

# 1 Executive Summary

The penetration testing of the testphp.vulnweb.com website identified several critical and high-severity vulnerabilities that could pose a significant risk to the security of the site and its users.

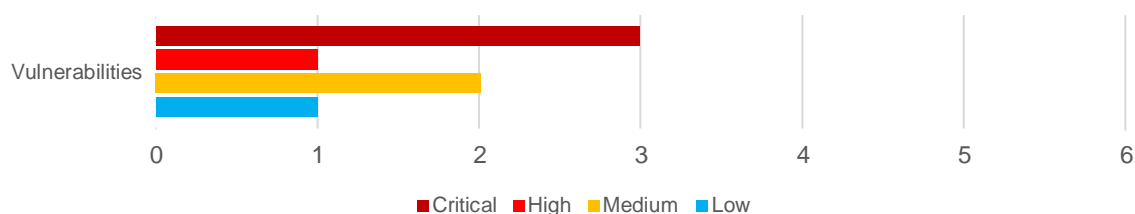
## 1.1 Assessment Summary

The most severe vulnerabilities discovered were SQL Injection and File Inclusion vulnerabilities, which could allow an attacker to gain unauthorized access to sensitive data or execute arbitrary code on the server. Additionally, the website's outdated PHP version and weak password policy were also classified as critical vulnerabilities that need to be addressed urgently.

The following table represents the penetration testing in-scope items and breaks down the issues, which were identified and classified by severity of risk. (Note that this summary table does not include the informational items):

Phase	Description	Critical	High	Medium	Low	Total
1	Web Penetration Testing	3	1	2	1	7
	Total	3	1	2	1	7

The graphs below represent a summary of the total number of vulnerabilities found up until issuing this current report:



## Strategic Recommendations

Overall, the testing process was effective in identifying the security weaknesses in the website, and the results should be used to prioritize and address the **Critical** and **High** vulnerabilities to protect the site and its users from potential attacks

## 1.2 Findings Overview

All the issues identified during the assessment are listed below with a brief description and risk rating for each issue.

Ref	Description	Risk
SQLI-1-1	SQL Injection vulnerabilities	CRITICAL
PHP-1-2	Outdated PHP version	CRITICAL
WP-1-3	Weak Password	CRITICAL
XSS-1-4	Cross-Site Scripting (XSS) vulnerabilities	HIGH
SFD-1-5	Sensitive files disclosure	MEDIUM
DID-1-6	Directory Index disclosure	MEDIUM
ID-1-8	Personally Identifiable Information Disclosure	LOW

## 2 Technical Details

### 2.1 SQL Injection vulnerabilities

CRITICAL

Ref ID: SQLI-1-1

It has been discovered that the system is vulnerable to SQL injection attacks, which could allow a malicious user to retrieve sensitive information such as usernames, passwords, and other confidential data stored in the database. This poses a significant risk to the confidentiality and integrity of the system's data, as well as the privacy of the users whose information may be exposed.

#### Vulnerability Details:

<b>Affects:</b>	https:// http://testphp.vulnweb.com/listproducts.php?cat=1
<b>Parameter(s)</b>	union select 1,2,3,4,5,6,group_concat(uname," ",pass," ",cc," ",address,"___",email),8,9,10,11 from users

#### Finding vulnerable URLs

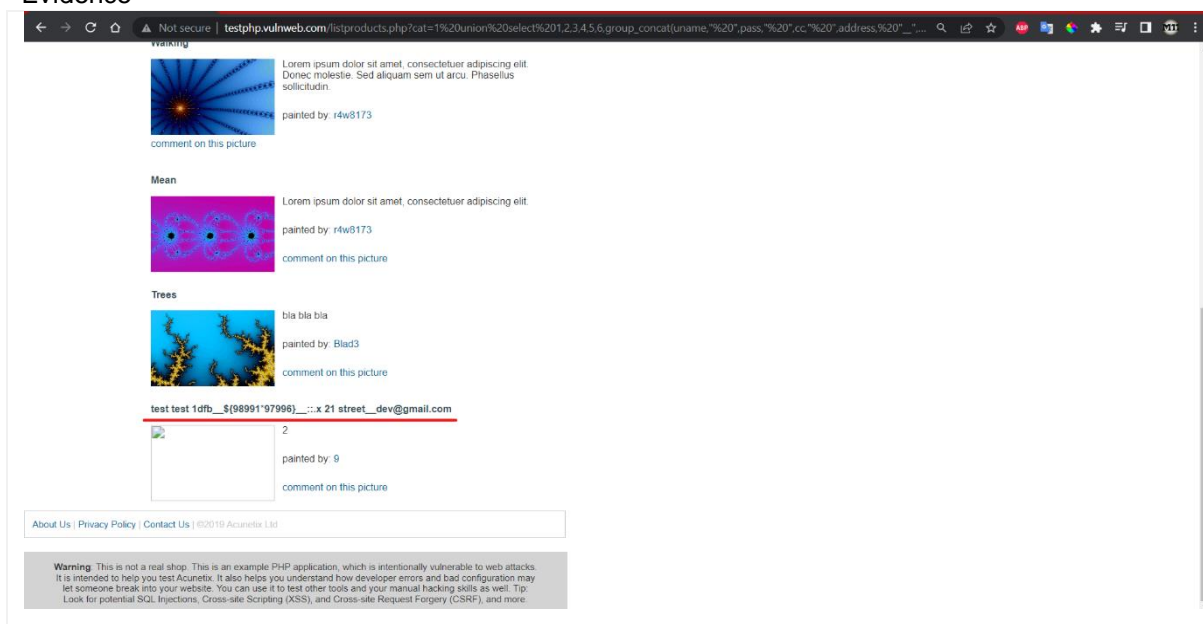
```
kali@Fourti: ~ x kali@Fourti: ~ x
{1.6.7#stable}
https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end
user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are
not responsible for any misuse or damage caused by this program

[*] starting @ 08:19:40 /2023-04-23/

[08:19:40] [INFO] fetched random HTTP User-Agent header value 'Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_6; sv
-se) AppleWebKit/533.20.25 (KHTML, like Gecko) Version/5.0.4 Safari/533.20.27' from file '/usr/share/sqlmap/data/txt
t/user-agents.txt'
do you want to check for the existence of site's sitemap(.xml) [y/N] N
[08:19:40] [INFO] starting crawler for target URL 'http://testphp.vulnweb.com/'
[08:19:40] [INFO] searching for links with depth 1
[08:19:41] [INFO] searching for links with depth 2
[08:19:41] [INFO] starting 10 threads
[08:19:42] [INFO] 3/13 links visited (23%)
got a 302 redirect to 'http://testphp.vulnweb.com/login.php'. Do you want to follow? [Y/n] Y
[08:19:43] [INFO] searching for links with depth 3
[08:19:43] [INFO] starting 10 threads
```

#### Evidence



A successful attack would consist in the following:

1. Identifying the vulnerable URL.
2. Injecting the SQL parameter's "union select 1,2,3,4,5,6,group\_concat(uname," ",pass," ",cc," ",address,"\_\_\_",email),8,9,10,11 from users" in our case.

**Remediation Guidance:**

1-Input Validation: Validate user input on the server-side. ensuring that any input received from the user, such as form data or query parameters, is properly formatted and validated before being used in a SQL query.

2-Parameterized Queries: Use parameterized queries to create SQL queries Instead of concatenating user input directly into the SQL query.

3-Least Privilege: Database users should only have the minimum permissions necessary to perform their specific tasks.

## 2.2 Outdated PHP version

**CRITICAL**

Ref ID: PHP-1-2

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 unauthorized access to the system.

### Vulnerability Details:

<b>Affects:</b>	http://testphp.vulnweb.com/
<b>Attack Vectors</b>	The possibility to gain access to multiple functionality in the website.
<b>References:</b>	<a href="https://www.cvedetails.com/version/399025/PHP-PHP-5.1.6.html">https://www.cvedetails.com/version/399025/PHP-PHP-5.1.6.html</a>

### Evidence

phpinfo.php in one of the URL (/secured) found by dribuster  
(<http://testphp.vulnweb.com/secured/phpinfo.php>).

testphp.vulnweb.com/secured/phpinfo.php		PHP Logo
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-regexp=php' '--with-zend-vm=CALL' '--disable-ipv6' '--prefix=/usr/local' 'i386-portbid-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 Vulnerability discovered in this 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	2	2					3	1				5
2012	9	4	2	1		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	

A successful attack would consist in the following:

- Utilizing the information provided by cvedetails to do the different types of attacks.

**Remediation Guidance:**

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.

## 2.3 Weak Password

**CRITICAL**

Ref ID: WP -1-3

The User have weak password, making it vulnerable to attacks by brute force. Weak passwords can be easily guessed or cracked, providing unauthorized access to sensitive data stored on the system. This poses a serious threat to the confidentiality and integrity of data, potentially resulting in compromised user accounts, loss of critical information, and overall system instability.

As we found out before the password is very weak and found in almost all wordlists and cracking it won't take too much time.

**test test 1234-5678-2300-9000 21 street\_email@email.com**

### Vulnerability Details:

<b>Affects:</b>	http://testphp.vulnweb.com/login.php
<b>Attack Vectors</b>	Users with weak passwords

### Evidence

The screenshot shows a web application interface. At the top, there's a navigation bar with links: home, categories, artists, disclaimer, your cart, guestbook, AJAX Demo, and Logout test. Below the navigation bar, there's a search bar labeled "search art" with a "go" button. To the left of the main content area, there's a sidebar with links: Browse categories, Browse artists, Your cart, Signup, Your profile, Our guestbook, AJAX Demo, and a "Links" section with Security art, PHP scanner, PHP vuln help, and Fractal Explorer. The main content area displays the user profile for "John Cena (test)". It includes a message: "On this page you can visualize or edit you user information." Below this, there's a form with fields for Name, Credit card number, E-Mail, Phone number, and Address. The fields are pre-filled with: Name: John Cena, Credit card number: 1234-5678-2300-9000, E-Mail: email@email.com, Phone number: 2323345, and Address: 21 street. There's an "update" button at the bottom right of the form. Below the form, there's a message: "You have 2 items in your cart. You visualize you cart here." At the bottom of the page, there's a footer with links: About Us, Privacy Policy, Contact Us, and a copyright notice: ©2019 Acunetix Ltd.

A successful attack would consist in the following:

- Using any brute force tools to test weak passwords(ex: burp suite).

**Remediation Guidance:**

1-Require strong passwords: Implement a password policy that requires users to create strong passwords, with a minimum length of 12 characters, a combination of upper and lowercase letters, numbers, and symbols.

2-Enforce password complexity: Configure the system to enforce password complexity rules, such as disallowing commonly used passwords or requiring passwords to be changed regularly.

3-Use multi-factor authentication: Implement multi-factor authentication (MFA) to provide an additional layer of security. MFA requires users to provide two or more authentication factors, such as a password and a fingerprint or a one-time code sent to their phone.

## 2.4 Cross-Site Scripting (XSS) vulnerabilities

HIGH

Ref ID: XSS-1-4

the system is exposed to XSS attacks due to inadequate measures in input validation and output encoding. This makes it possible for attackers to inject harmful scripts or code into the web pages.

### Vulnerability Details:

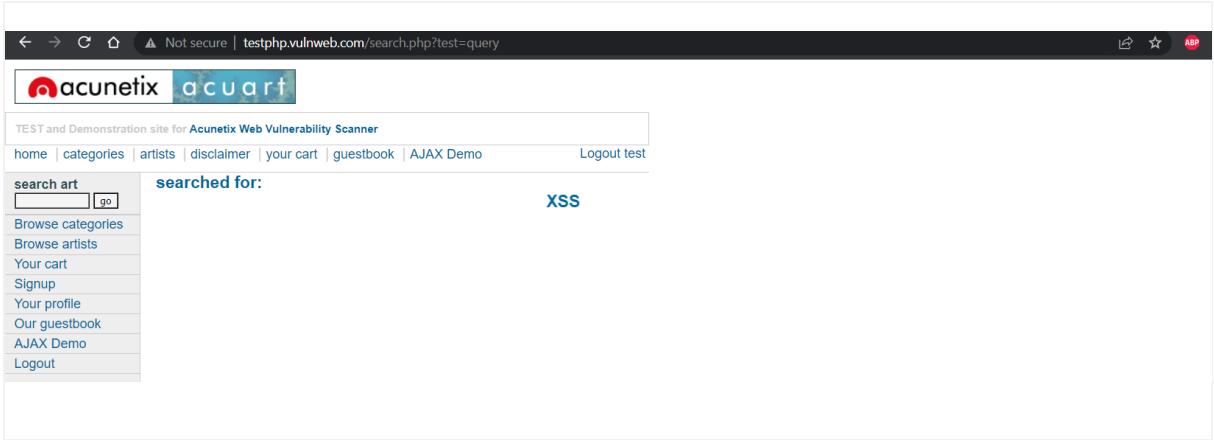
<b>Affects:</b>	http://testphp.vulnweb.com/search.php http://testphp.vulnweb.com/guestbook.php
<b>Parameter(s):</b>	<marquee onstart=alert(1)>XSS</marquee>

### Request.:

As we can see we can use any XSS payload in the search bar



### Evidence:



### Remediation Guidance:

- 1-Input validation: Implement strict input validation policies to ensure that user input is sanitized and free from any malicious code or script.
- 2-Output encoding: Encode output data using appropriate methods, such as HTML entity encoding or JavaScript escaping, to prevent script injection.
- 3-Content Security Policy (CSP): Utilize a CSP to restrict the execution of untrusted scripts and to mitigate the impact of any successful XSS attacks.



MEDIUM

Ref ID: SFD-1-5

It has been discovered that 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

### Vulnerability Details:

<b>Affects:</b>	<a href="http://testphp.vulnweb.com/admin/">http://testphp.vulnweb.com/admin/</a> <a href="http://testphp.vulnweb.com/ CVS/Root">http://testphp.vulnweb.com/ CVS/Root</a> <a href="http://testphp.vulnweb.com/secured/phpinfo.php">http://testphp.vulnweb.com/secured/phpinfo.php</a>
<b>References:</b>	<a href="https://github.com/v0re/dirb/blob/master/wordlists/common.txt">https://github.com/v0re/dirb/blob/master/wordlists/common.txt</a>

Request.:

## We used burp suit to find Critical Files Disclosure

The screenshot shows the Burp Suite interface with the 'Intruder' tab active. The 'Payloads' sub-tab is selected, displaying a list of 12 requests. The 'Attack' tab is also visible, showing the 'Results' list. The 'Payloads' list is filtered to show all items. The 'Attack' list is empty.

Request	Position	Payload	Status	Error	Timeout	Length	Comment
0			404			303	
1	1	!g!tignore	404			303	
2	1	!h!taccess	404			303	
3	1	!h!taccess	404			303	
4	1	!h!taccess	404			303	
5	1	!h!taccess	404			303	
6	1	!h!taccess	404			303	
7	1	!h!taccess	404			303	
8	1	!h!taccess	404			303	
9	1	!h!taccess	404			303	
10	1	!h!taccess	404			303	
11	1	!h!taccess	404			303	
12	1	!h!taccess	404			303	

Evidence:

Index of /admin/

and create.sql	11-May-2011 10:27	523
-------------------	-------------------	-----

**Remediation Guidance:**

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.

## MEDIUM

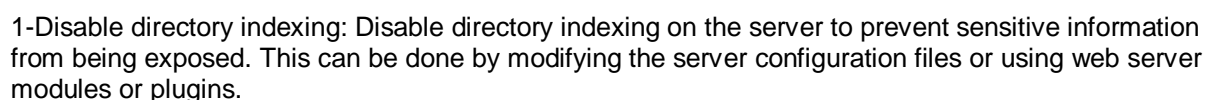
It has been identified that the system is susceptible to Directory Index disclosure, which could enable an unauthorized user to access sensitive information such as system configuration files, source code, and other critical data stored in the web server's directories.

**Affects:**

```
http://testphp.vulnweb.com/Flash/
http://testphp.vulnweb.com/CVS/
http://testphp.vulnweb.com/.idea/
```

<https://github.com/v0re/dirb/blob/master/wordlists/common.txt>

Same as before we used burp suit to find Directory Index Disclosure



2-Implement access controls: Ensure that sensitive directories and 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.

## 2.7 Personally Identifiable Information Disclosure

MEDIUM

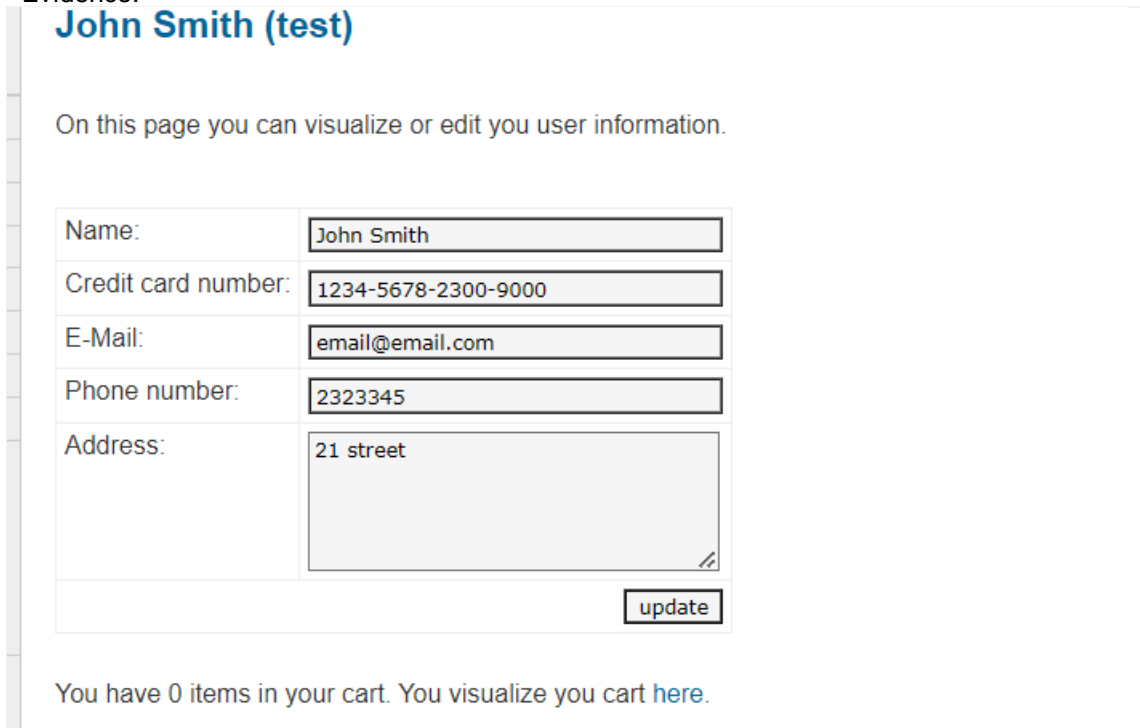
Ref ID: PIID-1-7

Information disclosure of personally identifiable information (PII) such as credit card numbers, addresses, full names, and phone numbers can be a serious security threat that can lead to identity theft, financial loss, and reputational damage.

### Vulnerability Details:

<b>Affects:</b>	<a href="http://testphp.vulnweb.com/Flash/">http://testphp.vulnweb.com/Flash/</a> <a href="http://testphp.vulnweb.com/CVS/">http://testphp.vulnweb.com/CVS/</a> <a href="http://testphp.vulnweb.com/.idea/">http://testphp.vulnweb.com/.idea/</a>
<b>References:</b>	<a href="https://github.com/v0re/dirb/blob/master/wordlists/common.txt">https://github.com/v0re/dirb/blob/master/wordlists/common.txt</a>

Evidence:



**John Smith (test)**

On this page you can visualize or edit you user information.

Name:	<input type="text" value="John Smith"/>
Credit card number:	<input type="text" value="1234-5678-2300-9000"/>
E-Mail:	<input type="text" value="email@email.com"/>
Phone number:	<input type="text" value="2323345"/>
Address:	<input type="text" value="21 street"/>

You have 0 items in your cart. You visualize you cart [here](#).

### Remediation Guidance:

- 1-Limit data retention: Implement policies to limit the retention of sensitive data to only what is necessary for business purposes. This can reduce the amount of data that is at risk of being exposed in the event of a security breach.
- 2-Conduct regular security audits: Regular security audits can help to identify vulnerabilities and security gaps in the system. This can help to identify areas where information disclosure may occur.
- 3-Use data masking: Implement data masking techniques to hide sensitive information from unauthorized users. This can include using techniques such as redaction or encryption to mask sensitive data