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

В. В. Подтопельный

ПРОГРАММИРОВАНИЕ КОМПОНЕНТОВ ОТКРЫТЫХ СИСТЕМ В ЗАЩИЩЁННОМ ИСПОЛНЕНИИ

Учебно-методическое пособие по изучению дисциплины
для студентов 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 института цифровых технологий ФГБОУ ВО
«Калининградский государственный технический университет»
Н. Я. Великите

Подтопельный, В. В.

Программирование компонентов открытых систем в защищённом исполнении: 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», 2025. – 66 с.

Учебно-методическое пособие является руководством по изучению дисциплины «Программирование компонентов открытых систем в защищённом исполнении». 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 5-го курса специальности 10.05.03 «Информационная безопасность автоматизированных систем».

Табл. 3, список лит. – 20 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……4
1. Тематический план……6
2. Содержание дисциплины 10
3.Методические рекомендации по подготовке к лабораторным занятиям..... 54
4. Методические указания по самостоятельной работе 55
5. Требования к аттестации по дисциплине 58
Заключение 62
Литература 62

ВВЕДЕНИЕ

Данное учебно-методическое пособие предназначено для студентов специальности 10.05.03 «Информационная безопасность автоматизированных систем», специализация «Безопасность открытых информационных систем», изучающих дисциплину «**Программирование компонентов открытых систем в защищённом исполнении».**

ОПК-7: Способен создавать программы на языках общего назначения, применять методы и инструментальные средства программирования для решения профессиональных задач, осуществлять обоснованный выбор инструментария программирования и способов организации программ

Целью освоения дисциплины «Программирование компонентов открытых систем в защищённом исполнении» является: освоение студентами знаний в области методологии программирования компонентов открытых систем, необходимых для успешного применения опыта на практике, с использованием методов и средств защиты.

В результате освоения дисциплины обучающийся должен:

знать:

– современные технологии программирования;

– эталонная модель взаимодействия открытых систем, основные протоколы, последовательность и содержание этапов построения и функционирования современных локальных и глобальных компьютерных сетей;

– методы тестирования и отладки программного и аппаратного обеспечения;

уметь:

– оценивать сложность алгоритмов и вычислений;

– создавать программы на языках общего назначения, применять методы и инструментальные средства программирования для решения профессиональных задач;

– осуществлять обоснованный выбор инструментария программирования и способов организации программ в защищенном исполнении;

– основные методы и способы защиты компонентов открытых систем.

Навыки:

– создавать программы на языках общего назначения;

– применять методы и инструментальные средства программирования для решения профессиональных задач;

– программирование компонентов открытых систем с использованием средств защиты.

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

Далее в пособии представлен тематический план, содержащий перечень изучаемых тем, обязательных лабораторных/практических 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.

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.

Типовое ПО на всех ПК:

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

1. ТЕМАТИЧЕСКИЙ ПЛАН

Таблица 1 – Тематический план

	Раздел (модуль) дисциплины	**Тема**	**Объем аудиторной работы, ч**	**Объем самостоя-тельной работы, ч**
	Лекции (6 семестр – 32 ч ауд., 37,85 ч. – сам. р.)			
1.1	Раздел 1: Основы программирования защищенных компонентов открытых систем	Введение в архитектуру открытых систем и принци-пы безопасности	4	4
1.2	Раздел 1: Основы программирования защищенных компонентов открытых систем	Методы защиты данных на уровне кода	4	4
1.3	Раздел 1: Основы программирования защищенных компонентов открытых систем	Ролевые модели доступа и управление привилегиями	4	4
1.4	Раздел 1: Основы программирования защищенных компонентов открытых систем	Безопасная работа с внешними API и сервисами	4	4
1.5	Раздел 2: Распределенные базы данных как ядро открытых систем	Архитектура распределенных баз данных	4	4
1.6	Раздел 2: Распределенные базы данных как ядро открытых систем	Обеспечение безопасности данных в распределен-ных БД	4	4
1.7	Раздел 2: Распределенные базы данных как ядро открытых систем	Репликация и синхронизация данных	4	4
1.8	Раздел 2: Распределенные базы данных как ядро открытых	Интеграция распределенных БД с защищенными компонентами	4	9,85

	Раздел (модуль) дисциплины	Тема	Объем аудиторной работы, ч	Объем самостоя-тельной работы, ч
	систем			
Лекции (7 семестр – 32 ч ауд., 38 ч – сам. р.)				
2.1	Раздел 3: Методы отладки защищенных распределённых компонентов открытых систем	Инструменты отладки распределенных систем	4	4
2.2	Раздел 3: Методы отладки защищенных распределённых компонентов открытых систем	Выявление уязвимостей в защищенных компонентах	4	4
2.3	Раздел 3: Методы отладки защищенных распределённых компонентов открытых систем	Отладка асинхронных и параллельных процессов	4	4
2.4	Раздел 3: Методы отладки защищенных распределённых компонентов открытых систем	Восстановление после сбоев и обеспечение отказо-устойчивости	4	4
2.5	Раздел 4: Особенности интеграции мер защиты в компоненты открытых систем	Шифрование данных в открытых экосистемах	4	10
2.6	Раздел 4: Особенности интеграции мер защиты в компоненты открытых систем	Безопасная интеграция сторонних сервисов	4	4
2.7	Раздел 4: Особенности интеграции мер защиты в компоненты открытых систем	Аудит и мониторинг защищенных систем	4	4
2.8	Раздел 4: Особенности интеграции мер защиты в компоненты открытых систем	Адаптация к новым угрозам и обновление защиты	4	4

	Раздел (модуль) дисциплины	Тема	Объем аудиторной работы, ч	Объем самостоя-тельной работы, ч
			64	**75,85**
		Лабораторные занятия (6 семестр)		
1.	Раздел 1: Основы программирования защищенных компонентов открытых систем	Архитектура открытых систем и принципы безопас-ности	8	-
2.	Раздел 1: Основы программирования защищенных компонентов открытых систем	Методы защиты данных на уровне кода	8	-
3.	Раздел 1: Основы программирования защищенных компонентов открытых систем	Безопасная работа с внешними API и сервисами	8	-
4.	Раздел 2: Распределенные базы данных как ядро открытых систем	Обеспечение безопасности данных в распределенных БД	8	-
Всего за семестр:			32	
Лабораторные занятия (7 семестр)				
1.	Раздел 3: Методы отладки защищенных распределённых компонентов открытых систем	Инструменты отладки распределенных систем	8	-
2.	Раздел 3: Методы отладки защищенных распределённых компонентов открытых систем	Восстановление после сбоев и обеспечение отказоустойчивости	8	-
3.	Раздел 4: Особенности интеграции мер защиты в	Шифрование данных в открытых экосистемах	8	-

	Раздел (модуль) дисциплины	Тема	Объем аудиторной работы, ч	Объем самостоя-тельной работы, ч
	компоненты открытых систем			
4.	Раздел 4: Особенности интеграции мер защиты в компоненты открытых систем	Аудит и мониторинг защищенных систем	8	-
		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 проекта. Защита	-	-
			34,75	**0**
		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**
		КА	12	
		РЭ	1,4	
		Всего	**176,15**	**75,85**
		ИТОГО 252		

2. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

Раздел 1. Основы программирования защищенных компонентов открытых систем

Тема 1.1 Введение в архитектуру открытых систем и принципы безопасности

Перечень изучаемых вопросов

1. Понятие открытых систем (Open Systems).
2. Угрозы безопасности в распределенных средах.
3. Принципы «безопасности по умолчанию» (Secure by Design).

Методические указания к изучению

Современные информационные системы функционируют в условиях глобальной цифровизации, где ключевое значение приобретают открытость, совместимость и безопасность. Архитектура открытых систем и принципы их защиты становятся фундаментом для разработки устойчивых к угрозам решений. Данная статья раскрывает основные аспекты, изучаемые в рамках дисциплины «Программирование компонентов открытых систем в защищённом исполнении», фокусируясь на трёх ключевых темах: понятии открытых систем, угрозах безопасности в распределённых средах и принципах «безопасности по умолчанию».

1. Понятие открытых систем (Open Systems)

Открытые системы – это программно-аппаратные комплексы, построенные на стандартизированных интерфейсах и протоколах, обеспечивающих взаимодействие между разнородными компонентами. Их главная цель – устранить зависимость от конкретных производителей и платформ, обеспечивая:

- Совместимость (interoperability) – способность систем обмениваться данными независимо от их архитектуры.
- Масштабируемость – возможность расширения функционала без полной переработки системы.
- Прозрачность – доступность документации и исходного кода (в случае open source).

Примеры открытых систем включают операционные системы на базе Linux, веб-стандарты (HTML, HTTP) и облачные платформы. Однако их главное преимущество – гибкость – одновременно создаёт риски, связанные с уязвимостью к кибератакам, что требует внедрения строгих мер безопасности.

2. Угрозы безопасности в распределённых средах

Распределённые системы, характерные для открытых архитектур, подвержены уникальным угрозам из-за децентрализованной структуры и зависимости от сетевых коммуникаций. Основные риски включают:

А. Атаки типа «человек посередине» (MITM) – перехват данных между узлами сети.

B. Распределённые атаки на отказ в обслуживании (DDoS) — перегрузка ресурсов путём координации множества источников.

C. Уязвимости в API – ошибки в интерфейсах взаимодействия компонентов, ведущие к утечкам данных.

D. Инъекционные атаки – внедрение вредоносного кода через невалидируемый ввод.

E. Несанкционированный доступ – компрометация узлов из-за слабой аутентификации.

Особую опасность представляют сетевые протоколы с недостаточным шифрованием (например, HTTP вместо HTTPS) и отсутствие сегментации сети, что позволяет злоумышленникам перемещаться между компонентами. Для минимизации угроз требуется применение принципов безопасности на этапе проектирования системы.

3. Принципы «безопасности по умолчанию» (Secure by Design)

Концепция Secure by Design предполагает интеграцию механизмов защиты в архитектуру системы с самого начала её разработки. Ключевые принципы:

1. Минимальные привилегии – каждый компонент получает ровно столько прав, сколько необходимо для работы.

2. Защита данных в движении и покое – использование шифрования (TLS, AES) и хеширования (SHA-256).

3. Глубинная защита (Defense in Depth) – многоуровневая система безопасности (брандмауэры, IDS/IPS, антивирусы).

4. Валидация и санитизация ввода – предотвращение инъекций через строгий контроль данных.

5. Аудит и мониторинг – журналирование событий и анализ аномалий в реальном времени.

6. Обновляемость – регулярное исправление уязвимостей в компонентах.

Пример реализации Secure by Design — использование микросервисной архитектуры с изолированными контейнерами (Docker, Kubernetes), где каждый сервис автономен и защищён отдельно. Это снижает риск распространения атак на всю систему.

Изучение архитектуры открытых систем и принципов безопасности позволяет создавать решения, сочетающие гибкость с устойчивостью к современным киберугрозам. Понимание рисков распределённых сред и следование принципам Secure by Design становятся обязательными навыками для разработчиков в условиях роста сложности атак. Интеграция безопасности в каждый этап жизненного цикла системы – не просто требование, но и необходимость для обеспечения доверия пользователей и соответствия регуляторным стандартам (GDPR, ISO 27001).

Таким образом, курс «Программирование компонентов открытых систем в защищённом исполнении» формирует компетенции, критически важные для построения цифровой инфраструктуры будущего.

Литература

1. Нестеров, С. А. Основы информационной безопасности / С. А. Нестеров. – 3-е изд., стер. – Санкт-Петербург: Лань, 2024. – 324 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/370967 (дата обращения: 09.10.2024). – ISBN 978-5-507-49077-6. – Текст : электронный (гл. 4).

2. Скулябина, О. В. Системный анализ в информационной безопасности: учеб. пособие / О. В. Скулябина, С. Ю. Страхов. – Санкт-Петербург: БГТУ «Военмех» им. Д. Ф. Устинова, 2021. – 50 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/220316 (дата обращения: 11.10.2024). – Текст : электронный (с. 8–45).

3. Вейцман, В. М. Проектирование информационных систем: учеб. пособие для вузов / В. М. Вейцман. – 2-е изд., стер. – Санкт-Петербург: Лань, 2022. – 316 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/208946 (дата обращения: 09.10.2024). – ISBN 978-5-8114-9982-3. – Текст : электронный. (темы 1, 11).

Контрольные вопросы

1. Дайте определение открытой системы. Какие три ключевые характеристики её отличают?
2. Почему совместимость (interoperability) критически важна для открытых систем? Приведите пример стандарта, обеспечивающего эту совместимость.
3. Как открытость архитектуры влияет на уязвимость системы к кибератакам? Объясните на примере.
4. Назовите три типа угроз, характерных для распределённых сред. Как атака DDoS эксплуатирует особенности таких систем?
5. Почему API часто становятся целью для злоумышленников в открытых системах? Какие меры защиты можно применить?
6. Чем опасен протокол HTTP в сравнении с HTTPS? Как это связано с угрозой «человек посередине»?
7. Объясните принцип «минимальных привилегий». Почему его соблюдение снижает риск несанкционированного доступа?
8. Что подразумевает концепция Defense in Depth? Приведите примеры инструментов для её реализации.
9. Как микросервисная архитектура и контейнеризация (например, Docker) способствуют безопасности открытых систем?
10. Почему валидация ввода данных считается обязательной практикой? Какие атаки она предотвращает?
11. Какие задачи решает аудит и мониторинг в контексте Secure by Design? Назовите инструменты для их реализации.

12. Как принципы Secure by Design связаны с регуляторными стандартами, такими как GDPR или ISO 27001?

Тема 1.2 Методы защиты данных на уровне кода

Перечень изучаемых вопросов

1. Шифрование данных в памяти и при передаче (AES, TLS).
2. Использование безопасных алгоритмов хеширования (SHA-3, ГОСТ Р 34.11-2012).
3. Практика работы с библиотеками криптографии (OpenSSL, Bouncy Castle).

Методические указания к изучению

В условиях роста киберугроз защита информации на уровне кода становится критически важной для предотвращения утечек и компрометации данных. Современные методы включают шифрование, использование безопасных алгоритмов хеширования и корректную работу с криптографическими библиотеками. В этой статье рассматриваются ключевые подходы к обеспечению безопасности данных непосредственно в процессе разработки программного обеспечения.

1. Шифрование данных в памяти и при передаче (AES, TLS)

Шифрование – это основа защиты конфиденциальности данных. На уровне кода применяются два основных типа:

- Шифрование в памяти (AES — Advanced Encryption Standard):
  - Симметричный алгоритм с длиной ключа 128, 192 или 256 бит.
  - Используется для защиты данных, хранящихся в оперативной памяти (например, паролей, токенов).
  - Пример реализации: шифрование конфиденциальных полей перед сохранением в базу данных.
- Шифрование при передаче (TLS – Transport Layer Security):
  - Обеспечивает безопасный обмен данными между клиентом и сервером.
  - Использует асимметричное шифрование для установки сессии и симметричное – для передачи данных.
  - Важно: актуальные версии протокола (TLS 1.3), отказ от устаревших (SSL 3.0).

Риски при неправильном использовании:

A. Слабые алгоритмы (например, DES вместо AES).
B. Неправильное управление ключами (хранение в открытом виде).

2. Использование безопасных алгоритмов хеширования (SHA-3, ГОСТ Р 34.11-2012)

Хеширование применяется для проверки целостности данных и безопасного хранения паролей. Критерии выбора алгоритма:

- Устойчивость к коллизиям – невозможность подбора двух входных данных с одинаковым хешем.

• Адаптивность к вычислительным мощностям – защита от перебора (соль, итерации).

Рекомендуемые алгоритмы:

• SHA-3 (Keccak):

– Современный стандарт, устойчивый к атакам на основе квантовых вычислений.

• ГОСТ Р 34.11-2012 («Стрибог»):

– Российский стандарт, используемый в государственных системах.

– Длина хеша – 256 или 512 бит.

Устаревшие алгоритмы, которых следует избегать: MD5, SHA-1.

Пример применения:

Хранение паролей в виде хеша с добавлением «соли» (salt) для предотвращения атак.

3. Практика работы с библиотеками криптографии (OpenSSL, Bouncy Castle)

Использование проверенных библиотек снижает риск ошибок в реализации криптографических методов.

Принципы работы:

1. Избегайте самописных решений – готовые библиотеки проходят аудит и обновления.

2. Актуальные версии – регулярное обновление для устранения уязвимостей.

3. Конфигурация безопасности – отключение слабых алгоритмов (например, RC4 в TLS).

Популярные библиотеки:

• OpenSSL:

– Реализация TLS/SSL, поддержка AES, RSA, эллиптических кривых.

– Используется в веб-серверах (Apache, Nginx).

• Bouncy Castle:

– Кроссплатформенная библиотека для Java и C#.

– Поддержка ГОСТ, PKCS#7, XML-шифрования.

Ошибки разработчиков:

• Использование режима ECB (Electronic Codebook) в AES вместо CBC или GCM.

• Неправильная генерация случайных чисел (например, через Math.random() вместо криптографически безопасных методов).

Литература

1. Нестеров, С. А. Основы информационной безопасности / С. А. Нестеров. – 3-е изд., стер. – Санкт-Петербург: Лань, 2024. – 324 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/370967 (дата обращения: 09.10.2024). – ISBN 978-5-507-49077-6. – Текст : электронный (гл. 2, 3).

2. Скулябина, О. В. Системный анализ в информационной безопасности:

учеб. пособие / О. В. Скулябина, С. Ю. Страхов. – Санкт-Петербург: БГТУ «Военмех» им. Д. Ф. Устинова, 2021. – 50 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/220316 (дата обращения: 11.10.2024). – Текст : электронный (с. 8–45).

3. Вейцман, В. М. Проектирование информационных систем: учеб. пособие для вузов / В. М. Вейцман. – 2-е изд., стер. – Санкт-Петербург: Лань, 2022. – 316 с. – Режим доступа: для авториз. пользователей. - Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/208946 (дата обращения: 09.10.2024). – ISBN 978-5-8114-9982-3. – Текст : электронный (темы 1, 11).

Контрольные вопросы

1. Чем отличается симметричное шифрование (AES) от асимметричного (RSA)? В каких сценариях применяется AES?
2. Почему TLS считается обязательным для защиты данных при передаче? Назовите минимум две угрозы, которые он предотвращает.
3. Какие риски возникают при использовании устаревших версий TLS (например, 1.0)?
4. Объясните, зачем добавлять «соль» (salt) при хешировании паролей. Приведите пример алгоритма, где это критично.
5. Почему SHA-3 считается более безопасным, чем SHA-2? В чём его ключевое отличие?
6. Для каких задач применяется ГОСТ Р 34.11-2012? В каких странах/системах он является стандартом?
7. Назовите три причины использовать OpenSSL вместо самописной реализации шифрования.
8. Какие уязвимости могут возникнуть при работе с библиотекой Bouncy Castle, если не обновлять её версию?
9. Почему режим ECB в AES считается небезопасным? Какой режим следует использовать вместо него?
10. Как обеспечить безопасное хранение ключей шифрования в коде приложения?
11. Какие алгоритмы хеширования нельзя использовать для защиты паролей? Обоснуйте ответ.
12. Приведите пример сценария, где необходимо комбинировать TLS и AES для защиты данных.

Тема 1.3 Основы программирования защищенных компонентов открытых систем

Перечень изучаемых вопросов

1. Реализация RBAC (Role-Based Access Control).
2. Интеграция с системами аутентификации (OAuth 2.0, OpenID Connect).

3. Минимизация прав приложений (Principle of Least Privilege).

Методические указания к изучению

Ролевые модели доступа и управление привилегиями Контроль доступа к данным и функционалу приложений – один из ключевых аспектов информационной безопасности. Неправильное управление правами пользователей и приложений может привести к утечкам данных, несанкционированным операциям и компрометации всей системы. В этой статье рассматриваются три фундаментальных подхода к защите данных на уровне кода: ролевое управление доступом (RBAC), интеграция с современными системами аутентификации и принцип минимальных привилегий.

1. Реализация RBAC (Role-Based Access Control)

RBAC – модель управления доступом, где права назначаются не пользователям напрямую, а их ролям в системе. Каждой роли соответствуют определённые разрешения, что упрощает администрирование и снижает риск ошибок.

Ключевые компоненты RBAC:

- Роли – группы прав (например, «Администратор», «Менеджер», «Гость»).
- Разрешения – действия, доступные роли (чтение, запись, удаление данных).
- Пользователи – назначаются на роли в зависимости от их функций.

Пример реализации:

- В корпоративной CRM-системе роль «Менеджер» позволяет просматривать и редактировать данные клиентов, но не удалять их.

Преимущества:

- Упрощение аудита прав («Кто имеет доступ к X?»).
- Быстрое изменение прав через редактирование ролей.
- Снижение риска «привилегированного creep» (накопления избыточных прав).

Лучшие практики:

- Регулярный пересмотр ролей.
- Запрет ролей с пересекающимися правами.

2. Интеграция с системами аутентификации (OAuth 2.0, OpenID Connect)

Современные приложения редко работают изолированно – они взаимодействуют с внешними сервисами и пользователями. Интеграция с протоколами аутентификации и авторизации обеспечивает безопасный доступ к ресурсам.

OAuth 2.0

Цель: Делегирование прав доступа третьим сторонам без передачи учётных данных. Сценарии:

– Авторизация приложений для доступа к API (например, доступ к Google Drive из стороннего сервиса).

– Выдача токенов с ограниченным сроком действия и правами.

OpenID Connect (OIDC)

- Цель: Аутентификация пользователей через доверенные провайдеры (Google, Microsoft).
- Особенности:

– Возвращает JWT-токен с информацией о пользователе (ID, email).

– Используется вместе с OAuth 2.0 для комбинированной аутентификации и авторизации.

Интеграция с RBAC:

- Роли могут назначаться на основе данных из токена OIDC (например, группа в Active Directory).
- Пример: Система предоставляет доступ к панели администратора только пользователям с ролью «admin» в токене.

Риски:

А. Утечка токенов.

В. Неправильная настройка scope в OAuth (избыточные права).

3. Минимизация прав приложений (Principle of Least Privilege)

Принцип минимальных привилегий (PoLP) — предоставление компонентам системы ровно тех прав, которые необходимы для выполнения их задач.

Применение на практике:

- Для пользователей: Ограничение доступа к данным и функциям, не связанным с их обязанностями.
- Для приложений:

1. Запрет доступа к файловой системе, кроме необходимых директорий.
2. Ограничение прав в базе данных (только SELECT, без DROP или UPDATE).

- Для микросервисов: Изоляция контейнеров (Docker) с отдельными правами.

Пример нарушения PoLP:

- Веб-приложение имеет полный доступ к базе данных. При SQL-инъекции злоумышленник получает возможность удалить таблицы.

Способы реализации:

- Использование ограниченных учётных записей для сервисов.
- Запуск приложений в sandbox-режиме.

Литература

1. Нестеров, С. А. Основы информационной безопасности / С. А. Нестеров. – 3-е изд., стер. – Санкт-Петербург: Лань, 2024. – 324 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/370967 (дата обращения: 09.10.2024). – ISBN 978-5-507-49077-6. – Текст : электронный (гл. 1–3).

2. Скулябина, О. В. Системный анализ в информационной безопасности: учеб. пособие / О. В. Скулябина, С. Ю. Страхов. – Санкт-Петербург: БГТУ «Военмех» им. Д. Ф. Устинова, 2021. – 50 с. – Режим доступа: для авториз.

пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/220316 (дата обращения: 11.10.2024). – Текст : электронный (с. 8–45).

3. Вейцман, В. М. Проектирование информационных систем: учеб. пособие для вузов / В. М. Вейцман. – 2-е изд., стер. – Санкт-Петербург: Лань, 2022. – 316 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/208946 (дата обращения: 09.10.2024). – ISBN 978-5-8114-9982-3. – Текст : электронный. (темы 1, 6, 7).

Контрольные вопросы

1. Объясните разницу между RBAC и ABAC (Attribute-Based Access Control). В каких сценариях предпочтительнее RBAC?
2. Как избежать конфликтов прав при создании ролей в RBAC? Приведите пример.
3. Чем отличается авторизация от аутентификации в контексте OAuth 2.0 и OpenID Connect?
4. Почему OAuth 2.0 не подходит для аутентификации пользователей? Как эту проблему решает OpenID Connect?
5. Какие данные содержит JWT-токен в OpenID Connect? Как они используются для назначения ролей?
6. Назовите три риска, связанных с неправильной настройкой scope в OAuth 2.0.
7. Как принцип минимальных привилегий предотвращает последствия SQL-инъекций?
8. Какие инструменты можно использовать для изоляции прав приложений в Docker?
9. Почему сервисные учётные записи должны иметь ограниченные права? Приведите пример из реальной системы.
10. Как совместить RBAC и PoLP в микросервисной архитектуре?
11. Какие метрики можно использовать для аудита соблюдения принципа минимальных привилегий?
12. Опишите сценарий, где нарушение PoLP привело к утечке данных.

1.4 Безопасная работа с внешними API и сервисами

Перечень изучаемых вопросов

1. Валидация входных данных (Sanitization, Input Validation).
2. Защита от атак (SQL Injection, XSS, CSRF).
3. Использование API-шлюзов с поддержкой безопасности (Kong, Apigee).

Методические указания к изучению

Безопасная работа с внешними API и сервисами Интеграция с внешними API и сервисами расширяет функционал приложений, но создаёт уязвимости, если не обеспечена должная защита. Злоумышленники часто используют слабые места во внешних взаимодействиях для атак на систему. В этой статье рассматриваются ключевые методы защиты данных при работе с API: валидация входных данных, защита от распространённых атак и использование специализированных API-шлюзов.

1. Валидация входных данных (Sanitization, Input Validation)

Любые данные, поступающие извне (от пользователей, партнёрских систем или API), должны рассматриваться как потенциально опасные.

Основные подходы:

- Валидация (Input Validation):

A. Проверка данных на соответствие ожидаемому формату (например, email, числовой диапазон).

B. Пример: регулярные выражения для проверки корректности телефонного номера.

- Санитизация (Sanitization):

A. Очистка данных от опасных символов или конструкций.

B. Пример: удаление тегов <script> из пользовательского ввода для предотвращения XSS.

Лучшие практики:

A. Использование белых списков (разрешение только известных безопасных паттернов).

B. Валидация на стороне сервера, даже если она выполнена на клиенте.

2. Защита от атак (SQL Injection, XSS, CSRF)

a) SQL Injection. Суть атаки: внедрение SQL-кода через входные данные (например, формы поиска). Пример уязвимого кода:

```
sql
Copy
"SELECT * FROM users WHERE login = '" + userInput + "';"
```

Если userInput = "admin'; DROP TABLE users;--", запрос удалит таблицу.

- Защита:

A. Использование параметризованных запросов (Prepared Statements).

B. ORM-библиотеки (Hibernate, SQLAlchemy), экранирующие входные данные.

b) XSS (Cross-Site Scripting)

- Суть атаки: внедрение JavaScript-кода, который выполняется в браузере жертвы.

- Пример:

```
html
Copy
<script>alert('XSS');</script>
Run HTML
```

• Защита:

A. Экранирование спецсимволов (например, замена < на <).

B. Использование Content Security Policy (CSP).

c) CSRF (Cross-Site Request Forgery)

Суть атаки: подделка запросов от имени авторизованного пользователя.

Пример: форма на вредоносном сайте, отправляющая запрос к банковскому приложению.

Защита:

A. CSRF-токены, генерируемые для каждой сессии.

B. Проверка заголовка Referer или Origin.

3. Использование API-шлюзов с поддержкой безопасности (Kong, Apigee)

API-шлюзы выступают промежуточным слоем между клиентами и серверными API, централизующим управление безопасностью.

Функции API-шлюзов:

• Аутентификация и авторизация:

– Интеграция с OAuth 2.0, JWT.

– Проверка прав доступа к эндпоинтам.

• Ограничение запросов (Rate Limiting):

– Защита от DDoS-атак и перегрузки сервисов.

– Шифрование трафика: принудительное использование HTTPS.

– Мониторинг и логирование: выявление подозрительных запросов.

Примеры решений:

A. Kong (поддержка плагинов для аутентификации, ACL, IP-фильтрации, гибкая конфигурация через REST API).

B. Google Apigee (анализ угроз в режиме реального времени; политики безопасности на основе машинного обучения.

Литература

1. Нестеров, С. А. Основы информационной безопасности / С. А. Нестеров. – 3-е изд., стер. – Санкт-Петербург: Лань, 2024. – 324 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/370967 (дата обращения: 09.10.2024). – ISBN 978-5-507-49077-6. – Текст : электронный (гл. 2, 3).

2. Скулябина, О. В. Системный анализ в информационной безопасности: учеб. пособие / О. В. Скулябина, С. Ю. Страхов. – Санкт-Петербург: БГТУ «Военмех» им. Д. Ф. Устинова, 2021. – 50 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/220316 (дата обращения: 11.10.2024). – Текст : электронный (с. 8–45).

3. Вейцман, В. М. Проектирование информационных систем: учеб. пособие для вузов / В. М. Вейцман. – 2-е изд., стер. – Санкт-Петербург: Лань, 2022. – 316 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/208946 (дата обращения: 09.10.2024). – ISBN 978-5-8114-9982-3. – Текст : электронный

(темы 1, 6, 7).

Контрольные вопросы

1. Чем отличается валидация данных от санитизации? Приведите примеры для каждого подхода.
2. Почему параметризованные запросы эффективны против SQL Injection? Объясните на примере кода.
3. Какие символы необходимо экранировать для защиты от XSS? Как это реализовать в веб-приложении?
4. Как CSRF-токены предотвращают подделку межсайтовых запросов? Опишите механизм работы.
5. Назовите три функции API-шлюза, которые повышают безопасность взаимодействия с внешними сервисами.
6. Почему проверки на стороне клиента недостаточно для защиты от вредоносного ввода?
7. Какие риски возникают при отсутствии Rate Limiting в API? Как их можно минимизировать?
8. Как Content Security Policy (CSP) защищает от XSS? Приведите пример настройки заголовка CSP.
9. Зачем API-шлюзы используют HTTPS принудительно? Какие угрозы это предотвращает?
10. Опишите сценарий, где уязвимость в API приводит к компрометации базы данных. Какие методы защиты могли бы это предотвратить?

Раздел 2: Распределенные базы данных как ядро открытых систем

Тема 2.1 Архитектура распределенных баз данных

Перечень изучаемых вопросов

– Выбор между согласованностью, доступностью и устойчивостью к разделению.

– Типы распределенных СУБД: Cassandra, CockroachDB, MongoDB.

Методические указания к изучению

Распределённые СУБД и CAP. В распределённых системах защита данных требует не только криптографии и контроля доступа, но и грамотного выбора архитектуры баз данных. CAP-теорема и особенности распределённых СУБД напрямую влияют на устойчивость системы к сбоям и атакам. В этой статье разбирается, как компромиссы между согласованностью, доступностью и устойчивостью к разделению (CAP) определяют безопасность данных, а также рассматриваются ключевые типы распределённых баз данных.

1. CAP-теорема и выбор между согласованностью, доступностью и устойчивостью к разделению

CAP-теорема утверждает, что распределённая система может одновременно гарантировать только два из трёх свойств:

- C (Consistency) – все узлы видят одни и те же данные в любой момент времени.
- A (Availability) – система всегда возвращает ответ, даже если часть узлов недоступна.
- P (Partition Tolerance) – система продолжает работать при сетевых сбоях (разделении сети).

Влияние на безопасность данных:

– CP-системы (Consistency + Partition Tolerance): гарантируют согласованность данных, даже если часть узлов недоступна.Пример: Финансовые системы, где критична точность данных (например, CockroachDB). Риски: Потеря доступности может быть использована для DDoS-атак.

– AP-системы (Availability + Partition Tolerance): обеспечивают доступность, но данные могут быть временно несогласованными. Пример: Социальные сети (Cassandra). Риски: Возможность чтения устаревших данных, что критично для транзакционных систем.

– CA-системы (Consistency + Availability): практически не существуют в распределённых средах, так как игнорируют разделение сети.

Выбор между C, A и P определяет, насколько система устойчива к атакам на доступность или целостность данных. Например, AP-системы уязвимы к атакам, эксплуатирующим несогласованность данных.

2. Типы распределённых СУБД: Cassandra, CockroachDB, MongoDB

a) Apache Cassandra

Тип: AP-система (Availability + Partition Tolerance).

Архитектура:

A. Masterless-кластер с линейной масштабируемостью.

B. Данные распределяются по узлам с использованием хеш-ринг-алгоритма.

Безопасность:

A. Встроенное шифрование данных на диске и при передаче.

B. Поддержка ролевого доступа (RBAC).

Использование: Высоконагруженные системы, где доступность важнее мгновенной согласованности (логирование, IoT).

b) CockroachDB

Тип: CP-система (Consistency + Partition Tolerance).

– Архитектура: распределённая реляционная СУБД с поддержкой ACID-транзакций, использует алгоритм Raft для консенсуса между узлами.

Безопасность:

A. Шифрование данных на уровне таблиц.

B. Интеграция с сертификатами TLS и OAuth 2.0.

Использование: Банковские системы, SaaS-платформы, где критична согласованность.

c) MongoDB

Тип: Гибридная система (CA/CP в зависимости от конфигурации).

Архитектура:

A. Документоориентированная СУБД с шардированием и репликацией.

B. Поддерживает транзакции в версиях 4.0+.

Безопасность:

– Шифрование на уровне поля (Field-Level Encryption).

– Аудит операций и детальный контроль доступа.

Использование: Контент-платформы, мобильные приложения.

3. Связь между CAP-теоремой и защитой данных

Согласованность (C): Гарантирует, что злоумышленник не сможет прочитать устаревшие или некорректные данные.

Доступность (A): Устойчивость к DDoS-атакам, но риск эксплуатации временных рассогласований.

Устойчивость к разделению (P): защита от сетевых атак, направленных на изоляцию узлов.

Пример: В AP-системе (Cassandra) злоумышленник может эксплуатировать временную несогласованность, чтобы получить доступ к устаревшим данным. В CP-системе (CockroachDB) такой риск снижен, но атака на доступность может парализовать часть узлов.

Литература

1. Нестеров, С. А. Основы информационной безопасности / С. А. Нестеров. – 3-е изд., стер. – Санкт-Петербург: Лань, 2024. – 324 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/370967 (дата обращения: 09.10.2024). – ISBN 978-5-507-49077-6. – Текст : электронный (гл. 1–3).

2. Скулябина, О. В. Системный анализ в информационной безопасности: учеб. пособие / О. В. Скулябина, С. Ю. Страхов. – Санкт-Петербург: БГТУ «Военмех» им. Д. Ф. Устинова, 2021. – 50 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/220316 (дата обращения: 11.10.2024). – Текст : электронный (с. 8–45).

3. Вейцман, В. М. Проектирование информационных систем: учеб. пособие для вузов / В. М. Вейцман. – 2-е изд., стер. – Санкт-Петербург: Лань, 2022. – 316 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/208946 (дата обращения: 09.10.2024). – ISBN 978-5-8114-9982-3. – Текст : электронный (темы 1, 6, 7).

Контрольные вопросы

1. Сформулируйте CAP-теорему. Какие два свойства может гарантировать распределённая система одновременно?

2. Почему CA-системы практически не используются в распределённых средах?

3. Как выбор AP-модели в Cassandra влияет на защиту данных? Какие угрозы становятся актуальными?

4. Объясните, как алгоритм Raft в CockroachDB обеспечивает согласованность данных.

5. Какие механизмы шифрования поддерживает MongoDB для защиты данных на уровне поля?

6. В чём разница между masterless-архитектурой (Cassandra) и шардированием с репликацией (MongoDB)?

7. Почему CockroachDB считается более подходящей для финансовых систем, чем Cassandra?

8. Как сетевой раздел (Partition) может быть использован для атаки на распределённую СУБД?

9. Какие компромиссы безопасности возникают при выборе между AP и CP моделями?

10. Опишите сценарий, где нарушение принципов CAP-теоремы приводит к утечке данных.

Тема 2.2 Обеспечение безопасности данных в распределенных БД

Перечень изучаемых вопросов

1. Шифрование данных на уровне полей (Field-Level Encryption).
2. Аудит изменений (Change Data Capture, CDC).
3. Реализация транзакций с поддержкой ACID в распределенных средах.

Методические указания к изучению

1. Шифрование данных на уровне полей (Field-Level Encryption)

Суть метода: Шифрование отдельных полей (например, номера телефона, паспорта) перед сохранением в БД. Даже при компрометации сервера злоумышленник не сможет прочитать зашифрованные данные без ключей.

Технологии и примеры:

- Прозрачное шифрование (TDE):

– шифрует данные на диске автоматически (используется в MongoDB, SQL Server).

– минус: администратор БД имеет доступ к ключам.

- Клиентское шифрование:

– данные шифруются на стороне приложения (например, с помощью библиотек AWS Encryption SDK).

– ключи хранятся в аппаратных модулях безопасности (HSM).

Преимущества:

- Соответствие стандартам (например, GDPR для персональных данных).
- Защита от утечек при взломе СУБД.

Пример реализации: В MongoDB Field-Level Encryption позволяет шифровать поля через JSON-схемы. Расшифровка происходит только при авторизованном доступе.

2. Аудит изменений (Change Data Capture, CDC). CDC — механизм отслеживания изменений в данных (вставка, обновление, удаление) и их записи в лог. Используется для анализа подозрительных операций и восстановления данных. Сценарии применения:

- Обнаружение аномалий: резкое удаление большого объёма данных может сигнализировать о атаке.
- Юридический аудит: требуется для доказательства целостности данных в суде.
- Синхронизация между системами: например, перенос изменений из основной БД в аналитическую.

Инструменты:

- Debezium: Отслеживает изменения в PostgreSQL, MySQL через бинарные логи.
- AWS DMS: Фиксирует изменения в облачных БД и реплицирует их.

Пример: при SQL-инъекции, изменившей баланс пользователей, CDC позволяет определить точное время и параметры вредоносного запроса.

3. Реализация транзакций с поддержкой ACID в распределённых средах

ACID — набор свойств, гарантирующих надежность операций:

- Atomicity (Атомарность): Транзакция выполняется целиком или откатывается.
- Consistency (Согласованность): Данные всегда соответствуют правилам БД.
- Isolation (Изоляция): Параллельные транзакции не влияют друг на друга.
- Durability (Долговечность): Результаты транзакции сохраняются после сбоев.

Проблемы в распределённых системах:

- Задержки сети: Могут привести к конфликтам при параллельных транзакциях.
- Распределённые блокировки: Сложность управления в условиях CAP-теоремы.

Решения:

- 2PC (Two-Phase Commit): Координатор подтверждает commit только после согласия всех узлов.
- Sagas: Длинные транзакции разбиваются на этапы с компенсационными действиями при ошибках.

Пример: в CockroachDB используется алгоритм Raft для достижения консенсуса между узлами, что позволяет сохранять ACID-свойства даже при сетевых разделах.

Литература

1. Нестеров, С. А. Основы информационной безопасности / С. А. Несте-

ров. – 3-е изд., стер. – Санкт-Петербург: Лань, 2024. – 324 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/370967 (дата обращения: 09.10.2024). – ISBN 978-5-507-49077-6. – Текст : электронный (гл. 1–3).

2. Скулябина, О. В. Системный анализ в информационной безопасности: учеб. пособие / О. В. Скулябина, С. Ю. Страхов. – Санкт-Петербург: БГТУ «Военмех» им. Д. Ф. Устинова, 2021. – 50 с. – Режим доступа: для авториз. пользователей. - Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/220316 (дата обращения: 11.10.2024). – Текст : электронный (с. 8–45).

3. Вейцман, В. М. Проектирование информационных систем: учеб. пособие для вузов / В. М. Вейцман. – 2-е изд., стер. – Санкт-Петербург: Лань, 2022. – 316 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/208946 (дата обращения: 09.10.2024). — ISBN 978-5-8114-9982-3. – Текст : электронный (темы 1, 6, 7).

Контрольные вопросы

1. Чем отличается прозрачное шифрование (TDE) от клиентского шифрования? В каком сценарии предпочтительнее TDE?
2. Почему Field-Level Encryption критически важен для соответствия GDPR? Приведите пример поля, которое должно быть зашифровано.
3. Какие риски возникают при хранении ключей шифрования на том же сервере, что и база данных?
4. Как Change Data Capture помогает в расследовании инцидентов информационной безопасности?
5. Назовите три инструмента для реализации CDC и опишите их ключевые особенности.
6. Почему в распределённых системах сложно обеспечить изоляцию транзакций (Isolation)? Как это решается в CockroachDB?
7. Чем отличается подход Sagas от 2PC при обработке распределённых транзакций?
8. Какие проблемы ACID-свойств могут возникнуть в AP-системах (например, Cassandra)?
9. Как аудит изменений связан с принципом минимальных привилегий (PoLP)?
10. Опишите сценарий, где отсутствие шифрования на уровне полей привело к утечке персональных данных.

Тема 2.3 Репликация и синхронизация данных

Перечень изучаемых вопросов

1. Механизмы репликации (Master-Slave, Multi-Master).
2. Конфликты данных и методы их разрешения (CRDT, Last Write Wins).

3. Защита каналов синхронизации (TLS, VPN).

Методические указания к изучению

Репликация данных – ключевой механизм обеспечения отказоустойчивости и доступности в распределённых системах. Однако синхронизация между узлами создаёт риски конфликтов и уязвимостей. В этой статье разбираются методы репликации, стратегии разрешения конфликтов и защита каналов передачи данных.

1. Механизмы репликации (Master-Slave, Multi-Master)

Репликация позволяет копировать данные между узлами для повышения производительности и устойчивости к сбоям.

a) Master-Slave (Primary-Secondary).

- Архитектура:
  - Один узел (Master) принимает запросы на запись.
  - Slave-узлы реплицируют данные с Master и обслуживают чтение.
- Преимущества:
  - Простота управления (одна точка записи).
  - Согласованность данных гарантирована (Slave синхронизируются с Master).
- Недостатки:
  - Риск потери доступности при отказе Master.
  - Высокая нагрузка на Master.
- Примеры: MySQL Replication, PostgreSQL Streaming Replication.

b) Multi-Master

- Архитектура:
  - Все узлы равноправны: запись возможна в любой узел.
  - Данные синхронизируются между всеми узлами асинхронно.
- Преимущества:
  - Высокая доступность (нет единой точки отказа).
  - Распределение нагрузки на запись.
- Недостатки:
  - Риск конфликтов данных при одновременной записи в разные узлы.
  - Сложность обеспечения согласованности.
- Примеры: Cassandra, CockroachDB.

Таблица 2 – Сравнение способов репликации

Параметр	Master-Slave	Multi-Master
Согласованность	Высокая (синхронная)	Слабее (асинхронная)
Доступность	Зависит от Master	Высокая
Сложность разрешения конфликтов	Нет	Высокая

2. Конфликты данных и методы их разрешения (CRDT, Last Write Wins)

В Multi-Master системах конфликты возникают, когда разные узлы изменяют одни и те же данные одновременно.

a) Last Write Wins (LWW)

Принцип: побеждает запись с последней временной меткой.

Плюсы: простота реализации, низкие накладные расходы.

Минусы: потеря данных: более ранние изменения перезаписываются.

Существует риск несогласованности часов узлов (требует NTP). Пример: cassandra использует LWW по умолчанию.

b) Conflict-Free Replicated Data Types (CRDT). Принцип ее работы: структуры данных, гарантирующие сходимость без явного разрешения конфликтов.

Типы:

1. Счётчики (например, Grow-Only Counter).
2. Списки с уникальными идентификаторами элементов.

Плюсы:

1. Автоматическое разрешение конфликтов.
2. Идеально для распределённых систем с высокой доступностью (AP в CAP).

Минусы:

1. Ограниченные типы данных.
2. Высокий объём метаданных.

Пример: Riak KV с CRDT для корзин (buckets).

3. Защита каналов синхронизации (TLS, VPN)

Данные при репликации уязвимы к перехвату, подмене или MITM-атакам.

a) TLS (Transport Layer Security). Цель: Шифрование трафика между узлами. Настройка использует сертификаты для аутентификации узлов.

Принудительное использование TLS 1.3. Пример:

bash

Copy

 # Настройка TLS в PostgreSQL (postgresql.conf)

ssl = on

ssl_cert_file = '/etc/certs/server.crt'

ssl_key_file = '/etc/certs/server.key'

b) VPN (Virtual Private Network)

Цель: Создание защищённого туннеля между узлами в публичных сетях. Использование подразумевает применение WireGuard или OpenVPN для шифрования всего трафика.

Изоляция репликации в приватной подсети. Пример: Репликация между облачными инстансами через VPN.

Риски при отсутствии защиты:

1. Перехват паролей и токенов аутентификации.
2. Внедрение вредоносных данных в поток репликации.

Литература

1. Нестеров, С. А. Основы информационной безопасности / С. А. Нестеров. – 3-е изд., стер. – Санкт-Петербург: Лань, 2024. – 324 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/370967 (дата обращения: 09.10.2024). – ISBN 978-5-507-49077-6. – Текст : электронный (гл. 2, 3).

2. Скулябина, О. В. Системный анализ в информационной безопасности: учеб. пособие / О. В. Скулябина, С. Ю. Страхов. – Санкт-Петербург: БГТУ «Военмех» им. Д. Ф. Устинова, 2021. – 50 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/220316 (дата обращения: 11.10.2024). – Текст : электронный (с. 8–45).

3. Вейцман, В. М. Проектирование информационных систем: учеб. пособие для вузов / В. М. Вейцман. – 2-е изд., стер. – Санкт-Петербург: Лань, 2022. – 316 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/208946 (дата обращения: 09.10.2024). – ISBN 978-5-8114-9982-3. – Текст : электронный (темы 1, 6, 7, 11).

Контрольные вопросы

1. Чем отличается роль Master от Slave в репликации Master-Slave? Какие операции разрешены Slave?
2. Почему в Multi-Master системах чаще возникают конфликты данных? Приведите пример сценария.
3. Какие проблемы могут возникнуть при использовании LWW в системе с рассинхронизированными часами узлов?
4. Объясните, как CRDT гарантирует сходимость данных без явного разрешения конфликтов.
5. Почему Cassandra по умолчанию использует LWW, а не CRDT?
6. Назовите три типа данных, для которых эффективно применение CRDT.
7. Какие настройки TLS необходимы для защиты канала репликации в PostgreSQL?
8. В чём преимущество VPN перед TLS для защиты синхронизации данных?
9. Как асимметричное шифрование используется в TLS для аутентификации узлов?
10. Опишите сценарий, где отсутствие шифрования канала репликации привело к утечке данных.

Тема 2.4 Интеграция распределенных БД с защищенными компонентами

Перечень изучаемых вопросов

1. Использование сервисной шины (Service Bus) для безопасного обмена данными.
2. Реализация паттерна CQRS (Command Query Responsibility Segregation).
3. Миграция данных с сохранением целостности и конфиденциальности.

Методические указания к изучению

Современные распределённые системы требуют не только высокой доступности, но и строгого соблюдения принципов информационной безопасности. Интеграция баз данных с защищёнными компонентами подразумевает использование специализированных архитектурных паттернов, безопасных протоколов передачи данных и методов миграции, исключающих утечки. В этой статье рассматриваются ключевые подходы к построению таких систем.

1. Использование сервисной шины (Service Bus) для безопасного обмена данными.

Сервисная шина – это промежуточный слой, обеспечивающий взаимодействие между компонентами системы через единую точку входа. Она управляет маршрутизацией сообщений, аутентификацией и шифрованием данных.

Механизмы безопасности в Service Bus:

1. Аутентификация и авторизация:

1.1 Интеграция с OAuth 2.0, JWT или сертификатами TLS. Пример: Azure Service Bus использует Shared Access Signatures (SAS) для контроля прав доступа.

2. Шифрование данных:

2.1 TLS для защиты трафика между отправителем и шиной, а также между шиной и потребителем.

2.2 Шифрование сообщений на уровне поля (например, с использованием AES-256).

3. Изоляция каналов:

A. Виртуальные сети (VNet) и приватные конечные точки для исключения доступа из публичного интернета.

Примеры технологий:

RabbitMQ с плагином TLS.

Kafka с SASL/SCRAM аутентификацией и шифрованием через SSL.

Преимущества:

B. Централизованный аудит операций.

C. Защита от MITM-атак и подмены сообщений.

4. Реализация паттерна CQRS (Command Query Responsibility Segregation)

CQRS – архитектурный паттерн, разделяющий операции записи (команды) и чтения (запросы) данных. Это позволяет оптимизировать производительность и безопасность. Применение для защиты данных:

4.1 Разделение прав доступа: команды (изменение данных) доступны только авторизованным службам или пользователям с повышенными привилегиями; запросы (чтение) могут использовать упрощённые проверки.

4.2 Изоляция моделей:

A. Модель записи (Command Model) работает с валидацией бизнес-правил.

B. Модель чтения (Query Model) оптимизирована для быстрого доступа, часто через материализованные представления. Пример: команда «Перевести деньги» требует многофакторной аутентификации.

Запрос «Показать баланс» доступен после базовой проверки токена.

Инструменты:

1. Event Sourcing (Axon Framework) для отслеживания изменений.

2. MediatR в .NET для разделения команд и запросов.

3. Миграция данных с сохранением целостности и конфиденциальности.

Миграция данных между системами — критически уязвимый этап, требующий защиты от потерь и утечек.

Методы обеспечения безопасности:

1. Шифрование данных в движении и покое:

Использование TLS для передачи и AES-256 для хранения временных дампов. Пример: AWS Database Migration Service (DMS) с поддержкой шифрования через AWS KMS.

2. Верификация целостности:

2.1Контрольные суммы (CRC32, SHA-256) для проверки отсутствия изменений при переносе.

2.2 Транзакционная миграция с откатом при ошибках.

3. Маскирование данных: замена чувствительных данных (номера карт, персональные идентификаторы) на анонимные значения перед переносом в тестовые среды.

Инструменты:

1. pg_dump с опцией шифрования для PostgreSQL.

2. AWS S3 + Server-Side Encryption для хранения резервных копий.

Риски при игнорировании защиты:

1. Утечка данных через незащищённые временные файлы.

2. Повреждение данных из-за сетевых сбоев.

Литература

1. Нестеров, С. А. Основы информационной безопасности / С. А. Нестеров. – 3-е изд., стер. – Санкт-Петербург: Лань, 2024. – 324 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/370967 (дата обращения: 09.10.2024). – ISBN 978-5-507-49077-6. – Текст : электронный (гл. 1–3).

2. Скулябина, О. В. Системный анализ в информационной безопасности: учеб. пособие / О. В. Скулябина, С. Ю. Страхов. – Санкт-Петербург: БГТУ «Военмех» им. Д. Ф. Устинова, 2021. – 50 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/220316 (дата обращения: 11.10.2024). – Текст : электронный (с. 8–45).

3. Вейцман, В. М. Проектирование информационных систем: учеб. пособие для вузов / В. М. Вейцман. – 2-е изд., стер. – Санкт-Петербург: Лань, 2022. – 316 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/208946 (дата обращения: 09.10.2024). – ISBN 978-5-8114-9982-3. — Текст : электронный (темы 1, 6, 7).

Контрольные вопросы

1. Какие механизмы аутентификации поддерживает Azure Service Bus? Как SAS повышает безопасность?
2. Объясните, как TLS защищает данные в сервисной шине. Какие версии протокола считаются устаревшими?
3. Чем отличается обработка команд от запросов в CQRS? Как это разделение влияет на безопасность?
4. Почему Event Sourcing часто сочетают с CQRS? Приведите пример из финансовой системы.
5. Какие риски возникают при миграции данных без шифрования временных файлов?
6. Как контрольные суммы (например, SHA-256) помогают сохранить целостность данных при переносе?
7. Зачем маскировать данные перед миграцией в тестовую среду? Какие методы маскирования вы знаете?
8. Назовите три преимущества использования Kafka с SASL/SCRAM вместо базовой настройки.
9. Как CQRS предотвращает утечки данных через интерфейсы чтения?
10. Опишите сценарий, где отсутствие изоляции каналов в сервисной шине привело к MITM-атаке.

Тема 3.1 Методы отладки защищенных распределённых компонентов открытых систем

Перечень изучаемых вопросов

1. Использование трассировки (Distributed Tracing: Jaeger, Zipkin).
2. Логирование и мониторинг (ELK Stack, Prometheus).
3. Анализ сетевого трафика (Wireshark, tcpdump).

Методические указания к изучению

Отладка распределённых систем – сложная задача из-за их масштабируемости, асинхронности и возможных уязвимостей в безопасности. Рассмотрим три ключевых метода: трассировку, логирование/мониторинг и анализ сетевого трафика.

Distributed Tracing – метод отслеживания запросов в распределённых системах, позволяющий анализировать задержки, ошибки и взаимодействия между сервисами.

Применение:

- Поиск узких мест в производительности.
- Анализ ошибок в микросервисной архитектуре.
- Обнаружение аномалий (например, несанкционированных вызовов API).

Инструменты: Jaeger.

- Компоненты:
  - Agent – сбор трейсов.
  - Collector – обработка и сохранение данных.
  - Query – поиск и визуализация.
  - Storage (Elasticsearch, Cassandra).
- Особенности:
  - Поддержка OpenTelemetry.
  - Гибкость в настройке.

Zipkin

- Проще в развёртывании, но менее функционален, чем Jaeger.
- Использует Spring Cloud Sleuth для интеграции с Java-приложениями.

Пример использования

```
bash
Copy
# Запуск Jaeger в Docker
docker run -d --name jaeger \
  -p 6831:6831/udp -p 16686:16686 \
  jaegertracing/all-in-one
```

После этого трейсы можно просматривать в веб-интерфейсе (http://localhost:16686).

1. Логирование и мониторинг (ELK Stack, Prometheus).
2. Централизованное логирование (ELK Stack)

ELK (Elasticsearch + Logstash + Kibana) — стек для сбора, обработки и визуализации логов.

- Elasticsearch – поиск и индексация.
- Logstash/Fluentd/Filebeat – сбор и фильтрация логов.
- Kibana – дашборды и аналитика.

Пример конфигурации Filebeat → Elasticsearch:

```
yaml
Copy
```

```
filebeat.inputs:
- type: log
  paths: /var/log/*.log

output.elasticsearch:
  hosts: ["localhost:9200"]
```

Мониторинг метрик (Prometheus + Grafana)

Prometheus – система мониторинга с pull-моделью (сам собирает метрики).

- Метрики: CPU, RAM, HTTP-запросы, кастомные метрики.
- Alertmanager – уведомления о проблемах.

Grafana — визуализация данных из Prometheus, Elasticsearch и др.

Пример запроса в PromQL:

```
promql
Copy
http_requests_total{status="500"} # подсчёт ошибок 500
```

Рассмотрите анализ сетевого трафика (Wireshark, tcpdump), перехват трафика, tcpdump (консольный инструмент)

```
bash
Copy
tcpdump -i eth0 port 443 -w traffic.pcap  # запись HTTPS-трафика
tcpdump -n 'tcp[tcpflags] & (syn|ack) == syn' # обнаружение сканирования портов
```

Рассмотрите сниффер Wireshark (GUI). Фильтры:

- http.request – HTTP-запросы.
- tcp.port == 22 – SSH-трафик.
- ip.src == 192.168.1.1 – трафик от определённого IP.

3.2. Анализ атак:

- DDoS – множественные SYN-запросы.
- MITM – подозрительные ARP-пакеты.
- Утечки данных – нестандартные DNS-запросы.

Пример анализа в Wireshark:

1. Открыть .pcap файл.
2. Применить фильтр dns для проверки DNS-трафика.
3. Найти аномалии (например, запросы к неизвестным доменам).

Литература

1. Нестеров, С. А. Основы информационной безопасности / С. А. Нестеров. – 3-е изд., стер. – Санкт-Петербург: Лань, 2024. – 324 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/370967 (дата обращения: 09.10.2024). – ISBN 978-5-507-49077-6. – Текст : электронный (гл. 2, 3).

2. Скулябина, О. В. Системный анализ в информационной безопасности: учеб. пособие / О. В. Скулябина, С. Ю. Страхов. – Санкт-Петербург: БГТУ

«Военмех» им. Д. Ф. Устинова, 2021. – 50 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/220316 (дата обращения: 11.10.2024). – Текст : электронный (с. 8–45).

3. Вейцман, В. М. Проектирование информационных систем: учеб. пособие для вузов / В. М. Вейцман. – 2-е изд., стер. – Санкт-Петербург: Лань, 2022. – 316 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/208946 (дата обращения: 09.10.2024). – ISBN 978-5-8114-9982-3. – Текст : электронный (темы 1, 6, 7, 11).

Контрольные вопросы

1. Что такое Distributed Tracing и как он помогает в отладке микросервисов? (Теория)
2. Назовите ключевые компоненты Jaeger и их функции. (Архитектура)
3. В чём отличие между Jaeger и Zipkin? Когда стоит выбрать каждый из них? (Сравнение инструментов)
4. Как OpenTelemetry упрощает трассировку в гетерогенных системах? (Стандарты)
5. Опишите роли Elasticsearch, Logstash и Kibana в стеке ELK. (Базовая настройка)
6. Почему Filebeat иногда используют вместо Logstash? Какие у этого преимущества? (Оптимизация)
7. Как Prometheus собирает метрики? Объясните разницу между pull и push-моделями. (Принципы работы)
8. Какие типы графиков в Grafana наиболее полезны для мониторинга безопасности? (Визуализация)
9. Как в Wireshark отфильтровать только HTTP-запросы с кодом ответа 500? (Практика фильтрации)
10. Какая команда tcpdump сохранит трафик с порта 443 в файл? (Консольные команды)
11. Как обнаружить сканирование портов с помощью Wireshark? (Анализ атак)
12. Какие признаки в сетевом трафике могут указывать на DDoS-атаку? (Безопасность).

Тема 3.2 Выявление уязвимостей в защищенных компонентах

Перечень изучаемых вопросов

1. Статический анализ кода (SonarQube, Checkmarx).
2. Динамический анализ (Fuzzing, Penetration Testing).
3. Тестирование на устойчивость к атакам (Chaos Engineering).

Методические указания к изучению

В современном мире кибербезопасности защита приложений требует комплексного подхода. Рассмотрите три ключевых методики выявления уязвимостей, которые должны быть в арсенале каждого специалиста по безопасности.

Статический анализ кода (SAST). Статический анализ кода представляет собой процесс проверки исходного кода без его выполнения. Этот метод особенно эффективен на ранних этапах разработки, когда исправление ошибок требует наименьших затрат.

Среди популярных инструментов стоит выделить SonarQube и Checkmarx. SonarQube – это открытая платформа для непрерывного анализа качества кода. Она поддерживает более 25 языков программирования и может интегрироваться в процесс CI/CD. Главное преимущество SonarQube – его расширяемость через плагины и относительно низкий порог входа.

Checkmarx, в свою очередь, предлагает более глубокий анализ, особенно в плане выявления уязвимостей безопасности. Его система отслеживает потоки данных через все уровни приложения, что позволяет находить сложные межмодульные уязвимости. Однако этот инструмент требует более серьезной настройки и обучения.

Основная проблема статического анализа - ложные срабатывания. Разработчики часто сталкиваются с ситуацией, когда инструмент указывает на потенциальную уязвимость, которая на самом деле не может быть эксплуатирована. Это требует дополнительного времени на верификацию результатов.

Динамический анализ. Динамический анализ проводится на работающем приложении и включает два основных подхода: фаззинг и пентестинг.

Фаззинг – это автоматизированная техника тестирования, при которой в программу подаются неожиданные, случайные или специально сформированные входные данные. Современные фаззеры, такие как AFL или libFuzzer, используют эволюционные алгоритмы для «обучения» и генерации все более эффективных тестовых случаев. Они особенно хороши для поиска уязвимостей, связанных с обработкой памяти, таких как переполнение буфера или использование после освобождения.

Пентестинг (тестирование на проникновение) – это имитация действий злоумышленника. В отличие от автоматизированного сканирования, пентестинг часто включает ручную работу специалиста, который пытается найти логические уязвимости в приложении. Инструменты вроде Burp Suite или OWASP ZAP помогают тестировщику, но не заменяют его экспертизу.

Особенность динамического анализа в том, что он может выявить уязвимости, которые невозможно обнаружить при статическом анализе, например, проблемы конфигурации или логические ошибки в бизнес-процессах. Однако он требует работающей системы и часто не покрывает весь код.

Тестирование на устойчивость (Chaos Engineering)

Chaos Engineering – это относительно новая дисциплина, которая выходит за рамки традиционного тестирования безопасности. Ее цель – проверить, как система ведет себя в условиях нештатных ситуаций. Основная идея заключается в преднамеренном внесении сбоев в работающую систему для проверки ее устойчивости. Это может быть отключение серверов, создание сетевых задержек, заполнение дискового пространства или имитация DDoS-атак. Инструменты вроде Chaos Monkey от Netflix или Gremlin позволяют автоматизировать эти процессы.

Важное отличие Chaos Engineering от традиционного тестирования – его проводят преимущественно в production-среде, но с важными ограничениями. Во-первых, эксперименты должны быть контролируемыми и обратимыми. Во-вторых, их проводят постепенно, начиная с наименее критичных компонентов. И в-третьих, обязательно наличие системы мониторинга, которая позволит быстро обнаружить и устранить проблемы.

Литература

1. Нестеров, С. А. Основы информационной безопасности / С. А. Нестеров. – 3-е изд., стер. – Санкт-Петербург: Лань, 2024. – 324 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/370967 (дата обращения: 09.10.2024). – ISBN 978-5-507-49077-6. – Текст : электронный (гл. 2, 3).

Контрольные вопросы

1. Каковы основные преимущества и недостатки статического анализа кода по сравнению с динамическим?
2. Как SonarQube помогает поддерживать качество кода в долгосрочной перспективе?
3. Какие типы уязвимостей лучше выявляет Checkmarx, чем SonarQube?
4. В чем принципиальное отличие dumb fuzzing от smart fuzzing?
5. Какие этапы включает в себя профессиональный пентест веб-приложения?
6. Почему фаззинг особенно эффективен для поиска уязвимостей в системном ПО?
7. Каковы основные принципы безопасного проведения Chaos Engineering экспериментов?
8. Как Chaos Monkey интегрируется с облачными платформами?
9. Какие метрики следует отслеживать при проведении тестов на устойчивость?
10. Почему комбинация разных методов тестирования дает лучшие результаты, чем использование какого-то одного подхода?

Тема 3.3 Отладка асинхронных и параллельных процессов

Перечень изучаемых вопросов

1. Работа с race condition и deadlock в распределенных системах.
2. Инструменты для анализа многопоточности (VisualVM, Intel VTune).
3. Тестирование edge cases в микросервисной архитектуре.

Методические указания к изучению

В эпоху распределенных систем и микросервисов разработчики все чаще сталкиваются с проблемами отладки асинхронных и параллельных процессов. Эти вызовы требуют особого подхода и специализированных инструментов.

Работа с race condition и deadlock в распределенных системах. Race condition (состояние гонки) и deadlock (взаимная блокировка) – одни из самых коварных проблем в параллельном программировании. В распределенных системах они проявляются особенно ярко из-за задержек сети и отсутствия общего состояния.

Состояние гонки возникает, когда несколько процессов или потоков пытаются изменить общие данные одновременно, и результат зависит от порядка выполнения операций. Классический пример – банковский перевод, когда два запроса на списание средств приходят практически одновременно. Без должной синхронизации это может привести к некорректному итоговому балансу.

Взаимная блокировка случается, когда два или более процесса ожидают освобождения ресурсов, занятых друг другом. В распределенных системах это усугубляется тем, что традиционные механизмы блокировок (мьютексы, семафоры) не всегда применимы.

Для предотвращения этих проблем применяют несколько стратегий:

1. Оптимистичные блокировки с проверкой версий данных
2. Распределенные транзакции (например, через SAGA-паттерн)
3. Использование event sourcing для обеспечения согласованности
4. Внедрение механизмов таймаутов и автоматического разблокирования

Особое внимание стоит уделить идемпотентности операций – свойству, позволяющему безопасно повторять запросы при сбоях сети или таймаутах.

Инструменты для анализа многопоточности

Для анализа проблем параллелизма существуют специализированные инструменты, каждый со своей областью применения.

VisualVM – бесплатный визуальный инструмент, входящий в комплект JDK. Он позволяет:

1. Мониторить состояние потоков в реальном времени.
2. Выявлять «зависшие» потоки.
3. Анализировать использование памяти.
4. Проводить профилирование CPU.

Intel VTune Amplifier – более мощное решение для глубокого анализа производительности. Его ключевые особенности:

1. Детальное профилирование на уровне процессорных инструкций

2. Анализ использования кэша.

3. Выявление «горячих точек» в коде.

4. Поддержка различных языков программирования.

Для распределенных систем особенно полезны распределенные трейсеры вроде Jaeger или Zipkin. Они позволяют отслеживать запросы через несколько сервисов и выявлять узкие места в асинхронных взаимодействиях.

Тестирование edge cases в микросервисной архитектуре

Тестирование граничных случаев (edge cases) в микросервисной архитектуре – отдельное искусство. Вот несколько типичных сценариев, которые стоит учитывать:

Сетевые проблемы:

1. Задержки между сервисами.

2. Частичная недоступность зависимостей.

3. Пакеты, приходящие в неправильном порядке.

Проблемы согласованности:

1. Конфликтующие обновления данных.

2. Расхождение состояний между репликами.

3. Проблемы часов в распределенной системе.

Особое внимание стоит уделить тестированию сценариев восстановления:

1. Как система ведет себя после длительного даунтайма.

2. Восстановление после потери сообщений.

3. Поведение при переполнении очередей.

Для таких тестов полезны инструменты вроде Chaos Monkey, которые могут искусственно создавать подобные ситуации в контролируемой среде.

Литература

1. Нестеров, С. А. Основы информационной безопасности / С. А. Нестеров. – 3-е изд., стер. – Санкт-Петербург: Лань, 2024. – 324 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/370967 (дата обращения: 09.10.2024). – ISBN 978-5-507-49077-6. – Текст : электронный (гл 4).

2. Скулябина, О. В. Системный анализ в информационной безопасности: учеб. пособие / О. В. Скулябина, С. Ю. Страхов. – Санкт-Петербург: БГТУ «Военмех» им. Д. Ф. Устинова, 2021. – 50 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/220316 (дата обращения: 11.10.2024). – Текст : электронный (с. 8–45).

3. Вейцман, В. М. Проектирование информационных систем: учеб. пособие для вузов / В. М. Вейцман. – 2-е изд., стер. – Санкт-Петербург: Лань, 2022. – 316 с. – Режим доступа: для авториз. пользователей. - Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/208946 (дата обращения: 09.10.2024). — ISBN 978-5-8114-9982-3. – Текст : электронный (тема 11).

Контрольные вопросы

1. Как отличить race condition от deadlock на практике?
2. Какие преимущества дает event sourcing при отладке асинхронных процессов?
3. Почему традиционные блокировки плохо подходят для распределенных систем?
4. Как VisualVM помогает выявлять проблемы с потоками?
5. В чем отличие анализа производительности в Intel VTune от простого профайлера?
6. Какие edge cases чаще всего упускают при тестировании микросервисов?
7. Как правильно тестировать поведение системы при частичной недоступности зависимостей?
8. Почему важно учитывать расхождение часов в распределенных системах?
9. Какие метрики стоит отслеживать при анализе параллельных процессов?
10. Как организовать тестирование edge cases без риска для production?

Тема 3.4 Восстановление после сбоев и обеспечение отказоустойчивости

Перечень изучаемых вопросов

1. Реализация механизмов retry, circuit breaker, bulkhead.
2. Использование резервных копий и реплик для восстановления.
3. Аудит инцидентов и постмортем-анализ (Postmortem Analysis).

Методические указания к изучению

В современной цифровой экосистеме, где простои систем могут обходиться компаниям в миллионы долларов, вопросы отказоустойчивости и восстановления после сбоев выходят на первый план. Эта статья рассматривает ключевые механизмы обеспечения надежности систем, подходы к резервированию и важность анализа инцидентов.

Реализация механизмов отказоустойчивости. Паттерн Retry (повторная попытка) — фундаментальный механизм обработки временных сбоев. Его эффективная реализация требует тонкой настройки нескольких параметров. Экспоненциальная задержка между попытками (exponential backoff) позволяет избежать перегрузки системы. Например, первая повторная попытка через 100 мс, вторая через 200 мс, третья через 400 мс и так далее. Важно установить разумный лимит количества попыток и учитывать идемпотентность операций – возможность их безопасного повторения без побочных эффектов.

Circuit Breaker (автоматический выключатель) действует по аналогии с электрическим предохранителем. Когда количество ошибок превышает пороговое значение, «цепь размыкается», и система временно прекращает выполнение

потенциально проблемных операций, давая зависимому сервису время на восстановление. В состоянии «разомкнутой цепи» можно реализовать fallback-механизмы – возвращать кэшированные данные или значения по умолчанию. Библиотеки вроде Hystrix (от Netflix) или Resilience4j предоставляют готовые реализации этого паттерна.

Bulkhead (переборка) заимствует концепцию из кораблестроения, где переборки предотвращают затопление всего судна при повреждении одного отсека. В ИТ-системах это означает изоляцию различных частей системы друг от друга. Например, выделение отдельных пулов потоков для разных типов операций или физическое разделение ресурсов. Микросервисная архитектура сама по себе является формой bulkhead-изоляции, но этот принцип можно применять и внутри отдельных сервисов.

Использование резервных копий и реплик

Современные подходы к резервированию данных эволюционировали от простого периодического бэкапирования к сложным системам непрерывной репликации. Репликация «master-slave» позволяет распределить нагрузку чтения, в то время как multi-master репликация обеспечивает более высокую доступность. Однако последняя требует решения проблем согласованности данных (consistency).

Хранилища данных можно разделить на три категории по стратегии восстановления:

- Холодные резервы – данные хранятся отдельно и требуют времени на развертывание
- Теплые резервы – системы в режиме ожидания, готовые к частичному запуску
- Горячие резервы – полностью работоспособные системы, готовые мгновенно взять на себя нагрузку

Особое внимание стоит уделить тестированию процедур восстановления. Практика показывает, что непроверенные резервные копии часто оказываются бесполезными в критический момент. Регулярные учебные восстановления должны стать частью эксплуатационных процедур.

Аудит инцидентов и постмортем-анализ

Постмортем-анализ (postmortem analysis) – это систематическое исследование инцидента, направленное не на поиск виноватых, а на выявление коренных причин и предотвращение повторения. Хороший постмортем-документ содержит:

1. Хронологию событий с точными временными метками
2. Воздействие на бизнес (время простоя, финансовые потери)
3. Цепочку принятия решений во время инцидента
4. Коренные причины (root cause), а не только поверхностные симптомы
5. Конкретные action items для улучшения системы

Метрики надежности типа MTBF (Mean Time Between Failures) и MTTR (Mean Time To Recovery) помогают количественно оценивать прогресс в повышении отказоустойчивости. Важно отслеживать не только частоту сбоев, но и

время восстановления, а также эффективность предпринятых корректирующих действий.

Литература

1. Нестеров, С. А. Основы информационной безопасности / С. А. Нестеров. – 3-е изд., стер. – Санкт-Петербург: Лань, 2024. – 324 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/370967 (дата обращения: 09.10.2024). – ISBN 978-5-507-49077-6. – Текст : электронный (гл. 2, 3).

2. Скулябина, О. В. Системный анализ в информационной безопасности: учеб. пособие / О. В. Скулябина, С. Ю. Страхов. – Санкт-Петербург: БГТУ «Военмех» им. Д. Ф. Устинова, 2021. – 50 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/220316 (дата обращения: 11.10.2024). – Текст : электронный (с. 8–45).

3. Вейцман, В. М. Проектирование информационных систем: учеб. пособие для вузов / В. М. Вейцман. – 2-е изд., стер. – Санкт-Петербург: Лань, 2022. – 316 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/208946 (дата обращения: 09.10.2024). – ISBN 978-5-8114-9982-3. – Текст : электронный (темы 11).

Контрольные вопросы

1. Как выбрать оптимальную стратегию задержки между retry-попытками для конкретной системы?
2. В чем принципиальное отличие circuit breaker от простого ограничения количества попыток?
3. Какие сценарии демонстрируют преимущество bulkhead-изоляции перед простым масштабированием?
4. Как multi-master репликация влияет на согласованность данных в распределенных системах?
5. Почему горячие резервы не всегда являются оптимальным выбором с точки зрения затрат?
6. Какие метрики наиболее точно отражают эффективность процедур восстановления?
7. Как избежать культуры обвинений при проведении постмортем-анализа?
8. Какие элементы постмортем-документа наиболее важны для предотвращения повторных инцидентов?
9. Как количественно оценить эффективность внедренных улучшений после анализа инцидента?
10. Почему тестирование процедур восстановления должно быть регулярной практикой, а не разовым мероприятием?

Раздел 4: Особенности интеграции мер защиты в компоненты открытых систем

Тема 4.1 Шифрование данных в открытых экосистемах

Перечень изучаемых вопросов

1. Интеграция аппаратных модулей безопасности (HSM, TPM).
2. Использование стандартов PKI и цифровых сертификатов.
3. Реализация end-to-end шифрования в микросервисах.

Методические указания к изучению

В условиях роста кибератак и регуляторных требований шифрование данных стало обязательным элементом защиты информации. Однако в открытых экосистемах, где взаимодействуют разнородные системы, реализация криптографических механизмов требует особого подхода. Рассмотрим ключевые аспекты интеграции аппаратных и программных решений для обеспечения безопасности данных.

Интеграция аппаратных модулей безопасности (HSM, TPM). Аппаратные модули безопасности – специализированные устройства, защищающие криптографические ключи и операции.

HSM (Hardware Security Module). Назначение: Генерация, хранение и использование ключей в изолированной среде.

Сценарии применения:

1. Защита корневых сертификатов в PKI.
2. Шифрование транзакций в банковских системах.
3. Подпись кода в DevOps-цепочках.
4. Пример: Облачные HSM (AWS CloudHSM, Azure Dedicated HSM) позволяют управлять ключами без физического доступа.
5. TPM (Trusted Platform Module)

Отличие от HSM: Встроен в устройство, обеспечивает целостность ПО на уровне прошивки.

Функции:

1. Secure Boot – проверка подлинности ОС при загрузке.
2. Хранение ключей для BitLocker/VeraCrypt.
3. Стандарт: TPM 2.0 поддерживает алгоритмы RSA, ECC, SHA-256.

Проблемы интеграции:

1. Высокая стоимость HSM для малого бизнеса.
2. Сложность обновления прошивки TPM без простоев.
3. Использование стандартов PKI и цифровых сертификатов. PKI (Public Key Infrastructure) – инфраструктура для управления ключами и сертификатами.

Компоненты PKI:

1. Центр сертификации (CA): Выпускает и подписывает сертификаты.
2. Реестр отозванных сертификатов (CRL): Список недействительных сертификатов.

3. Хранилище ключей: Защищенная база для приватных ключей.

Цифровые сертификаты (X.509).

Структура:

1. Публичный ключ.
2. Информация о владельце (Subject).
3. ЭЦП центра сертификации.

Типы:

1. SSL/TLS для веб-серверов.
2. Client certificates для аутентификации пользователей.
3. Code Signing для подписи ПО.

Проблемы управления:

1. Жизненный цикл сертификатов: Автоматизация выдачи и обновления с помощью инструментов вроде HashiCorp Vault или Certbot.

2. Отзыв сертификатов: Использование OCSP (Online Certificate Status Protocol) для проверки статуса в реальном времени. Пример: Let’s Encrypt предоставляет бесплатные SSL-сертификаты с автоматическим обновлением каждые 90 дней.

End-to-end (E2E) шифрование гарантирует, что данные остаются зашифрованными на всём пути передачи между микросервисами.

Технологии.

TLS 1.3: для защиты канала связи.

Асимметричное шифрование: Передача сессионных ключей через алгоритмы вроде ECDH.

Ключевые хранилища: HashiCorp Vault, AWS KMS для централизованного управления.

Схема работы: сервис А запрашивает публичный ключ сервиса Б из хранилища. Данные шифруются публичным ключом Б. Сервис Б расшифровывает данные своим приватным ключом.

Проблемы:

3. Производительность: Шифрование больших объемов данных увеличивает задержки. Решение – использование аппаратного ускорения (AES-NI).

4. Сложность отладки: Затруднен анализ зашифрованных логов. Решение — разделение ключей для логирования и продакшена.

Пример архитектуры: Service Mesh (Istio, Linkerd), автоматическое управление TLS-сертификатами между микросервисами.

Сервисы шифрования: выделенные микросервисы для криптоопераций, изолированные через HSM.

Литература

1. Нестеров, С. А. Основы информационной безопасности / С. А. Нестеров. – 3-е изд., стер. – Санкт-Петербург: Лань, 2024. – 324 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/370967 (дата обращения: 09.10.2024). – ISBN 978-5-

507-49077-6. – Текст : электронный (гл. 2, 3).

2. Скулябина, О. В. Системный анализ в информационной безопасности: учеб. пособие / О. В. Скулябина, С. Ю. Страхов. –Санкт-Петербург: БГТУ «Военмех» им. Д. Ф. Устинова, 2021. – 50 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/220316 (дата обращения: 11.10.2024). – Текст : электронный (с. 8–45).

3. Вейцман, В. М. Проектирование информационных систем: учеб. пособие для вузов / В. М. Вейцман. – 2-е изд., стер. – Санкт-Петербург: Лань, 2022. – 316 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. ––URL: https://e.lanbook.com/book/208946 (дата обращения: 09.10.2024). – ISBN 978-5-8114-9982-3. – Текст : электронный (темы 1, 6, 7, 11).

Контрольные вопросы

1. Чем отличается роль HSM от TPM в защите ключей шифрования?
2. Какие риски возникают при хранении корневого сертификата CA на обычном сервере?
3. Почему Let’s Encrypt не подходит для подписи кода, а только для SSL?
4. Как OCSP предотвращает использование отозванного сертификата?
5. Какие алгоритмы шифрования предпочтительны для E2E в микросервисах и почему?
6. Как Service Mesh упрощает управление TLS в распределенных системах?
7. Почему асимметричное шифрование не используют для всего трафика между микросервисами?
8. Какие метрики стоит отслеживать при внедрении аппаратного шифрования?
9. Как организовать ротацию ключей в системе с 100+ микросервисами?
10. В чем преимущество использования ECDH перед RSA при обмене ключами?

Тема 4.2 Безопасная интеграция сторонних сервисов

Перечень изучаемых вопросов

1. Работа с API в защищенных средах (OAuth 2.0, JWT).
2. Проверка доверия к внешним поставщикам (Third-Party Risk Management).
3. Использование Service Mesh (Istio, Linkerd) для контроля трафика.

Методические указания к изучению

В современной цифровой экосистеме практически ни одно приложение не существует изолированно. Интеграция с внешними сервисами стала неотъемлемой частью разработки, но вместе с тем принесла новые вызовы в области безопасности. Утечки данных через уязвимые API, компрометация учетных данных и нарушения доступности сервисов – все это реальные угрозы, с которыми сталкиваются разработчики. В данной статье мы подробно рассмотрим три ключевых аспекта безопасной интеграции сторонних сервисов.

Работа с API в защищенных средах

Современные подходы к защите API значительно эволюционировали за последние годы. Устаревшие методы аутентификации, такие как Basic Auth, уступили место более безопасным и гибким решениям.

OAuth 2.0 стал фактическим стандартом для делегированного доступа. Его архитектура предусматривает разделение ролей между клиентом, сервером авторизации и ресурсным сервером. Это позволяет реализовать принцип минимальных привилегий – приложение получает ровно тот уровень доступа, который необходим для его работы, не требуя полномочий пользователя.

JWT (JSON Web Tokens) предлагают удобный способ передачи информации о аутентификации между сторонами. Эти токены содержат три основные части: заголовок, полезную нагрузку и подпись. Важно понимать, что JWT сами по себе не обеспечивают конфиденциальность – они лишь гарантируют целостность данных. Для защиты конфиденциальной информации следует использовать JWE (JSON Web Encryption).

Практическая реализация защищенного API требует внимания к нескольким аспектам. Во-первых, необходимо обеспечить надежное хранение секретов клиента. Во-вторых, следует тщательно настраивать сроки действия токенов - слишком длительные могут увеличить риск компрометации, а слишком короткие - создать неудобства для пользователей. В-третьих, важно реализовать механизм отзыва токенов на случай компрометации.

Проверка доверия к внешним поставщикам

Интеграция с внешними сервисами неизбежно означает передачу части контроля над безопасностью вашей системы третьей стороне. Управление этими рисками требует системного подхода.

Первым шагом в оценке поставщика должен стать анализ его репутации и истории безопасности. Существуют специализированные сервисы, такие как Security Scorecard, которые позволяют получить объективную оценку безопасности компании. Также стоит изучить публичные отчеты об инцидентах и уязвимостях.

Техническая оценка включает проверку используемых поставщиком протоколов и стандартов безопасности. Важно убедиться, что сервис поддерживает современные методы шифрования, имеет механизмы защиты от атак и регулярно проходит независимые аудиты безопасности.

Юридические аспекты не менее важны. Соглашение об уровне услуг (SLA) должно четко определять ответственность сторон в случае инцидентов.

Особое внимание следует уделить вопросам соответствия требованиям GDPR, CCPA и другим регуляторным нормам, если ваше приложение работает с персональными данными.

Непрерывный мониторинг – ключевой элемент управления рисками. Даже после тщательного отбора поставщика необходимо отслеживать изменения в его политике безопасности, обновления API и сообщения об уязвимостях. Автоматизированные системы мониторинга могут помочь оперативно выявлять подозрительную активность.

Использование Service Mesh для контроля трафика

В распределенных системах традиционные методы обеспечения безопасности часто оказываются недостаточно эффективными. Service Mesh предлагает принципиально новый подход к управлению безопасностью микросервисов.

Архитектура Service Mesh основана на концепции sidecar-прокси, которые перехватывают весь сетевой трафик между сервисами. Это позволяет реализовать единую политику безопасности без необходимости модификации самого приложения. Istio, один из самых популярных реализаций Service Mesh, использует Envoy в качестве прокси-сервера.

Взаимная аутентификация TLS (mTLS) является фундаментальной возможностью Service Mesh. В отличие от традиционного TLS, где только сервер доказывает свою подлинность клиенту, mTLS требует аутентификации обеих сторон. Это значительно снижает риск атак «человек посередине» во внутренней сети.

Контроль доступа на основе ролей (RBAC) в Service Mesh позволяет точно определять, какие сервисы могут взаимодействовать друг с другом. Политики могут быть настроены на уровне отдельных методов API, что обеспечивает детализированное управление доступом.

Наблюдаемость – еще одно важное преимущество Service Mesh. Собираемые метрики, логи и трассировки позволяют не только оперативно выявлять аномалии, но и проводить ретроспективный анализ инцидентов. Интеграция с системами мониторинга безопасности (SIEM) делает эту информацию еще более ценной.

Литература

1. Нестеров, С. А. Основы информационной безопасности / С. А. Нестеров. – 3-е изд., стер. – Санкт-Петербург: Лань, 2024. – 324 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/370967 (дата обращения: 09.10.2024). – ISBN 978-5-507-49077-6. – Текст : электронный (гл. 3, 4).

2. Скулябина, О. В. Системный анализ в информационной безопасности: учеб. пособие / О. В. Скулябина, С. Ю. Страхов. – Санкт-Петербург : БГТУ «Военмех» им. Д. Ф. Устинова, 2021. – 50 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/220316 (дата обращения: 11.10.2024). – Текст : электронный (стр. 8 - 45)

3. Вейцман, В. М. Проектирование информационных систем: учеб. пособие для вузов / В. М. Вейцман. – 2-е изд., стер. – Санкт-Петербург: Лань, 2022. – 316 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/208946 (дата обращения: 09.10.2024). – ISBN 978-5-8114-9982-3. – Текст : электронный (темы 1, 6, 7, 11).

Контрольные вопросы

1. Каковы основные преимущества OAuth 2.0 по сравнению с традиционными методами аутентификации?
2. Как архитектура JWT способствует масштабируемости распределенных систем?
3. Какие факторы следует учитывать при определении оптимального времени жизни токенов доступа?
4. Какие критерии наиболее важны при выборе внешнего API-провайдера с точки зрения безопасности?
5. Как SLA может помочь в управлении рисками при интеграции с внешними сервисами?
6. Какие технические показатели наиболее точно отражают уровень безопасности стороннего сервиса?
7. В чем заключаются основные архитектурные различия между Istio и Linkerd?
8. Как mTLS в Service Mesh обеспечивает защиту внутреннего трафика?
9. Какие возможности Service Mesh наиболее полезны для соответствия регуляторным требованиям?
10. Как можно интегрировать мониторинг безопасности Service Mesh с существующей SIEM-системой?

Тема 4.3 Аудит и мониторинг защищенных систем

Перечень изучаемых вопросов

1. Настройка SIEM-систем (Splunk, Elastic Security).
2. Реализация аудита изменений (Audit Logging).
3. Автоматизация проверок на соответствие стандартам (ISO 27001, GDPR).

Методические указания к изучению

В современной цифровой среде безопасность информационных систем требует не только профилактических мер защиты, но и постоянного контроля их эффективности. Аудит и мониторинг становятся критически важными компонентами комплексной стратегии безопасности, позволяя выявлять угрозы в режиме реального времени и оперативно реагировать на инциденты. В данной статье мы рассмотрим три ключевых аспекта построения эффективной системы аудита и мониторинга.

SIEM (Security Information and Event Management) системы представляют собой мощный инструмент для централизованного сбора и анализа данных безопасности. Современные решения, такие как Splunk и Elastic Security, предлагают широкие возможности для мониторинга защищенных систем.

При настройке SIEM-системы первостепенное значение имеет определение релевантных источников данных. Важно включить в мониторинг не только традиционные сетевые устройства и серверы, но и облачные сервисы, контейнерные платформы и IoT-устройства. Каждый источник должен быть настроен с учетом его специфики и важности для общей инфраструктуры.

Нормализация данных – ключевой этап подготовки информации к анализу. Различные устройства и приложения генерируют события в разных форматах, и задача SIEM – привести их к единому виду. Это позволяет создавать универсальные правила корреляции, не зависящие от конкретного производителя оборудования.

Создание правил корреляции требует глубокого понимания как технических аспектов, так и бизнес-процессов организации. Эффективные правила должны балансировать между чувствительностью (способностью обнаруживать реальные угрозы) и уровнем ложных срабатываний. Машинное обучение все чаще применяется для улучшения этого баланса.

Визуализация данных в SIEM-системах играет важную роль в оперативном принятии решений. Современные платформы предлагают customizable dashboards, которые позволяют отображать наиболее критичные метрики безопасности в удобном для анализа виде.

Система аудита изменений (Audit Logging) является фундаментальным компонентом защищенной инфраструктуры. Она позволяет не только выявлять потенциальные угрозы, но и восстанавливать ход событий после инцидента.

При проектировании системы аудита необходимо учитывать принцип минимальной достаточности. Чрезмерное логирование может привести как к проблемам с производительностью, так и к сложностям при анализе данных. Важно определить, какие именно события представляют ценность с точки зрения безопасности и соответствия требованиям.

Целостность логов – критически важное свойство системы аудита. Технические решения должны гарантировать, что записи не могут быть изменены или удалены без соответствующего разрешения. Использование цифровых подписей и хэширования помогает обеспечить эту целостность.

Хранение логов требует особого внимания. Политики хранения должны учитывать, как требования регуляторов, так и операционные потребности организации. Все чаще используются гибридные подходы, сочетающие локальное хранение «горячих» данных с архивированием в облаке.

Анализ логов изменений должен быть как автоматизированным (для оперативного выявления аномалий), так и ручным (для расследования сложных инцидентов). Интеграция системы аудита с SIEM позволяет существенно повысить эффективность мониторинга.

Эффективная система аудита и мониторинга является неотъемлемой частью современной инфраструктуры безопасности. Комбинация SIEM-систем, надежного аудита изменений и автоматизированных проверок соответствия позволяет организациям не только оперативно реагировать на угрозы, но и демонстрировать соблюдение регуляторных требований. Важно понимать, что построение такой системы - это непрерывный процесс, требующий регулярного обновления и адаптации к изменяющимся условиям и новым видам угроз. Инвестиции в качественный аудит и мониторинг окупаются за счет снижения рисков и повышения общей устойчивости информационных систем.

Литература

1. Нестеров, С. А. Основы информационной безопасности / С. А. Нестеров. – 3-е изд., стер. – Санкт-Петербург: Лань, 2024. – 324 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/370967 (дата обращения: 09.10.2024). – ISBN 978-5-507-49077-6. – Текст : электронный (гл. 2, 3).

2. Скулябина, О. В. Системный анализ в информационной безопасности: учеб. пособие / О. В. Скулябина, С. Ю. Страхов. — Санкт-Петербург : БГТУ «Военмех» им. Д. Ф. Устинова, 2021. – 50 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/220316 (дата обращения: 11.10.2024). – Текст : электронный (с. 8–45).

3. Вейцман, В. М. Проектирование информационных систем: учеб. пособие для вузов / В. М. Вейцман. – 2-е изд., стер. – Санкт-Петербург: Лань, 2022. – 316 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/208946 (дата обращения: 09.10.2024). – ISBN 978-5-8114-9982-3. – Текст : электронный (темы 1, 6, 7, 11).

Контрольные вопросы

1. Какие критерии следует учитывать при выборе источников данных для SIEM-системы?
2. Как нормализация данных в SIEM влияет на эффективность обнаружения угроз?
3. Какие методы обеспечения целостности логов изменений наиболее надежны?
4. Как определить оптимальный срок хранения журналов аудита для конкретной организации?
5. В чем преимущества интеграции систем аудита изменений с SIEM?
6. Каким образом подход «Инфраструктура как код» способствует обеспечению соответствия стандартам?
7. Как Continuous Compliance интегрируется в процессы DevOps?
8. Какие показатели следует включать в автоматически генерируемые отчеты о соответствии?

9. Как машинное обучение улучшает качество корреляции событий в SIEM?

10. Какие особенности необходимо учитывать при настройке аудита для облачных сред?

Тема 4.4 Адаптация к новым угрозам и обновление защиты

Перечень изучаемых вопросов

1. Внедрение Zero Trust Architecture.
2. Использование машинного обучения для обнаружения аномалий.
3. Планирование жизненного цикла безопасности (Security Lifecycle Management).

Методические указания к изучению

Современный ландшафт киберугроз напоминает гонку вооружений: злоумышленники постоянно совершенствуют методы атак, а защитникам приходится адаптироваться в режиме реального времени. Традиционные подходы, основанные на периметровой безопасности и статичных правилах, утрачивают эффективность. В этой статье разберем три ключевых стратегии, позволяющих организациям не просто реагировать на угрозы, но и опережать их.

1. Внедрение Zero Trust Architecture

Концепция Zero Trust («Никому не доверяй») родилась как ответ на крах модели «замкнутого периметра». Её суть – в отказе от автоматического доверия к любым субъектам, включая внутренние системы и пользователей.

Основные принципы Zero Trust:

1. Верификация каждого запроса: доступ предоставляется только после многофакторной аутентификации и проверки контекста (устройство, местоположение, время).
2. Минимальные привилегии: права выдаются ровно на тот объем действий, который необходим для задачи.
3. Микросегментация сети: разбиение инфраструктуры на изолированные зоны с индивидуальными политиками доступа.

Практическая реализация:

- Использование SDP (Software-Defined Perimeter) для создания динамических периметров.
- Интеграция с IAM-системами (Okta, Azure AD) для управления идентификацией.
- Внедрение аналитики поведения (UEBA) для выявления отклонений в действиях пользователей.

Пример: компания Google внедрила Zero Trust в рамках проекта BeyondCorp, отказавшись от VPN. Доступ к корпоративным ресурсам теперь зависит не от местоположения сотрудника, а от его роли и уровня доверия.

2. Использование машинного обучения для обнаружения аномалий

Машинное обучение (ML) стало ключевым инструментом борьбы с неизвестными угрозами. В отличие от сигнатурных методов, ML анализирует паттерны поведения, выявляя отклонения, которые не соответствуют «норме».

Применение ML в безопасности:

- Обнаружение аномального сетевого трафика: алгоритмы изолируют подозрительные соединения, даже если они используют шифрование.
- Анализ поведения пользователей и сущностей (UEBA): выявление компрометированных учетных записей через аномальную активность.
- Прогнозирование уязвимостей: предсказание рисков на основе данных о прошлых инцидентах и текущей конфигурации систем.

Примеры технологий:

- Darktrace использует нейронные сети для построения «цифрового иммунитета».
- AWS GuardDuty анализирует логи облачных сервисов, обнаруживая подозрительные API-вызовы.

Проблемы внедрения:

- Ложные срабатывания: необходимость тонкой настройки моделей под специфику организации.
- Этический аспект: риски чрезмерного мониторинга сотрудников.

Планирование жизненного цикла безопасности

Безопасность – не разовое мероприятие, а непрерывный процесс. Security Lifecycle Management предполагает системный подход к обновлению защиты на всех этапах существования системы.

Этапы жизненного цикла:

1. Проектирование: внедрение безопасности «по умолчанию» (Security by Design).

2. Реализация: автоматизация проверок кода (SAST, DAST) и зависимостей (SCA).

3. Эксплуатация: регулярные пентесты и red teaming.

4. Утилизация: безопасное удаление данных и вывод систем из эксплуатации.

Инструменты и методологии:

- DevSecOps: интеграция безопасности в CI/CD-конвейеры. Например, использование GitLab Secret Detection для поиска утечек ключей в коде.
- Threat Intelligence: платформы вроде MISP для анализа актуальных угроз.
- Фреймворки управления рисками: NIST Cybersecurity Framework, ISO 27001.

Кейс: Компания Microsoft применяет подход Secure Future Initiative, где каждое обновление продукта сопровождается переоценкой архитектурных рисков.

Адаптация к новым угрозам требует перехода от реактивной к проактивной безопасности. Zero Trust, машинное обучение и управление жизненным циклом – не отдельные технологии, а части единой стратегии.

Важно помнить, что ни одна система не может быть полностью защищена: цель – создать среду, где стоимость взлома превышает потенциальную выгоду для злоумышленника. Ключ к успеху – в сочетании технологий, процессов и культуры безопасности, где каждый сотрудник становится «человеческим фаерволом»

Литература

1. Нестеров, С. А. Основы информационной безопасности / С. А. Нестеров. – 3-е изд., стер. – Санкт-Петербург: Лань, 2024. – 324 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/370967 (дата обращения: 09.10.2024). – ISBN 978-5-507-49077-6. – Текст : электронный (гл. 2, 3).

2. Скулябина, О. В. Системный анализ в информационной безопасности: учеб. пособие / О. В. Скулябина, С. Ю. Страхов. – Санкт-Петербург: БГТУ «Военмех» им. Д. Ф. Устинова, 2021. – 50 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/220316 (дата обращения: 11.10.2024). – Текст : электронный (с. 8–45).

Вейцман, В. М. Проектирование информационных систем: учеб. пособие для вузов / В. М. Вейцман. – 2-е изд., стер. – Санкт-Петербург: Лань, 2022. – 316 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/208946 (дата обращения: 09.10.2024). – ISBN 978-5-8114-9982-3. – Текст : электронный (темы 1, 6, 7, 11).

Контрольные вопросы

1. Почему модель Zero Trust считается более устойчивой к внутренним угрозам?
2. Какие данные необходимы для обучения ML-моделей в системах обнаружения аномалий?
3. Как микросегментация сети снижает риски lateral movement?
4. В чем отличие UEBA от традиционных SIEM-систем?
5. Какие этапы жизненного цикла чаще всего игнорируются при внедрении безопасности?
6. Как DevSecOps влияет на скорость реагирования на уязвимости?
7. Какие этические дилеммы возникают при использовании ML для мониторинга сотрудников?
8. Почему Threat Intelligence важен для планирования жизненного цикла безопасности?
9. Как SDP защищает от атак на устаревшие VPN-решения?
10. Какие метрики используют для оценки эффективности Security Lifecycle Management?

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» (таблица 1).

4. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО 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 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. ТРЕБОВАНИЯ К АТТЕСТАЦИИ ПО ДИСЦИПЛИНЕ

Текущая аттестация

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

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 3).

Таблица 3 – Система оценок и критерии выставления оценки

Система оценок / **Критерий**				
	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	Может найти необходимую информацию в рамках поставленной	Может найти, интерпретировать и систематизировать необходимую	Может найти, систематизировать необходимую информацию, а также выявить новые,

Система оценок / Критерий				
	0–40 %	**41–60 %**	**61–80 %**	**81–100 %**
	«неудовлетворительно»	**«удовлетворительно»**	**«хорошо»**	**«отлично»**
	«не зачтено»	**«зачтено»**		
	фрагменты информации в рамках поставленной задачи	задачи	информацию в рамках поставленной задачи	дополнительные источники информации в рамках поставленной задачи
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

Примерные вопросы к зачету/экзамену по дисциплине
Вопросы к зачету

1. Назовите три ключевые характеристики открытых систем и приведите примеры.
2. Объясните, почему совместимость (interoperability) важна для открытых систем.
3. Как принцип «безопасности по умолчанию» влияет на проектирование распределенных систем?
4. В чем разница между симметричным и асимметричным шифрованием? Где применяется AES?
5. Почему SHA-3 считается более безопасным, чем SHA-2?
6. Какие ошибки разработчиков снижают эффективность использования библиотек криптографии?
7. Опишите модель RBAC и ее преимущества.
8. Чем отличается OAuth 2.0 от OpenID Connect?
9. Как принцип минимальных привилегий предотвращает SQL-инъекции?
10. Какие методы защиты от XSS вы знаете?
11. Зачем API-шлюзы используют rate limiting?
12. Объясните разницу между валидацией и санитизацией входных данных.
13. Сформулируйте CAP-теорему. Какие два свойства гарантирует CP-система?
14. Почему Cassandra относится к AP-системам?
15. Как Field-Level Encryption защищает данные в MongoDB?
16. Для чего используется Change Data Capture (CDC)?
17. В чем недостатки подхода Last Write Wins (LWW)?
18. Как TLS защищает каналы синхронизации данных?
19. Какие функции выполняет сервисная шина (Service Bus)?
20. Как паттерн CQRS улучшает безопасность данных?
21. Для чего используется Distributed Tracing (Jaeger/Zipkin)?
22. Как Wireshark помогает анализировать сетевые атаки?
23. Назовите три инструмента для статического анализа кода.
24. В чем разница между фаззингом и пентестингом?

Вопросы к экзамену

1. Объясните, как открытость архитектуры влияет на уязвимость системы. Приведите пример.
2. Какие угрозы характерны для распределенных сред? Как DDoS-атака использует их особенности?
3. Как микросервисная архитектура реализует принцип Defense in Depth?

4. Почему режим ECB в AES считается небезопасным? Какой режим следует использовать вместо него?

5. Опишите сценарий совместного использования TLS и AES для защиты данных.

6. Какие алгоритмы хеширования нельзя использовать для паролей и почему?

7. Сравните RBAC и ABAC. В каких случаях предпочтителен RBAC?

8. Как JWT-токены используются для аутентификации в OpenID Connect?

9. Какие риски возникают при нарушении принципа минимальных привилегий?

10. Как CSRF-токены предотвращают подделку запросов? Опишите механизм.

11. Почему проверки на стороне клиента недостаточны для защиты от вредоносного ввода?

12. Приведите пример уязвимости API, ведущей к компрометации БД.

13. Почему CockroachDB подходит для финансовых систем, а Cassandra – нет?

14. Как алгоритм Raft в CockroachDB обеспечивает согласованность данных?

15. Какие проблемы безопасности возникают при выборе AP-модели?

16. Чем клиентское шифрование отличается от прозрачного (TDE)?

17. Как аудит изменений помогает в расследовании инцидентов?

18. Почему в распределенных системах сложно обеспечить изоляцию транзакций?

19. Как CRDT решает конфликты данных без явного разрешения?

20. Почему в Multi-Master чаще возникают конфликты, чем в Master-Slave?

21. Какие риски возникают при использовании VPN для синхронизации данных?

22. Как CQRS разделяет права доступа между командами и запросами?

23. Зачем маскировать данные перед миграцией в тестовую среду?

24. Опишите MITM-атаку через незащищенную сервисную шину.

25. Как VisualVM помогает анализировать многопоточные приложения?

26. Какие метрики в Prometheus указывают на аномалии безопасности?

27. Почему комбинация SAST и DAST эффективнее каждого метода по отдельности?

28. Как Chaos Engineering тестирует устойчивость систем к атакам?

29. Как избежать race condition в распределенных транзакциях?

30. Какие edge-кейсы важно тестировать в микросервисной архитектуре?

31. Объясните принцип работы Circuit Breaker. Приведите пример использования.

32. Почему горячие резервы не всегда экономически выгодны?

33. В чем разница между HSM и TPM?

34. Как PKI обеспечивает доверие в цифровых сертификатах?

35. Почему E2E-шифрование сложно реализовать в микросервисах?

36. Какие риски возникают при использовании устаревших версий OAuth 2.0?

37. Как Service Mesh (Istio) реализует mTLS для внутреннего трафика?

38. Какие данные нормализуют в SIEM-системах и зачем?

39. Как автоматизировать проверки на соответствие GDPR?

40. Какие этапы включает Security Lifecycle Management?

41. Как Zero Trust Architecture предотвращает lateral movement?

42. Приведите пример использования ML для обнаружения аномалий в сетевом трафике.

43. Проанализируйте, как принципы Secure by Design реализуются в Kubernetes.

44. Сравните подходы к безопасности в реляционных и NoSQL БД.

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. Нестеров, С. А. Основы информационной безопасности / С. А. Нестеров. – 3-е изд., стер. – Санкт-Петербург: Лань, 2024. – 324 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/370967 (дата обращения: 09.10.2024). – ISBN 978-5-507-49077-6. – Текст : электронный.

2. Вейцман, В. М. Проектирование информационных систем: учеб. пособие для вузов / В. М. Вейцман. – 2-е изд., стер. Санкт-Петербург: Лань, 2022. – 316 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/208946 (дата обраще-

ния: 09.10.2024). – ISBN 978-5-8114-9982-3. – Текст : электронный.

3. Скулябина, О. В. Системный анализ в информационной безопасности: учеб. пособие / О. В. Скулябина, С. Ю. Страхов. – Санкт-Петербург: БГТУ «Военмех» им. Д. Ф. Устинова, 2021. – 50 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/220316 (дата обращения: 11.10.2024). – Текст : электронный.

Дополнительные источники

4. Мандрица, И. В. Управление проектами по информационной безопасности и экономика защиты информации. Ч. 1 / И. В. Мандрица, В. И. Петренко, О. В. Мандрица. – Санкт-Петербург: Лань, 2023. – 124 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/311825 (дата обращения: 10.10.2024). – ISBN 978-5-507-45723-6. – Текст : электронный.

5. Рейн, Т. С. Основы информационной безопасности: учеб. пособие / Т. С. Рейн, В. В. Торгулькин. – Кемерово КемГУ, 2024. – 117 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/427526 (дата обращения: 09.10.2024). – ISBN 978-5-8353-3270-0. – Текст : электронный.

6. Остроух, А. В. Проектирование информационных систем: монография / А. В. Остроух, Н. Е. Суркова. – 2-е изд., стер. – Санкт-Петербург: Лань, 2021. – 164 с. – Режим доступа: для авториз. пользователей. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/175513 (дата обращения: 06.12.2024). – ISBN 978-5-8114-8377-8. – Текст : электронный.

Учебно-методические пособия, нормативная литература

7. Бабаева, А. А. Проектирование открытых систем в защищенном исполнении: учебно-методическое пособие по изучению дисциплины для студентов специальности 10.05.03 «Проектирование открытых систем в защищенном исполнении» / А. А. Бабаева. – Калининград: Изд-во ФГБОУ ВО «КГТУ», 2022. – 30 с. - URL: https://www.klgtu.ru/vikon/sveden/files/zif/UMP_Proektirovanie_otkrytyx_sistem_v_zaschischennom_ispolnenii.pdf (дата обращения: 08.12.2024). — Текст : электронный.

8. Бабаева, А. А. Проектирование открытых систем в защищенном исполнении: учебно-методическое пособие по выполнению лабораторных работ для студентов специальности 10.05.03 Информационная безопасность автоматизированных систем / А. А. Бабаева. – Калининград: Изд-во ФГБОУ ВО «КГТУ», 2022. – 38 с. – URL:

https://www.klgtu.ru/vikon/sveden/files/ziz/UMP_Proektirovanie_otkrytyx_sistem_v_zaschischennom_ispolnenii_(laboratornye_raboty).pdf (дата обращения: 08.12.2024). – Текст : электронный.

9. Данилина, И. И. Программирование на языке C# в среде Microsoft Visual Studio: учебно-методическое пособие / И. И. Данилина. – Екатеринбург: 2018. – 65 с. – Режим доступа: для авториз. пользователей. – Лань: электронно-библиотечная система. – URL: https://e.lanbook.com/book/121392 (дата обращения: 09.10.2024). – Текст : электронный.

10. «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.

11. «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.

12. 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.

13. 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.

14. 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.

15. 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.

16. 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.

17. «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.

18. «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.

19. «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.

20. «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

ПРОГРАММИРОВАНИЕ КОМПОНЕНТОВ ОТКРЫТЫХ СИСТЕМ В ЗАЩИЩЁННОМ ИСПОЛНЕНИИ

Редактор С. Кондрашова
Корректор Т. Звада

Уч.-изд. л. 5,2. Печ. л. 4,1.

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