VAPT REPORT

Table of Contents

1. **Project Overview**
2. **Scope of Work**
3. **Objectives**
4. **Executive Summary**
5. **Tools Used**
6. **Methodology**
    - Reconnaissance
    - Vulnerability Identification
    - Exploitation
    - Post-Exploitation
    - Reporting
7. **Findings & Exploited Vulnerabilities**
    - OS Command Injection
    - SQL Injection
    - Cross-Site Scripting (XSS)
    - Cross-Site Request Forgery (CSRF)
    - Burp Suite Intercepts & Responses
8. **Screenshots & Evidence Placeholders**
9. **Remediation Recommendations**
10. **Lessons Learned**
11. **Conclusion**

1. Project Overview

This capstone project simulates a **real-world web application penetration test** by targeting and exploiting vulnerabilities in **bWAPP (Buggy Web Application)**. The application is deliberately insecure and widely used for security training and research.

The engagement focused on identifying, exploiting, and documenting multiple vulnerabilities while using industry-standard penetration testing practices.

2. Scope of Work

- **Target**: bWAPP hosted in a controlled lab environment on Kali Linux.

- **Exclusions**: Only the bWAPP application was tested. No external systems, networks, or production applications were in scope.

- **Testing Window**: Lab-based (simulated timeline).

3. Objectives

- Set up and configure bWAPP in Kali Linux.
- Perform vulnerability assessment and penetration testing using manual techniques and automated tools.
- Exploit four common vulnerabilities: **OS Command Injection, SQL Injection, Cross-Site Scripting, and CSRF**.
- Capture HTTP request/response traffic using **Burp Suite**.
- Prepare and present a **professional penetration testing report**.
- Demonstrate understanding of **OWASP Top 10 vulnerabilities**.

4. Executive Summary

The assessment successfully demonstrated how an attacker can exploit insecure coding practices in bWAPP.

Four vulnerabilities were identified and exploited:

- **OS Command Injection** – Allowed execution of arbitrary system commands.
- **SQL Injection** – Enabled retrieval of sensitive database information.
- **Cross-Site Scripting (XSS)** – Allowed injection of malicious JavaScript into user sessions.
- **CSRF** – Enabled unauthorized actions without proper user verification.

The findings highlight the importance of secure coding practices, input validation, and proper use of security controls.

Risk Rating Summary:

- **Critical**: SQL Injection, OS Command Injection
- **High**: CSRF
- **Medium**: XSS

5. Tools Used

- **Kali Linux** (penetration testing OS)
- **bWAPP Application** (vulnerable web app hosted with docker)
- **Burp Suite Community Edition** (proxy, interception, and manipulation)
- **Web Browser (Firefox/Chrome)** configured to route traffic through Burp Suite
- **Command Line Utilities** (cURL, netcat, etc. for testing payloads)

6. Methodology

6.1 Reconnaissance

Initial exploration of the application to identify available features and attack surfaces.

6.2 Vulnerability Identification

Systematic testing for OWASP Top 10 vulnerabilities using manual probing and Burp Suite.

6.3 Exploitation

Crafted and executed payloads for OS command injection, SQL injection, XSS, and CSRF.

6.4 Post-Exploitation

Verified successful exploitation by extracting sensitive data, executing arbitrary commands, and demonstrating real impacts.

6.5 Reporting

All findings were documented, with evidence captured via screenshots and Burp Suite intercepts.

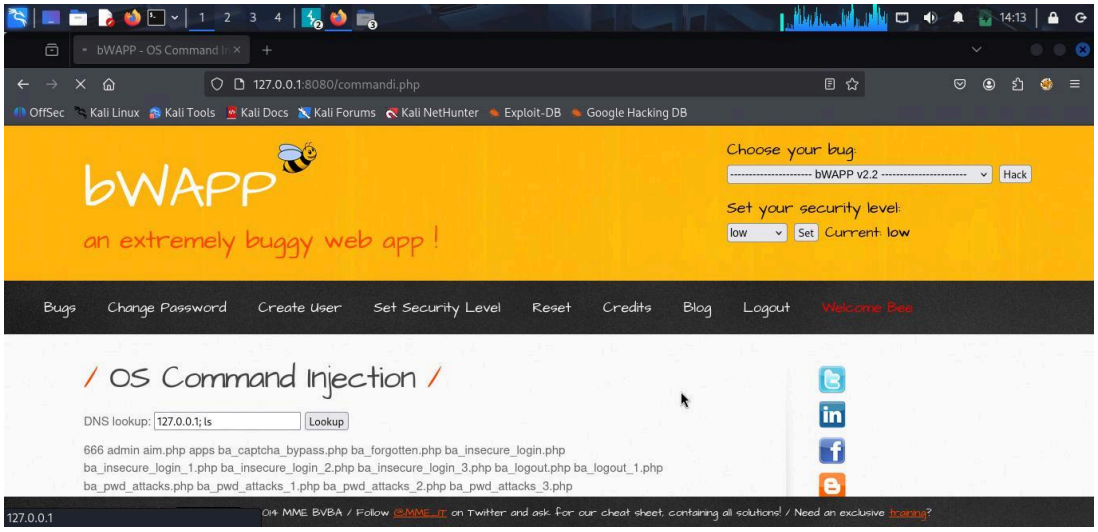
Findings & Exploited Vulnerabilities

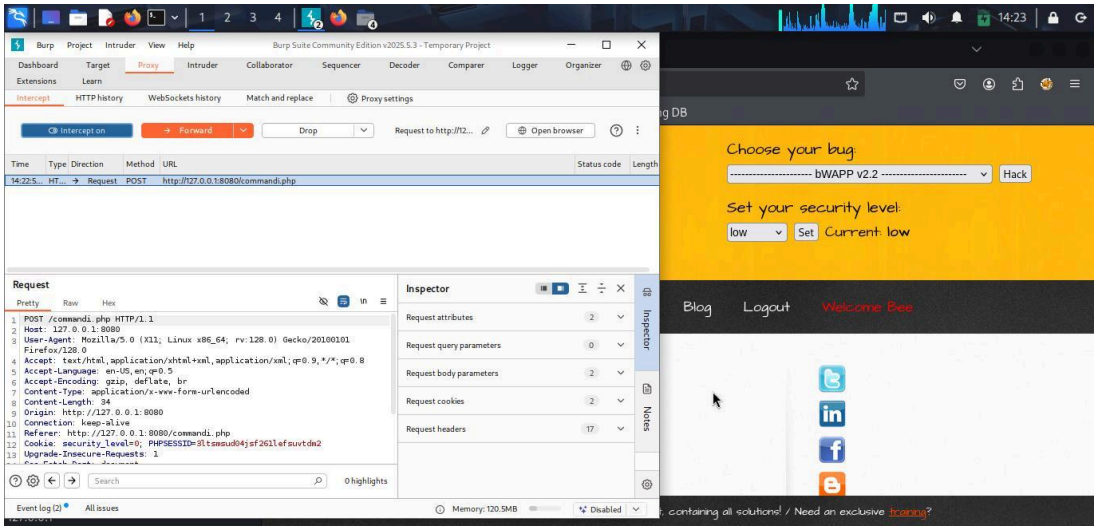
OS Command Injection

- **Description**: User input was directly concatenated into system commands.

- **Impact**: Attackers can execute arbitrary commands, gaining full system control.

- **Evidence**:



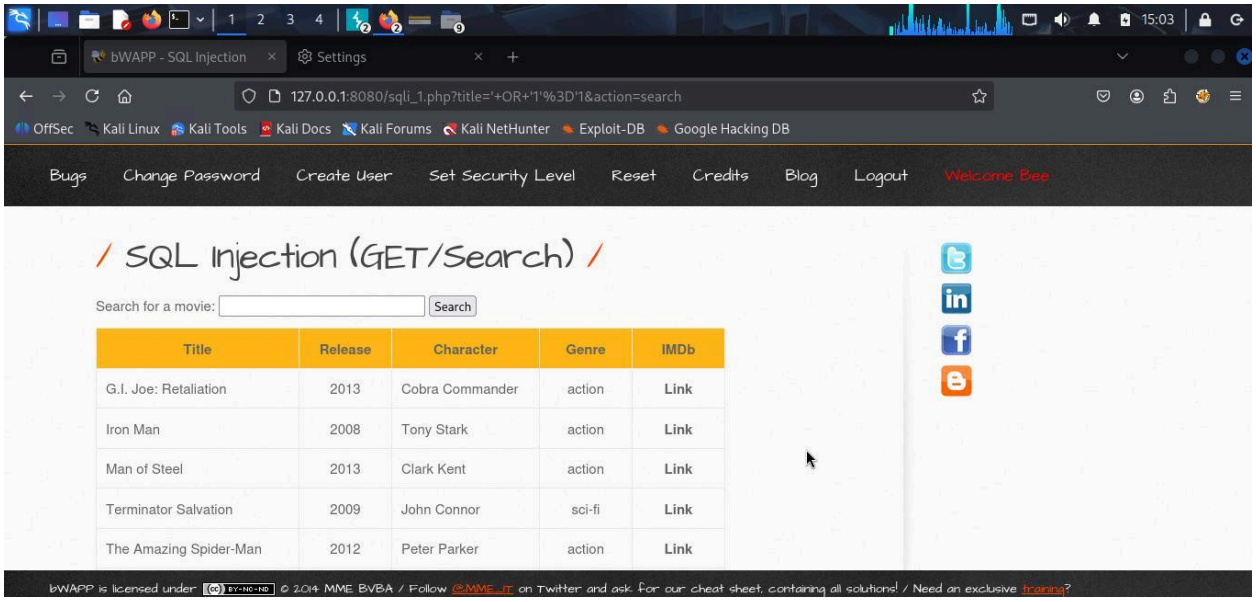

Burp Project Intruder View Help
Burp Suite Community Edition v2025.5.3 - Temporary Project
Dashboard Target Proxy Intruder Collaborator Sequencer Decoder Comparer Logger Organizer
Extensions Learn
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on Forward Drop Open browser
Time Type Direction Method URL Status code Length
14:22:5... HT... → Request POST http://127.0.0.1:8080/commandi.php
Request
Pretty Raw Hex
POST /commandi.php HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 34
Origin: http://127.0.0.1:8080
Connection: keep-alive
Referer: http://127.0.0.1:8080/commandi.php
Cookie: security_level=0; PHPSESSID=3ltsmsud04jsf261lefsuvtdm2
Upgrade-Insecure-Requests: 1
Search 0 highlights
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 2
Request cookies 2
Request headers 17
Inspector Notes
Event log (2) All issues Memory: 120.5MB Disabled
14:23
Choose your bug:
---------- bWAPP v2.2 ---------- Hack
Set your security level:
low Set Current: low
Blog Logout Welcome Bee
containing all solutions! / Need an exclusive training?

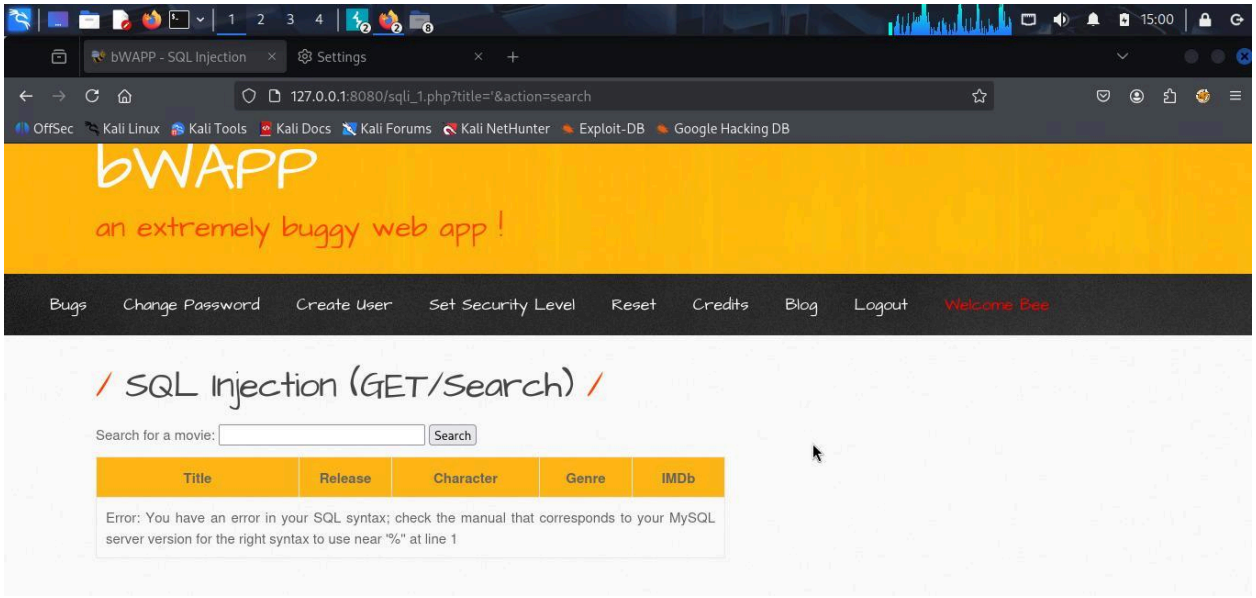
SQL Injection

- **Description**: Input fields failed to sanitize user input, allowing SQL queries to be manipulated.

- **Impact**: Unauthorized database access, sensitive data disclosure.

- **Evidence**:

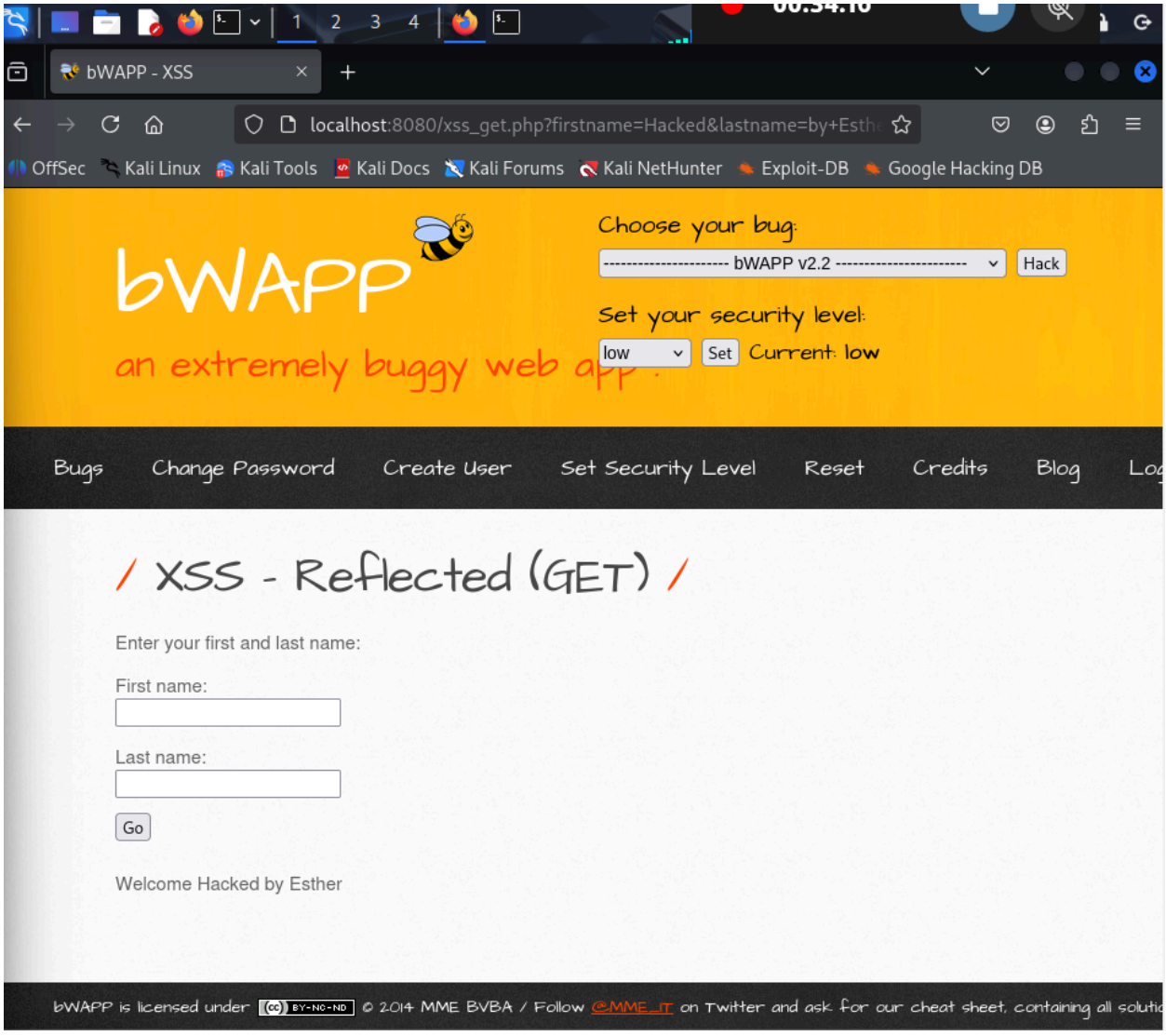



15:00
bWAPP - SQL Injection
Settings
127.0.0.1:8080/sqli_1.php?title='&action=search
OffSec
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
bWAPP
an extremely buggy web app !
Bugs
Change Password
Create User
Set Security Level
Reset
Credits
Blog
Logout
Welcome Bee
/ SQL Injection (GET/Search) /
Search for a movie:
Search
Title
Release
Character
Genre
IMDb
Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '%'' at line 1

Cross-Site Scripting (XSS)

- **Description**: Application accepted and reflected malicious JavaScript without sanitization.

- **Impact**: Session hijacking, credential theft, phishing attacks.
- **Evidence**:

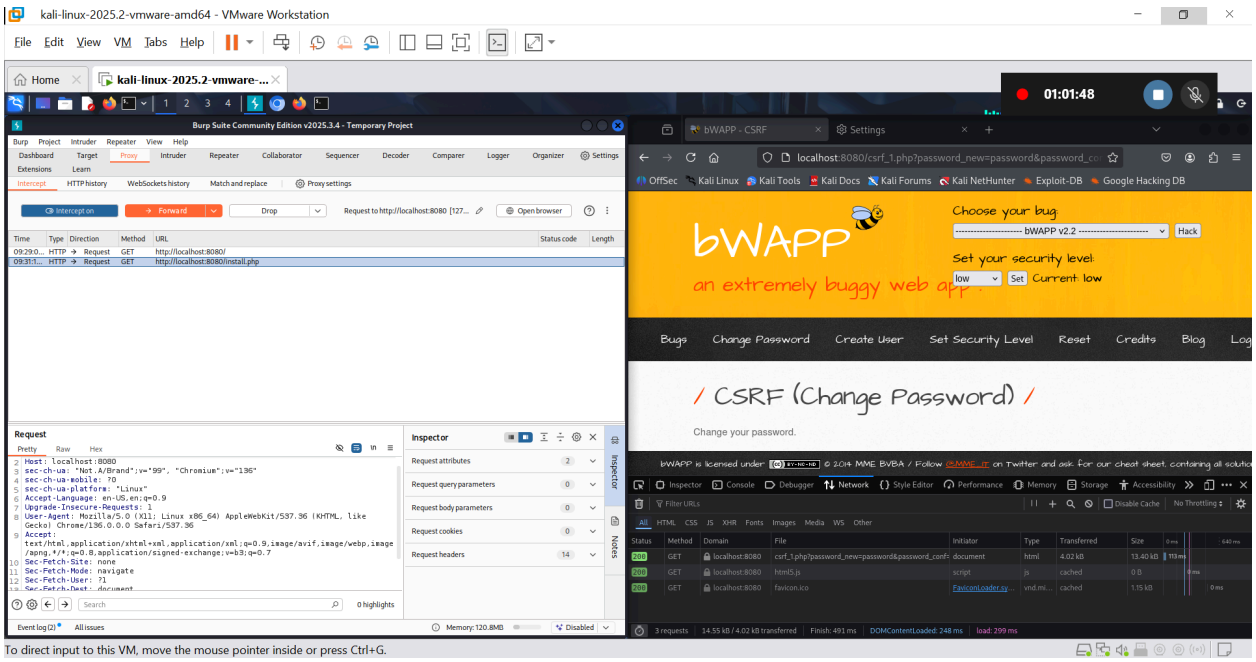


Cross-Site Request Forgery (CSRF)

- **Description**: Absence of anti-CSRF tokens allowed attackers to perform unauthorized actions.

- **Impact**: Passwords changed without user consent.

- **Evidence**:

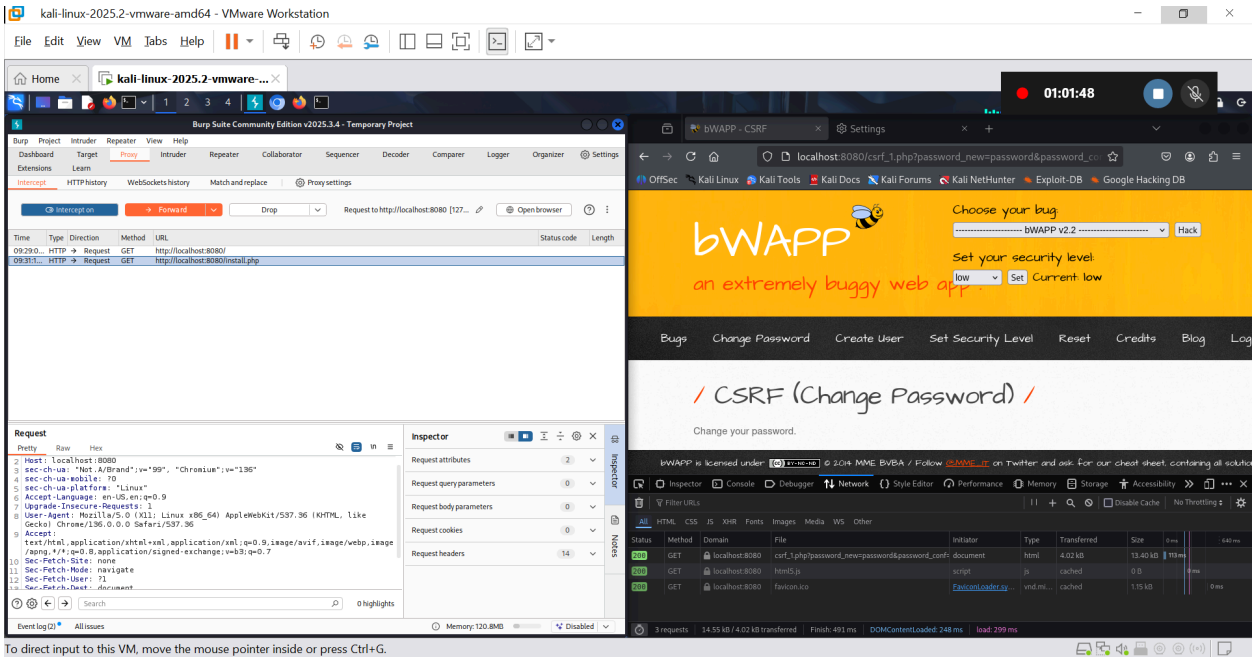


Burp Suite Intercepts & Responses

- Captured request/response pairs showed vulnerable parameters being exploited.

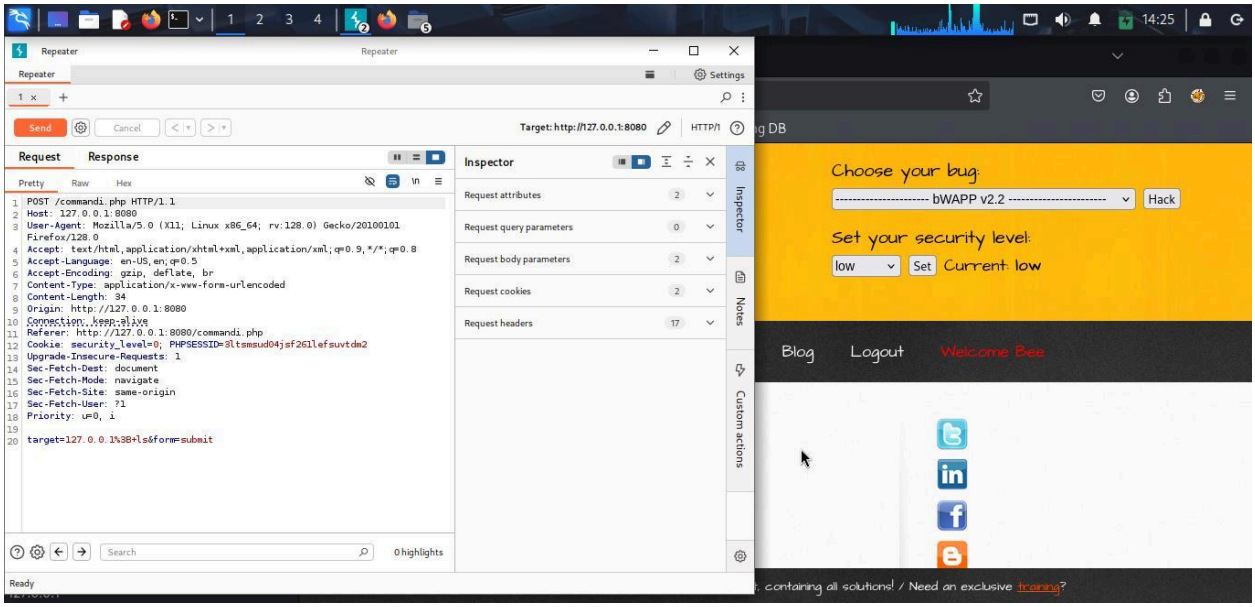
- Examples include POST requests with malicious payloads in **SQL Injection** and **CSRF attacks**.

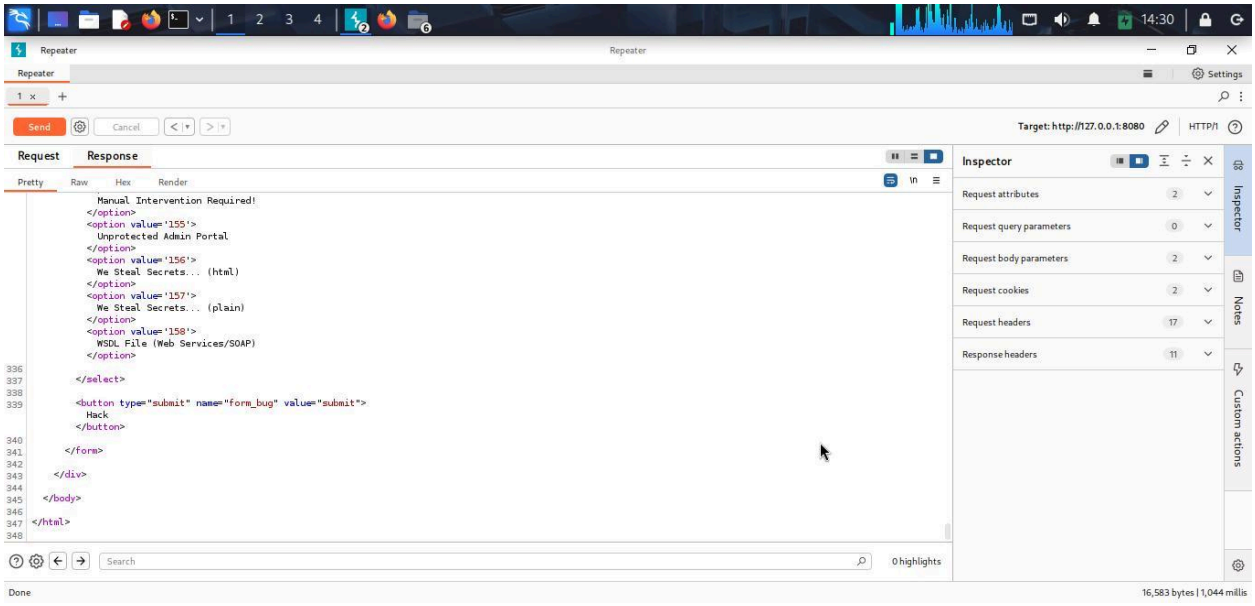
- **Evidence**:



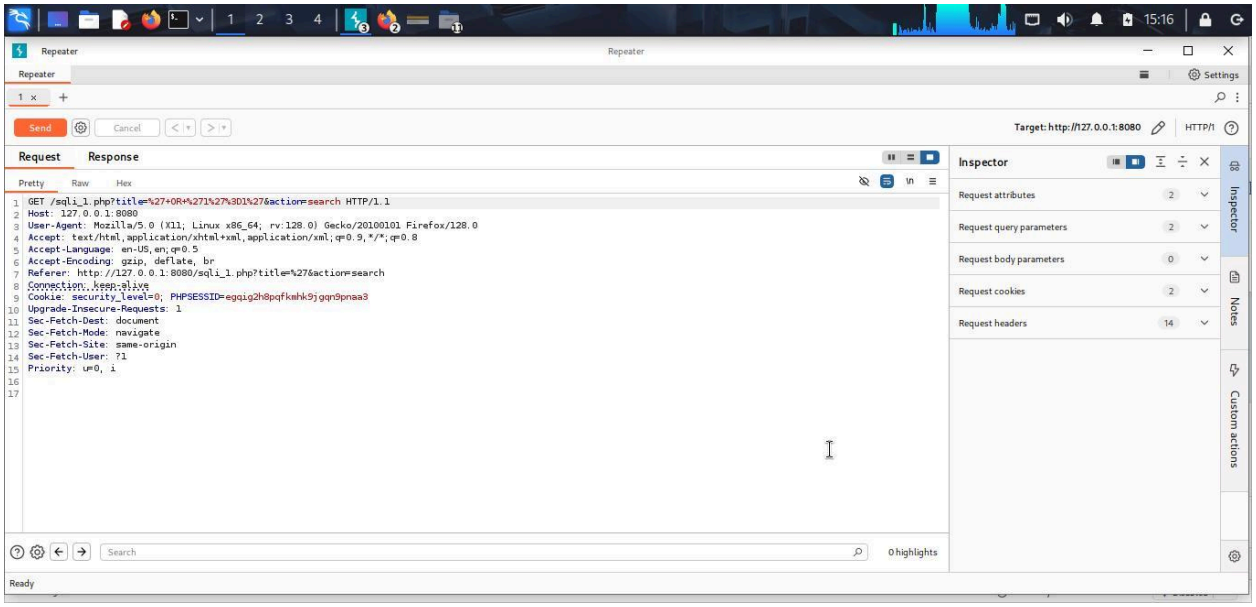
Screenshots & Evidence (Burpsuite)

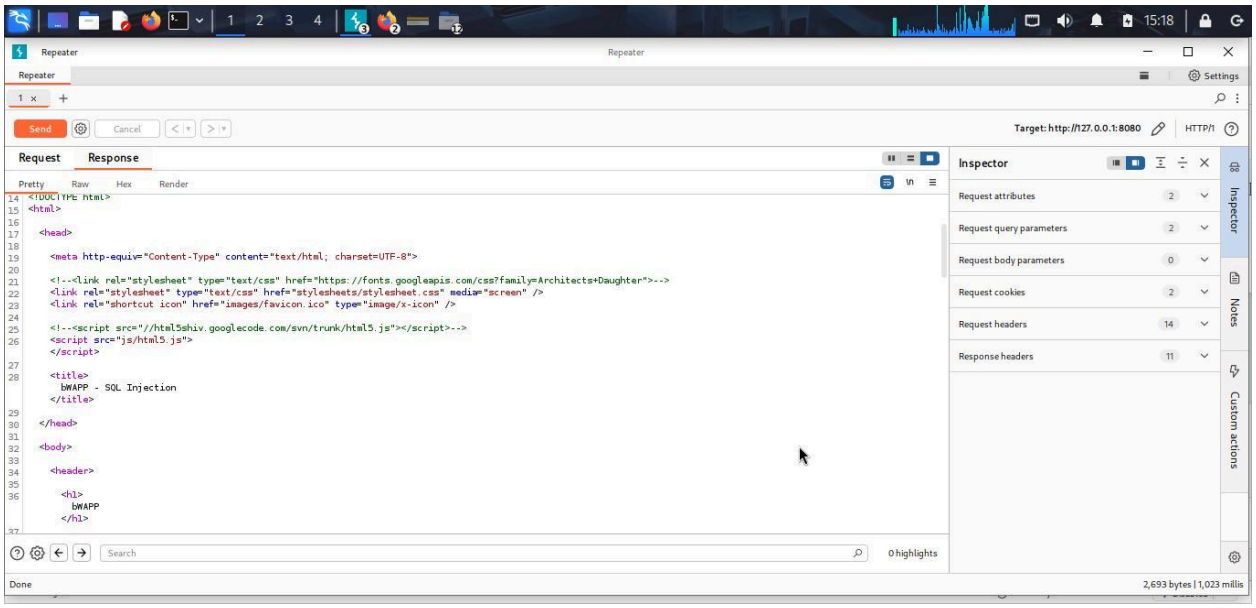
- ***Figure 1**: OS Command Injection output (request/response)*



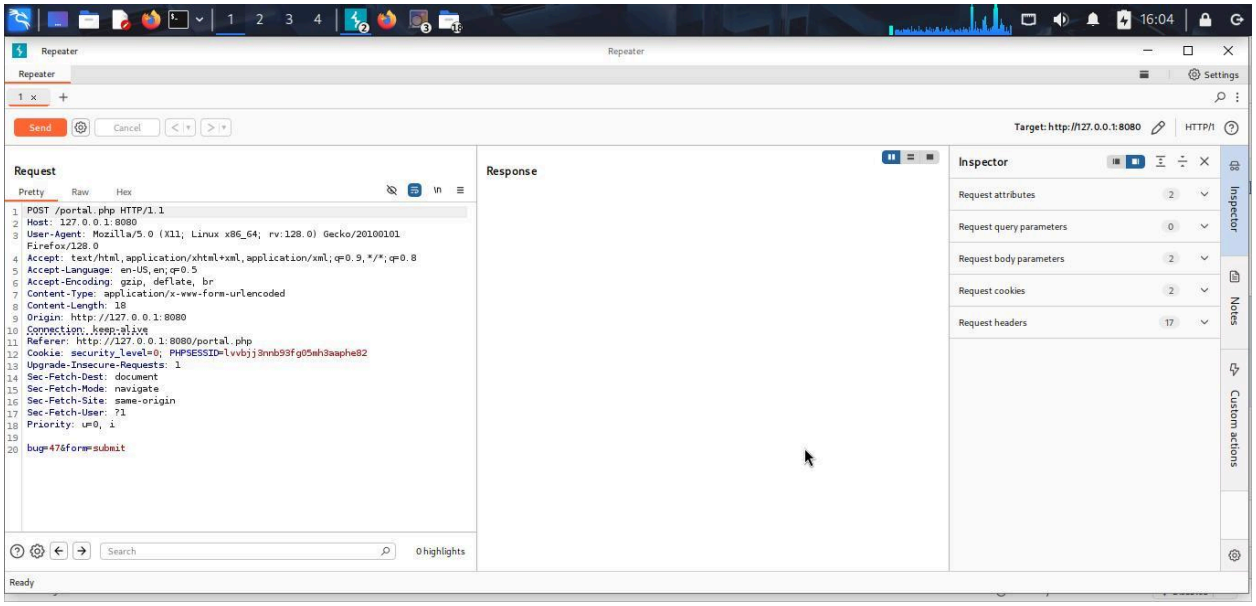


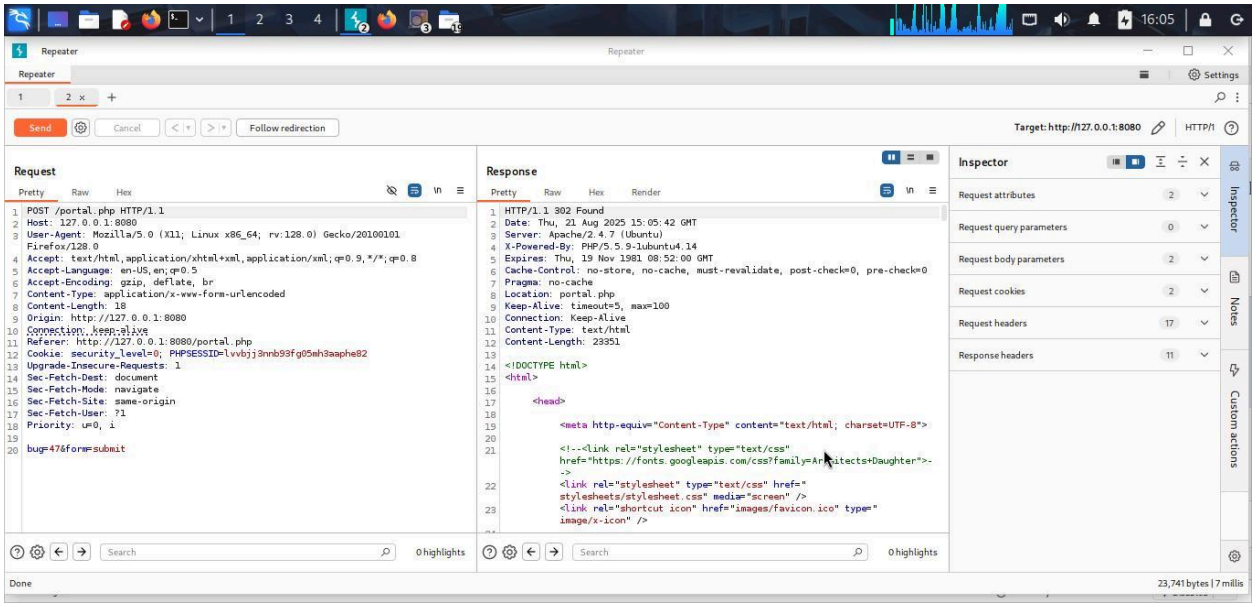
- ***Figure 2**: SQL Injection output (request/response)*



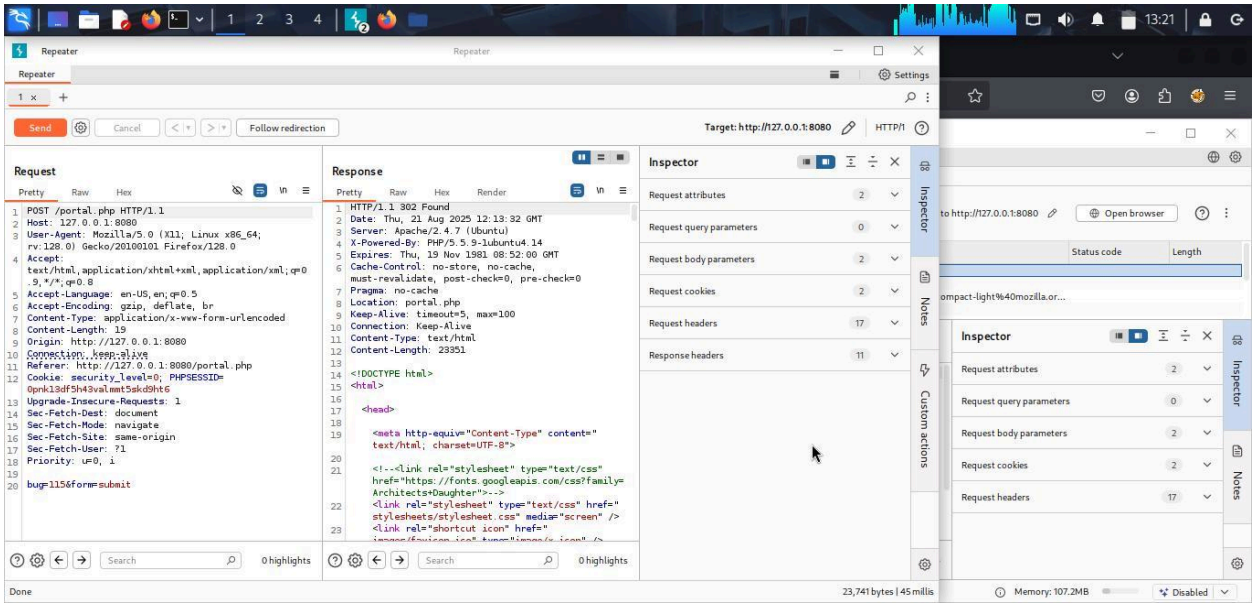


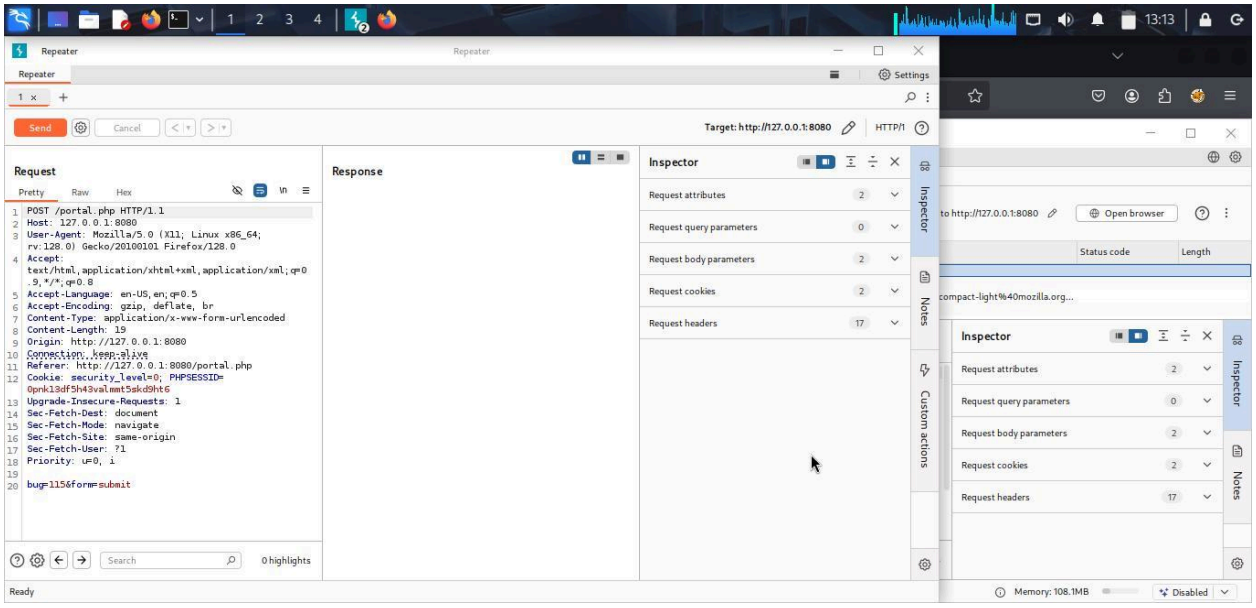
- ***Figure 3**: XSS payload execution output(request/response)*


Repeater
Send
Cancel
Target: http://127.0.0.1:8080
HTTP/1
Request
Pretty
Raw
Hex
POST /portal.php HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 18
Origin: http://127.0.0.1:8080
Connection: keep-alive
Referer: http://127.0.0.1:8080/portal.php
Cookie: security_level=0; PHPSESSID=lvvbjj3nnb93fg05mh3aaphe82
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
bug=47&form=submit
Response
Inspector
Request attributes
Request query parameters
Request body parameters
Request cookies
Request headers
Search
0 highlights
Ready

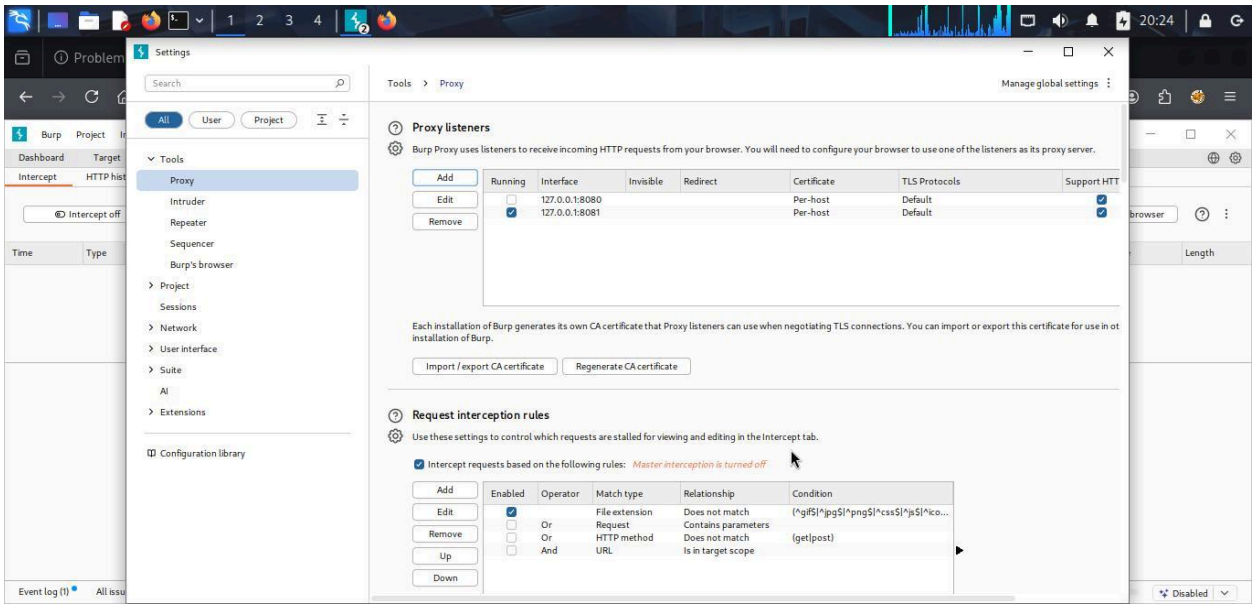

Repeater
Send
Cancel
Follow redirection
Target: http://127.0.0.1:8080
HTTP/1
Request
Pretty
Raw
Hex
POST /portal.php HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 18
Origin: http://127.0.0.1:8080
Connection: keep-alive
Referer: http://127.0.0.1:8080/portal.php
Cookie: security_level=0; PHPSESSID=lvvbjj3nnb93fg05mh3aaphe82
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
bug=47&form=submit
Response
Pretty
Raw
Hex
Render
HTTP/1.1 302 Found
Date: Thu, 21 Aug 2025 15:05:42 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Location: portal.php
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
Content-Length: 23351
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<!--<link rel="stylesheet" type="text/css" href="https://fonts.googleapis.com/css?family=Architects+Daughter">-->
<link rel="stylesheet" type="text/css" href="stylesheets/stylesheet.css" media="screen" />
<link rel="shortcut icon" href="images/favicon.ico" type="image/x-icon" />
Inspector
Request attributes
Request query parameters
Request body parameters
Request cookies
Request headers
Response headers
Search
0 highlights
Done
23,741 bytes | 7 millis

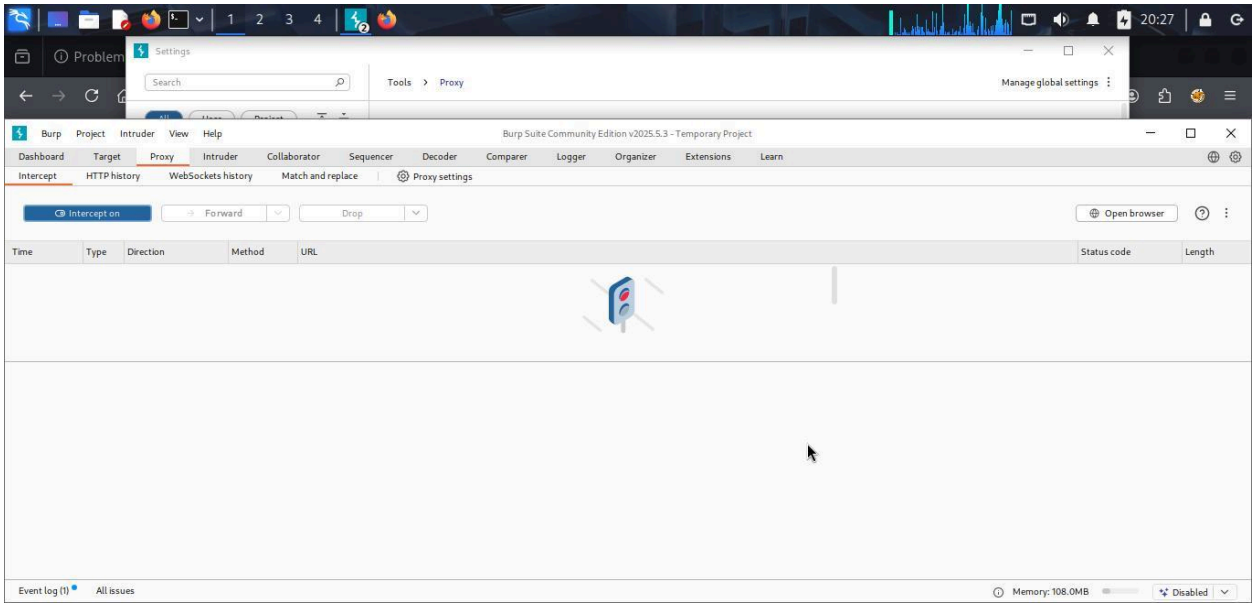
- ***Figure 4**: CSRF exploited via Burp Suite intercept (request/response)*





- ***Figure 5**: Burp Suite request/response*





9. Remediation Recommendations

- Implement strict **input validation and sanitization**.
- Use **parameterized queries (prepared statements)** to prevent SQL injection.
- Employ **CSRF tokens** in sensitive operations.
- Apply **output encoding** to mitigate XSS.
- Restrict unnecessary system command execution.
- Regularly test applications against **OWASP Top 10 vulnerabilities**.

10. Lessons Learned

- The importance of secure coding practices cannot be overstated.
- Burp Suite is a powerful tool for discovering and exploiting vulnerabilities.
- Many real-world breaches stem from simple, preventable flaws.
- Thorough documentation strengthens the value of a penetration test.

11. Conclusion

This project successfully simulated a penetration testing engagement on a vulnerable application. Multiple high-risk vulnerabilities were identified and exploited, demonstrating the severe impact of insecure development practices.

By applying the remediation steps recommended, organizations can significantly improve their security posture and protect against real-world attacks.

Note: For details on the step by step procedure,

Please visit the link: VAPT pentest project