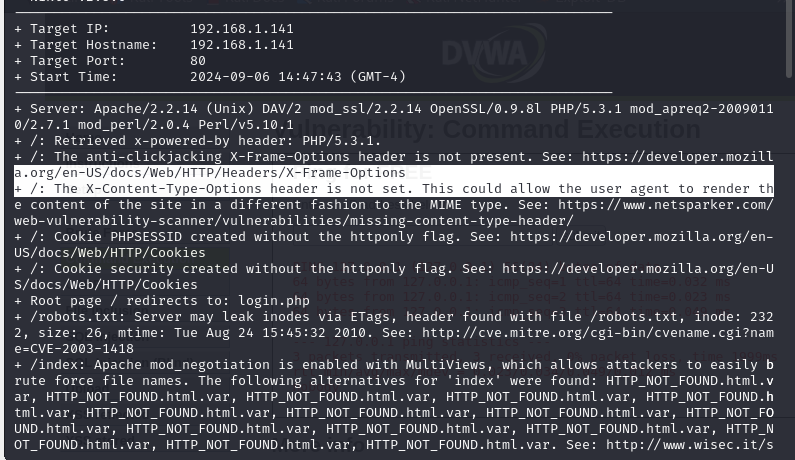
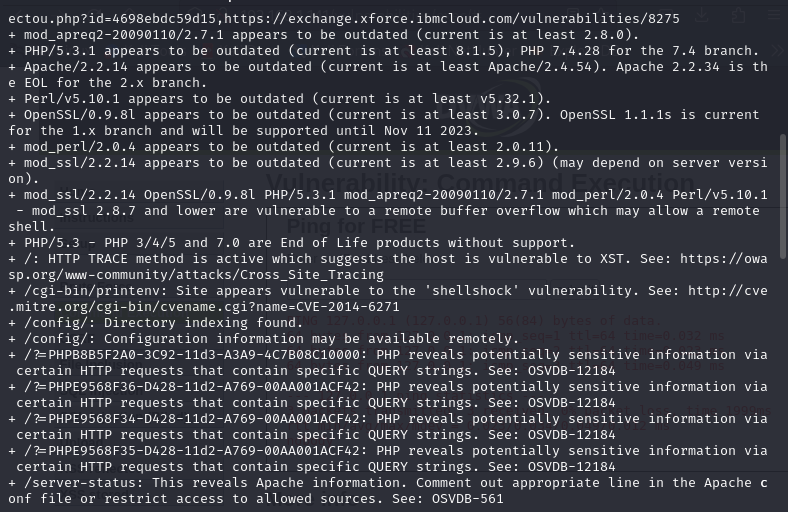
Pentesting Report

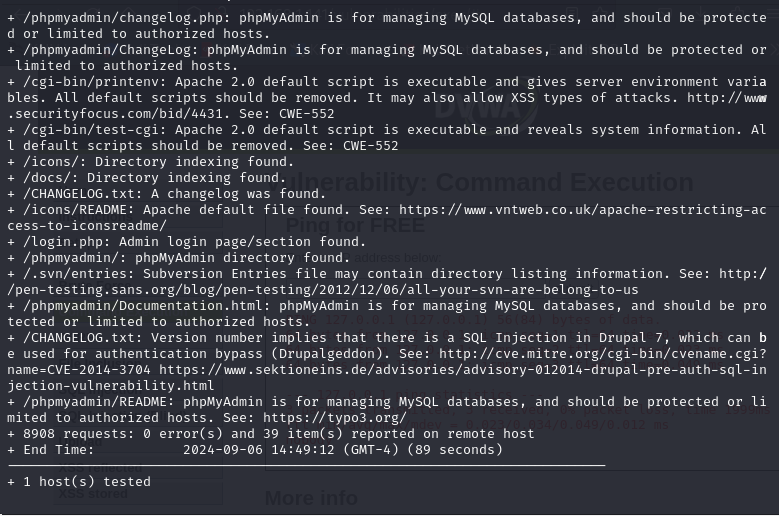
1. **Introduction**

This document provides the details of a penetration test performed on the Damn Vulnerable Web Application (DVWA). The purpose of the test is to identify security weaknesses and vulnerabilities, assess their risk levels, and offer recommendations for remediation. The tests conducted focused on multiple known web application vulnerabilities.

1. **Scan Results**

****

****

****

Following vulnerabilities were identified in the scan:

* Shellshock Vulnerability (CVE-2014-6271)

This vulnerability allows remote attackers to execute arbitrary code via crafted environment variables in certain services.

**Severity**: Critical

**Remediation :** Patch the Bash shell to a version that is not vulnerable to Shellshock.

* Remote Buffer Overflow in mod\_ssl (CVE-2003-1418)

mod\_ssl is vulnerable to a remote buffer overflow, which can lead to remote shell execution.

**Severity**: Critical

**Remediation**: Update mod\_ssl and OpenSSL to the latest versions.

* Cross-Site Tracing (XST) Vulnerability

HTTP TRACE method is active, which makes the host vulnerable to cross-site tracing attacks.

**Severity**: High

**Remediation**: Disable the TRACE method in the server configuration.

* Outdated PHP Version

The server is running an outdated and unsupported version of PHP (5.3.1), which has known vulnerabilities, including remote code execution.

**Severity**: High

**Remediation**: Update PHP to a supported version (e.g., PHP 8.1.x).

* Directory Indexing Enabled

Directory indexing is enabled, which allows attackers to list files and directories in the /config/, /icons/, and /docs/ directories.

**Severity**: High

**Remediation**: Disable directory indexing in the server configuration.

* Apache mod\_negotiation Brute Force

Apache’s mod\_negotiation with MultiViews is enabled, allowing attackers to brute force file names.

**Severity**: High

**Remediation**: Disable MultiViews in the Apache configuration.

* Missing HTTP Security Headers (X-Frame-Options, X-Content-Type-Options)

The server is missing key security headers, increasing the risk of clickjacking and MIME-type confusion attacks.

**Severity**: Medium

**Remediation**: Add the X-Frame-Options and X-Content-Type-Options headers.

* Cookies Missing HttpOnly Flag

Cookies PHPSESSID and security are missing the HttpOnly flag, which increases the risk of XSS attacks.

**Severity**: Medium

**Remediation**: Set the HttpOnly flag for session cookies to mitigate the risk of XSS.

* Outdated Apache Version

The server is running an outdated version of Apache (2.2.14), which has known vulnerabilities.

**Severity**: Medium

**Remediation**: Update Apache to the latest stable version (e.g., 2.4.54 or higher).

* Potential Sensitive Information Disclosure

PHP is revealing sensitive information via certain query strings.

**Severity**: Low

**Remediation**: Disable these PHP information disclosures or apply proper filtering.

* Outdated Perl and mod\_perl

The server is running outdated versions of Perl (v5.10.1) and mod\_perl (2.0.4), which may have security vulnerabilities.

**Severity**: Low

**Remediation**: Update Perl and mod\_perl to the latest versions.

# 2. Scope of Testing

The penetration testing included testing for the following vulnerabilities:  
1. Command Injection  
2. File Upload  
3. File Inclusion  
4. Reflected Cross-Site Scripting (XSS)  
5. Stored Cross-Site Scripting (XSS)  
6. SQL Injection  
7. Blind SQL Injection  
8. Brute Force  
9. Cross-Site Request Forgery (CSRF)

# 3. Testing Methodology

The testing process followed a structured methodology designed to identify security vulnerabilities in DVWA. The methodology included the following steps:  
1. Information Gathering  
2. Vulnerability Identification  
3. Exploitation (where applicable)  
4. Reporting  
Testing was conducted in a controlled environment using manual techniques and automated tools.

# 4. Summary of Findings

The table below summarizes the vulnerabilities identified during testing:

Vulnerability Severity CVSS Score  
-----------------------------------------------------------------  
Command Injection High 7.5  
File Upload High 7.1  
File Inclusion High 7.0  
Stored XSS Medium 6.5  
Reflected XSS Medium 6.1  
SQL Injection High 8.1  
Blind SQL Injection High 7.8  
Brute Force Low 4.0  
CSRF Medium 6.3

# 5. Results of Vulnerabilities Exploitation

# The vulnerabilities listed below were successfully exploited during the penetration test:

* **Brute Force**

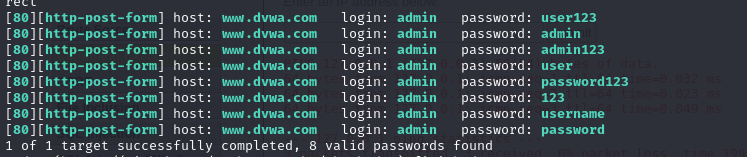
Severity: Low to Medium

CVSS Score: 5.0 - 6.0 (Medium)

Description: Repeatedly attempts to guess login credentials by trying different combinations of usernames and passwords.

Impact: Account compromise if weak passwords are used.

Remediation : Do not use default credentials and create strong passwords.

****

**Tool Used :** Hydra

* **Command Execution**

Severity: Critical

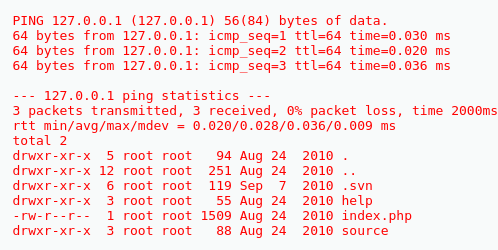
CVSS Score: 9.8 (Critical)

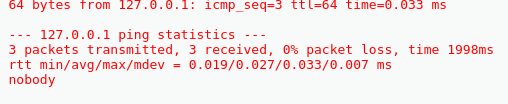
Description: Allows an attacker to execute arbitrary commands on the host operating system, potentially gaining full control.

Impact: Complete system compromise, privilege escalation.

Remediation : Use proper input sanitization methods and principle of least access/manage access controls properly.

127.0.0.1; ls -la





* **File Upload**

Severity: High

CVSS Score: 8.0 (High)

Description: If an attacker can upload files without validation, it can lead to remote code execution by uploading malicious files (like web shells).

Impact: Remote code execution, potential takeover of the web server.

Remediation : Ensure that only specific file types are allowed to be uploaded.Scan all uploaded files for malicious content using antivirus or malware scanners.Do not store uploaded files in directories where they can be executed.

Installing backdoors



* **File Inclusion**

Severity: Critical

CVSS Score: 9.0 - 9.5 (Critical)

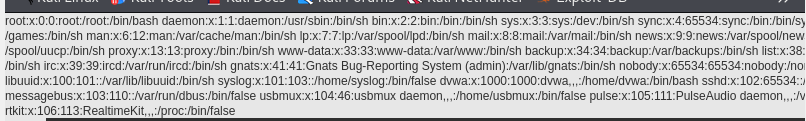
Description: Allows an attacker to include remote files for execution, leading to remote code execution.

Impact: Server compromise, possible access to sensitive data.

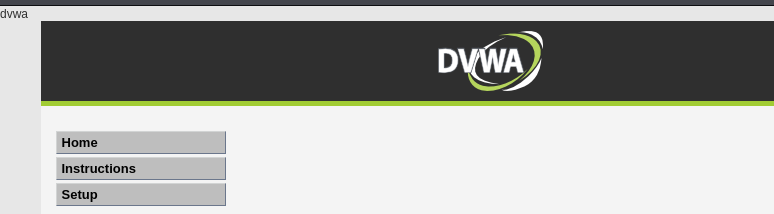
Remediation :

* Ensure that any file paths or filenames provided by the user are properly validated.
* Avoid dynamically constructing file paths based on user input.
* Disable the use of potentially dangerous functions such as include(), require(), eval(), and others in web applications

<http://192.168.1.141/vulnerabilities/fi/?page=../../../../../../../../../etc/passwd>



<http://192.168.1.141/vulnerabilities/fi/?page=../../../../../../../../../etc/hostname>



* **Reflected XSS**

Severity: Medium

CVSS Score: 6.1 (Medium)

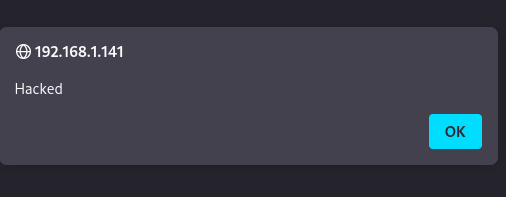
Description: Allows an attacker to inject malicious scripts into a webpage that gets reflected back to the user's browser.

Impact: Session hijacking, phishing, defacement.

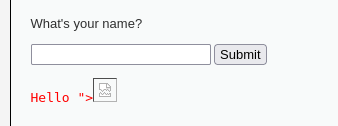
Remediation:

* Ensure that all user input is properly validated and filtered.
* Use a whitelist approach to only allow expected characters for each field, especially for fields used in URL parameters.
* Always encode user input before rendering it in the browser, especially in HTML, JavaScript, and URL contexts.

script>alert("Hacked")</script>



<img src=x onerror=alert(1)//>



* **Stored XSS**

Severity: Medium

CVSS Score: 7.5 (High)

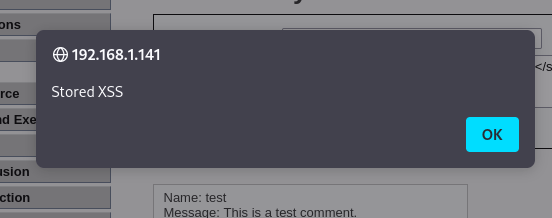
Description: The malicious script is stored on the server and executed when a user views the page. This has a higher impact than reflected XSS.

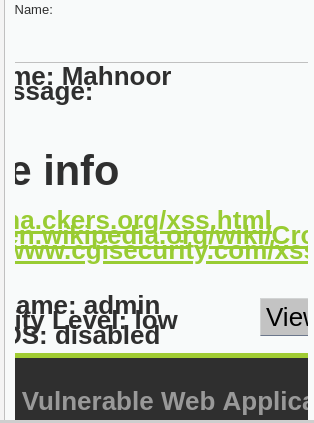
Impact: More persistent attacks, session hijacking, possible access to sensitive information.

Remediation :

* Implement strict input validation on all fields that accept user input.
* Use a whitelist approach to only allow expected characters and reject any malicious input.

<marquee><h1><script>alert('Stored XSS')</script></h1></marquee>





* **Cross Site Request Forgery (CSRF)**

Severity: Medium

CVSS Score: 6.5 (Medium)

Description: Tricks a user into submitting requests unknowingly, potentially performing actions on behalf of the attacker.

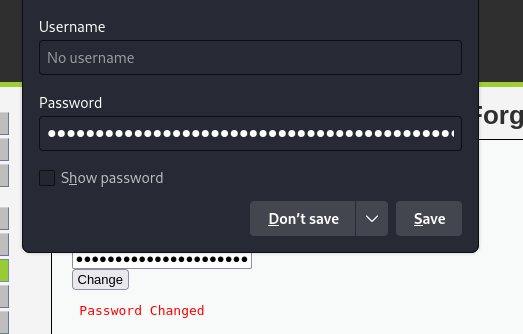
Impact: Account takeover, unintended actions on the victim's behalf.

script :

var script = document.createElement("script");

script.src = "https://example.com/change-email?callback=csrf&new\_email=attacker@example.com";

document.body.appendChild(script);



# 6. Recommendations

**Patch Management**:

* Update all outdated software components (Apache, PHP, OpenSSL, Perl, mod\_ssl) to the latest supported versions. These components contain known vulnerabilities that must be patched.

**Input Validation and Output Encoding**:

* Implement strict input validation and output encoding to prevent vulnerabilities such as SQL Injection, XSS (Reflected and Stored), and Command Injection.

**Security Headers**:

* Add the missing X-Frame-Options, X-Content-Type-Options, and Strict-Transport-Security headers to prevent clickjacking, MIME-type confusion attacks, and ensure secure communications.

**File Upload and Inclusion**:

* Implement file validation mechanisms to ensure only authorized and safe file types can be uploaded. Disable directory indexing and prevent File Inclusion vulnerabilities by validating input paths and disabling dangerous functions like eval() and include().

**CSRF Protection**:

* Implement CSRF tokens on all forms and verify them on the server side to mitigate CSRF attacks.

**Session Management**:

* Set secure attributes on session cookies (HttpOnly and Secure) to prevent them from being accessible to scripts and being transmitted over insecure connections.

**Brute Force Protection**:

* Implement rate-limiting and account lockout mechanisms after repeated failed login attempts to mitigate brute-force attacks.

# 7. Conclusion

This penetration test identified several high-risk and critical vulnerabilities in the DVWA environment, including command injection, file inclusion, SQL injection, and outdated software. Immediate action is required to patch outdated components, implement input validation, and harden security configurations.

By addressing these issues, DVWA will significantly reduce its attack surface and enhance its overall security posture.