BUBT | BANGLADESH UNIVERSITY OF BUSINESS AND TECHNOLOGY

Committed to Academic Excellence

Documantation-01

Course Title : Cyber Security and Digital Forensic

Course Code : CSE 413

Prepared by: Md. Mursalin Hasan Nirob

Command Injection, XSS Attacks, and Proxy Setup

Table of Contents

1. DVWA Setup Report

2. Burp Suite Proxy Setup Report

3. Command Injection Attack Documentation (Low → High)

4. Stored XSS Attack Documentation (Low → High)

5. Reflected XSS Attack Documentation (Low → High)

6. Summary & Recommendations

1. DVWA Setup Report

Overview
This report documents the complete setup process for Damn Vulnerable Web Application (DVWA) on a Linux system, detailing each configuration step required to establish a functional penetration testing environment.

Prerequisites
Linux system with Apache2 web server
MySQL database server
PHP (version 8.2 or higher)
Git for repository management

Setup Process
1. DVWA Installation
Objective: Clone and prepare the DVWA application files

Steps:
Navigate to the web server root directory:
cd /var/www/html
Clone the DVWA repository from GitHub:
git clone https://github.com/digininja/DVWA.git
Configure the application by copying the distribution config file:
cd DVWA/config
cp config.inc.php.dist config.inc.php
Edit config.inc.php to update database credentials, specifically modifying the password field to match your database setup.

2. Database Configuration
Objective: Set up MySQL database with appropriate user permissions

Steps:
Start MySQL service:
sudo systemctl start mysql
Access MySQL as root user:
sudo mysql -u root
Create DVWA database user and configure permissions:
CREATE USER 'dvwa'@'localhost' IDENTIFIED BY 'your_configured_password';
GRANT ALL PRIVILEGES ON dvwa.* TO 'dvwa'@'localhost';

FLUSH PRIVILEGES;
EXIT;

3. PHP Configuration
Objective: Configure PHP settings to support DVWA functionality

Steps:
Identify your PHP version:
php -v
Edit the PHP configuration file (path varies by version):
sudo nano /etc/php/8.2/apache2/php.ini
Note: Version number may differ (e.g., 8.4) - verify your specific PHP version
Locate and modify the following settings:
- allow_url_fopen = On
- allow_url_include = On

4. Apache2 Service Management
Objective: Start and configure the web server

Steps:
Start Apache2 service:
sudo systemctl start apache2
Enable Apache2 to start on boot:
sudo systemctl enable apache2

5. DVWA Database Initialization
Objective: Create and populate the DVWA database schema

Steps:
Access the DVWA setup page via web browser:
http://localhost/DVWA/setup.php
Click the "Create/Reset Database" button to initialize the database structure and sample data.

6. DVWA Access and Authentication
Objective: Access the application interface

Steps:
Navigate to the login page:
http://localhost/DVWA/login.php
Log in using default credentials:
Username: admin
Password: password
Upon successful authentication, you will be redirected to the main DVWA dashboard where various vulnerability categories are available for testing.

Post-Setup Verification
Confirm all DVWA modules display green status indicators on the setup page
Verify database connectivity
Test basic functionality by attempting a simple vulnerability exercise

Security Considerations
This setup is intended for educational and testing purposes only
Ensure DVWA is deployed in an isolated environment
Never expose DVWA to production networks or public internet
Use this environment exclusively for authorized penetration testing practice

Available Attack Categories
Once successfully configured, DVWA provides hands-on experience with:
- SQL Injection
- Cross-Site Scripting (XSS)
- Cross-Site Request Forgery (CSRF)
- File Upload vulnerabilities
- Command Injection
- Authentication bypass techniques

This setup provides a controlled environment for practicing ethical hacking techniques and understanding web application security vulnerabilities.

2. Burp Suite Proxy Setup Report

Overview
This report documents the configuration process for setting up Burp Suite proxy integration with Firefox browser, enabling comprehensive web application security testing through traffic interception and analysis.

Prerequisites
Burp Suite Professional or Community Edition installed
Firefox web browser
Administrative access to modify browser security settings

Setup Process
1. Burp Suite Proxy Configuration
Objective: Configure Burp Suite to listen for incoming proxy connections

Steps:
Launch Burp Suite application
Navigate to the Proxy tab in the main interface
Access Options sub-tab under Proxy settings
Configure proxy listener:
Interface: 127.0.0.1:8080
Protocol: HTTP/HTTPS
Ensure the listener is set to "Running" status

2. Firefox Browser Proxy Configuration
Objective: Direct browser traffic through Burp Suite proxy

Steps:
Open Firefox browser
Access network settings:
Navigate to Settings → General → Network Settings
Click Settings button in the "Network Proxy" section
Configure manual proxy settings:
Select Manual proxy configuration
HTTP Proxy: 127.0.0.1 Port: 8080
HTTPS Proxy: 127.0.0.1 Port: 8080
Ensure "Use this proxy server for all protocols" is checked

3. SSL Certificate Installation

Objective: Install Burp Suite's CA certificate to handle HTTPS traffic

Steps:
Download CA Certificate:
Open new Firefox tab
Navigate to: https://burp
Download the CA certificate file from the Burp Suite certificate page
Install Certificate in Firefox:
Access Firefox Settings
Search for "certificate" in the settings search bar
Click View Certificates button
In the Certificate Manager popup window:
Select the Authorities tab
Click Import button
Browse and select the downloaded Burp CA certificate file
Certificate Trust Settings: Check all available options:
"Trust this CA to identify websites"
"Trust this CA to identify email users"
"Trust this CA to identify software developers"
Click OK to complete installation

4. Firefox Advanced Configuration
Objective: Enable localhost proxy connections for comprehensive testing

Steps:
Access Firefox advanced configuration:
Type about:config in the address bar and press Enter
Accept the warning message about advanced settings
Configure localhost proxy setting:
Use the search bar to locate: network.proxy.allow_hijacking_localhost
Double-click the preference name or click the toggle button
Change the value from false to true
This enables proxy interception of localhost traffic

Verification and Testing
1. Connection Verification
Navigate to any HTTPS website in Firefox
Verify that requests appear in Burp Suite's HTTP history tab
Confirm that HTTPS traffic is successfully decrypted and displayed

2. Localhost Testing
Access local applications (e.g., http://localhost/DVWA)
Verify that localhost traffic is captured in Burp Suite
Test both HTTP and HTTPS local connections

Configuration Summary

Component	Setting	Value
Burp Suite Proxy	Interface	127.0.0.1:8080
Firefox HTTP Proxy	Address:Port	127.0.0.1:8080
Firefox HTTPS Proxy	Address:Port	127.0.0.1:8080
CA Certificate Trust Level		Full (all options)
Localhost Hijacking	Status	Enabled (true)

Security Considerations
Best Practices
Use this configuration only in isolated testing environments
Disable proxy settings when not performing security testing
Regularly update Burp Suite to maintain current security features
Store CA certificates securely and remove when testing is complete

Important Notes
The CA certificate installation allows Burp Suite to decrypt HTTPS traffic
Localhost hijacking enables testing of local web applications
All web traffic will be visible in Burp Suite when proxy is active
Consider creating separate Firefox profiles for security testing vs. regular browsing

Troubleshooting
Common Issues
Connection refused: Verify Burp Suite proxy listener is running
Certificate errors: Ensure CA certificate is properly installed in Authorities tab
Localhost not intercepted: Confirm network.proxy.allow_hijacking_localhost is set to true
Performance issues: Consider adjusting Burp Suite memory allocation for large applications

This configuration establishes a comprehensive web application security testing environment, enabling detailed analysis of HTTP/HTTPS traffic for vulnerability assessment and penetration testing activities.

3. Command Injection Attack Documentation

Introduction
This document describes the procedure followed to conduct command injection attacks on DVWA, progressing through security levels from Low to High.

Prerequisites
DVWA installed and accessible locally or in a lab environment
Command Injection module in DVWA available for testing
Basic Linux command line knowledge
Video reference: "Command-Injection.mp4" for real-time demonstration

Setup
Access DVWA through http://localhost/DVWA/login.php
Log in with default user credentials (admin/password)
Navigate to the Command Injection vulnerability page
Adjust DVWA's security level to start from Low and progress upward to High

Attack Execution Steps

1. Command Injection Attack at Low Security Level
Goal: Exploit the vulnerability without any input sanitization or validation.

Procedure:
- Set security level to Low in DVWA.
- On the Command Injection page, input a simple payload such as:
127.0.0.1; ls -l
127.0.0.1; cd /etc/php
The semicolon (;) allows command chaining on Linux systems.
- Submit the form; observe the output displaying a list of files indicating command execution.
- Test additional commands like:
127.0.0.1; whoami
to confirm remote command execution.
- Use various Linux commands to explore the system's response.

2. Command Injection Attack at Medium Security Level
Goal: Bypass basic input filtering mechanisms.

Procedure:

- Change DVWA security to Medium.
- Retry known commands; observe if special characters like ; or && are filtered or blocked.
- Use bypass techniques such as:
URL encoding special characters
Using alternative shell syntax:
127.0.0.1|ls
127.0.0.1`ls`
- Observe the results to confirm if command execution is achievable despite restrictions.
- Iterate testing with encoded payloads or obfuscated input to bypass sanitization.

3. Command Injection Attack at High Security Level
Goal: Test advanced filtering that blocks common injection characters.

Procedure:
- Set DVWA to High security.
- Attempt commands that avoid classic delimiters (;, |, &&).
- Experiment with:
Using backticks or command substitution mechanisms:
127.0.0.1 $(ls)
- Chained commands using pipes or redirections.
- Use encoding tricks like double URL encoding or Unicode representations.
- Supplement testing with proxy tools (e.g., Burp Suite) to manipulate and fine-tune input payloads.
- Confirm any successful command execution by output displayed on the webpage.

Considerations and Recommendations
- Always perform command injection testing in a legal, controlled environment.
- Use automated tools and manual testing in parallel for comprehensive coverage.
- Learn how different security settings on DVWA simulate real-world defenses.
- Study the filtering rules to develop custom payloads for evasion.
- Protect live environments by implementing input validation, output encoding, and least privilege principles.

4. Stored XSS Attack Documentation

Preparation Before the Attack
Before starting the Stored Cross-Site Scripting (XSS) attack on DVWA, it is essential to prepare the environment and configure settings properly:

DVWA Setup:
Ensure DVWA application is properly installed and accessible, typically at http://localhost/DVWA/login.php.
Log in using valid credentials (default: admin/password).
Set DVWA security level; begin testing with Low, then progress to Medium and High to observe different filtering mechanisms.

Understand Target Page:
Navigate to the XSS (Stored) section within DVWA.
Identify input fields where malicious payloads can be submitted (e.g., message/comment box).
Review how submitted inputs are displayed on the page and whether they are stored persistently.

Browser and Tools Setup:
Use a modern browser with developer tools active.
Optionally, configure a proxy like Burp Suite for request/response inspection and modification.
Familiarize yourself with how the application behaves with benign inputs.

Attack Procedure
The video demonstrates injecting malicious scripts into stored XSS vulnerable fields, observing how they are executed for all users visiting the page.

1. Stored XSS Attack at Low Security Level
Goal: Inject JavaScript code that is stored and executed without sanitization.
Steps:
- Set DVWA Security to Low.
- In the input box (e.g., message/comment), enter a script payload like:
<script>alert('Test XSS Low')</script>
- Submit the form.
- Reload or navigate to the page where comments/messages are displayed.
- Observe the alert box popping up, confirming the payload is stored and executed for all visitors.

- Test with multiple payloads to confirm persistent execution.

2. Stored XSS Attack at Medium Security Level
Goal: Bypass input filtering that blocks direct <script> tags but may allow event handlers or alternative tags.
Steps:
- Switch DVWA security to Medium.
- Submit payloads that avoid <script> tags but use event handlers, e.g.:
<img src=x onerror=alert('XSS Medium')>
- Submit and reload the page to verify execution.
- Try alternative SVG or other HTML-based payloads known to bypass filters:
<svg onload=alert('XSS Bypass')>
- Use proxy or browser dev tools to encode or obfuscate payloads if needed.

3. Stored XSS Attack at High Security Level
Goal: Overcome strong filtering and sanitization mechanisms.
Steps:
- Set DVWA Security to High.
- Attempt advanced XSS payloads that leverage less common event handlers or encoded payloads, for example:
<body onload=alert('XSS High')>
<iframe src="javascript:alert('XSS')"></iframe>
- Experiment with encoding such as URL or Unicode to evade filters.
- Submit payloads and check if alerts pop up when the page is viewed.
- Use Burp Suite to manipulate requests on the fly to test payload variations and evasions.
- Confirm successful attacks by alert prompts or unexpected script behavior.

Recommendations
- Test incrementally from low to high security levels to understand filtering improvements.
- Use proxy tools to fine-tune payload delivery and observe server responses.
- Implement strict input validation and output encoding to mitigate stored XSS in production.
- Always conduct testing in legal and isolated environments to avoid unintended impact.

5. Reflected XSS Attack Documentation

Preparation Before the Attack
Before conducting the XSS attack on DVWA, the following setup and preparation steps should be followed:

Environment Setup:
Ensure DVWA is installed and accessible, typically at http://localhost/DVWA/login.php.
Login to DVWA with valid credentials (default: admin/password).
Verify the web server, database, and DVWA services are running properly.

Configure Security Level:
DVWA allows choosing security levels: Low, Medium, and High.
Begin with security level set to Low and progress gradually to higher levels for analysis.

Browser Configuration:
Use a modern web browser, preferably with developer tools enabled.
Optionally configure a proxy tool like Burp Suite to intercept and analyze requests and responses for deeper test control.

Understand the Vulnerable Page:
Navigate to the XSS (Reflected) section in DVWA.
Familiarize yourself with input fields where injection will be tested (e.g., Name input box).
Review response behavior by submitting benign inputs first.

Attack Procedure
The video demonstrates injecting and exploiting reflected XSS vulnerabilities across security levels.

1. Attacking at the Low Security Level
Goal: Inject and execute arbitrary JavaScript with no filtering.
Steps:
- Ensure DVWA security is set to Low.
- In the Name input field on the reflected XSS page, enter the following payload: <script>alert('Hello XSS Low')</script>
- Submit the form.
- Observe the pop-up alert box showing the message “Hello XSS Low,” confirming

successful script execution.
- This initial success shows that user input is reflected without sanitization or encoding.

2. Attacking at the Medium Security Level
Goal: Bypass basic filtering that blocks direct script tags.
Steps:
- Change the DVWA security level to Medium.
- Enter an event-handler based payload that bypasses script tag filtering, for example:
<img src=x onerror=alert('XSS Medium')>
- Submit the payload.
- Confirm that the alert box is displayed, indicating successful execution despite the blocking of <script> tags.
- Experiment with SVG payloads known for filter evasion:
<svg onload=alert('XSS Bypass')>
- Adjust payloads incrementally, testing the effectiveness of different tags and event handler attributes.

3. Attacking at the High Security Level
Goal: Find advanced payloads to bypass stricter input validation and filtering.
Steps:
- Switch DVWA security to High.
- Attempt more sophisticated payloads that avoid traditional script tags or use less common event attributes:
<body onload=alert('XSS High')>
<iframe src="javascript:alert('XSS High')"></iframe>
- Use encoding techniques such as URL encoding, Unicode characters, or alternate syntax to evade filters.
- Submit payloads and observe the output for successful execution.
- Leverage developer tools and proxy interceptors like Burp Suite to fine-tune inputs and identify bypasses.
- Confirm popup alerts or other script behaviors to validate the injection.

Summary
Key takeaways:
- Low Level: Direct script injections work flawlessly due to no protections.
- Medium Level: Blocking of <script> tags forces use of alternative HTML event handlers and tags.
- High Level: Requires sophisticated payloads, encoding, and evasion techniques to

succeed.

Important Notes
- Always conduct testing in a controlled, legal environment.
- Use proxy tools and browser consoles to enhance payload crafting and analysis.
- Understand filters in place to develop targeted bypass payloads.

This attack flow helps in understanding how real-world web apps defend against reflected XSS and how attackers try to circumvent these defenses.

6. Summary & Recommendations

- Always test DVWA and related attacks in isolated lab environments, never on production.
- DVWA security levels demonstrate how different protections work (Low → High).
- Burp Suite provides powerful features for intercepting, modifying, and analyzing requests.
- Command Injection, Stored XSS, and Reflected XSS are critical vulnerabilities to understand.
- Protect real applications using:
  * Strong input validation
  * Proper output encoding
  * Principle of least privilege
  * Regular penetration testing