CYNICAL

cynicaltechnology.com

VULNERABILITY ASSESSMENT AND PENETRATION TESTING

REPORT

07/07/2025

Prepared For:

MCFOSTER INC.
New York, USA

Prepared By:

CYNICAL TECHNOLOGY PVT. LTD
Kathmandu, Nepal

Table of Contents

Confidentiality Statement

This document contains information that is confidential and proprietary, which shall not be disclosed outside McFoster Inc. transmitted, duplicated, or used in whole or in part for any purpose other than its intended purpose. Any use or disclosure in whole or in part of this information without explicit written permission of McFoster Inc. is prohibited Cynical Technology Pvt. Ltd. makes no warranty that the information contained in this document is complete or error-free.

The specific scope was identified by McFoster Inc. our subsequent test work, study of issues in detail, and developing action plans are directed towards the issues identified. Consequently, this report may not necessarily comment on all the weaknesses perceived as important by McFoster Inc.

Disclaimer

The issues identified and proposed action plans in this report are based on our testing. We made specific efforts to verify the accuracy and authenticity of the applications. The identification of the issues in the report is mainly based on the tests carried out during the limited time. The outcome of the analysis may not be exhaustive and represent all possibilities, though we have taken reasonable care to cover the major eventualities.

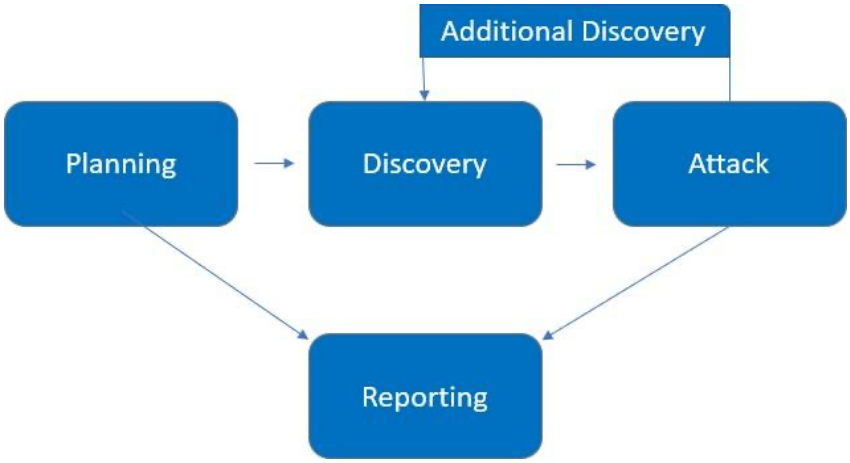
The vulnerabilities reported in this report are valid as July 7, 2025. Any vulnerability that may have been discovered after this or any exploitation does not come under the preview of this report. Any configuration changes or software/hardware updates made on hosts/machines on the application covered in this test after the date mentioned may impact the security posture either positively or negatively and hence invalidate the claims & observations in this report. Whenever there is an update on the application, we recommend that you conduct a penetration test to ensure that your security posture is unaffected.

Assessment Overview

McFoster Inc. engaged Cynical Technology Pvt. Ltd. to perform a security assessment of their web application to ensure the security of their user's data, personal information, and security of their infrastructure. The purpose of the assessment was to utilize active exploitation techniques to evaluate the application and security against best practice criteria to validate its security mechanisms and identify application-level vulnerabilities as well.

Phases of penetration testing activities include the following:

- Planning – Customer goals are gathered, and rules of engagement are 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 can lead to full system compromise. Patch immediately with a clear action plan.
High	7.0-8.9	Exploitation could cause elevated privileges and potentially a loss of data or downtime
Medium	4.0-6.9	Exploitation may require extra steps such as social engineering.
Low	0.1-3.9	Vulnerabilities are non-exploitable but would reduce an organization's attack surface.

Scope

ASSET	URL	ASSESSMENT TYPE
Web Application	http://testphp.vulnweb.com	Blackbox

Key Strengths and Weaknesses

The web application utilizes up-to-date components, ensuring that there are no inherent Common Vulnerabilities and Exposures (CVEs) affecting the application at the time of the assessment. This reflects a proactive approach to maintaining component security and reducing the risk associated with outdated vulnerabilities. This system has been implemented properly, ensuring that users have appropriate access rights, which is crucial for maintaining the security integrity of the application. However, the testing did reveal exploitable vulnerabilities in the application.

Despite the overall effective implementation of access control mechanisms, some issues were identified in the web application's access control system. These issues could potentially lead to unauthorized access if not addressed promptly.

Any vulnerabilities in the web application itself could lead to significant security breaches, suggesting a need for additional security layers and monitoring around this critical component.

Scope Exclusions

Per the client's request, Cynical Technology Pvt. Ltd. did not perform any of the following attacks during testing:

- Denial of Service (DoS)
- Social Engineering

All other attacks not specified above in exclusions were permitted by McFoster Inc.

Client Allowances

McFoster Inc. provided Cynical Technology Pvt. Ltd. with the following allowances:

- Web Application

Executive Summary

The purpose of the engagement was to utilize active exploitation techniques to evaluate the security of the application against best practice criteria validate its security mechanisms and identify application-level vulnerabilities.
The following sections provide a high-level overview of vulnerabilities discovered, successful and unsuccessful attempts, and strengths and weaknesses.

Scoping and Time Limitations

Scoping during the engagement did not permit denial of service or social engineering across all testing components.

Time limitations were in place for testing. Vulnerability assessment and penetration testing were done for 30 business days.

Meeting Logs

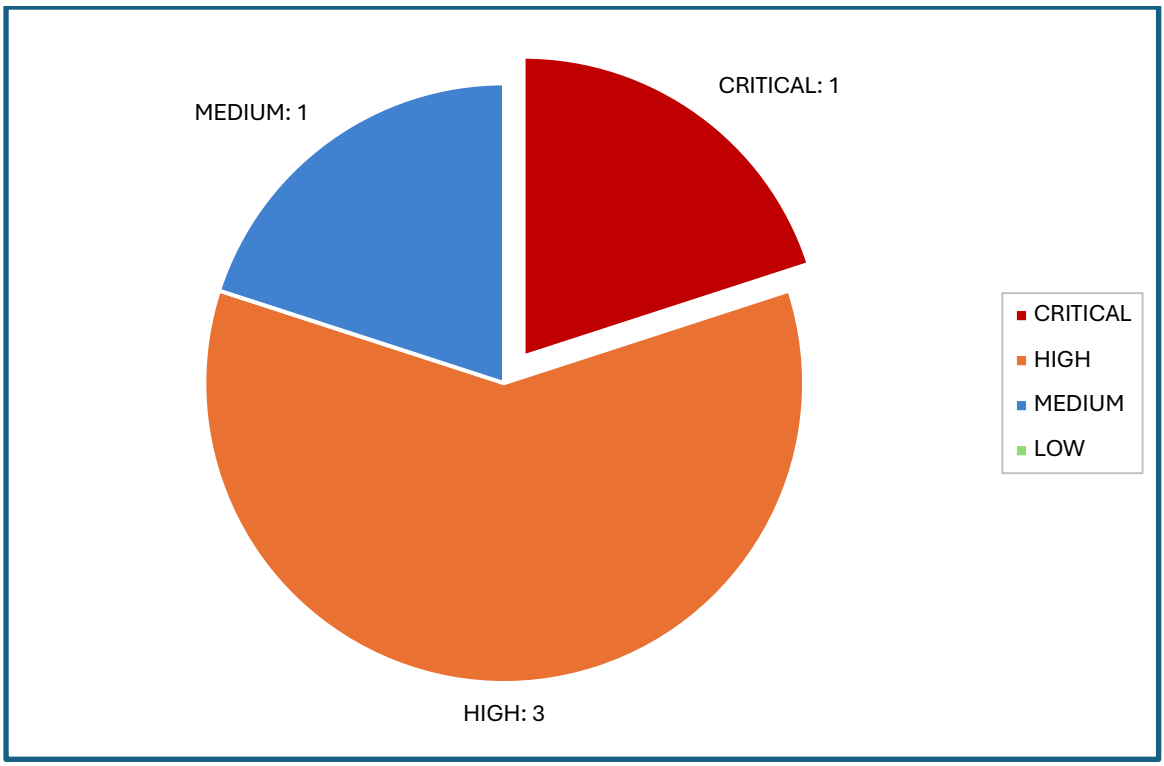
S. N	Meeting Date	Time	Members	Remark
1	7/1/2025	12:00 PM	Cynical Security team and McFoster Inc. development team	We had a discussion regarding the vulnerabilities identified and its impact and the fixes.

VAPT Findings

Report Statistics

During our security testing, we encountered a total of 5 vulnerabilities on the company's IT assets. The company has maintained a good level of access control rules and security best practices. However, several vulnerabilities were discovered during our assessment period. The summary is as follows:

TOTAL FINDINGS	5
SEVERITY: CRITICAL	1
SEVERITY: HIGH	3
SEVERITY: MEDIUM	1
SEVERITY: LOW	0


CRITICAL: 1
MEDIUM: 1
HIGH: 3
CRITICAL
HIGH
MEDIUM
LOW

WEB APPLICATION FINDINGS

Finding-001: Credentials exposed via directory listing (HIGH) [7.5]

Description:	A sensitive file containing plaintext credentials is publicly accessible at http://testphp.vulnweb.com/pictures/credentials.txt. This file is stored in a web-accessible directory without any access control, exposing usernames and passwords to anyone who visits the URL.
Impact:	• Attackers can use these credentials to log in, impersonate users, or escalate privileges. • If credentials belong to admin or reused elsewhere, full system compromise is possible. • Exposed data helps attackers plan further attacks within or beyond the target scope. • Storing sensitive data in accessible locations may violate security best practices and data protection regulations (e.g., GDPR, CPRA)
System:	Web Application
URL:	http://testphp.vulnweb.com/pictures/credentials.txt

HTTP Request:	`GET /pictures/credentials.txt HTTP/1.1`
	`Host: testphp.vulnweb.com`
	`Accept-Language: en-US,en;q=0.9`
	`Upgrade-Insecure-Requests: 1`
	`User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36`
	`Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7`
	`Accept-Encoding: gzip, deflate, br`
	`Cookie: login=test%2Ftest`
	`Connection: keep-alive`

Status:	Resolved

Steps To Reproduce

1. Go to this URL

   `http://testphp.vulnweb.com/pictures/credentials.txt`
2. Observe that credential for user test is exposed.

Evidence

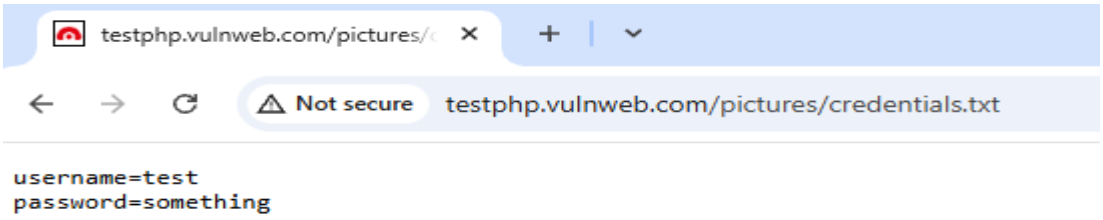


Figure 1.1: Credential Expose in plain text

Resolved Evidence

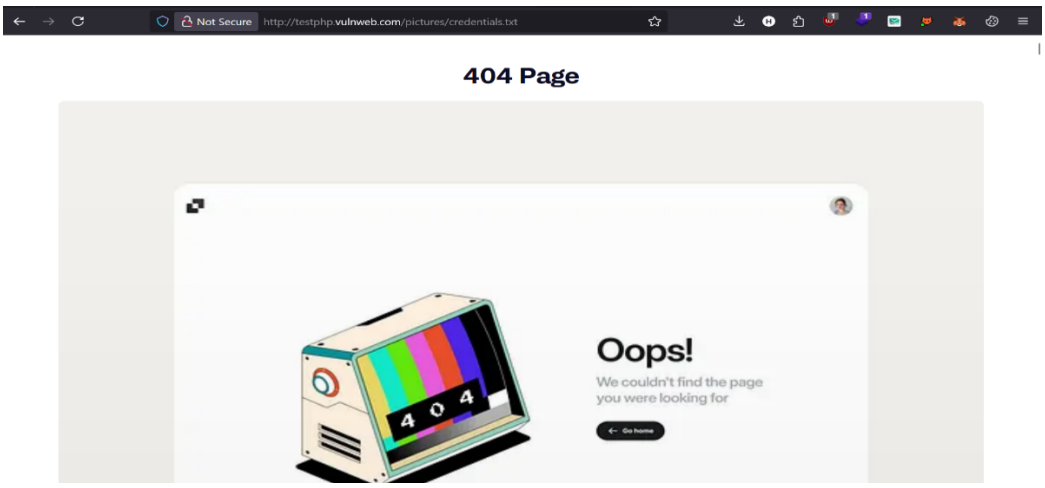


Figure 1.2: Credential Expose in plain text is fixed as we can see that page shows 404 Status.

Remediation

To address this vulnerability, the following measures should be implemented:

- Immediately remove the exposed credentials.txt file from the public directory.
- Rotate all credentials exposed in the file to prevent unauthorized reuse.
- Restrict access to sensitive directories using server configurations (e.g., .htaccess, nginx rules).
- Implement automated scanning for exposed files and secrets using tools like truffleHog, git-secrets, or custom scripts.

Finding-002: SQL Injection in login authentication page (CRITICAL) [9.4]

Description:	The login page at http://testphp.vulnweb.com/login.php is vulnerable to SQL Injection, allowing an attacker to bypass authentication by injecting a malicious payload such as ' OR '1'='1 into the password field. This input manipulates the backend SQL query to always return true, effectively granting access without valid credentials. As a result, attackers can log in as any user, including administrators, leading to unauthorized access, potential privilege escalation, exposure of sensitive user data, and full compromise of the application's access control mechanisms.
Impact:	• Attackers can log in as any user (even admin) without valid credentials by injecting malicious SQL (e.g., ' OR '1'='1), bypassing authentication checks entirely. • If admin accounts are targeted, attackers gain full control over the application, potentially modifying or deleting data, changing settings, or adding backdoors. • Once authenticated, attackers can access sensitive user data and internal functionality intended only for authorized users. • Bypassing auth via SQLi can lead to violations of data protection laws (like GDPR), triggering fines, lawsuits, and reputational damage.
System:	Web Application
URL:	http://testphp.vulnweb.com/login.php

HTTP Request:	`POST /userinfo.php HTTP/1.1`
	`Host: testphp.vulnweb.com`
	`Content-Length: 27`
	`Cache-Control: max-age=0`
	`Accept-Language: en-US,en;q=0.9`
	`Origin: http://testphp.vulnweb.com`

	Content-Type: application/x-www-form-urlencoded Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7 Referer: http://testphp.vulnweb.com/login.php Accept-Encoding: gzip, deflate, br Connection: keep-alive uname=test&pass=' OR '1'='1
Status:	Unresolved

Steps To Reproduce

1. Open the Login Page:
   - URL: http://testphp.vulnweb.com/login.php
2. Prepare the Malicious Payload:
   - Username (uname): test *(can be anything or even blank)*
   - Password (pass): ' OR '1'='1
   - This bypasses password verification due to SQL logic flaw.
3. Send a Malicious POST Request:
4. Use Burp Suite or Repeater:
   - Intercept the request and modify the POST body like:

```
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
Content-Type: application/x-www-form-urlencoded

uname=test&pass=' OR '1'='1
```

   - Forward the request.
5. Observe the Result:
   - You'll be redirected to userinfo.php and shown a logged-in user profile page — bypassing the password check.

Evidence

Request

Pretty Raw Hex

```
1  POST /userinfo.php HTTP/1.1
2  Host: testphp.vulnweb.com
3  Content-Length: 27
4  Cache-Control: max-age=0
5  Accept-Language: en-US,en;q=0.9
6  Origin: http://testphp.vulnweb.com
7  Content-Type: application/x-www-form-urlencoded
8  Upgrade-Insecure-Requests: 1
9  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
10 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
   ;q=0.7
11 Referer: http://testphp.vulnweb.com/login.php
12 Accept-Encoding: gzip, deflate, br
13 Connection: keep-alive
14
15 uname=test&pass=' OR '1'='1
```

Figure 2.1: SQL payload to bypass authentication

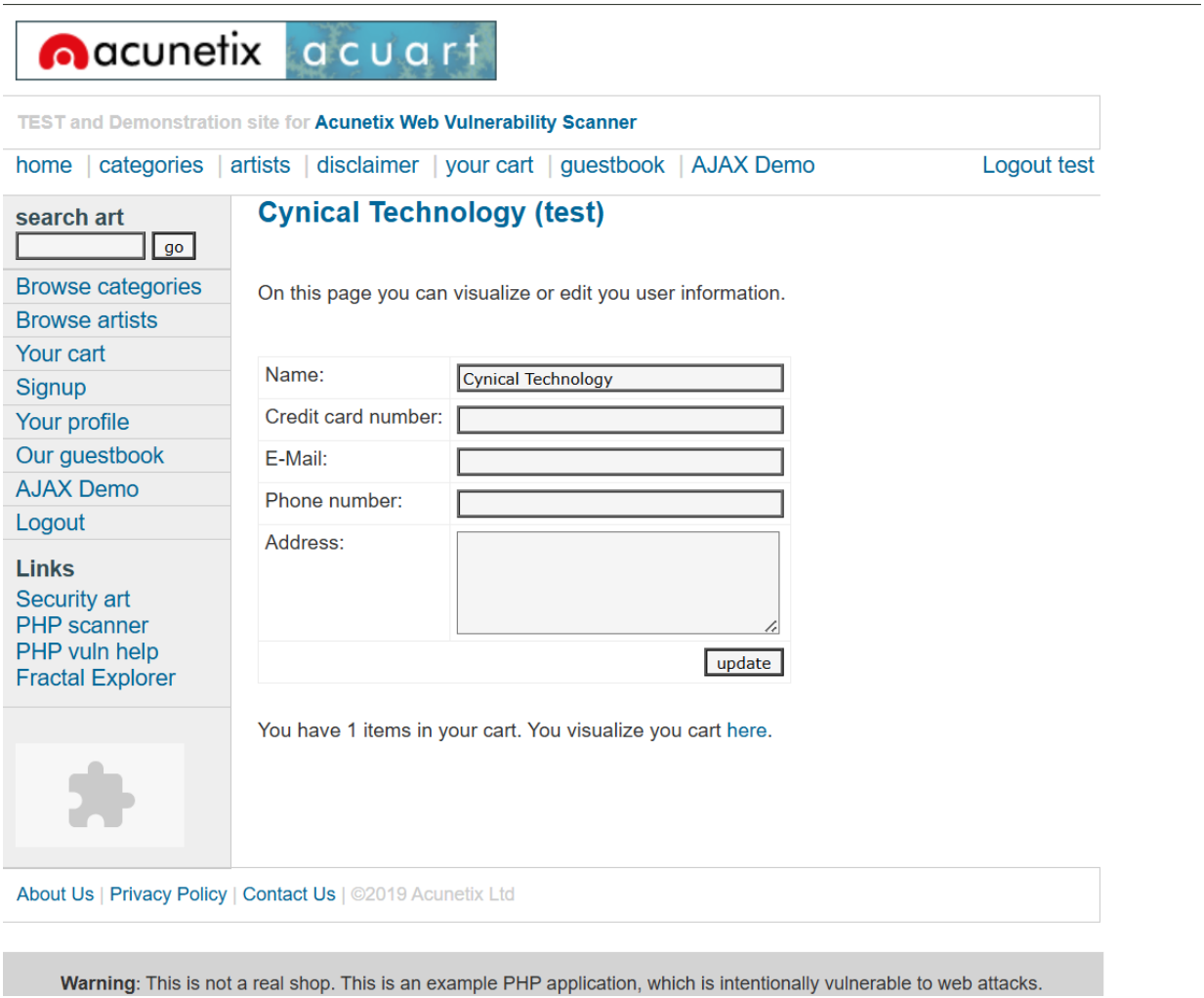


Figure 2.2: Successfully logged in to test user

Remediation

To address this vulnerability, the following measures should be implemented:

- Use Prepared Statements: Ensure all SQL queries use safe, parameterized inputs (e.g., using mysqli or PDO in PHP) to separate data from code.
- Input Validation & Escaping: Strictly validate and sanitize user input (e.g., allow only expected characters in usernames and passwords).
- Use ORM or Secure Frameworks: Implement an Object-Relational Mapping (ORM) tool that handles query binding securely by default.
- Implement Web Application Firewall (WAF): As an additional layer, use a WAF to detect and block common SQL injection attempts.

Finding-003: Directory traversal on /showimage.php?file= endpoint (HIGH) [7.4]

Description:	The showimage.php endpoint is designed to allow users to view images by specifying the image path via the file parameter (e.g., file=./pictures/1.jpg). However, this input is not properly validated or sanitized, making the endpoint vulnerable to Local File Inclusion (LFI) attacks. An attacker can exploit this by supplying crafted input such as ../../etc/passwd to traverse the server's directory structure and include arbitrary files outside the intended image directory. This misconfiguration allows unauthorized access to sensitive system files and can potentially be leveraged for further attacks like remote code execution if log or session injection is possible.
Impact:	• Read sensitive files like /etc/passwd, .htaccess, configuration files (config.php, wp-config.php, etc.). • Contents of /etc/passwd can reveal system usernames. • If LFI allows inclusion of web server logs or session files, it can lead to Remote Code Execution (RCE) via log poisoning or session hijacking.
System:	Web Application
URL:	http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg

HTTP Request:	`GET /showimage.php?file=../../etc/passwd HTTP/1.1`
	`Host: testphp.vulnweb.com`
	`Accept-Language: en-US,en;q=0.9`
	`Upgrade-Insecure-Requests: 1`
	`User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36`
	`Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7`

	Accept-Encoding: gzip, deflate, br Cookie: login=test%2Ftest Connection: keep-alive
Status:	Resolved

Steps To Reproduce

1. Access the Legitimate Image Viewer:
   - Open your browser and visit:

     `http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg`

   - You should see the image load normally.
2. Test for Directory Traversal:
   - Modify the file parameter in the URL to point outside the intended directory, for example:

     `http://testphp.vulnweb.com/showimage.php?file=../../etc/passwd`

   - Press Enter to load the URL.
3. Observe the Response:
   - If vulnerable, the server will display the contents of the /etc/passwd file (or the equivalent on Windows, e.g., C:\Windows\win.ini).
   - This confirms directory traversal and arbitrary file read through the image viewer.

Evidence

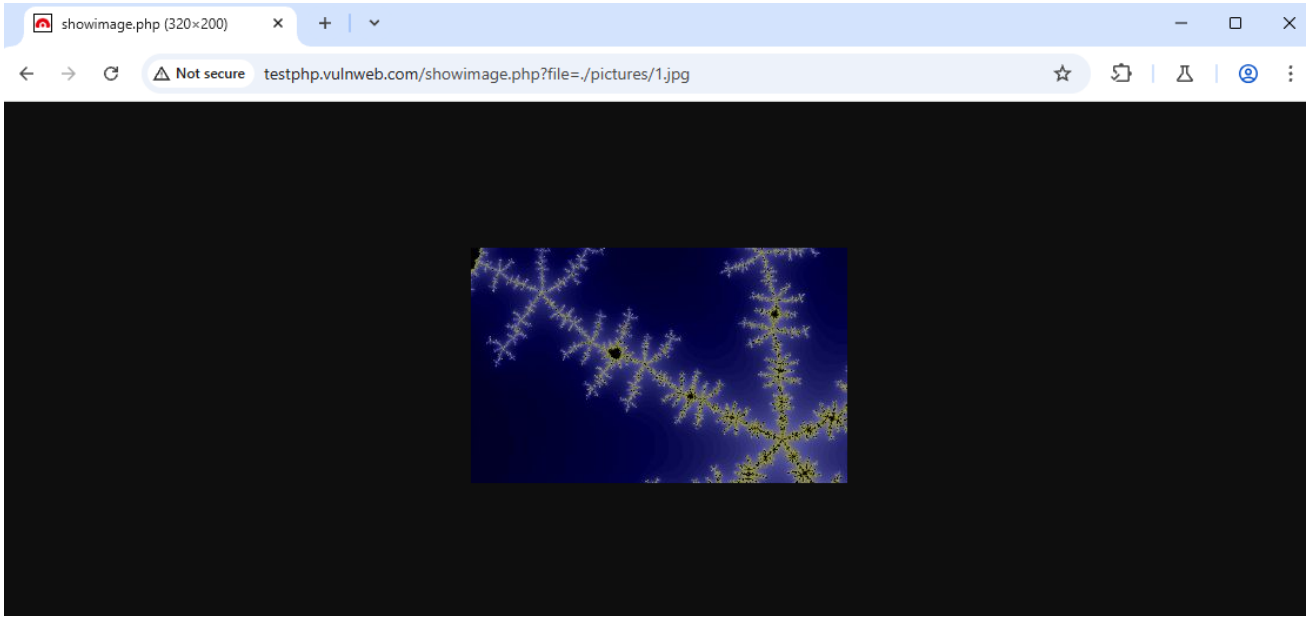


Figure 3.1: Fractal Image received from server

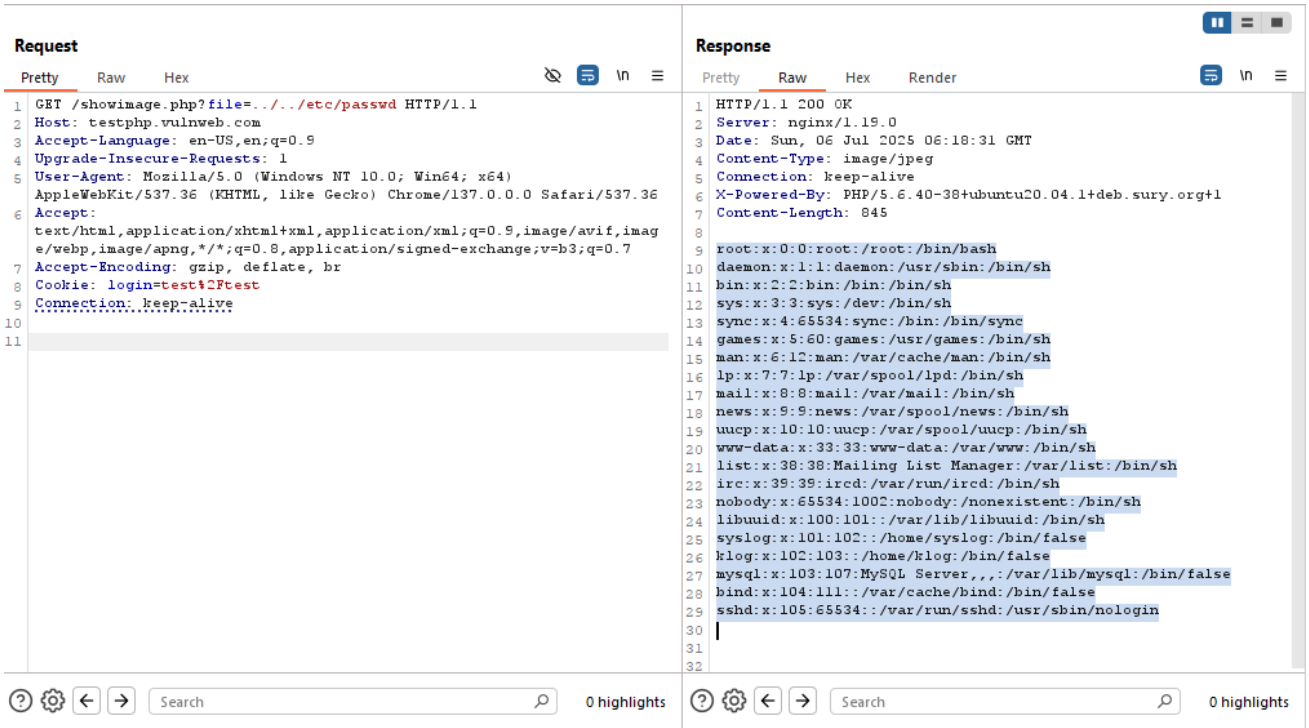


Figure 3.2: Exploiting Local File Inclusion to view sensitive files

Resolved Evidence

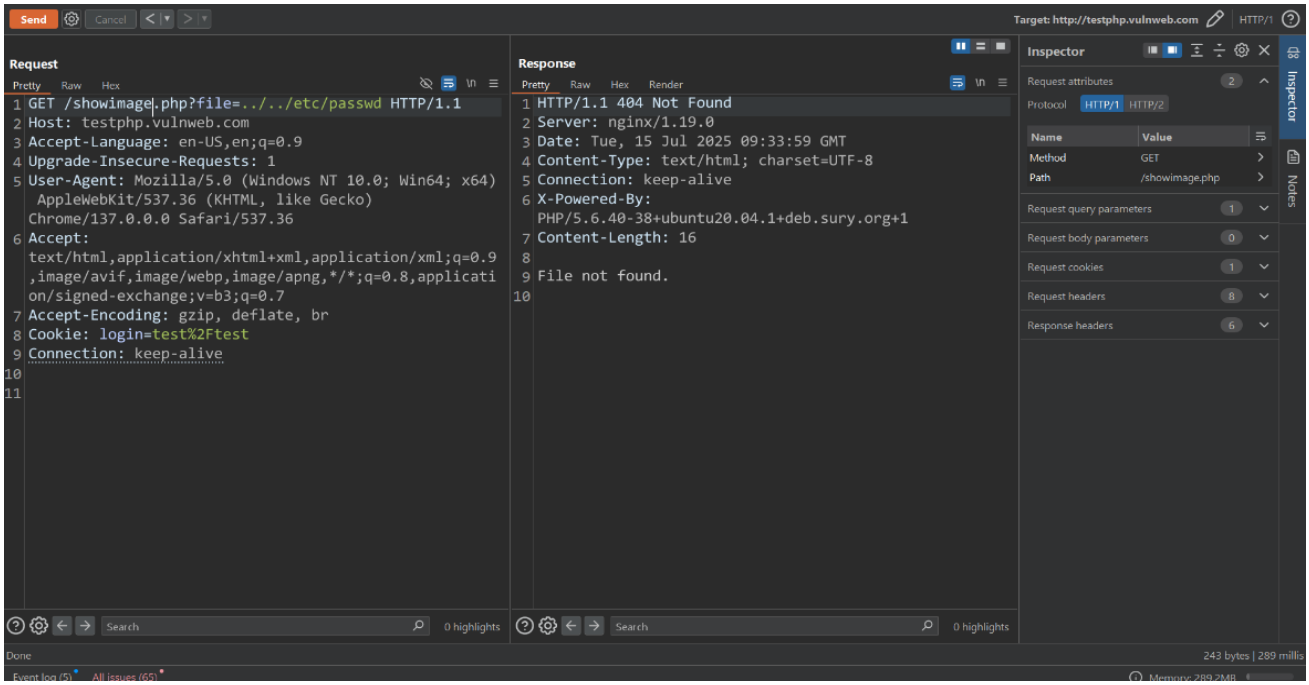


Figure 3.3: Local File Inclusion to view sensitive files is resolved.

Remediation

To address this vulnerability, the following measures should be implemented:

- Validate and sanitize user input: Whitelist only known-good filenames (e.g., image files from a specific folder).
- Use static file paths: Do not pass user input directly to include(), require(), or file_get_contents().
- Disable dangerous PHP functions: Restrict allow_url_include, allow_url_fopen, and use hardened server configurations.
- Run application with least privileges: Ensure the web server cannot access sensitive OS-level files.

Finding-004: Reflected Cross-Site Scripting (XSS) in listproducts.php via cat parameter (MEDIUM) [6.3]

Description:	The listproducts.php page takes a user-supplied cat parameter to filter products by category. However, this input is not properly sanitized or encoded before being reflected in the HTML response. This allows an attacker to inject malicious JavaScript code that executes in the victim's browser immediately after clicking a crafted link or submitting a manipulated request. This is a classic Reflected Cross-Site Scripting (XSS) vulnerability affecting the web application's input handling.
Impact:	• Attackers can steal session cookies, enabling account takeover. • Malicious scripts can capture login credentials or other sensitive information. • Combined with browser vulnerabilities, can execute arbitrary code on the client.
System:	Web Application
URL:	http://testphp.vulnweb.com/listproducts.php?cat=%22%3E%3Cscript%3Ealert(%22XSS%22)%3C/script%3E
HTTP Request:	GET /listproducts.php?cat=%22%3E%3Cscript%3Ealert(%22XSS%22)%3C/script%3E HTTP/1.1 Host: testphp.vulnweb.com Accept-Language: en-US,en;q=0.9 Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7 Accept-Encoding: gzip, deflate, br Connection: keep-alive

Status:	Unresolved

Steps To Reproduce

1. Access the vulnerable URL with a normal parameter:

   `http://testphp.vulnweb.com/listproducts.php?cat=2`

   This loads the product list for category 2 without any issues.

2. Inject the XSS payload into the cat parameter by modifying the URL:

   `http://testphp.vulnweb.com/listproducts.php?cat=2"><script>alert("XSS")</script>`

   - URL-encoded form (to paste directly in browser):

     `http://testphp.vulnweb.com/listproducts.php?cat=2%22%3E%3Cscript%3Ealert(%22XSS%22)%3C/script%3E`

3. Observe the Behavior:
   - When you open the URL, an alert box with the message "XSS" will pop up.
   - This confirms that the input is reflected unsanitized in the page HTML and executed as JavaScript.

Evidence

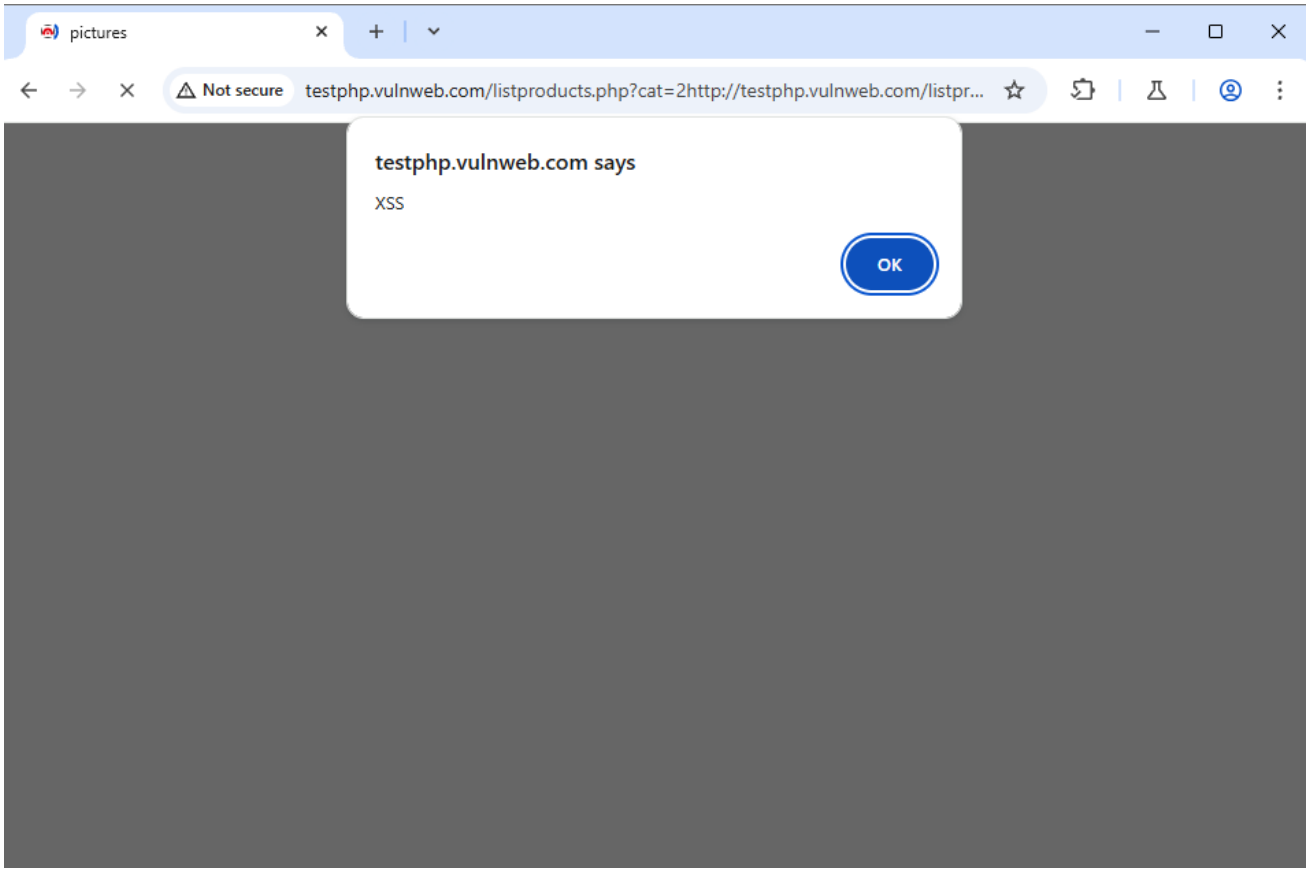


Figure 4.1: Reflected Cross Site Scripting from `cat` parameter

Remediation

To address this vulnerability, the following measures should be implemented:

- Input Validation and Encoding: Always sanitize and properly HTML-encode user input before reflecting it back in the page.
- Use Security Libraries/Frameworks: Implement context-aware escaping (e.g., OWASP ESAPI or built-in framework templating engines).
- Content Security Policy (CSP): Add CSP headers to restrict script execution to trusted sources.
- HTTPOnly & Secure Cookies: Reduce cookie theft risks by setting these flags.

Finding-005: Exposed Database Initialization File (HIGH) [7.0]

Description:	The create.sql file is publicly accessible and contains SQL statements used to initialize the backend database schema for the application. It exposes the structure of several backend tables, including forum, artists, categ, and pictures, along with sensitive implementation details such as table names, field types, and expected data — all of which can significantly aid an attacker during reconnaissance. This file can be accessed at: http://testphp.vulnweb.com/admin/create.sql.
Impact:	• Attackers gain direct insight into the database schema, which aids in crafting SQL Injection payloads more efficiently. • Revealing internal table structures may expose business logic or proprietary design. • If other files (like dumps or backups) exist similarly, credentials might also be exposed.
System:	Web Application
URL:	http://testphp.vulnweb.com/admin/create.sql

HTTP Request:	`GET /admin/create.sql HTTP/1.1`
	`Host: testphp.vulnweb.com`
	`Accept-Language: en-US,en;q=0.9`
	`Upgrade-Insecure-Requests: 1`
	`User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36`
	`Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7`
	`Referer: http://testphp.vulnweb.com/admin/`
	`Accept-Encoding: gzip, deflate, br`
	`Connection: keep-alive`

Stratus:	Resolved

Steps To Reproduce

1. Open a Web Browser.
2. Navigate to the Admin Directory:
   - Go to:

     `http://testphp.vulnweb.com/admin/`

   - You will see a directory listing if directory browsing is enabled.
3. Locate and Click on create.sql:

Evidence

Figure 5.1: Expose database initialization via directory listing

Resolved Evidence

Figure 5.2: Expose database initialization via directory listing is removed.

Remediation

To address this vulnerability, the following measures should be implemented:

- Remove Public Access: Do not store .sql, .bak, .zip, or other sensitive backup/configuration files in web-accessible directories.
- Use .htaccess or Server Rules: Block access to sensitive file types (e.g., via Apache or NGINX rules).
- File Integrity Monitoring: Set up tools to detect when sensitive files appear in public directories.
- Secure DevOps Practices: Ensure deployment pipelines do not push staging/dev artifacts (like create.sql) to production environments.

Conclusion

In conclusion, the comprehensive VAPT conducted on McFoster Inc.'s Web application, Network Infrastructure, and mobile application has not only identified key strengths in security practices but also highlighted areas requiring attention to mitigate potential vulnerabilities. McFoster Inc.'s dedication to utilizing up-to-date components devoid of known vulnerabilities exemplifies its commitment to cybersecurity and the protection of its digital ecosystem.

Throughout the assessment, McFoster Inc.'s efforts in collaborating with our team in mitigating vulnerabilities for robust access control measures have been commendable. These measures have significantly contributed to the secure and resilient operation of their services, ensuring the confidentiality, integrity, and availability of user data.

McFoster Inc.'s proactive response to the findings of this VAPT, aligned with industry best practices and regulatory standards, underscores their dedication to maintaining a secure and trustworthy digital environment. Their commitment to continuous improvement and adherence to stringent security measures positions McFoster Inc. as a responsible entity in the digital finance sector, poised to effectively manage and mitigate cybersecurity risks.

We acknowledge McFoster Inc.'s collaborative and responsive approach throughout this assessment and encourage their ongoing engagement with cybersecurity initiatives. This VAPT offers valuable insights into McFoster Inc.'s current security posture, laying a solid foundation for future enhancements and the sustained protection of its digital assets.

Cynical Technology Pvt. Ltd.

Kathmandu, Nepal
+977-01-4530730
info@cynicaltechnology.com