# Project:3

# Automated vs Manual Testing Project

DVWA Security Assessment

This project compares the effectiveness of automated security scanners with manual testing techniques using DVWA as the target application.

---

## Objective

The objectives of this project were:

- To perform automated security testing on DVWA using common tools
- To manually test the same application using interception and crafted payloads
- To document and compare findings from both approaches
- To analyze strengths and weaknesses of automated scanners

---

## Scope of Testing

The scope of this assessment included:

- DVWA running locally on Kali Linux
- Web application testing only

All testing was performed in a controlled lab environment.

---

## Tools Used

- Kali Linux
- DVWA
- Nikto
- OWASP ZAP
- sqlmap
- Burp Suite
- Firefox Browser

---

# Methodology

The testing methodology followed these phases:

- Automated vulnerability scanning
- Manual interception and testing
- Validation of findings
- Comparison and analysis

---

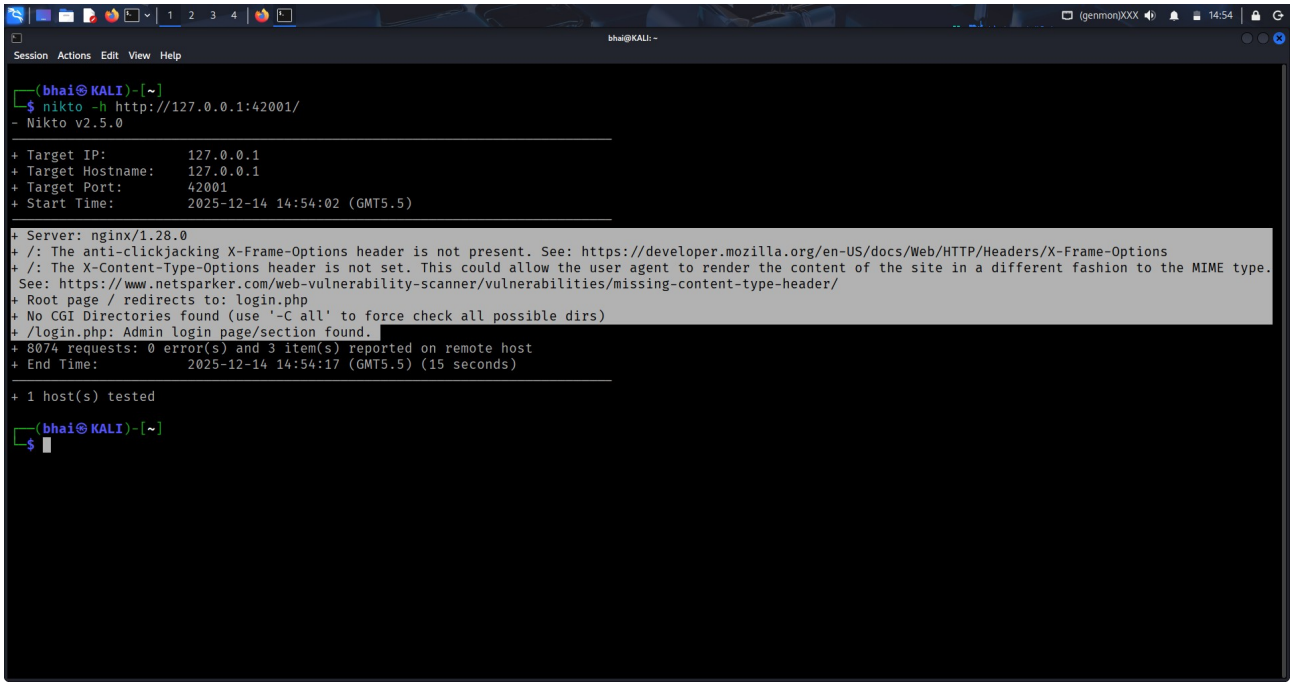
# Automated Testing on DVWA

## Nikto Scan

Nikto was used to identify web server misconfigurations and known issues.

Command used:

`nikto -h http://127.0.0.1/42001/`




Findings:

- Missing HTTP security headers
- Outdated server configuration
- Information disclosure issues, page redirection

Nikto focused mainly on server-level weaknesses rather than application logic.

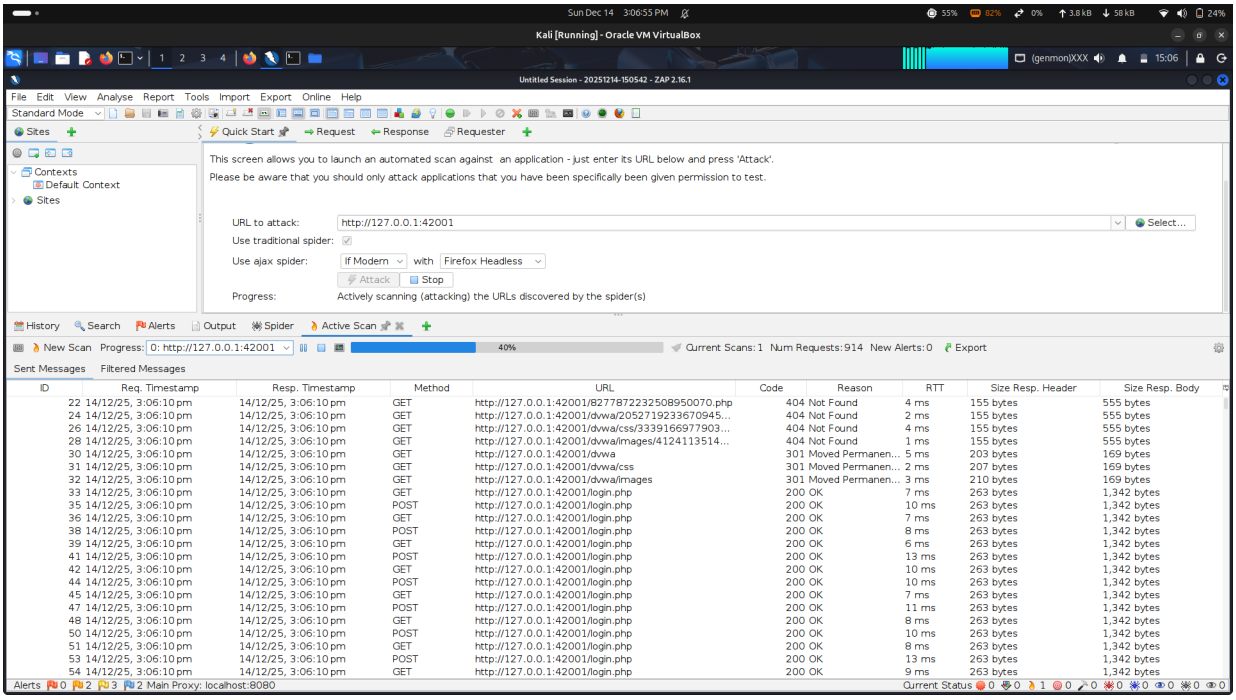
---

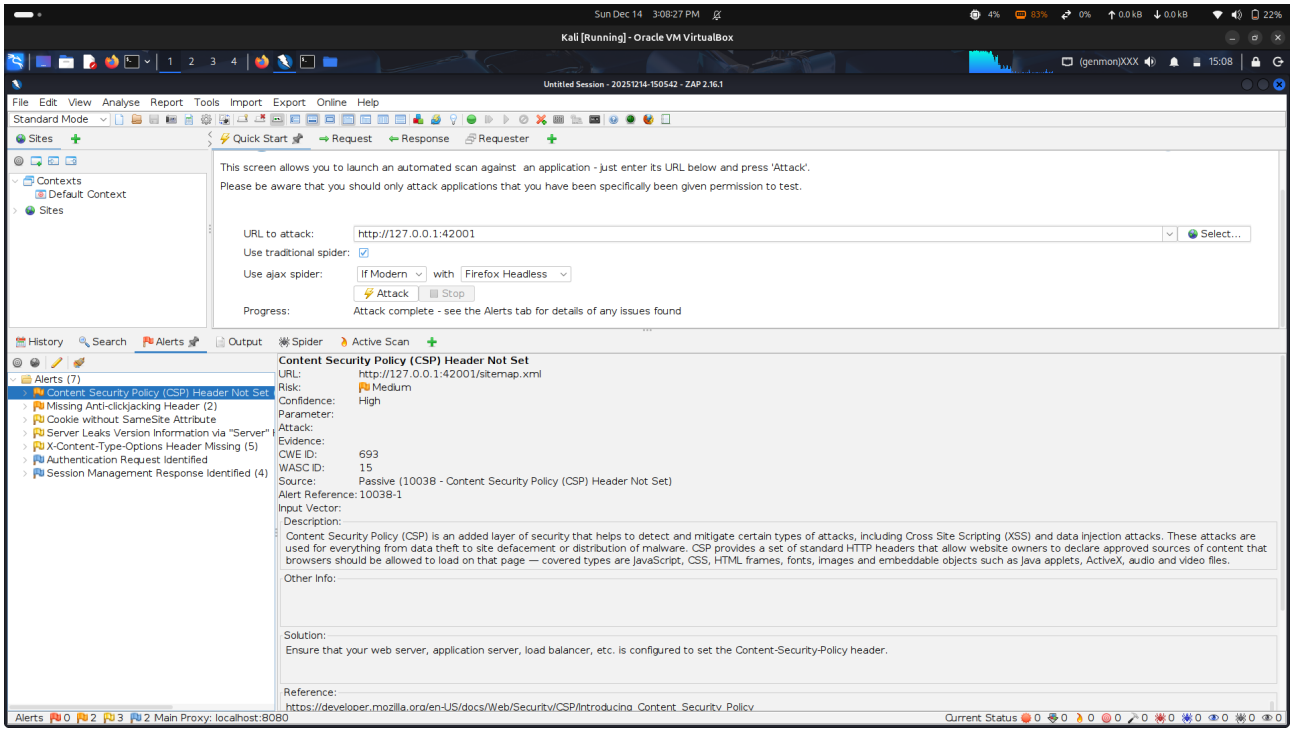
# OWASP ZAP Scan

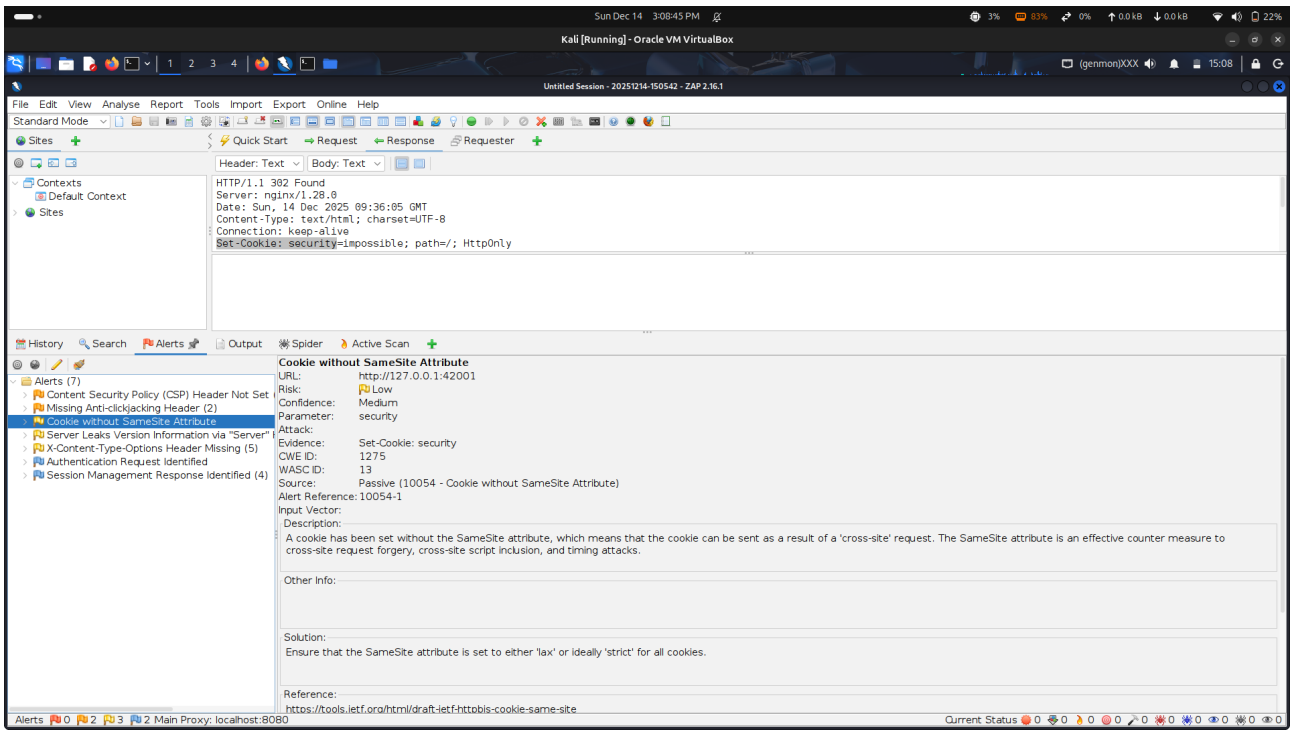
OWASP ZAP was used to perform an automated web application scan.

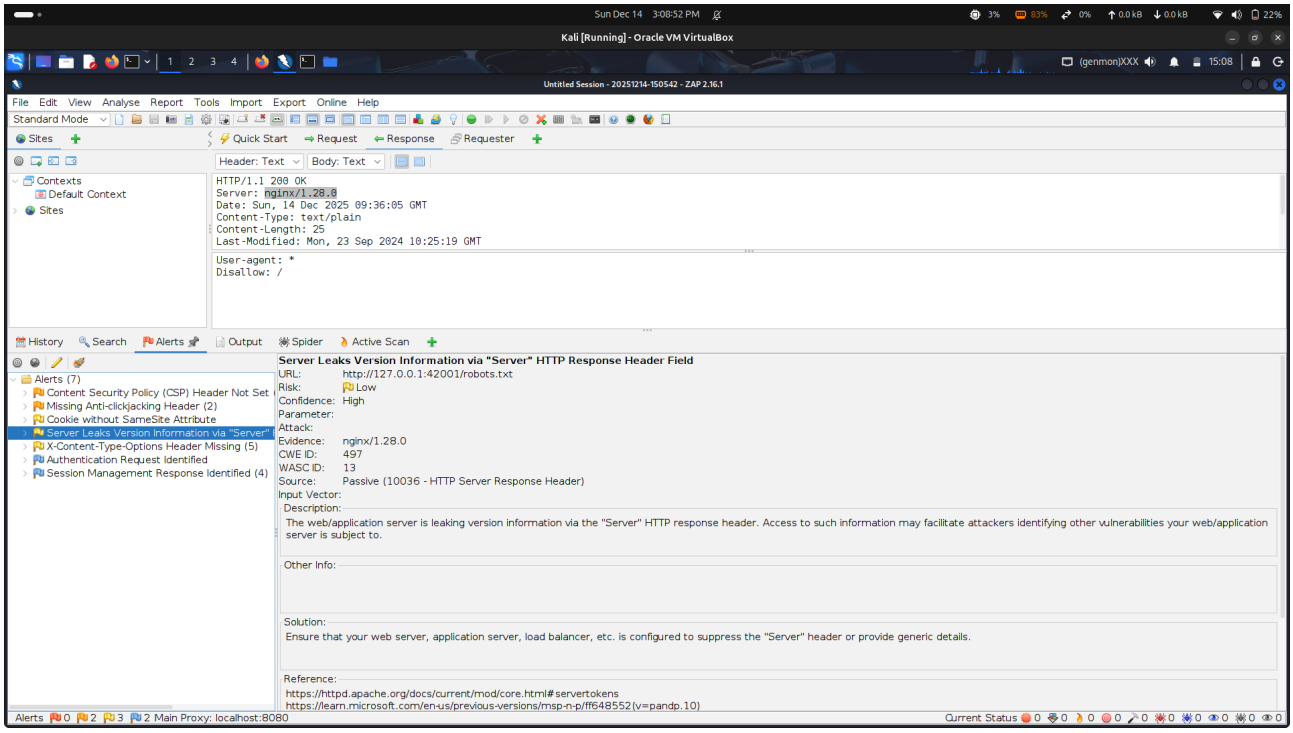
Steps performed:

- DVWA opened in browser through ZAP
- Spidering enabled to discover pages
- Active scan initiated






Findings:

- Content policy header missing
- Cookie security issues without samesite parameter
- Missing security headers
- Server Leak version

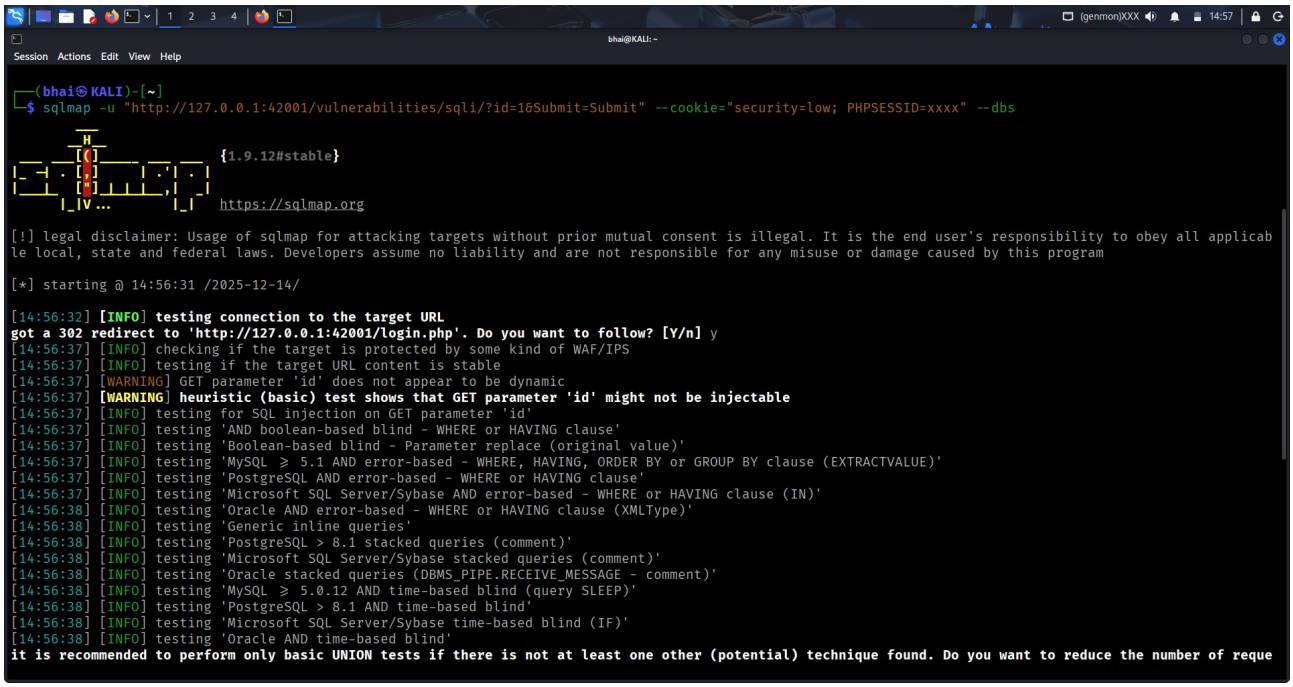
ZAP provided broad coverage but also generated some low-confidence alerts.

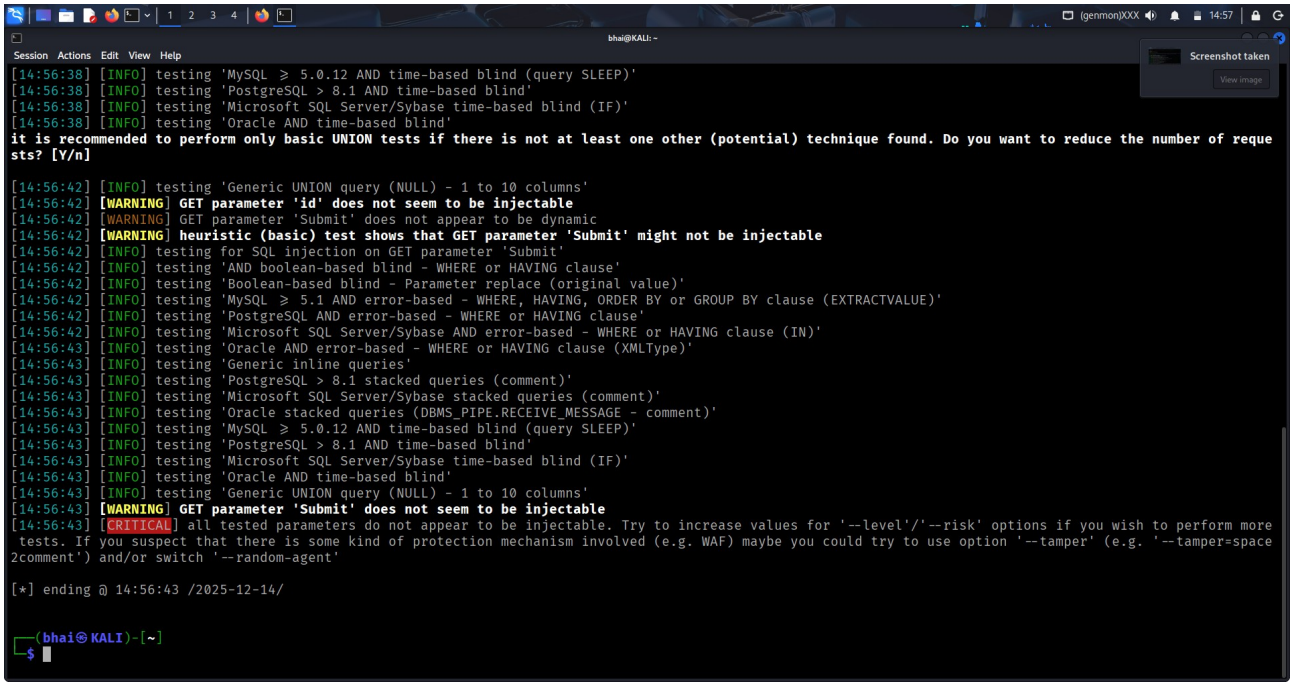
## sqlmap Scan

sqlmap was used to test for SQL Injection vulnerabilities.

Command used:

```
sqlmap -u "http://127.0.0.1/42001/vulnerabilities/sqli/?id=1&Submit=Submit" --
cookie="security=low; PHPSESSID=xxxx" –dbs
```




Findings:

- GET parameter 'Submit' does not seem to be injectable

- GET parameter 'id' does not seem to be injectable
- GET parameter 'Submit' does not appear to be dynamic
- heuristic (basic) test shows that GET parameter 'Submit' might not be injectable

sqlmap successfully automated exploitation of SQL Injection without manual payload crafting.

---

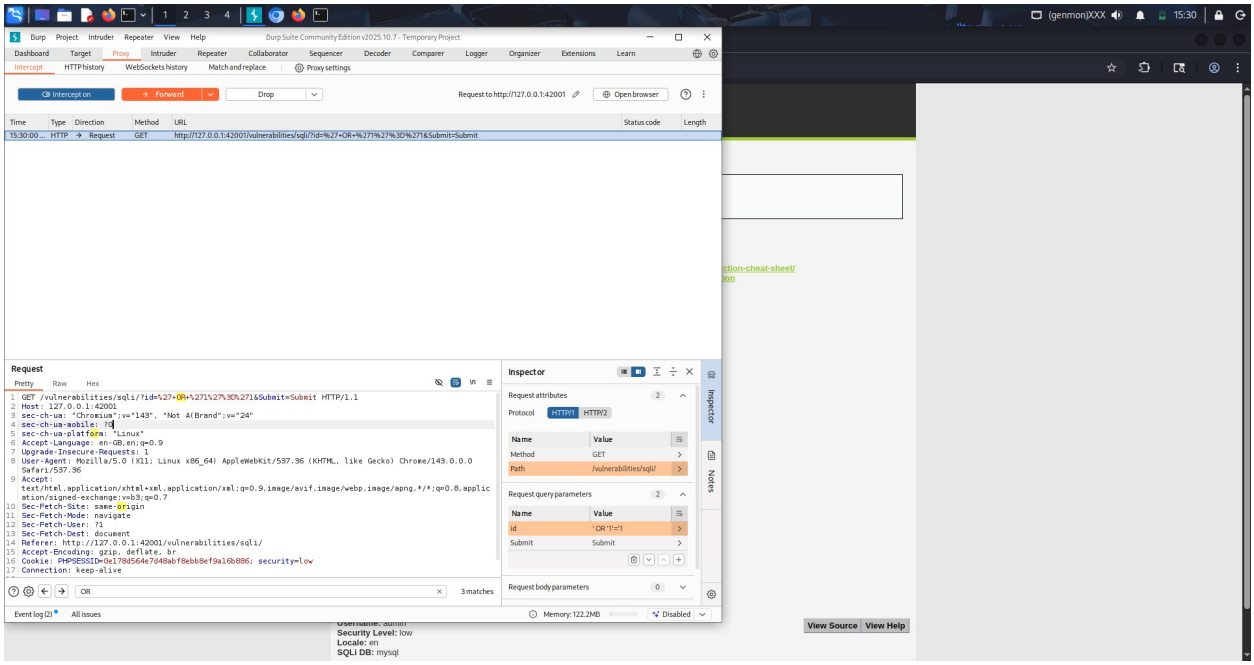
# Manual Testing on DVWA

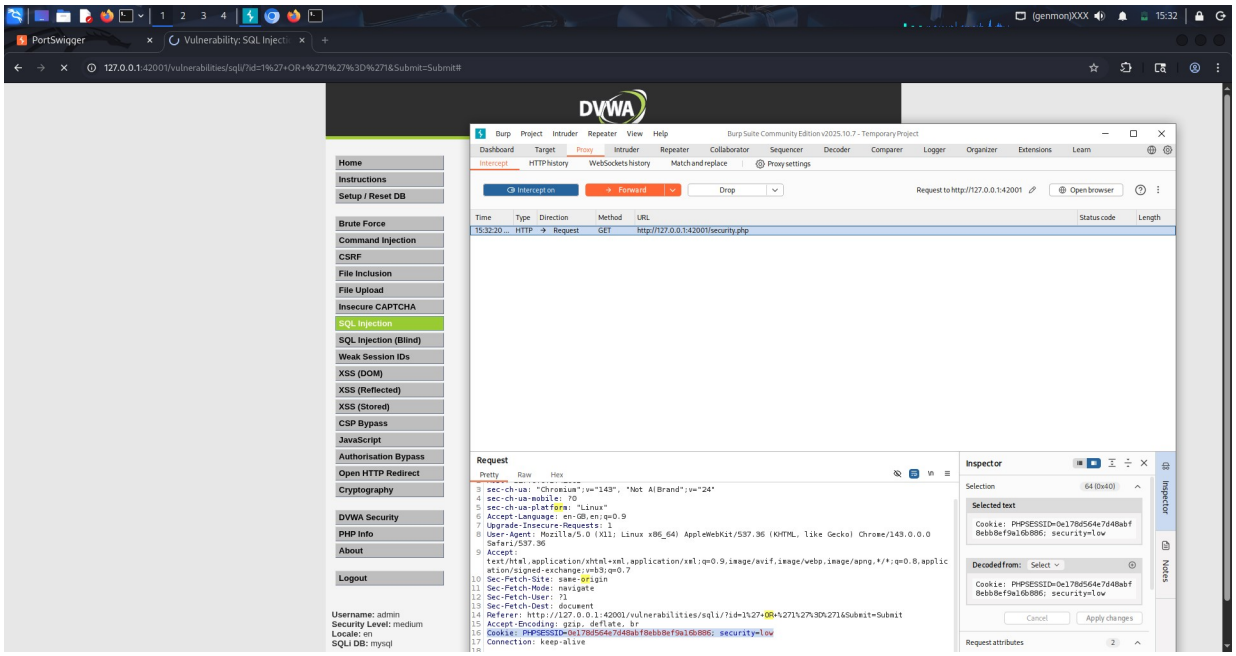
## Manual SQL Injection Testing

Manual SQL Injection testing was performed using crafted payloads.

Payload used:

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





Observation:

- Authentication bypass was successful
- Database results were manipulated

Manual testing helped understand how the vulnerability actually worked.

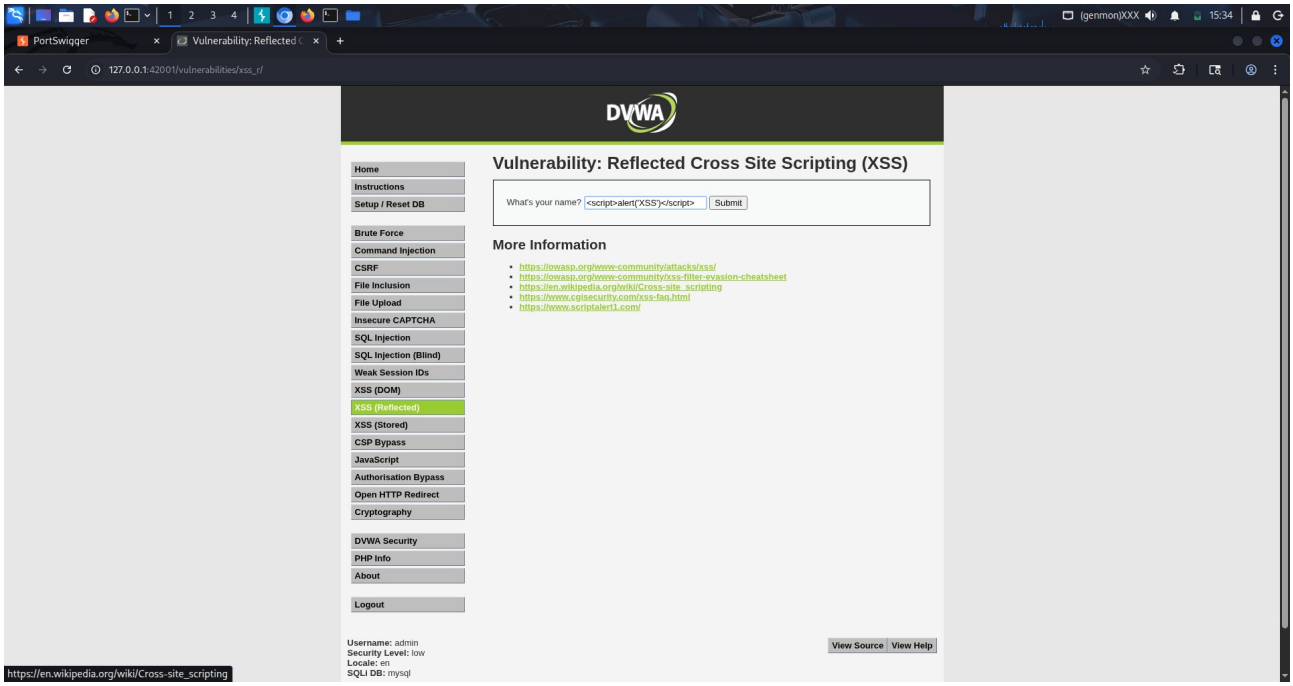
---

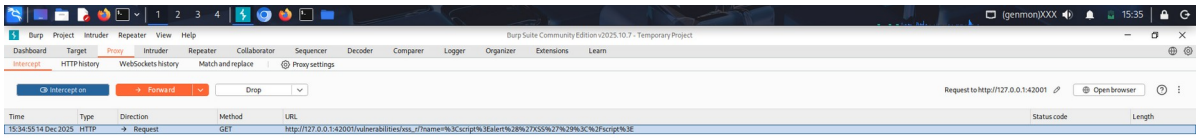
## Manual Cross-Site Scripting Testing

Manual XSS testing was performed using input fields.

Payload used:

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



Observation:

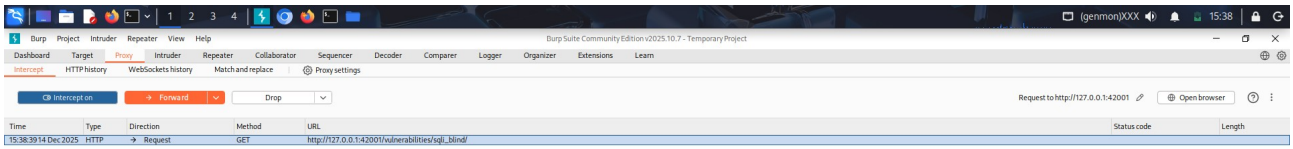
- JavaScript executed in the browser
- Alert popup confirmed XSS

Manual testing allowed precise payload control and confirmation.

---

### Burp Suite Interception

Burp Suite was used to intercept and modify HTTP requests.





Observation:

- Parameters could be manipulated
- Hidden inputs were identified
- Request and response behavior was clearly visible

Burp helped in understanding application logic and data flow.

---

# Comparison of Automated and Manual Testing

Automated scanners quickly identified common vulnerabilities such as missing headers, SQL Injection, and basic XSS. They saved time and provided wide coverage of the application.

Manual testing allowed deeper understanding of vulnerabilities, precise payload crafting, and validation of exploitability. Some issues, such as logical flaws and parameter manipulation, were better identified manually.

Automated tools sometimes produced false positives, while manual testing required more skill and time.

---

## Issues Identified by Automation Only

- Missing HTTP security headers
- Server misconfigurations
- Low-risk informational findings
- Broad vulnerability patterns

---

## Issues Identified by Manual Testing Only

- Precise SQL Injection behavior
- Context-specific XSS payload execution
- Parameter manipulation
- Application logic understanding

---

## Executive Summary

This assessment compared automated and manual security testing techniques on DVWA. Automated tools efficiently detected common vulnerabilities and configuration issues, while manual testing provided deeper insight and accurate validation of exploitability.

The results demonstrate that relying on only one approach is insufficient for comprehensive security testing.

---

## Strengths and Weaknesses of Automated Scanners

Strengths:

- Fast and efficient scanning
- Broad coverage of application
- Useful for initial reconnaissance
- Easy to use for beginners

Weaknesses:

- False positives
- Limited understanding of application logic

- Cannot fully validate exploit impact
- Miss complex or chained vulnerabilities

---

# Conclusion

Automated scanners and manual testing complement each other in web application security assessments. Automated tools are ideal for initial discovery, while manual testing is essential for accurate validation and deeper analysis.

A balanced approach using both methods provides the most effective and reliable security testing results.

---

# Learning Outcomes

- Understanding of automated and manual testing approaches
- Hands-on experience with Nikto, ZAP, sqlmap, and Burp Suite
- Improved vulnerability analysis skills
- Better understanding of tool limitations
- Professional security assessment reporting experience