Vulnerability Assessment Summary Report

PREPARED BY

muLearn circle : Cyber ettayis
code:CYCYBVAS123

Content

Summary:

This Vulnerability Assessment report presented by muLean Circle Cyber ettayis contains reports of various vulnerabilities findings present in the website http://testphp.vulnweb.com along with the respective Issue severity, Issue Details, Issue Background and POF.

The various assessments were conducted using manual and automated methods and tools.

Also it should be noted that ,the issues presented over here only covers a small number of issues that could be completely verified manually along with POC within the given timelimit.

With more time we are confident to present more issues with more proof.
Several minor vulnerabilities are also mentioned.

1.Blind SQL

Severity : Medium
Confidence : Certain

Issue Detail :

SQL injection is a technique employed to compromise data-driven applications. It involves the insertion of malicious SQL statements into entry fields, aiming to execute actions like extracting the database contents for unauthorised access.

Issue Background :

Successful SQL injection relies on exploiting security vulnerabilities in an application's software. This occurs when user input lacks proper filtering for string literal escape characters within SQL statements or when input isn't strongly typed, leading to unexpected execution.

Issue Remediation:

Input Validation and Parameterized Queries:

- Implement strict input validation to prevent unauthorised SQL statements.
- Use parameterized queries to ensure a clear separation between data and SQL commands, minimising the risk of injection.

Web Application Firewall (WAF):

- Deploy a Web Application Firewall to monitor and filter HTTP traffic, effectively detecting and blocking SQL injection attempts.

Regular Security Audits:

- Conduct regular security audits and penetration testing to proactively identify and address SQL injection vulnerabilities, ensuring ongoing resilience against evolving threats.

POC:

Request

```
1  GET /product.php HTTP/1.1
2  Host: testphp.vulnweb.com
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
   Firefox/115.0
4  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
   q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  DNT: 1
8  Connection: close
9  Upgrade-Insecure-Requests: 1
10 Pragma: no-cache
11 Cache-Control: no-cache
12
```

Response:

2.Cross Site Scripting

Severity : **Medium**
Confidence : **Firm**

Issue Detail:

The name of an arbitrarily supplied URL parameter is copied into the HTML document as plain text between
tags. The payload <script>alert(Vulnerability_Exposed)</script> was submitted in the name of an arbitrarily supplied URL
parameter. This input was echoed unmodified in the application's response.
This behaviour demonstrates that it is possible to inject new HTML tags into the returned document.

Issue Background:

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user’s browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page

Issue Remediation:

In most situations where user-controllable data is copied into application responses, cross-site scripting attacks can be prevented using two layers of defences:

● Input should be validated as strictly as possible on arrival, given the kind of content that it is expected to contain. For example, personal names should consist of alphabetical and a small range of typographical characters, and be relatively short; a year of birth should consist of exactly four numerals; email addresses should match a well-defined regular expression. Input which fails the validation should be rejected, not sanitized.

● User input should be HTML-encoded at any point where it is copied into application responses. All HTML metacharacters, including < > " ' and =, should be replaced with the corresponding HTML entities (< > etc).

Vulnerability classifications

● CWE-79: Improper Neutralisation of Input During Web Page Generation ('Cross-site Scripting')

● CWE-80: Improper Neutralisation of Script-Related HTML Tags in a Web Page (Basic XSS)
● CWE-116: Improper Encoding or Escaping of Output

● CWE-159: Failure to Sanitize Special Elemen

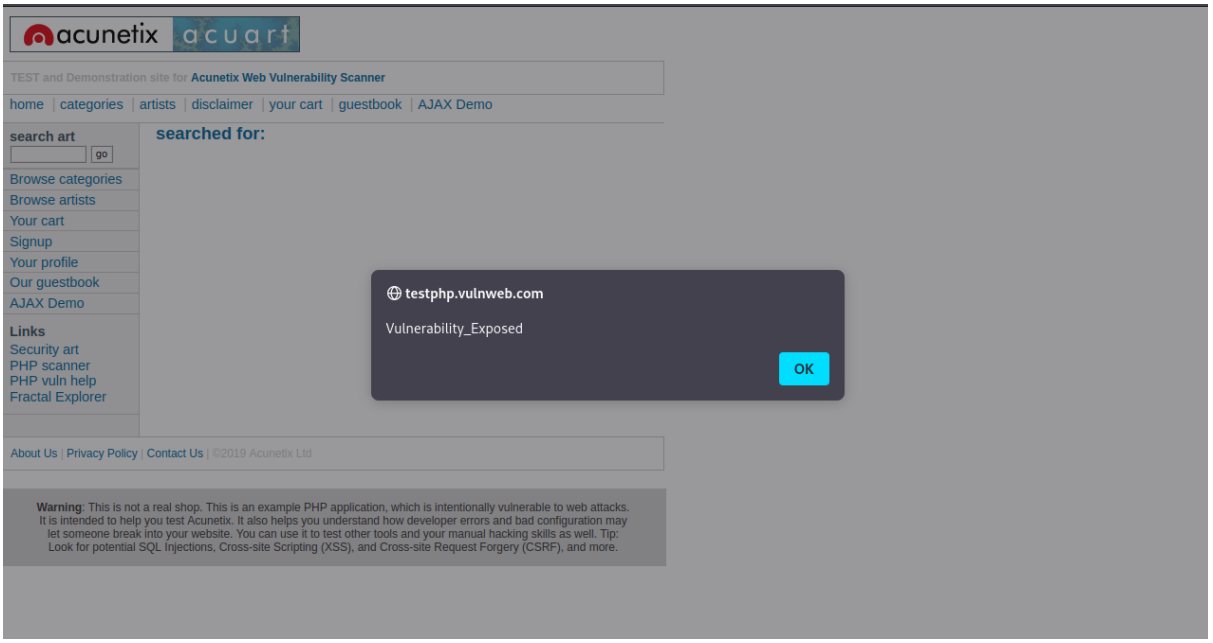
POC:

Request:

```
POST /search.php?test=query HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 22
Origin: http://testphp.vulnweb.com
Connection: close
Referer: http://testphp.vulnweb.com/search.php?test=query
Upgrade-Insecure-Requests: 1

searchFor=<script>alert("Vulnerability_Exposed")</script>=&goButton=go
```

Response:



3.Outdated PHP version

Severity : Critical
Confidence : Low

Issue Detail:
The system is currently running an older version of PHP (5.1.6) that's vulnerable to security threats. As a result, there is a risk that malicious actors may be able to exploit known vulnerabilities in the PHP version to access sensitive data, compromise the application's stability, or even gain unauthorised access to the system.

Issue Background:
Outdated PHP websites are like houses with weak locks and doors. Hackers can find these weak spots and break in. For example:

Cross-Site Scripting (XSS): Attackers can sneak malicious code into web pages, stealing users' data or hijacking their sessions.

SQL Injection: Hackers can manipulate input fields to access a website's database, potentially stealing or altering sensitive information.

Remote Code Execution (RCE): Attackers can run their code on the server, taking control of the website or accessing other parts of the server.

File Inclusion: Insecure code allows attackers to include harmful files, leading to unauthorised access and data breaches.

Insecure Session Management: Weak session handling lets attackers hijack user sessions, gaining unauthorised access.

Directory Traversal: Hackers can navigate outside the web folder, accessing sensitive files.

To protect against these, website owners must update PHP, use secure coding practices, and regularly check for vulnerabilities. This is like fixing the locks, securing windows, and keeping the house in good shape to prevent break-ins.

Issue Remediation:

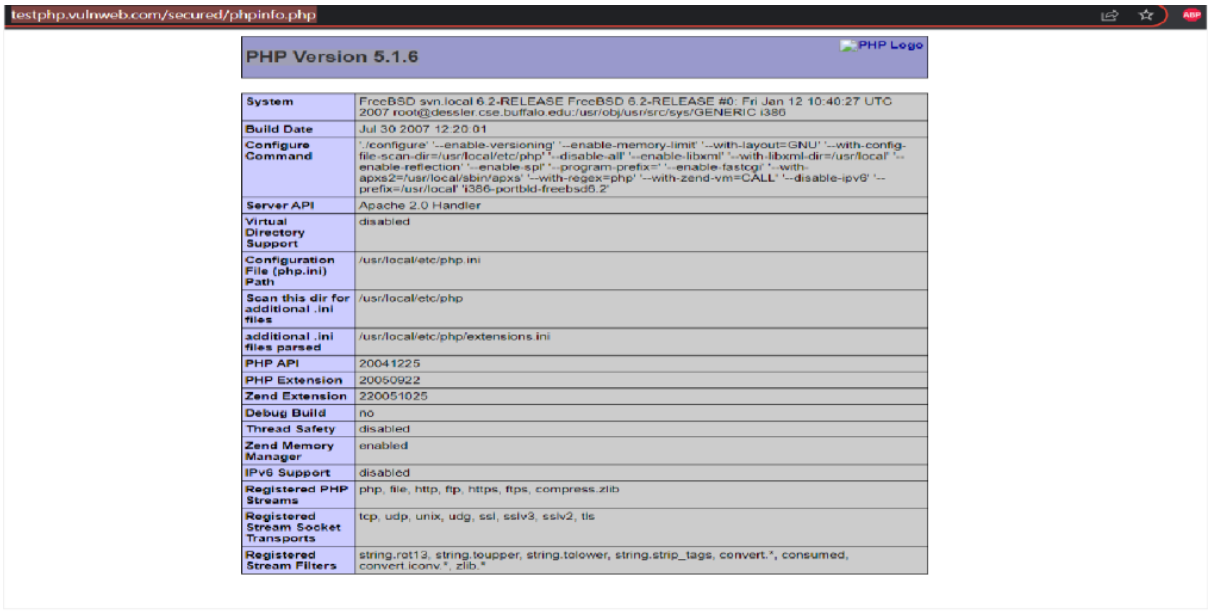
1-Upgrade to the latest version: The most effective way to address vulnerabilities caused by outdated PHP versions is to upgrade to the latest version of PHP.

2-Apply security patches: Applying security patches that have not been released for your version of PHP.

3-Use a web application firewall (WAF): A WAF can provide an additional layer of security by monitoring incoming traffic and blocking any requests that are deemed suspicious or potentially malicious.

POC:

phpinfo.pfp found in /secured by dribuster..



testphp.vulnweb.com/secured/phpinfo.php

PHP Version 5.1.6

System	FreeBSD svn.local 6.2-RELEASE FreeBSD 6.2-RELEASE #0: Fri Jan 12 10:40:27 UTC 2007 root@dessler.cse.buffalo.edu:/usr/obj/usr/src/sys/GENERIC i386
Build Date	Jul 30 2007 12:20:01
Configure Command	'./configure' '--enable-versioning' '--enable-memory-limit' '--with-layout=GNU' '--with-config-file-scan-dir=/usr/local/etc/php' '--disable-all' '--enable-libxml' '--with-libxml-dir=/usr/local' '--enable-reflection' '--enable-spl' '--program-prefix=' '--enable-fastcgi' '--with-apxs2=/usr/local/sbin/apxs' '--with-regex=php' '--with-zend-vm=CALL' '--disable-ipv6' '--prefix=/usr/local' 'i386-portbld-freebsd6.2'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php.ini
Scan this dir for additional .ini files	/usr/local/etc/php
additional .ini files parsed	/usr/local/etc/php/extensions.ini
PHP API	20041225
PHP Extension	20050922
Zend Extension	220051025
Debug Build	no
Thread Safety	disabled
Zend Memory Manager	enabled
IPv6 Support	disabled
Registered PHP Streams	php, file, http, ftp, https, ftps, compress.zlib
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls
Registered Stream Filters	string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, convert.iconv.*, zlib.*

Multiple vulnerabilities discovered in the version.

Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion	# of exploits
2006	3		1	1						2					
2007	33	5	11	10						5	1				
2008	12		2	2				2		3	1				
2009	9	1					2				1				
2010	2			1		1	1			2					
2011	20	14	1	7	2					3	1				5
2012	9	4	2	1		1		1		1					
2013	6	2		2						1	1				
2014	1									1	1				
2015	1		1	1							1				
2017	1	1													
Total	97	27	18	25	2	2	3	3		18	7				5
% Of All		27.8	18.6	25.8	2.1	2.1	3.1	3.1	0.0	18.6	7.2	0.0	0.0	0.0	

4.Sensitive Information Disclosure

Severity : Medium

Confidence : Certain

Issue Detail:

The system is vulnerable to sensitive files disclosure, which could allow an unauthorized user to gain access to confidential information. This poses a significant risk to the privacy and security of the system's data, as well as the individuals whose information may be exposed.

Issue Background:

Sensitive information typically underscores the critical importance of protecting confidential data from unauthorized access and disclosure. It highlights that the compromise of sensitive information can have severe consequences, including financial loss, damage to reputation, legal ramifications, and breaches of privacy. This section may provide context on the types of sensitive data at risk, such as personally identifiable information (PII), financial records, or intellectual property. It emphasises the necessity of robust security measures, such as encryption, access controls, and data classification, to maintain a high level of confidence in safeguarding sensitive information.

Issue Remediation:

1-Conduct a risk assessment: Conduct a thorough risk assessment to identify all sensitive files and data -in the system, including where they are stored, who has access to them, and the potential risks associated with them.

2-Implement access controls: Ensure that sensitive files are only accessible to authorized personnel who require access to perform their job duties. Use role-based access controls to limit access to sensitive files and data.

3-Use encryption: Implement encryption to protect sensitive files both at rest and in transit. This can include encryption of file storage, database encryption, and encrypted file transfers.

Vulnerability details:

Affects:

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

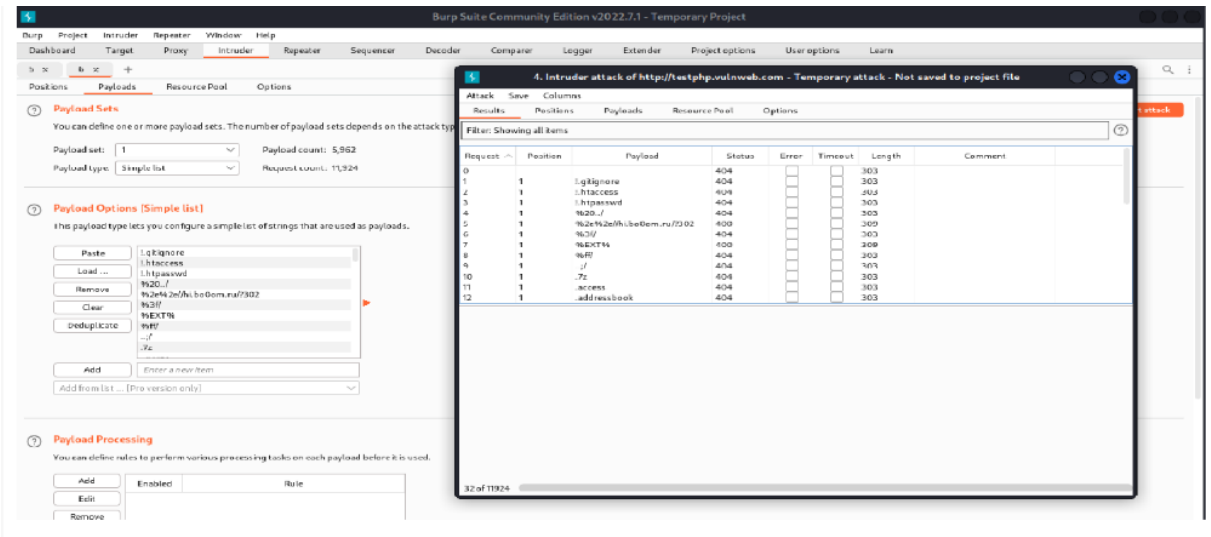
http://testphp.vulnweb.com/CVS/Root

http://testphp.vulnweb.com/secured/phpinfo.php

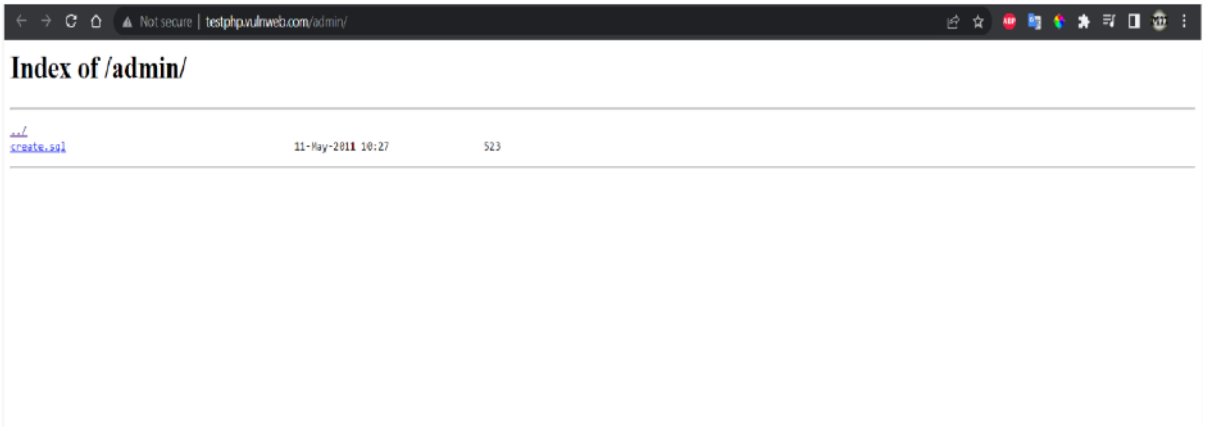
POC:

Request

We used burp suite find critical files disclosure



Response



Conclusion:

In conducting this comprehensive vulnerability assessment, we have systematically identified and evaluated potential security risks within the system. The analysis encompassed various aspects, including network vulnerabilities, application weaknesses, and potential exposure of sensitive information.

Our findings underscore the importance of a proactive approach to cybersecurity. Addressing the identified vulnerabilities is crucial to fortify the system against potential exploits, unauthorized access, and data breaches. The recommended remediation measures, outlined in the assessment, provide a strategic roadmap for enhancing the overall security posture.

It is imperative for stakeholders to prioritize the implementation of these remedial actions promptly. Additionally, ongoing vigilance and regular assessments are vital components of a robust security strategy to adapt to evolving threats and vulnerabilities.

By addressing the identified issues and fostering a culture of continuous improvement in cybersecurity practices, the organization can significantly reduce the risk landscape and bolster its resilience against potential security threats.