

Документ подписан простой электронной подписью

Информация о владельце:

ФИО: Узунов Федор Владимирович

Должность: Ректор

Дата подписания: 30.09.2026 08:54:43

Уникальный программный ключ:

fd935d10451b860e912264c0378b448452bfdb607f94388008e39877e6bcb55

**АВТОНОМНАЯ НЕКОММЕРЧЕСКАЯ ОРГАНИЗАЦИЯ
«ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ ВЫСШЕГО ОБРАЗОВАНИЯ»**

«УНИВЕРСИТЕТ ЭКОНОМИКИ И УПРАВЛЕНИЯ»

Факультет экономики, управления и юриспруденции

Кафедра управления и бизнес-информатики



УТВЕРЖДАЮ

Проректор по учебно-методической работе

_____/ Г.П. Узунова

«02» февраля 2026 г.

**ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ПО ДИСЦИПЛИНЕ
ПРАВОВОЕ ОБЕСПЕЧЕНИЕ ЗАЩИТЫ ИНФОРМАЦИИ**

Направление подготовки
10.03.01 «Информационная безопасность»
Профиль
Безопасность компьютерных систем

Квалификация выпускника: бакалавр

Для всех
форм обучения

Симферополь, 2026 г.

1. ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ

Средства оценивания в ходе текущего контроля:

- устные опросы в ходе семинарских занятий;
- рефераты;
- тестирование;
- практические задания, выполняемые в ходе семинарского (практического) занятия или рекомендуемые для самостоятельной работы.

Коды компетенции	Результаты освоения ОПОП	Перечень планируемых результатов обучения по дисциплине
ПК-5	Способен принимать участие в формировании, организовывать и поддерживать выполнение комплекса мер по обеспечению информационной безопасности, управлять процессом их реализации	ПК-5.1. Знает принципы формирования комплекса мер по обеспечению информационной безопасности предприятия (организации); ПК-5.2. владеет методами организации и управления деятельностью служб защиты информации на предприятии

1.1 Вопросы к текущему контролю

1. Понятие «Разработка защищенного ПО» (РБПО). Основные отличия от традиционного тестирования безопасности после завершения разработки.
2. Жизненный цикл безопасной разработки (Secure SDLC / SSDLC). Описание этапов и интеграция практик ИБ в каждый из них.
3. Экономическое обоснование концепции *Shift Left*. Сравнение стоимости исправления уязвимостей на этапах требований, кодирования и эксплуатации.
4. Стандарт ГОСТ Р 56939-2016: область применения, основные требования к разработке ГИС и ИСПДн.
5. Модели зрелости процессов разработки (*Maturity Models*): сравнительный анализ Microsoft SAMM и BSIMM.
6. Концепция «Security by Design» и «Privacy by Design». Примеры реализации принципов при проектировании архитектуры БД.
7. Управление техническим долгом безопасности (*Security Technical Debt*).
8. Роль комплаенса и регуляторов (ФСТЭК, ФСБ) в процессе создания средств защиты информации (СЗИ) в РФ.
9. Программно-аппаратные среды исполнения (TEE, Secure Enclave): их роль в защите мобильных приложений и транзакций.
10. Лицензионная чистота и управление юридическими рисками при использовании Open Source компонентов.
11. Методология STRIDE: расшифровка аббревиатуры и примеры угроз для веб-сервиса авторизации.
12. Построение деревьев атак (*Attack Trees*): алгоритм составления и расчет вероятности реализации атаки.
13. Модель LINDDUN для анализа приватностных угроз применительно к системам обработки биометрических данных.
14. Метод PASTA (Process for Attack Simulation and Threat Analysis): этапы бизнес-анализа и технического моделирования.
15. Проектирование микросервисной архитектуры с использованием принципа наименьших привилегий (*POLP*) и эшелонированной обороны (*Defense in Depth*).
16. Угрозы API: специфика моделирования рисков для REST и GraphQL интерфейсов.

17. Безопасность облачных архитектур: модель разделенной ответственности (*Shared Responsibility Model*) у провайдеров IaaS/PaaS.
18. Использование визуальных языков моделирования (UML/SysML) для описания протоколов безопасного обмена данными.
19. Анализ поверхности атаки (*Attack Surface Analysis*): методика минимизации доступных входов в систему.
20. Дизайн-ревью безопасности (*Security Design Review*): критерии оценки качества архитектуры перед началом спринта разработки.
21. Уязвимость A01:2021 (Broken Access Control). Разбор механизмов IDOR и вертикального повышения привилегий.
22. Защита от инъекций (A03:2021). Почему параметризованные запросы (*Prepared Statements*) являются единственным надежным методом против SQLi?
23. Межсайтовый скриптинг (XSS, A03:2021). Различия Stored, Reflected и DOM-based XSS. Механизмы нейтрализации.
24. Небезопасная десериализация объектов (A08:2021). Принцип работы цепочек гаджетов (*Gadget Chains*) на примере Java или PHP.
25. Использование компонентов с известными уязвимостями (A06:2021). Кейс Log4Shell (Log4j): причины катастрофического масштаба.
26. Нарушение целостности криптографии (A02:2021). Типичные ошибки использования собственных алгоритмов шифрования и генерации случайных чисел.
27. Ошибки идентификации и аутентификации (A07:2021). Защита от Brute Force, Credential Stuffing и слабых политик паролей.
28. Целочисленные переполнения (*Integer Overflow/Underflow*) в языках низкого уровня (C/C++). Методы предотвращения через SafeInt и проверку границ.
29. Server-Side Request Forgery (SSRF). Механизм атаки и методы фильтрации запросов на уровне приложения и сети.
30. Логические уязвимости бизнес-логики (Business Logic Flaws). Примеры обхода корзины покупок или манипуляции курсами валют.
31. Хранение секретов (паролей, токенов API, ключей). Критика хранения в .env файлах и переменных окружения CI/CD.
32. Работа с хранилищами секретов: HashiCorp Vault, AWS Secrets Manager, Azure Key Vault. Принципы динамической выдачи учетных записей.
33. Утечка секретов через публичные репозитории Git. Инструменты автоматического поиска (Gitleaks, TruffleHog) и предкоммит-хуки.
34. Hardening серверной инфраструктуры и контейнеров. CIS Benchmarks для Linux-серверов и Docker.
35. Заголовки безопасности HTTP (*Hardening Headers*): Content-Security-Policy, Strict-Transport-Security, X-Frame-Options.
36. Управление конфигурацией инфраструктуры как кода (*IaC Security*). Поиск уязвимостей в Terraform и CloudFormation шаблонах.
37. Безопасная работа с файловой системой: предотвращение Path Traversal и Race Conditions (TOCTOU).
38. Настройка кросс-доменных политик: сравнение JSONP, CORS и механизма preflight-запросов.
39. Обезличивание и маскирование данных (*Data Masking*) в средах разработки и тестирования.
40. Аудит конфигурации облака (CSPM - *Cloud Security Posture Management*).
41. Статический анализ (SAST): принцип работы, ложноположительные (*False Positives*) и ложноотрицательные (*False Negatives*) срабатывания.
42. Динамический анализ (DAST/IAST): имитация атак на работающее приложение. Преимущества интерактивного анализа (IAST) над классическим DAST.

43. Анализ состава ПО (SCA - *Software Composition Analysis*). Автоматизация отслеживания уязвимостей в зависимостях (SBOM).
44. Фаззинг-тестирование (*Fuzzing*): мутационный vs генерационный фаззинг. Применение инструментов AFL++ и libFuzzer.
45. Символьное выполнение (*Symbolic Execution*) и формальные методы верификации критических участков кода.
46. Тестирование на проникновение (*Penetration Testing*): различие между Black Box, Gray Box и White Box подходами.
47. Анализ покрытия кода тестами (*Code Coverage*) с точки зрения безопасности. Достаточно ли 100% покрытия для гарантии отсутствия уязвимостей?
48. Reverse Engineering бинарных файлов: инструменты Ghidra/IDA Pro для поиска закладок и недекларированных возможностей (НДВ).
49. Контрольные суммы и цифровые подписи исполняемых файлов. Проверка целостности дистрибутива перед установкой.
50. Red Teaming: оценка готовности всей организации (людей, процессов и техники) к целенаправленной атаке.
51. Интеграция сканеров безопасности в пайплайны CI/CD (*DevSecOps*). Стратегии Fail-Fast и Break-Build.
52. Безопасность контейнеризации (Docker/Kubernetes). Специфика оркестрации: Network Policies, Pod Security Standards, Secrets management в K8s.
53. Runtime Application Self-Protection (RASP) и Web Application Firewall (WAF): место в архитектуре и ограничения эффективности.
54. Мониторинг безопасности (*Application Logging*). Что логировать, а что категорически нельзя (GDPR, персональные данные).
55. EDR/XDR системы для защиты конечных точек разработчиков и сборочных агентов.
56. Планирование непрерывности бизнеса (BCP) и восстановления после сбоев (DRP) для ИТ-инфраструктуры разработки.
57. Процесс управления уязвимостями (*Vulnerability Management Lifecycle*): приоритизация по CVSS v3.1 и SLA на устранение.
58. Реагирование на инциденты ИБ (*Incident Response*). Этапы подготовки, обнаружения, сдерживания и извлечения уроков (*Lessons Learned*).
59. Этика исследователя безопасности. Правовые аспекты Bug Bounty программ и политики ответственного разглашения (VDP).
60. Безопасный вывод ПО из эксплуатации (*Sunsetting/Data Disposal*). Методы гарантированного удаления данных с носителей.

1.2 Темы рефератов:

1. Концепция «Shift Left» в разработке ПО: экономический анализ стоимости исправления уязвимостей на разных этапах жизненного цикла (SDLC).
2. Сравнительный анализ maturity-моделей: Microsoft SAMM (Software Assurance Maturity Model) против BSIMM (Building Security In Maturity Model).
3. Реализация требований ГОСТ Р 56939-2016 в цикле разработки защищенного программного обеспечения для государственных информационных систем.
4. DevSecOps: мифы и реальность. Проблемы автоматизации сканирований безопасности (SAST/DAST) без замедления релизного цикла.
5. Методология Privacy by Design: интеграция принципов защиты персональных данных (согласно GDPR и ФЗ-152) на этапе архитектуры.
6. Применение методологии STRIDE для моделирования угроз микросервисной архитектуры (на примере системы онлайн-банкинга).
7. Анализ приватностных угроз с помощью модели LINDDUN применительно к мобильным медицинским приложениям.

8. Построение сценариев атак (Attack Trees): от выбора цели до реализации вектора проникновения.
9. Архитектурные паттерны безопасности: сравнительный анализ концепции Zero Trust Network Access (ZTNA) и классического периметрального подхода.
10. Безопасность API: проектирование REST и GraphQL интерфейсов с защитой от BOLA (IDOR) и массового перечисления (Mass Assignment).
11. Уязвимость A01:2021 (Broken Access Control): глубокий анализ механизмов горизонтального и вертикального повышения привилегий.
12. Критический разбор защиты от SQL-инъекций: почему параметризованные запросы (*Prepared Statements*) — единственный надежный метод, а WAF могут быть обойдены.
13. Межсайтовый скриптинг (XSS): эволюция методов защиты от контекстного экранирования до Content Security Policy (CSP).
14. небезопасная десериализация объектов (Java Deserialization Gadget Chains): механизмы атаки и стратегии безопасной замены форматов (JSON vs Binary).
15. Целочисленные переполнения и ошибки управления памятью в языках C/C++: использование AddressSanitizer (ASan) и библиотек безопасного типа (SafeInt).
16. Атаки на цепочки поставок (Supply Chain Attacks): кейсы SolarWinds, event-stream (npm) и Log4j. Уроки для команд разработки.
17. SBOM (Software Bill of Materials) как инструмент прозрачности: форматы SPDX и CycloneDX, автоматическая генерация и проверка состава ПО.
18. SCA (Software Composition Analysis): сравнительный анализ инструментов поиска уязвимостей в open-source компонентах (Snyk, OWASP Dependency-Check, GitHub Dependabot).
19. Риски использования непроверенных Docker-образов: методы «очистки» (hardening) контейнеров и создание минималистичных базовых образов (Distroless).
20. Воспроизводимые сборки (Reproducible Builds): гарантия того, что бинарный файл соответствует исходному коду и не содержит внедренных закладок.
21. Статический анализ (SAST): проблема ложноположительных срабатываний (*False Positives*). Методы настройки правил (Rule Tuning) для снижения шума.
22. Фаззинг-тестирование (*Fuzzing*): мутационный фаззинг против генерационного (на основе грамматик). Применение AFL++ и libFuzzer.
23. Интерактивный анализ приложений (IAST): преимущества перед классическим DAST (динамическим анализом) при поиске сложных цепочек уязвимостей.
24. Методы безопасного деплоя: Blue-Green развертывание и Canary Releases как инструменты минимизации рисков при обновлении production-среды.
25. Безопасность API: специфика тестирования GraphQL-интерфейсов (Batching attacks, Alias Overloading) в сравнении с REST.
26. Управление секретами (Secrets Management): отказ от хранения паролей и токенов в коде (Git), использование HashiCorp Vault или облачных менеджеров секретов.
27. Разработка плана реагирования на инциденты ИБ (Incident Response Plan) для команды разработчиков программного обеспечения.
28. Формальные методы верификации кода: использование языка F* или фреймворка seL4 для доказательства отсутствия критических ошибок в ядре ОС.
29. Этика исследователя безопасности (White Hat): юридические аспекты проведения тестов на проникновение и политика ответственного разглашения (*VDP/Bug Bounty*).
30. Runtime Application Self-Protection (RASP): принципы работы встраиваемых средств защиты, анализирующих поведение приложения во время выполнения.

1.3 Тестовые задания

1. В чем заключается основной экономический смысл концепции «Shift Left» (Сдвиг влево) при разработке ПО? а) В переносе тестирования интерфейса на самый

ранний этап. б) В снижении стоимости исправления уязвимости за счет ее обнаружения на этапе проектирования, а не после релиза. в) В использовании только Open Source инструментов для экономии бюджета. г) В ускорении написания кода за счет отказа от документации. *Правильный ответ: б*

2. **Как расшифровывается аббревиатура STRIDE в методологии моделирования угроз Microsoft?** а) Security, Testing, Review, Implementation, Deployment, Encryption. б) Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege. в) Storage, Transfer, Retrieval, Indexing, Deletion, Erasure. г) SQL, TCP/IP, REST, IDE, DNS, Email. *Правильный ответ: б*

3. **Уязвимость IDOR (Insecure Direct Object Reference), позволяющая одному пользователю получить доступ к данным другого путем подмены user_id=12 на user_id=13, относится к категории OWASP Top 10...** а) A01:2021 — Нарушение контроля доступа (Broken Access Control). б) A03:2021 — Инъекции (Injection). в) A06:2021 — Устаревшие компоненты (Vulnerable and Outdated Components). г) A05:2021 — Ошибки конфигурации (Security Misconfiguration). *Правильный ответ: а*

4. **Почему использование параметризованных запросов (Prepared Statements) считается единственным надежным способом защиты от SQL-инъекций?** а) Потому что они автоматически шифруют данные перед отправкой в БД. б) Потому что они строго разделяют исполняемый код запроса и пользовательские данные, делая экранирование ненужным. в) Потому что они работают быстрее обычных строковых запросов. г) Потому что они блокируют IP-адрес злоумышленника после первой попытки атаки. *Правильный ответ: б*

5. **Что является главной целью анализа состава программного обеспечения (SCA — Software Composition Analysis)?** а) Поиск уязвимостей (CVE) и лицензионных рисков в сторонних библиотеках и open-source компонентах. б) Проверка качества собственного написанного кода разработчиков. в) Измерение скорости загрузки страниц веб-приложения. г) Подсчет количества строк кода в проекте. *Правильный ответ: а*

6. **Какой тип анализа безопасности выполняется путем непосредственного выполнения программы и отправки ей специально сформированных вредоносных входных данных («мусора»)?** а) Статический анализ (SAST). б) Динамический анализ (DAST) или фаззинг-тестирование. в) Анализ исходного кода линтером. г) Ручной аудит архитектуры. *Правильный ответ: б*

7. **Атака Log4Shell (эксплуатация уязвимости в библиотеке Log4j) стала классическим примером проблемы...** а) небезопасной десериализации. б) отсутствия проверки входных данных в формах регистрации. в) уязвимостей в цепочке поставок (Supply Chain Attacks) через сторонние зависимости. г) слабых паролей администраторов базы данных. *Правильный ответ: в*

8. **Межсайтовый скриптинг (XSS) типа Stored (Хранимый) отличается от Reflected тем, что вредоносный скрипт...** а) выполняется мгновенно в URL-строке браузера. б) сохраняется на сервере (например, в базе комментариев) и выполняется у каждого посетителя страницы. в) требует физического доступа к компьютеру жертвы. г) работает только в браузере Internet Explorer. *Правильный ответ: б*

9. **Для чего предназначен Content Security Policy (CSP) в заголовках HTTP?** а) Для сжатия передаваемых данных. б) Для строгого ограничения источников, из которых браузер может загружать скрипты, стили и изображения, что защищает от XSS. в) Для автоматического обновления SSL-сертификатов. г) Для хранения сессии пользователя на стороне сервера. *Правильный ответ: б*

10. **Что проверяет статический анализатор (SAST) в первую очередь?** а) Скорость ответа сервера под нагрузкой. б) Исходный код, байт-код или бинарный файл без фактического запуска приложения. в) Настройки сетевого экрана (Firewall). г) Наличие обновлений операционной системы. *Правильный ответ: б*

11. Проблема «Целочисленного переполнения» (*Integer Overflow*) чаще всего приводит к... а) Зависанию графического интерфейса. б) Выделению буфера памяти недостаточного размера с последующим переполнением буфера. в) Неправильному отображению даты в календаре. г) Увеличению времени компиляции проекта. *Правильный ответ: б*
12. Концепция «Secure by Design» означает, что безопасность... а) Добавляется патчем после того, как произошла первая хакерская атака. б) Является фундаментальным свойством архитектуры и закладывается на этапе требований и проектирования. в) Полностью делегируется облачному провайдеру (WAF/Cloudflare). г) Обеспечивается только сложным паролем администратора. *Правильный ответ: б*
13. небезопасная десериализация объектов возникает, когда приложение... а) Преобразует недоверенные данные от пользователя обратно в объекты в памяти без должной проверки их структуры. б) Передает данные по протоколу HTTP вместо HTTPS. в) Использует слишком длинные имена переменных. г) Хранит логи в текстовом файле вместо базы данных. *Правильный ответ: а*
14. Инструменты вроде HashiCorp Vault или AWS Secrets Manager используются в DevSecOps для... а) Генерации случайных чисел для игр. б) Безопасного хранения и управления секретами (паролями, токенами API, ключами), исключая их хранение в открытом виде в коде (Git). в) Архивации старых версий проекта. г) Визуализации диаграмм классов UML. *Правильный ответ: б*
15. Что такое SBOM (*Software Bill of Materials*)? а) Счет на оплату услуг разработчика. б) Формализованный инвентаризационный список всех компонентов, библиотек и зависимостей, входящих в состав ПО. в) Протокол передачи данных между микросервисами. г) Название популярного редактора кода. *Правильный ответ: б.*
16. Концепция внедрения проверок безопасности на самые ранние этапы жизненного цикла разработки (SDLC) для снижения стоимости исправления ошибок называется «Shift _____» (Сдвиг влево). *(Правильный ответ: Left)*
17. В методологии моделирования угроз STRIDE буква S означает _____ (подделка личности или процесса аутентификации). *(Правильный ответ: Spoofing / Спуфинг)*
18. Уязвимость, позволяющая злоумышленнику получить доступ к данным других пользователей путем прямой подмены идентификатора объекта в запросе (например, GET /api/user/10 на GET /api/user/11), называется Insecure Direct Object Reference и относится к категории нарушения _____ доступа. *(Правильный ответ: контроля / Access Control)*
19. Для защиты от SQL-инъекций наиболее надежным методом является использование параметризованных запросов, которые также называют _____ выражениями (*Prepared Statements*). *(Правильный ответ: подготовленными)*
20. Автоматизированный процесс поиска известных уязвимостей (CVE) в сторонних библиотеках и open-source компонентах проекта называется анализом состава ПО или сокращенно _____ (SCA). *(Правильные слова: Software Composition Analysis)*
21. Атака типа _____ Cross Site Scripting (Хранимый XSS) характеризуется тем, что вредоносный скрипт сохраняется на сервере (например, в базе данных комментариев) и выполняется у всех посетителей страницы. *(Правильный ответ: Stored / Сохраненный)*
22. Механизм браузеров Content Security Policy (CSP) позволяет разработчикам ограничить список доверенных источников для загрузки скриптов, что эффективно предотвращает атаки _____ выполнения кода. *(Правильный ответ: межсайтового / XSS)*
23. Статический анализ безопасности кода (SAST) изучает исходный текст приложения без его фактического _____, в отличие от динамического анализа (DAST). *(Правильный ответ: запуска / исполнения)*

24. Процесс преобразования входных данных из внешнего представления (строки JSON/XML) во внутренний формат объектов программы называется _____.
Ошибки в этом процессе часто приводят к удаленному выполнению кода. (Правильный ответ: десериализацией)

25. Документ, представляющий собой полный инвентаризационный список всех программных компонентов и зависимостей проекта, называется Bill of Materials, сокращенно ____ (SBOM). (Правильные слова: Software Bill of Materials)

26. Соответствие между этапом SDLC и активностью DevSecOps

Этап жизненного цикла Активность по обеспечению безопасности

- | | |
|---------------------------------------|--|
| 1. Проектирование (Design) | А. Проведение фаззинг-тестирования (Fuzzing) API в тестовой среде. |
| 2. Написание кода (Coding) | Б. Моделирование угроз с использованием методологии STRIDE. |
| 3. Сборка и тестирование (Build/Test) | В. Сканирование секретов (паролей, токенов) в Git-репозитории перед коммитом (pre-commit hooks). |
| 4. Эксплуатация (Deployment) | Г. Мониторинг аномалий поведения контейнера в production-среде. |

Правильные ответы: 1–Б, 2–В, 3–А, 4–Г.

27. Соответствие между аббревиатурой OWASP Top 10:2021 и типом угрозы

Категория
OWASP

Тип угрозы

- | | |
|-------------|--|
| 1. A01:2021 | А. Небезопасная десериализация, внедрение шаблонов (Injection). |
| 2. A03:2021 | Б. Использование компонентов с известными уязвимостями (Log4j, OpenSSL). |
| 3. A06:2021 | В. Нарушение контроля доступа (IDOR, Privilege Escalation). |
| 4. A08:2021 | Г. Отказ от регистрации событий и мониторинга инцидентов. |
| 5. A09:2021 | Д. Недостатки логики программного обеспечения (Business Logic Flaws). |

Правильные ответы: 1–В, 2–А, 3–Б, 4–Д, 5–Г. (Примечание: A09 — Security Logging and Monitoring Failures).

283. Соответствие между инструментом/средством и его назначением

Инструмент /
Средство

Назначение

- | | |
|------------------------------|--|
| 1. SonarQube / Checkmarx | А. Безопасное хранение секретов, ключей API и паролей вне исходного кода. |
| 2. HashiCorp Vault | Б. Интерактивный анализ безопасности приложений (IAST), работающий внутри сервера. |
| 3. Contrast Assess | В. Статический анализатор кода (SAST) для поиска уязвимостей в тексте программы. |
| 4. Docker Bench for Security | Г. Проверка конфигурации движка Docker на соответствие стандартам CIS Benchmarks. |

Правильные ответы: 1–В, 2–А, 3–Б, 4–Г.

29. Соответствие между моделью угроз и фокусом внимания

Модель
угроз

Фокус внимания

- | | |
|-----------|--|
| 1. STRIDE | А. Угрозы приватности данных (Linkability, Identifiability). |
| 2. PASTA | Б. Технические свойства информации (Конфиденциальность, Целостность, Доступность). |

Модель угроз	Фокус внимания
3. LINDDUN	В. Моделирование атак с точки зрения злоумышленника (Risk-centric подход).
4. CIA Triad	Г. Подделка, Раскрытие, Отказ от авторства, Нарушение доступности, Повышение привилегий.

Правильные ответы: 1–Г, 2–В, 3–А, 4–Б.

30. Соответствие между типом анализа и его характеристикой

Тип анализа	Характеристика
1. SAST (Static)	А. Требуется запущенного приложения; имитирует действия хакера снаружи («Черный ящик»).
2. DAST (Dynamic)	Б. Изучает код изнутри; находит ошибки на ранних этапах, но дает много ложноположительных срабатываний («Белый ящик»).
3. IAST (Interactive)	В. Анализирует состав сторонних библиотек на наличие известных CVE.
4. SCA (Composition)	Г. Комбинирует подходы, работая внутри приложения во время тестов, предоставляя контекст выполнения («Серый ящик»).

Правильные ответы: 1–Б, 2–А, 3–Г, 4–В.

1.4 Практические задания

Задание 1. Сравните стоимость исправления уязвимости на этапах «Анализ требований», «Тестирование» и «Эксплуатация». Объясните экономический смысл концепции *Shift Left* с точки зрения кривой Барри Боэма.

Задание 2. Опишите структуру ГОСТ Р 56939-2016. Какие обязательные работы по анализу безопасности должны быть выполнены при разработке ГИС или ИСПДн согласно этому стандарту?

Задание 3. Сравните модели зрелости процессов разработки: Microsoft SAMM (*Software Assurance Maturity Model*) и BSIMM (*Building Security In Maturity Model*). В чем принципиальное различие их подходов к оценке организации?

Задание 4. Раскройте содержание этапа Threat Modeling в цикле *Microsoft SDL*. Почему этот этап критически важен до начала написания кода?

Задание 5. Что такое SBOM (*Software Bill of Materials*)? Как наличие актуального SBOM помогает оперативно реагировать на появление уязвимостей типа Log4j (*Log4Shell*)?

Задание 6. Примените методологию STRIDE для моделирования угроз мобильного банковского приложения. Приведите по одному примеру угрозы для каждой буквы аббревиатуры применительно к функции перевода средств.

Задание 7. Построение дерева атак (*Attack Tree*) для сценария «Несанкционированный доступ к панели администратора веб-сайта».

Задание 8. Проектирование микросервисной архитектуры с использованием принципа наименьших привилегий (*POLP*). Как настроить права контейнера Docker так, чтобы он не мог обращаться к файловой системе хоста (*root on host*)?

Задание 9. Анализ приватностных угроз методом LINDDUN. Выберите один компонент системы обработки персональных данных и опишите угрозу типа «Identifiability» (Идентифицируемость).

Задание 10. Архитектурный паттерн «Секурити-дизайн ревью» (*Security Design Review*). Составьте чек-лист из 5 вопросов, которые архитектор должен задать разработчику перед утверждением схемы базы данных.

Задание 11. Уязвимость A01:2021 (*Broken Access Control*). Разберите пример атаки IDOR (*Insecure Direct Object Reference*): GET /api/orders?id=100. Предложите два метода защиты (на уровне контроля доступа и на уровне случайных идентификаторов).

Задание 12. Критический разбор защиты от SQL-инъекций. Почему использование функций экранирования строк (`addslashes`, `mysql_real_escape_string`) считается ненадежным по сравнению с параметризованными запросами (*Prepared Statements*)?

Задание 13. Межсайтовый скриптинг (XSS). Опишите механизм Stored XSS. Как заголовок ответа `Content-Security-Policy: default-src 'self'` предотвращает выполнение встроенных скриптов?

Задание 14. Небезопасная десериализация объектов (A08:2021). На примере Java покажите, как цепочка гаджетов (*Gadget Chain*) позволяет выполнить произвольный код через входящий JSON/XML поток.

Задание 15. Использование компонентов с известными уязвимостями (A06:2021). Разработайте политику управления зависимостями (*Dependency Management Policy*) для команды Node.js разработчиков, включающую частоту сканирований и правила обновления мажорных версий.

Задание 16. Целочисленные переполнения (*Integer Overflow*) в C++. Покажите ошибочный фрагмент кода при выделении памяти `malloc(size * count)` и объясните, как использование типов фиксированной ширины (`uint32_t`) решает проблему.

Задание 17. Ошибки конфигурации безопасности (*Security Misconfiguration*). Проанализируйте типичный `docker-compose.yml` файл и найдите минимум три ошибки (например, форвард портов базы данных наружу, отсутствие пользователя без `root`-прав, хранение переменных окружения в открытом виде).

Задание 18. Хранение секретов. Сравните подходы: хранение ключей в файлах `.env`, использование переменных окружения CI/CD и специализированные хранилища (HashiCorp Vault, AWS Secrets Manager). Почему первые два способа считаются небезопасными при компрометации runtime-среды?

Задание 19. Защита репозитория исходного кода. Механизмы предотвращения утечки секретов через Git (Pre-commit hooks, Gitleaks) и действия при обнаружении коммита с паролем в публичном доступе.

Задание 20. Заголовки безопасности HTTP (*Hardening*). Назовите пять ключевых заголовков (`Strict-Transport-Security`, `X-Frame-Options` и др.) и объясните, какой вид атак блокирует каждый из них.

Задание 21. Статический анализ (SAST). Проблема ложноположительных срабатываний (*False Positives*). Опишите процесс верификации отчета инструмента (SonarQube, Checkmarx) человеком («цензура») и критерии закрытия задачи как "Won't Fix".

Задание 22. Динамический анализ (DAST). Принцип работы «черного ящика». Почему DAST-сканеры часто пропускают логические уязвимости авторизации, но хорошо находят инъекции?

Задание 23. Фаззинг-тестирование (*Fuzzing*). Отличие генерационного фаззинга (на основе грамматик протокола) от мутационного. Для каких систем (парсеры бинарных файлов vs REST API) эффективен каждый из них?

Задание 24. Интерактивный анализ (IAST). Как инструменты вроде Contrast Assess получают контекст выполнения (стек вызовов, данные запроса) внутри сервера приложения и почему это точнее, чем классический DAST?

Задание 25. Анализ состава ПО (SCA). Как работают базы данных известных уязвимостей (NVD, VulnDB)? Опишите путь от фиксации версии библиотеки в `package.json` до получения оповещения о новой CVE.

Задание 26. Автоматизация проверок в CI/CD пайплайне. Нарисуйте схему последовательного запуска SCA -> SAST -> Unit Tests -> Container Scanning. Что произойдет, если SCA обнаружил критическую уязвимость (CVSS > 9.0)?

Задание 27. Контейнерная безопасность. Этапы сканирования образа Docker:

1. На этапе сборки (Build-time).
2. При развертывании в кластер Kubernetes (Admission Controller).
3. Runtime-мониторинг (Falco). Цель каждого этапа.

Задание 28. Безопасность API. Специфика тестирования GraphQL-интерфейсов (проблемы Batching Attacks и Alias Overloading). Чем защита GraphQL отличается от защиты классического REST?

Задание 29. План реагирования на инциденты (*Incident Response Plan*). Опишите этапы NIST SP 800-61 (Подготовка, Обнаружение, Анализ, Сдерживание, Устранение, Восстановление) применительно к инциденту взлома веб-сервера.

Задание 30. Этика исследователя безопасности. Различия между Bug Bounty программой и политикой ответственного разглашения (VDP). Юридические риски проведения тестов на проникновение без предварительного согласования правил взаимодействия (*Rules of Engagement*).

2. КРИТЕРИИ ОЦЕНИВАНИЯ ПРИ ПРОВЕДЕНИИ ТЕКУЩЕГО КОНТРОЛЯ

Вид контроля	Наименование работы	Наименование оценочных средств	Шкала оценивания
Текущий контроль	- Вопросы для обсуждения на занятиях; - Устные опросы по ранее изученному материалу; - Письменные работы: рефераты, тестовые задания; - Практические задания; - Рефераты и доклады по темам (вопросам), вынесенным на самостоятельную работу.	Оценка выступлений на практическом (семинарском) занятии, проверка заданий и аудиторных работ, устный опрос, оценивание докладов, рефератов	отлично хорошо удовлетворительно неудовлетворительно

Критерии оценивания устных ответов обучающихся

Шкала оценивания	Характеристика оценивания
отлично	оценивается ответ, который показывает прочные знания основных процессов изучаемой предметной области, отличается глубиной и полнотой раскрытия темы; владение терминологическим аппаратом; умение объяснять сущность, явлений, процессов, событий, делать выводы и обобщения, давать аргументированные ответы, приводить примеры; свободное владение монологической речью, логичность и последовательность ответа.
хорошо	оценивается ответ, обнаруживающий прочные знания основных процессов изучаемой предметной области, отличается глубиной и полнотой раскрытия темы; владение терминологическим аппаратом; умение объяснять сущность, явлений, процессов, событий, делать выводы и обобщения, давать аргументированные ответы, приводить примеры; свободное владение монологической речью, логичность и последовательность ответа. Однако допускается одна - две неточности в ответе.
удовлетворит	оценивается ответ, свидетельствующий в основном о знании процессов

ельно	изучаемой предметной области, отличающийся недостаточной глубиной и полнотой раскрытия темы; знанием основных вопросов теории; слабо сформированными навыками анализа явлений, процессов, недостаточным умением давать аргументированные ответы и приводить примеры; недостаточно свободным владением монологической речью, логичностью и последовательностью ответа. Допускается несколько ошибок в содержании ответа.
неудовлетворительно	оценивается ответ, обнаруживающий незнание процессов изучаемой предметной области, отличающийся неглубоким раскрытием темы; незнанием основных вопросов теории, несформированными навыками анализа явлений, процессов; неумением давать аргументированные ответы, слабым владением монологической речью, отсутствием логичности и последовательности. Допускаются серьезные ошибки в содержании ответа.

Критерии оценивания работы обучающихся на практических и семинарских занятиях

Шкала оценивания	Показатели	Критерии
Отлично	1. Полнота выполнения практического и тестового задания (полнота ответа); 2. Своевременность выполнения задания; 3. Последовательность и рациональность выполнения	Задание решено самостоятельно. При этом составлен правильный алгоритм решения задания, в логических рассуждениях, в выборе формул и решении нет ошибок, получен верный ответ, задание решено рациональным способом. Дан правильный и исчерпывающий ответ на поставленные теоретические и тестовые вопросы, в которых обучающийся показал всестороннее системное знание программного материала, усвоение основной и дополнительной литературы, четкое владение понятийным аппаратом.
Хорошо	практического задания (логичность и четкость ответа); 4. Правильность ответов на вопросы; 5. Самостоятельность решения (владение дополнительным материалом); 6. Знание нормативно-законодательной базы и терминологии курса	Задание решено с помощью преподавателя. При этом составлен правильный алгоритм решения задания, в логическом рассуждении и решении нет существенных ошибок; правильно сделан выбор формул для решения; есть объяснение решения, но задание решено нерациональным способом или допущено не более двух несущественных ошибок, получен верный ответ. На поставленные теоретические и тестовые вопросы, при которых обучающийся показал достаточный уровень знаний основного программного материала: освоение информации лекционного курса и учебных пособий, овладение понятийным аппаратом, методикой исследований при попытке анализа различных ситуаций.
Удовлетворительно		Задание решено с подсказками преподавателя. Задание решено в общем виде. Обучающийся показал средний уровень знаний основного программного материала, но не мог убедительно аргументировать свой ответ, ошибся в использовании понятийного аппарата, показал недостаточные знания литературных источников.
Неудовлетв		Задание не решено.

Шкала оценивания	Показатели	Критерии
удовлетворительно		Обучающийся продемонстрировал значительные пробелы в знаниях основного программного материала, не аргументировал свой ответ, показал неудовлетворительные знания понятийного аппарата и специальной литературы.

Критерии оценивания рефератов

Средство контроля	Критерии оценивания	Шкала оценивания
Реферат	Выполнены все требования к написанию и защите реферата: обозначена проблема и обоснована её актуальность, сделан краткий анализ различных точек зрения на рассматриваемую проблему и логично изложена собственная позиция, сформулированы выводы, тема раскрыта полностью, выдержан объём, соблюдены требования к внешнему оформлению, даны правильные ответы на дополнительные вопросы. Реферат раскрывает поднятую проблематику в полном объеме.	отлично
	Основные требования к реферату и его защите выполнены, но при этом допущены недочёты. В частности, имеются неточности в изложении материала; отсутствует логическая последовательность в суждениях; не выдержан объём реферата; имеются упущения в оформлении; на дополнительные вопросы при защите даны неполные ответы. В реферате имеются неточности и предметная область выступления раскрыта не в полной мере.	хорошо
	Имеются существенные отступления от требований к реферированию. В частности: тема освещена лишь частично; допущены фактические ошибки в содержании реферата или при ответе на дополнительные вопросы; во время защиты отсутствует вывод. В реферате не в полной степени раскрыт понятийный аппарат, имеются существенные неточности в процессе формирования выводов.	удовлетворительно
	Тема реферата не раскрыта, обнаруживается существенное непонимание проблемы. Тема реферата не раскрыта или выполнена не по существу ранее поставленного вопроса. Реферат не сдан / доклад не сделан.	неудовлетворительно

Критерии оценивания тестов

Средство контроля	Критерии оценивания – процент положительных ответов	Шкала оценивания
Тестирование	90-100	отлично
	70-89	хорошо
	40-69	удовлетворительно
	< 39	неудовлетворительно

3. ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

Средства оценивания в ходе промежуточной аттестации:

- вопросы для зачета с оценкой;
- тестовые задания к зачету.

Коды компетенции	Результаты освоения ОПОП	Перечень планируемых результатов обучения по дисциплине
ПК-5	Способен принимать участие в формировании, организовывать и поддерживать выполнение комплекса мер по обеспечению информационной безопасности, управлять процессом их реализации	ПК-5.1. Знает принципы формирования комплекса мер по обеспечению информационной безопасности предприятия (организации); ПК-5.2. владеет методами организации и управления деятельностью служб защиты информации на предприятии

3.1. Вопросы к зачету

1. Понятие «Разработка защищенного ПО» (РБПО). Основные отличия от традиционного тестирования безопасности после завершения разработки.
2. Жизненный цикл безопасной разработки (Secure SDLC / SSDLC). Описание этапов и интеграция практик ИБ в каждый из них.
3. Экономическое обоснование концепции *Shift Left*. Сравнение стоимости исправления уязвимостей на этапах требований, кодирования и эксплуатации.
4. Стандарт ГОСТ Р 56939-2016: область применения, основные требования к разработке ГИС и ИСПДн.
5. Модели зрелости процессов разработки (*Maturity Models*): сравнительный анализ Microsoft SAMM и BSIMM.
6. Концепция «Security by Design» и «Privacy by Design». Примеры реализации принципов при проектировании архитектуры БД.
7. Управление техническим долгом безопасности (*Security Technical Debt*).
8. Роль комплаенса и регуляторов (ФСТЭК, ФСБ) в процессе создания средств защиты информации (СЗИ) в РФ.
9. Программно-аппаратные среды исполнения (TEE, Secure Enclave): их роль в защите мобильных приложений и транзакций.
10. Лицензионная чистота и управление юридическими рисками при использовании Open Source компонентов.
11. Методология STRIDE: расшифровка аббревиатуры и примеры угроз для веб-сервиса авторизации.
12. Построение деревьев атак (*Attack Trees*): алгоритм составления и расчет вероятности реализации атаки.
13. Модель LINDDUN для анализа приватностных угроз применительно к системам обработки биометрических данных.
14. Метод PASTA (Process for Attack Simulation and Threat Analysis): этапы бизнес-анализа и технического моделирования.
15. Проектирование микросервисной архитектуры с использованием принципа наименьших привилегий (*POLP*) и эшелонированной обороны (*Defense in Depth*).
16. Угрозы API: специфика моделирования рисков для REST и GraphQL интерфейсов.
17. Безопасность облачных архитектур: модель разделенной ответственности (*Shared Responsibility Model*) у провайдеров IaaS/PaaS.

18. Использование визуальных языков моделирования (UML/SysML) для описания протоколов безопасного обмена данными.
19. Анализ поверхности атаки (*Attack Surface Analysis*): методика минимизации доступных входов в систему.
20. Дизайн-ревью безопасности (*Security Design Review*): критерии оценки качества архитектуры перед началом спринта разработки.
21. Уязвимость A01:2021 (Broken Access Control). Разбор механизмов IDOR и вертикального повышения привилегий.
22. Защита от инъекций (A03:2021). Почему параметризованные запросы (*Prepared Statements*) являются единственным надежным методом против SQLi?
23. Межсайтовый скриптинг (XSS, A03:2021). Различия Stored, Reflected и DOM-based XSS. Механизмы нейтрализации.
24. небезопасная десериализация объектов (A08:2021). Принцип работы цепочек гаджетов (*Gadget Chains*) на примере Java или PHP.
25. Использование компонентов с известными уязвимостями (A06:2021). Кейс Log4Shell (Log4j): причины катастрофического масштаба.
26. Нарушение целостности криптографии (A02:2021). Типичные ошибки использования собственных алгоритмов шифрования и генерации случайных чисел.
27. Ошибки идентификации и аутентификации (A07:2021). Защита от Brute Force, Credential Stuffing и слабых политик паролей.
28. Целочисленные переполнения (*Integer Overflow/Underflow*) в языках низкого уровня (C/C++). Методы предотвращения через SafeInt и проверку границ.
29. Server-Side Request Forgery (SSRF). Механизм атаки и методы фильтрации запросов на уровне приложения и сети.
30. Логические уязвимости бизнес-логики (Business Logic Flaws). Примеры обхода корзины покупок или манипуляции курсами валют.
31. Хранение секретов (паролей, токенов API, ключей). Критика хранения в .env файлах и переменных окружения CI/CD.
32. Работа с хранилищами секретов: HashiCorp Vault, AWS Secrets Manager, Azure Key Vault. Принципы динамической выдачи учетных записей.
33. Утечка секретов через публичные репозитории Git. Инструменты автоматического поиска (Gitleaks, TruffleHog) и предкоммит-хуки.
34. Hardening серверной инфраструктуры и контейнеров. CIS Benchmarks для Linux-серверов и Docker.
35. Заголовки безопасности HTTP (*Hardening Headers*): Content-Security-Policy, Strict-Transport-Security, X-Frame-Options.
36. Управление конфигурацией инфраструктуры как кода (*IaC Security*). Поиск уязвимостей в Terraform и CloudFormation шаблонах.
37. Безопасная работа с файловой системой: предотвращение Path Traversal и Race Conditions (TOCTOU).
38. Настройка кросс-доменных политик: сравнение JSONP, CORS и механизма preflight-запросов.
39. Обезличивание и маскирование данных (*Data Masking*) в средах разработки и тестирования.
40. Аудит конфигурации облака (CSPM - *Cloud Security Posture Management*).
41. Статический анализ (SAST): принцип работы, ложноположительные (*False Positives*) и ложноотрицательные (*False Negatives*) срабатывания.
42. Динамический анализ (DAST/IAST): имитация атак на работающее приложение. Преимущества интерактивного анализа (IAST) над классическим DAST.
43. Анализ состава ПО (SCA - *Software Composition Analysis*). Автоматизация отслеживания уязвимостей в зависимостях (SBOM).

44. Фаззинг-тестирование (*Fuzzing*): мутационный vs генерационный фаззинг. Применение инструментов AFL++ и libFuzzer.
45. Символьное выполнение (*Symbolic Execution*) и формальные методы верификации критических участков кода.
46. Тестирование на проникновение (*Penetration Testing*): различие между Black Box, Gray Box и White Box подходами.
47. Анализ покрытия кода тестами (*Code Coverage*) с точки зрения безопасности. Достаточно ли 100% покрытия для гарантии отсутствия уязвимостей?
48. Reverse Engineering бинарных файлов: инструменты Ghidra/IDA Pro для поиска закладок и недекларированных возможностей (НДВ).
49. Контрольные суммы и цифровые подписи исполняемых файлов. Проверка целостности дистрибутива перед установкой.
50. Red Teaming: оценка готовности всей организации (людей, процессов и техники) к целенаправленной атаке.
51. Интеграция сканеров безопасности в пайплайны CI/CD (*DevSecOps*). Стратегии Fail-Fast и Break-Build.
52. Безопасность контейнеризации (Docker/Kubernetes). Специфика оркестрации: Network Policies, Pod Security Standards, Secrets management в K8s.
53. Runtime Application Self-Protection (RASP) и Web Application Firewall (WAF): место в архитектуре и ограничения эффективности.
54. Мониторинг безопасности (*Application Logging*). Что логировать, а что категорически нельзя (GDPR, персональные данные).
55. EDR/XDR системы для защиты конечных точек разработчиков и сборочных агентов.
56. Планирование непрерывности бизнеса (BCP) и восстановления после сбоев (DRP) для ИТ-инфраструктуры разработки.
57. Процесс управления уязвимостями (*Vulnerability Management Lifecycle*): приоритизация по CVSS v3.1 и SLA на устранение.
58. Реагирование на инциденты ИБ (*Incident Response*). Этапы подготовки, обнаружения, сдерживания и извлечения уроков (*Lessons Learned*).
59. Этика исследователя безопасности. Правовые аспекты Bug Bounty программ и политики ответственного разглашения (VDP).
60. Безопасный вывод ПО из эксплуатации (*Sunsetting/Data Disposal*). Методы гарантированного удаления данных с носителей

3.2. Задания для зачета:

Задание 1. Сравните стоимость исправления уязвимости на этапах «Анализ требований», «Тестирование» и «Эксплуатация». Объясните экономический смысл концепции *Shift Left* с точки зрения кривой Барри Боэма.

Задание 2. Опишите структуру ГОСТ Р 56939-2016. Какие обязательные работы по анализу безопасности должны быть выполнены при разработке ГИС или ИСПДн согласно этому стандарту?

Задание 3. Сравните модели зрелости процессов разработки: Microsoft SAMM (*Software Assurance Maturity Model*) и BSIMM (*Building Security In Maturity Model*). В чем принципиальное различие их подходов к оценке организации?

Задание 4. Раскройте содержание этапа Threat Modeling в цикле *Microsoft SDL*. Почему этот этап критически важен до начала написания кода?

Задание 5. Что такое SBOM (*Software Bill of Materials*)? Как наличие актуального SBOM помогает оперативно реагировать на появление уязвимостей типа Log4j (*Log4Shell*)?

Задание 6. Примените методологию STRIDE для моделирования угроз мобильного банковского приложения. Приведите по одному примеру угрозы для каждой буквы аббревиатуры применительно к функции перевода средств.

Задание 7. Построение дерева атак (*Attack Tree*) для сценария «Несанкционированный доступ к панели администратора веб-сайта».

Задание 8. Проектирование микросервисной архитектуры с использованием принципа наименьших привилегий (*POLP*). Как настроить права контейнера Docker так, чтобы он не мог обращаться к файловой системе хоста (*root on host*)?

Задание 9. Анализ приватностных угроз методом LINDDUN. Выберите один компонент системы обработки персональных данных и опишите угрозу типа «Identifiability» (Идентифицируемость).

Задание 10. Архитектурный паттерн «Секурити-дизайн ревью» (*Security Design Review*). Составьте чек-лист из 5 вопросов, которые архитектор должен задать разработчику перед утверждением схемы базы данных.

Задание 11. Уязвимость A01:2021 (*Broken Access Control*). Разберите пример атаки IDOR (*Insecure Direct Object Reference*): GET /api/orders?id=100. Предложите два метода защиты (на уровне контроля доступа и на уровне случайных идентификаторов).

Задание 12. Критический разбор защиты от SQL-инъекций. Почему использование функций экранирования строк (*addslashes*, *mysql_real_escape_string*) считается ненадежным по сравнению с параметризованными запросами (*Prepared Statements*)?

Задание 13. Межсайтовый скриптинг (XSS). Опишите механизм Stored XSS. Как заголовок ответа Content-Security-Policy: default-src 'self' предотвращает выполнение встроенных скриптов?

Задание 14. Небезопасная десериализация объектов (A08:2021). На примере Java покажите, как цепочка гаджетов (*Gadget Chain*) позволяет выполнить произвольный код через входящий JSON/XML поток.

Задание 15. Использование компонентов с известными уязвимостями (A06:2021). Разработайте политику управления зависимостями (*Dependency Management Policy*) для команды Node.js разработчиков, включающую частоту сканирований и правила обновления мажорных версий.

Задание 16. Целочисленные переполнения (*Integer Overflow*) в C++. Покажите ошибочный фрагмент кода при выделении памяти `malloc(size * count)` и объясните, как использование типов фиксированной ширины (`uint32_t`) решает проблему.

Задание 17. Ошибки конфигурации безопасности (*Security Misconfiguration*). Проанализируйте типичный `docker-compose.yml` файл и найдите минимум три ошибки (например, форвард портов базы данных наружу, отсутствие пользователя без `root`-прав, хранение переменных окружения в открытом виде).

Задание 18. Хранение секретов. Сравните подходы: хранение ключей в файлах `.env`, использование переменных окружения CI/CD и специализированные хранилища (HashiCorp Vault, AWS Secrets Manager). Почему первые два способа считаются небезопасными при компрометации runtime-среды?

Задание 19. Защита репозитория исходного кода. Механизмы предотвращения утечки секретов через Git (*Pre-commit hooks*, *Gitleaks*) и действия при обнаружении коммита с паролем в публичном доступе.

Задание 20. Заголовки безопасности HTTP (*Hardening*). Назовите пять ключевых заголовков (*Strict-Transport-Security*, *X-Frame-Options* и др.) и объясните, какой вид атак блокирует каждый из них.

Задание 21. Статический анализ (SAST). Проблема ложноположительных срабатываний (*False Positives*). Опишите процесс верификации отчета инструмента (SonarQube, Checkmarx) человеком («цензура») и критерии закрытия задачи как "Won't Fix".

Задание 22. Динамический анализ (DAST). Принцип работы «черного ящика». Почему DAST-сканеры часто пропускают логические уязвимости авторизации, но хорошо находят инъекции?

Задание 23. Фаззинг-тестирование (Fuzzing). Отличие генерационного фаззинга (на основе грамматик протокола) от мутационного. Для каких систем (парсеры бинарных файлов vs REST API) эффективен каждый из них?

Задание 24. Интерактивный анализ (IAST). Как инструменты вроде Contrast Assess получают контекст выполнения (стек вызовов, данные запроса) внутри сервера приложения и почему это точнее, чем классический DAST?

Задание 25. Анализ состава ПО (SCA). Как работают базы данных известных уязвимостей (NVD, VulnDB)? Опишите путь от фиксации версии библиотеки в package.json до получения оповещения о новой CVE.

Задание 26. Автоматизация проверок в CI/CD пайплайне. Нарисуйте схему последовательного запуска SCA -> SAST -> Unit Tests -> Container Scanning. Что произойдет, если SCA обнаружил критическую уязвимость (CVSS > 9.0)?

Задание 27. Контейнерная безопасность. Этапы сканирования образа Docker:

4. На этапе сборки (Build-time).
5. При развертывании в кластер Kubernetes (Admission Controller).
6. Runtime-мониторинг (Falco). Цель каждого этапа.

Задание 28. Безопасность API. Специфика тестирования GraphQL-интерфейсов (проблемы Batching Attacks и Alias Overloading). Чем защита GraphQL отличается от защиты классического REST?

Задание 29. План реагирования на инциденты (Incident Response Plan). Опишите этапы NIST SP 800-61 (Подготовка, Обнаружение, Анализ, Сдерживание, Устранение, Восстановление) применительно к инциденту взлома веб-сервера.

Задание 30. Этика исследователя безопасности. Различия между Bug Bounty программой и политикой ответственного разглашения (VDP). Юридические риски проведения тестов на проникновение без предварительного согласования правил взаимодействия (Rules of Engagement).

4. ОСНОВНЫЕ КРИТЕРИИ ОЦЕНИВАНИЯ ПРИ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

Шкала оценивания уровня сформированности универсальной компетенций (зачет с оценкой)

Формируемые уровни освоения компетенций	Критерии оценивания	Шкала оценивания
Высокий уровень	Изложено правильное понимание вопроса, четко и самостоятельно дан исчерпывающий ответ, содержание раскрыто полно, профессионально, грамотно. Обучающимся усвоена взаимосвязь основных понятий дисциплины в их значении для приобретаемой профессии. Отражает успешное и систематическое применение навыков и умений по данной дисциплине в соответствии с ФГОС.	отлично

Базовый уровень	Изложено правильное понимание вопроса, дано достаточно подробное описание предмета ответа, приведены и раскрыты в тезисной форме основные понятия, относящиеся к предмету ответа. Ответ отражает полное знание учебно-программного материала, систематический характер знаний по дисциплине, а также наличие базового уровня овладения практическими умениями и навыками по данной дисциплине в соответствии с ФГОС	хорошо
Пороговый уровень	Ответ отражает теоретические знания основного учебно-программного материала в объеме, необходимом для дальнейшей учебы и предстоящей работы по профессии. Данная оценка может быть выставлена обучающемуся, допустившему неточности в ответе, но обладающими необходимыми знаниями для их устранения под руководством преподавателя, отмечен начальный уровень овладения практическими умениями и навыками по данной дисциплине в соответствии с ФГОС	удовлетворительно
Неудовлетворительный уровень	При ответе обучающегося обнаружено отсутствие знаний, умений и навыков и/или фрагментарные знания основного учебно-программного материала.	неудовлетворительно

Текущий контроль и промежуточная аттестация осуществляются в соответствии с «Положением о текущей и промежуточной аттестации обучающихся в Автономной некоммерческой организации «Образовательная организация высшего образования» «Университет экономики и управления».

Вид промежуточной аттестации – зачет с оценкой.

Форма проведения промежуточной аттестации – письменный зачет.