HOME ABOUT CONTENT OTHERS

WEB APPLICATION SECURITY TESTING PROJECT

Manual Vulnerability Assessment using Burp Suite

SELF-LEARNING CYBERSECURITY PROJECT
ALIZHA SYAFAHIRA | 2026

PROJECT OVERVIEW

PERFORMED MANUAL WEB APPLICATION SECURITY TESTING

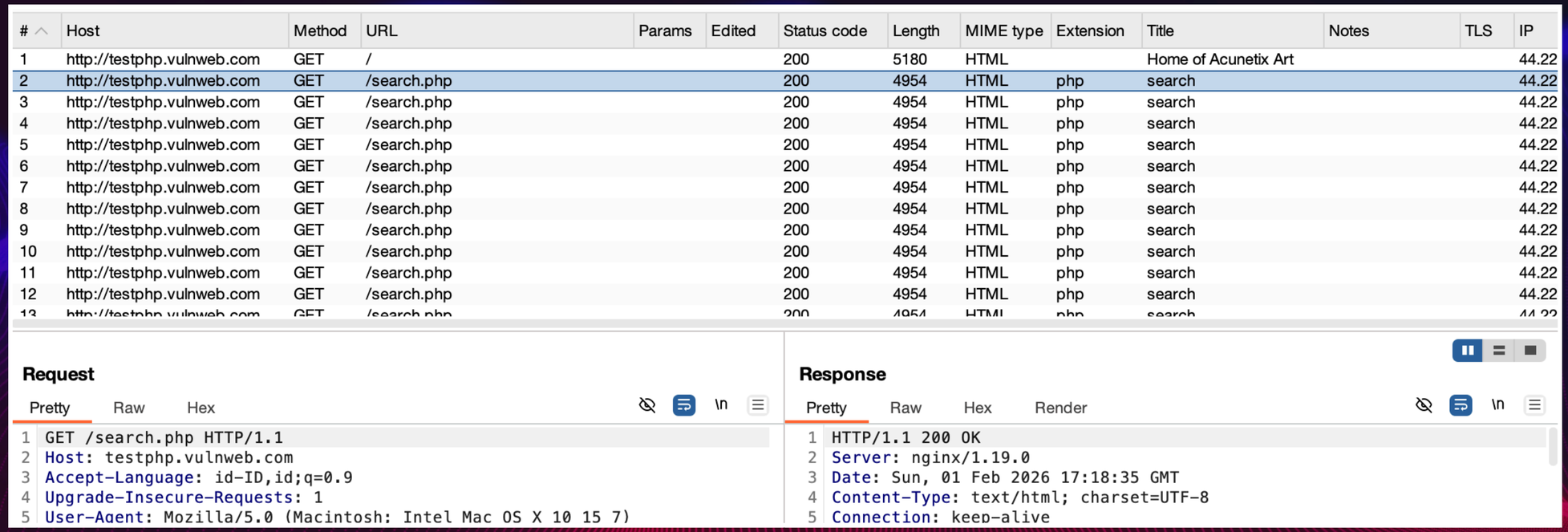
IDENTIFIED COMMON VULNERABILITIES SUCH AS XSS AND SQL INJECTION

TESTING CONDUCTED IN A CONTROLLED AND ETHICAL ENVIRONMENT

TARGET APPLICATION:

testphp.vulnweb.com (intentionally vulnerable test site)

TOOLS & METHODOLOGY



#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP
1	http://testphp.vulnweb.com	GET	/			200	5180	HTML		Home of Acunetix Art			44.22
2	http://testphp.vulnweb.com	GET	/search.php			200	4954	HTML	php	search			44.22
3	http://testphp.vulnweb.com	GET	/search.php			200	4954	HTML	php	search			44.22
4	http://testphp.vulnweb.com	GET	/search.php			200	4954	HTML	php	search			44.22
5	http://testphp.vulnweb.com	GET	/search.php			200	4954	HTML	php	search			44.22
6	http://testphp.vulnweb.com	GET	/search.php			200	4954	HTML	php	search			44.22
7	http://testphp.vulnweb.com	GET	/search.php			200	4954	HTML	php	search			44.22
8	http://testphp.vulnweb.com	GET	/search.php			200	4954	HTML	php	search			44.22
9	http://testphp.vulnweb.com	GET	/search.php			200	4954	HTML	php	search			44.22
10	http://testphp.vulnweb.com	GET	/search.php			200	4954	HTML	php	search			44.22
11	http://testphp.vulnweb.com	GET	/search.php			200	4954	HTML	php	search			44.22
12	http://testphp.vulnweb.com	GET	/search.php			200	4954	HTML	php	search			44.22
13	http://testphp.vulnweb.com	GET	/search.php			200	4954	HTML	php	search			44.22

TOOLS

- Burp Suite (Intercept, Repeater)

METHODOLOGY

- Intercept HTTP requests
- Analyze parameters and responses
- Inject payloads manually
- Validate vulnerabilities via server response

IDENTIFIED VULNERABILITY: REFLECTED XSS

VULNERABILITY TYPE:
REFLECTED CROSS-SITE SCRIPTING (XSS)

ENDPOINT: SEARCH.PHP

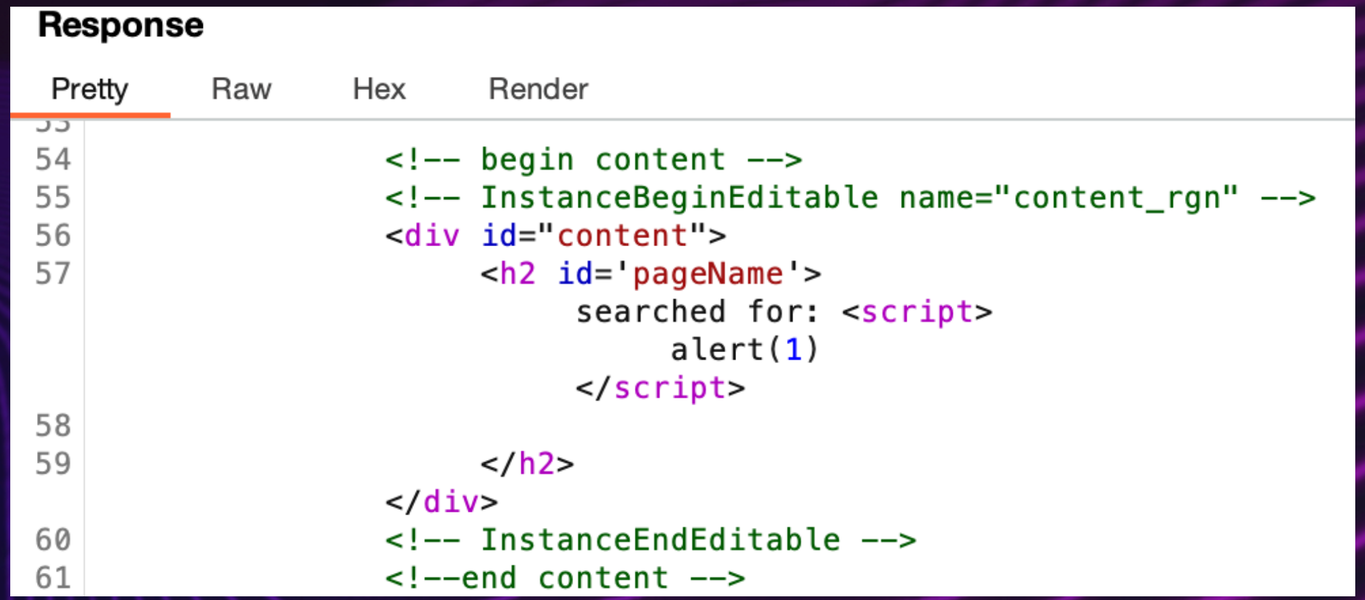
SEVERITY: MEDIUM

XSS PROOF OF CONCEPT

USER INPUT IS REFLECTED DIRECTLY INTO THE HTML RESPONSE

NO INPUT SANITIZATION OR OUTPUT ENCODING OBSERVED

INJECTED JAVASCRIPT PAYLOAD IS RENDERED IN THE BROWSER CONTEXT



SQL INJECTION EVIDENCE

```
GET /listproducts.php?cat=1' HTTP/1.1
```

```
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
```

SINGLE QUOTE (') CAUSED DATABASE ERROR

INDICATES UNSANITIZED INPUT PASSED TO SQL QUERY

DATABASE ERROR MESSAGES EXPOSED TO USER

MITIGATION & RECOMMENDATIONS

- Validate and sanitize all user input on the server side
- Use parameterized queries to prevent SQL Injection
- Encode user input before rendering it in HTML to mitigate XSS
- Conduct regular security testing during development

SECURITY IMPACT

The identified vulnerabilities allow attackers to inject malicious scripts and manipulate backend database queries.
If exploited in a real-world environment, these issues could lead to data leakage, session hijacking, and unauthorized system access.

CONCLUSION

This project demonstrates hands-on experience in manual web application security testing.
Through HTTP request analysis and controlled payload injection, multiple security vulnerabilities were identified in a deliberately vulnerable environment.
The testing process emphasizes ethical testing practices and responsible security research.

KEY TAKEAWAYS

- Gained hands-on experience in manual web security testing
- Learned to analyze HTTP requests and responses using Burp Suite
- Understood how improper input handling leads to XSS vulnerabilities
- Applied ethical and responsible testing practices

THANK YOU

THANK YOU FOR REVIEWING THIS PROJECT

THIS PROJECT REFLECTS MY INTEREST IN WEB APPLICATION SECURITY AND CONTINUOUS LEARNING IN CYBERSECURITY