Vulnerability Report on foophones.securitybrigade.com:8080

1)CLICKJACKING

**Vulnerable Url**- "foophone.securitybrigade.com:8080"

**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 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**-

Using a similar technique, keystrokes can also be hijacked. With a carefully crafted combination of stylesheets, iframes, and text boxes, a user can be led to believe they are typing in the password to their email or bank account, but are instead typing into an invisible frame controlled by the attacker.

**Prevention-**

1)Sending the proper Content Security Policy (CSP) frame-ancestors directive response headers that instruct the browser to not allow framing from other domains. (This replaces the older X-Frame-Options HTTP headers.)

2)Employing defensive code in the UI to ensure that the current frame is the most top level window

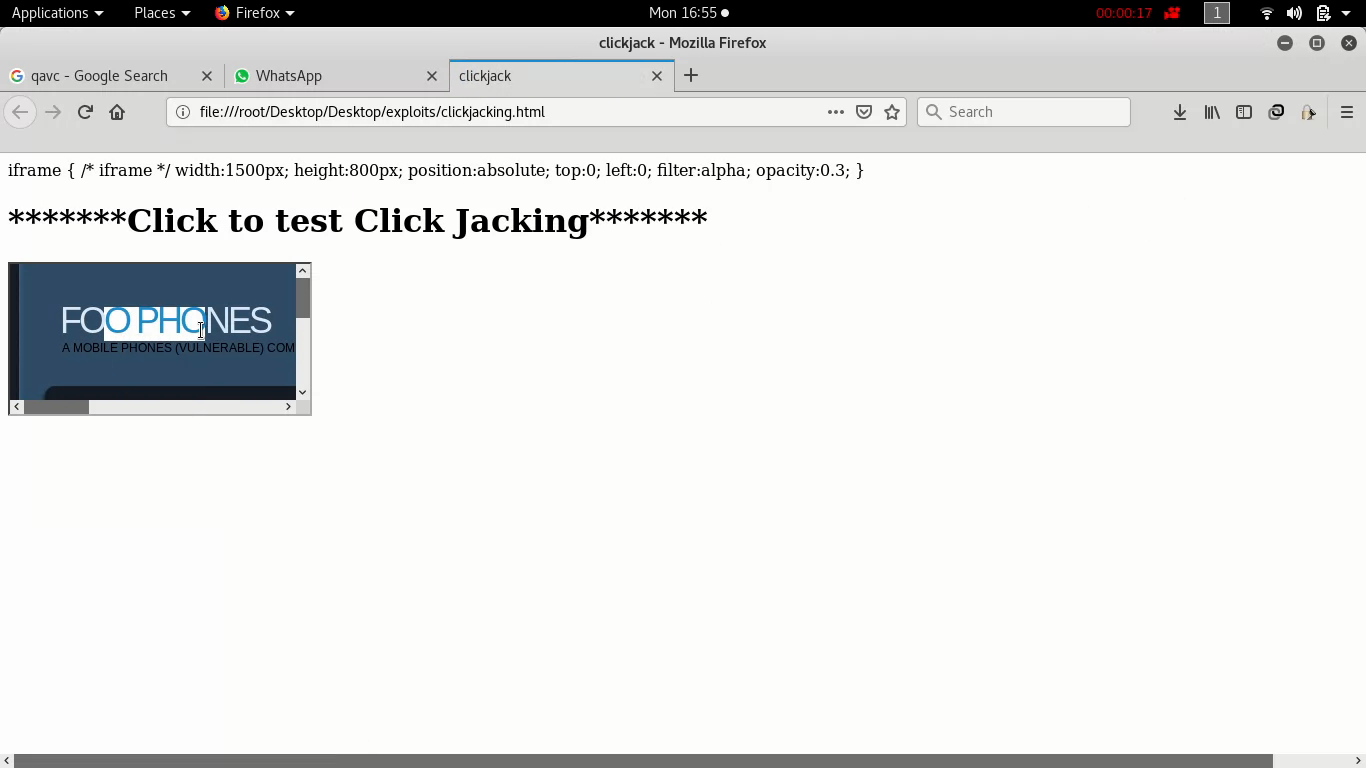
**P.O.C**

1)Take a normal html page ,that has particular setting of display as in the exploit below.

2)Embed the name of the website within the iframe tag.

3)Just open the page.

4)Video is also attached to the report to demonstrate this process.



Clickjacking

2) USERNAME POLICY BYPASS

**Vulnerable endpoint**- "foophones.securitybrigade.com:8080/register.php"

**Description-**

The username has text field has '\*\*' infront of it.Which means it should have atleast 4 letter in its name as per policy of website.

If a user inputs 3 word letter or a 2 word letter than it should be rejected.But here the client side as well and server side are not checking the user of username , hence it's easily bypassed as I entered the 3 letter username it also got accepted.

**P.O.C-**

1)Open the register page.

2)Enter the username "cyb"

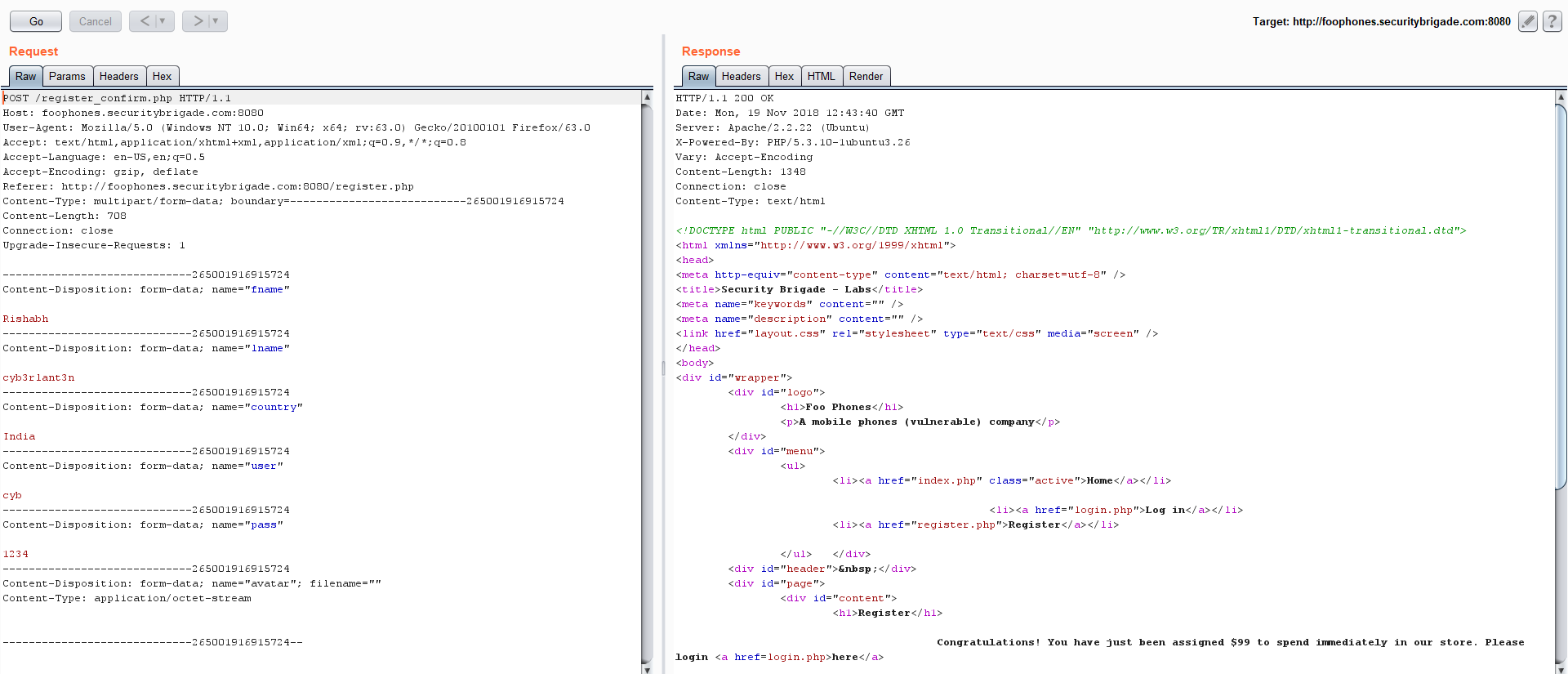
3)check the response

4)The user actually gets registered with 3 letter username.

5)A pic"usernamebypass.png" has been attached to this report to demonstrate the process

**Prevention**-

Implementation of server side as well as client side checks on the length of input.



Username policy bypass

3)SENSITIVE DATA EXPOSURE(NO HTTPS(ENCRYPTION PROTOCOL))

**Vulnerable url**-"foophones.securitybrigade.com:8080/login.php"

**Description**-

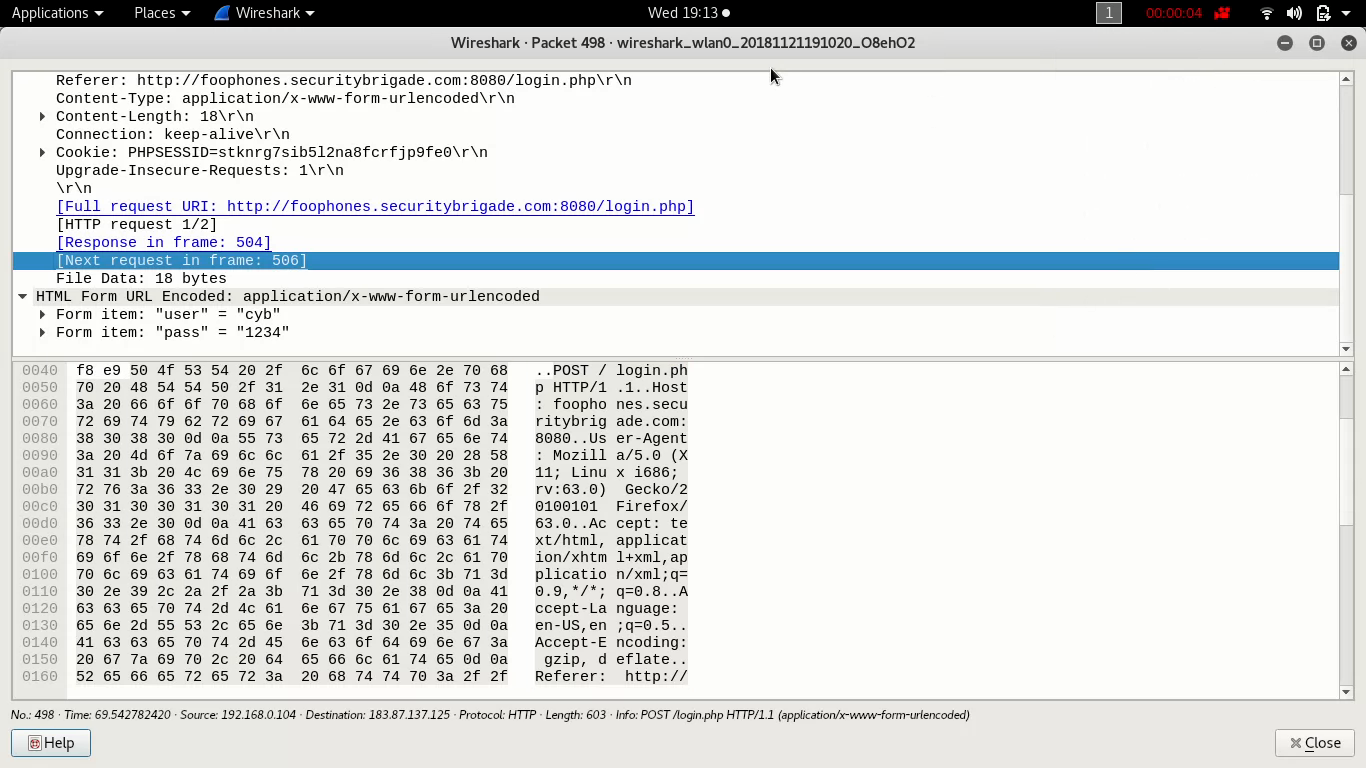
The website given above does not uses https or any other form of encryption. As a result of which if an attacker intercepts a user request to the website he will be easily able to get the user credentials as well as other important details.

**P.O.C-**

1)Set up a open wifi and wireshark in the monitor mode.

2)As soon as user connects to the website by entering his credentials , his credntials can be easily seen on the interface of wireshark.

3)The screenshot nohttps.png has been attached to demonstrate this.



Password in plain text captured

4)NO HTTP SECURE FLAG HAS BEEN USED

**Vulnerable url**- http://foophones.securitybrigade.com:8080/

**Description-**

The website above does not uses http secure flag.

Even if the https is being used or not the http secure flag must be set.Because if the http secure flag is not set then when the user vists any site within the scope of cookie , and attacker manipulates the user to send some malicious/unwanted request then the user will ultimately end up submitting cookies in the plain encrypted form.Thats why it is neccessary to http secure flag set.

**P.O.C-**

1)Set up a burp proxy.

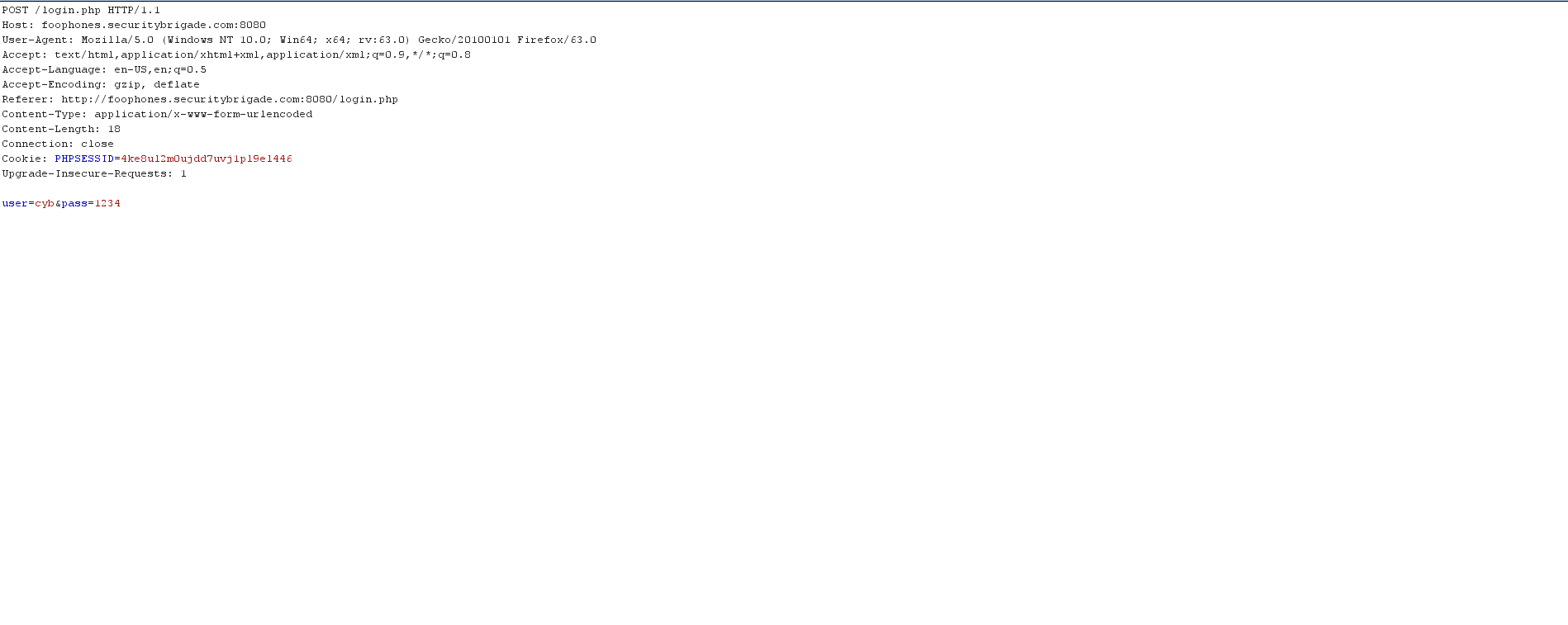
2)Enter the credentials on the browser and intercept in burpsuite.

3)You can easily see no htp secure flag is being set.

4)A screenshot "nosecureflag.png" has been attached to demonstrate the process.

**Remediation-**

Use http secure flag for all request.



No http secure flag

5)NO RATE LIMIT ON LOGIN

**Vulnerable url-**

http://foophones.securitybrigade.com:8080/login.php

**Description-**

The login page should have a fixed number of login attempt within a particular timeframe.After a fixed number of false login attempts the account shouldbe locked out .

**Impact-**

If no rate limit is being imposed ,then login form would be easily vulnerable to bruteforce attack.

**P.O.C-**

1)Set up the burp proxy .

2)Capture the login request .

3)Send it to intruder and take password field into consideration.

4)Now use your own wordlist and start the bruteforce atack.

5)My word list has total 58 words out which only last one is the correct password.

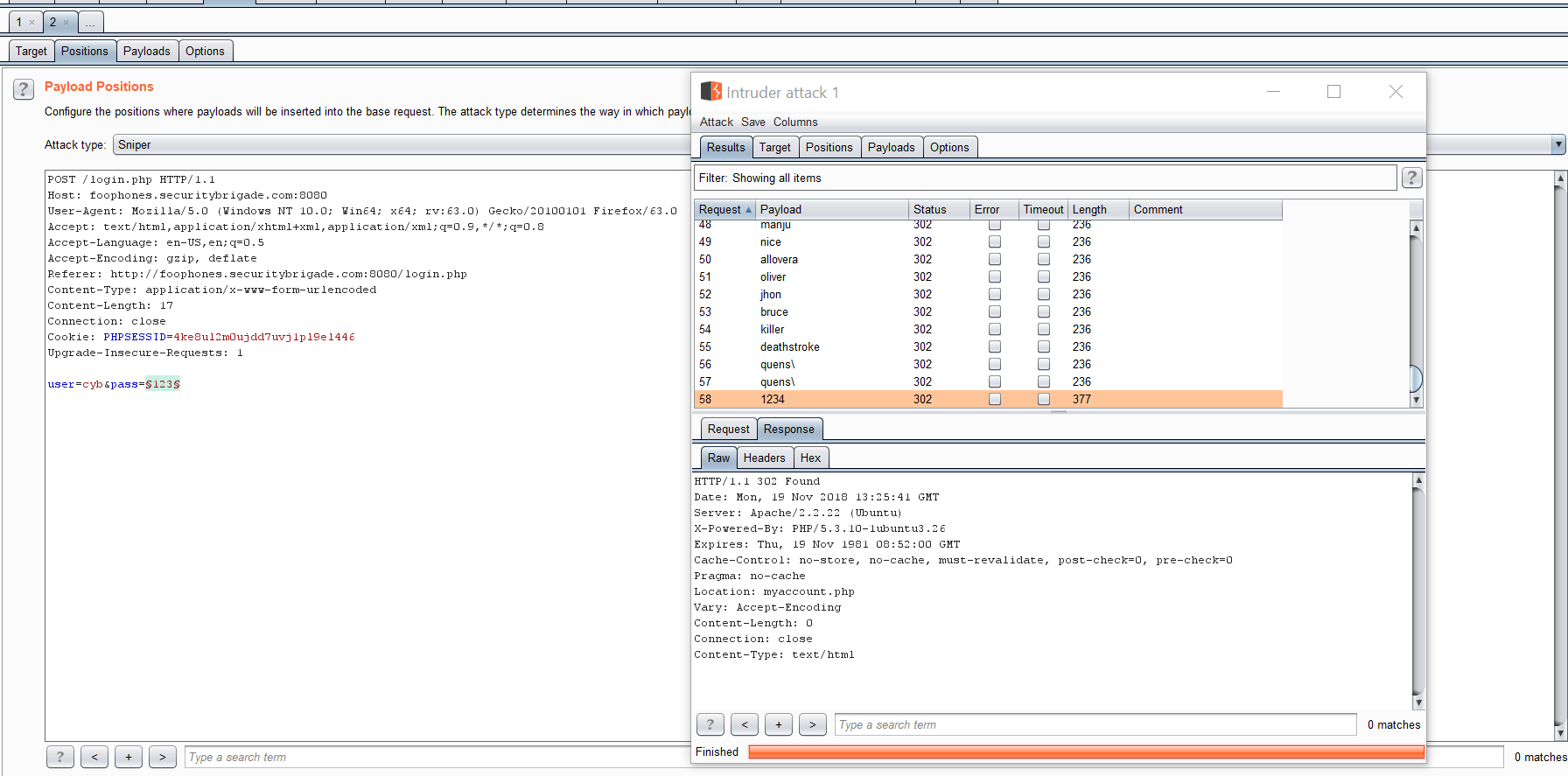
6)In result we can clearly see that inspite of so many failed logins at 58th attempt when correct password is found login attempt is successful.

7)A screenshot named "noratelimit.png" has been attached to prove the concept.

**Remediation-**

1)Account lockout for some period of time after some attempts or

2)Use of captchas.



No rate limit on login

6)ACCESS TO ROBOT.TXT

**Vulnerable url-**

http://foophones.securitybrigade.com:8080/robot.txt

**Description-**

During the reconnaissance stage of a web application testing, the tester usually uses a list of known subdirectories to brute force the server and find hidden resources.Depending on the uptake of certain web technologies, it needs to be refreshed on a regular basis.As you may see, the directive disallow gives an attacker precious knowledge on what may be worth looking at. Additionally, if that is true for one site, it is worth checking for another.

**P.O.C**

1)Just type the url http://foophones.securitybrigade.com:8080/robot.txt

2)You will list of directories that search engines are not allowed to access.

**Remediation-**

The robots.txt file is not itself a security threat, and its correct use can represent good practice for non-security reasons. You should not assume that all web robots will honor the file's instructions. Rather, assume that attackers will pay close attention to any locations identified in the file. Do not rely on robots.txt to provide any kind of protection over unauthorized access.

7)CSRF LOGOUT

**Vulnerable url**-http://foophones.securitybrigade.com:8080/myaccount.php

**Description-**

Critical actions on user account require a csrf token along with the request that is being sent so as to uniquely identify the request .

In this no csrf token is being used on logout form.Thus this issue can be easily exploited by sending the user a exploit html page to theuser as soons as user opens that page he will be logged out of that page.

**Impact-**

1)Attacker can force the victim to execute malicious actions such as making unintentional payment,logout while he is logged in to his account.

**P.O.C-**

1)Generate csrf exploit using the source code of the page.

2)Now just send this page to a victim who is logged in .

3)As soons as he opens the page he will be logged out.

exploit code used-

<html>

<!-- CSRF PoC - generated by Burp Suite Professional -->

<body>

<script>history.pushState('', '', '/')</script>

<form action="http://foophones.securitybrigade.com:8080/logout.php">

<input type="submit" value="Submit request" />

</form>

</body>

</html>

The a screenshot for the csrf request "csrflogout.png" is also attached below.

**Remediation-**

Use of csrf token on critical request by user.

8)XSS PROTECTION HEADER IS NOT DEFINED

**Vulnerable url-** http://foophones.securitybrigade.com:8080/

**Description**-

Some browsers, including Internet Explorer, contain built-in filters designed to protect against cross-site scripting (XSS) attacks. Applications can instruct browsers to disable this filter by setting the following response header:

X-XSS-Protection: 0

This behavior does not in itself constitute a vulnerability; in some cases XSS filters may themselves be leveraged to perform attacks against application users. However, in typical situations XSS filters do provide basic protection for application users against some XSS vulnerabilities in applications. The presence of this header should be reviewed to establish whether it affects the application's security posture.

**P.O.C-**

1)Using nikto web application scanner I got to know about unavailabity of xss-protection header.

2)This can be verified by a piece of javascript code that is present in the source code. <script>alert("123")</script>

3)The piece is executed every time we log out of a account genrating a alert box.

**Remediation-**

Review whether the application needs to disable XSS filters. In most cases you can gain the protection provided by XSS filters without the associated risks by using the following response header:

X-XSS-Protection: 1; mode=block

When this header is set, browsers that detect an XSS attack will simply render a blank page instead of attempting to sanitize the injected script. This behavior is considerably less likely to introduce new security issues.

9)WEB PARAMETER TAMPERING

**Vulnerable url-**http://foophones.securitybrigade.com:8080/buy\_confirm.php

**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.

**Impact-**

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 likes Webscarab and Paros proxy are mostly used.

The attack success depends on integrity and logic validation mechanism errors, and its exploitation can result in other consequences including XSS, SQL Injection, file inclusion, and path disclosure attacks.

**P.O.C-**

1)Set up the burp proxy

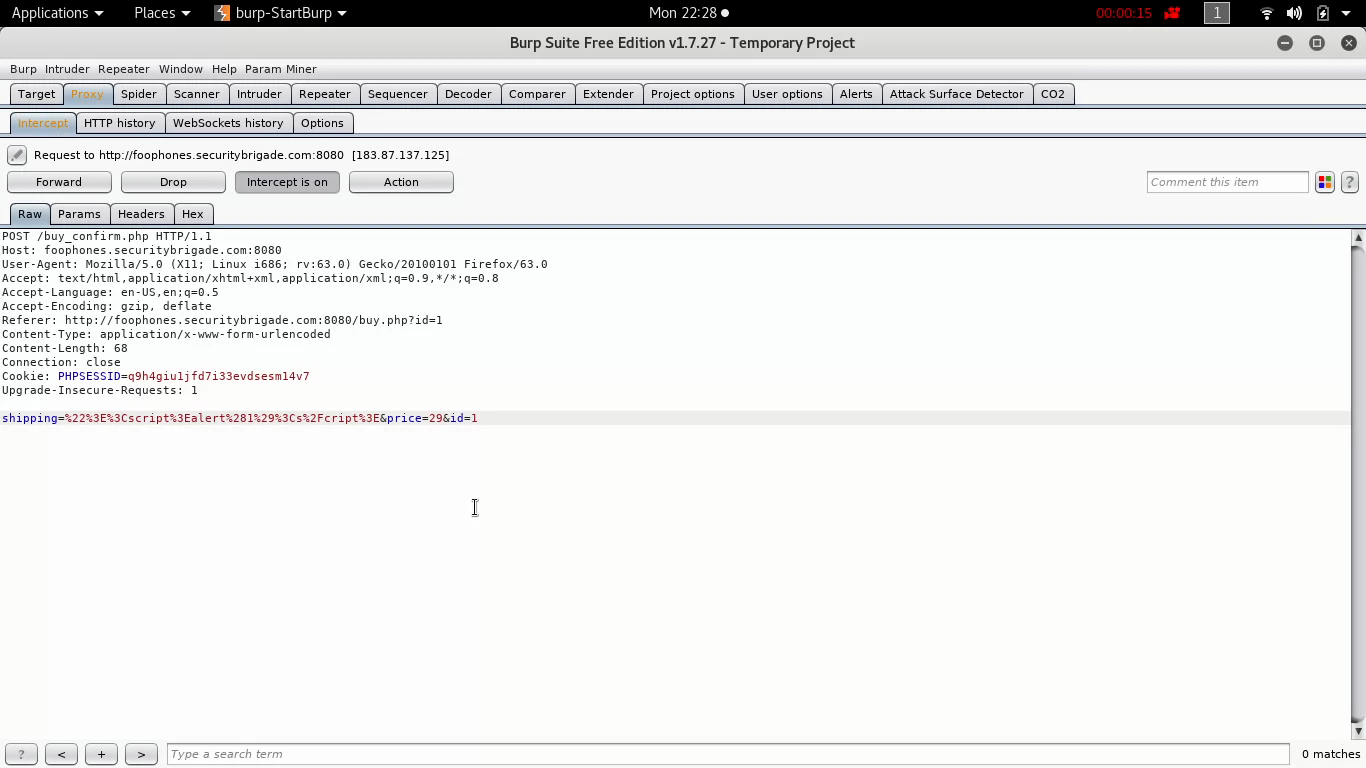
2)Capture the post request which conatins data regarding cost of the product.

3)Now change the cost parameter as per convinience and submit the request.

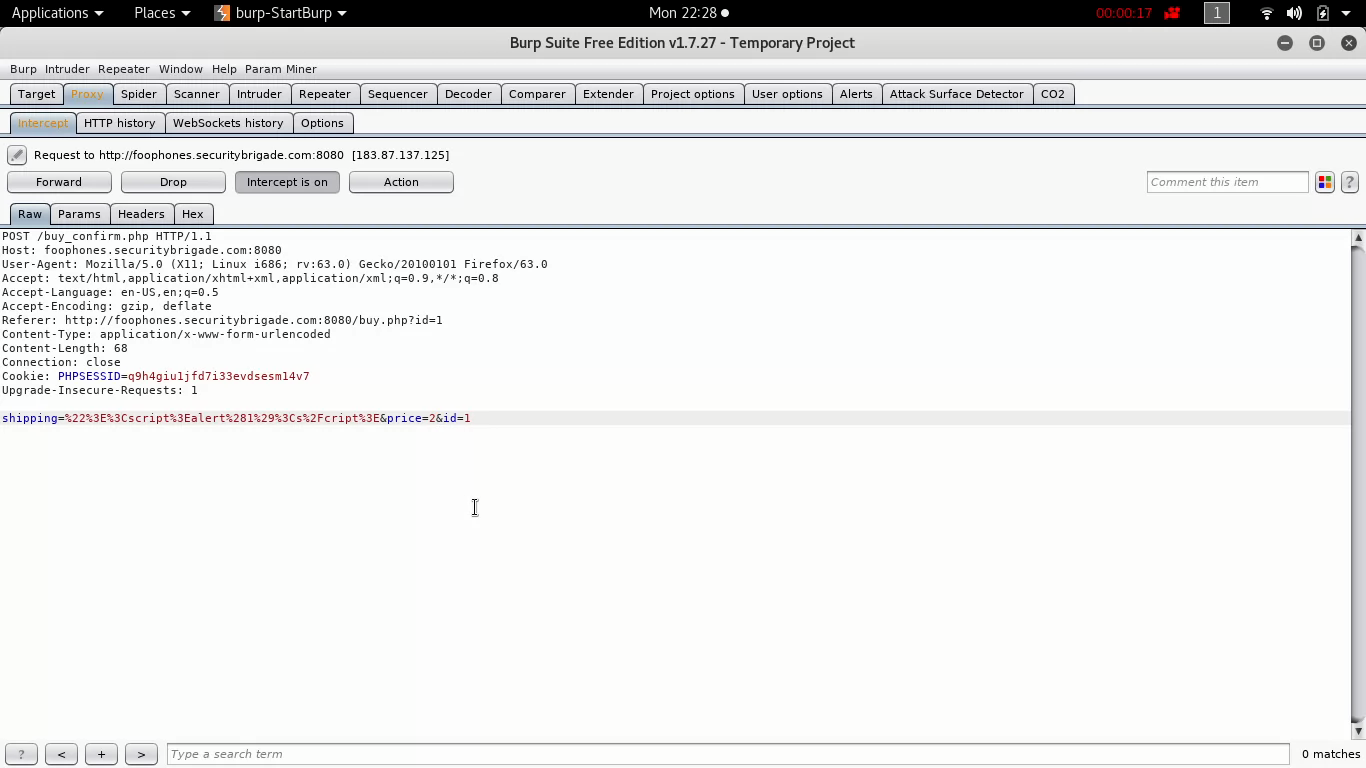
4)You will see that amount detected was the one that was entered after tampering.

5)To demonstrate this issu the video paramertertampering.avg is being attached.

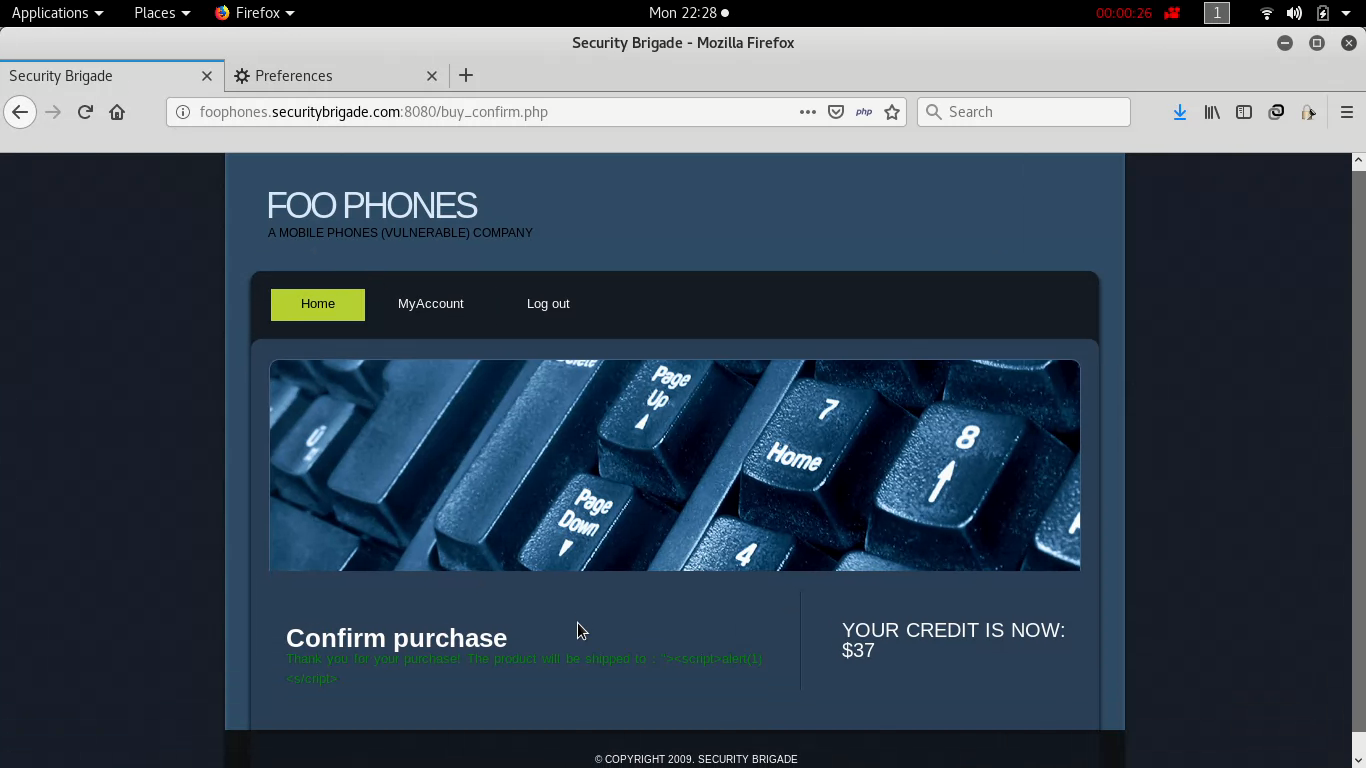
video "parametertampering.avg" has been attached to demontrate this issue.



Original request



Tampered request



Request successful

10)CSRF ON PURCHASE

**Vulnerable url**-http://foophones.securitybrigade.com:8080/buy.php?id=1

**Description-**

Critical actions on user account require a csrf token along with the request that is being sent so as to uniquely identify the request .In this no csrf token is being used on post request to purchase a item.

**Impact-**

Thus this issue can be easily exploited by sending the user a exploit html page to theuser as soons as user opens that page he will be logged out of that page.

**P.O.C**

1)Capture a request to buy using burp and create a csrf exploit.

<html>

<body>

<script>history.pushState('', '', '/')</script>

<form action="http://foophones.securitybrigade.com:8080/buy\_confirm.php" method="POST">

<input type="hidden" name="shipping" value="oli" />

<input type="hidden" name="price" value="29" />

<input type="hidden" name="id" value="1" />

<input type="submit" value="Submit request" />

</form>

</body>

</html>

2)When the user is being logged in , sent this exploit code to the victim.

3)As soons as victim would open this request , purchse will happen automatically and the money will be deducted from his account.

4)For proof of concept a named csrfpurchse.avg has been attached.

**Remediation-**

Use of the csrf token in each and every critical client request.

11) Directory Listing

**Vulnerable url-** http://foophones.securitybrigade.com:8080/images/products

**Description-**

Web servers can be configured to automatically list the contents of directories thst do not have index page present.This can aid the attacker by quickly enabling them to identify the resources within the given path , and proceed directly to analyzing and attacking those resources.

**Impact-**

It particularly increases the exposure of sensitive files within the directory.

**P.O.C**

1)Use dirbusters using a customized wordlist to check for other sensitive directories.

2)From above step we got the path http://foophones.securitybrigade.com:8080/images/.

2)You will get access to content of directory.

**Remediation-**

1)Configure your web server to prevent the directory listing for all path beneath the webroot.

2)Place in each directory a default file that the web server will display instead of running the directory listing.

12)PASSWORD FIELD WITH AUTOCOMPLETE ENABLED

**Description-**

Most browsers have a facility to remember user credentials that are entered into HTML forms. This function can be configured by the user and also by applications that employ user credentials. If the function is enabled, then credentials entered by the user are stored on their local computer and retrieved by the browser on future visits to the same application.

**Impact-**

The stored credentials can be captured by an attacker who gains control over the user's computer. Further, an attacker who finds a separate application vulnerability such as cross-site scripting may be able to exploit this to retrieve a user's browser-stored credentials.

**Remediation-**

Include the attribute autocomplete=off within the FORM tag or within relevant input tags.

13)APACHE SERVER USED IS OUTDATED

**Server name**-Apache/2.2.22 (Ubuntu)

**Description**-

The apache version that is being used over here is completed outdated and marked out of market. Large number of vulnerabilities have been discovere in version 2.2.22 which are enlisted in the form cve issues.

For getting the list of al CVE issues refer to url-www.cvedetails.com/vulnerability-list/vendor\_id-45/product\_id-66/version\_id-47560/Apache-Http-Server-2.2.2.html

**Impact-**

One of the prime issues faced by this server is CVE-2013-1896.According to which mod\_dv.c in the apache HTTP server does not prperly determine whether DAVis enabled for uri,which allows remote attackers to cause a denial of service via a MERGE request in which uri is configured for handling the mod\_dav\_svn module.

**Remediation-**

Update the apache web server to latest version Apache httpd 2.4.37

14)NO VALID SPF RECORD FOUND

**Vulnerable url**-http://foophones.securitybrigade.com:8080

**Description**-

An SPF record is a type of Domain Name Service (DNS) record that identifies which mail servers are permitted to send email on behalf of your domain. The purpose of an SPF record is to prevent spammers from sending messages with sender addresses of your domain.

**Impact**

Missing SPF record allows hackers to send spam emails by using an email address that includes your domain name as its suffix.

**P.O.C**

1)Open the website www.kitterman.com

2)Enter the name of the website for which spf records has to be sent.

3)You will see that no valid spf record are present.

4)Now in order to send fake mail , open www.emkeiz.com

5)Enter the email of victim and email of of company domain like security@foophones.securitybrigade.com

6)You will see that , anyone can easily send mails pretending to be an authorised mail from your website

Screenshot spfrecord.png has been attached.

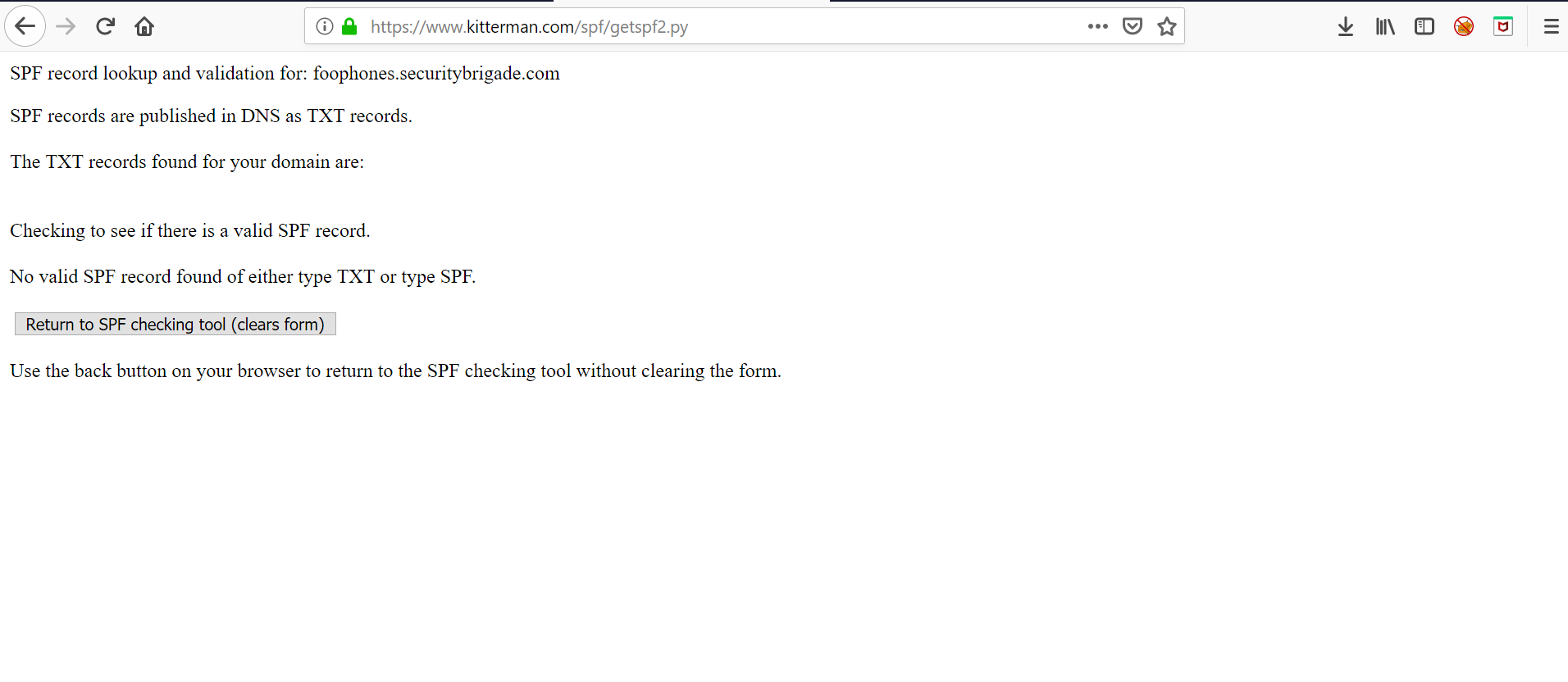
**Remediation**-

If no emails are sent from the domain (this is easily changed if you want to start to send emails in the future), a simple SPF policy that disallows all emails is recommended:

v=spf1 -all

To fully implement your SPF policy, there is only one step left, adding it to the DNS record for the domain. Log in to control the name server. If you don’t

know where that is, the default name server from the domain registrar (such as GoDaddy and NameCheap) is probably used and that is where you should log in to manage the DNS records. A TXT record should now be added with the value of the selected SPF policy. In many cases, the SPF policy needs to be placed within quotes.



No valid spf record found

15)Session fixation

**Vulnerbale url**-foophones.securitybrigade.com/myaccount.php

**Description**-

Session Fixation is an attack that permits an attacker to hijack a valid user session. The attack explores a limitation in the way the web application manages the session ID, more specifically the vulnerable web application. When authenticating a user, it doesn’t assign a new session ID, making it possible to use an existent session ID. The attack consists of obtaining a valid session ID (e.g. by connecting to the application), inducing a user to authenticate himself with that session ID, and then hijacking the user-validated session by the knowledge of the used session ID. The attacker has to provide a legitimate Web application session ID and try to make the victim's browser use it.

**Impact-**

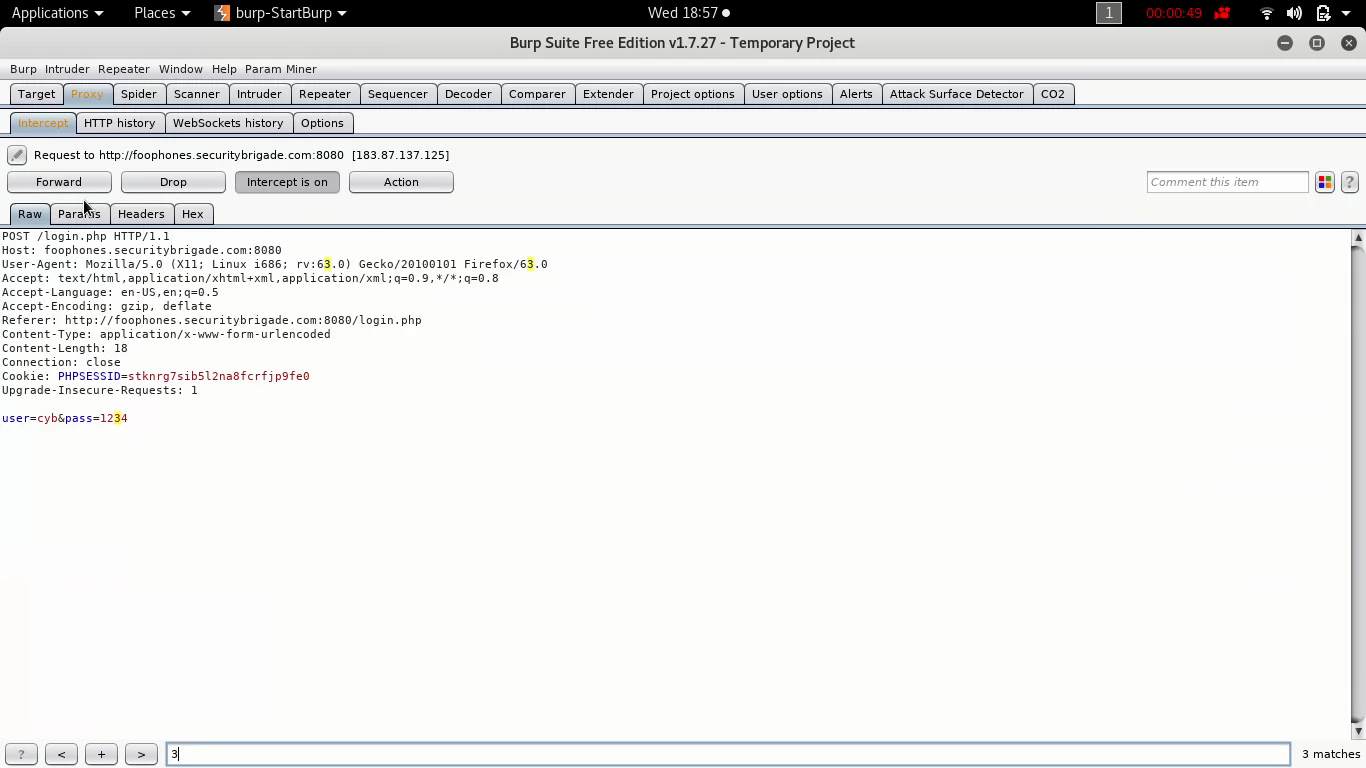
1)The Session Fixation attack fixes an established session on the victim's browser, so the attack starts before the user logs in

**P.O.C**

1)To testify this issue , check the cookie afteruser logs out.

2)Check the cookie when another user logs in.

3)You will see that both the cookies are same.



Request before logging in



Request after logging in

16)REMOTE CODE EXECUTION

**Vulnerable url**- http://foophones.securitybrigade.com:8080/image/avatars/image.php

**Description-**

Code Injection is the general term for attack types which consist of injecting code that is then interpreted/executed by the application. This type of attack exploits poor handling of untrusted data. These types of attacks are usually made possible due to a lack of proper input/output data validation, for example:

1)allowed characters (standard regular expressions classes or custom)

2)data format

3)amount of expected data

**Impact-**

1)Easy execution of a php shell,giving access to all directories and folder.

2)It can be also used to establish a backdoor on server by using php reverse shell.

3)It can lead to execution of malicious javascript that is stored via this upload.

**P.O.C-**

1)Open the registration page,you will see an option of upload image.

2)Now I will upload a php shell that I have stired with extension .png

3)As soon as you submit the registration request intercept it using web browser.

4)Now just change the extension from image.php.png to image.php and submit.

5)Now when you login into your account and access home page , you intercept a request to image directory for file image.php.

6)Now reopen this url seperately on browser .

7)You will see the shell getting executed.

Screenshot "shellupload.png" is being attached to demonstrate this.

**Remediation -**

In most cases, it is enough to remove all potentially dangerous symbols to defend the application. Assume that the "param" parameter is passed via HTTP GET or POST request to the application and is later used in the eval().Before uploading of any content to file to server,it should be subjected to server side security checks.



Shell execution(RCE)

17)DOM BASED XSS

**Vulnerable url**-

http://foophones.securitybrigade.com:8080/add\_user.php?user=admin'<scRipt>alert(9)</script>

**Description-**

DOM Based XSS (or as it is called in some texts, “type-0 XSS”) is an XSS attack wherein the attack payload is executed as a result of modifying the DOM “environment” in the victim’s browser used by the original client side script, so that the client side code runs in an “unexpected” manner. That is, the page itself (the HTTP response that is) does not change,

but the client side code contained in the page executes differently due to the malicious modifications that have occurred in the DOM environment.

**payload used**- {username}'<scRipt>alert(9)</script>

**Impact-**

1)It can be used to redirect te users to malicious site.

2)It can be usd by the attacker to extract user cookies and send it to a site hosted by the victim.

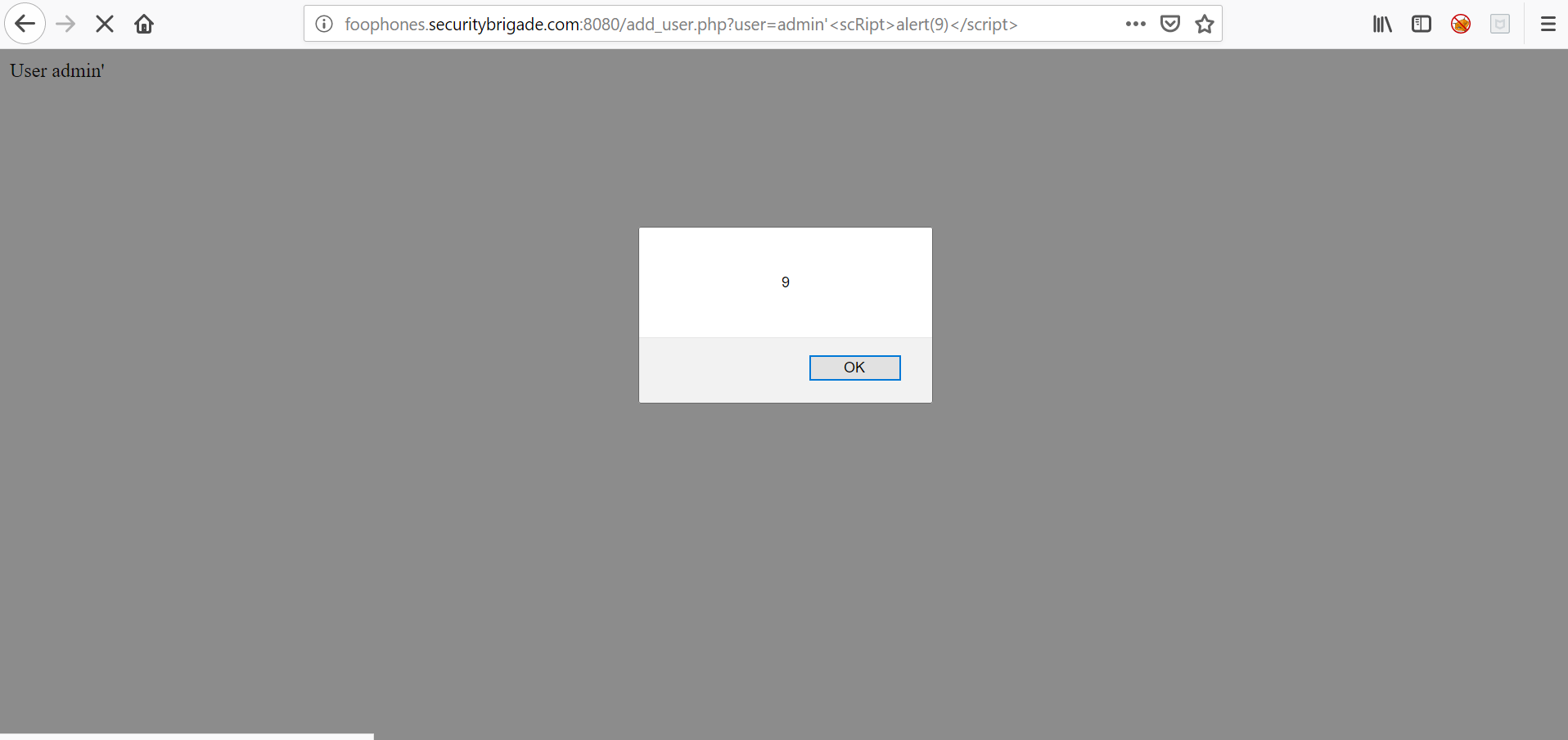
**P.O.C-**

1)Fuzzing the given url http://foophones.securitybrigade.com:8080/ with dirbusterusing custom wordlist in order to find the hidden files and directories.

2)Using this fuzzing I found a path http://foophones.securitybrigade.com:8080/add\_user.php

3)Now brute forcing the parameter with custom list of paylod for xss, generated 200 response for payload shown above.

screenshot for dom based xss "DOMxss.png" has been attached below.



DOM based xss

18)SQL INJECTION BLIND

**Vulnerable url**- http://foophones.securitybrigade.com:8080/add\_user.php?user=test

**Description**-

Blind SQL (Structured Query Language) injection is a type of SQL Injection attack that asks the database true or false questions and determines the answer based on the applications response. This attack is often used when the web application is configured to show generic error messages, but has not mitigated the code that is vulnerable to SQL injection.

**Impact-**

When an attacker exploits SQL injection, sometimes the web application displays error messages from the database complaining that the SQL Query's syntax is incorrect. Blind SQL injection is nearly identical to normal SQL Injection, the only difference being the way the data is retrieved from the database. When the database does not output data to the web page, an attacker is forced to steal data by asking the database a series of true or false questions. This makes exploiting the SQL Injection vulnerability more difficult, but not impossible.

**P.O.C-**

1)Fuzzing the given url http://foophones.securitybrigade.com:8080/ with dirbusterusing custom wordlist in order to find the hidden files and directories.

2)Using this fuzzing I found a path http://foophones.securitybrigade.com:8080/add\_user.php?user=.

3)Using sqlmap execute the following query

"sqlmap.py -u "http://foophones.securitybrigade.com:8080/add\_user.php?user=test" -p "user" --risk="3" --level="3" --user-agent="Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.99 Safari/537.36" --cookie="PHPSESSID=75rahn64p2s13bvhff8it7mfp4" --headers="Host:foophones.securitybrigade.com:8080\nAccept:text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8\nAccept-Encoding:gzip, deflate\nAccept-Language:en-us,en;q=0.5\nCache-Control:no-cache\nReferer:http://foophones.securitybrigade.com:8080/scripts/functions.js" --dbms="MySQL" --batch"

4)Following will be the result that is obtained.

"[23:10:12] [INFO] the back-end DBMS is MySQL

web server operating system: Linux Ubuntu 13.04 or 12.04 or 12.10 (Raring Ringtail or Precise Pangolin or Quantal Quetzal)

web application technology: Apache 2.2.22, PHP 5.3.10

back-end DBMS: MySQL >= 5.0.12

[23:10:12] [INFO] fetched data logged to text files under '/root/.sqlmap/output/foophones.securitybrigade.com'"

**Remediation-**

A robust method for mitigating the threat of SQL injection-based vulnerabilities is to use parameterized queries (prepared statements). Almost all modern languages provide built-in libraries for this.

Wherever possible, do not create dynamic SQL queries or SQL queries with string concatenation.

19)SERVER SIDE SECURITY MISCONFIGURATION(Information Disclosure)

**Vunerable area**- Server configuration

**Description-**

This kind of vulnerability generally results in error messages that disclose the information about the system intead of just giving simple error messages.

**Impact-**

These kind of vulnerability can lead disclosure of -

1)Operating system

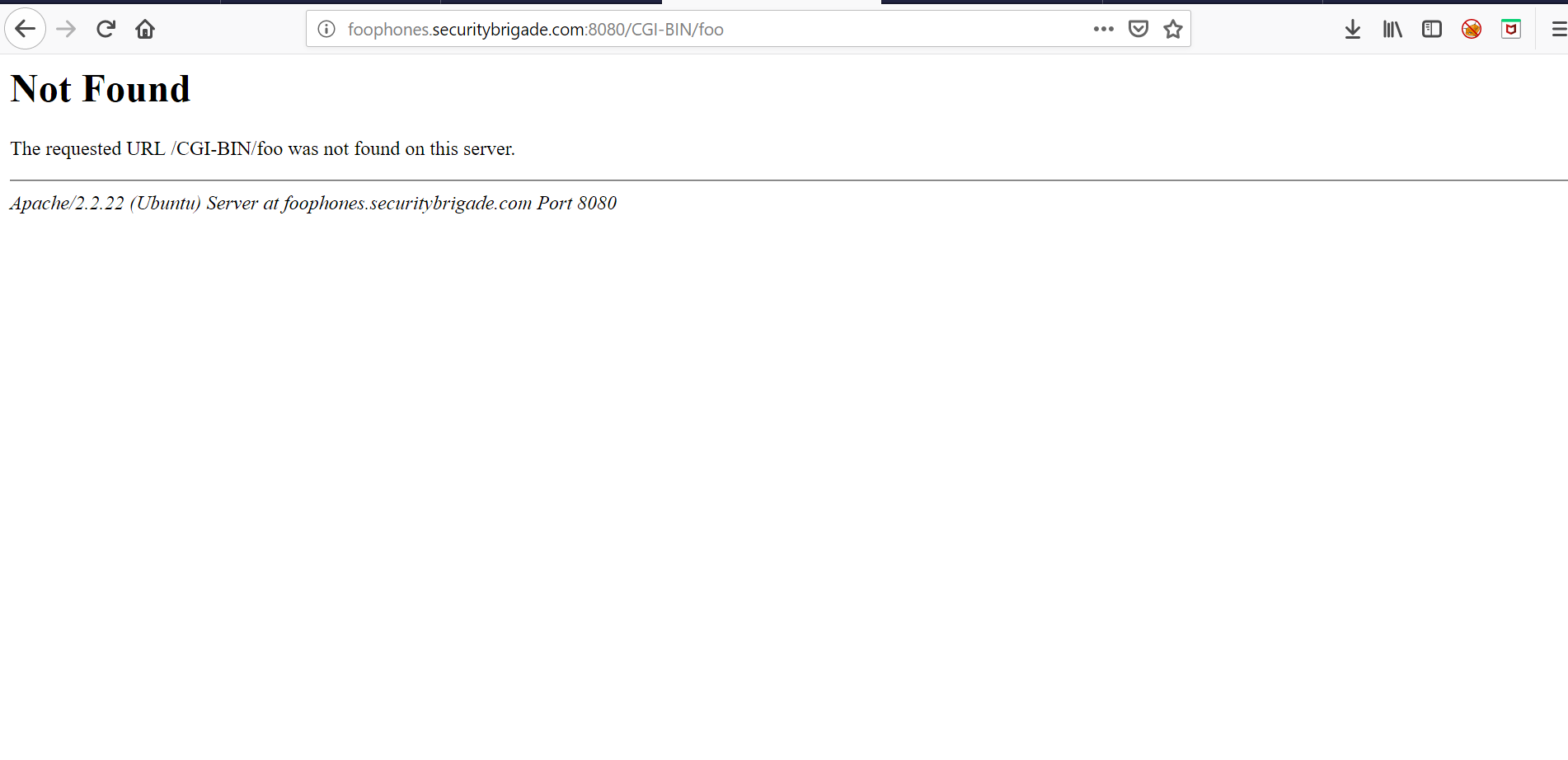
2)Server version

3)SSl version etc

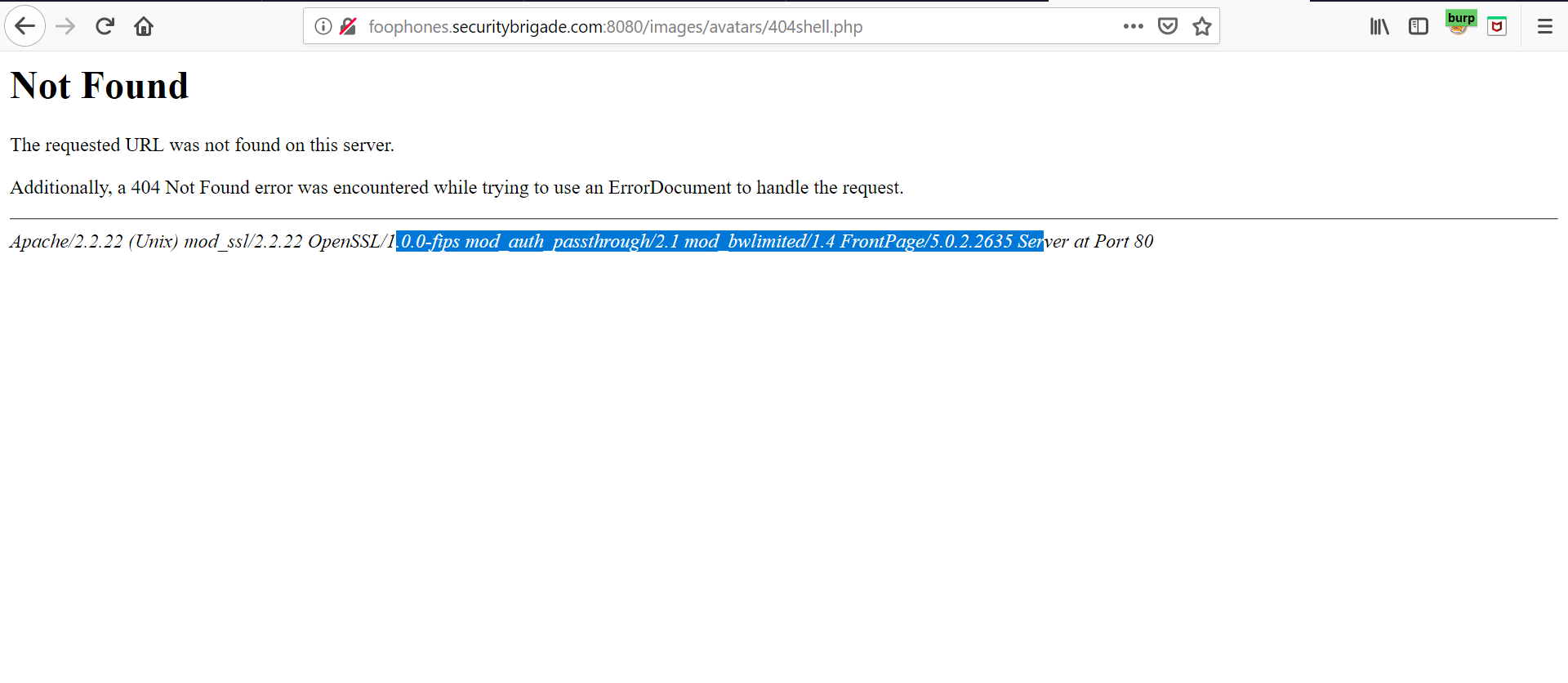
**P.O.C**

1)Enter the url http://foophones.securitybrigade.com:8080/abc

2)It would genrate error mesage giving server name and version ,operating system etc.



Server side information disclosure



Server side information disclosure

.

21)OUT OF DATE VERSION OF PHP

**Description-**

The version of the php identified conatin is 5.3.10.The is an outdated version and encompasses many vulnerabilities .The most critical vulnerability present in this version is phar\_parse\_tarfile integer overflow vulnerability.The vulnerability is in the phar\_parse\_tarfile function due to improper handling of memory operations while the affected system is processing a crafted Phar archive with an empty entry file name.

**Impact-**

An unauthenticated, remote attacker could exploit the vulnerability by convincing a user to open a malicious archive that submits crafted input to the vulnerable function. Processing the file may cause an integer underflow error and could cause the affected software to terminate unexpectedly.

**Remediation-**

1)Install the updates realeased or upgrade to latest version 7.1.7