Приложение 1.2.6 к ОПОП по специальности
09.02.11 Разработка и управление программным обеспечением

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет им. Г.И. Носова»

Многопрофильный колледж

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ЛАБОРАТОРНЫХ ЗАНЯТИЙ УЧЕБНОЙ ДИСЦИПЛИНЫ

МДК.02.06 Безопасность программного обеспечения

для обучающихся специальности
09.02.11 Разработка и управление программным обеспечением

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

Важную часть теоретической и профессиональной практической подготовки обучающихся составляют лабораторные занятия.

Состав и содержание лабораторных занятий направлены на реализацию Федерального государственного образовательного стандарта среднего профессионального образования.

Ведущей дидактической целью лабораторных занятий является экспериментальное подтверждение и проверка существенных теоретических положений (законов, зависимостей).

В соответствии с рабочей программой учебной дисциплины «Безопасность программного обеспечения» предусмотрено проведение лабораторных занятий.

В результате их выполнения, обучающийся должен уметь:

У 2.1.1 разрабатывать и внедрять системы защиты баз данных от несанкционированного доступа;

У 2.1.9 учитывать требования к масштабируемости, производительности и безопасности при проектировании модуля;

У 2.2.1 разрабатывать модули программного обеспечения с использованием различных языков программирования и технологий;

У 2.2.5 обеспечивать безопасность, производительность и масштабируемость при разработке модулей;

У 2.2.8 улучшать производительность модулей, выявляя и устраняя узкие места;

У 2.2.9 проводить анализ и мониторинг производительности приложений;

У 2.2.10 применять инструменты для рефакторинга и оптимизации программного кода;

У 2.3.1 интегрировать модули и компоненты, обеспечивая их взаимодействие;

У 2.3.2 работать с API и устанавливать соединения между компонентами;

У 2.3.5 работать с различными форматами данных и протоколами передачи данных;

У 2.4.1 анализировать требования к программному обеспечению и составлять планы тестирования;

У 2.4.2 создавать тестовые сценарии и тест-кейсы для проверки функциональности и соответствия требованиям;

У 2.4.3 выполнять тестирование программного обеспечения вручную и автоматизировать процесс тестирования;

У 2.4.4 анализировать результаты тестирования и документировать найденные ошибки;

У 2.4.5 разрабатывать стратегии отладки и исправлять ошибки в программном обеспечении;

У 2.4.6 выполнять модульные тесты с использованием инструментов тестирования, в том числе автоматизированного тестирования;

У 2.4.8 составлять отчет о выполнении тестирования ПО;

У 2.5.1 описывать функциональность модулей в документации;

У 2.5.2 создавать диаграммы для иллюстрации работы модулей;

У 2.5.5 вести журнал изменений и фиксировать обновления программных модулей;

У 2.5.7 включать в документацию особенности модулей, такие как ограничения, уязвимости или оптимальные настройки;

У 2.5.8 проводить регулярное обновление документации при изменении модулей или добавлении нового функционала.

Содержание лабораторных занятий ориентировано на подготовку обучающихся к освоению профессионального модуля программы подготовки специалистов среднего звена по специальности и овладению ***профессиональными компетенциями:***

ПК 2.2. Разрабатывать модули программного обеспечения

ПК 2.4 Выполнять тестирование и отладку программного обеспечения

ПК 2.5 Осуществлять документирование программных модулей программного обеспечения

А также формированию *общих компетенций*:

ОК 01. Выбирать способы решения задач профессиональной деятельности применительно

к различным контекстам

ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста

ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках

Лабораторные занятия проводятся в рамках соответствующей темы, после освоения дидактических единиц, которые обеспечивают наличие знаний, необходимых для ее выполнения.

2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Лабораторное занятие №103

SQL инъекции - эксплуатация и защита уязвимого приложения

Цель: освоить механизмы безопасной авторизации OAuth 2.0 с использованием PKCE (Proof Key for Code Exchange) и реализовать защиту от перехвата кода авторизации.

Выполнив работу, вы будете уметь:

У 2.2.5 обеспечивать безопасность, производительность и масштабируемость при разработке модулей;

У 2.4.3 выполнять тестирование программного обеспечения вручную и автоматизировать процесс тестирования;

У 2.4.4 анализировать результаты тестирования и документировать найденные ошибки;

У 2.5.7 включать в документацию особенности модулей, такие как ограничения, уязвимости или оптимальные настройки.

Материальное обеспечение: как в таблице 2.3

ПК, уязвимое веб-приложение, SQLMap, IDE, СУБД

Задание:

1. Проанализировать уязвимое приложение на наличие SQL-инъекций
2. Эксплуатировать уязвимости с помощью SQLMap и вручную
3. Реализовать защиту с использованием параметризованных запросов
4. Протестировать защищенное приложение на устойчивость к атакам

Порядок выполнения работы:

1. Развертывание уязвимого приложения
2. Разведка: поиск точек ввода данных
3. Эксплуатация: извлечение данных через UNION-атаки, blind SQLi
4. Реализация защиты: замена конкатенации на параметризованные запросы
5. Тестирование защиты: повторная проверка уязвимостей

Форма представления результата:

Отчет с листингами кода (уязвимого и защищенного), скриншотами эксплуатации, результатами тестирования SQLMap

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №104

XSS атаки - создание и предотвращение межсайтового скриптинга

Цель: научиться тестировать XSS-уязвимости различных типов, применять методы экранирования и Content Security Policy для защиты

Выполнив работу, вы будете уметь:

У 2.2.5 обеспечивать безопасность, производительность и масштабируемость при разработке модулей;

У 2.3.2 работать с API и устанавливать соединения между компонентами;

У 2.4.3 выполнять тестирование программного обеспечения вручную и автоматизировать процесс тестирования;

У 2.4.5 разрабатывать стратегии отладки и исправлять ошибки в программном обеспечении.

Материальное обеспечение:

ПК, браузеры, инструменты сканирования, IDE, веб-серверы.

Задание:

1. Создать тестовое приложение с XSS-уязвимостями
2. Реализовать атаки различных типов XSS
3. Внедрить механизмы защиты (экранирование, CSP)
4. Протестировать эффективность защиты

Порядок выполнения работы:

1. Создание уязвимого приложения с формами ввода
2. Эксплуатация Reflected XSS через параметры URL
3. Эксплуатация Stored XSS через сохранение в БД
4. Реализация экранирования htmlspecialchars/escape
5. Настройка заголовков Content-Security-Policy
6. Проверка защиты с помощью XSS-пейлодов

Форма представления результата:

Отчет с кодом приложения, примерами XSS-пейлодов, конфигурацией CSP, скриншотами атак и защиты

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №105
CSRF защита - реализация токенов и проверки Origin/Referer

Цель: освоить методы защиты от CSRF-атак, научиться внедрять CSRF-токены и настраивать валидацию заголовков.

Выполнив работу, вы будете уметь:

У 2.2.5 обеспечивать безопасность, производительность и масштабируемость при разработке модулей;

У 2.3.2 работать с API и устанавливать соединения между компонентами;

У 2.4.3 выполнять тестирование программного обеспечения вручную и автоматизировать процесс тестирования;

У 2.4.5 разрабатывать стратегии отладки и исправлять ошибки в программном обеспечении.

Материальное обеспечение:

ПК, веб-фреймворки, IDE, браузеры, прокси-инструменты

Задание:

1. Создать приложение с CSRF-уязвимостью
2. Реализовать CSRF-атаку через поддельный запрос
3. Внедрить защиту с использованием токенов
4. Настроить проверку заголовков Origin/Referer
5. Реализовать SameSite cookies

Порядок выполнения работы:

1. Разработка уязвимого приложения с формами
2. Создание CSRF-атаки (HTML-форма с автоотправкой)
3. Генерация уникальных токенов для сессий
4. Добавление токенов во все формы
5. Серверная проверка токенов
6. Настройка SameSite=Strict для cookies
7. Валидация заголовков Origin и Referer

Форма представления результата:

Отчет с кодом (уязвимым и защищенным), примером CSRF-атаки, описанием механизма защиты токенами

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №106
Составление модели угроз для типового веб-приложения

Цель: научиться применять методологию STRIDE для выявления угроз безопасности и документировать модель угроз.

Выполнив работу, вы будете уметь:

У 2.1.9 учитывать требования к масштабируемости, производительности и безопасности при проектировании модуля;

У 2.4.1 анализировать требования к программному обеспечению и составлять планы тестирования;

У 2.5.1 описывать функциональность модулей в документации;

У 2.5.2 создавать диаграммы для иллюстрации работы модулей.

Материальное обеспечение:

ПК, средства моделирования угроз, IDE, офисное ПО

Задание:

1. Проанализировать архитектуру веб-приложения
2. Выявить компоненты и потоки данных
3. Применить STRIDE для каждой категории угроз
4. Составить документ модели угроз
5. Предложить контрмеры

Порядок выполнения работы:

1. Описание архитектуры приложения (диаграмма DFD)
2. Идентификация активов (данные, сервисы)
3. Анализ угроз по STRIDE
4. Оценка рисков (вероятность × воздействие)
5. Разработка контрмер для каждой угрозы
6. Оформление документа модели угроз

Форма представления результата:

Документ с моделью угроз (DFD, таблица угроз STRIDE, матрица рисков, план контрмер)

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №107
Настройка безопасной аутентификации с JWT и refresh токенами

Цель: реализовать безопасную систему аутентификации с использованием JWT access и refresh токенов.

Выполнив работу, вы будете уметь:

У 2.2.1 разрабатывать модули программного обеспечения с использованием различных языков программирования и технологий;

У 2.2.5 обеспечивать безопасность, производительность и масштабируемость при разработке модулей;

У 2.3.2 работать с API и устанавливать соединения между компонентами;

У 2.4.5 разрабатывать стратегии отладки и исправлять ошибки в программном обеспечении.

Материальное обеспечение:

ПК, серверная среда, IDE, инструменты работы с токенами

Задание:

1. Реализовать выдачу JWT access токенов
2. Настроить refresh токены для обновления сессии
3. Реализовать валидацию и проверку подписи
4. Внедрить механизм отзыва токенов
5. Настроить безопасное хранение токенов

Порядок выполнения работы:

1. Генерация JWT с claim (sub, exp, iat, iss)
2. Подпись токена алгоритмом HS256/RS256
3. Реализация endpoint /login для выдачи токенов
4. Создание refresh token с длительным сроком
5. Endpoint /refresh для обновления access token
6. Middleware для проверки токенов
7. Blacklist для отозванных токенов
8. HttpOnly cookies для refresh токенов

Форма представления результата:

Отчет с листингами кода, примерами JWT токенов (расшифрованными), схемой работы refresh flow

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №108
Шифрование данных с использованием AES и RSA

Цель: освоить применение симметричного (AES) и асимметричного (RSA) шифрования, научиться управлять криптографическими ключами.

Выполнив работу, вы будете уметь:

У 2.2.1 разрабатывать модули программного обеспечения с использованием различных языков программирования и технологий;

У 2.2.5 обеспечивать безопасность, производительность и масштабируемость при разработке модулей;

У 2.4.5 разрабатывать стратегии отладки и исправлять ошибки в программном обеспечении;

У 2.5.7 включать в документацию особенности модулей, такие как ограничения, уязвимости или оптимальные настройки.

Материальное обеспечение:

ПК, криптографические библиотеки, IDE, терминал

Задание:

1. Сгенерировать ключи для AES и RSA
2. Реализовать шифрование/расшифрование AES
3. Реализовать шифрование/расшифрование RSA
4. Создать гибридную схему шифрования
5. Обеспечить безопасное хранение ключей

Порядок выполнения работы:

1. Генерация AES-256 ключа (32 байта)
2. Шифрование данных AES в режиме CBC/GCM
3. Генерация RSA ключевой пары (2048+ бит)
4. Шифрование данных открытым ключом RSA
5. Расшифрование закрытым ключом RSA
6. Гибридная схема: AES для данных, RSA для ключа
7. Сохранение ключей в зашифрованном виде

Форма представления результата:

Отчет с кодом шифрования/расшифрования, примерами зашифрованных данных, описанием управления ключами

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №109
Хэширование паролей с salt и adaptive functions (bcrypt, Argon2)

Цель: научиться правильно хэшировать пароли с использованием современных адаптивных функций и соли.

Выполнив работу, вы будете уметь:

У 2.1.1 разрабатывать и внедрять системы защиты баз данных от несанкционированного доступа;

У 2.2.5 обеспечивать безопасность, производительность и масштабируемость при разработке модулей;

У 2.4.3 выполнять тестирование программного обеспечения вручную и автоматизировать процесс тестирования;

У 2.4.4 анализировать результаты тестирования и документировать найденные ошибки.

Материальное обеспечение:

ПК, криптографические библиотеки, IDE, СУБД

Задание:

1. Реализовать хэширование паролей с bcrypt
2. Реализовать хэширование с Argon2
3. Настроить параметры сложности
4. Создать систему проверки паролей
5. Реализовать миграцию со старых хэшей

Порядок выполнения работы:

1. Генерация уникальной соли для каждого пароля
2. Хэширование с bcrypt (cost factor 12+)
3. Хэширование с Argon2id (память, итерации, параллелизм)
4. Сохранение хэша в БД (включая соль и параметры)
5. Проверка пароля: извлечение параметров, вычисление хэша, сравнение
6. Реализация re-hash при изменении параметров
7. Сравнение производительности алгоритмов

Форма представления результата:

Отчет с кодом хэширования, примерами хэшей, таблицей сравнения алгоритмов, рекомендациями по параметрам

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №110
Анализ сетевого трафика с помощью Wireshark

Цель: освоить захват и анализ сетевых пакетов, научиться выявлять аномалии и уязвимости в сетевом трафике.

Выполнив работу, вы будете уметь:

У 2.2.9 проводить анализ и мониторинг производительности приложений;
У 2.4.1 анализировать требования к программному обеспечению и составлять планы тестирования;
У 2.4.4 анализировать результаты тестирования и документировать найденные ошибки;
У 2.4.8 составлять отчет о выполнении тестирования ПО.

Материальное обеспечение:

ПК, Wireshark, сетевые интерфейсы, фильтры анализа

Задание:

1. Настроить захват трафика на сетевом интерфейсе
2. Проанализировать HTTP/HTTPS трафик
3. Выявить передачу конфиденциальных данных
4. Обнаружить признаки сетевых атак
5. Создать фильтры для детектирования аномалий

Порядок выполнения работы:

1. Запуск Wireshark, выбор интерфейса
2. Захват трафика при посещении сайтов
3. Применение фильтров: http, tcp.port==80; tls, tcp.port==443; ip.addr, tcp.flags
4. Анализ HTTP-запросов (методы, заголовки, cookies)
5. Поиск паролей в открытом виде
6. Выявление сканирования портов (SYN flood)
7. Создание display filters для аномалий
8. Экспорт объектов из трафика

Форма представления результата:

Отчет со скриншотами Wireshark, примерами фильтров, анализом найденных уязвимостей, рекомендациями по защите

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №111
Настройка HTTPS и создание самоподписанных сертификатов

Цель: научиться генерировать SSL/TLS сертификаты, настраивать HTTPS на веб-сервере и проверять цепочки доверия.

Выполнив работу, вы будете уметь:

У 2.1.1 разрабатывать и внедрять системы защиты баз данных от несанкционированного доступа;

У 2.2.5 обеспечивать безопасность, производительность и масштабируемость при разработке модулей;

У 2.4.5 разрабатывать стратегии отладки и исправлять ошибки в программном обеспечении.

Материальное обеспечение:

ПК, OpenSSL, веб-серверы, IDE, браузеры

Задание:

1. Сгенерировать самоподписанный SSL-сертификат
2. Создать CA и подписать сертификат
3. Настроить HTTPS на веб-сервере
4. Реализовать перенаправление HTTP→HTTPS
5. Проверить конфигурацию TLS

Порядок выполнения работы:

1. Генерация приватного ключа RSA (openssl genrsa)
2. Создание CSR (Certificate Signing Request)
3. Самоподпись сертификата (openssl req -x509)
4. Создание собственного CA: Генерация CA ключа, Самоподпись CA сертификата, Подпись серверного сертификата CA
5. Настройка Nginx/Apache для HTTPS
6. Конфигурация TLS 1.2/1.3, cipher suites
7. HSTS заголовок
8. Проверка в браузере и SSL Labs

Форма представления результата:

Отчет с командами OpenSSL, конфигурацией веб-сервера, скриншотами работы HTTPS, результатом проверки SSL Labs

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №112
Защита от brute-force атак с ограничением попыток входа

Цель: реализовать механизмы защиты от перебора паролей, включая rate limiting, блокировку IP и учет неудачных попыток.

Выполнив работу, вы будете уметь:

У 2.2.5 обеспечивать безопасность, производительность и масштабируемость при разработке модулей;

У 2.3.5 работать с различными форматами данных и протоколами передачи данных;

У 2.4.3 выполнять тестирование программного обеспечения вручную и автоматизировать процесс тестирования.

Материальное обеспечение:

ПК, серверная среда, IDE, системы мониторинга

Задание:

1. Реализовать учет неудачных попыток входа
2. Внедрить временную блокировку после N попыток
3. Настроить rate limiting по IP
4. Реализовать экспоненциальную задержку
5. Создать систему уведомлений о атаках

Порядок выполнения работы:

1. Создание таблицы failed_login_attempts (IP, user_id, timestamp)
2. Подсчет попыток за временное окно (например, 5 попыток за 15 минут)
3. Блокировка при превышении лимита: Временная (30 минут), Progressive (увеличение времени)
4. Rate limiting на уровне endpoint (Redis/Token Bucket)
5. Экспоненциальная задержка ответа
6. Уведомления (email, logs) при подозрительной активности
7. Whitelist для доверенных IP
8. CAPTCHA после N неудачных попыток

Форма представления результата:

Отчет с кодом защиты, схемой алгоритма блокировки, тестами на brute-force, логами срабатывания защиты

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №113
Безопасная работа с файлами

Цель: освоить методы безопасной загрузки и хранения файлов, научиться валидировать MIME-типы и защищать файловые пути.

Выполнив работу, вы будете уметь:

У 2.2.5 обеспечивать безопасность, производительность и масштабируемость при разработке модулей;

У 2.4.4 анализировать результаты тестирования и документировать найденные ошибки;

У 2.5.7 включать в документацию особенности модулей, такие как ограничения, уязвимости или оптимальные настройки.

Материальное обеспечение:

ПК, веб-фреймворки, IDE, сканеры безопасности

Задание:

1. Реализовать загрузку файлов с валидацией
2. Проверять MIME-типы и содержимое файлов
3. Защититься от directory traversal атак
4. Ограничить размер и тип файлов
5. Безопасно хранить загруженные файлы

Порядок выполнения работы:

1. Валидация расширения файла (белый список)
2. Проверка MIME-type из заголовка и содержимого (magic bytes)
3. Санитизация имени файла: Удаление опасных символов, Генерация уникального имени
4. Защита от path traversal: Проверка на "../"; Использование basename(); Разрешенные директории
5. Ограничение размера файла
6. Хранение файлов вне webroot
7. Сканирование на вирусы (ClamAV)
8. Content-Disposition: attachment для скачивания

Форма представления результата:

Отчет с кодом загрузки, примерами атак (path traversal, malicious files), описанием механизмов защиты

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №114

Реализация безопасной десериализации данных

Цель: научиться безопасно парсить данные, предотвращать уязвимости десериализации и использовать безопасные форматы.

Выполнив работу, вы будете уметь:

У 2.3.1 интегрировать модули и компоненты, обеспечивая их взаимодействие;
У 2.3.2 работать с API и устанавливать соединения между компонентами;
У 2.4.5 разрабатывать стратегии отладки и исправлять ошибки в программном обеспечении.

Материальное обеспечение:

ПК, IDE, библиотеки парсинга, тестовые среды

Задание:

1. Проанализировать уязвимости десериализации
2. Реализовать безопасный парсинг JSON
3. Защититься от XXE в XML
4. Безопасно работать с YAML
5. Внедрить валидацию схемы данных

Порядок выполнения работы:

1. Анализ уязвимостей: Python pickle → RCE; PHP unserialize → object injection; Java deserialization → gadget chains
2. Безопасный JSON: json.loads() вместо eval(); Валидация схемы (JSON Schema)
3. Защита XML от XXE: Отключение внешних сущностей; Использование defuse-библиотек;
4. YAML безопасный парсинг: yaml.safe_load() вместо load(); Ограничение типов
5. Валидация входных данных
6. Сериализация только примитивных типов
7. Подпись данных для проверки целостности

Форма представления результата:

Отчет с примерами уязвимого и безопасного кода, тестами на десериализацию, рекомендациями по форматам

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №115
Настройка CORS политик для веб-приложений

Цель: освоить настройку Cross-Origin Resource Sharing, научиться ограничивать источники и методы запросов.

Выполнив работу, вы будете уметь:

У 2.2.8 улучшать производительность модулей, выявляя и устраняя узкие места;
У 2.2.9 проводить анализ и мониторинг производительности приложений;
У 2.4.4 анализировать результаты тестирования и документировать найденные ошибки.

Материальное обеспечение:

ПК, веб-серверы, IDE, браузеры, инструменты тестирования API

Задание:

1. Реализовать CORS для API
2. Настроить разрешенные источники (origins)
3. Ограничить HTTP методы
4. Контролировать заголовки
5. Реализовать preflight запросы

Порядок выполнения работы:

1. Понимание Same-Origin Policy
2. Настройка CORS заголовков: Access-Control-Allow-Origin; Access-Control-Allow-Methods; Access-Control-Allow-Headers; Access-Control-Allow-Credentials; Access-Control-Max-Age
3. Реализация whitelist доменов
4. Обработка OPTIONS preflight запросов
5. Настройка для разных endpoint: Public API: широкий доступ; Private API: строгие ограничения
6. Тестирование из разных origin
7. Проверка в браузере (DevTools → Network)
8. Избегание unsafe конфигураций (*)

Форма представления результата:

Отчет с конфигурацией CORS, примерами запросов/ответов, таблицей политик для разных endpoint, тестами

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №116
Защита от DDOS атак с помощью rate limiting

Цель: научиться настраивать правила ограничения запросов, использовать балансировщики нагрузки и WAF для защиты от DDoS.

Выполнив работу, вы будете уметь:

У 2.2.5 обеспечивать безопасность, производительность и масштабируемость при разработке модулей;

У 2.4.6 выполнять модульные тесты с использованием инструментов тестирования, в том числе автоматизированного тестирования;

У 2.5.8 проводить регулярное обновление документации при изменении модулей или добавлении нового функционала.

Материальное обеспечение:

ПК, Nginx/HAProxy, системы мониторинга, IDE

Задание:

1. Настроить rate limiting в Nginx
2. Реализовать ограничение по IP и endpoint
3. Настроить балансировщик нагрузки
4. Внедрить базовый WAF правила
5. Мониторить аномальный трафик

Порядок выполнения работы:

1. Nginx rate limiting: limit_req_zone в http блоке; limit_req в location; Алгоритм leaky bucket
2. Ограничение по IP: 10 запросов/секунду; Burst с nodelay
3. Ограничение по endpoint: /api/login: строгие лимиты; /static лимиты
4. HAProxy балансировка: Round-robin алгоритм; Health checks
5. WAF правила: Блокировка SQLi паттернов; Блокировка XSS паттернов; GeoIP фильтрация
6. Мониторинг: Счетчики запросов; Графики трафика; Alerts при аномалиях

Форма представления результата:

Отчет с конфигурацией Nginx/HAProxy, правилами WAF, скриншотами мониторинга, тестами на нагрузку

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №117
Безопасная работа с памятью в приложениях

Цель: освоить методы предотвращения переполнений буфера, научиться управлять указателями и использовать безопасные типы данных.

Выполнив работу, вы будете уметь:

У 2.2.10 применять инструменты для рефакторинга и оптимизации программного кода;
У 2.4.5 разрабатывать стратегии отладки и исправлять ошибки в программном обеспечении;
У 2.5.5 вести журнал изменений и фиксировать обновления программных модулей.

Материальное обеспечение:

ПК, языки программирования (C/C++/Rust), отладчики, анализаторы памяти

Задание:

1.Проанализировать уязвимости переполнения буфера
2. Реализовать безопасную работу с памятью
3. Использовать защищенные функции
4. Применить статический анализ кода
5. Сравнить безопасность C/C++ и Rust

Порядок выполнения работы:

1. Демонстрация уязвимостей: Buffer overflow (strcpy, gets); Use-after-free; Double free; Null pointer dereference
2. Эксплуатация переполнения: перезапись return address; Shellcode injection
3. Защита в C/C++: strncpy вместо strcpy; snprintf вместо sprint; Проверка границ массивов; Умные указатели (unique_ptr, shared_ptr)
4. Инструменты защиты: ASLR, DEP, Stack Canaries; Valgrind для проверки утечек; AddressSanitizer
5. Rust для безопасности: Borrow checker; Ownership система; Отсутствие null pointer; Гарантии безопасности типов
6. Сравнение: уязвимый vs безопасный код

Форма представления результата:

Отчет с примерами уязвимого и безопасного кода, результатами анализа Valgrind/Sanitizer, сравнением языков

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №118
Создание безопасного API с валидацией всех входных данных

Цель: освоить принципы безопасной разработки API, научиться реализовывать комплексную валидацию всех входных данных для предотвращения уязвимостей.

Выполнив работу, вы будете уметь:

У 2.1.9 учитывать требования к масштабируемости, производительности и безопасности при проектировании модуля;

У 2.2.5 обеспечивать безопасность, производительность и масштабируемость при разработке модулей;

У 2.4.8 составлять отчет о выполнении тестирования ПО;

У 2.5.1 описывать функциональность модулей в документации.

Материальное обеспечение:

ПК, веб-фреймворк (Flask/FastAPI/Django), IDE, инструменты тестирования API (Postman), библиотеки валидации (Pydantic, Marshmallow, Cerberus)

Задание:

1. Разработать REST API с минимум 3 endpoint
2. Реализовать валидацию всех входных параметров (query, path, body, headers)
3. Внедрить проверку типов данных, диапазонов, форматов
4. Реализовать санитизацию входных данных
5. Настроить обработку ошибок валидации
6. Создать документацию API с описанием правил валидации

Порядок выполнения работы:

1. Спроектировать REST API с 3 endpoints (POST/GET/PUT /users)
2. Определить правила валидации для каждого поля (тип, длина, диапазон, формат, обязательность)
3. Реализовать валидацию с использованием Pydantic/Marshmallow: Query, path, body параметры; Заголовки запроса
4. Внедрить санитизацию данных (очистка от опасных символов, нормализация, проверка на SQL injection)
5. Настроить обработку ошибок валидации (HTTP 400, детализация ошибок)
6. Протестировать API (позитивные/негативные тесты, граничные значения)
7. Создать документацию OpenAPI/Swagger с описанием правил валидации

Форма представления результата:

Отчет с листингами кода API, схемами валидации (Pydantic/Marshmallow), примерами запросов/ответов, результатами тестирования валидации, документацией OpenAPI

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №119
Создание secure OAuth 2.0 провайдера с PKCE и защитой от атак

Цель: научиться реализовывать OAuth 2.0 авторизацию с использованием PKCE и механизмов защиты от перехвата токенов.

Выполнив работу, вы будете уметь:

У 2.2.4 создавать интерфейсы для взаимодействия с другими модулями и системами;

У 2.2.5 обеспечивать безопасность, производительность и масштабируемость при разработке модулей;

У 2.4.2 создавать тестовые сценарии и тест-кейсы для проверки функциональности и соответствия требованиям;

У 2.4.5 разрабатывать стратегии отладки и исправлять ошибки в программном обеспечении.

Материальное обеспечение:

ПК, OAuth 2.0 библиотека, Postman, IDE

Задание:

1. Реализовать OAuth 2.0 provider
2. Настроить PKCE authentication flow
3. Реализовать генерацию authorization code
4. Настроить access и refresh tokens
5. Выполнить тестирование безопасности

Порядок выполнения работы:

1. Настройка OAuth provider: Authorization endpoint; Token endpoint; Redirect URI
2. PKCE flow: Code verifier; Code challenge; SHA-256 challenge
3. Работа с токенами: Access token; Refresh token; Время жизни токенов
4. Защита OAuth: Проверка redirect URI; Защита от replay attacks; CSRF state parameter
5. Тестирование: Проверка авторизации; Проверка невалидных запросов
6. Документирование: Описание OAuth flow; Скриншоты авторизации

Форма представления результата:

Отчет с конфигурацией OAuth provider, примерами запросов, токенами и результатами тестирования

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа выполнена в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №120
Имплементация JWE (JSON Web Encryption) для защищённых токенов

Цель: научиться реализовывать шифрование токенов с использованием стандарта JWE для безопасной передачи данных.

Выполнив работу, вы будете уметь:

У 2.2.1 разрабатывать модули программного обеспечения с использованием различных языков программирования и технологий;

У 2.2.5 обеспечивать безопасность, производительность и масштабируемость при разработке модулей;

У 2.4.2 создавать тестовые сценарии и тест-кейсы для проверки функциональности и соответствия требованиям;

У 2.5.7 включать в документацию особенности модулей, такие как ограничения, уязвимости или оптимальные настройки.

Материальное обеспечение:

ПК, JWT/JWE библиотеки, OpenSSL, IDE

Задание:

1. Настроить JWE библиотеку
2. Реализовать шифрование payload
3. Реализовать расшифровку токенов
4. Настроить работу с ключами
5. Выполнить тестирование токенов

Порядок выполнения работы:

1. Настройка JWE: Генерация ключей; RSA/AES алгоритмы
2. Создание токенов: Header; Payload; Encryption
3. Расшифровка токенов: Проверка подписи; Декодирование payload
4. Безопасность ключей: Хранение ключей; Rotation ключей
5. Тестирование: Проверка валидности токенов; Проверка ошибок дешифрования
6. Документирование: Описание структуры JWE; Примеры токенов

Форма представления результата:

Отчет с исходным кодом JWE, примерами токенов, результатами тестирования и скриншотами работы приложения

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа выполнена в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №121
Разработка безопасного voting system с homomorphic encryption

Цель: научиться реализовывать безопасную систему электронного голосования с использованием механизмов шифрования и защиты целостности голосов.

Выполнив работу, вы будете уметь:

У 2.2.3 анализировать требования и определять функциональность модуля;

У 2.2.5 обеспечивать безопасность, производительность и масштабируемость при разработке модулей;

У 2.4.3 выполнять тестирование программного обеспечения вручную и автоматизировать процесс тестирования;

У 2.5.1 описывать функциональность модулей в документации.

Материальное обеспечение:

ПК, Python/Node.js, криптографические библиотеки, IDE

Задание:

1. Реализовать систему электронного голосования
2. Настроить шифрование голосов
3. Реализовать защиту от повторного голосования
4. Настроить хранение результатов
5. Выполнить тестирование безопасности системы

Порядок выполнения работы:

1. Создание voting system: Регистрация пользователей; Интерфейс голосования; Хранение голосов
2. Шифрование голосов: Генерация ключей; Шифрование данных; Homomorphic encryption
3. Защита голосования: Проверка уникальности голоса; Защита от replay attacks
4. Подсчет голосов: Расшифровка результатов; Проверка целостности
5. Тестирование: Проверка повторного голосования; Проверка изменения данных
6. Документирование: Описание архитектуры системы; Скриншоты интерфейса

Форма представления результата:

Отчет с исходным кодом voting system, результатами тестирования, схемой работы и скриншотами приложения

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа выполнена в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №122
Реализация secure password manager с client-side encryption

Цель: научиться реализовывать безопасное хранение учетных данных с использованием клиентского шифрования.

Выполнив работу, вы будете уметь:

У 2.2.5 обеспечивать безопасность, производительность и масштабируемость при разработке модулей;

У 2.4.3 выполнять тестирование программного обеспечения вручную и автоматизировать процесс тестирования;

У 2.5.3 программировать с использованием комментариев для документирования кода;

У 2.5.7 включать в документацию особенности модулей, такие как ограничения, уязвимости или оптимальные настройки.

Материальное обеспечение:

ПК, Python/JavaScript, криптографические библиотеки, IDE

Задание:

1. Реализовать password manager
2. Настроить client-side encryption
3. Реализовать мастер-пароль
4. Настроить безопасное хранение данных
5. Выполнить тестирование приложения

Порядок выполнения работы:

1. Создание password manager: Интерфейс приложения; CRUD операции для паролей
2. Client-side encryption: AES шифрование; Генерация ключей; Работа с IV
3. Мастер-пароль: Проверка сложности; Derive key через PBKDF2/Argon2
4. Защита данных: Secure storage; Автоблокировка приложения
5. Тестирование: Проверка дешифрования; Проверка некорректных ключей
6. Документирование: Описание архитектуры; Скриншоты работы приложения

Форма представления результата:

Отчет с исходным кодом password manager, примерами зашифрованных данных, результатами тестирования и скриншотами интерфейса

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа выполнена в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №123
Настройка HSM эмулятора для аппаратной защиты ключей

Цель: научиться использовать HSM-эмулятор для безопасного хранения и обработки криптографических ключей.

Выполнив работу, вы будете уметь:

У 2.4.6 выполнять модульные тесты с использованием инструментов тестирования, в том числе автоматизированного тестирования;

У 2.4.8 составлять отчет о выполнении тестирования ПО;

У 2.5.5 вести журнал изменений и фиксировать обновления программных модулей;

У 2.5.7 включать в документацию особенности модулей, такие как ограничения, уязвимости или оптимальные настройки.

Материальное обеспечение:

ПК, SoftHSM/OpenSC, OpenSSL, IDE

Задание:

1. Установить HSM-эмулятор
2. Создать криптографические ключи
3. Настроить хранение ключей
4. Выполнить операции шифрования
5. Выполнить тестирование HSM

Порядок выполнения работы:

1. Установка HSM: Настройка SoftHSM; Инициализация токена
2. Работа с ключами: Генерация RSA ключей; Экспорт/импорт ключей
3. Настройка доступа: PIN-код; Права доступа
4. Криптографические операции: Подпись данных; Шифрование; Проверка подписи
5. Тестирование: Проверка хранения ключей; Проверка доступа
6. Документирование: Конфигурация HSM; Скриншоты работы

Форма представления результата:

Отчет с конфигурацией HSM, командами настройки, результатами тестирования и скриншотами работы

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа выполнена в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №124
Разработка secure API gateway с JWT verification и rate limiting

Цель: научиться реализовывать безопасный API gateway с проверкой JWT-токенов и ограничением количества запросов.

Выполнив работу, вы будете уметь:

У 2.2.6 оптимизировать проектируемые модули для повышения их эффективности и качества;

У 2.2.5 обеспечивать безопасность, производительность и масштабируемость при разработке модулей;

У 2.4.7 использовать системы контроля дефектов ПО;

У 2.4.8 составлять отчет о выполнении тестирования ПО.

Материальное обеспечение:

ПК, Nginx/Kong/HAProxy, JWT библиотеки, Postman, IDE

Задание:

1. Настроить API gateway
2. Реализовать JWT verification
3. Настроить rate limiting
4. Реализовать журналирование запросов
5. Выполнить тестирование безопасности

Порядок выполнения работы:

1. Настройка gateway: Маршрутизация API; Proxy_pass; Балансировка нагрузки
2. JWT verification: Проверка подписи токена; Проверка expiration time; Валидация claims
3. Rate limiting: Ограничение запросов по IP; Ограничение по endpoint
4. Мониторинг: Логирование запросов; Отслеживание ошибок; Анализ аномального трафика
5. Тестирование: Проверка невалидных токенов; Проверка превышения лимитов
6. Документирование: Описание конфигурации gateway; Скриншоты тестирования

Форма представления результата:

Отчет с конфигурацией API gateway, примерами JWT, результатами тестирования и скриншотами мониторинга

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа выполнена в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №125
Создание hardware-backed key storage для мобильного приложения

Цель: научиться реализовывать защищенное хранение криптографических ключей с использованием аппаратных механизмов мобильных платформ.

Выполнив работу, вы будете уметь:

У 2.2.7 работать с системой контроля версий;

У 2.2.5 обеспечивать безопасность, производительность и масштабируемость при разработке модулей;

У 2.5.4 использовать специальные метки/теги для отметки важных частей кода в документации;

У 2.5.7 включать в документацию особенности модулей, такие как ограничения, уязвимости или оптимальные настройки.

Материальное обеспечение:

ПК, Android Studio/Xcode, Android Keystore или Apple Keychain, IDE

Задание:

1. Настроить secure key storage
2. Реализовать генерацию ключей
3. Настроить хранение ключей
4. Реализовать доступ к ключам
5. Выполнить тестирование безопасности

Порядок выполнения работы:

1. Настройка secure storage: Android Keystore/Apple Keychain; Конфигурация доступа
2. Генерация ключей: RSA/AES ключи; Настройка параметров безопасности
3. Работа с ключами: Шифрование данных; Расшифровка данных
4. Защита ключей: Ограничение экспорта; Аппаратная защита
5. Тестирование: Проверка доступа к ключам; Проверка работы шифрования
6. Документирование: Описание архитектуры хранения; Скриншоты приложения

Форма представления результата:

Отчет с исходным кодом приложения, настройками secure storage, результатами тестирования и скриншотами работы

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа выполнена в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №126
Настройка certificate transparency logs для мониторинга SSL сертификатов

Цель: научиться выполнять мониторинг SSL/TLS сертификатов с использованием certificate transparency logs.

Выполнив работу, вы будете уметь:

У 2.4.4 анализировать результаты тестирования и документировать найденные ошибки;

У 2.4.8 составлять отчет о выполнении тестирования ПО;

У 2.5.6 разбивать модули на логические блоки и описывать каждый блок отдельно;

У 2.5.8 проводить регулярное обновление документации при изменении модулей или добавлении нового функционала.

Материальное обеспечение:

ПК, OpenSSL, crt.sh, браузер, IDE

Задание:

1. Изучить принцип работы CT Logs
2. Настроить мониторинг сертификатов
3. Выполнить анализ SSL сертификатов
4. Проверить наличие подозрительных сертификатов
5. Подготовить отчет

Порядок выполнения работы:

1. Certificate Transparency: Изучение структуры CT Logs; Работа с SCT
2. Мониторинг сертификатов: crt.sh; Проверка сертификатов домена
3. Анализ сертификатов: Issuer; Expiration date; SAN поля
4. Проверка безопасности: Поиск подозрительных сертификатов; Анализ ошибок TLS
5. Тестирование: Проверка валидности сертификатов; Проверка цепочки доверия
6. Документирование: Скриншоты проверки сертификатов; Описание результатов

Форма представления результата:

Отчет с результатами мониторинга сертификатов, скриншотами CT Logs и результатами анализа SSL/TLS

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа выполнена в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №127
Разработка secure session management с защитой от hijacking

Цель: научиться реализовывать безопасное управление пользовательскими сессиями и механизмы защиты от session hijacking.

Выполнив работу, вы будете уметь:

У 2.2.5 обеспечивать безопасность, производительность и масштабируемость при разработке модулей;

У 2.2.8 улучшать производительность модулей, выявляя и устраняя узкие места;

У 2.4.4 анализировать результаты тестирования и документировать найденные ошибки;

У 2.4.5 разрабатывать стратегии отладки и исправлять ошибки в программном обеспечении.

Материальное обеспечение:

ПК, Node.js/PHP/Python, браузер, Postman, IDE

Задание:

1. Реализовать систему пользовательских сессий
2. Настроить secure cookie параметры
3. Реализовать защиту от hijacking
4. Настроить тайм-аут сессий
5. Выполнить тестирование безопасности

Порядок выполнения работы:

1. Настройка session management: Session ID; Session storage; Cookies
2. Secure cookies: HttpOnly; Secure; SameSite параметры
3. Защита сессий: Ротация session ID; Проверка User-Agent/IP; Logout invalidation
4. Тайм-ауты: Idle timeout; Absolute timeout
5. Тестирование: Проверка перехвата cookies; Проверка фиксации сессии
6. Документирование: Скриншоты работы приложения; Описание механизма защиты

Форма представления результата:

Отчет с исходным кодом session management, примерами cookie, результатами тестирования и скриншотами работы приложения

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа выполнена в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.

Лабораторное занятие №128
Создание cryptographically secure RNG (random number generator)

Цель: научиться реализовывать криптографически стойкий генератор случайных чисел для использования в системах безопасности.

Выполнив работу, вы будете уметь:

У 2.2.9 проводить анализ и мониторинг производительности приложений;

У 2.2.10 применять инструменты для рефакторинга и оптимизации программного кода;

У 2.4.5 разрабатывать стратегии отладки и исправлять ошибки в программном обеспечении;

У 2.5.1 описывать функциональность модулей в документации.

Материальное обеспечение:

ПК, Python/Node.js, OpenSSL, IDE

Задание:

1. Реализовать cryptographically secure RNG
2. Настроить генерацию случайных значений
3. Выполнить анализ энтропии
4. Проверить качество генерации
5. Выполнить тестирование RNG

Порядок выполнения работы:

1. Создание RNG: Использование secure random API; Генерация байтов
2. Криптографическая стойкость: Источники энтропии; Seed generation
3. Анализ генерации: Проверка распределения; Проверка повторяемости
4. Производительность: Benchmark генерации; Анализ скорости работы
5. Тестирование: Проверка randomness; Проверка устойчивости к предсказанию
6. Документирование: Описание алгоритма; Скриншоты тестирования

Форма представления результата:

Отчет с исходным кодом RNG, результатами тестирования, анализом энтропии и скриншотами работы приложения

Критерии оценки:

«5» (отлично): выставляется студенту, если лабораторная работа выполнена в полном объеме, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач.

«4» (хорошо): выставляется студенту, если при выполнении задания допущены незначительные ошибки, решение оформлено с соблюдением установленных правил; студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач;

«3» (удовлетворительно): выставляется студенту, если задание выполнено с «грубыми» ошибками, решение оформлено без соблюдения установленных правил;

«2» (неудовлетворительно): выставляется студенту, если работа не выполнена.