1. Шифрование данных

Задание 1.1: Реализуйте функцию в Python, которая шифрует и дешифрует текстовые данные с использованием библиотеки `cryptography`. Примеры использования шифрования AES.

Задание 1.2: Создайте приложение на Flask, которое принимает текстовые данные от пользователя, шифрует их с помощью AES и сохраняет зашифрованные данные в базе данных.

Задание 1.3: Реализуйте код для шифрования и дешифрования данных в JavaScript с использованием Web Crypto API. Продемонстрируйте это на простой веб-странице.

Задание 1.4: Напишите скрипт на Python, который сравнивает зашифрованные и расшифрованные данные, чтобы убедиться, что они совпадают.

2. Кодирование и экранирование данных

Задание 2.1: Напишите функцию на Python, которая кодирует и декодирует строку с использованием Base64.

Задание 2.2: Создайте HTML-страницу с формой ввода. Используйте JavaScript для экранирования пользовательского ввода перед его отправкой на сервер.

Задание 2.3: Реализуйте сервер на Flask, который принимает данные в формате JSON и экранирует специальным символы перед сохранением в базе данных.

Задание 2.4: Напишите JavaScript-функцию, которая преобразует пользовательский ввод в HTML-сущности, чтобы предотвратить XSS-атаки.

3. Сторонние компоненты

Задание 3.1: Проанализируйте список сторонних библиотек в проекте. Опишите, как вы будете проверять их безопасность и актуальность.

Задание 3.2: Создайте пример веб-приложения на Flask, использующего стороннюю библиотеку для обработки форм. Обеспечьте обновление этой библиотеки и следите за уязвимостями.

Задание 3.3: Напишите скрипт, который сканирует ваш проект на наличие устаревших или небезопасных сторонних компонентов.

Задание 3.4: Реализуйте простой механизм уведомлений в вашем приложении, который предупреждает о доступных обновлениях сторонних библиотек.

4. Заголовки безопасности

Задание 4.1: Настройте заголовок `Content Security Policy` в приложении Flask для защиты от XSS-атак.

Задание 4.2: Добавьте заголовок `X-Content-Type-Options` к ответам вашего сервера для предотвращения MIME-sniffing.

Задание 4.3: Реализуйте на HTML-странице использование заголовков безопасности, таких как `Strict-Transport-Security` и `X-Frame-Options`.

Задание 4.4: Напишите скрипт на Python для установки нескольких заголовков безопасности, включая `X-XSS-Protection` и `Referrer-Policy`, и протестируйте их с использованием инструмента для анализа HTTP-запросов.

5. Безопасность файлов cookies

Задание 5.1: Реализуйте на сервере Flask установку и настройку cookies с атрибутами `HttpOnly` и `Secure`.

Задание 5.2: Напишите JavaScript-код, который проверяет наличие и безопасность cookies на стороне клиента.

Задание 5.3: Создайте пример веб-приложения, в котором cookies используются для хранения пользовательских настроек. Обеспечьте их защиту от атак.

Задание 5.4: Настройте механизм очистки cookies после выхода пользователя из системы на сервере Flask.

6. Пароли и хранилище

Задание 6.1: Реализуйте функцию на Python для хеширования паролей с использованием библиотеки `bcrypt` и сохраните их в базе данных.

Задание 6.2: Напишите скрипт на Flask для проверки паролей пользователей и обеспечения их безопасности при хранении.

Задание 6.3: Создайте пример приложения на JavaScript, которое показывает, как безопасно хранить пароли с помощью Web Storage API и обеспечивать их защиту.

Задание 6.4: Реализуйте механизм для автоматического обновления паролей и их хешей в базе данных.

7. Загрузка файлов

Задание 7.1: Создайте сервер на Flask, который безопасно обрабатывает загрузку файлов, проверяя расширение и размер файла.

Задание 7.2: Напишите клиентский код на HTML и JavaScript для загрузки файлов на сервер и обработки ошибок при загрузке.

Задание 7.3: Реализуйте функцию в Python для проверки содержимого загруженных файлов на наличие потенциально вредоносного кода.

Задание 7.4: Создайте пример веб-приложения, в котором загружаемые файлы сохраняются в отдельной защищенной директории.

8. Ошибки и исключения

Задание 8.1: Реализуйте обработку исключений на сервере Flask и настройте логирование ошибок в файл.

Задание 8.2: Напишите код на JavaScript для обработки и отображения ошибок на клиентской стороне без раскрытия конфиденциальной информации.

Задание 8.3: Создайте механизм для отображения пользовательских сообщений об ошибках и логирования технических ошибок в веб-приложении.

Задание 8.4: Реализуйте стратегию обработки ошибок в приложении на Python, которая предотвращает утечку информации о внутренней структуре системы.

9. Проверка и санитизация данных

Задание 9.1: Напишите функцию на Python, которая проверяет и санитизирует входные данные от пользователя, предотвращая SQL-инъекции.

Задание 9.2: Реализуйте код на JavaScript для проверки форм ввода на наличие вредоносных данных и их санитизации перед отправкой на сервер.

Задание 9.3: Создайте сервер на Flask, который использует библиотеки для проверки и очистки вводимых данных от HTML-тегов и спецсимволов.

Задание 9.4: Напишите пример использования регулярных выражений для проверки корректности ввода данных на стороне клиента и сервера.

10. Авторизация и аутентификация

Задание 10.1: Реализуйте аутентификацию и авторизацию с использованием JWT в приложении на Flask. Пример кода для создания и проверки токенов.

Задание 10.2: Напишите клиентский код на JavaScript для отправки запросов с JWT в заголовках и обработки ответов сервера.

Задание 10.3: Создайте пример простого веб-приложения с формой логина, которая аутентифицирует пользователей и предоставляет доступ к защищенным маршрутам на основе их ролей.

Задание 10.4: Реализуйте функцию в Python для создания и проверки сессионных токенов для аутентификации пользователей на сервере.

11. Принцип наименьших привилегий

Задание 11.1: Реализуйте в приложении на Flask разграничение доступа на основе ролей пользователей. Определите права доступа для обычных пользователей и администраторов.

Задание 11.2: Создайте пример настройки прав доступа на уровне файловой системы для различных пользователей и групп.

Задание 11.3: Напишите скрипт для проверки и минимизации прав доступа процессов и учетных записей в вашем серверном окружении.

Задание 11.4: Реализуйте механизм для предоставления минимально необходимых привилегий для выполнения задач в вашем веб-приложении.