

External Web Application Penetration Testing

PREPARED FOR:

MASAR Cybersecurity Training Program



Confidential

DOCUMENT VERSION CONTROL

DOCUMENT VERSION CONTROL

Data Classification – Client Confidential

Client Name	<i>MASAR Cybersecurity Training Program</i>
Project name	VlunShop With OWASP Top 10
Authors	Qasim Thiabat & Abdalrahman Almary
Approved by	
Version	<i>1.0</i>
Submission Date	<i>August 27, 2025</i>

Table of Contents

Section 1: Executive Summary	4
1.2 Results Summary	5
Section 2: Detailed Penetration Testing Results	7
2.1 Introduction	7
2.2 Restrictions	7
2.3 Tools	7
2.4 High Risk Exploitable Vulnerabilities	7
2.5 Medium Risk Exploitable Vulnerabilities	9
2.6 Low Risk Exploitable Vulnerabilities	9
2.7 CONCLUSION	14

Section 1: Executive Summary

This report presents the results of a security assessment conducted on the VulnShop Training Application, a deliberately vulnerable web application designed to demonstrate the OWASP Top 10 (2021) risks. The scope of this project focused on identifying, exploiting, and documenting the first five vulnerabilities from the OWASP list:

A01: Broken Access Control

A02: Cryptographic Failures

A03: Injection

A04: Insecure Design

A05: Security Misconfiguration. The penetration testing was carried out between **August 25, 2025**, and **August 27, 2025**.

A series of tests were conducted against the in-scope assets using industry-standard testing tools and, where appropriate, manual testing techniques. The objective was to identify and validate actual or potentially exploitable security vulnerabilities that, if exploited, could result in direct or indirect damage to the target environment.

The key findings identified during this engagement have been categorized based on severity levels — **High**, **Medium**, and **Low**. Detailed descriptions of these findings, their associated risks, and recommended remediation steps are provided in the “Detailed Penetration Testing Results” section of this report.

The following represents the definition and the description of each severity rate.

Impact	Description
High	A vulnerability that can be exploited by the attacker and cause huge damage to application components and data.
Medium	A vulnerability that can be exploited by the attacker and cause moderate damage to application components and data.
Low	A vulnerability that can be exploited by the attacker to understand application underlying technologies and versions which can be utilized in further attacks.

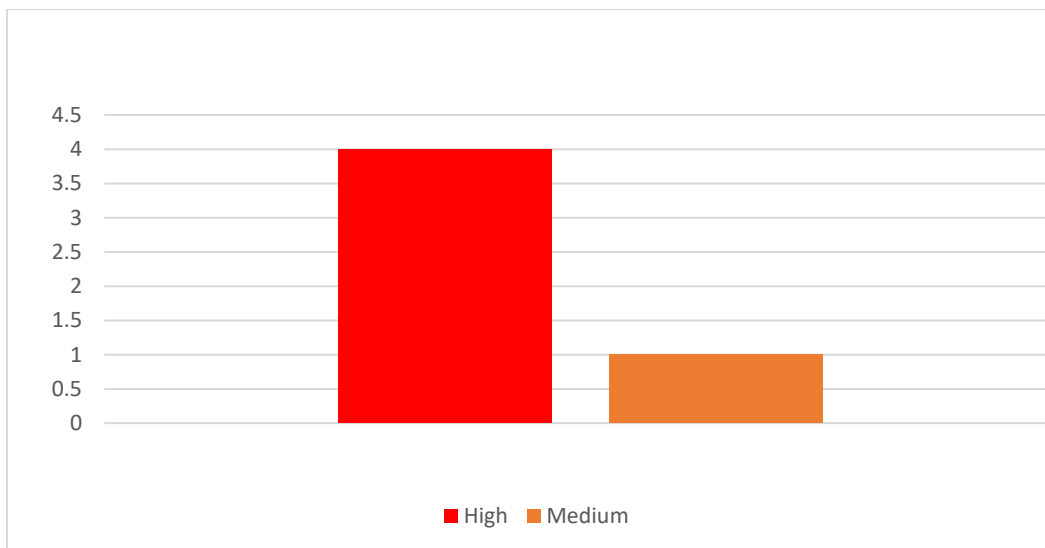
1.1 Scope Details

The scope of evaluation and testing covered the following assets:

#	Host	Platform
1	https://github.com/Q2004D/Vulnshop-OWASP-Top-10	Web Application

1.2 Results Summary

Below is the graphical representation of total identified vulnerabilities during the penetration testing service. These vulnerabilities are classified based on the severity in variant color codes as shown below.



The following table represents the identified vulnerabilities along with the severity of each.

Vul. Ref.	Vulnerability	Severity
A01:2021	Broken Access Control	High
A02:2021	Cryptographic Failures	High
A03:2021	Injection	High
A04:2021	Insecure Design	Medium
A05:2021	Security Misconfiguration	High

Section 2: Detailed Penetration Testing Results

2.1 Introduction

This section provides a detailed description of the vulnerabilities identified in the VulnShop Training Application during the security assessment. Each vulnerability has been mapped to the OWASP Top 10 (2021) and is presented with a clear explanation of the weakness, severity rating, potential business impact, affected systems, proof of exploitation, and recommended remediation steps. The vulnerabilities included in this section are limited to the first five OWASP Top 10 categories (A01–A05), as the scope of the project was focused on demonstrating the most common and impactful risks in web applications. All findings were validated through both manual testing and controlled exploitation in order to ensure accuracy and avoid false positives.

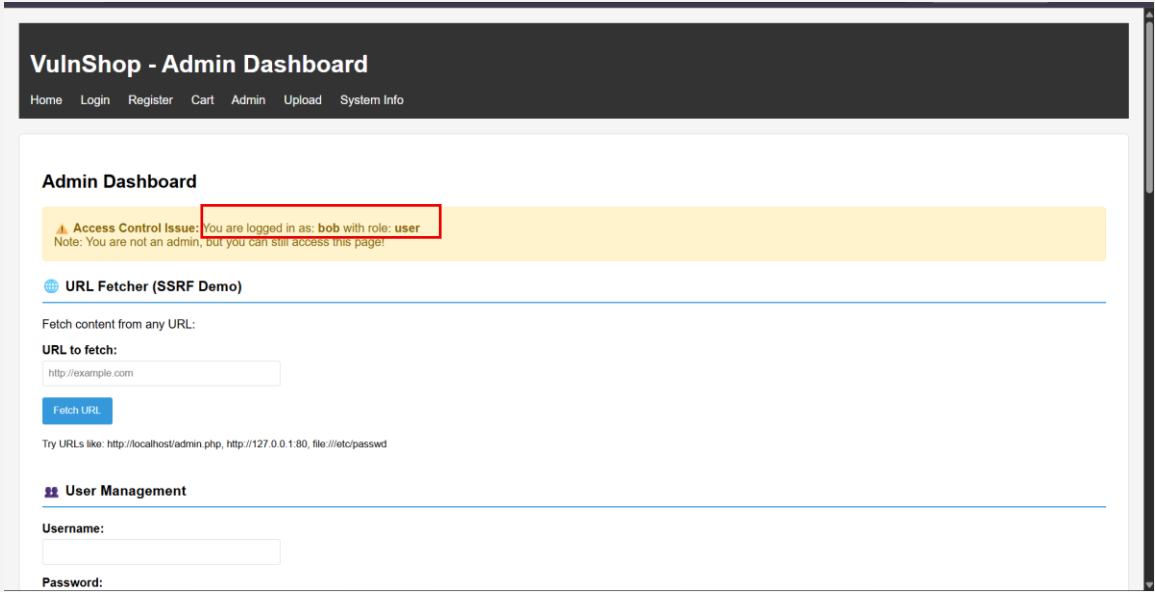
2.2 Restrictions

2.3 Tools

- 1- Linux Kali.
- 2- Wpscan.
- 3- Whatweb.
- 4- Curl.
- 5- Burb suite
- 6- nmap
- 7- Feroxbuster.

2.4 High Risk Exploitable Vulnerabilities

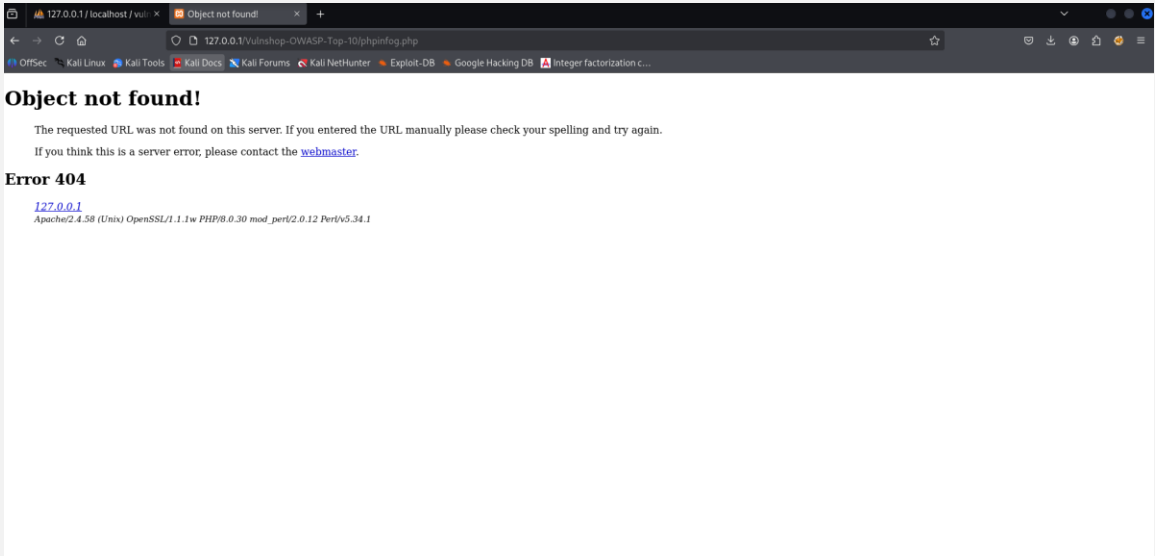
The following section represents **High** exploitable vulnerabilities that were identified during Penetration testing service.

Reference	A01:2021
Vulnerability	Insufficient access control on administrative and sensitive functions
Description	The application does not properly enforce role-based access control. Any authenticated user can directly access restricted pages such as admin.php or upload.php. Additionally, during registration (register.php), a normal user can escalate privileges by selecting the Administrator role without any server-side validation.
Severity	High
Impact	Unauthorized users can gain administrative privileges. Attackers can add/edit users, manage products, and view sensitive information such as plaintext passwords. Full compromise of the application, leading to potential data leakage and system manipulation.
Affected Systems	admin.php upload.php register.php
Recommendation	Implement proper Role-Based Access Control (RBAC). Enforce role checks on the server side instead of relying on client-side validation Remove the ability for users to self-assign the admin role during registration. Introduce middleware or access control mechanisms that validate user roles before granting access to sensitive pages.
Exploitation Results	 The screenshot shows the 'VulnShop - Admin Dashboard' interface. At the top, there is a navigation bar with links: Home, Login, Register, Cart, Admin, Upload, and System Info. Below this, the main content area is titled 'Admin Dashboard'. A yellow warning box is present, stating: 'Access Control Issue: You are logged in as: bob with role: user. Note: You are not an admin, but you can still access this page!'. Below the warning, there are two sections: 'URL Fetcher (SSRF Demo)' and 'User Management'. The 'URL Fetcher' section has a text input field labeled 'URL to fetch:' containing 'http://example.com' and a 'Fetch URL' button. Below it, it says 'Try URLs like: http://localhost/admin.php, http://127.0.0.1:80, file:///etc/passwd'. The 'User Management' section has input fields for 'Username:' and 'Password:'.

Reference	A02:2021
-----------	----------

Vulnerability	Plaintext password storage and exposure of sensitive information.																																	
Description	The application stores and displays user passwords in plaintext without any hashing or encryption. In multiple sections (e.g., admin.php and register.php), user credentials are directly inserted into the database and later shown to administrators. Additionally, sensitive configuration information (database credentials, error messages, phpinfo.php) is exposed to any user without restrictions.																																	
Severity	High																																	
Impact	Attackers gaining access to the database can immediately view and reuse user credentials. Exposure of plaintext passwords increases the risk of credential reuse attacks across other systems. Leaked configuration data (e.g., DB credentials, PHP info) can assist attackers in further exploiting the system.																																	
Affected Systems	register.php (inserts plaintext password into database) login.php (verifies passwords directly from plaintext storage) admin.php (displays passwords of all users in a table) config.php (hardcoded DB credentials + error messages) phpinfo.php (full system configuration disclosure)																																	
Recommendation	Use strong hashing algorithms (e.g., bcrypt, Argon2) to store passwords securely Never display user passwords in any interface. Remove hardcoded credentials from source code and use secure configuration management. Disable error display in production and restrict access to system information pages like phpinfo.php.																																	
Exploitation Results	<div><div>Existing Users:</div><table><tr><th>ID</th><th>Username</th><th>Password</th><th>Role</th><th>Created</th></tr><tr><td>1</td><td>admin</td><td>admin123</td><td>admin</td><td>2025-08-24 14:35:48</td></tr><tr><td>2</td><td>john</td><td>password123</td><td>user</td><td>2025-08-24 14:35:48</td></tr><tr><td>3</td><td>alice</td><td>alice456</td><td>user</td><td>2025-08-24 14:35:48</td></tr><tr><td>4</td><td>bob</td><td>12345</td><td>user</td><td>2025-08-24 14:35:48</td></tr></table><div><div>System Information</div><div><div>View PHP Info</div><div>File Upload</div></div><div>Server Details:</div><table><tr><td>Server Software</td><td>Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12</td></tr><tr><td>PHP Version</td><td>8.2.12</td></tr><tr><td>Document Root</td><td>C:/xampp/htdocs</td></tr><tr><td>Server Name</td><td>localhost</td></tr></table></div></div>	ID	Username	Password	Role	Created	1	admin	admin123	admin	2025-08-24 14:35:48	2	john	password123	user	2025-08-24 14:35:48	3	alice	alice456	user	2025-08-24 14:35:48	4	bob	12345	user	2025-08-24 14:35:48	Server Software	Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12	PHP Version	8.2.12	Document Root	C:/xampp/htdocs	Server Name	localhost
ID	Username	Password	Role	Created																														
1	admin	admin123	admin	2025-08-24 14:35:48																														
2	john	password123	user	2025-08-24 14:35:48																														
3	alice	alice456	user	2025-08-24 14:35:48																														
4	bob	12345	user	2025-08-24 14:35:48																														
Server Software	Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12																																	
PHP Version	8.2.12																																	
Document Root	C:/xampp/htdocs																																	
Server Name	localhost																																	

Reference	A03:2021
-----------	----------

Reference	A05:2021
Vulnerability	Exposed configuration files, unnecessary features enabled, and verbose error messages.
Description	The application and server environment expose sensitive information due to weak configuration. For example, phpinfo.php is publicly accessible, revealing detailed server configuration. Database credentials are stored in plaintext inside config.php. Additionally, verbose PHP error messages are displayed when queries fail, leaking database structure and query details. These weaknesses make it easier for attackers to discover entry points and exploit vulnerabilities.
Severity	High
Impact	Exposure of server environment (PHP version, extensions, document root). Disclosure of database credentials from configuration files. Attackers can gather detailed knowledge of the system, making exploitation easier.
Affected Systems	phpinfo.php (reveals full server configuration). config.php (hardcoded plaintext DB credentials). Any PHP page with query failure (shows verbose error messages).
Recommendation	Remove or restrict access to diagnostic files like phpinfo.php. Do not store credentials in plaintext inside application code; use environment variables or secure secrets management. Disable verbose error messages in production (set display_errors=0 and use error logging instead). Follow the principle of least functionality: disable unused services and features.
Exploitation Results	

2.5 Medium Risk Exploitable Vulnerabilities

The following section represents **Medium** exploitable vulnerabilities that were identified during Penetration testing service.

Reference	A04:2021
Vulnerability	Lack of validation on business logic, allowing abnormal operations.
Description	The application does not enforce proper restrictions on business logic operations, such as product quantity and pricing. Users can manipulate requests (e.g., cart or checkout forms) to perform actions outside the intended workflow, such as ordering negative quantities, setting arbitrary prices, or bypassing validation checks. This results from insecure design choices rather than simple coding mistakes.
Severity	Medium
Impact	Attackers can purchase items for free by manipulating product price in the request Negative quantities can cause incorrect stock levels or even add balance to the user's account. Business workflow can be bypassed, leading to financial loss or database inconsistencies.
Affected Systems	cart.php (no validation on product quantity → accepts negative or very large numbers). checkout.php (does not revalidate total price on the server side).
Recommendation	Implement strict server-side validation for all business logic (quantities, prices, totals). Never rely solely on client-side validation (JavaScript). Use constraints in the database (e.g., ensure product quantity ≥ 0). Conduct threat modeling during the design phase to identify misuse cases.
Exploitation Results	<pre>1 POST /Vulnshop-OWASP-Top-10/cart.php HTTP/1.1 2 Host: 127.0.0.1 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Content-Type: application/x-www-form-urlencoded 8 Content-Length: 36 9 Origin: http://127.0.0.1 10 Connection: keep-alive 11 Referer: http://127.0.0.1/Vulnshop-OWASP-Top-10/index.php 12 Cookie: PHPSESSID=7df6gd3uspf2qp584gmji4idn3 13 Upgrade-Insecure-Requests: 1 14 Sec-Fetch-Dest: document 15 Sec-Fetch-Mode: navigate 16 Sec-Fetch-Site: same-origin 17 Sec-Fetch-User: ?1 18 Priority: u=0, i 19 20 product_id=3&quantity=-10&add_to_cart=</pre>

VulnShop - Shopping Cart

[Home](#) [Login](#) [Register](#) [Cart](#) [Admin](#) [Upload](#) [System Info](#)

Your Shopping Cart

Added to cart successfully!

Product	Price	Quantity	Subtotal	Actions
Headphones Premium wireless headphones...	\$199.99	<div><div>-10</div><div>50</div></div> <div>Available: 50 ▲ Negative quantity!</div>	\$-1,999.90	<div>Remove</div>
Total:			\$-1,999.90	

[Continue Shopping](#)

[Update Cart](#) [Checkout \(\\$-1,999.90\)](#)

CONCLUSION

The VulnShop Training Application successfully demonstrated how common security flaws from the OWASP Top 10 (2021) can be exploited in real-world scenarios. By focusing on the first five vulnerabilities (A01–A05), the project highlights the most prevalent and impactful weaknesses found in modern web applications: poor access control, weak cryptography, injection flaws, insecure design, and misconfiguration.

Although this project intentionally stops at A05, these examples are sufficient to convey the critical importance of secure coding practices, regular configuration reviews, and implementing defense-in-depth strategies. Addressing these issues in production systems would significantly strengthen overall security posture and reduce exposure to cyberattacks.

END OF REPORT