Building and Analyzing a Vulnerable Website

PREPARED FOR: National Cyber Security Center

Project name:	**Building and Analyzing a Vulnerable Website**
Authors	Ghayda Habes Alquaissi , Abdullah Riyad Tarawneh , Wessam Mohammad Alkhushman
Approved by	Abdalazez irsheidat , Hesham Abubaker
Version	1.0
Submission Date	September 15/2024

Table of Contents:

1.Abstract.

This report documents the results of the test. The penetration test was conducted on September 15, 2024, and a series of tests were conducted on vuln web Site, a vulnerable website built for security testing purposes using testing tools and manual testing techniques, to identify actual or potentially exploitable security vulnerabilities, which if exploited could result in direct or indirect damage to the customer's name.

2. Introduction.

In this task we have to build a vulnerable website called Vuln web Site, this website was built by some students who are trainees in the (Masar) program of the National Cyber Security Center. This website contains a number of common vulnerabilities in the world of cyber security that some people use for hacking purposes and causing harm to users or companies. This website is a simple example of a vulnerable website. Below is a detailed report of the results we got during this process.

3. Methodology.

1.At first, we built a vulnerable website using phpStorm , which are programs dedicated to creating web pages and websites that contain many languages used in websites such as HTML, JavaScript, CSS, PHP, and other languages. The site was named Vulne web Site. This site contains some common and harmful security vulnerabilities. It required us to build the site for some important programs such as Xampp and use PHP My Admin to build a database that is linked to the site to perform some operations such as logging in. While building the site, we put some common security vulnerabilities (XSS (Cross-Site Scripting),SQL injection ,File Upload Vulnerability, . After making sure that the site is working well and that the vulnerabilities are working as required, web scanners such as OWASP ZAP and Skipfish were used to take an official report on the extent of the site's weakness and how many vulnerabilities it contains and take a more detailed report on the status of the site that was built. After that, a comprehensive assessment of all discovered vulnerabilities was conducted and compared in terms of the following criteria.

1) **Description:** This is a detailed explanation of a security vulnerability or technical issue. In the context of vulnerabilities, it describes the nature of the flaw and how it affects the system or software. The description usually outlines the potential damage or impact if the vulnerability is exploited.
2) **CVE (Common Vulnerabilities and Exposures):**
   CVE: This is a standardized international system for identifying known vulnerabilities and exposures. Each vulnerability is assigned a unique identifier called a CVE ID, such as CVE-2023-1234. CVE provides a universal reference framework that allows security researchers and organizations to discuss vulnerabilities consistently across different systems and software.

3) **CVSS (Common Vulnerability Scoring System):**
   CVSS: This is a scoring system used to assess the severity of security vulnerabilities, with scores ranging from 0.0 (least severe) to 10.0 (most severe). CVSS provides a numerical value to describe the potential impact of a vulnerability.

The score is based on factors such as exploitability, confidentiality impact, integrity impact, and system availability.

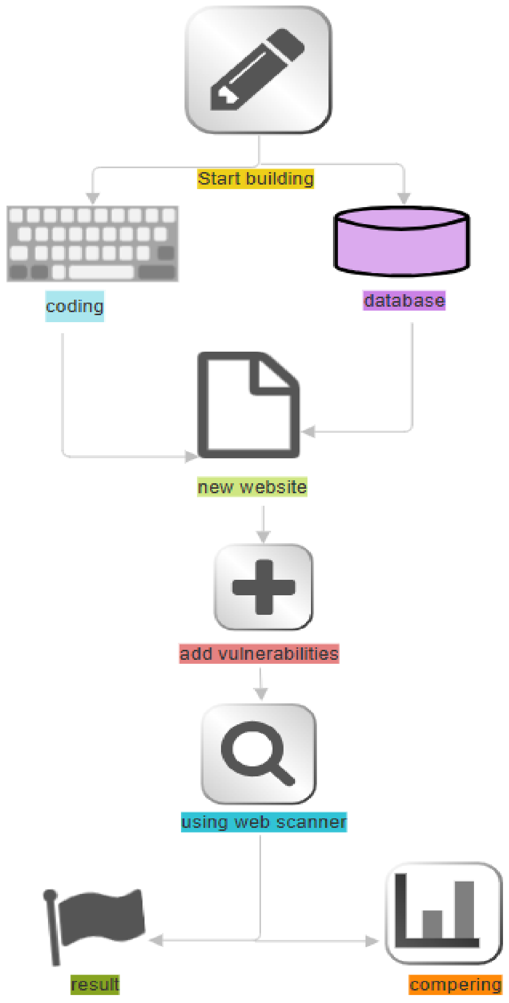
a) Main components of CVSS:
b) Base Score: The fundamental score that reflects the intrinsic characteristics of a vulnerability.
c) Temporal Score: Adjusts the Base Score based on factors like exploit maturity.
d) Environmental Score: Customizes the score based on the specific environment the vulnerability impacts.

4) Severity

Severity: Refers to the level of impact a vulnerability can have, based on the CVSS score. The severity is typically classified into:

- **Low**: Minimal impact, not a significant threat.
- **Medium**: Moderate impact, requires attention.
- **High**: Severe impact, should be addressed quickly.
- **Critical**: Very high impact, poses an immediate and serious risk to system security, and requires urgent action.

The following figure shows how this approach was implemented, Figure (1).



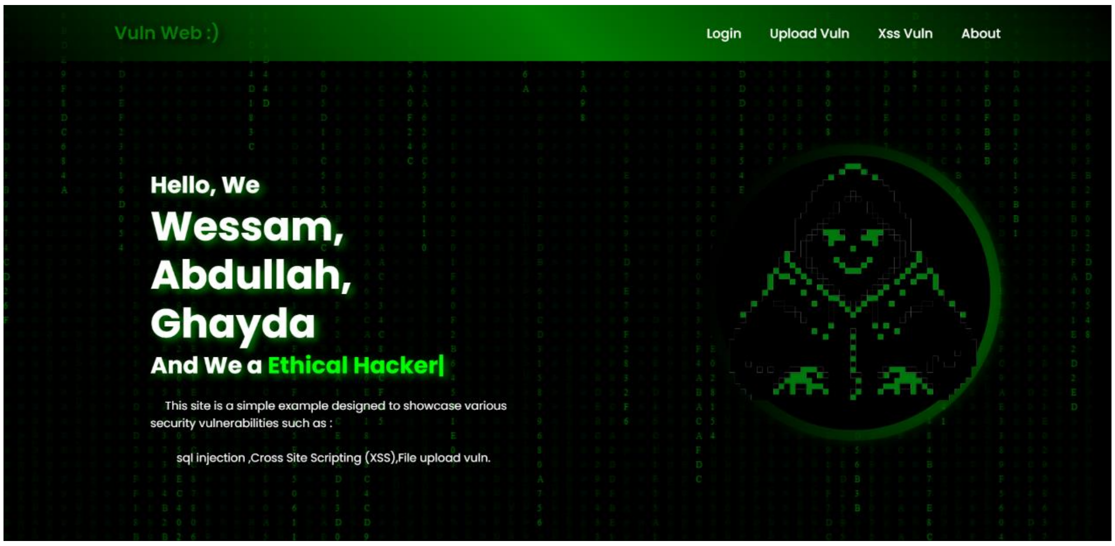
3.Tools :

The following section represents all the tools we used to get the access permission and then escalate the privileges to the maximum user:

1. Visual Studio Code.
2. Xampp.
3. Programming languages(HTML ,CSS, JavaScript,.PHP,SQL)
4. OWASP ZAP
5. Skipfish

4.Implementation:

First we wrote the front page of our website where we added a paragraph to introduce the website which contains icons of pages with security vulnerabilities. The following image shows the following: Figure(2).



Hello, We
Wessam,
Abdullah,
Ghayda
And We a Ethical Hacker|

This site is a simple example designed to showcase various security vulnerabilities such as :

sql injection ,Cross Site Scripting (XSS),File upload vuln.

Then we created SQL injection vulnerability, The following figure shows how the vulnerability works, figure(3) and figure (4)

Figure (3)

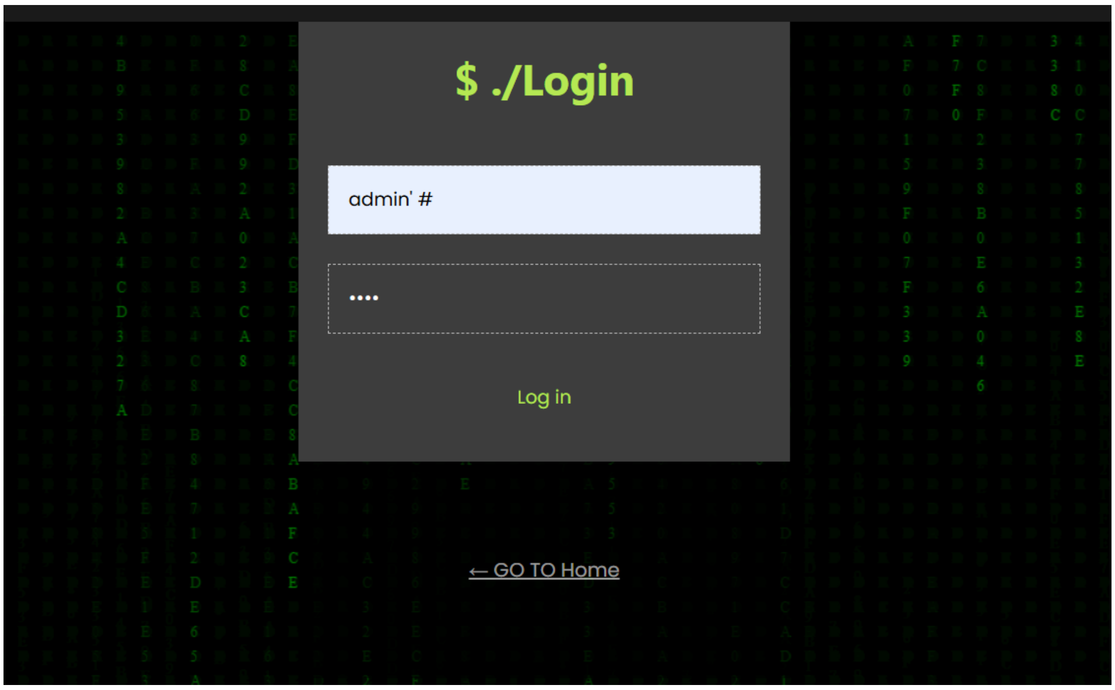
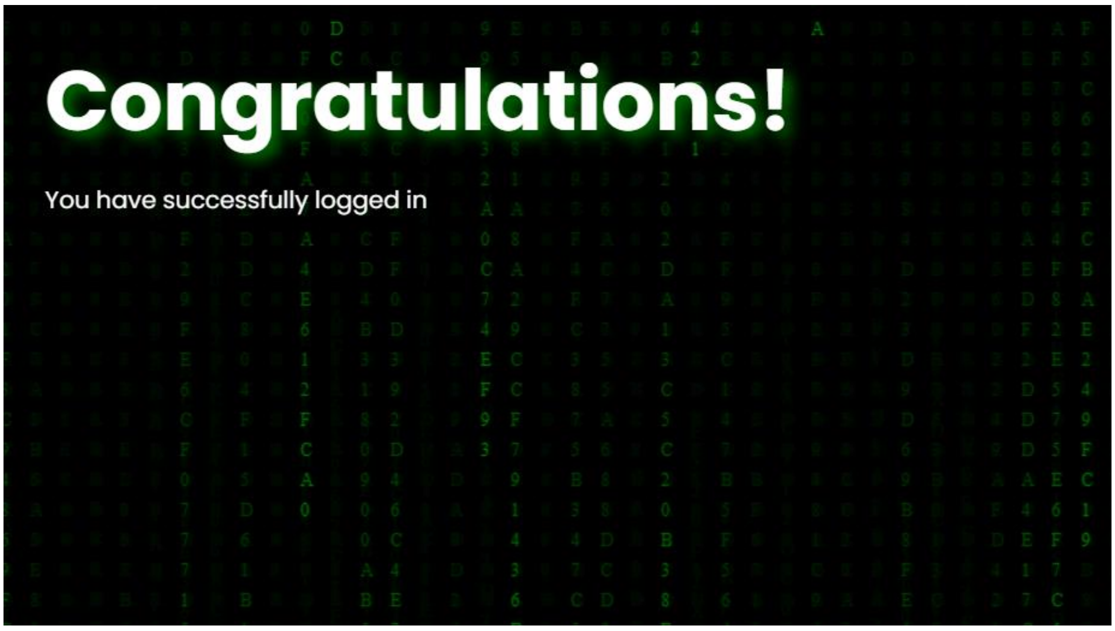


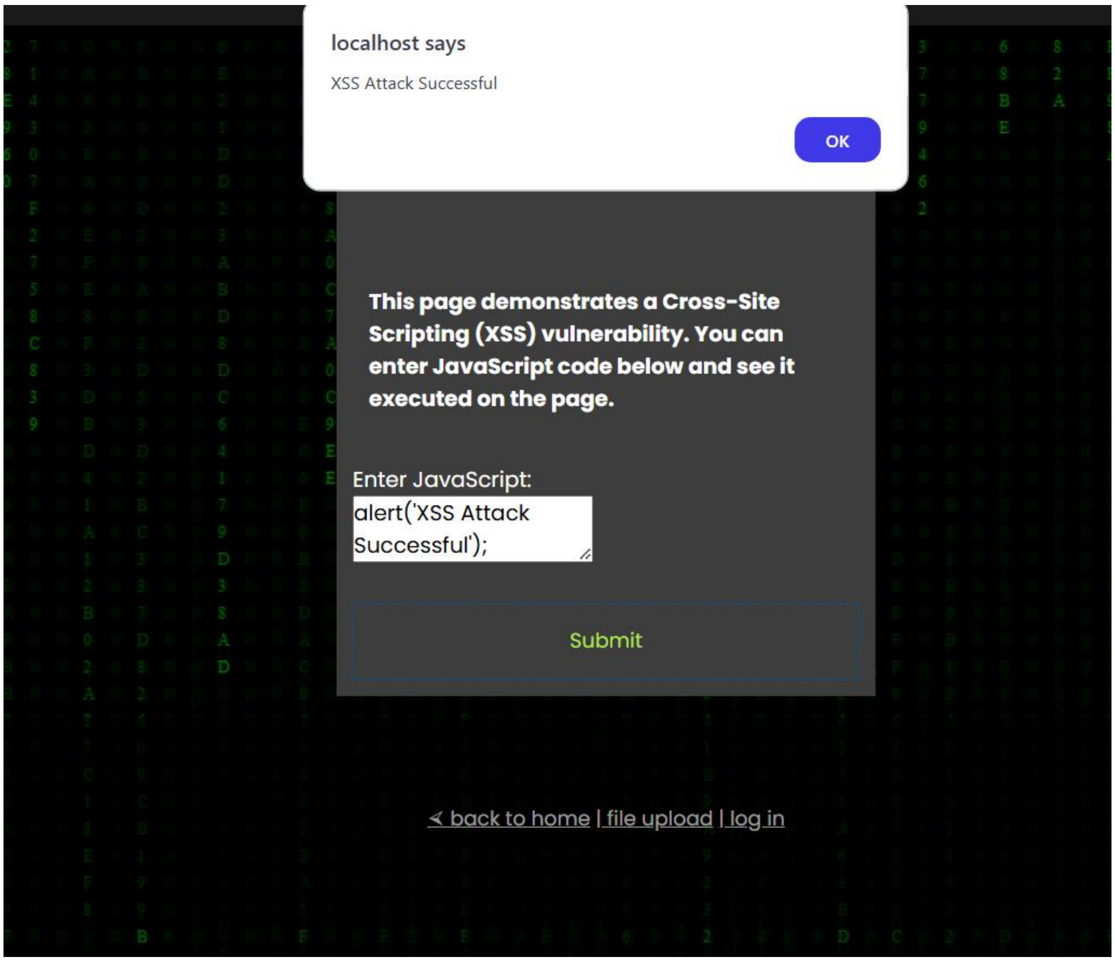
Figure (4)


Congratulations!

You have successfully logged in

# SELECT * FROM users WHERE username='admin' #' AND password='$pass'

Then we created the XSS vulnerability. The following figure shows how it works. Figure (5)



localhost says

XSS Attack Successful

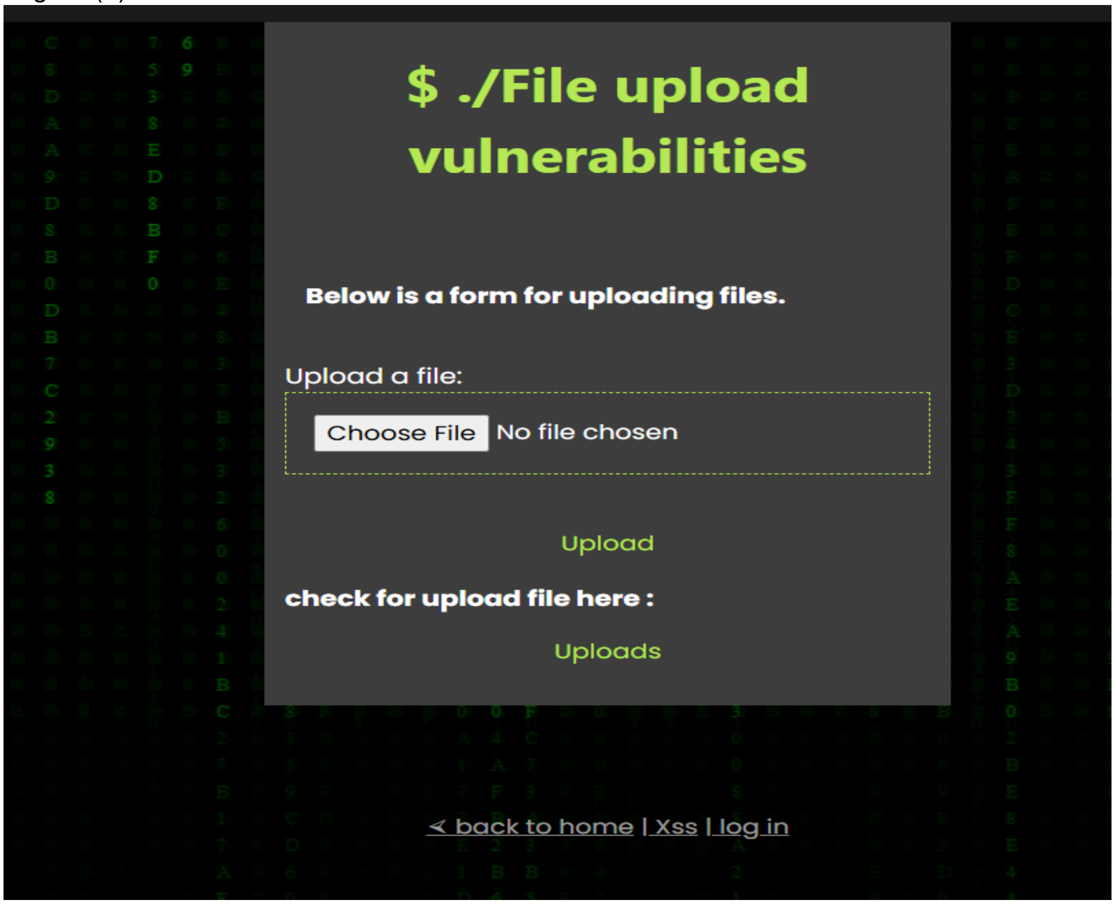
This page demonstrates a Cross-Site Scripting (XSS) vulnerability. You can enter JavaScript code below and see it executed on the page.

Enter JavaScript:

```
alert('XSS Attack Successful');
```

Then we created File Upload Vulnerability . The following figure shows how it works. Figure (6),(7),(8),(9)

Figure (6)


$ ./File upload vulnerabilities

Below is a form for uploading files.

Upload a file:

Choose File No file chosen

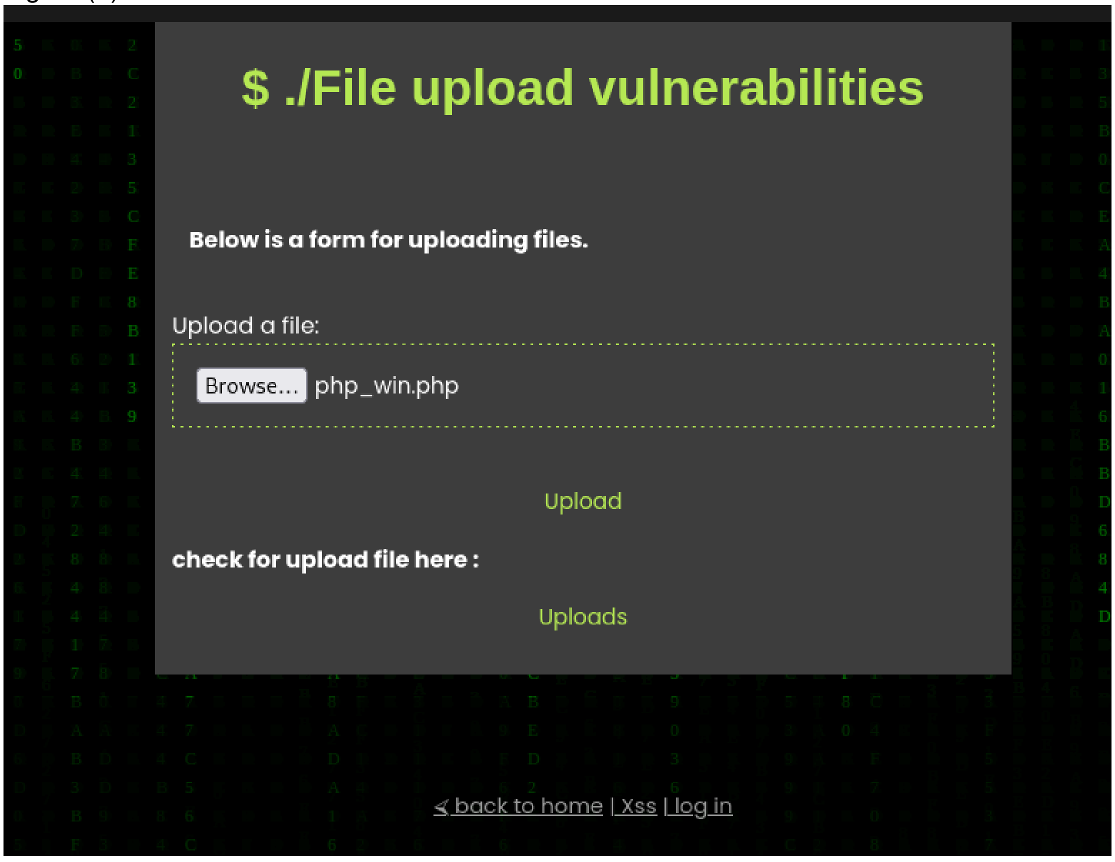
Upload

check for upload file here :

Uploads

< back to home | Xss | log in

Figure (7)


$ ./File upload vulnerabilities

Below is a form for uploading files.

Upload a file:

Browse... php_win.php

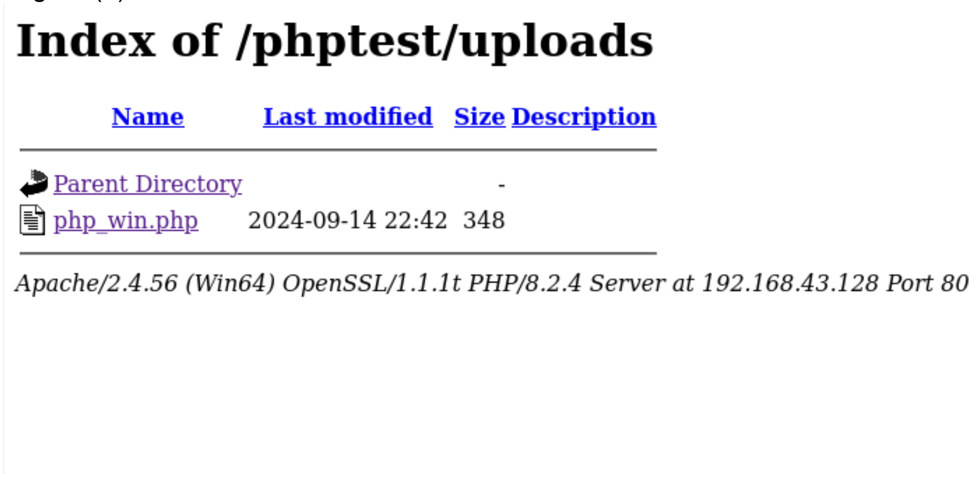
Upload

check for upload file here :

Uploads

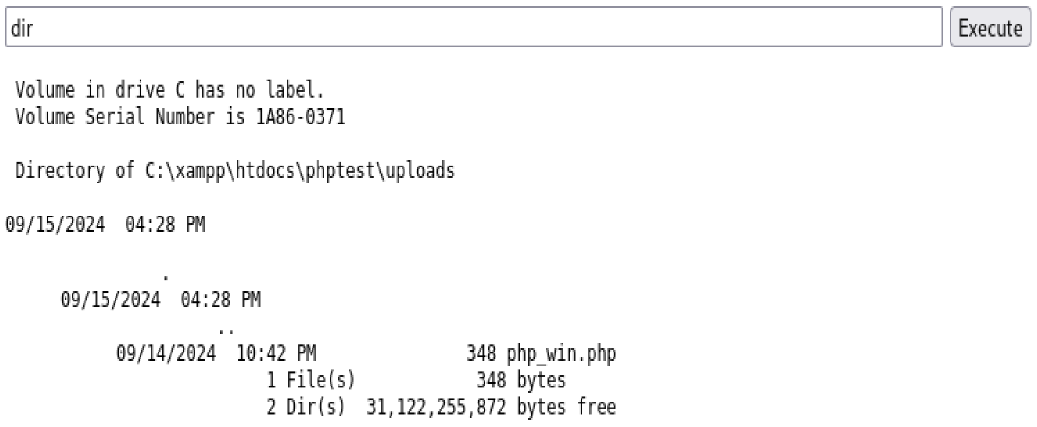
< back to home | Xss | log in

Figure (8)



```
Index of /phptest/uploads

      Name            Last modified     Size Description
----------------------------------------------------------
 Parent Directory                         -
 php_win.php       2024-09-14 22:42    348
----------------------------------------------------------
Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.2.4 Server at 192.168.43.128 Port 80
```

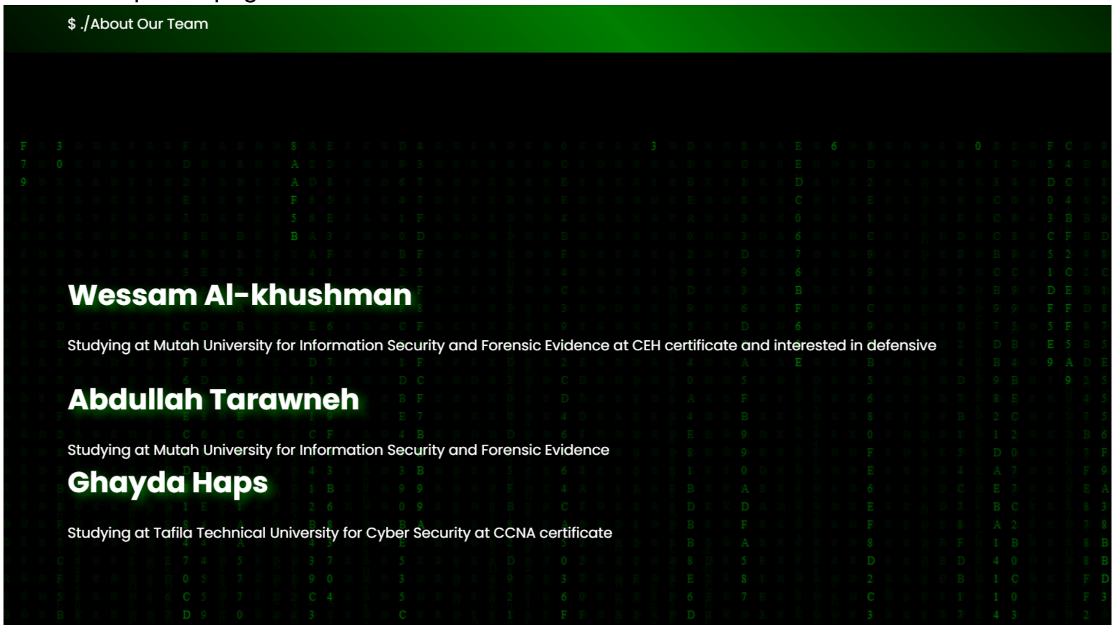
Figure (9)



```
 Volume in drive C has no label.
 Volume Serial Number is 1A86-0371

 Directory of C:\xampp\htdocs\phptest\uploads

09/15/2024  04:28 PM

                .
     09/15/2024  04:28 PM
                      ..
          09/14/2024  10:42 PM               348 php_win.php
                         1 File(s)            348 bytes
                         2 Dir(s)  31,122,255,872 bytes free
```

# We upload php extension payload

This is our profile page.



$ ./About Our Team

Wessam Al-khushman

Studying at Mutah University for Information Security and Forensic Evidence at CEH certificate and interested in defensive

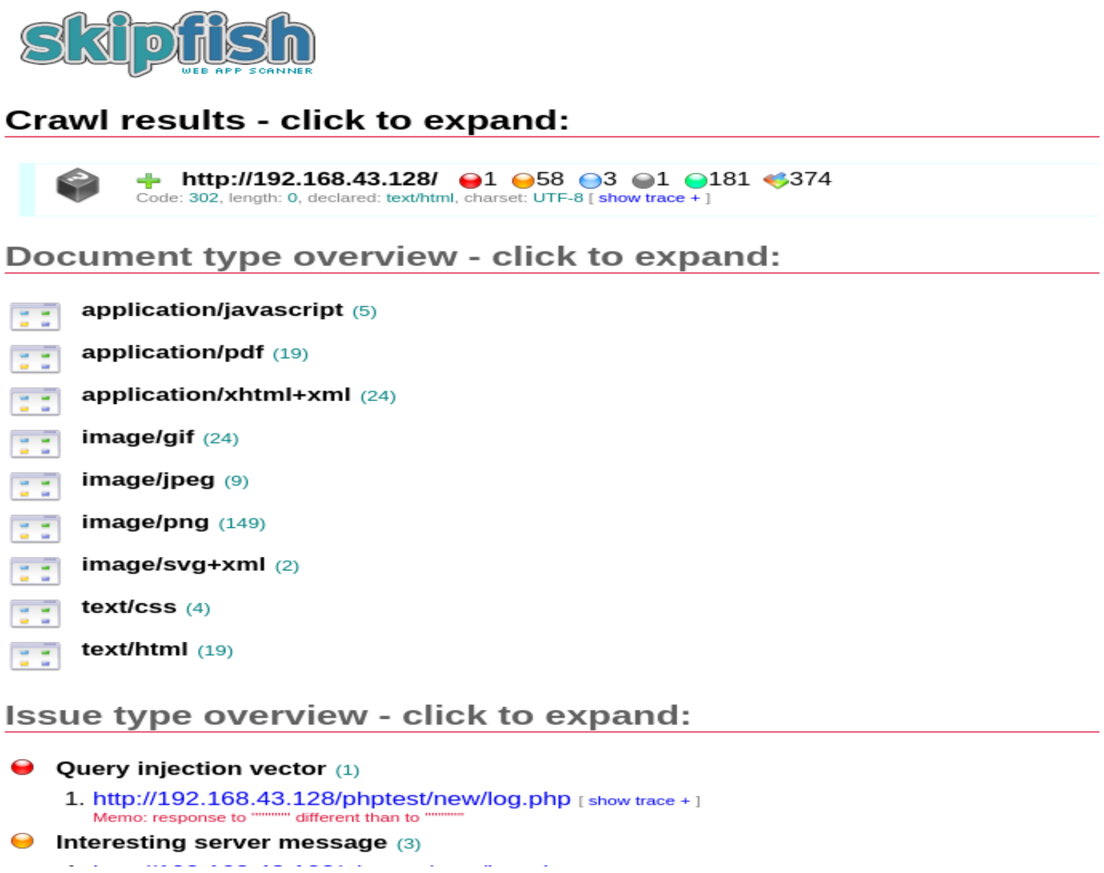
Abdullah Tarawneh

Studying at Mutah University for Information Security and Forensic Evidence

Ghayda Haps

Studying at Tafila Technical University for Cyber Security at CCNA certificate

After we finished building the site and made sure all vulnerabilities were working properly, we ran a site scan using the OWASP ZAP and Skipfish The following figure shows how it works. Figure(10),(11),(12),(13).



skipfish WEB APP SCANNER

Crawl results - click to expand:

http://192.168.43.128/ 🔴1 🟠58 🔵3 ⚪1 🟢181 374
Code: 302, length: 0, declared: text/html, charset: UTF-8 [show trace +]

Document type overview - click to expand:

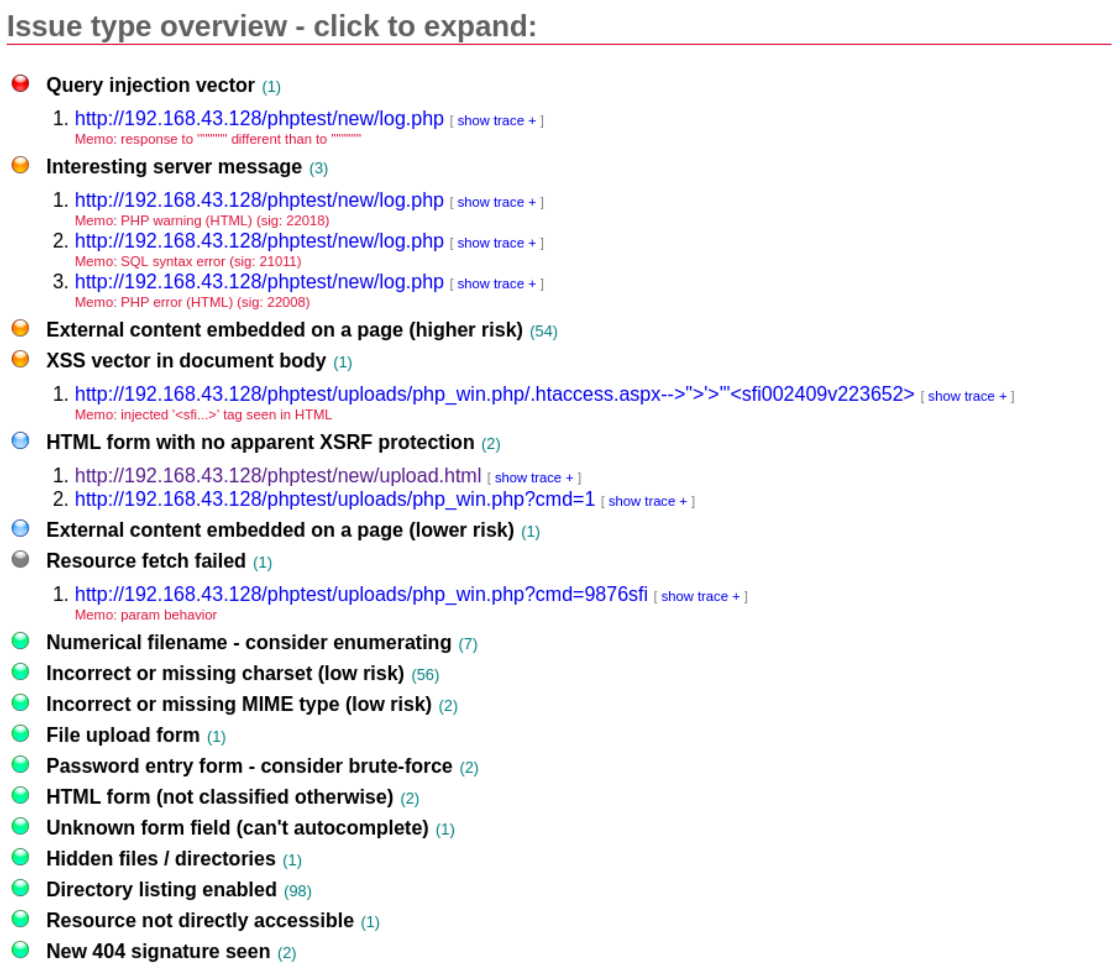
application/javascript (5)
application/pdf (19)
application/xhtml+xml (24)
image/gif (24)
image/jpeg (9)
image/png (149)
image/svg+xml (2)
text/css (4)
text/html (19)

Issue type overview - click to expand:

🔴 Query injection vector (1)
1. http://192.168.43.128/phptest/new/log.php [show trace +]
Memo: response to '''''''''' different than to ''''''''''

🟠 Interesting server message (3)

Figure (11)



Issue type overview - click to expand:

● Query injection vector (1)

1. http://192.168.43.128/phptest/new/log.php [show trace +]
   Memo: response to "''''''''" different than to "''''''''"

● Interesting server message (3)

1. http://192.168.43.128/phptest/new/log.php [show trace +]
   Memo: PHP warning (HTML) (sig: 22018)
2. http://192.168.43.128/phptest/new/log.php [show trace +]
   Memo: SQL syntax error (sig: 21011)
3. http://192.168.43.128/phptest/new/log.php [show trace +]
   Memo: PHP error (HTML) (sig: 22008)

● External content embedded on a page (higher risk) (54)

● XSS vector in document body (1)

1. http://192.168.43.128/phptest/uploads/php_win.php/.htaccess.aspx-->">'>'"<sfi002409v223652> [show trace +]
   Memo: injected '<sfi...>' tag seen in HTML

● HTML form with no apparent XSRF protection (2)

1. http://192.168.43.128/phptest/new/upload.html [show trace +]
2. http://192.168.43.128/phptest/uploads/php_win.php?cmd=1 [show trace +]

● External content embedded on a page (lower risk) (1)

● Resource fetch failed (1)

1. http://192.168.43.128/phptest/uploads/php_win.php?cmd=9876sfi [show trace +]
   Memo: param behavior

● Numerical filename - consider enumerating (7)

● Incorrect or missing charset (low risk) (56)

● Incorrect or missing MIME type (low risk) (2)

● File upload form (1)

● Password entry form - consider brute-force (2)

● HTML form (not classified otherwise) (2)

● Unknown form field (can't autocomplete) (1)

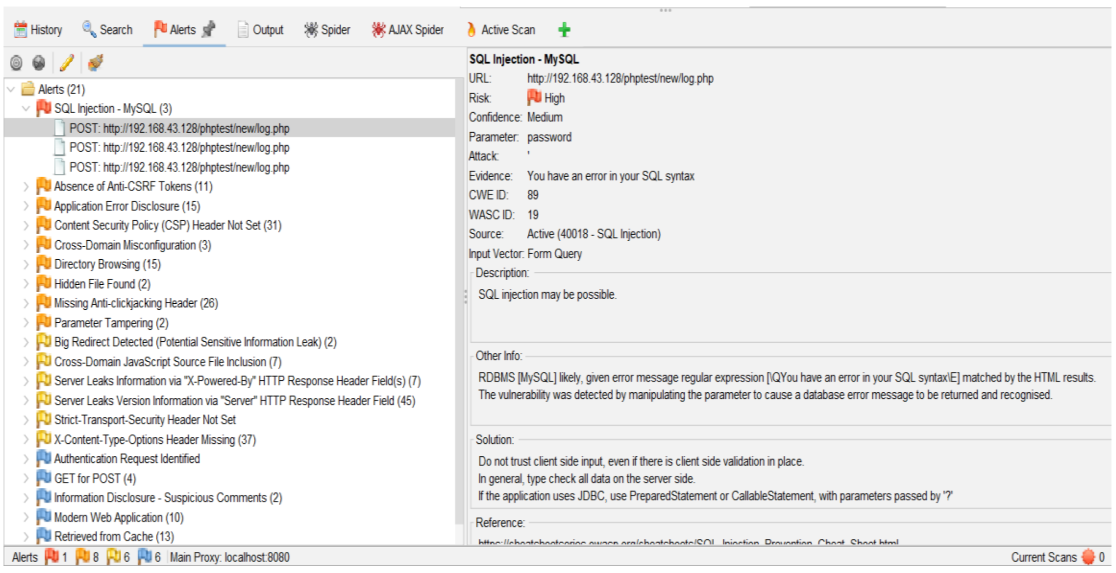
● Hidden files / directories (1)

● Directory listing enabled (98)

● Resource not directly accessible (1)

● New 404 signature seen (2)

Figure (12)



Alerts (21)
- SQL Injection - MySQL (3)
  - POST: http://192.168.43.128/phptest/new/log.php
  - POST: http://192.168.43.128/phptest/new/log.php
  - POST: http://192.168.43.128/phptest/new/log.php
- Absence of Anti-CSRF Tokens (11)
- Application Error Disclosure (15)
- Content Security Policy (CSP) Header Not Set (31)
- Cross-Domain Misconfiguration (3)
- Directory Browsing (15)
- Hidden File Found (2)
- Missing Anti-clickjacking Header (26)
- Parameter Tampering (2)
- Big Redirect Detected (Potential Sensitive Information Leak) (2)
- Cross-Domain JavaScript Source File Inclusion (7)
- Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s) (7)
- Server Leaks Version Information via "Server" HTTP Response Header Field (45)
- Strict-Transport-Security Header Not Set
- X-Content-Type-Options Header Missing (37)
- Authentication Request Identified
- GET for POST (4)
- Information Disclosure - Suspicious Comments (2)
- Modern Web Application (10)
- Retrieved from Cache (13)

SQL Injection - MySQL

URL:	http://192.168.43.128/phptest/new/log.php
Risk:	High
Confidence:	Medium
Parameter:	password
Attack:	'
Evidence:	You have an error in your SQL syntax
CWE ID:	89
WASC ID:	19
Source:	Active (40018 - SQL Injection)
Input Vector:	Form Query

Description:

SQL injection may be possible.

Other Info:

RDBMS [MySQL] likely, given error message regular expression [\QYou have an error in your SQL syntax\E] matched by the HTML results.
The vulnerability was detected by manipulating the parameter to cause a database error message to be returned and recognised.

Solution:

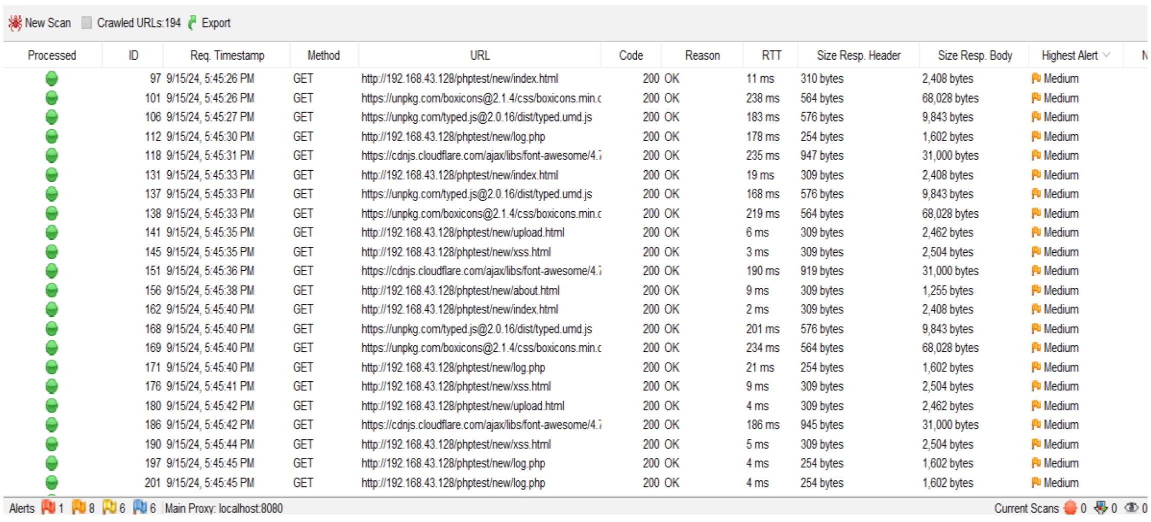
Do not trust client side input, even if there is client side validation in place.
In general, type check all data on the server side.
If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?'

Reference:

https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

Alerts 1 8 6 6 Main Proxy: localhost:8080 Current Scans 0

Figure (13)



New Scan | Crawled URLs:194 | Export

Processed	ID	Req. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert
●	97	9/15/24, 5:45:26 PM	GET	http://192.168.43.128/phptest/new/index.html	200	OK	11 ms	310 bytes	2,408 bytes	Medium
●	101	9/15/24, 5:45:26 PM	GET	https://unpkg.com/boxicons@2.1.4/css/boxicons.min.c	200	OK	238 ms	564 bytes	68,028 bytes	Medium
●	106	9/15/24, 5:45:27 PM	GET	https://unpkg.com/typed.js@2.0.16/dist/typed.umd.js	200	OK	183 ms	576 bytes	9,843 bytes	Medium
●	112	9/15/24, 5:45:30 PM	GET	http://192.168.43.128/phptest/new/log.php	200	OK	178 ms	254 bytes	1,602 bytes	Medium
●	118	9/15/24, 5:45:31 PM	GET	https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7	200	OK	235 ms	947 bytes	31,000 bytes	Medium
●	131	9/15/24, 5:45:33 PM	GET	http://192.168.43.128/phptest/new/index.html	200	OK	19 ms	309 bytes	2,408 bytes	Medium
●	137	9/15/24, 5:45:33 PM	GET	https://unpkg.com/typed.js@2.0.16/dist/typed.umd.js	200	OK	168 ms	576 bytes	9,843 bytes	Medium
●	138	9/15/24, 5:45:33 PM	GET	https://unpkg.com/boxicons@2.1.4/css/boxicons.min.c	200	OK	219 ms	564 bytes	68,028 bytes	Medium
●	141	9/15/24, 5:45:35 PM	GET	http://192.168.43.128/phptest/new/upload.html	200	OK	6 ms	309 bytes	2,462 bytes	Medium
●	145	9/15/24, 5:45:35 PM	GET	http://192.168.43.128/phptest/new/xss.html	200	OK	3 ms	309 bytes	2,504 bytes	Medium
●	151	9/15/24, 5:45:36 PM	GET	https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7	200	OK	190 ms	919 bytes	31,000 bytes	Medium
●	156	9/15/24, 5:45:38 PM	GET	http://192.168.43.128/phptest/new/about.html	200	OK	9 ms	309 bytes	1,255 bytes	Medium
●	162	9/15/24, 5:45:40 PM	GET	http://192.168.43.128/phptest/new/index.html	200	OK	2 ms	309 bytes	2,408 bytes	Medium
●	168	9/15/24, 5:45:40 PM	GET	https://unpkg.com/typed.js@2.0.16/dist/typed.umd.js	200	OK	201 ms	576 bytes	9,843 bytes	Medium
●	169	9/15/24, 5:45:40 PM	GET	https://unpkg.com/boxicons@2.1.4/css/boxicons.min.c	200	OK	234 ms	564 bytes	68,028 bytes	Medium
●	171	9/15/24, 5:45:40 PM	GET	http://192.168.43.128/phptest/new/log.php	200	OK	21 ms	254 bytes	1,602 bytes	Medium
●	176	9/15/24, 5:45:41 PM	GET	http://192.168.43.128/phptest/new/xss.html	200	OK	9 ms	309 bytes	2,504 bytes	Medium
●	180	9/15/24, 5:45:42 PM	GET	http://192.168.43.128/phptest/new/upload.html	200	OK	4 ms	309 bytes	2,462 bytes	Medium
●	186	9/15/24, 5:45:42 PM	GET	https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7	200	OK	186 ms	945 bytes	31,000 bytes	Medium
●	190	9/15/24, 5:45:44 PM	GET	http://192.168.43.128/phptest/new/xss.html	200	OK	5 ms	309 bytes	2,504 bytes	Medium
●	197	9/15/24, 5:45:45 PM	GET	http://192.168.43.128/phptest/new/log.php	200	OK	4 ms	254 bytes	1,602 bytes	Medium
●	201	9/15/24, 5:45:45 PM	GET	http://192.168.43.128/phptest/new/log.php	200	OK	4 ms	254 bytes	1,602 bytes	Medium

Alerts 1 8 6 6 Main Proxy: localhost:8080 | Current Scans 0 0 0

5.Risk Exploitable Vulnerability

After conducting the scan and extracting the results, we compared the vulnerabilities found on this site through the table shown below.

Vulnerability	XSS (Cross-Site Scripting)	SQL Injection	File Upload Vulnerability
Description	XSS allows attackers to inject malicious scripts into web pages viewed by other users. This can be used to steal cookies, session tokens, or other sensitive information.	JavaScript Injection involves inserting malicious JavaScript code into a web application. This can lead to unauthorized actions being performed on behalf of the user or sensitive information being leaked.	File Upload Vulnerability allows attackers to upload malicious files to a server, potentially leading to remote code execution or exposure of sensitive data.
CVE	CVE-2020-11022	CVE-2021-22987	**CVE-2020-11081**
CVSS	4.0 _7.0	5.0_7.0	**6.0_8.0**
Severity	high	High	**High**

6.CONCLUSION

In conclusion, the security assessment of [Site Name] revealed several critical vulnerabilities, XSS, SQL Injection, and File Upload Vulnerability. These issues pose significant risks and require immediate remediation to enhance the site's security and protect sensitive data.

7. Reference

- https://cupc4k3.medium.com/html-injection-vulnerability-in-kanboard-group-management-d9fe5154bb1b
- https://www.cvedetails.com/vulnerability-list/opxss-1/xss.html
- https://vuldb.com/?id.254098
- https://www.cvedetails.com/cve/CVE-2021-35237/
- https://nvd.nist.gov/vuln/detail/CVE-2022-22797
- https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=file+upload