Web Application Vulnerability Testing

Introduction

Web application vulnerability testing is the process of identifying security weaknesses in web applications that can be exploited by attackers. This practical focuses on understanding common OWASP Top 10 vulnerabilities using a deliberately vulnerable application. Tools like DVWA and Burp Suite are used to analyze, exploit, and observe insecure application behavior. The goal is to gain hands-on experience in web application security testing and mitigation.

—

Tools Used

- Kali Linux
- Burp Suite Community Edition
- Damn Vulnerable Web Application (DVWA)
- Mozilla Firefox

—

OWASP Top 10

OWASP Top 10 represents the most critical web application security risks identified by the Open Web Application Security Project (OWASP). These vulnerabilities highlight common weaknesses in modern web applications that attackers frequently exploit.

Some common OWASP Top 10 vulnerabilities include:

- SQL Injection (SQLi)
- Cross-Site Scripting (XSS)
- Broken Authentication
- Security Misconfiguration
- Sensitive Data Exposure

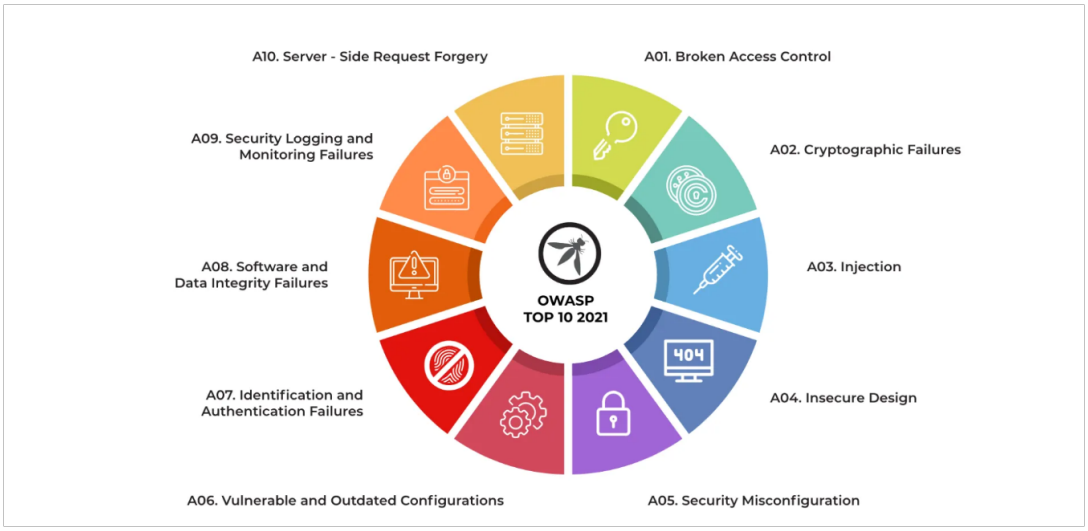

Figure 1: OWASP Top 10 Web Application Security Risks

Step 1: Setup Vulnerable Application (DVWA)

DVWA is installed and configured on Kali Linux using Apache and PHP. The application runs locally on the system.

URL: `http://localhost/dvwa`

Figure 2: DVWA Setup Page on Kali Linux

Step 2: Login to DVWA

The default credentials are used to log in to the DVWA dashboard.

Username: admin
Password: password

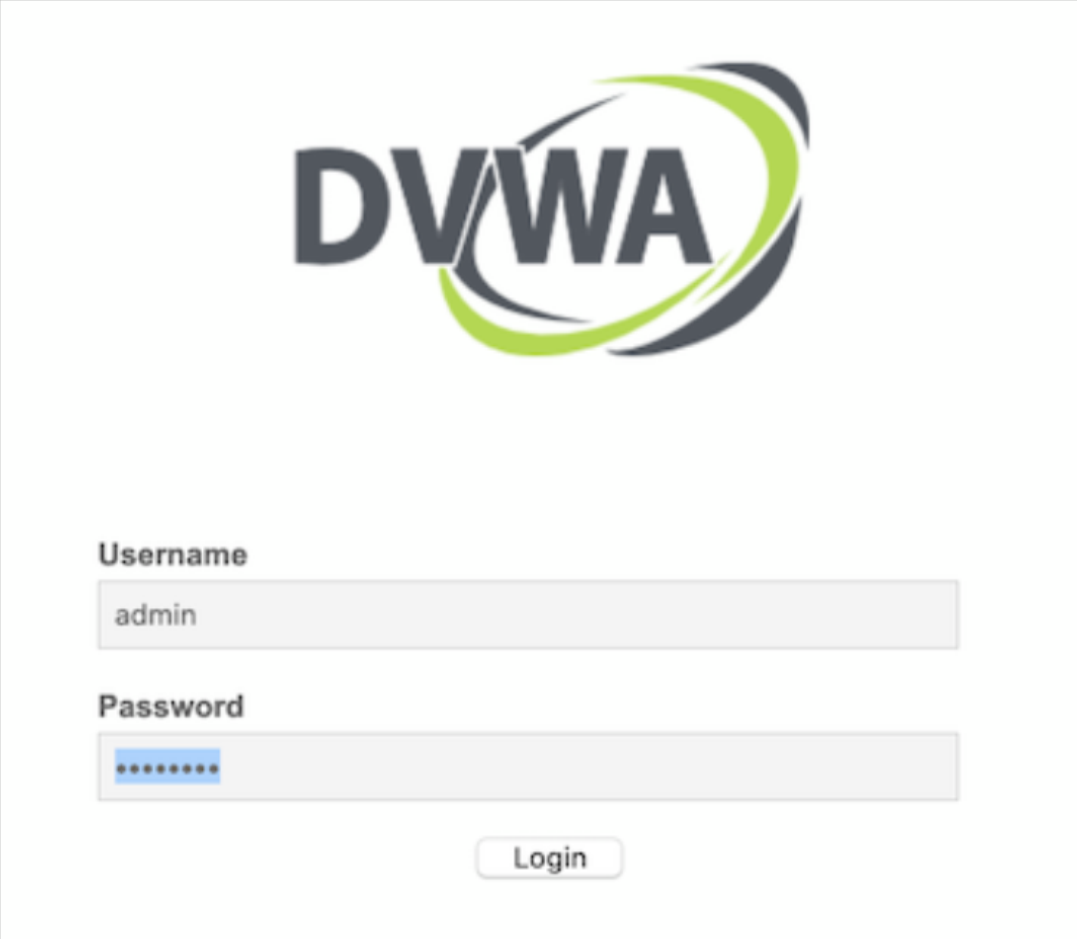

Figure 3: DVWA Login Page

—

Step 3: Configure Burp Suite Proxy

Burp Suite Community Edition is launched and the browser proxy is configured to intercept HTTP requests.

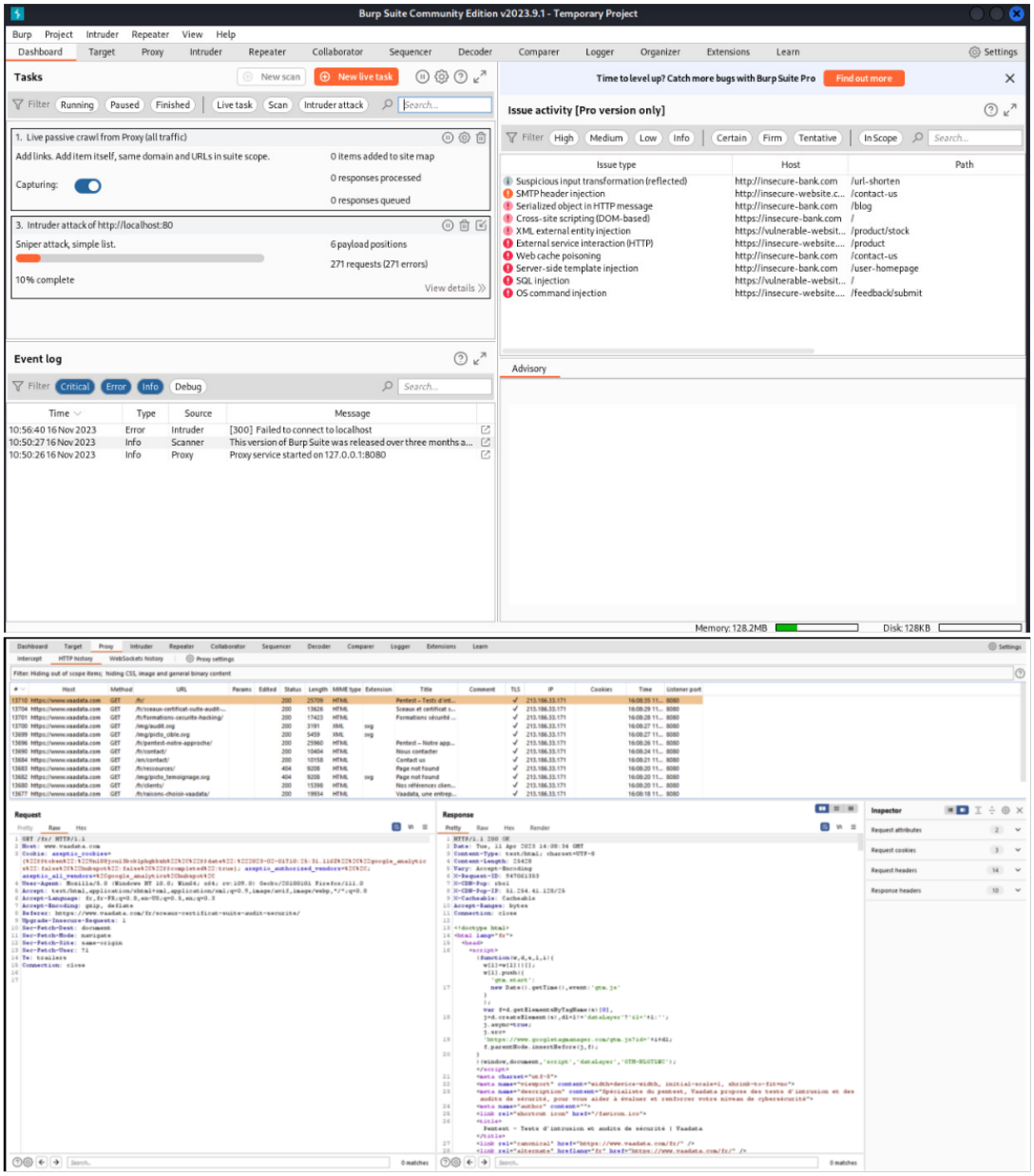

Figure 4: Burp Suite Proxy Intercept Enabled

Step 4: SQL Injection Testing

SQL Injection is tested using the DVWA SQL Injection module by inserting malicious SQL queries.

Sample Payload:

```
' OR '1'='1
```

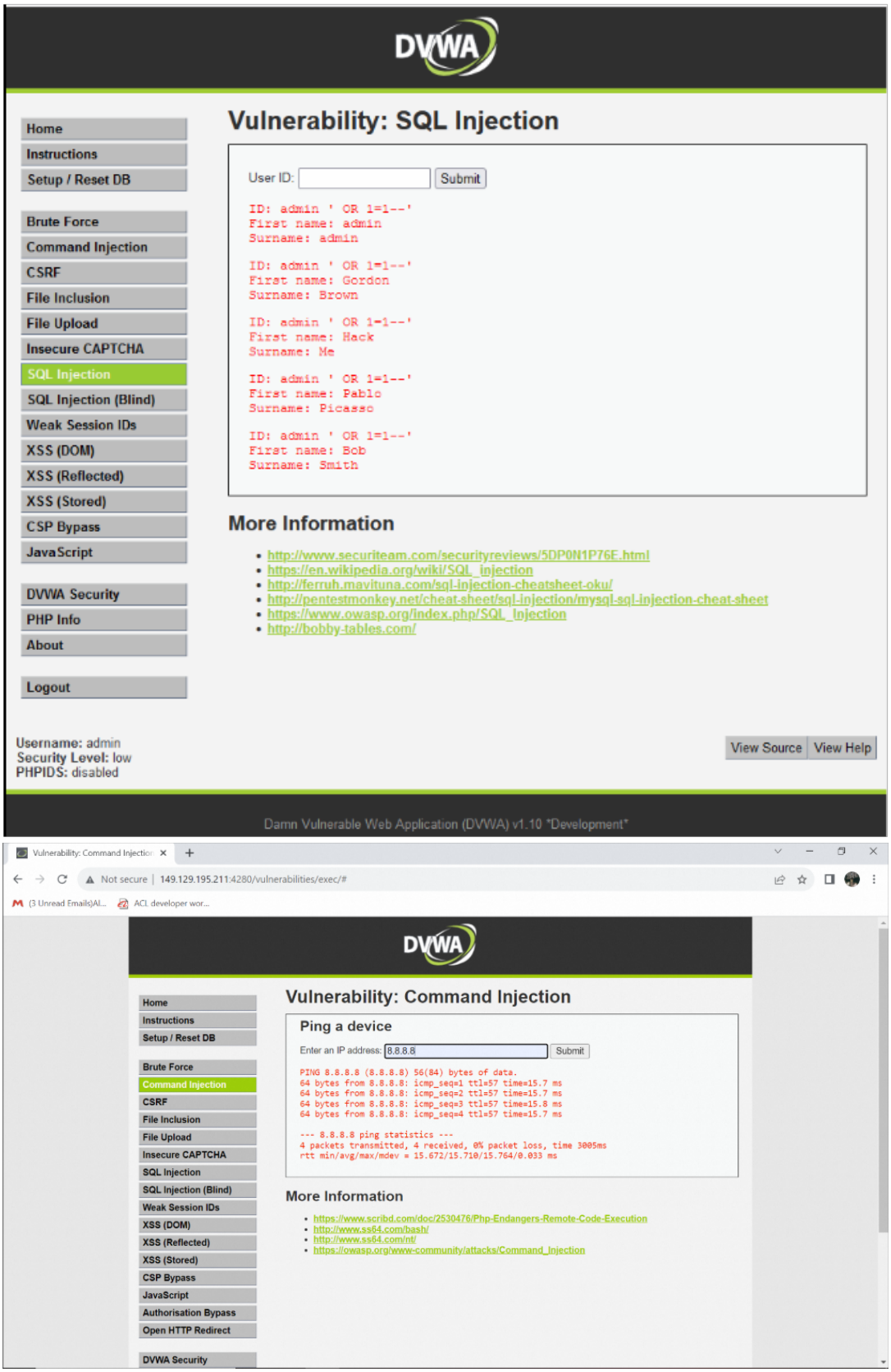

Figure 5: SQL Injection Vulnerability in DVWA

Step 5: Cross-Site Scripting (XSS) Testing

XSS vulnerability is tested by injecting JavaScript code into user input fields.

Sample Payload:

```
<script>alert('XSS')</script>
```

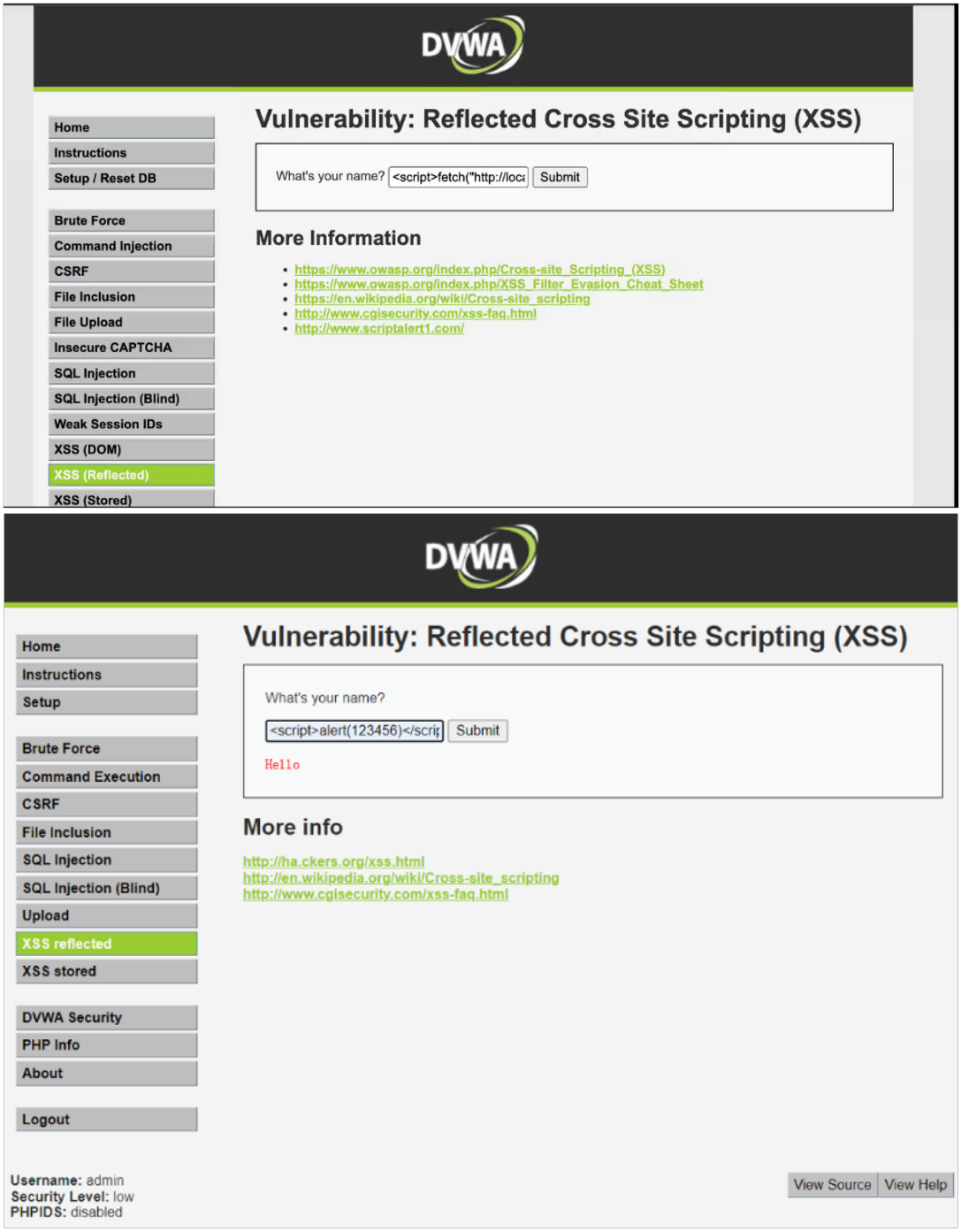

Figure 6: XSS Alert Box Triggered in Browser

Step 6: Observe Application Responses

The server responses are analyzed using Burp Suite to identify insecure behaviors and data leakage.

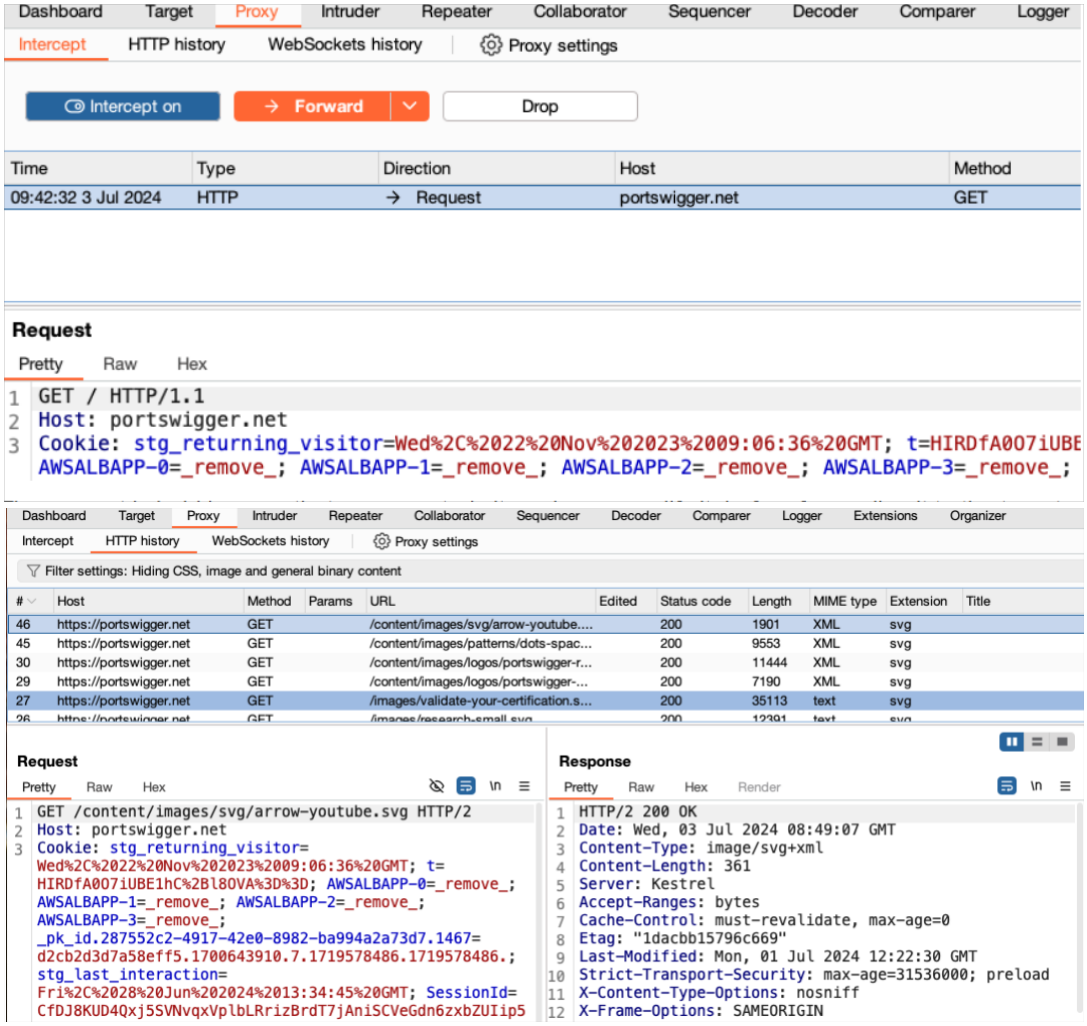

Figure 7: HTTP Response Analysis in Burp Suite

—

Step 7: Documentation of Vulnerabilities

Vulnerability	Payload Used	Impact
SQL Injection	' OR '1'='1	Unauthorized database access
Cross-Site Scripting	¡script¿alert()¡/script¿	Execution of malicious scripts

—

Step 8: Suggested Mitigations

- Use prepared statements and parameterized queries

- Implement proper input validation and output encoding
- Disable detailed error messages
- Apply least privilege principle
- Regular security testing and updates

—

Summary

This practical demonstrated how common web vulnerabilities such as SQL Injection and XSS can be identified using Burp Suite and DVWA. Proper security controls and secure coding practices are essential to protect web applications.

—