронный (гл. 4–6).

2. Киренберг, Г. А. Информационная безопасность современных операционных систем: учеб. пособие / Киренберг, Г. А. – Кемерово: КузГТУ им. Т.Ф. Горбачева, 2022. – 138 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/295736 (дата обращения: 06.12.2024). – ISBN 978-5-00137-320-9. – Текст : электронный (гл. 1).

3. Потерпеев, Г. Ю. Безопасность операционных систем: учеб. пособие / Г. Ю. Потерпеев, В. С. Нефедов, А. А. Криулин. – Москва: РТУ МИРЭА, 2021. – 93 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/182416 (дата обращения: 06.12.2024). – ISBN 978-5-7339-1393-3. – Текст : электронный (гл. 2).

4. Зверева, О. М. Операционные системы: учеб. пособие / О. М. Зверева; науч. ред. Л. Г. Доросинский; Уральский федеральный университет им. первого Президента России Б. Н. Ельцина. – Екатеринбург: Издательство Уральского университета, 2020. – 223 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=699030 (дата обращения: 03.11.2024). – ISBN 978-5-7996-3146-8. – Текст : электронный (гл. 6).

Контрольные вопросы

1. Какие типы прерываний существуют и как они влияют на безопасность ОС?
2. Объясните роль таблицы векторов прерываний (IDT) в архитектуре x86.
3. Как централизованный диспетчер прерываний предотвращает конфликты между устройствами?
4. Почему сохранение контекста процесса критически важно при обработке прерываний?
5. Какие уязвимости могут возникнуть при некорректной обработке системных вызовов?
6. Как инструкция syscall отличается от int 0x80 с точки зрения производительности и безопасности?
7. Какие механизмы защищают обработчики прерываний от подмены злоумышленником?
8. Почему атаки типа «двойное извлечение» эффективны против ядра ОС?
9. Как SMAP предотвращает доступ ядра к пользовательской памяти?
10. Какие методы аудита используются для отслеживания подозрительных системных вызовов?

Тема 1.6 Управление вводом-выводом

Перечень изучаемых вопросов

1. Организация взаимодействия ос с устройствами ввода-вывода.
2. Многослойная модель подсистемы ввода-вывода.
3. Менеджеры ввода-вывода.
4. Драйверы устройств.

Методические указания к изучению

Управление вводом-выводом (I/O) представляет собой один из ключевых механизмов операционных систем, обеспечивающий взаимодействие между программным обеспечением и аппаратными устройствами. Этот процесс охватывает не только передачу данных, но и управление ресурсами, синхронизацию операций и защиту от несанкционированного доступа. В условиях роста сложности кибератак, направленных на периферийные устройства и драйверы, безопасность подсистемы ввода-вывода становится критической для обеспечения целостности и конфиденциальности данных. Уязвимости в этой области, такие как некорректная обработка прерываний или недостаточная изоляция драйверов, могут быть использованы для внедрения вредоносного кода, перехвата информации или нарушения работы системы. Данная статья раскрывает архитектурные и программные аспекты управления I/O, фокусируясь на методах минимизации рисков и методологии изучения этих механизмов.

Организация взаимодействия ОС с устройствами ввода-вывода
Взаимодействие операционной системы с устройствами ввода-вывода начинается на этапе инициализации оборудования. Во время загрузки ОС ядро идентифицирует подключенные устройства через шинные интерфейсы (PCI, USB, SATA) и загружает соответствующие драйверы. Современные ОС, такие как Windows и Linux, используют технологию Plug-and-Play (PnP), позволяющую динамически обнаруживать и настраивать устройства без необходимости ручного вмешательства. Однако эта автоматизация несет риски: злоумышленник может подключить вредоносное устройство (например, Rubber Ducky, имитирующее клавиатуру), которое будет воспринято системой как легитимное. Для противодействия таким угрозам применяются механизмы контроля целостности драйверов. Например, в Windows функция *Secure Boot* и *Driver Signature Enforcement* блокируют загрузку драйверов без цифровой подписи, сертифицированной Microsoft. В Linux аналогичные задачи решаются через модуль *dm-verity*, проверяющий целостность драйверов на основе хэш-сумм.

Важным аспектом безопасности является управление правами доступа. Операционные системы используют модели *Mandatory Access Control (MAC),* такие как SELinux или AppArmor, которые ограничивают взаимодействие процессов с устройствами на уровне политик. Например, политика может запрещать веб-браузеру доступ к USB-портам, чтобы предотвратить копирование данных на съемные носители. В macOS для управления устройствами используется фреймворк *Endpoint Security*, который позволяет отслеживать и блокировать подключение неавторизованных девайсов на уровне ядра.

Многослойная модель подсистемы ввода-вывода

Архитектура подсистемы ввода-вывода строится на многоуровневой абстракции, что обеспечивает модульность и снижает риски эксплуатации уязвимостей. Верхний уровень включает интерфейсы прикладного программирования (API), такие как системные вызовы read() и write() в Unix-подобных системах или Win32 API в Windows. Эти интерфейсы предоставляют приложениям унифицированный доступ к устройствам, скрывая аппаратные особенности. Например, приложение, записывающее данные на жесткий диск, не должно знать, используется ли магнитная запись или флеш-память.

Средний уровень подсистемы представлен менеджерами ввода-вывода, которые отвечают за буферизацию, кэширование и планирование операций. Менеджеры оптимизируют производительность, минимизируя количество обращений к физическим устройствам. Однако здесь возникают риски, связанные с утечкой данных через кэш. Для их устранения применяются методы шифрования буферов. Например, в современных файловых системах (e.g., ZFS) используется *Transparent Encryption*, автоматически шифрующий данные перед записью в кэш.

Нижний уровень включает драйверы устройств и непосредственное взаимодействие с аппаратурой. Драйверы транслируют логические запросы в физические команды, понятные контроллерам устройств. Безопасность на этом уровне обеспечивается за счет изоляции драйверов. В микроядерных ОС

(например, QNX) драйверы выполняются в пользовательском пространстве, что предотвращает крах всей системы при их сбое. В Linux для изоляции драйверов используется фреймворк *vfio* (Virtual Function I/O), позволяющий запускать их внутри виртуальных машин.

Особое внимание уделяется защите от атак через *DMA (Direct Memory Access)*. Устройства с DMA могут напрямую обращаться к памяти, минуя процессор, что создает угрозу чтения или модификации данных. Для блокировки таких атак применяются технологии *IOMMU* (Input-Output Memory Management Unit), которые транслируют физические адреса в виртуальные и контролируют права доступа. Например, в процессорах AMD технология *AMD-Vi*, а в Intel — *VT-d*.

Менеджеры ввода-вывода и их роль в безопасности

Менеджеры ввода-вывода выполняют роль посредников между приложениями и драйверами, обеспечивая синхронизацию и управление ресурсами. Одной из их ключевых задач является предотвращение *состояний гонки (race conditions)*. Например, при одновременной записи двух процессов в один файл менеджер блокирует доступ к сектору диска до завершения первой операции. В распределенных системах для этого используются алгоритмы типа *Lamport Timestamps* или *Distributed Locks*.

Еще одной функцией менеджеров является контроль целостности данных. При передаче информации между устройствами применяются механизмы проверки на основе *CRC (Cyclic Redundancy Check)* или *хеш-функций*. Например, при чтении данных с сетевой карты менеджер может проверять контрольные суммы пакетов, чтобы исключить их подмену. В системах хранения данных используются технологии *RAID* с зеркалированием и четностью, позволяющие восстанавливать информацию при частичном повреждении.

В контексте безопасности менеджеры также реализуют политики *QoS (Quality of Service)*, ограничивающие пропускную способность устройств для предотвращения *DDoS-атак*. Например, сетевой менеджер может ограничить количество запросов к веб-серверу с одного IP-адреса.

Драйверы устройств: безопасность на уровне ядра

Драйверы, работающие в пространстве ядра, представляют повышенную угрозу из-за их привилегированного доступа к ресурсам системы. Уязвимости в драйверах, такие как переполнение буфера или некорректная обработка исключений, позволяют злоумышленникам выполнить произвольный код с правами ядра. Классическим примером является атака *Blue Pill*, использующая уязвимости драйверов виртуализации для внедрения гипервизора-руткита.

Для минимизации рисков применяются следующие стратегии:

1. **Верификация драйверов**: Современные ОС требуют цифровой подписи драйверов. В Windows для этого используется *Windows Hardware Compatibility Program*, а в Linux — модульная система подписей, основанная на сертификатах.

2. **Изоляция драйверов**: Технологии типа *User-Mode Driver Framework (UMDF)* в Windows позволяют запускать драйверы в пользовательском пространстве. В Linux аналогичную роль играет *usermode-helper API*.

3. **Мониторинг активности**: Инструменты вроде *eBPF (Extended Berkeley Packet Filter)* в Linux позволяют отслеживать системные вызовы драйверов и блокировать подозрительные операции.

4. **Защита памяти**: Механизмы *DEP (Data Execution Prevention)* и *ASLR (Address Space Layout Randomization)* препятствуют эксплуатации уязвимостей, делая адреса исполняемого кода непредсказуемыми.

Примером успешной защиты является исправление уязвимости *EternalBlue*, эксплуатировавшей ошибку в драйвере SMBv1. Патч Microsoft исключил переполнение буфера, а также добавил дополнительные проверки целостности пакетов.

При работе над курсовой работой обратите внимание на:

– описание особенностей ввода-вывода ОС, которые требуется приводить в первой (теоретической) главе (в соответствии с темой курсовой работы).

Литература

1. Окороков, В. А. Безопасность операционных систем / В. А. Окороков. – Санкт-Петербург: Лань, 2024. – 228 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/367472 (дата обращения: 06.12.2024). – ISBN 978-5-507-48297-9. – Текст : электронный (гл. 4–6).

2. Киренберг, Г. А. Информационная безопасность современных операционных систем: учеб. пособие / Киренберг, Г. А. – Кемерово: КузГТУ им. Т.Ф. Горбачева, 2022. – 138 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/295736 (дата обращения: 06.12.2024). – ISBN 978-5-00137-320-9. – Текст : электронный (гл. 1).

3. Потерпеев, Г. Ю. Безопасность операционных систем: учеб. пособие / Г. Ю. Потерпеев, В. С. Нефедов, А. А. Криулин. – Москва: РТУ МИРЭА, 2021. – 93 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/182416 (дата обращения: 06.12.2024). – ISBN 978-5-7339-1393-3. – Текст : электронный (гл. 2).

4. Зверева, О. М. Операционные системы: учеб. пособие / О. М. Зверева; науч. ред. Л. Г. Доросинский; Уральский федеральный университет им. первого Президента России Б. Н. Ельцина. – Екатеринбург: Издательство Уральского университета, 2020. – 223 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=699030 (дата обращения: 03.11.2024). – ISBN 978-5-7996-3146-8. – Текст : электронный (гл. 6).

Контрольные вопросы

1. Приведите особенности ввода-вывода по отношению к программному модулю, запросившему операцию.
2. Приведите особенности четырех слоев системы ввода-вывода.
3. Приведите особенности верхнего слоя менеджера.
4. Приведите особенности нижнего слоя менеджера.

Тема 1.7 Файловая система

Перечень изучаемых вопросов

1. Организация файловой систем (ФС), типы файлов.
2. Иерархическая структура файловой системы.
3. понятие о монтировании.
4. Физическая организация файловой системы.
5. Общая модель файловой системы.
6. Понятие о журналируемых файловых системах.
7. Физическая организация и адресация в файле.
8. Файловая система FAT.

Методические указания к изучению. Рекомендуется изучить устройство подсистемы ввода-вывода в различных ОС. Например, сравнить реализацию I/O в монолитных (Windows) и микроядерных (QNX) системах. Полезным ресурсом станет документация Microsoft *Windows Driver Kit* и исходные коды Linux на GitHub.

Рассмотреть следующие аспекты темы: основные цели использования файла, основные функции ФС, обычные файлы, специальные файлы, файлы-каталоги, параметры прав доступа, два основных подхода к определению прав доступа, понятие о монтировании, общая модель файловой системы (на логическом уровне, на физическом уровне), размещение файла в виде связанного списка кластеров дисковой памяти, непрерывное размещение файла, использование связанного списка индексов, перечисление номеров кластеров, занимаемых этим файлом, Таблица FAT (значения индексного указателя, размер таблицы FAT и разрядность используемых в ней индексных указателей определяется количеством кластеров в области данных, метод хранения адресной информации о файлах, ограничения FAT в Windows), Файловая система exFAT, Файловая система NTFS (основными отличительными свойствами NTFS, главная таблица файлов MFT (Master File Table), структура тома NTFS, записи о системных файлах NTFS в MFT, структура файлов NTFS), Файловая система Ext (логическая организация файловой системы ext, структурная организация файловой системы ext, система адресации данных в файловой системе ext, три режима журналирования).

Безопасность подсистемы ввода-вывода операционных систем требует комплексного подхода, сочетающего аппаратную защиту, строгую

верификацию драйверов и многоуровневую изоляцию компонентов. Изучение этой темы должно включать не только теорию, но и практику анализа уязвимостей, разработки безопасного кода и работы с инструментами мониторинга. Понимание механизмов управления I/O позволяет создавать системы, устойчивые как к классическим угрозам (переполнение буфера), так и к современным атакам (DMA, BadUSB). Это направление остается критически важным в условиях роста числа устройств Интернета вещей (IoT) и увеличения поверхностей атаки в корпоративных сетях.

При работе над курсовой работой обратите внимание на:

– описание особенностей ФС ОС, которые требуется приводить в первой (теоретической) главе (в соответствии с темой курсовой работы).

Литература

1. Зверева, О. М. Операционные системы: учеб. пособие / О. М. Зверева; науч. ред. Л. Г. Доросинский; Уральский федеральный университет им. первого Президента России Б. Н. Ельцина. – Екатеринбург: Издательство Уральского университета, 2020. – 223 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=699030 (дата обращения: 03.11.2024). – ISBN 978-5-7996-3146-8. – Текст : электронный (гл. 7).

Контрольные вопросы

1. Приведите особенности ФС ОС.
2. Приведите особенности ФС FAT
3. Приведите особенности ФС NTFS.
4. Приведите особенности ФС ext.
5. Какие механизмы в Windows и Linux предотвращают загрузку неподписанных драйверов и как они влияют на безопасность?
6. Объясните принцип работы IOMMU и его роль в предотвращении DMA-атак. Приведите пример использования в современных процессорах.
7. Почему микроядерные архитектуры считаются более безопасными для изоляции драйверов? Сравните с монолитным ядром.
8. Как состояние гонки при доступе к устройству может быть использовано для атаки? Опишите метод синхронизации, предотвращающий эту угрозу.
9. Какие риски возникают при использовании технологии Plug-and-Play и как они минимизируются?
10. Опишите сценарий атаки через уязвимый драйвер сетевого адаптера. Какие инструменты можно использовать для обнаружения такой атаки?
11. Как механизмы шифрования на уровне менеджеров ввода-вывода (например, в ZFS) защищают данные в кэше?
12. Какие этические и технические проблемы возникают при разработке драйверов с открытым исходным кодом?

Раздел 2. Подсистемы защиты операционной системы

Тема 2.1 Основные функции подсистемы защиты операционной системы

Перечень изучаемых вопросов

1. Политика изолированной программной среды**.**
2. Политика контроля информационных потоков**.**
3. Политика контроля прав доступа**.**

Методические указания к изучению

С традиционной точки зрения средства управления доступом позволяют специфицировать и контролировать действия, которые субъекты (пользователи и процессы) могут выполнять над объектами (информацией и другими компьютерными ресурсами). В данном разделе речь пойдет о логическом управлении доступом, которое, в отличие от физического, реализуется программными средствами.

Рассмотрим формальную постановку задачи в традиционной трактовке. Имеется совокупность субъектов и набор объектов. Задача логического управления доступом состоит в том, чтобы для каждой пары «субъект-объект» определить множество допустимых операций и контролировать выполнение установленного порядка.

Отношение «субъекты-объекты» можно представить в виде матрицы доступа, в строках которой перечислены субъекты, в столбцах - объекты, а в клетках, расположенных на пересечении строк и столбцов, записаны дополнительные условия (например, время и место действия) и разрешенные виды доступа.

Матрицу доступа, ввиду ее разреженности, неразумно хранить в виде двухмерного массива. Обычно ее хранят по столбцам, т. е. для каждого объекта поддерживается список «допущенных» субъектов вместе с их правами.

Списки доступа – исключительно гибкое средство. С их помощью легко выполнить требование о гранулярности прав с точностью до пользователя. Посредством списков несложно добавить права или явным образом запретить доступ (например, чтобы наказать нескольких членов группы пользователей). Безусловно, списки являются лучшим средством произвольного управления доступом.

Подавляющее большинство операционных систем и систем управления базами данных реализуют именно произвольное управление доступом. Основное достоинство произвольного управления – гибкость.

Политика изолированной программной среды реализуется путем изоляции субъектов системы друг от друга и путем контроля порождения новых субъектов таким образом, чтобы в системы могли активизироваться субъекты только из определенного списка.

Политика контроля информационных потоков основана на разделении всех возможных информационных потоков между объектами системы на 2 непересекающихся множества: благоприятные и неблагоприятные информационные потоки. Цель данной политики – обеспечить невозможность возникновения в системе неблагоприятных информационных потоков.

Подсистема защиты операционной системы (ОС) служит центральным элементом обеспечения информационной безопасности, формируя барьеры между потенциальными угрозами и критически важными ресурсами. Её архитектура базируется на трёх фундаментальных политиках: изолированной программной среды, контроля информационных потоков и контроля прав доступа. Эти политики не только предотвращают несанкционированный доступ, но и минимизируют последствия эксплуатации уязвимостей, будь то человеческие ошибки или целенаправленные кибератаки. В условиях роста сложности угроз, таких как атаки типа zero-day или целевые APT-кампании, понимание механизмов защиты ОС становится ключевым для разработки устойчивых систем. Данная статья детально исследует принципы реализации этих политик, их взаимосвязь и методику изучения.

Политика изолированной программной среды

Изоляция в операционных системах – это стратегия, направленная на ограничение взаимодействия между компонентами системы для предотвращения горизонтального перемещения угроз. Её реализация охватывает несколько уровней:

1. **Аппаратная изоляция**: Современные процессоры предоставляют функции виртуализации (Intel VT-x, AMD-V) и защиты памяти (NX bit, SMAP), которые разделяют выполнение кода ядра и пользовательских процессов. Например, технология NX (No Execute) помечает определённые области памяти как неисполняемые, блокируя атаки через переполнение буфера.

2. **Виртуальная память**: Каждый процесс работает в собственном виртуальном адресном пространстве, транслируемом в физические адреса через таблицы страниц. Это предотвращает чтение или модификацию памяти других процессов. В Linux для усиления изоляции используется механизм KASLR (Kernel Address Space Layout Randomization), случайным образом распределяющий адреса ядра, что усложняет эксплуатацию уязвимостей.

3. **Контейнеризация и микроядра**: Технологии вроде Docker используют пространства имён (namespaces) и контрольные группы (cgroups) для изоляции файловых систем, сетевых интерфейсов и ресурсов. Микроядерные ОС (например, QNX) выполняют драйверы и сервисы в пользовательском пространстве, снижая риск краха всей системы из-за ошибки в одном компоненте.

4. **Изоляция на уровне гипервизора**: Гипервизоры типа Type 1 (KVM, Hyper-V) обеспечивают полную виртуализацию, где каждая виртуальная машина (ВМ) работает в изолированном окружении. Технологии типа AMD SEV (Secure Encrypted Virtualization) шифруют память ВМ, защищая её от несанкционированного доступа со стороны гипервизора.

Примером комплексной изоляции является проект **gVisor** от Google, который запускает контейнеры в легковесной песочнице, эмулируя системные вызовы для минимизации контакта с хост-ядром. Это предотвращает атаки, эксплуатирующие уязвимости ядра, такие как Dirty COW.

Политика контроля информационных потоков

Контроль информационных потоков регулирует передачу данных между субъектами (процессами, пользователями) и объектами (файлами, сетевыми ресурсами). Эта политика базируется на моделях безопасности, адаптированных под различные сценарии:

a. **Модель Белла-ЛаПадулы**: Фокусируется на конфиденциальности, запрещая чтение данных выше уровня допуска субъекта («no read up») и запись ниже («no write down»). Применяется в военных системах, где, например, агент с уровнем «секретно» не может прочитать документ «совершенно секретно», но может записать в файл с более низким уровнем.

b. **Модель Биба**: Обеспечивает целостность, запрещая запись данных в объекты с более высоким уровнем целостности («no write up») и чтение из объектов с более низким («no read down»). Используется в финансовых системах для предотвращения модификации критических транзакций.

Реализация в современных ОС:

a. **SELinux/AppArmor**: Реализуют мандатный контроль доступа (MAC), назначая метки конфиденциальности (SELinux) или профили поведения (AppArmor). Например, политика SELinux может запрещать веб-серверу доступ к домашним каталогам пользователей, даже если права DAC разрешают.

b. **Тегирование данных**: Файлы помечаются метаданными (например, через расширенные атрибуты в ext4), указывающими уровень секретности. Системы DLP используют эти метки для блокировки передачи данных через неавторизованные каналы.

c. **Шифрование на транспортном уровне**: TLS и VPN-туннели шифруют данные в пути, предотвращая перехват. В Windows функция BitLocker обеспечивает шифрование дисков, а в Linux – dm-crypt.

Сценарий применения: В корпоративной сети DLP-система может анализировать исходящий трафик, блокируя отправку файлов с меткой «конфиденциально» через мессенджеры. Одновременно межсетевой экран (например, iptables) ограничивает доступ к внутренним ресурсам только для IP-адресов филиалов.

Политика контроля прав доступа

Права доступа определяют, кто и как может взаимодействовать с ресурсами системы. Эволюция моделей управления доступом отражает растущие требования к гибкости и безопасности:

1. **Дискреционный контроль доступа (DAC)**. Владелец объекта самостоятельно назначает права. В Unix-системах это реализовано через разрешения rwx и списки ACL (Access Control Lists). Однако DAC уязвим к ошибкам: пользователь может случайно открыть доступ к конфиденциальному файлу.

2. **Ролевой контроль доступа (RBAC)**. Права привязаны к ролям, а не пользователям. В Windows группы Active Directory (например, «Администраторы», «Гости») определяют уровень доступа. RBAC упрощает управление в крупных организациях, но может привести к избыточным привилегиям («ролевой взрыв»).

3. **Атрибутивный контроль доступа (ABAC)**. Решения принимаются на основе атрибутов (должность, время доступа, местоположение). Например, доступ к бухгалтерской системе разрешён только с корпоративного IP в рабочее время. В облачных средах (AWS IAM) ABAC используется для тонкой настройки прав.

Современные механизмы:

a. **Capabilities в Linux**. Заменяют привилегии root на отдельные права (например, CAP_SYS_ADMIN для управления системой). Демон может иметь CAP_NET_BIND_SERVICE для привязки к порту 80 без полного доступа.

b. **Mandatory Integrity Control (MIC) в Windows**. Процессы и объекты помечаются уровнями целостности (низкий, средний, высокий). Браузер с низким уровнем не может модифицировать системные файлы, даже если пользователь – администратор.

c. **Политики наименьших привилегий**. macOS использует sandbox-профили для ограничения приложений. Например, программа для редактирования фото не получит доступ к микрофону без явного разрешения.

При работе над курсовой работой обратите внимание на:

– описание особенностей политики прав доступа в ОС, которые требуется приводить в первой (теоретической) главе (в соответствии с темой курсовой работы).

Литература

1. Окороков, В. А. Безопасность операционных систем / В. А. Окороков. – Санкт-Петербург: Лань, 2024. – 228 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/367472 (дата обращения: 06.12.2024). – ISBN 978-5-507-48297-9. – Текст : электронный (гл. 7).

2. Киренберг, Г. А. Информационная безопасность современных операционных систем: учеб. пособие / Киренберг, Г. А. – Кемерово: КузГТУ им. Т.Ф. Горбачева, 2022. – 138 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/295736 (дата обращения: 06.12.2024). – ISBN 978-5-00137-320-9. – Текст : электронный (гл. 3).

3. Потерпеев, Г. Ю. Безопасность операционных систем: учеб. пособие / Г. Ю. Потерпеев, В. С. Нефедов, А. А. Криулин. – Москва: РТУ МИРЭА, 2021. – 93 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/182416

(дата обращения: 06.12.2024). – ISBN 978-5-7339-1393-3. – Текст : электронный (гл. 1, 4).

4. Зверева, О. М. Операционные системы: учеб. пособие / О. М. Зверева; науч. ред. Л. Г. Доросинский; Уральский федеральный университет им. первого Президента России Б. Н. Ельцина. – Екатеринбург: Издательство Уральского университета, 2020. – 223 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=699030 (дата обращения: 03.11.2024). – ISBN 978-5-7996-3146-8. – Текст : электронный (гл. 11).

Контрольные вопросы

1. Приведите особенности матриц доступа.
2. Приведите особенности контроля информационных потоков
3. Приведите особенности контроля прав доступа.
4. Приведите особенности моделей прав доступа.
5. Какие аппаратные технологии (Intel VT-x, NX bit) обеспечивают изоляцию процессов и как они интегрированы в ОС?
6. Объясните, как модель Биба предотвращает модификацию финансовых транзакций. Приведите пример из банковской системы.
7. Почему ролевая модель (RBAC) может привести к «ролевому взрыву» и как это решается в ABAC?
8. Как KASLR усложняет эксплуатацию уязвимостей ядра? Опишите на примере атаки на драйвер устройства.
9. Какие инструменты в Linux позволяют назначать capabilities вместо полных прав root? Приведите сценарий использования.
10. Как Mandatory Integrity Control в Windows ограничивает вредоносное ПО с низким уровнем целостности?
11. Опишите процесс настройки DLP для блокировки передачи конфиденциальных данных через USB. Какие метаданные используются?
12. Какие уроки можно извлечь из инцидента с Equifax для улучшения политик контроля прав доступа?

Тема 2.2 Управление доступом в операционных системах семейства UNIX

Перечень изучаемых вопросов

1. Идентификатор пользователя, группы.
2. Модель полномочий Linux.
3. Дополнительные атрибуты безопасности.

Методические указания к изучению

Администратор системы – пользователь root – имеет UID равный нулю (0). Кроме администратора, есть еще несколько идентификаторов пользователей, которые автоматически создаются системой при установке.

EUID – это «эффективный» UID процесса. Управление правами доступа. **Sticky bit (он же бит закрепления в памяти), SUID (он же SetUserID), SGID (он же SetGroupID).** Дополнительные возможности по управлению правами доступа к файлам

Построение файловой системы и разграничение доступа к файловым объектам имеет особенности, присущие данному семейству ОС. Рассмотрим кратко эти особенности. Все дисковые накопители (тома) объединяются в единую ***виртуальную файловую систему*** путем операции монтирования тома. При этом содержимое тома проецируется на выбранный каталог файловой системы. Элементами файловой системы являются также все устройства, подключаемые к защищаемому компьютеру (монтируемые к файловой системе). Поэтому разграничение доступа к ним осуществляется через файловою систему.

Каждый файловый объект имеет индексный дескриптор, в котором среди прочего хранится информация о разграничении доступа к данному файловому объекту. Права доступа делятся на три категории: доступ для владельца, доступ для группы и доступ для остальных пользователей. В каждой категории определяются права на чтение, запись и исполнение (в случае каталога – просмотр).

Идентификатор пользователя называется UID – User Identifier, а идентификатор его группы – GID – Group Identifier. При каждом входе пользователя в систему, ядро Юникса регистрирует его UID и GID и выполняет все последующие процессы (программы) пользователя в соответствии с назначенными его UID и GID правами доступа.

Администратор системы – пользователь root – имеет UID равный нулю (0), и на него не распространяются никакие ограничения системы. То есть, он может читать любой файл в системе, добавлять-удалять устройства, администрировать аккаунты пользователей и делать все остальные присущие администрированию системы действия.

Кроме администратора, есть еще несколько идентификаторов пользователей, которые автоматически создаются системой при установке - такие как daemon (uid=1), bin (uid=2), sys (uid=3), adm (uid=4), lp, uucp, и nobody. Конкретные номера идентификаторов пользователей для этих имен, а также наличие приведенных здесь и других специальных системных аккаунтов **зависят от конкретного Unix.**

Эти системные аккаунты используются автоматически для разделения и безопасного выполнения системных задач (т.е. чтобы многие операции можно было запускать с полномочиями этих аккаунтов, а не суперпользовательскими).

EUID – это «эффективный» UID процесса. EUID используется для того, чтобы определить, к каким ресурсам и файлам у процесса есть право доступа. У большинства процессов UID и EUID будут одинаковыми. Исключение составляют программы, у которых установлен бит смены идентификатора пользователя.

Помимо этого, в ОС данного класса используется эффективный идентификатор группы (EGID). GID – это идентификационный номер группы данного процесса. EGID связан с GID также, как EUID с UID.

Для каждого объекта файловой системы в модели полномочий Linux есть три типа полномочий: полномочия *чтения* (*r от read*), *записи* (*w от write*) и *выполнения* (*x от execution*). В полномочия записи входят также возможности удаления и изменения объекта. Право выполнения можно установить для любого файла. Потенциально, любой файл в системе можно запустить на выполнение, как программу в Windows. В Linux является ли файл исполняемым или нет, определяется не по его расширению, а по правам доступа. Кроме того, эти полномочия указываются отдельно для *владельца* файла, членов *группы* файла и для всех *остальных*.

Управление правами доступа происходит с помощью команды ***chmod***, управление владельцем файла происходит с помощью команды chown.

В Linux кроме прав чтения, выполнения и записи, есть еще 3 дополнительных атрибута:

1. Sticky bit (он же бит закрепления в памяти). Sticky bit появился в пятой редакции UNIX в 1974 г. для использования в исполняемый файлах. Он применялся для уменьшения времени загрузки наиболее часто используемых программ. После закрытия программы код и данные оставались в памяти, а следующий запуск происходил быстрее. (отсюда и название – бит закрепления в памяти)

Сегодня sticky bit используется в основном для каталогов, чтобы защитить в них файлы. В такой каталог может писать ЛЮБОЙ пользователь. Из такой директории пользователь может удалить только те файлы, владельцем которых он является. Примером может служить директория /tmp, в которой запись открыта для всех пользователей, но нежелательно удаление чужих файлов.

2. SUID (он же Set User ID). Атрибут исполняемого файла, позволяющий запустить его с правами владельца. В Unix-подобных системах приложение запускается с правами пользователя, запустившего указанное приложение.

3. SGID (он же Set Group ID). Аналогичен SUID, но относиться к группе. При этом, если для каталога установлен бит SGID, то создаваемые в нем объекты будут получать группу владельца каталога, а не пользователя.

В Linux права доступа сохраняются в inode файла, и поскольку inode у каждого файла свой собственный, права доступа у каждого файла свои. Так же, права доступа пользователя и группы не суммируются. Если программа выполняется с правами пользователя и группы, которым принадлежит файл – работают только права хозяина файла.

Управление доступом в UNIX-подобных операционных системах – это основа их безопасности, формирующая барьеры между пользователями, процессами и критически важными ресурсами. С момента создания UNIX в 1969 году управление доступом эволюционировало от простой дискреционной модели до сложных систем, включающих мандатный контроль, списки доступа и

расширенные атрибуты. Сегодня, в эпоху облачных вычислений и контейнеризации, эти механизмы остаются актуальными, но требуют глубокого понимания для противодействия таким угрозам, как горизонтальное перемещение в сетях, эксплуатация привилегированных процессов и утечки данных. Данная статья исследует архитектуру управления доступом в UNIX, её исторические корни, современные расширения и методику изучения, необходимую для защиты систем в условиях растущей сложности кибератак.

Идентификатор пользователя и группы: основа разграничения прав

Каждый субъект в UNIX-системе — пользователь, процесс или служба – идентифицируется уникальным числовым идентификатором (UID) и списком групп (GID). Эти идентификаторы определяют права доступа к файлам, сетевым ресурсам и системным вызовам.

Структура системных файлов

- **/etc/passwd**: Содержит записи о пользователях в формате username:x:UID:GID:gecos:/home/dir:/bin/sh. Поле x указывает, что пароль хранится в зашифрованном виде в /etc/shadow, защищённом от чтения обычными пользователями.
- **/etc/group**: Определяет группы и их GID, а также список членов. Например, строка developers:x:1001:alice,bob создаёт группу developers с пользователями Alice и Bob.
- **/etc/shadow**: Хранит хэши паролей, соль (salt) и параметры истечения сроков. Современные системы используют алгоритмы вроде SHA-512 с 8-байтовой солью для замедления атак перебора.

Привилегии суперпользователя

- **UID 0**: Учётная запись root обладает неограниченными правами, включая доступ к /dev/mem (прямое управление памятью) и изменение системных конфигураций. Работа под root крайне рискованна — даже случайная команда rm -rf / может уничтожить систему.
- **sudo и su**: Инструменты делегирования привилегий. Файл /etc/sudoers определяет, какие команды пользователи могут выполнять с повышенными правами. Например, строка alice ALL=(ALL) NOPASSWD: /usr/bin/systemctl restart apache позволяет Alice перезапускать Apache без ввода пароля.

Реальные и эффективные идентификаторы

a. Процесс наследует **реальный UID/GID** пользователя, запустившего его. Однако при выполнении файла с **SUID-битом** (например, /usr/bin/passwd) процесс получает **эффективный UID** владельца файла (в данном случае root). Это позволяет выполнять привилегированные операции без постоянных прав суперпользователя.

b. **Уязвимости SUID**. Если программа с SUID содержит ошибку (например, переполнение буфера), злоумышленник может выполнить произвольный код с правами владельца. Пример – уязвимость CVE-2021-4034 в pkexec, где ошибка обработки аргументов позволила локальную эскалацию привилегий.

Модель полномочий Linux: классические права и их ограничения

Классическая модель прав доступа в UNIX базируется на трёх категориях: владелец (user), группа (group) и остальные (others). Каждой категории назначаются права на чтение (r), запись (w) и выполнение (x).

Права файлов и каталогов

- **Файлы**:
- **r: Чтение содержимого.**
- **w: Модификация или удаление.**
- **x: Запуск как исполняемого файла.**
- Каталоги:
  - r: Просмотр списка файлов.
- **w: Создание или удаление файлов.**
- **x: Доступ к содержимому файлов (требуется для cd).**

Пример: Права drwxr-xr-- для каталога означают, что владелец может читать, записывать и открывать каталог; группа – читать и открывать; остальные – только читать.

Специальные биты: SUID**, SGID, Sticky Bit**

a. **SUID (Set User ID)**. Устанавливается командой chmod u+s file. Пример: passwd требует прав root для изменения /etc/shadow, поэтому имеет SUID-бит: -rwsr-xr-x. Риск: устаревшие SUID-программы (например, mount) могут стать вектором атак.

b. **SGID (Set Group ID)**. Для файлов: Запуск с эффективным GID владельца. Для каталогов: Новые файлы наследуют группу каталога, а не создателя. Полезно для совместной работы (например, общий каталог /var/www с SGID).

c. **Sticky Bit**. Устанавливается командой chmod +t dir. В каталогах (например, /tmp) запрещает удаление файлов невладельцами. Без Sticky Bit пользователь мог бы удалить файлы других, если у каталога есть право w.

Ограничения классической модели

1. **Негибкость**: Невозможно назначить права для нескольких пользователей или групп вне основной тройки (user, group, others).
2. **Отсутствие контекстного доступа**: Права не учитывают время, местоположение или тип операции (например, запрет редактирования файла в нерабочие часы).
3. **Слабый контроль процессов**: Нет механизмов ограничения действий процессов (например, запрет на запись в сетевые сокеты).

Дополнительные атрибуты безопасности

Для преодоления ограничений классической модели UNIX-системы используют расширенные механизмы управления доступом.

Access Control Lists (ACL)

ACL позволяют назначать права для конкретных пользователей и групп, игнорируя базовые категории.

- **Установка прав**:

```
bash
Copy
setfacl -m u:alice:rwx,g:developers:r-- file.txt  # Правадля Alice игруппы developers
setfacl -x g:developers file.txt                  # Удаление прав группы
```

- **Просмотр ACL**:

```
bash
Copy
getfacl file.txt
# Вывод:
# user:alice:rwx
# group:developers:r--
```

- **Маски ACL**: Ограничивают максимальные права для групп (аналогично umask).

SELinux: Мандатный контроль доступа

SELinux (Security-Enhanced Linux) реализует модель принудительного контроля доступа (MAC), где права определяются политиками, а не владельцами.

- **Контексты безопасности**: Каждому файлу и процессу назначается метка (например, user_u:object_r:httpd_sys_content_t).
- **Политики**:

a. **Targeted**: Блокирует только определённые службы (например, веб-серверы).

b. **Strict**: Полный контроль для всех процессов.

- **Пример**: Политика может запретить процессу Apache (httpd_t) доступ к файлам с меткой user_home_t, даже если классические права разрешают чтение.
- **Инструменты**:

a. semanage — управление контекстами.

b. audit2allow — генерация правил из логов нарушений.

Capabilities: Гранулярные привилегии

Вместо предоставления полного доступа root, Linux позволяет назначать отдельные привилегии через capabilities.

- **Примеры capabilities**:

a. CAP_NET_BIND_SERVICE: Привязка к портам<1024.

b. CAP_SYS_ADMIN: Администрирование системы (аналог прав root).

c. CAP_DAC_OVERRIDE: Игнорирование прав доступа к файлам.

- **Назначение**:

```
bash
Copy
setcap cap_net_bind_service=+ep /usr/bin/my_server  # Разрешитьпривязкукпорту 80
```

- **Риски**: Злоупотребление capabilities (например, CAP_DAC_OVERRIDE) может обойти классические права.

Атрибуты файлов (chattr)

Утилита chattr управляет атрибутами файлов на уровне файловой системы:

- **immutable (+i)**: Запрещает изменение, удаление и переименование файла даже для root.

```
bash
Copy
chattr +i /etc/resolv.conf  # Защита DNS-настроек
```

- **append-only (+a)**: Разрешает только добавление данных (полезно для логов).
- **secure deletion (+s)**: Перезапись данных нулями перед удалением.

При работе над курсовой работой обратите внимание на:

– описание особенностей политики прав доступа в ОС, которые требуется приводить в первой (теоретической) главе (в соответствии с темой курсовой работы).

Литература

1. Окороков, В. А. Безопасность операционных систем / В. А. Окороков. – Санкт-Петербург: Лань, 2024. – 228 с. – Режи м доступа: для авториз. пользователей. – Лань: электронно-библиотечная система. – URL: https://e.lanbook.com/book/367472 (дата обращения: 06.12.2024). – ISBN 978-5-507-48297-9. – Текст : электронный (гл. 7, с. 201–210).

2. Киренберг, Г. А. Информационная безопасность современных операционных систем: учеб. пособие / Киренберг, Г. А. – Кемерово: КузГТУ им. Т. Ф. Горбачева, 2022. – 138 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/295736 (дата обращения: 06.12.2024). – ISBN 978-5-00137-320-9. – Текст : электронный (гл. 3).

3. Потерпеев, Г. Ю. Безопасность операционных систем: учеб. пособие / Г. Ю. Потерпеев, В. С. Нефедов, А. А. Криулин. – Москва: РТУ МИРЭА, 2021. – 93 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/182416 (дата обращения: 06.12.2024). – ISBN 978-5-7339-1393-3. – Текст : электронный (гл. 4).

4. Зверева, О. М. Операционные системы: учеб. пособие / О. М. Зверева; науч. ред. Л. Г. Доросинский; Уральский федеральный университет им. первого Президента России Б. Н. Ельцина. – Екатеринбург: Издательство Уральского университета, 2020. – 223 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=699030 (дата обращения: 03.11.2024). – ISBN 978-5-7996-3146-8. – Текст : электронный (гл. 11).

Контрольные вопросы

1. Приведите особенности матриц доступа в ОС Linux.
2. Приведите особенности контроля информационных потоков в ОС Linux.
3. Приведите особенности контроля прав доступа в ОС Linux.
4. Приведите особенности моделей прав доступа в ОС Linux.
5. Права доступа: Объясните, почему каталог с правами 777 опасен, даже если Sticky Bit установлен.
6. SUID и процессы: Как эффективный UID влияет на системные вызовы setuid() и execve()?
7. SELinux: Что произойдёт, если процесс с контекстом httpd_t попытается прочитать файл с меткой shadow_t?
8. Capabilities: Почему назначение CAP_DAC_OVERRIDE эквивалентно предоставлению прав root?
9. ACL: Как маски ACL взаимодействуют с правами групп? Приведите пример.
10. Атрибуты файлов: Можно ли удалить файл с атрибутом immutable через rm -f? Обоснуйте.

Тема 2.3 Управление доступом в операционных системах семейства WINDOWS

Перечень изучаемых вопросов

1. Объекты доступа.
2. Субъекты доступа
3. Методы доступа к объектам.
4. Права доступа к объектам.
5. Привилегии субъектов.
6. Маркер доступа пользователя.

Методические указания к изучению

Рассмотреть следующие аспекты темы

ОС Windows поддерживает 22 метода доступа субъектов к объектам. Шесть из них представляют собой стандартные методы доступа и поддерживаются для объектов всех типов.

Каждому методу доступа соответствует право на его осуществление. Эти права доступа называются специфичными, поскольку они специфичны для каждого типа объектов. Для каждого типа объектов может поддерживаться до 16 специфичных прав доступа.

В Windows каждый субъект доступа обладает некоторым набором привилегий. Привилегии представляют собой права на выполнение субъектом действий, касающихся системы в целом, а не отдельных ее объектов. В Windows каждый пользователь (в том числе и каждый псевдопользователь), работающий в системе, имеет свой маркер доступа (access token). Каждому

процессу Windows назначается первичный маркер доступа (primary access token). Все пользователи и псевдопользователи ОС, включая администраторов и ОС, обладают ограниченными полномочиями.

Атрибуты защиты объекта Windows описываются специальной структурой данных, называемой дескриптором защиты (security descriptor).

В Windows NT все объекты ОС являются объектами доступа. Иерархия типов объектов имеет древовидную структуру. Операции, определенные над объектами некоторого типа, наследуются и объектами всех подтипов данного типа. Элементы списка избирательного контроля доступа н (access control entries, ACE).

Windows позволяет прикладным и сервисным процессам создавать объекты доступа нестандартных типов, перед созданием которых процесс должен зарегистрировать в системе данный тип объекта. Разграничение доступа субъектов к нестандартным объектам организуется так же, как и к стандартным.

Субъекты доступа. Субъекты доступа, которые поддерживает ОС Windows.

1. Пользователи – обычные пользователи и псевдопользователи. К псевдопользователям относятся следующие субъекты доступа:

• SYSTEM – ОС локального компьютера; этот псевдопользователь всегда входит в группу Administrators и имеет все привилегии;

• псевдопользователи с именами вида <имя_компьютера>$, где <имя_компьютера> – сетевое имя компьютера; эти псевдопользователи представляют ОС других компьютеров сети и используются при аутентификации рабочей станции на контроллере домена.

2. Группы пользователей. Группы пользователей могут пересекаться, т.е. каждый пользователь может входить в несколько групп. При этом для совместимости с программным интерфейсом POSIX, поддерживаемым среди групп, в которые входит пользователь, выделяется первичная группа, которая играет роль той единственной группы, в которую может входить пользователь в POSIX.

3. Специальные (временные) группы. В отличие от обычных групп членство пользователя в таких группах определяется ОС в зависимости от действий пользователя. Специальная группа не может быть объявлена первичной группой пользователя.

4. Относительные субъекты. Эти субъекты имеют смысл только в применении к объекту, для которого определяются права доступа. Существуют следующие относительные субъекты:

• CREATOR_OWNER – владелец объекта;

• CREATOR GROUP – первичная группа владельца объекта.

Относительные субъекты используются, если нужно описать права доступа пользователей к объектам по принципу «что кому принадлежит, то ему и доступно».

Также существует несколько предопределенных идентификаторов, которые используются ОС внутренне или зарезервированы для последующих версий.

Идентификаторы остальных субъектов доступа уникальны.

Члены предопределенной группы Administrators могут создавать, удалять и изменять свойства любых субъектов, а члены группы Account Operators – создавать, удалять и изменять свойства только непривилегированных субъектов (обычных пользователей). Если не ограничиваться использованием только стандартных средств администрирования, то для работы со списком субъектов достаточно иметь полный доступ к нескольким ключам реестра.

Станция в составе рабочей группы. Ресурсы каждой рабочей станции доступны другим рабочим станциям в составе группы. Информация безопасности, управляющая допуском пользователей к ресурсам рабочей станции, хранится на этой станции.

Член домена. В домене предусмотрено централизованное управление безопасностью через базу данных, хранящуюся на одном из серверов домена, который называется первичным контроллером домена (Primary Domain Con troller – P O C). Клиентская рабочая станция в составе гетерогенной вычислительной сети (например, IntranetWare и WindowsNT). Удаленный доступ к ресурсам данной рабочей станции невозможен.

Изложенная выше структура носит название NT Directory Services (NTDS).

Профили пользователя представляют собой набор параметров, определяющих:

- настройки рабочего стола пользователя (положение значков, обои и т. д.);
- автоматические подключения сетевых дисков при входе в сеть;
- приложения, которые запускаются при старте операционной системы.

Различают следующие типы профилей пользователя:

- локальный профиль, хранящийся на рабочей станции;
- блуждающий, хранящийся на сервере – первичном контроллере домена – и изменяющийся пользователем;
- мандатный, хранящийся на сервере – первичном контроллере домена – и не изменяющийся пользователем.

Системная политика (System Policy) в Windows NT представляет собой некоторый набор значений, который присваивается соответствующим параметрам реестра в момент аутентификации пользователя в сеть. Системная политика определяется для пользователя, группы пользователей, и для компьютера. В случае, если какой-то параметр системной политики противоречит настройкам профиля пользователя, используется настройка системной политики.

Использование профилей и системных политик позволяет создать замкнутую рабочую среду, облегчающую выполнение пользователем

производственных задач и затрудняющих выполнение не относящихся к основной производственной деятельности действий.

Помимо перечисленных выше групп определена группа Everyone. Эта группа включает по умолчанию в себя всех пользователей Windows NT. Список членов этой группы не может быть изменен. Стандартные группы не могут быть переименованы.

Методы доступа к объектам. ОС Windows поддерживает 22 метода доступа субъектов к объектам. Шесть из них представляют собой стандартные методы доступа и поддерживаются для объектов всех типов:

• удаление объекта;

• получение атрибутов защиты объекта;

• изменение атрибутов защиты объекта;

• изменение владельца объекта; при этом субъект может объявить новым владельцем объекта только себя;

• ACCESS_SYSTEM_SECURITY – получение и изменение параметров аудита в отношении объекта;

• SYNCHRONIZE – метод доступа, заключающийся в вызове системной функции WaitForSingleObject для данного объекта или функции WaitForMultipleObjects для списка объектов, включающего данный объект. Эти функции используются, когда поток должен ожидать какое-то изменение в состоянии объекта, не затрачивая на это процессорного времени. Обычно этот метод доступа применяется к объектам синхронизации, реже – к процессам и потокам.

Для каждого типа объекта поддерживается до шестнадцати специфичных методов доступа.

Права доступа к объектам. Каждому методу доступа соответствует право на его осуществление. Эти права доступа называются специфичными, поскольку они специфичны для каждого типа объектов. Для каждого типа объектов может поддерживаться до 16 специфичных прав доступа.

Каждому стандартному методу доступа, за исключением ACCESS_SYSTEM_SECURITY, также соответствует право доступа, дающее возможность реализации соответствующего метода доступа. Такие права доступа называются стандартными.

Windows NT поддерживает также общие (generic) или отображаемые (mapped) права доступа. Каждое из отображаемых прав доступа представляет собой некоторую комбинацию стандартных и специфичных прав доступа. Отображаемые права доступа могут быть предоставлены для доступа к объекту любого типа, однако конкретное содержание отображаемого права доступа зависит от типа объекта. Процесс преобразования отображаемого права доступа в набор прав на реализацию методов доступа к объекту называется отображением права доступа. Порядок отображения отображаемых методов доступа для объектов конкретного типа определяется при регистрации данного типа объектов.

Отображаемые права доступа введены в систему разграничения доступа по следующим двум причинам:

– отображаемые права доступа позволяют пользователю устанавливать права доступа к объекту, ничего не зная о специфике объектов данного типа;

– отображаемые права доступа необходимы для обеспечения совместимости с POSIX.

Последним классом прав доступа, поддерживаемых Windows, являются виртуальные права доступа, которые не могут быть предоставлены субъекту, но могут быть запрошены им.

Привилегии субъектов. В Windows каждый субъект доступа обладает некоторым набором привилегий. Привилегии представляют собой права на выполнение субъектом действий, касающихся системы в целом, а не отдельных ее объектов. Существуют следующие привилегии:

• завершать работу ОС и перезагружать компьютер;

• устанавливать системное время;

• анализировать производительность одного процесса тп.

Существует еще несколько привилегий, идентификаторы которых зарезервированы для использования в будущих версиях Windows.

При входе в систему пользователь получает привилегии, предоставленные ему индивидуально, а также привилегии, предоставленные группам, в которые он входит. Назначать привилегии субъектам доступа может только администратор. Обычно все привилегии пользователя, кроме привилегии получать оповещения от файловых систем, выключены. Для того, чтобы пользователь смог воспользоваться своей привилегией, он должен вначале ее включить с помощью системного вызова AdjustTokenPrivileges. После использования привилегию рекомендуется снова выключить.

Некоторые из перечисленных привилегий позволяют субъектам, обладающим ими, преодолевать те или иные элементы защиты ОС.

Маркер доступа пользователя. В Windows каждый пользователь (в том числе и каждый псевдопользователь), работающий в системе, имеет свой маркер доступа.

Маркер доступа (access token) – это объект специального вида, содержащий следующую информацию:

• идентификатор пользователя;

• идентификаторы групп и специальных групп, в которые входит пользователь;

• привилегии пользователя;

• идентификатор сеанса работы пользователя, к которому относится маркер доступа;

• атрибуты защиты, которые назначаются по умолчанию новым объектам, созданным пользователем в текущем сеансе работы;

• имя и идентификатор подсистемы, выдавшей маркер доступа (Advapi, если пользователь вошел в систему локально, NtLogon, если пользователь вошел в систему по сети через SMB-сервер, и т. д.);

• некоторую служебную информацию.

Маркер доступа содержит всю информацию о пользователе, необходимую системе разграничения доступа для принятия решений о предоставлении пользователю доступа к тем или иным объектам.

Каждому процессу Windows NT назначается первичный маркер доступа (primary access token) – маркер доступа пользователя, запустившего данный процесс. Субъект, обладающий соответствующей привилегией, может назначить процессу другой первичный маркер доступа. Отдельным потокам процесса могут назначаться свои маркеры доступа – маркеры олицетворения (impersonation access tokens).

Дескриптор защиты. Атрибуты защиты объекта Windows описываются специальной структурой данных, называемой дескриптором защиты (security descriptor), который содержит следующую информацию:

• идентификатор владельца объекта;

• идентификатор первичной группы владельца объекта;

• список избирательного контроля доступа (discretionary access control list, DACL) - список, полностью описывающий права различных субъектов на объект;

• системный список контроля доступа (system access control list, SACL) – используется при генерации сообщений аудита.

Если объект не имеет дескриптора защиты, при обращениях субъектов к нему права доступа не проверяются. В этом случае любой субъект имеет абсолютные права на данный объект.

Дескриптор защиты хранится вместе с объектом, при этом формат хранения объекта должен предоставлять такую возможность.

Элементы списка избирательного контроля доступа называются элементами контроля доступа (access control entries, ACE). Каждый элемент контроля доступа разрешает или запрещает некоторому субъекту определенные права доступа к объекту. Если список избирательного контроля доступа отсутствует в дескрипторе защиты, всем субъектам предоставляются все права доступа к объекту.

Владелец имеет право изменять дескриптор защиты объекта, даже если это явно запрещено ему списком избирательного контроля доступа. Пользователь, обладающий привилегией объявлять себя владельцем объекта (привилегия администратора), может объявлять себя владельцем тех объектов, для которых это явно запрещено ему списком избирательного контроля доступа. В остальном список избирательного контроля доступа полностью описывает права различных субъектов на доступ к данному объекту.

Каждая запись управления доступом (ACE) состоит из идентификатора пользователя или группы пользователей и совокупности разрешенных методов доступа.

Стандартные средства работы с файлами не поддерживают механизм разграничения доступа в полном объеме. Для того чтобы список избирательного контроля доступа файла можно было просматривать и

редактировать, информация, содержащаяся в этом списке, должна удовлетворять следующим требованиям:

– все элементы списка, запрещающие доступ субъектов к объекту, должны находиться в начале списка;

– все элементы списка, запрещающие доступ субъектов к объекту, должны запрещать им все права доступа к объекту.

В отличие от UNIX в Windows отсутствует суперпользователь. Все пользователи и псевдопользователи ОС, включая администраторов и ОС, обладают ограниченными полномочиями. Однако субъект, обладающий привилегией администратора, может получить доступ к любому объекту ОС по любому методу доступа за исключением метода ACCESS_SYSTEM_SECURITY, для которого требуется привилегия аудитора. Для этого субъект должен вы полнить следующие действия:

– используя привилегию администратора, объявить себя владельцем объекта;

– используя полномочия владельца, предоставить себе необходимые права доступа к объекту;

– обратиться к объекту, используя полученные права.

При работе над курсовой работой обратите внимание на:

– описание особенностей политики прав доступа в ОС, которые требуется приводить в первой (теоретической) главе (в соответствии с темой курсовой работы).

Литература

1. Окороков, В. А. Безопасность операционных систем / В. А. Окороков. – Санкт-Петербург: Лань, 2024. – 228 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/367472 (дата обращения: 06.12.2024). – ISBN 978-5-507-48297-9. – Текст : электронный (гл. 7).

2. Киренберг, Г. А. Информационная безопасность современных операционных систем: учеб. пособие / Киренберг, Г. А. . – Кемерово: КузГТУ им. Т. Ф. Горбачева, 2022. – 138 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/295736 (дата обращения: 06.12.2024). – ISBN 978-5-00137-320-9. – Текст : электронный (гл. 3).

3. Потерпеев, Г. Ю. Безопасность операционных систем: учеб. пособие / Г. Ю. Потерпеев, В. С. Нефедов, А. А. Криулин. – Москва: РТУ МИРЭА, 2021. – 93 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/182416 (дата обращения: 06.12.2024). – ISBN 978-5-7339-1393-3. – Текст : электронный (гл. 1, 4).

4. Зверева, О. М. Операционные системы: учеб. пособие / О. М. Зверева; науч. ред. Л. Г. Доросинский; Уральский федеральный университет

им. первого Президента России Б. Н. Ельцина. – Екатеринбург: Издательство Уральского университета, 2020. – 223 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=699030 (дата обращения: 03.11.2024). – ISBN 978-5-7996-3146-8. – Текст : электронный (гл. 11).

Контрольные вопросы

1. Приведите особенности матриц доступа в ОС Windows.
2. Приведите особенности контроля информационных потоков в ОС Windows.
3. Приведите особенности контроля прав доступа в ОС Windows.
4. Приведите особенности моделей прав доступа в ОС Windows.

Тема 2.4 Идентификация и аутентификация в ОС LINUX

Перечень изучаемых вопросов

1. Стандартная процедура идентификации и аутентификации.
2. Основные механизмы защиты паролей.
3. Шифрование паролей.

Методические указания к изучению

Linux обеспечивает защиту паролей с помощью трех основных механизмов:

– шифрование паролей.

– механизм «теневых паролей».

– механизм подключаемых модулей аутентификации PAM (Pluggable Authentication Modules).

Механизм «теневых паролей». Механизм PAM представляет собой набор открытых библиотек подключаемых модулей аутентификации (PAM), предназначенных для выполнения ввода пароля и проверки его подлинности.

В Linux для шифрования паролей используется алгоритм DES. Зашифрованный пароль обычно помещается в файл /etc/passwd.

Тема «Идентификация и аутентификация в ОС Linux» занимает центральное место в курсе «Безопасность операционных систем», поскольку эти процессы являются основой обеспечения контролируемого доступа к ресурсам системы. В условиях современных киберугроз, когда несанкционированный доступ может привести к компрометации данных или нарушению работы системы, глубокое понимание механизмов идентификации и аутентификации становится критически важным. В рамках данной темы рассматриваются стандартная процедура идентификации и аутентификации, основные механизмы защиты паролей и методы их шифрования в контексте ОС Linux. Эти аспекты не только формируют базис для безопасного управления пользователями, но и открывают возможности для анализа уязвимостей, связанных с некорректной реализацией данных процессов. Настоящая статья подробно раскрывает указанные вопросы, дополняя их

методическими рекомендациями, направленными на эффективное освоение материала студентами и развитие у них навыков обеспечения безопасности на уровне операционной системы.

Стандартная процедура идентификации и аутентификации

Идентификация и аутентификация в ОС Linux представляют собой двухэтапный процесс, обеспечивающий определение личности пользователя и подтверждение его права доступа к системе. Идентификация начинается с того момента, когда пользователь предоставляет уникальный идентификатор, чаще всего в виде имени пользователя (login), которое система сопоставляет с записями в соответствующих базах данных, таких как файл /etc/passwd. Этот файл содержит информацию о пользователях, включая их имена, идентификаторы (UID), домашние директории и используемые оболочки. Успешная идентификация позволяет системе установить, с кем она имеет дело, но не гарантирует доступа — для этого требуется аутентификация.

Аутентификация в стандартной процедуре предполагает проверку предоставленного пользователем секрета, обычно пароля, который сравнивается с хранимой в системе информацией. В Linux пароли традиционно хранятся в зашифрованном виде в файле /etc/shadow, доступном только привилегированным пользователям (например, root). Процесс аутентификации осуществляется через вызовы к библиотекам PAM (Pluggable Authentication Modules), которые предоставляют гибкий и модульный подход к проверке подлинности. PAM позволяет интегрировать различные методы аутентификации — от паролей до биометрических данных – и настраивать их через конфигурационные файлы в директории /etc/pam.d/ или основной файл /etc/pam.conf. После успешной аутентификации система предоставляет пользователю доступ к его окружению, определяемому правами и настройками, указанными в системных файлах.

С точки зрения безопасности, стандартная процедура подвержена рискам, связанным с перехватом учетных данных, слабостью паролей или ошибками в конфигурации PAM.Например, недостаточная защита файла /etc/shadow или использование устаревших методов аутентификации могут стать уязвимостями. Методически изучение этого вопроса следует начинать с анализа структуры файлов /etc/passwd и /etc/shadow, их назначения и принципов работы. Студентам рекомендуется рассмотреть последовательность действий при входе в систему, начиная от ввода логина на терминале и заканчивая запуском пользовательской оболочки, с акцентом на роль PAM. Лабораторная часть может включать настройку простого сценария аутентификации через PAM и анализ логов (например, /var/log/auth.log), что позволяет понять, как система фиксирует попытки входа и выявляет потенциальные угрозы.

Файлы учетных записей

1. **/etc/passwd**:

Содержит записи о пользователях в формате: username:x:UID:GID:GECOS:/home/dir:/bin/bash

a. Поле x указывает, что пароль хранится в зашифрованном виде в /etc/shadow.

b. UID (User ID) и GID (Group ID) определяют права доступа.

c. Корневой каталог и оболочка задают среду пользователя.

2. **/etc/shadow**:

Хранит зашифрованные пароли и параметры безопасности: username:6salt$hash:last_changed:min_age:max_age:warn:inactive:expire

a. Поля min_age и max_age задают минимальный и максимальный срок действия пароля.

b. inactive определяет период блокировки учетной записи после истечения пароля.

Роль PAM (Pluggable Authentication Modules)

PAM предоставляет гибкую инфраструктуру для настройки аутентификации. Конфигурационные файлы в `/etc/pam.d/` определяют, какие модули используются для проверки паролей, управления сессиями и контроля доступа. Например, модуль `pam_unix.so` проверяет пароль через `/etc/shadow`, а `pam_tally2.so` блокирует учетную запись после нескольких неудачных попыток.

Пример аутентификации по SSH-ключу

Для повышения безопасности вместо паролей часто используются ключи:

1. Пользователь генерирует пару ключей (публичный и приватный) командой `ssh-keygen`.

2. Публичный ключ добавляется в `~/.ssh/authorized_keys` на сервере.

3. При подключении клиент доказывает владение приватным ключом через криптографический алгоритм (например, Ed25519).

Основные механизмы защиты паролей

Защита паролей в ОС Linux представляет собой комплекс программных и административных мер, направленных на предотвращение их компрометации. Пароли являются ключевым элементом аутентификации, и их безопасность напрямую влияет на общую защищенность системы. Одним из базовых механизмов защиты выступает ограничение доступа к файлу /etc/тень, где хранятся зашифрованные пароли. Этот файл доступен только пользователю root и группе shadow с соответствующими правами (обычно 640), что минимизирует риск чтения паролей непривилегированными пользователями или процессами. В отличие от более ранних систем, где пароли хранились в открытом файле /etc/passwd, разделение данных между этими файлами значительно повысило уровень безопасности.

Дополнительно используются механизмы контроля качества паролей, реализованные через модули PAM, такие как pam_passwdqc или pam_cracklib. Эти модули проверяют сложность пароля при его создании или изменении,

требуя минимальной длины, наличия символов разных регистров, цифр и специальных знаков, а также запрещая использование легко угадываемых комбинаций (например, имени пользователя или повторяющихся последовательностей). Это снижает вероятность успешных атак методом перебора (brute force) или использования словарей. Еще одним важным механизмом является ограничение срока действия паролей, задаваемое в /etc/shadow через параметры, такие как максимальный возраст пароля (обычно 90 дней) и период предупреждения перед истечением срока. После истечения срока действия пользователь обязан обновить пароль, что предотвращает длительное использование скомпрометированных учетных данных.

С точки зрения безопасности, данные механизмы не лишены недостатков. Например, слабая политика паролей или ошибки в настройке PAM могут снизить их эффективность, а физический доступ к системе позволяет обойти ограничения через загрузку в однопользовательский режим. Методически изучение этого вопроса требует погружения в структуру файла /etc/shadow и его полей (например, хэш пароля, даты изменения), а также анализа работы PAM-модулей. Студентам следует рассмотреть примеры настройки политик паролей в файлах /etc/pam.d/passwd и изучить влияние различных параметров на безопасность. Лабораторная работа может включать изменение прав доступа к файлу /etc/shadow с последующим тестированием системы на попытки несанкционированного чтения, а также настройку модуля pam_passwdqc для оценки его влияния на устойчивость паролей к атакам.

Политики сложности паролей

- Утилита pam_pwquality:

Настраивается через /etc/security/pwquality.conf и контролирует:

a. **Минимальную длину (minlen = 12).**
b. **Наличие разных классов символов (цифры, заглавные буквы).**
c. **Запрет простых последовательностей (12345) или повторяющихся символов (аааа).**

Блокировка учетных записей. Модуль pam_faillock.so фиксирует неудачные попытки входа в /var/log/faillock и блокирует пользователя после превышения лимита (например, 5 попыток).

Двухфакторная аутентификация (2FA). Для критически важных систем применяется 2FA, где пароль дополняется вторым фактором:

- **TOTP (Time-Based One-Time Password)**: Интеграция с Google Authenticator или Authy через модуль pam_google_authenticator.so.
- **Аппаратные токены**. Использование U2F-ключей (YubiKey) с поддержкой PAM-модуля pam_u2f.so.

Защита от перебора (brute-force)

- **Fail2Ban**:

Утилита анализирует логи (/var/log/auth.log) и блокирует IP-адреса, выполняющие подозрительные попытки входа.

- **Rate Limiting в SSH**:

Настройка sshd_config для ограничения числа подключений:

```
bash
Copy
MaxAuthTries 3
LoginGraceTime 1m
```

Шифрование паролей

Шифрование паролей в ОС Linux является неотъемлемой частью процесса аутентификации, обеспечивая защиту учетных данных от прямого доступа даже в случае компрометации файла /etc/shadow. В отличие от хранения паролей в открытом виде, что было характерно для ранних систем, современные версии Linux используют криптографические хэш-функции для преобразования паролей в необратимые последовательности символов. Исторически первой широко используемой функцией была DES (Data Encryption Standard), которая, однако, имела ограничения по длине пароля (до 8 символов) и уязвимость к атакам перебора из-за небольшого размера ключа. Впоследствии DES была заменена более надежными алгоритмами, такими как MD5, который стал стандартом в 1990-х годах благодаря большей длине хэша и устойчивости к простым атакам.

На сегодняшний день в Linux чаще всего применяется алгоритм SHA-512, интегрированный в библиотеку crypt, которая используется для шифрования паролей. SHA-512 обеспечивает высокий уровень криптографической стойкости благодаря 512-битному хэшу и использованию соли (salt) – случайной строки, добавляемой к паролю перед хэшированием. Соль предотвращает атаки с использованием радужных таблиц (rainbow tables), где заранее вычисленные хэши сравниваются с украденными данными, поскольку каждый пароль с уникальной солью генерирует уникальный хэш. Например, запись в /etc/shadow может выглядеть как 6<salt>$<hash>, где 6 указывает на SHA-512, за которым следуют соль и результат хэширования. Это обеспечивает дополнительный уровень защиты, усложняя задачу злоумышленника даже при наличии доступа к файлу.

С точки зрения безопасности, шифрование паролей подвержено рискам, связанным с устаревшими алгоритмами (например, MD5 все еще встречается в некоторых системах) или недостаточной длиной соли. Кроме того, слабые пароли остаются уязвимыми к атакам перебора, несмотря на использование современных методов хэширования. Методически изучение шифрования паролей следует начинать с анализа эволюции алгоритмов – от DES к SHA-512 – с акцентом на их криптографические свойства. Студентам рекомендуется изучить структуру записей в /etc/shadow и процесс генерации хэшей с помощью утилит, таких как mkpasswd. Лабораторная часть может включать создание тестовых паролей с различными алгоритмами шифрования и их анализ на устойчивость к атакам с использованием инструментов, таких

как John the Ripper, что позволяет понять сильные и слабые стороны каждого метода.

Шифрование паролей в Linux направлено на предотвращение их раскрытия даже при компрометации файла /etc/shadow.

Алгоритмы хэширования

- Исторические алгоритмы:

DES: Устаревший метод с длиной соли 2 символа.
MD5: Более стойкий, но уязвимый к коллизиям.

- Современные стандарты:

SHA-256/SHA-512: Используются по умолчанию в современных дистрибутивах. Хэш генерируется с солью длиной 16 символов.
yescrypt: Алгоритм, устойчивый к GPU-атакам, применяется в новых версиях Linux.

Соль (salt) и её роль

Соль – это случайная строка, добавляемая к паролю перед хэшированием. Она исключает возможность использования радужных таблиц (rainbow tables) для массового взлома паролей. Например, пароль secret с солью abcd превращается в хэш sha512(abcdsecret), уникальный для каждого пользователя.

Пример генерации хэша

Использование утилиты mkpasswd для создания SHA-512 хэша:

```
bash
Copy
mkpasswd -m sha-512 -S "s0m3s4lt" "my_password"
# Результат: $6$s0m3s4lt$4h7Hr...
```

Уязвимости и защита

- **Атаки по словарю**
- Использование инструментов вроде John the Ripper для подбора паролей. Защита – политика сложности и регулярная смена паролей.
- **Перехват в памяти**. Технологии вроде TRESOR защищают ключи шифрования от извлечения через DMA-атаки.

Методические указания по изучению темы

Освоение темы «Идентификация и аутентификация в ОС Linux» требует последовательного подхода, сочетающего теоретическую базу с практическими навыками. На первом этапе студенты должны разобраться в стандартной процедуре идентификации и аутентификации, изучив взаимодействие файлов /etc/passwd, /etc/shadow и модулей PAM. Анализ реального процесса входа в систему (например, через терминал или SSH) с использованием логов /var/log/auth.log поможет понять последовательность действий и точки контроля безопасности. Лабораторная работа на этом этапе может включать настройку PAM для различных сценариев аутентификации,

таких как обязательное использование пароля и дополнительного фактора, с последующим анализом результатов.

Переходя к основным механизмам защиты паролей, студенты должны сосредоточиться на изучении структуры файла /etc/shadow и роли PAM-модулей в обеспечении безопасности. Рекомендуется рассмотреть примеры конфигураций /etc/pam.d/passwd и /etc/security/pwquality.conf, изменяя параметры сложности паролей и оценивая их влияние на устойчивость к атакам. Практическое задание может предусматривать моделирование попыток несанкционированного доступа к файлу /etc/shadow с изменением прав (chmod/chown) и анализ последствий для системы, что закрепляет понимание ограничений доступа.

Изучение шифрования паролей начинается с анализа криптографических алгоритмов, их эволюции и применения в Linux. Студентам следует освоить генерацию хэшей с помощью утилит, таких как mkpasswd или openssl, и изучить влияние соли на безопасность. Лабораторная часть может включать тестирование устойчивости паролей к атакам перебора с использованием инструментов, таких как Hashcat или John the Ripper, с последующим сравнением результатов для разных алгоритмов (MD5, SHA-512). Для интеграции знаний рекомендуется провести итоговое занятие, где студенты представят анализ безопасности процесса аутентификации в Linux, предложив рекомендации по его улучшению, основанные на изученных механизмах и методах шифрования.

При работе над курсовой работой обратите внимание на:

– описание особенностей механизма аутентификации в ОС, которые требуется приводить в первой (теоретической) главе (в соответствии с темой курсовой работы).

Литература

1. Окороков, В. А. Безопасность операционных систем / В. А. Окороков. – Санкт-Петербург: Лань, 2024. – 228 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/367472 (дата обращения: 06.12.2024). – ISBN 978-5-507-48297-9. – Текст : электронный (гл. 7, с. 196–201).

2. Киренберг, Г. А. Информационная безопасность современных операционных систем: учеб. пособие / Киренберг, Г. А. – Кемерово: КузГТУ им. Т. Ф. Горбачева, 2022. – 138 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/295736 (дата обращения: 06.12.2024). – ISBN 978-5-00137-320-9. – Текст : электронный. (Глава 3)

3. Потерпеев, Г. Ю. Безопасность операционных систем: учеб. пособие / Г. Ю. Потерпеев, В. С. Нефедов, А. А. Криулин. – Москва: РТУ МИРЭА, 2021. – 93 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/182416

(дата обращения: 06.12.2024). – ISBN 978-5-7339-1393-3. – Текст : электронный (гл. 1, 4).

4. Зверева, О. М. Операционные системы: учеб. пособие / О. М. Зверева; науч. ред. Л. Г. Доросинский; Уральский федеральный университет им. первого Президента России Б. Н. Ельцина. – Екатеринбург: Издательство Уральского университета, 2020. – 223 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=699030 (дата обращения: 03.11.2024). – ISBN 978-5-7996-3146-8. – Текст : электронный (гл. 11).

Контрольные вопросы

1. Приведите особенности аутентификации в ОС Linux.
2. Приведите особенности использования паролей в ОС Linux.
3. Приведите особенности механизмы контроля паролей в ОС Linux.
4. Приведите особенности моделей прав доступа в ОС Linux.
5. Как стандартная процедура идентификации и аутентификации в Linux использует файлы /etc/passwd и /etc/shadow, и какие уязвимости могут возникнуть при их некорректной настройке?
6. Каким образом модули PAM обеспечивают гибкость аутентификации, и как ошибки в их конфигурации могут повлиять на безопасность системы?
7. Почему разделение данных между файлами /etc/passwd и /etc/shadow повышает безопасность паролей, и какие меры защищают файл /etc/shadow от несанкционированного доступа?
8. Как механизмы контроля качества паролей, такие как pam_passwdqc, снижают риск атак методом перебора, и какие ограничения они имеют?
9. В чем преимущества использования SHA-512 для шифрования паролей по сравнению с DES и MD5, и как соль усиливает криптографическую стойкость?
10. Какие угрозы безопасности возникают при использовании устаревших алгоритмов шифрования паролей, и как их можно выявить в современной системе Linux?
11. Как ограничение срока действия паролей в /etc/shadow влияет на защиту системы, и какие проблемы могут возникнуть при его неправильной настройке?
12. Каким образом злоумышленник может использовать слабые пароли для компрометации системы, и какие методы тестирования позволяют оценить их устойчивость к атакам?

Тема 2.5 Идентификация и аутентификация **В ОС Windows**

Перечень изучаемых вопросов

1. Стандартная процедура идентификации и аутентификации.
2. Основные механизмы защиты паролей.
3. Шифрование паролей.

Методические указания к изучению

Верхний уровень подсистемы аутентификации Windows включает в себя процесс аутентификации WinLogon.exe и так называемые библиотеки-провайдеры или просто провайдеры – заменяемые библиотеки, реализующие большую часть высокоуровневых функций процесса аутентификации.

При выполнении второго этапа данной процедуры WinLogon использует привилегии псевдопользователя SYSTEM создавать маркеры доступа и выступать от имени ОС, а при выполнении третьего этапа – привилегию назначать процессам маркеры доступа.

Нижний уровень подсистемы аутентификации отвечает за хранение учетной информации о пользователях, в том числе и эталонных образов паролей. При аутентификации нижний уровень передает среднему уровню эталонный образ пароля пользователя, а при авторизации - список групп и привилегий пользователя.

Для генерации образа пароля стандартный пакет аутентификации MSV 1.0 применяет хэш-функцию MD4. Для совместимости с более ранними версиями Windows MSV 1.0 поддерживает другой формат образа пароля.

Администратор определяет, могут ли пользователи самостоятельно менять пароль в случае истечения максимального срока его действия или они должны уведомлять его об этом.

Механизм автоматической блокировки (lock out) при превышении максимально допустимого количества неудачных попыток входа в систему не распространяется на пользователя Administrator.

Подсистема аутентификации Windows состоит из нескольких программных модулей, связанных между собой, и разделена на три уровня. Средний уровень подсистемы аутентификации пользуется услугами нижнего уровня и предоставляет услуги верхнему.

Верхний уровень подсистемы аутентификации Windows включает в себя процесс аутентификации WinLogon.exe и так называемые библиотеки-провайдеры или просто провайдеры – заменяемые библиотеки, реализующие большую часть высокоуровневых функций процесса аутентификации.

WinLogon представляет собой обычный процесс Win32 API, выполняющийся от имени псевдопользователя SYSTEM. WinLogon автоматически запускается при старте ОС и остается активным до выключения питания или перезагрузки.

При входе пользователя в систему с локальной консоли в качестве провайдера по умолчанию выступает библиотека msgina.dll, которая осуществляет все взаимодействия между локальным пользователем и процессом аутентификации.

Вход пользователя в ОС производится следующим образом.

1. Провайдер получает от пользователя идентифицирующую и аутентифицирующую информацию.

2. Провайдер осуществляет аутентификацию, передавая имя и пароль на средний уровень подсистемы аутентификации с помощью системного вызова LogonUser. При этом, если аутентификация прошла успешно, создается маркер доступа пользователя.

3. Если маркер доступа пользователя создан успешно, провайдер осуществляет авторизацию пользователя, запуская процесс Userlnit.exe от имени аутентифицированного пользователя.

4. Процесс Userlnit загружает индивидуальные настройки пользователя из его профиля (profile), монтирует ключ реестра, соответствующий данному пользователю, и загружает программную среду пользователя. После этого Userlnit завершает работу.

При выполнении второго этапа данной процедуры WinLogon использует привилегии псевдопользователя SYSTEM создавать маркеры доступа и выступать от имени ОС, а при выполнении третьего этапа - привилегию назначать процессам маркеры доступа. Таким образом, если эти привилегии не будут предоставлены псевдопользователю SYSTEM, вход пользователей в систему станет невозможен.

В средний уровень подсистемы аутентификации входят локальный распорядитель безопасности (local security authority, LSA) и пакеты аутентификации - заменяемые библиотеки, реализующие большую часть низкоуровневых функций аутентификации.

Так же, как и WinLogon, LSA представляет собой обычный процесс (по имени lsass.exe), выполняющийся от имени псевдопользователя SYSTEM. Аварийное завершение LSA приводит к аварийному завершению работы всей ОС. Как и WinLogon, LSA передоверяет большинство своих функций заменяемым библиотекам. Стандартная схема аутентификации реализуется пакетом MSV 1.0 (msv1_0.dll), могут быть установлены и другие пакеты аутентификации.

Пакет аутентификации осуществляет аутентификацию пользователя в процессе обработки системного вызова LogonUser. Аутентификация производится следующим образом.

1. Пакет аутентификации получает от верхнего уровня имя и пароль пользователя и генерирует образ пароля.

2. Используя услуги нижнего уровня, пакет аутентификации получает эталонный образ пароля и сравнивает его с образом пароля из п. 1.

3. При совпадении образов паролей LSA получает от нижнего уровня информацию о том, может ли данный пользователь начинать в данный момент работу с данной рабочей станцией.

4. При положительном результате проверки LSA формирует маркер доступа пользователя, получая необходимую информацию от нижнего уровня подсистемы аутентификации.

5. LSA передает сформированный маркер доступа верхнему уровню подсистемы аутентификации.

Нижний уровень подсистемы аутентификации отвечает за хранение учетной информации о пользователях, в том числе и эталонных образов паролей. При аутентификации нижний уровень передает среднему уровню эталонный образ пароля пользователя, а при авторизации - список групп и привилегий пользователя.

При стандартной конфигурации ОС нижний уровень подсистемы аутентификации включает в себя систему управления списком пользователей (Security Account Manager, SAM) и сервис NetLogon. SAM используется для извлечения информации из реестра локального компьютера, а NetLogon - информации из реестра контроллера домена. Администраторы системы могут устанавливать и другие сервисы аналогичного назначения.

Механизм автоматической блокировки (lock out) при превышении максимально допустимого количества неудачных попыток входа в систему не распространяется на пользователя Administrator.

Для пользователей могут быть установлены следующие флаги:

− пользователь обязан сменить пароль при ближайшем входе в систему (для вновь зарегистрированных пользователей);

− пользователь не может менять свой пароль (применяется для «групповых» пользователей (Guest, Anonymous и т. д.));

− на пользователя не распространяется ограничение максимального срока действия пароля (применяется в совокупности с предыдущим требованием);

− пользователь не может работать в системе (применяется для временного блокирования учетной записи пользователя).

Кроме того, могут быть введены следующие требования к процедуре авторизации пользователя. Может быть явно указан путь к профилю (profile) пользователя. В этом случае индивидуальные настройки пользователя будут загружаться не из системной директории локального компьютера, а из той директории того компьютера, которая указана в пути к профилю. В результате индивидуальные настройки пользователя могут быть сделаны одинаковыми для нескольких компьютеров. Пользователю может быть назначен скрипт (программа или командный файл), который будет автоматически выполняться при каждом входе пользователя в систему, локальном или удаленном. Пользователю может быть назначена домашняя директория, которая становится текущей по умолчанию для всех его программ.

Для пользователей домена Windows могут быть введены следующие дополнительные требования к процедурам идентификации, аутентификации и авторизации:

− время работы пользователя с ОС может быть ограничено;

− количество компьютеров, с которых пользователь может входить в домен, может быть ограничено (до восьми компьютеров);

− может быть установлена автоматическая блокировка учетной записи пользователя по истечении определенного времени;

− пользователю может быть запрещен интерактивный вход на любой компьютер домена; в этом случае пользователь может работать с доменом только извне.

Помимо вышеперечисленных требований и ограничений при идентификации и аутентификации пользователя также осуществляется проверка одной из следующих «привилегий»:

− входить в систему интерактивно;

− входить в систему через SMB-сервер;

− запускать сервис от своего имени;

− запускать от своего имени пакетное задание.

Альтернативные схемы идентификации и аутентификации. Поскольку и провайдеры, и пакеты аутентификации являются заменяемыми компонентами подсистемы аутентификации, администратор ОС может, установив нестандартный провайдер и/или пакет аутентификации, реализовать в Windows NT любую другую схему аутентификации. Для этого необходимо разместить в системной директории Windows NT необходимые библиотеки и внести изменения в соответствующие ключи реестра.

При этом в качестве аутентифицирующей информации может использоваться произвольная строка Unicode длиной до 128 символов.

Защита паролей в Windows реализуется через политики безопасности, технологии шифрования и дополнительные методы аутентификации.

Политики паролей:

- **Групповые политики (GPO)**: Настраиваются через gpedit.msc или доменные GPO. Основные параметры:

a. Минимальная длина пароля (рекомендуется 12 символов).

b. Требование сложности (цифры, буквы разных регистров, спецсимволы).

c. Максимальный срок действия (90 дней).

d. Запрет повторного использования паролей (24 предыдущих).

- **Блокировка учетных записей**: Активируется после N неудачных попыток входа (по умолчанию – 5 попыток за 30 мин).

Дополнительные методы аутентификации:

- **Windows Hello**. Использует биометрию (отпечаток пальца, распознавание лица) или PIN-код, привязанный к устройству.
- **Аппаратные ключи**. Поддержка FIDO2-устройств (YubiKey) для входа без пароля.
- **Многофакторная аутентификация (MFA)**. Интеграция с Azure AD MFA для подтверждения через мобильное приложение или SMS.

Credential Guard. Технология, изолирующая хэши паролей и секреты в виртуализированной среде (Virtual Secure Mode), чтобы предотвратить их кражу вредоносным ПО.

Шифрование паролей

Windows использует несколько методов шифрования для защиты паролей как в локальных, так и в доменных средах.

Локальное хранение паролей:

- **SAM-файл**: Хранит хэши паролей в форматах LM (устаревший) и NTLM. Для шифрования SAM используется системный ключ (Syskey), который сохраняется в реестре или на USB-носителе.
- **NTLM-хэш**: Алгоритм MD4, применяемый к паролю в кодировке Unicode. Например, пароль P@ssw0rd преобразуется в хэш C23413A8A1E7665FAAD3B435B51404EE.

Доменные среды (Active Directory):

- **Kerberos**: Использует симметричное шифрование AES или RC4. Пароли хранятся в виде хэшей, но аутентификация выполняется через билеты (tickets), защищенные сессионными ключами.
- **Атрибут unicodePwd**: В AD пароли хранятся в зашифрованном виде с использованием хэша NTLMv2 или Kerberos AES.

Защита от атак:

- **LSA Protection**: Защищает LSASS от инъекции кода и дампа памяти.
- **Salting**: В современных версиях Windows (10/11) используется соль для генерации хэшей, что усложняет атаки по радужным таблицам.

Пример уязвимости: Протокол LM (Lan Manager) сохранял пароли в двух частях по 7 символов, что упрощало брутфорс. В современных системах LM отключен по умолчанию.

Методические указания по изучению темы

Теоретический анализ:

1. Изучите архитектуру LSASS и его роль в аутентификации.
2. Сравните протоколы NTLM и Kerberos: скорость, безопасность, сценарии использования.
3. Исследуйте работу Credential Guard и Virtual Secure Mode.

Лабораторные задания:

1. Настройка групповых политик паролей:

a. Откройте gpedit.msc → Конфигурация компьютер → Политики → Параметры безопасности → Политики учетных записей.

b. Установите минимальную длину пароля 10 символов и требование сложности.

Анализ SAM-файла. Используйте утилиту mimikatz (в контролируемой среде) для извлечения хэшей.

2. Включение Windows Hello. Настройте вход по отпечатку пальца в параметрах учетной записи.

При работе над курсовой работой обратите внимание на:

– описание особенностей механизма аутентификации в ОС, которые требуется приводить в первой (теоретической) главе (в соответствии с темой курсовой работы).

Литература:

1. Окороков, В. А. Безопасность операционных систем / В. А. Окороков. – Санкт-Петербург: Лань, 2024. – 228 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/367472 (дата обращения: 06.12.2024). – ISBN 978-5-507-48297-9. – Текст : электронный (гл. 7).

2. Киренберг, Г. А. Информационная безопасность современных операционных систем: учеб. пособие / Киренберг, Г. А. – Кемерово: КузГТУ им. Т. Ф. Горбачева, 2022. – 138 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/295736 (дата обращения: 06.12.2024). – ISBN 978-5-00137-320-9. – Текст : электронный (гл. 3).

3. Потерпеев, Г. Ю. Безопасность операционных систем: учеб. пособие / Г. Ю. Потерпеев, В. С. Нефедов, А. А. Криулин. – Москва: РТУ МИРЭА, 2021. – 93 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/182416 (дата обращения: 06.12.2024). – ISBN 978-5-7339-1393-3. – Текст : электронный (гл. 4).

4. Зверева, О. М. Операционные системы: учеб. пособие / О. М. Зверева; науч. ред. Л. Г. Доросинский; Уральский федеральный университет им. первого Президента России Б. Н. Ельцина. – Екатеринбург: Издательство Уральского университета, 2020. – 223 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=699030 (дата обращения: 03.11.2024). – ISBN 978-5-7996-3146-8. – Текст : электронный (гл. 11).

Контрольные вопросы:

1. Приведите особенности аутентификации в ОС Windows.
2. Приведите особенности использования паролей в ОС Windows.
3. Приведите особенности механизмы контроля паролей в ОС Windows.
4. Приведите особенности моделей прав доступа в ОС Windows.

Тема 2.6 Аудит ОС

Перечень изучаемых вопросов

1. Процедура аудита применительно к ОС.
2. Требования к аудиту операционной системы.
3. Политика аудита.
4. Реализация аудита в UNIX.
5. Реализация аудита в Windows.

Методические указания к изучению

Политика аудита представляет собой совокупность правил, определяющих то, какие события должны регистрироваться в журнале аудита.

Для обеспечения надежной защиты ОС 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.

Система контроля регистрирует события в ОС, связанные с защитой информации, записывая их в контрольный журнал, в котором возможна фиксация проникновения в систему и неправильного использования ресурсов.

Политика аудита. Множество событий, информация о которых записывается в журнал аудита, определяется политикой аудита, которую назначают пользователи-аудиторы. Windows позволяет регистрировать в журнале аудита события следующих категорий:

• вход / выход пользователя из системы;
• доступ субъектов к объектам;
• использование субъектами доступа опасных привилегий;
• изменения в списке пользователей;
• изменения в политике безопасности;
• системные события;
• запуск и завершение процессов.

Считаются опасными следующие привилегии субъектов:

• получать оповещения от файловой системы;
• добавлять записи в журнал аудита;
• создавать маркеры доступа;
• назначать маркеры доступа процессам;
• создавать резервные копии информации;
• восстанавливать информацию на дисках с резервных копий;
• отлаживать программы.

Одним из наиболее сложных вопросов является вопрос о том, какой должна быть политика аудита. Политика аудита настолько сильно связана с особенностями эксплуатации конкретного экземпляра ОС, что сформулировать эталонную политику просто невозможно. Даже для конкретного экземпляра ОС нельзя сформулировать адекватную политику аудита «на все времена». Политика аудита должна постоянно меняться, реагируя на изменения в конфигурации ОС и на зарегистрированные опасные события.

При определении политики аудита следует иметь в виду, что адекватность политики заключается в том, что регистрируется ровно столько событий, сколько необходимо. Если подсистема аудита регистрирует слишком много событий, то, с одной стороны, журнал аудита переполняется слишком быстро, а с другой – аудитору трудно выделить в огромном объеме информации важные события.

Тема «Аудит операционных систем» занимает особое место в курсе «Безопасность операционных систем», поскольку аудит представляет собой фундаментальный инструмент для мониторинга, анализа и обеспечения

безопасности системного уровня. В условиях нарастающих киберугроз, таких как несанкционированный доступ, эксплуатация уязвимостей и внутренние инциденты, аудит становится неотъемлемой частью управления безопасностью, позволяя выявлять подозрительные действия, отслеживать изменения и обеспечивать соответствие нормативным требованиям. В рамках данной темы рассматриваются процедура аудита применительно к операционным системам, требования к его проведению, политика аудита, а также конкретные подходы к реализации аудита в UNIX и Windows. Эти вопросы раскрывают как теоретические основы, так и Лабораторные аспекты применения аудита, подчеркивая его роль в обнаружении угроз и повышении защищенности ОС. Настоящая статья представляет собой развернутый анализ указанных вопросов, дополненный методическими рекомендациями, которые направлены на глубокое освоение материала студентами и развитие у них аналитических и практических навыков в области информационной безопасности.

Процедура аудита применительно к ОС

Аудит операционной системы представляет собой систематический процесс сбора, анализа и интерпретации данных о событиях, происходящих в системе, с целью оценки ее безопасности и выявления потенциальных инцидентов. Процедура аудита начинается с определения целей, которые могут включать мониторинг доступа к файлам, отслеживание действий пользователей, регистрацию изменений конфигурации или обнаружение попыток эксплуатации уязвимостей. На этом этапе администратор или специалист по безопасности определяет, какие события подлежат фиксации, исходя из специфики системы и уровня риска, связанного с ее эксплуатацией.

Следующим шагом процедуры становится настройка механизмов аудита, встроенных в ОС, которые обеспечивают регистрацию событий в виде логов. Эти механизмы включают в себя системные вызовы, службы журналирования и специализированные утилиты, которые фиксируют данные о действиях пользователей, процессов и ядра. Например, в ОС могут регистрироваться попытки входа в систему, запуск программ, изменения прав доступа или ошибки выполнения операций. Собранные данные сохраняются в лог-файлах, таких как системные журналы или специализированные базы данных аудита, которые затем анализируются для выявления аномалий. Анализ может проводиться вручную или с использованием автоматизированных инструментов, таких как системы управления событиями безопасности (SIEM), которые коррелируют события и выявляют паттерны, указывающие на угрозы.

Заключительный этап процедуры аудита предполагает интерпретацию результатов и принятие мер на основе полученной информации. Это может включать усиление политики безопасности, устранение уязвимостей или расследование инцидентов с привлечением дополнительных ресурсов. С точки зрения безопасности, процедура аудита уязвима к атакам, направленным на подмену или удаление логов, что требует защиты самих данных аудита. Методически изучение этого вопроса следует начинать с анализа целей и этапов аудита, рассматривая их на примере реальных сценариев, таких как

мониторинг действий администратора или обнаружение несанкционированного доступа. Студентам полезно изучить структуру типичных лог-файлов и процесс их генерации в ОС, а лабораторная работа может включать настройку базового аудита с последующим анализом логов для выявления подозрительных событий, что закрепляет понимание процедуры и ее роли в обеспечении безопасности.

Аудит ОС включает последовательность этапов, направленных на обеспечение прозрачности и контроля над системными процессами. Первым шагом является планирование, где определяются цели аудита: проверка соответствия политикам безопасности, обнаружение аномалий или подготовка к сертификации. На этом этапе выбираются ключевые объекты аудита — пользовательские действия, доступ к файлам, изменения реестра, сетевые соединения.

Далее следует сбор данных с использованием встроенных инструментов ОС (например, auditd в Linux или Event Viewer в Windows) и сторонних решений (Splunk, Graylog). Данные записываются в логи, которые должны быть защищены от модификации с помощью механизмов целостности (хеширование, цифровые подписи).

Анализ данных предполагает поиск паттернов, указывающих на угрозы: множественные неудачные попытки входа, несанкционированный доступ к чувствительным файлам, подозрительные процессы. Для автоматизации применяются SIEM-системы (Security Information and Event Management), которые агрегируют данные и генерируют оповещения.

Завершающий этап — формирование отчёта, включающего выявленные уязвимости, рекомендации по устранению рисков и доказательства соответствия стандартам (ISO 27001, PCI DSS). Отчёт служит основой для корректировки политик безопасности и проведения профилактических мероприятий.

Аудит ОС должен соответствовать ряду требований, обеспечивающих его эффективность и легитимность:

1. Полнота: регистрация всех критических событий, включая входы в систему, доступ к данным, изменения прав доступа и запуск привилегированных команд.

2. Невозможность отказа (Non-repudiation): гарантия того, что пользователь не сможет отрицать совершённые действия. Достигается через привязку событий к уникальным идентификаторам (SID в Windows, UID в UNIX).

3. Конфиденциальность и целостность: логи должны храниться в зашифрованном виде, а доступ к ним – ограничен. Для проверки целостности используются алгоритмы хеширования (SHA-256).

4. Своевременность: реальное время мониторинга и оповещения о подозрительных событиях.

5. Соответствие стандартам: выполнение требований GDPR (хранение персональных данных), PCI DSS (защита платёжных данных), NIST SP 800-53.

Требования к аудиту операционной системы формируются на основе нормативных документов, стандартов безопасности и специфики эксплуатации

ОС. Основным требованием является полнота охвата событий, подлежащих регистрации, что подразумевает фиксацию всех значимых действий, влияющих на безопасность системы. Это включает мониторинг входов и выходов пользователей, изменений конфигурационных файлов, запуска привилегированных процессов и доступа к критическим ресурсам. Полнота обеспечивается правильной настройкой подсистемы аудита, которая должна быть достаточно гибкой, чтобы адаптироваться к различным уровням риска и задачам организации.

Другим важным требованием выступает достоверность данных аудита, гарантирующая, что зарегистрированные события точно отражают реальные действия в системе. Это достигается за счет защиты логов от модификации или удаления, например, через ограничение прав доступа к файлам журналов и использование механизмов цифровой подписи или хэширования для проверки их целостности. Требование своевременности подразумевает, что события фиксируются в реальном времени или с минимальной задержкой, что позволяет оперативно реагировать на инциденты. Кроме того, данные аудита должны быть доступны для анализа в удобном формате, что требует их структурирования и хранения в читаемом виде, таком как текстовые файлы или базы данных с возможностью поиска.

С точки зрения безопасности, требования к аудиту включают обеспечение конфиденциальности логов, поскольку они могут содержать чувствительную информацию, такую как имена пользователей или параметры операций. Также важно соответствие нормативным стандартам, например, ISO 27001 или PCI DSS, которые предписывают определенные правила ведения аудита в системах, обрабатывающих финансовые или персональные данные. Методически изучение этого вопроса предполагает анализ нормативных требований и их влияния на настройку аудита. Студентам следует рассмотреть примеры стандартов и их применимость к ОС, а также изучить влияние требований на архитектуру системы. Лабораторная работа может включать разработку набора требований для аудита тестовой ОС с последующей проверкой их реализации, что помогает понять баланс между полнотой, достоверностью и производительностью.

Политика аудита представляет собой формализованный набор правил и настроек, определяющих, какие события подлежат регистрации, как они обрабатываются и хранятся в операционной системе. Она разрабатывается на основе целей безопасности организации и требований, предъявляемых к системе, и служит руководством для настройки подсистемы аудита. Политика включает определение категорий событий, таких как попытки аутентификации, изменения файлов, запуск процессов или сетевые подключения, а также указание уровня детализации — от минимального (только критические события) до расширенного (все действия пользователей и системы). Например, в высокорисковой среде политика может предписывать регистрацию всех операций с привилегированными учетными записями, тогда как в менее критичных системах достаточно отслеживать только неудачные попытки входа.

Кроме того, политика аудита регулирует управление логами, включая их ротацию, архивирование и сроки хранения. Это важно для предотвращения переполнения дискового пространства и обеспечения доступности данных для анализа в случае инцидента. Политика также определяет меры защиты логов от несанкционированного доступа или подделки, например, через использование шифрования или перенаправление данных на удаленный сервер. С точки зрения безопасности, политика аудита должна быть сбалансирована: избыточная детализация может привести к снижению производительности системы, тогда как недостаточная — к упущению важных событий.

Методически изучение политики аудита следует начинать с анализа ее структуры и целей, рассматривая примеры реальных политик для разных типов систем (серверов, рабочих станций). Студентам полезно изучить влияние различных настроек на объем и содержание логов, а также их роль в расследовании инцидентов. Лабораторная работа может включать разработку политики аудита для тестовой среды с последующей настройкой системы и анализом результатов, что позволяет оценить эффективность выбранных параметров и их соответствие требованиям безопасности.

Реализация аудита в UNIX-системах, таких как Linux или FreeBSD, базируется на встроенных подсистемах, обеспечивающих гибкость и детализированный контроль событий. Одним из основных инструментов в Linux является подсистема Audit, интегрированная в ядро и управляемая через утилиту auditctl. Эта подсистема позволяет настраивать правила аудита для регистрации системных вызовов (например, open, execve), изменений файлов и действий пользователей. Логи аудита сохраняются в файлы, такие как /var/log/audit/audit.log, и содержат подробную информацию о времени события, пользователе, процессе и результате операции. Управление правилами осуществляется через конфигурационные файлы в /etc/audit/audit.rules, где можно указать, например, мониторинг доступа к критическим файлам, таким как /etc/passwd.

Дополнительно в UNIX используются традиционные механизмы журналирования, такие как syslog, которые фиксируют системные события в файлах /var/log/messages или /var/log/syslog. Для анализа логов применяются утилиты, такие как ausearch и aureport, которые позволяют фильтровать события по заданным критериям и генерировать отчеты. С точки зрения безопасности, реализация аудита в UNIX уязвима к атакам на логи (удаление или подмена), а также к перегрузке системы при чрезмерной детализации правил. Методически изучение этого вопроса предполагает анализ работы подсистемы Audit на примере Linux, начиная с настройки базовых правил и заканчивая интерпретацией логов. Студентам следует рассмотреть примеры конфигураций audit.rules и их влияние на производительность. Практическая работа может включать настройку аудита для мониторинга изменений в /etc/shadow с последующим анализом логов через ausearch, что демонстрирует возможности UNIX в выявлении угроз.

Реализация аудита в UNIX

В UNIX-системах аудит реализуется через подсистему **auditd**, которая настраивается файлами в /etc/audit/.

Ключевые компоненты:

- **auditctl**: Утилита для управления правилами аудита. Например, команда auditctl -w /etc/passwd -p wa -k user_changes отслеживает запись и изменение атрибутов файла /etc/passwd, помечая события ключом «user_changes».
- **ausearch**: Поиск событий по фильтрам (время, ключ, тип).
- **aureport**: Генерация сводных отчётов (статистика по пользователям, системным вызовам).

Пример правила для отслеживания использования sudo:

```
bash
Copy
auditctl -a always,exit -F arch=b64 -S execve -F path=/usr/bin/sudo -k admin_actions
```

Это правило регистрирует все запуски sudo, сохраняя их в лог с меткой «admin_actions».

Интеграция с rsyslog: Логи могут дублироваться на удалённый сервер для предотвращения потери данных при компрометации локальной системы.

Реализация аудита в Windows

Реализация аудита в Windows основана на встроенной подсистеме Windows Event Logging, которая регистрирует события в журналах, доступных через Event Viewer. Аудит настраивается через локальные политики безопасности (secpol.msc) или групповые политики (GPO) в доменной среде, позволяя фиксировать такие категории событий, как вход в систему, доступ к объектам (файлам, реестру), изменения политик и запуск процессов. Логи сохраняются в журналах Security, System и Application, расположенных в формате EVT/EVTX в каталоге C:\Windows\System32\winevt\Logs. Например, журнал Security фиксирует попытки аутентификации, указывая имя пользователя, время и результат операции.

Для управления аудитом в Windows используется механизм SACL (System Access Control List), который применяется к объектам (файлам, папкам, ключам реестра) и определяет, какие действия с ними подлежат регистрации. Анализ логов осуществляется через Event Viewer или с помощью PowerShell-командлетов, таких как Get-EventLog или Get-WinEvent, которые позволяют фильтровать события и экспортировать их для дальнейшего изучения. С точки зрения безопасности, реализация аудита в Windows уязвима к очистке логов (например, через wevtutil cl) или переполнению журнала при недостаточном размере хранилища. Методически изучение этого вопроса начинается с анализа структуры журналов и настройки аудита через GPO. Студентам полезно рассмотреть настройку SACL для файла и анализ событий в Event Viewer. Практическая работа может включать настройку аудита входа в систему с последующим моделированием неудачных попыток и анализом логов, что подчеркивает роль аудита в расследовании инцидентов.

В Windows аудит настраивается через **Групповые политики** (GPO) и **Локальные политики безопасности**.

Основные инструменты:

1. **Event Viewer**: Просмотр логов из категорий:
   a. Security: События входа, изменения прав доступа.
   b. System: Ошибки драйверов, запуск служб.
   c. Application: События приложений.

2. **Auditpol**: Командная строка для управления политиками. Например, auditpol /set /category:«Object Access» /success:enable /failure: enable активирует аудит доступа к объектам.

3. **PowerShell**: Скрипты для автоматизации сбора логов (Get-EventLog, Get-WinEvent).

Настройка аудита доступа к файлам:

- В свойствах файла или папки → Вкладка "Безопасность" → Дополнительно → Аудит → Добавить субъекта (пользователь/группа) → Выбрать типы доступа (чтение, запись).

Интеграция с Active Directory: В доменных средах политики аудита централизованно применяются через GPO, что упрощает управление в крупных организациях.

Дополнительные методические указания по изучению темы

Освоение темы «Аудит ОС» требует последовательного подхода, объединяющего теоретическую базу с практическими навыками. На первом этапе студенты должны изучить процедуру аудита, анализируя ее этапы – от определения целей до интерпретации логов — на примере типичных сценариев, таких как мониторинг доступа к файлам. Лабораторная работа на этом этапе может включать настройку базового аудита в тестовой ОС с последующим анализом сгенерированных логов, что помогает понять процесс и его значение для безопасности.

Переходя к требованиям к аудиту, студенты должны сосредоточиться на изучении нормативных стандартов (например, ISO 27001) и их влияния на настройку системы. Анализ требований полноты, достоверности и своевременности следует дополнить рассмотрением их реализации в реальных ОС. Практическое задание может предусматривать разработку требований для аудита тестовой системы с последующей проверкой их выполнения, что закрепляет понимание баланса между функциональностью и производительностью.

Изучение политики аудита предполагает анализ ее структуры и влияния на регистрацию событий. Студентам следует рассмотреть примеры политик для разных типов систем, изучая их настройку и влияние на объем логов. Лабораторная работа может включать создание политики аудита для тестовой среды с последующим анализом результатов через логи, что демонстрирует ее роль в управлении безопасностью.

При изучении реализации аудита в UNIX студенты должны погрузиться в работу подсистемы Audit, анализируя настройку правил через auditctl и

интерпретацию логов с помощью ausearch. Практическая часть может включать мониторинг изменений в системных файлах с последующим анализом результатов, что подчеркивает гибкость UNIX. Для Windows следует сосредоточиться на настройке аудита через GPO и SACL, изучая журналы через Event Viewer и PowerShell. Практическое задание может предусматривать настройку аудита доступа к файлу с моделированием действий и анализом логов, что демонстрирует возможности Windows.

Для интеграции знаний рекомендуется провести итоговый семинар, где студенты представят сравнительный анализ аудита в UNIX и Windows, предложив рекомендации по улучшению безопасности на основе изученных подходов. Такой подход обеспечивает глубокое освоение материала и развитие аналитических навыков.

При работе над курсовой работой обратите внимание на:

– описание особенностей механизма аутентификации в ОС, которые требуется приводить в первой (теоретической) главе (в соответствии с темой курсовой работы).

Литература

1. Окороков, В. А. Безопасность операционных систем / В. А. Окороков. – Санкт-Петербург: Лань, 2024. – 228 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/367472 (дата обращения: 06.12.2024). – ISBN 978-5-507-48297-9. – Текст : электронный (гл. 7).

2. Киренберг, Г. А. Информационная безопасность современных операционных систем: учеб. пособие / Киренберг, Г. А. – Кемерово: КузГТУ имени Т.Ф. Горбачева, 2022. – 138 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/295736 (дата обращения: 06.12.2024). – ISBN 978-5-00137-320-9. – Текст : электронный (гл. 3).

3. Потерпеев, Г. Ю. Безопасность операционных систем: учеб. пособие / Г. Ю. Потерпеев, В. С. Нефедов, А. А. Криулин. – Москва: РТУ МИРЭА, 2021. – 93 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/182416 (дата обращения: 06.12.2024). – ISBN 978-5-7339-1393-3. – Текст : электронный (гл. 5).

4. Зверева, О. М. Операционные системы: учеб. пособие / О. М. Зверева; науч. ред. Л. Г. Доросинский; Уральский федеральный университет им. первого Президента России Б. Н. Ельцина. – Екатеринбург: Издательство Уральского университета, 2020. – 223 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=699030 (дата обращения: 03.11.2024). – ISBN 978-5-7996-3146-8. – Текст : электронный (гл. 11)

Контрольные вопросы

1. Приведите особенности аудита в ОС Windows.
2. Приведите особенности аудита в ОС Linux.
3. Приведите особенности построения политики аудита в ОС.
4. Приведите особенности процедур аудита в ОС.
5. Какие этапы включает процедура аудита ОС и как они связаны с обнаружением инцидентов?
6. Объясните, почему аудит должен соответствовать принципу «невозможности отказа».
7. Как политика аудита влияет на выбор событий для регистрации в Windows и UNIX?
8. Опишите процесс настройки аудита доступа к файлу /etc/passwd в Linux с использованием auditctl.
9. Какие категории событий рекомендуется аудировать в Windows для обнаружения брутфорс-атак?
10. Как интеграция с SIEM-системами повышает эффективность аудита?
11. Какие риски возникают при избыточном сборе данных аудита?
12. Сравните возможности auditd (UNIX) и Event Viewer (Windows) для расследования инцидентов.

3. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ПОДГОТОВКЕ К ЛАБОРАТОРНЫМ ЗАНЯТИЯМ

Лабораторные занятия направлены на углубление знаний и закрепление основных понятий и методов, изучаемых в рамках дисциплины. Основная цель занятий – научиться применять теоретические знания для решения прикладных задач.

Рекомендации к подготовке

1. Изучение лекционного материала и конспектов:

– перед лабораторными занятиями необходимо проработать соответствующие разделы лекционного материала;

– рекомендуется вести конспект занятий и дополнительно пересмотреть его перед началом лабораторного занятия.

2. Проработка учебной литературы:

– используйте основные учебники и методические пособия, рекомендованные преподавателем;

– для глубокого понимания теории рекомендуется обращаться к дополнительной литературе.

3. Решение типовых задач:

– проработать примеры и типовые задачи из учебных материалов, методических указаний;

– выполните задачи, предложенные преподавателем для самостоятельной работы, что обеспечит лучшее понимание методов и подходов к решению задач.

4. Подготовка вопросов:

– составьте список вопросов по материалу, вызвавшему затруднения;

– обсуждение этих вопросов на лабораторном занятии поможет устранить пробелы в знаниях.

5. Вопросы для самоконтроля:

– перед каждым занятием рекомендуется проверять себя с помощью вопросов для самоконтроля из методических указаний к лекционным занятиям. Это позволит оценить уровень своей подготовки.

Тематический план лабораторных занятий приводится в разделе «Тематический план».

4. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ПОДГОТОВКЕ К САМОСТОЯТЕЛЬНОЙ РАБОТЕ

В лекциях по предмету излагаются основные знания по курсу дисциплины. Самостоятельная работа имеет особое значение для прочного усвоения материала. Она помогает научиться правильно, ориентироваться в научной литературе, самостоятельно мыслить и находить правильные ответы на возникающие вопросы. В ходе всех видов занятий происходит углубление и закрепление знаний студентов, вырабатывается умение правильно излагать свои мысли.

Самостоятельная работа выполняет ряд функций, к которым относятся:

- развивающая (повышение культуры умственного труда, приобщение к творческим видам деятельности, обогащение интеллектуальных способностей студентов);
- информационно-обучающая (учебная деятельность студентов на аудиторных занятиях, неподкрепленная самостоятельной работой, становится малорезультативной);
- ориентирующая и стимулирующая (процессу обучения придается профессиональное ускорение);
- воспитывающая (формируются и развиваются профессиональные качества специалиста):
- исследовательская (новый уровень профессионально-творческого мышления).

В основе самостоятельной работы студентов лежат принципы: самостоятельности, развивающе-творческой направленности, целевого планирования, личностно-деятельностного подхода.

Самостоятельная работа студентов проводиться с целью:

- систематизации и закрепления полученных теоретических знаний и практических умений студентов;
- углубления и расширения теоретических знаний;
- формирования умений использовать нормативную, правовую, справочную документацию и специальную литературу;
- развития познавательных способностей и активности студентов: творческой инициативы, самостоятельности, ответственности и организованности;
- формирования самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации;
- развития исследовательских умений.

Для достижения указанной цели студенты на основе плана самостоятельной работы должны решать следующие задачи:

– изучить рекомендуемые литературные источники:

– изучить основные понятия, представленные в глоссарии;

– ответить на контрольные вопросы:

– решить предложенные задачи, кейсы, ситуации;

– выполнить контрольные и курсовые работы.

Работа студентов в основном складывается из следующих элементов:

1. Изучение и усвоение в соответствии с учебным планом программного материала по всем учебным дисциплинам:

2. Выполнение письменных контрольных и курсовых работ;

3. Подготовка и сдача зачетов, курсовых работ, итоговых экзаменов:

4. Написание и защита дипломной работы.

Самостоятельная работа включает такие формы работы, как:

- индивидуальное занятие (домашние занятия) – важный элемент в работе студента по расширению и закреплению знаний;
- конспектирование лекций;
- получение консультаций для разъяснений по вопросам изучаемой дисциплины;
- подготовка ответов на вопросы тестов;
- подготовка к экзамену;
- выполнение контрольных, курсовых проектов и дипломных работ;
- подготовка научных докладов, рефератов, эссе;
- анализ деловых ситуаций (мини кейсов) и др.

Содержание внеаудиторной самостоятельной работы определяется в соответствии с рекомендуемыми видами заданий в соответствии с рабочей программой учебной дисциплины. Распределение объема времени на внеаудиторную самостоятельную работу в режиме дня студента не регламентируется расписанием.

Виды заданий для внеаудиторной самостоятельной работы, их содержание и характер могут иметь вариативный и дифференциальный характер, учитывать специфику специальности, изучаемой дисциплины, индивидуальные особенности студента.

Видами заданий для внеаудиторной самостоятельной работы могут быть:

Для овладения знаниями:

- чтение текста (учебника, первоисточника, дополнительной литературы);
- составление плана текста;
- конспектирование текста;
- выписки из текста;
- работа со словарями и справочниками;
- исследовательская работа;
- использование аудио- и видеозаписи;
- работа с электронными информационными ресурсами и ресурсами Internet:

Для закрепления и систематизации знании:

- работа с конспектом лекции (обработка текста);

• повторная работа над учебным материалом (учебника, первоисточника, дополнительной литературы, аудиовидеозаписей):
• составление плана и тезисов ответа;
• выполнение тестовых заданий;
• ответы на контрольные вопросы;
• аннотирование, реферирование, рецензирование текста;
• подготовка сообщений к выступлению на семинаре, конференции; подготовка рефератов, докладов;
• работа с компьютерными программами;
• подготовка к сдаче экзамена;

Для формирования умений:
• решение задач и упражнений по образцу;
• решение вариативных задач и упражнений:
• выполнение расчетно-графических работ;
• решение ситуационных производственных (профессиональных) задач;
• участие в научных и практических конференциях;
• проектирование и моделирование разных видов и компонентов профессиональной деятельности;
• создание проспектов, проектов, моделей;
• экспериментальная работа, участие в НИР;
• рефлексивный анализ профессиональных умений с использованием аудиовидеотехники и компьютерных расчетных программ, и электронных практикумов;
• подготовка курсовых проектов, работ и дипломных работ;

Правильная организация самостоятельных учебных занятий, их систематичность, целесообразное планирование рабочего времени позволяет привить студентам умения и навыки в овладении, изучении, усвоении и систематизации приобретаемых знаний в процессе обучения, обеспечивать высокий уровень успеваемости в период обучения, привить навыки повышения профессионального уровня в течение всей трудовой деятельности.

5. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ

Подробные указания приведены в учебно-методическом пособии по выполнению курсовых работ для студентов специальности 10.05.03 «Информационная безопасность автоматизированных систем» по дисциплине «Безопасность операционных систем»

Цели каждой отдельной курсовой работы должны раскрывать выбранную студентом тему. Курсовая работа предназначена для углубления студентами теоретических и практических навыков в области обеспечения информационной безопасности операционных систем. Современные требования к специалистам предполагают не только глубокое знание теоретических основ и принципов использования информационных технологий. Будущие специалисты должны иметь четкое представление обо всех этапах создания и эксплуатации информационных технологий, уметь осуществлять выбор из широкого арсенала современных средств и методов защиты информации в операционных системах наиболее адекватные поставленной задаче. Курсовая работа – это одна из форм учебной (творческой и научно-исследовательской) работы, ее выполнение является обязательным для всех студентов очной и заочной форм обучения. Выполнение курсовой работы представляет собой самостоятельное решение студентом под руководством преподавателя частной задачи или проведение исследования по одному из вопросов, изучаемых в цикле специальных дисциплин (по ГОС ВПО) или в дисциплинах профессионального цикла (по ФГОС ВПО). Основной целью выполнения курсовых работ (проектов) является закрепление, углубление и обобщение знаний, полученных студентом за время теоретического и практического обучения, расширение объема профессионально значимых умений и навыков. Содержание курсовых работ (проектов) должно отвечать учебным задачам дисциплины, увязываться с последующей работой выпускников по специальности /направлению подготовки.

Поэтому в цели и задачи курсовой работы входят:

1) закрепление практических навыков настройки политик безопасности операционных систем, полученных на лабораторных занятиях по дисциплине «Безопасность операционных систем»;

2) углубление теоретических и практических знаний в области методологии отладки политик безопасности операционных систем;

3) развитие навыков самостоятельного планирования задач защиты операций и ключевой информации операционных систем;

4) получение опыта сбора регистрируемых событий, и обработки регистрируемых событий в операционной системе;

5) приобретение навыков создания резервных копий операционных систем.

Выполнение курсовой работы позволяет расширить и закрепить приобретенные студентом в ходе обучения в вузе теоретические знания и продемонстрировать полученные навыки по самостоятельной постановке и решению конкретной задачи, а также продемонстрировать владение профессиональными навыкам в области защиты информации.

При выполнении курсовой работы обучающимся рекомендуется использование элементов дистанционных образовательных технологий с использованием информационных и учебно-методических ресурсов. При этом график курсовой работы должен определяться количеством часов, указанным в учебном плане.

Важнейшими требованиями при выполнении курсовой работы для студента являются ее самостоятельность и актуальность, связанная с решением вопросов по заданиям или по тематике работ промышленных, коммерческих или научно-исследовательских организаций; использованием современной программной и аппаратной базы; справочных материалов; новейших методов организации расчетов, проектирования и исследований.

Обучающийся выбирает тему курсовой работы из числа предложенных тем. При выборе темы курсовой работы (КР) необходимо учесть возможность дальнейшего ее развития, углубления и конкретизации, а также использования в курсовой работе.

Обучающийся может предложить свою тему с обоснованием целесообразности ее разработки и при согласовании с заведующим кафедрой и/или научным руководителем.

Выбранная тема курсовой работы должна быть согласована с научным руководителем. Изменения темы курсовой работы могут быть внесены только после согласования с научным руководителем.

При выборе темы курсовой работы необходимо учитывать следующие условия:

- соответствие темы курсовой работы содержанию дисциплины, по которой выполняется курсовая работа; актуальность проблемы;
- наличие специальной литературы и возможность получения фактических данных, необходимых для анализа;
- собственные научные интересы и способности обучающегося; преемственность исследований, начатых в предыдущих курсовых работах (проектах) и в период учебных практик;
- исключение по возможности дублирования (дословного совпадения формулировок) тем курсовых работ (проектов), выполняемых обучающимися (группой обучающихся).

Также при самостоятельном определении темы студенту требуется учесть свой опыт в выбранной сфере, наличие соответствующих знаний и навыков, а также имеющихся наработок по предполагаемой тематике. Это, прежде всего, относится к тем, кто долго собирал и обрабатывал материал по той или иной проблематике, участвовал в НИРС, научных конференциях, имеет публикации

в научных журналах, сборниках и т. д. Научный руководитель может быть преподаватель выпускающей кафедры

Студенту следует периодически информировать научного руководителя о ходе выполнения курсовой работы, консультироваться по вызывающим затруднения или сомнения теоретическим и практическим вопросам, обязательно ставить в известность о возможных проблемах в выполнении работы и её содержания. Изменение выбранной ранее темы курсовой работы возможно при согласовании с научным руководителем.

Курсовая работа выполняется студентом в период семестра, когда по учебному плану изучается соответствующая дисциплина.

Курсовая работа представляет собой решение практической, научно-исследовательской задачи одной из актуальных проблем в области защиты операционных систем,

Объектами курсовой работы могут быть методы поиска уязвимостей операционных систем, методы анализа уязвимости операционных, способы повышения защищенности операционных систем, специфика комплектования системного обеспечения в целях повышения информационной безопасности.

При выполнении курсовой работы должно быть предусмотрено:

– обоснование актуальности и важности решаемой задачи обеспечения информационной безопасности выбранного объекта;

– анализ проблемной области защиты операционных систем;

– определение, анализ возможных путей и способов исследования и описание выбранных методов и средств решения поставленных задач;

– методы и способы решения проблем безопасности операционных систем.

При определении темы и соответственно порядка разработки курсовой работы можно придерживаться следующего плана:

– точная формулировка темы, целей и задач выполнения курсовой работы;

– изучение специфики проблемной области;

– выявление уже существующих решений и определение их эффективности;

– обоснование предложений по решению проблем в области информационной защиты операционных систем;

– реализация предложенных средств и методов защиты, исследования меры защищенности операционных систем и их компонентов;

– проверка работоспособности предложенных мер защиты.

Курсовая работа предусматривает **следующие этапы:**

1. **Подготовка** к выполнению курсовой работы заключается в изучении литературы по выбранной проблеме, сборе исходных данных по рассматриваемым проблемам. На этом этапе изучаются цели функционирования и развития объекта, его обеспеченность средствами защиты, каналы уязвимости, Студент собирает, обобщает и систематизирует материалы, необходимые для разработ-

ки предложений Полученные материалы используются во введении и аналитической части работы.

2. Разработка темы. На основе собранных и обобщенных материалов, формулируются способы решения задач и разрабатываются алгоритмы решения задач, определяется специфика и порядок их реализации, реализуются предложенные решения, обосновывается эффективность разработки, исследований, решений.

3. Этап включает оформление курсовой работы. При этом выполняется:

– систематизация и обработка материалов курсовой работы;

– отбор материала для оформления содержательной части работы и составление структуры ее изложения, подготовка необходимого иллюстративного материала и т. д.;

– определение направлений и основного содержания предложений, выявление необходимости дополнительного сбора материалов; формирование чернового варианта разработки в целом;

– сбор дополнительных материалов, детальная разработка и обоснование выдвинутых предложений;

– уточнение аналитической и исследовательской части работы;

– редактирование и окончательное оформление отобранного материала;

– оформление иллюстративного материала.

4. Заключительным этапом подготовки курсовой работы к защите является предъявление ее преподавателю ИБ. К этому моменту курсовая работа должна быть подписана студентом.

Список типовых (примерных) тем

1. Идентификация и аутентификация пользователя. Изменение полномочий пользователя MacOC.
2. Регистрация событий системы.
3. Установка и обновление программного обеспечения в ОС Linux, FreeBSD. Представление о пакете rpm.
4. Сборка ядра ОС Linux.
5. Сборка ядра ОС FreeBSD.
6. Командные интерпретаторы ОС.
7. Создание разделов и файловых систем ОС Linux. Монтирование файловых систем.
8. Журналируемая файловая система.
9. Дисковые квоты в ОС Linux, FreeBSD, Windows.
10. Реализация функций информационной безопасности в файловой системе FAT.
11. Реализация функций информационной безопасности в файловой системе NTFS.
12. Реализация функций информационной безопасности в файловой системе FAT 32.

13. Реализация функций информационной безопасности в файловой системе ext3.

14. Шифрованная файловая система EFS.

15. Ядро и вспомогательные модули ОС Linux

16. API функции с точки зрения безопасности ОС Windows.

17. Файловая система EXFAT.

18. Управление учетными записями пользователей и групп в ОС Linux, Windows, FreeBSD.

19. Планирование процессов ОС Linux.

20. Сравнительный анализ функций безопасности Windows 7 и Windows 2003.

21. Настройка модуля безопасности SE в ОС Linux.

22. Особенности аудита в ОС Linux.

23. Особенности аудита в ОС Windows.

24. Особенности аутентификации пользователя с помощью протокола Kerberos.

25. Безопасность ядра ОС UNIX.

26. Темы, предложенные к разработке обучающимися, в случае обоснованности целесообразности её разработки для практического применения в области информационной безопасности операционных систем.

6. ТРЕБОВАНИЯ К АТТЕСТАЦИИ ПО ДИСЦИПЛИНЕ

Текущая аттестация

В ходе изучения дисциплины студентам предстоит пройти следующие этапы текущей аттестации:

Преподаватель вправе выбрать методику оценивания знаний студентов: традиционная зачетно-экзаменационная, либо балльно-рейтинговая.

Зачет может проводиться как в традиционной форме, так и в виде экзаменационного тестирования. Тестовые задания для проведения экзаменационного тестирования приведены в фонде оценочных средств по дисциплине

Выбрана традиционная зачетно-экзаменационная методика оценивания знаний

Предусматривается: зачет, экзамен, курсовая работа

Преподаватель вправе выбрать методику оценивания знаний студентов: традиционная зачетно-экзаменационная либо балльно-рейтинговая**.** Выбрана традиционная зачетно-экзаменационная методика оценивания знаний. Предусматриваются: зачет, экзамен, курсовая работа.

В ходе изучения дисциплины студентам предстоит пройти следующие этапы текущей аттестации: контроль самостоятельной работы по темам дисциплины, контроль выполнения лабораторных работ.

К оценочным средствам текущего контроля успеваемости относятся:

– тестовые задания открытого и закрытого типов.

Промежуточная аттестация в форме зачета проходит по результатам прохождения всех видов текущего контроля успеваемости. В отдельных случаях (при не прохождении всех видов текущего контроля) зачет может быть проведен в виде тестирования.

Экзамен может проводиться как в традиционной форме, так и в виде экзаменационного тестирования. Тестовые задания для проведения экзаменационного тестирования приведены в фонде оценочных средств по дисциплине.

Критерии оценки результатов освоения дисциплины

Универсальная система оценивания результатов обучения включает в себя системы оценок: 1) «отлично», «хорошо», «удовлетворительно», «неудовлетворительно»; 2) «зачтено», «не зачтено»; 3) 100–балльную/процентную систему и правило перевода оценок в пятибалльную систему (таблица 2).

Таблица 2 – Система оценок и критерии выставления оценки

Система оценок / **Критерий**	**2**	**3**	**4**	**5**
	0–40 %	**41–60 %**	**61–80 %**	**81–100 %**
	«неудовлетворительно»	**«удовлетворительно»**	**«хорошо»**	**«отлично»**
	«не зачтено»	**«зачтено»**		
1 Системность и полнота знаний в отношении изучаемых объектов	Обладает частичными и разрозненными знаниями, которые не может научно-корректно связывать между собой (только некоторые из которых может связывать между собой)	Обладает минимальным набором знаний, необходимым для системного взгляда на изучаемый объект	Обладает набором знаний, достаточным для системного взгляда на изучаемый объект	Обладает полнотой знаний и системным взглядом на изучаемый объект
2 Работа с информацией	Не в состоянии находить необходимую информацию, либо в состоянии находить отдельные фрагменты информации в рамках поставленной задачи	Может найти необходимую информацию в рамках поставленной задачи	Может найти, интерпретировать и систематизировать необходимую информацию в рамках поставленной задачи	Может найти, систематизировать необходимую информацию, а также выявить новые, дополнительные источники информации в рамках поставленной задачи

Система оценок / Критерий	2	3	4	5
	0–40 %	**41–60 %**	**61–80 %**	**81–100 %**
	«неудовлетворительно»	**«удовлетворительно»**	**«хорошо»**	**«отлично»**
	«не зачтено»	**«зачтено»**		
3 Научное осмысление изучаемого явления, процесса, объекта	Не может делать научно-корректных выводов из имеющихся у него сведений, в состоянии проанализировать только некоторые из имеющихся у него сведений	В состоянии осуществлять научно-корректный анализ предоставленной информации	В состоянии осуществлять систематический и научно-корректный анализ предоставленной информации, вовлекает в исследование новые релевантные задаче данные	В состоянии осуществлять систематический и научно-корректный анализ предоставленной информации, вовлекает в исследование новые релевантные поставленной задаче данные, предлагает новые ракурсы поставленной задачи
4 Освоение стандартных алгоритмов решения профессиональных задач	В состоянии решать только фрагменты поставленной задачи в соответствии с заданным алгоритмом, не освоил предложенный алгоритм, допускает ошибки	В состоянии решать поставленные задачи в соответствии с заданным алгоритмом	В состоянии решать поставленные задачи в соответствии с заданным алгоритмом, понимает основы предложенного алгоритма	Не только владеет алгоритмом и понимает его основы, но и предлагает новые решения в рамках поставленной задачи

Оценивание тестовых заданий закрытого типа осуществляется по системе зачтено/не зачтено («зачтено» – 41–100 % правильных ответов; «не зачтено» – менее 40 % правильных ответов) или пятибалльной системе (оценка «неудовлетворительно» – менее 40 % правильных ответов; оценка «удовлетворительно» – от 41 до 60 % правильных ответов; оценка «хорошо» – от 61 до 80 % правильных ответов; оценка «отлично» – от 81 до 100 % правильных ответов).

Тестовые задания открытого типа оцениваются по системе «зачтено/ не зачтено». Оценивается верность ответа по существу вопроса, при этом не учитывается порядок слов в словосочетании, верность окончаний, падежи.

Завершающим этапом изучения дисциплины является итоговая аттестация, представляющая собой экзамен.

Допуск к итоговой аттестации возможен при:

– наличии всех выполненных, сданных (проверенных, защищенных) лабораторных работах, наличии отчётов по ним;

– наличии показателей приемлемого уровня освоения материалов курса: более 50 % посещений от общего числа требуемых по учебному плану.

Примерные вопросы к зачету/экзамену по дисциплине

Вопросы к зачету

1. Пользовательский интерфейс ОС. Классификация программных средств
2. Основные функции ОС. Классификация ОС.
3. Концепция процесса. Типология процессов
4. Концепция ресурсов. Концепция виртуальности.
5. Концепция прерывания. Классы прерываний.
6. Классификация операционных систем. Состав ядра ОС.
7. Модули ядра. Перечислить вспомогательные модули ОС и режимы.
8. Многослойная структура ОС. Микроядерная архитектура.
9. Управление процессами ОС. Понятия задание, задача, поток, нить и процесс.
10. Контекст процесса. Особенности работы нити процесса.
11. Планирование процессов. Концепции планирования процессов. Понятие кванта.
12. Способы организации процесса. Особенности организации процесса. Проблемы выполнения процессов на процессоре.
13. Понятие прерывания. Типы прерывания. Последовательность при обработке прерываний. Способы выполнения прерываний.
14. Особенности управления памятью в ОС.
15. Особенности работы виртуальной памяти и swapping. Алгоритмы распределения памяти. Алгоритмы управления памятью.
16. Механизмы распределения адресов в ОС. Распределение при реальных и виртуальных адресациях.
17. Файловые системы. Общая организация ФС.
18. Особенности ФС FAT и exFAT.
19. Особенности ФС NTFS.
20. Особенности файловых систем ext.
21. Сравнительный анализ файловых систем.

Вопросы к экзамену

1. Пользовательский интерфейс ОС. Классификация программных средств
2. Основные функции ОС. Классификация ОС.
3. Концепция процесса. Типология процессов
4. Концепция ресурсов. Концепция виртуальности.
5. Концепция прерывания. Классы прерываний.
6. Классификация операционных систем. Состав ядра ОС.
7. Модули ядра. Перечислить вспомогательные модули ОС и режимы.
8. Многослойная структура ОС. Микроядерная архитектура.

9. Управление процессами ОС. Понятия задание, задача, поток, нить и процесс.

10. Контекст процесса. Особенности работы нити процесса.

11. Планирование процессов. Концепции планирования процессов. Понятие кванта.

12. Способы организации процесса. Особенности организации процесса. Проблемы выполнения процессов на процессоре.

13. Понятие прерывания. Типы прерывания. Последовательность при обработке прерываний. Способы выполнения прерываний.

14. Особенности управления памятью в ОС.

15. Особенности работы виртуальной памяти и swapping. Алгоритмы распределения памяти. Алгоритмы управления памятью.

16. Механизмы распределения адресов в ОС. Распределение при реальных и виртуальных адресациях.

17. Файловые системы. Общая организация ФС.

18. Особенности ФС FAT и exFAT.

19. Особенности ФС NTFS.

20. Особенности файловых систем ext.

21. Сравнительный анализ файловых систем.

22. Особенности реализации функции безопасности в ОС. Краткие различия организации безопасности Unix и Windows.

23. Организация безопасности в Unix-системах.

24. Аутентификация в Unix-системах.

25. Аспекты механизмов безопасности Windows (ACE, ACL, SACL, DACL).

26. Привилегии субъектов в Windows. Маркеры доступа. Дескриптор защиты.

27. Авторизация в ОС Windows

28. Дополнительные модули безопасности ОС.

29. Аудит в ОС

30. Способы создания и управления процессами в ОС Linux.

31. Основные команды MS-DOS, ОС Linux. Особенности создания Bat-файлов, shell-сценариев.

ЗАКЛЮЧЕНИЕ

Правильная организация учебных занятий, их систематичность, целесообразное планирование рабочего времени позволяет привить студентам умения и навыки в овладении, изучении, усвоении и систематизации приобретаемых знаний в процессе обучения, обеспечивать высокий уровень успеваемости в период обучения, привить навыки повышения профессионального уровня в течение всей трудовой деятельности.

ЛИТЕРАТУРА

Основные источники

1. Окороков, В. А. Безопасность операционных систем / В. А. Окороков. – Санкт-Петербург: Лань, 2024. – 228 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/367472 (дата обращения: 06.12.2024). – ISBN 978-5-507-48297-9. – Текст : электронный.

2. Киренберг, Г. А. Информационная безопасность современных операционных систем: учеб. пособие / Киренберг, Г. А. – Кемерово: КузГТУ им. Т. Ф. Горбачева, 2022. – 138 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/295736 (дата обращения: 06.12.2024). – ISBN 978-5-00137-320-9. – Текст : электронный.

3. Потерпеев, Г. Ю. Безопасность операционных систем: учеб. пособие / Г. Ю. Потерпеев, В. С. Нефедов, А. А. Криулин. – Москва: РТУ МИРЭА, 2021. – 93 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/182416 (дата обращения: 06.12.2024). – ISBN 978-5-7339-1393-3. – Текст : электронный.

4. Зверева, О. М. Операционные системы: учеб. пособие / О. М. Зверева; науч. ред. Л. Г. Доросинский; Уральский федеральный университет им. первого Президента России Б. Н. Ельцина. – Екатеринбург: Издательство Уральского университета, 2020. – 223 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=699030 (дата обращения: 03.11.2024). – ISBN 978-5-7996-3146-8. – Текст : электронный.

5. Кобылянский, В. Г. Операционные системы, среды и оболочки: учеб. пособие для вузов / В. Г. Кобылянский. – 3-е изд., стер. – Санкт-Петербург: Лань, 2022. – 120 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/254651 (дата обращения: 09.10.2024). – ISBN 978-5-507-44969-9. – Текст : электронный.

Дополнительная литература

6. Национальная безопасность: учебник / В. И. Абрамов, М. А. Газимагомедов, К. К. Гасанов [и др.]; под ред. К. К. Гасанова, Н. Д. Эриашвили, О. А. Мироновой. – 3-е изд., перераб. и доп. – Москва: Юнити-Дана, 2023. – 288 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=700171 (дата обращения: 05.06.2024). – ISBN 978-5-238-03639-7. – Текст : электронный.

7. Власенко, А. Ю. Операционные системы: учеб. пособие / А. Ю. Власенко, С. Н. Карабцев, Т. С. Рейн. – Кемерово: Кемеровский государственный университет, 2019. – 161 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=574269 (дата обращения: 03.11.2024). – ISBN 978-5-8353-2424-8. – Текст : электронный.

8. Потерпеев, Г. Ю. Сборник практических занятий для дисциплины безопасность операционных систем. Практикум: учеб. пособие / Г. Ю. Потерпеев, О. В. Трубиенко, Д. П. Абрамов. – Москва: РТУ МИРЭА, 2023. – Ч. 1. – 2023. – 65 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/368750 (дата обращения: 06.12.2024). – ISBN 978-5-7339-1803-7. – Текст : электронный.

9. Программно-аппаратные средства обеспечения информационной безопасности: лаб. практикум для студентов специальности 10.05.03 «Информ. безопасность автоматизир. систем» / Федер. агентство по рыболовству, Калининигр. гос. техн. ун-т, Балт. гос. акад. рыбопромыслового флота; сост.: А. Г. Жестовский, В. В. Подтопельный. – 2-е изд., перераб. и доп. – Калининград: БГАРФ, 2019. – Режим доступа: для авториз. пользователей. – URL: https://lib.klgtu.ru/web/index.php (дата обращения: 05.11.2024). – ISBN 978-5-238-03639-7. – Текст : электронный. – Ч. 1. Защита компьютерной информации и компьютерных систем от вредоносных программ.

10. Жестовский, А. Г. Программно-аппаратные средства обеспечения информационной безопасности: учеб. пособие для студентов высш. учеб. заведений, обучающихся по направлению «Информ. безопасность», по прогр. подгот. бакалавров, магистров, специалистов / А. Г. Жестовский, В. В. Подтопельный; Федер. агентство по рыболовству [и др.]. – Калининград: БГАРФ, 2018. – Текст : непосредственный. – Ч. 2. Настройка систем защиты информации от несанкционированного доступа. – 2018. – 100 с.

11. Подтопельный, В. В. Программно-аппаратные средства обеспечения информационной безопасности: учеб. пособие для студентов специальности 10.05.03 «Информ. безопасность автоматизир. систем» всех форм обучения / В. В. Подтопельный. – Калининград: БГАРФ, 2020. – Текст : непосредственный. – Ч. 3. Поиск и извлечение вредоносных программ в программной среде. – 2020. – 99 с.

12. Подтопельный, В. В. Программно-аппаратные средства обеспечения информационной безопасности: учеб. пособие для студентов специальности 10.05.03 «Информ. безопасность автоматизир. систем» всех форм обучения / В. В. Подтопельный; Федер. агентство по рыболовству [и др.].–- Калининград:

БГАРФ, 2020. – Текст : непосредственный. – Ч. 4. Настройка подсистем СЗИ. – 2021. – 97 с.

13. Аверченков, В. И. Аудит информационной безопасности: учеб. пособие / В. И. Аверченков. – 4-е изд., стер. – Москва: ФЛИНТА, 2021. – 269 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=93245 (дата обращения: 05.11.2024). – ISBN 978-5-9765-1256-6. – Текст : электронный.

Учебно-методические пособия, нормативная литература

14. Подтопельный, В. В. Безопасность операционных систем: учебно-методическое пособие по изучению дисциплины для студентов специальности 10.05.03 «Информационная безопасность автоматизированных систем», специализация «Безопасность открытых информационных систем». – Калининград: Изд-во ФГБОУ ВО «КГТУ», 2022. – 51 с. – URL: https://www.klgtu.ru/vikon/sveden/files/vih/UMP_Bezopasnosty_operacionnyx_sistem(1).pdf (дата обращения: 08.12.2024). – Текст : электронный.

15. Подтопельный, В. В. Безопасность операционных систем: учебно-методическое пособие по выполнению лабораторных работ по дисциплине для студентов специальности 10.05.03 «Информационная безопасность автоматизированных систем» специализация «Безопасность открытых информационных систем». – Калининград: Изд-во ФГБОУ ВО «КГТУ», 2022. – 181 с. – URL: https://www.klgtu.ru/vikon/sveden/files/aij/UMP_Bezopasnosty_operacionnyx_sistem_(laboratornye_raboty)(1).pdf (дата обращения: 08.12.2024). – Текст : электронный.

16. Безопасность операционных систем: метод. указания по выполнению курсовых работ для студентов специальности 10.05.03 «Информ. безопасность автоматизир. систем» очной формы обучения / Федер. агентство по рыболовству [и др.]; авт.-сост. В. В. Подтопельный. – Калининград: БГАРФ, 2023. – 53 с. – Текст : непосредственный.

17. «Доктрина информационной безопасности Российской Федерации» (утв. Указом Президентом РФ 05.12.2016 № 646 (в действующей редакции). – Режим доступа: для авториз. пользователей из справ.-правовой системы КонсультантПлюс. – Текст: электронный.

18. «Конституция Российской Федерации» (принята всенародным голосованием 12.12.1993 с изменениями, одобренными в ходе общероссийского голосования 01.07.2020) (в действующей редакции). – Режим доступа: для авториз. пользователей из справ.-правовой системы КонсультантПлюс. – Текст: электронный.

19. Федеральный закон от 28.12.2010 N 390-ФЗ «О безопасности» (в действующей редакции). – Режим доступа: для авториз. пользователей из справ.-правовой системы КонсультантПлюс. – Текст: электронный.

20. Федеральный закон от 27.07.2006 N 149-ФЗ «Об информации, информационных технологиях и о защите информации» (в действующей редакции). – Режим доступа: для авториз. пользователей из справ.-правовой системы КонсультантПлюс. – Текст: электронный.

21. Федеральный закон от 27.07.2006 N 152-ФЗ «О персональных данных» (в действующей редакции). – Режим доступа: для авториз. пользователей из справ.-правовой системы КонсультантПлюс. – Текст: электронный.

22. Закон РФ от 21.07.1993 N 5485-1 «О государственной тайне» (в действующей редакции). – Режим доступа: для авториз. пользователей из справ.-правовой системы КонсультантПлюс. – Текст: электронный.

23. Указ Президента РФ от 06.03.1997 N 188 «Об утверждении Перечня сведений конфиденциального характера» (в действующей редакции). – Режим доступа: для авториз. пользователей из справ.-правовой системы КонсультантПлюс. – Текст: электронный.

24. "ГОСТ Р 50739-95. Средства вычислительной техники. Защита от несанкционированного доступа к информации. Общие технические требования" (принят и введен в действие Постановлением Госстандарта РФ от 09.02.1995 N 49) (в действующей редакции). – Режим доступа: для авториз. пользователей из справ.-правовой системы КонсультантПлюс. – Текст: электронный.

25. «ГОСТ Р 50922-2006. Национальный стандарт Российской Федерации. Защита информации. Основные термины и определения» (утв. и введен в действие Приказом Ростехрегулирования от 27.12.2006 N 373-ст) (в действующей редакции). – Режим доступа: для авториз. пользователей из справ.-правовой системы КонсультантПлюс. – Текст: электронный.

26. «Руководящий документ. Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации» (утв. Решением Гостехкомиссии России от 30.03.1992) (в действующей редакции). – Режим доступа: для авториз. пользователей из справ.-правовой системы КонсультантПлюс. – Текст: электронный.

27. «Руководящий документ. Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации» (утв. Решением Гостехкомиссии России 30.03.1992) (в действующей редакции). – Режим доступа: для авториз. пользователей из справ.-правовой системы КонсультантПлюс. – Текст: электронный.

Локальный электронный методический материал

Владислав Владимирович Подтопельный

БЕЗОПАСНОСТЬ ОПЕРАЦИОННЫХ СИСТЕМ

Редактор С. Кондрашова
Корректор Т. Звада

Уч.-изд. л. 6,8. Печ. л. 5,6.

Издательство федерального государственного бюджетного
образовательного учреждения высшего образования
«Калининградский государственный технический университет».
236022, Калининград, Советский проспект, 1