DVWA
Security Assessment Findings Report

Business Confidential

Date: September 18th, 2024 Project: DC-001
Version 1.0

Table of Contents

Contents

Confidentiality Statement

This document is the exclusive property of DVWA and Cryothink Security (CRS). This document contains proprietary and confidential information. Duplication, redistribution, or use, in whole or in part, in any form, requires consent of both DVWA and CRS.

DVWA may share this document with auditors under non-disclosure agreements to demonstrate penetration test requirement compliance.

Disclaimer

A penetration test is considered a snapshot in time. The findings and recommendations reflect the information gathered during the assessment and not any changes or modifications made outside of that period.

Time-limited engagements do not allow for a full evaluation of all security controls. CRS prioritized the assessment to identify the weakest security controls an attacker would exploit. CRS recommends conducting similar assessments on an annual basis by internal or third-party assessors to ensure the continued success of the controls.

Contact Information

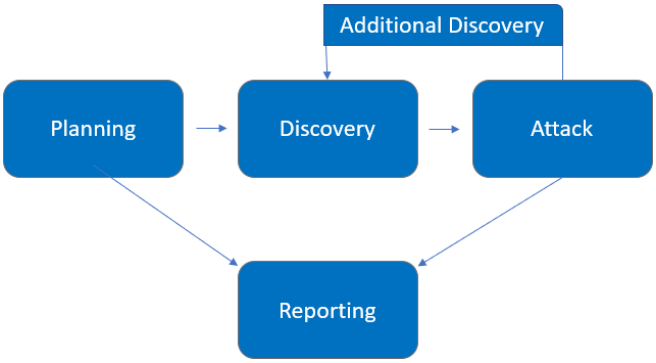
Name	Title	Contact Information
CR Security		
Fajar Yasodana	Lead Penetration Tester	Email: fajaryasodana@gmail.com

Assessment Overview

From September 15th, 2024 to September 18th, 2024, DVWA engaged CRS to evaluate the security posture of its infrastructure compared to current industry best practices that included an internal network penetration test.

Phases of penetration testing activities include the following:

- Planning – Customer goals are gathered and rules of engagement obtained.
- Discovery – Perform scanning and enumeration to identify potential vulnerabilities, weak areas, and exploits.
- Attack – Confirm potential vulnerabilities through exploitation and perform additional discovery upon new access.
- Reporting – Document all found vulnerabilities and exploits, failed attempts, and company strengths and weaknesses.



Finding Severity Ratings

The following table defines levels of severity and corresponding CVSS score range that are used throughout the document to assess vulnerability and risk impact.

Severity	CVSS V3 Score Range	Definition
Critical	9.0-10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
High	7.0-8.9	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.
Moderate	4.0-6.9	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
Low	0.1-3.9	Vulnerabilities are non-exploitable but would reduce an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window.
Informational	N/A	No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

Risk Factors

Risk is measured by two factors: Likelihood and Impact:

Likelihood

Likelihood measures the potential of a vulnerability being exploited. Ratings are given based on the difficulty of the attack, the available tools, attacker skill level, and client environment.

Impact

Impact measures the potential vulnerability's effect on operations, including confidentiality, integrity, and availability of client systems and/or data, reputational harm, and financial loss.

Scope

Assessment	Details
Low Security Internal Penetration Test	10.x.x.x/8

Scope Exclusions

Per client request, CRS did not perform any of the following attacks during testing:

- Denial of Service (DoS)
- Phishing/Social Engineering

All other attacks not specified above were permitted by DVWA.

Client Allowances

DVWA provided CRS the following allowances:

- Admin access to website
- Website security is set to low

Vulnerability Summary & Report Card

The following tables illustrate the vulnerabilities found by impact and recommended remediations:

Low Security Internal Penetration Test Findings

8	1	0	0	0
Critical	High	Moderate	Low	Informational

Finding	Severity	Recommendation
Low Security Internal Penetration Test		
LSIPT-001: Login Brute Force Attack	Critical	Set the admin password to be a strong password, Implement account lockout mechanisms after a few failed attempts, use CAPTCHA, enforce strong password policies, and monitor failed login attempts for brute-force patterns.
LSIPT-002: Command Injection	Critical	Properly sanitize and validate user input, and use parameterized queries. Restrict the execution of dangerous system commands and minimize the exposure of internal system functionality.
LSIPT-003: Cross Site Request Forgery	High	Use anti-CSRF tokens in forms and validate them on the server. Implement same-site cookies and prompt user confirmation for critical actions.
LSIPT-004: Reflected XSS	Critical	Sanitize and escape user input, and use appropriate content security policies (CSP).
LSIPT-005: Stored XSS	Critical	Sanitize inputs and outputs to prevent script injection. Implement proper HTML encoding.
LSIPT-006: DOM XSS	Critical	Sanitize inputs and avoid writing untrusted data directly to the DOM.
LSIPT-007: File Inclusion	Critical	Validate and sanitize file paths.

LSIPT-008: File Upload	Critical	Only allow specific file types (e.g., images) and reject dangerous file types such as .php, .exe, .js, etc. Analyze the content of uploaded files to ensure they match the expected type (e.g., checking MIME types). Store uploads in directories where they cannot be executed, such as outside the web root.
LSIPT-009: SQL Injection	Critical	Use parameterized queries and prepared statements to ensure user input is safely handled. Avoid constructing SQL queries directly from user input.

Technical Findings

Low Security Internal Penetration Test Findings

Finding LSIPT-001: Login Brute Force Attack (Critical)

Exploit Technique	Brute-forcing login credentials by trying multiple password combinations. Using Hydra tool to automate login attempts with a wordlist of commonly used passwords.
Exploit Payload	A hydra script hydra -l admin -P /snap/seclists/current/Passwords/500-worst-passwords.txt 127.0.0.1 http-get-form "/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:Username and/or password incorrect.:H=Cookie: PHPSESSID=497d9581a44a6588d9c6ce1ecbf2a9f2; security=low" -s 4280
Risk	An attacker can gain unauthorized access to the application by successfully guessing valid credentials.

Evidence

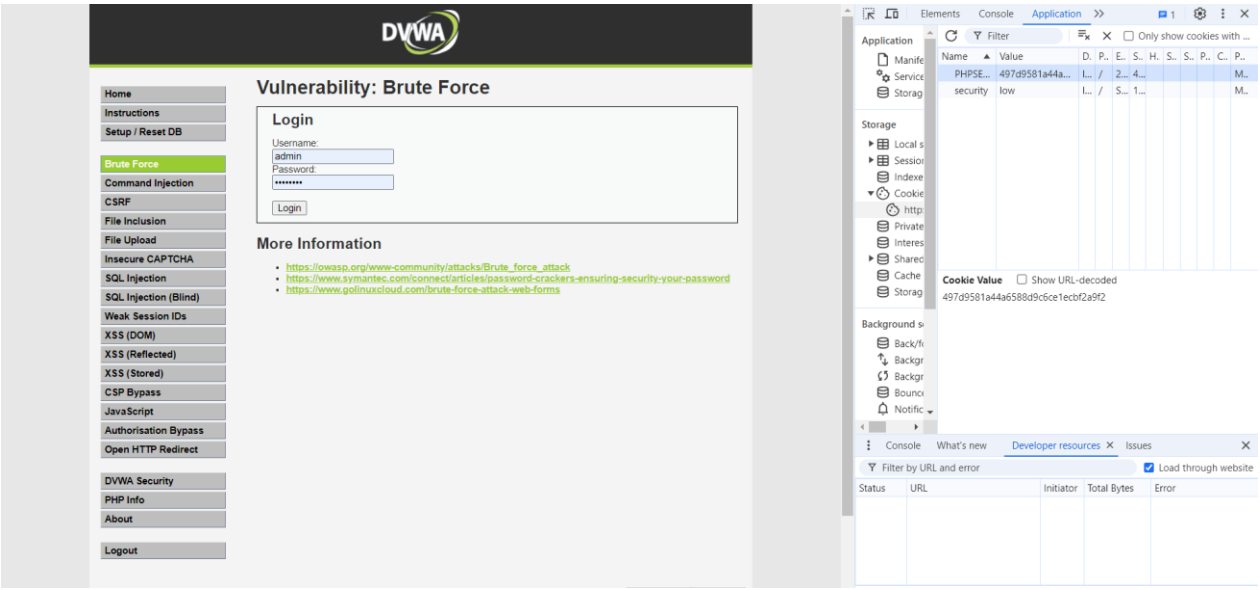


Figure 1: Gathering cookie

```
fajar@MSI ~/Documents/ITS/LBE/DVWA/DVWA git:master $ hydra -l admin -P /snap/seclists/current/Passwords/500-worst-passwords.txt 127.0.0.1 http-get-form "/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:Username and/or password incorrect.:H=Cookie: PHPSESSID=497d9581a44a6588d9c6ce1ecbf2a9f2; security=low" -s 4280
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-09-18 15:39:08
[DATA] max 16 tasks per 1 server, overall 16 tasks, 499 login tries (l:1/p:499), ~32 tries per task
[DATA] attacking http-get-form://127.0.0.1:4280/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:Username and/or password incorrect.:H=Cookie: PHPSESSID=497d9581a44a6588d9c6ce1ecbf2a9f2; security=low
[4280][http-get-form] host: 127.0.0.1   login: admin   password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-09-18 15:39:09
fajar@MSI ~/Documents/ITS/LBE/DVWA/DVWA git:master $
```

Figure 2: Exploit with hydra

Remediation

Set the admin password to be a strong password, Implement account lockout mechanisms after a few failed attempts, use CAPTCHA, enforce strong password policies, and monitor failed login attempts for brute-force patterns.

Finding LSIPT-002: Command Injection (Critical)

Exploit Technique	Command Injection vulnerability where unsanitized user input is executed as part of system commands. Injecting OS commands through input fields, for example.
Exploit Payload	Attacker append commands like ls /etc to explore the server's directory structure. 127.0.0.1 & ls /etc
Risk	This can lead to full system compromise if the attacker gains access to sensitive files or escalates privileges..

Evidence

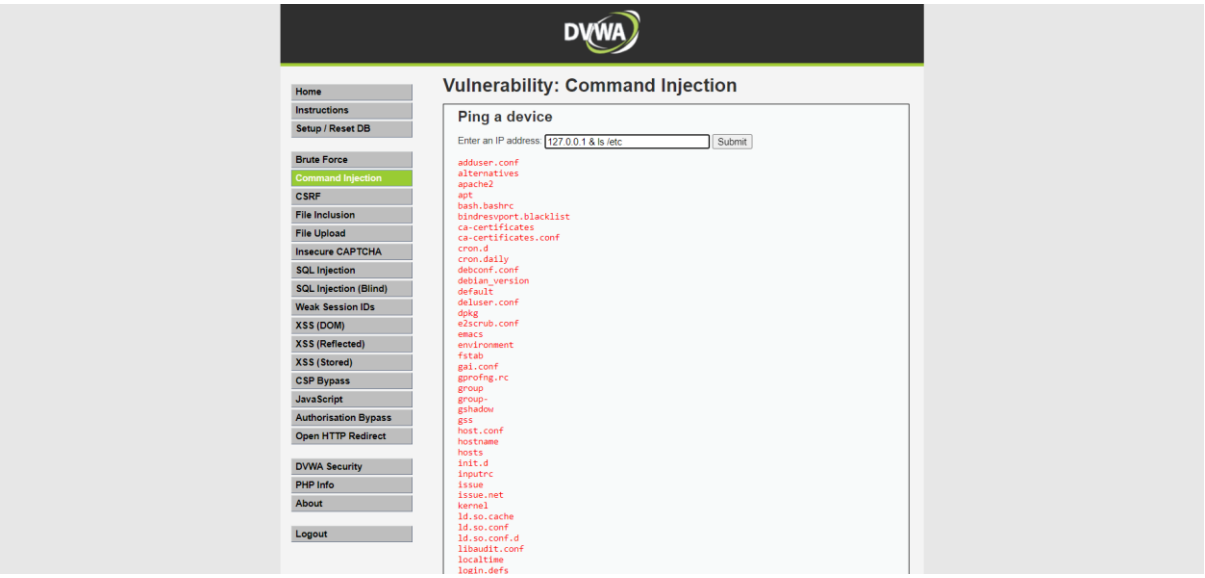


Figure 1: Sending malicious payload

Remediation

Properly sanitize and validate user input, and use parameterized queries. Restrict the execution of dangerous system commands and minimize the exposure of internal system functionality.

Finding LSIPT-003: Cross Site Request Forgery (High)

Exploit Technique	Cross-Site Request Forgery (CSRF) tricking an authenticated user into unknowingly executing unwanted actions on a web application.
Exploit Payload	A forged request that performs actions under the authenticated user's session.
Risk	Allows an attacker to perform actions like changing account details or transferring funds on behalf of the victim.

Evidence

```
<body>
    <form action="http://localhost:4280/vulnerabilities/csrf/?" method="GET">
        <input type="password" name="password_new"><br>
        <input type="password" name="password_conf"><br>
        <input type="hidden" name="Change" value="Change"><br>
        <input type="submit" value="Click ME!">
    </form>
</body>
```

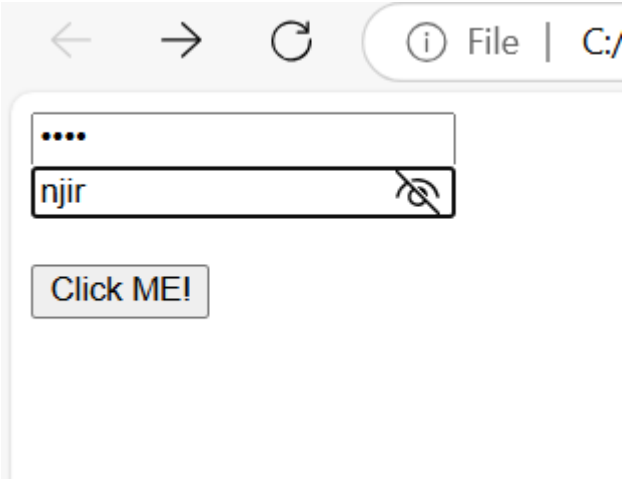


Figure 1: The bait malicious form for user

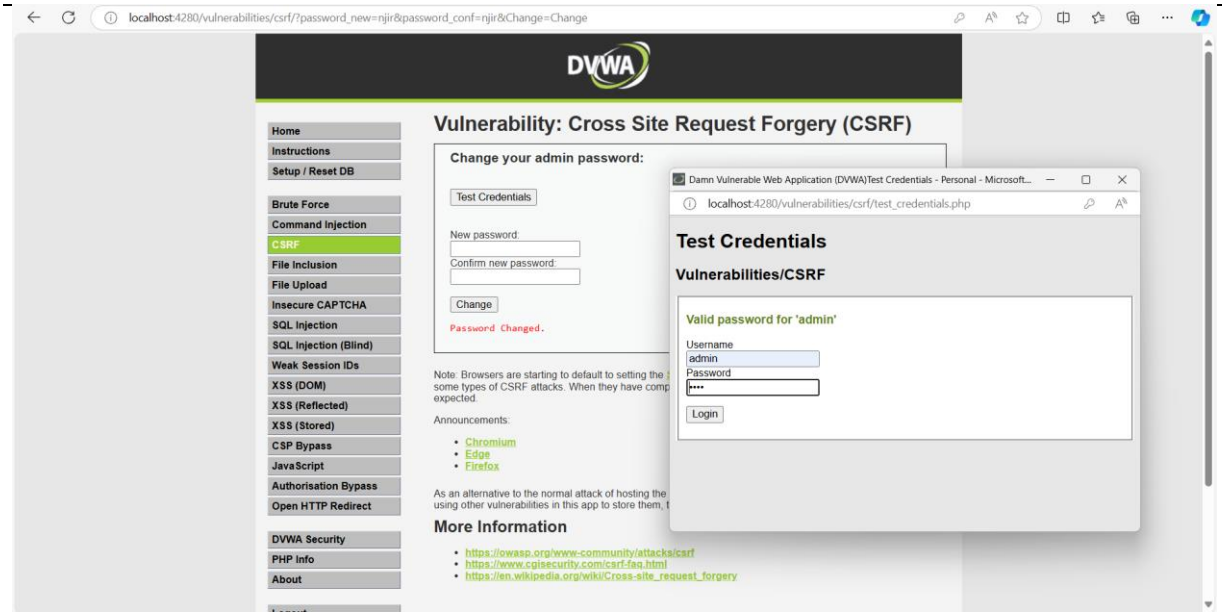


Figure 2: After user click, it redirects to the site and performs unwanted action

Remediation

Use anti-CSRF tokens in forms and validate them on the server. Implement same-site cookies and prompt user confirmation for critical actions.

Finding LSIPT-004: Reflected XSS (Critical)

Exploit Technique	Reflected XSS occurs when user input is immediately returned by the server without proper encoding. Injecting malicious scripts through URL or form fields that are reflected in the response.
Exploit Payload	Attacker inject JavaScript, which will be executed in the user's browser when they open the crafted URL.
Risk	Can lead to session hijacking, data theft.

Evidence

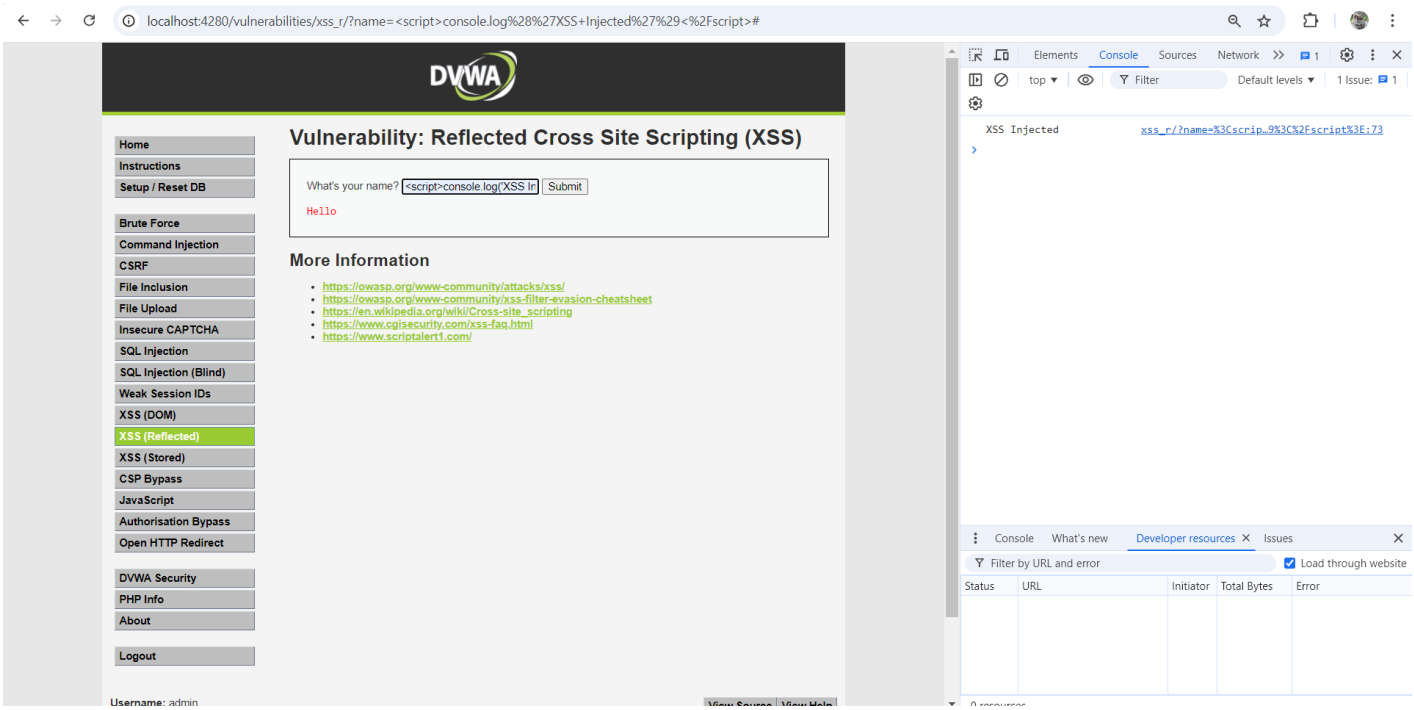


Figure 1: Url with a malicious params, that can execute javascript

```php
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Feedback for end user
    echo '<pre>Hello ' . $_GET[ 'name' ] . '</pre>';
}

?>
```

Figure 2: A vulnerable server code that just appends user params and returns it

Remediation

Sanitize and escape user input, and use appropriate content security policies (CSP).

Finding LSIPT-005: Stored XSS (Critical)

Exploit Technique	Malicious scripts are stored on the server and executed when users visit the affected pages. Injecting a script into a form that stores data (e.g., comment sections).
Exploit Payload	Attacker uploads a malicious payload with a script. The script is stored on the server and executed each time the data is loaded.
Risk	Persistent attacks where all users visiting the page can be affected, leading to widespread compromise.

Evidence

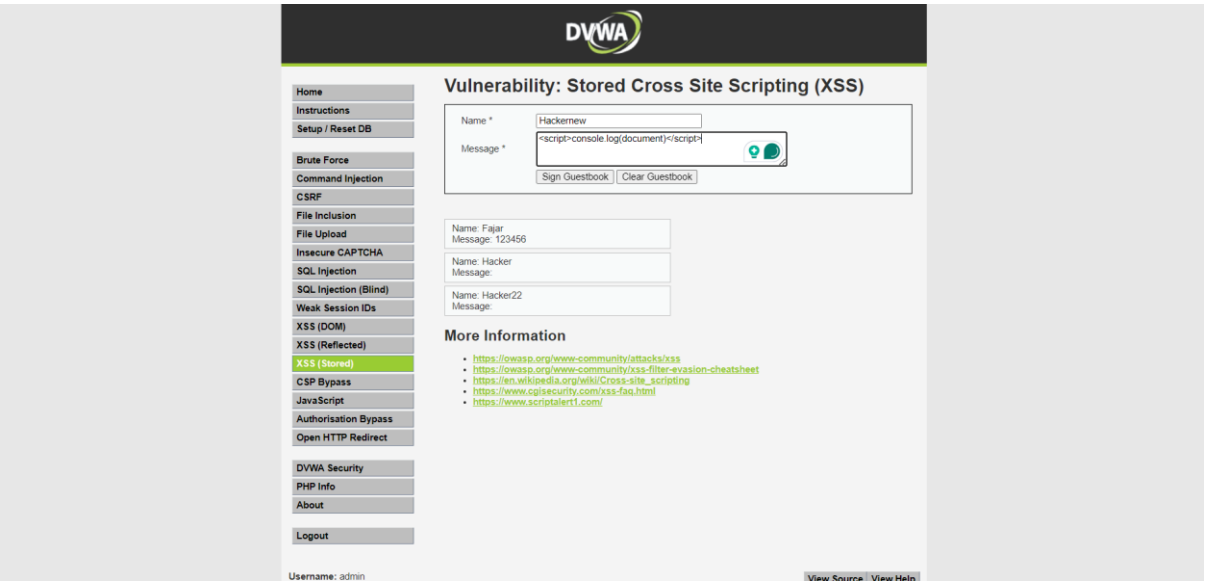


Figure 1: An attacker uploading a malicious payload

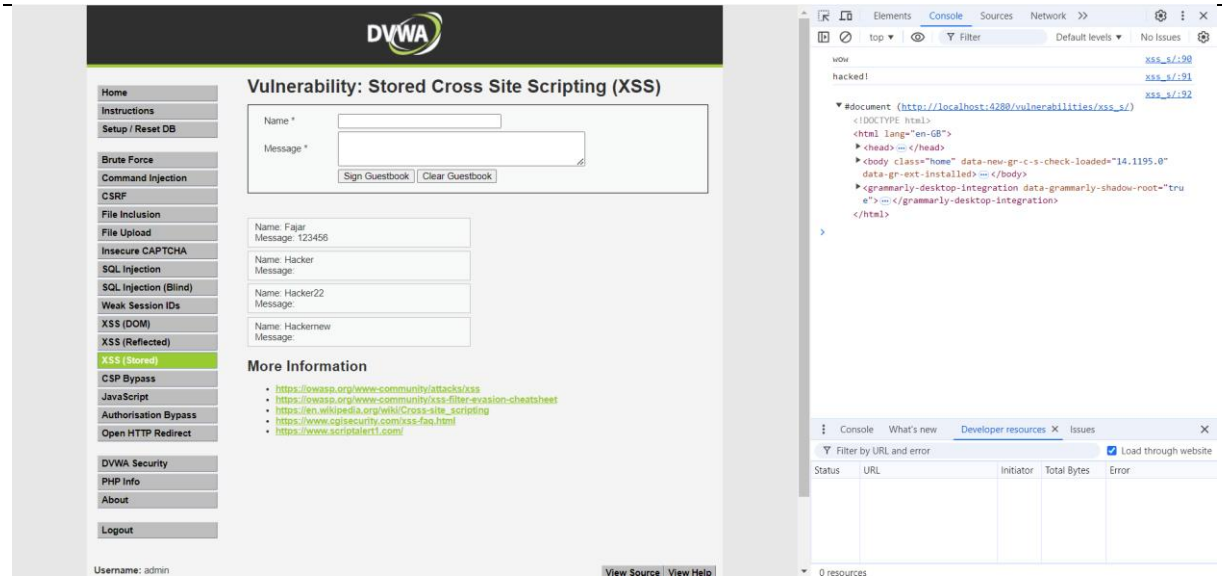


Figure 2: A user anytime it loads the content.

Remediation

Sanitize inputs and outputs to prevent script injection. Implement proper HTML encoding.

Finding LSIPT-006: DOM XSS (Critical)

Exploit Technique	A variant of XSS where the vulnerability is within the client-side JavaScript code. Manipulating the DOM (Document Object Model) of the page to execute malicious scripts.
Exploit Payload	Attacker manipulates the client-side script behavior to inject malicious content into the page.
Risk	Similar to other XSS types, can lead to session hijacking or data theft.

Evidence

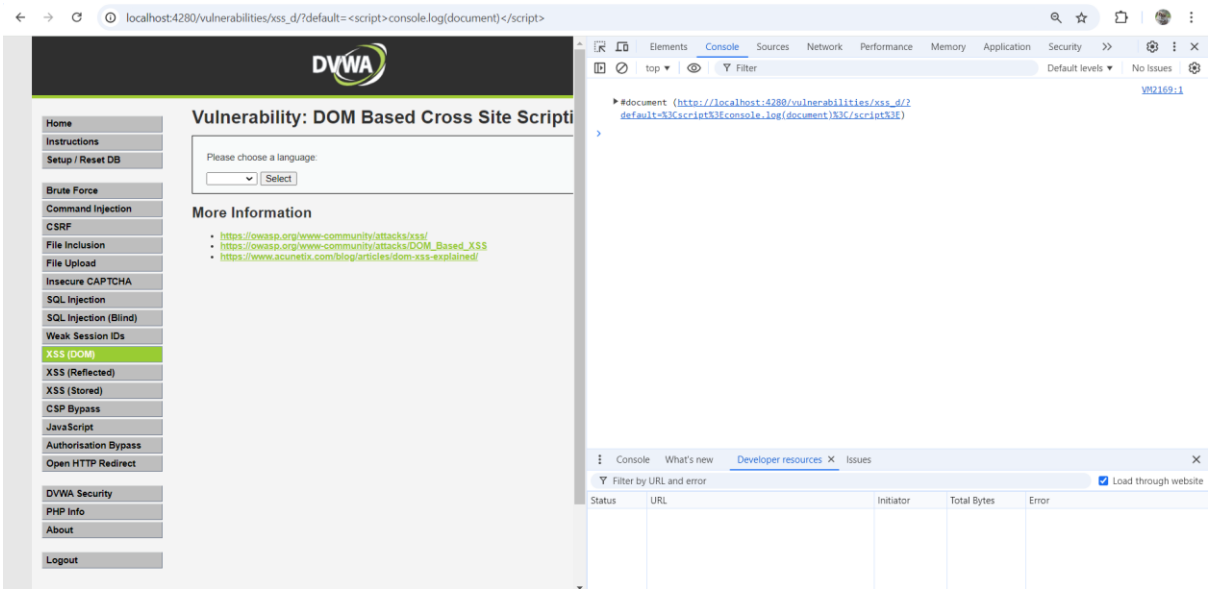


Figure 1: A user opening a malicious url

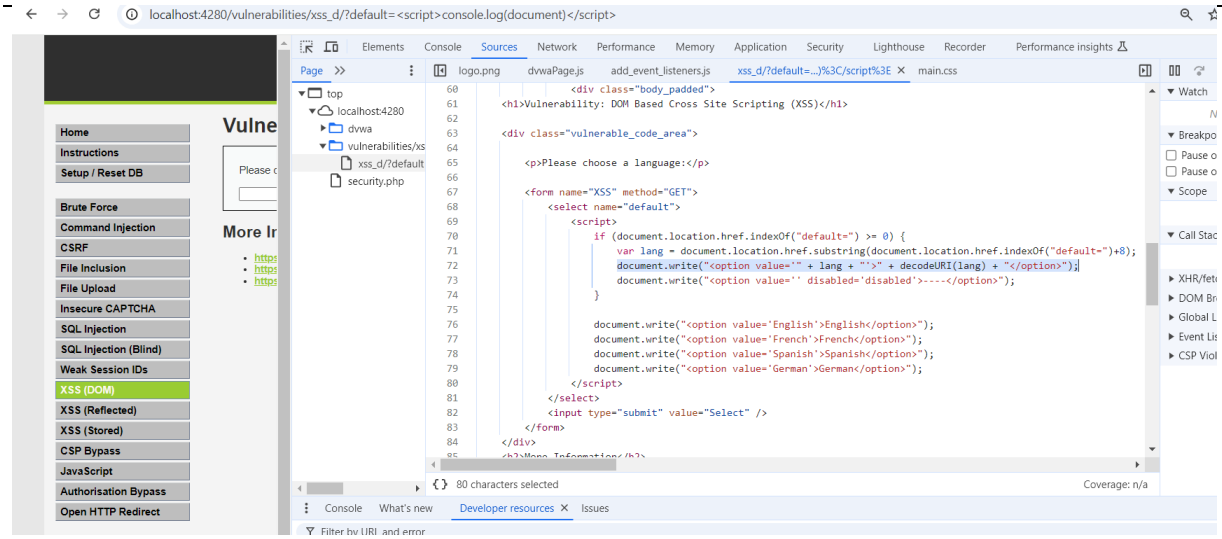


Figure 2: The malicious params is executed within the DOM when document.write

Remediation

Sanitize inputs and avoid writing untrusted data directly to the DOM.

Finding LSIPT-007: File Inclusion (Critical)

Exploit Technique	File inclusion vulnerabilities, allowing an attacker to include unauthorized files. The attacker includes remote or local files through vulnerable file paths.
Exploit Payload	Attacker can access sensitive local files with an unsanitized file request from a url.
Risk	Validate and sanitize file paths, restrict file types that can be uploaded, and disable the inclusion of files from external sources.

Evidence

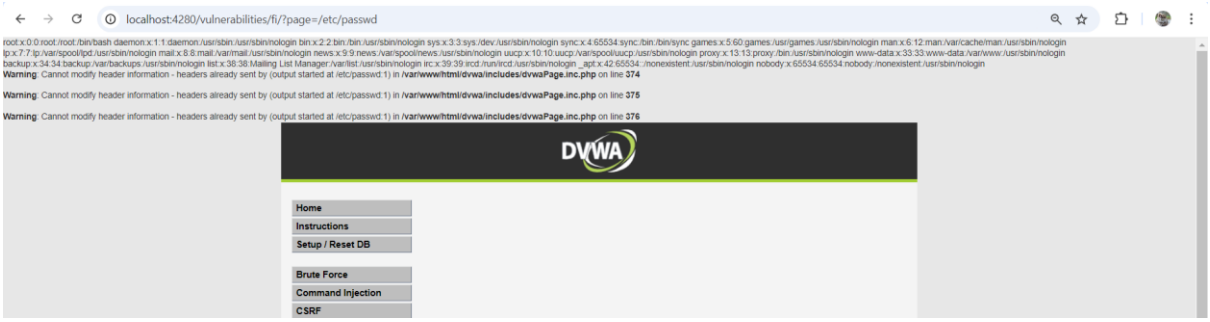


Figure 1: An attacker sending a malicious request, containing the file path from the url

Remediation

Validate and sanitize file paths.

Finding LSIPT-008: File Upload (Critical)

Exploit Technique	Unrestricted or insecure file upload where an attacker can upload files without proper validation. Uploading a malicious file (such as a script or executable) that the server processes or stores without verifying its content or type.
Exploit Payload	Uploads a file containing malicious code (e.g., a PHP shell script) that can be executed by visiting the file's URL.
Risk	Execute arbitrary code on the server. Compromise sensitive data. Gain complete control of the application or server by escalating privileges. Store malware that other users could download.

Evidence

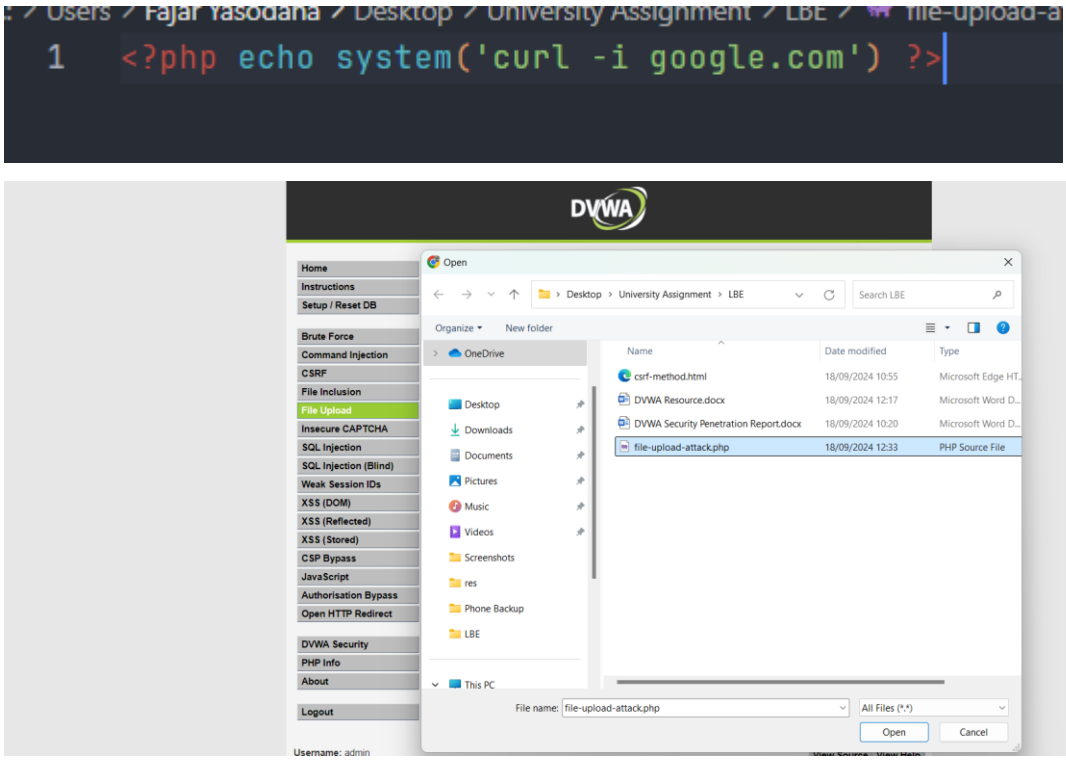


Figure 1: An attacker uploading a malicious file that can be executed by the server when reading it

Figure 2: The server reads the file, and executes the code in it. Then sends the sensitive response

Remediation

Only allow specific file types (e.g., images) and reject dangerous file types such as .php, .exe, .js, etc. Analyze the content of uploaded files to ensure they match the expected type (e.g., checking MIME types). Store uploads in directories where they cannot be executed, such as outside the web root.

Finding LSIPT-009: SQL Injection (Critical)

Exploit Technique	SQL Injection involves manipulating SQL queries through unsanitized user input.
Exploit Payload	Injecting SQL statements into a vulnerable form field or URL parameter.
Risk	Allows attackers to access or modify sensitive data, bypass authentication, or execute database commands.

Evidence

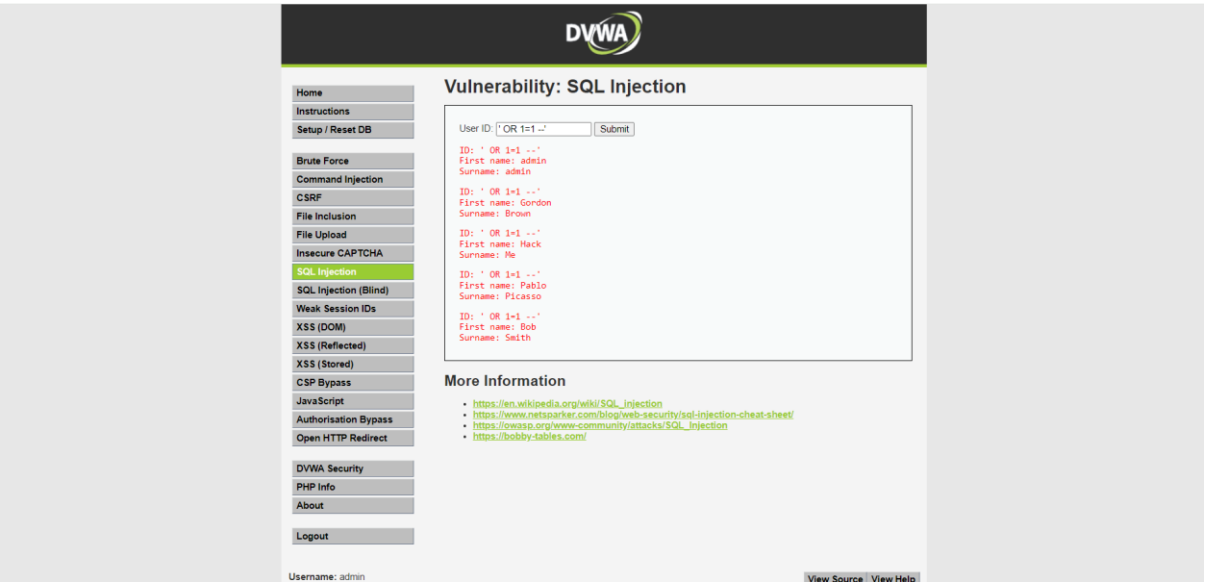


Figure 1: An attacker sending a malicious payload containing an SQL statement.

Remediation

Use parameterized queries and prepared statements to ensure user input is safely handled. Avoid constructing SQL queries directly from user input.

Last Page