MY FIRST BOOT2ROOT LEARNING EXPERIENCE

Documentation of thought process and learning notes

15 April 2021

# A. Introduction

My first attempt at a boot2root machine based on recommendations on reddit, and more importantly, a Rick and Morty themed one at that.

This is the documentation of my thought process and hopefully the start of an incremental learning journey into the world of CTFs, and the cybersecurity domain.

# B. Additional Information

Vulnerable machine used: [RICKDICULOUSLYEASY: 1](https://www.vulnhub.com/entry/rickdiculouslyeasy-1,207/)

This is a fedora server vm, created with virtualbox.

It is a very simple Rick and Morty themed boot to root.

There are 130 points worth of flags available (each flag has its points recorded with it), you should also get root.

It's designed to be a beginner ctf, if you're new to pen testing, check it out!

**Walkthroughs used for learning process:**

<https://medium.com/pentestsec/rickdiculouslyeasy-vulnhub-16a54ac2a8e1>

<https://www.hackingarticles.in/hack-rickdiculouslyeasy-vm-ctf-challenge/>

**VirtualBox Network**

Internal Network with 10.10.10.0/24 for network address, and DHCP range of 10.10.10.100 -150.

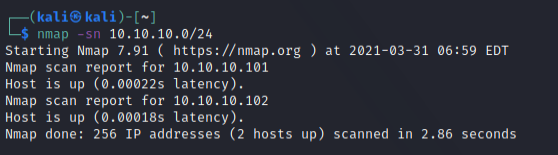
# C. Initial Host Discovery

Already know VM has IP address of 10.10.10.101 from VirtualBox settings

Will still run the **nmap** command for learning the basics of how to use it.

-sn is the option to scan entire network of a network address based on the subnet mask. This is the quickest and easiest way to learn what is on the network in a more reliable way than a simple ping sweep

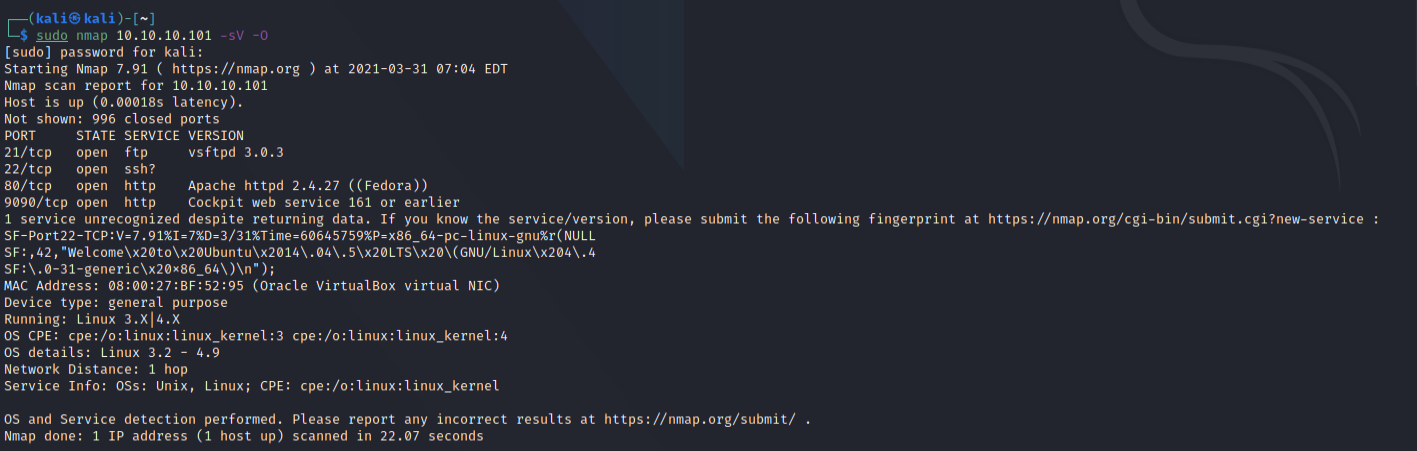
-sn will perform a no port scan



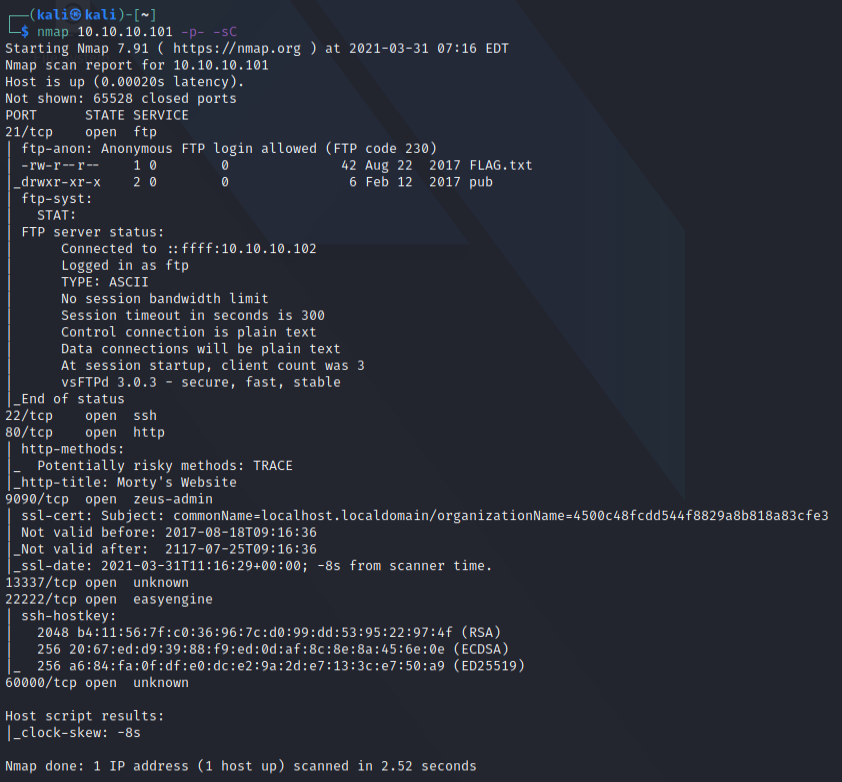
To find out what software, and its version listening on a port, I use sudo nmap 10.10.10.101 -sV -O

-sV is to determine what software and version is listening on the host ports (--version)

-O is to identify the OS of the host operating system



Nmap 10.10.10.101 -p- sC for deeper probe into the host



## Learning Notes

* -p- is used to specify scanning all TCP ports (port range 1-65535)
* -sC (use all scripts) is used to identify specific applications listening on ports, scan for known exploits agains those applications, and scan for common misconfigurations of services
* -T4 (timing templates) could be used to speed up probe. T0 being slowest and T5 being fastest in terms of scan delay before giving up. Since latency is extremely low on VMs, could have used -T4. (note that seldom used on real-world networks with actual latency)
* -sS (stealth scan / TCP SYN). Relatively unobtrusive and stealthy since it never compltetes
* TCP connections. A clear, reliable differentiation between open, closed and filtered states
* -A (all) runs all scripts and -O
* For even deeper scan, could have used options -p [opened ports discovered] -sV -O (or -A)
* For purpose of home lab, could use options: -p- -T4 -sS first before doing -sC?

# D. Scan Results

Nmap scan report for 10.10.10.101

PORT STATE SERVICE

21/tcp open ftp -> allows anonymous FTP login, has FLAG inside

22/tcp open ssh -> SSH, so probably need some sort of credentials?

80/tcp open http -> web server

9090/tcp open zeus-admin -> not sure what this means but SSL, so web server?

13337/tcp open unknown

22222/tcp open easyengine -> not sure what this means, but SSH

60000/tcp open unknown

# E. Starting the Process

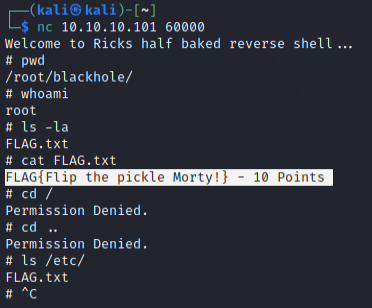
Thought process for sequence of ports to test:

60000 > 21 > 13337 > 9090 >> 80 > SSH ports (22 and 22222)

Based on what I think would be the easiest way to navigate around the scan results.

# F. Port 60000

Since it’s “unknown”, first instinct is to **netcat** the port, which gives welcome message of “Welcome to Ricks half baked reverse shell…” and a shell? (no idea what this really means at the moment)



No idea what I’m supposed to do after getting the flag or if in a real-world situation there’ll be something more I should be exploring, but felt learning objectives were met and moved on first.

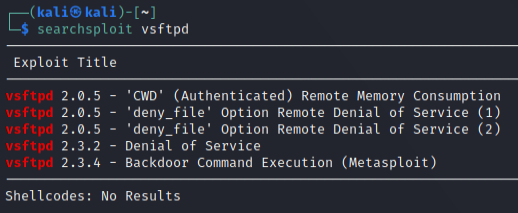
FLAG{Flip the pickle Morty!} – 10/130Points

## Learning Notes:

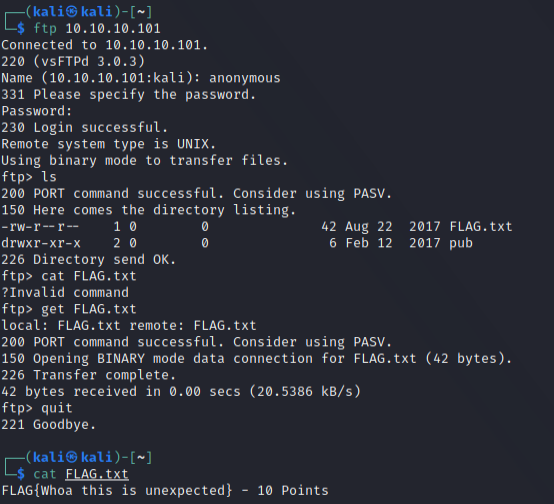
* -nc (netcat) is a network utility that uses TCP and UDP connections to read / write in a network. It can do port scanning, banner grabbing, transferring a file, and generating a reverse connection to name a few
* It acts as a simple TCP/UDP/SCTP/SSL client for interacting with web servers. My current understanding is that it’s useful for banner grabbing for CTF VMs.
* Banner Grabbing is useful for intel-reconnaissance. It gets software banner information and often exposes critical information, and can reveal insecure and vulnerable applications which could lead to service exploitation.

# G. Port 21 (Anonymous login FTP)

Remembered reading something previously about using Metasploit for vsFTPd versions but did not return a result for the version that the VM is running (3.0.3)



Realized after that again, that the service allowed for anonymous logins, and there was already a file called “FLAG” inside.



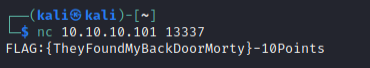
FLAG{Whoa this is unexpected} – 20/130 Points

## Learning notes:

* You can use ls in an FTP session but can’t use cat. Use **get** to download the file locally first before utilizing the cat command.

# H. Port 13337

Seeing as it’s another port being utilized by an unknown service, first instinct is to **netcad** it, similar to port 60000.



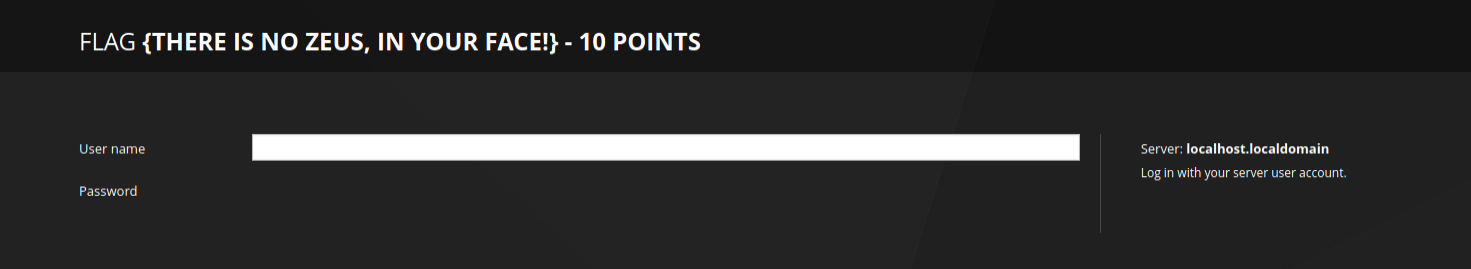
FLAG:{TheyFoundMyBackDoorMorty} - 30/130 Points

## Learning notes:

* **Netcad** will be my friend. Will Google more on its actual real-world / practical usages.

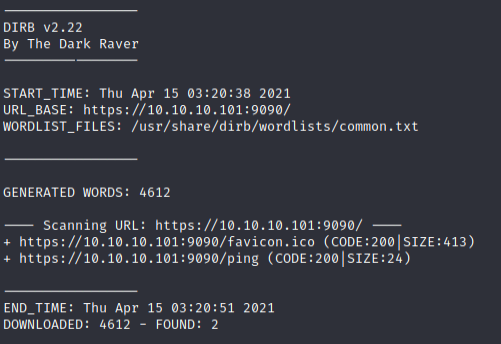
# I. Port 9090 (SSL = Web Server?)

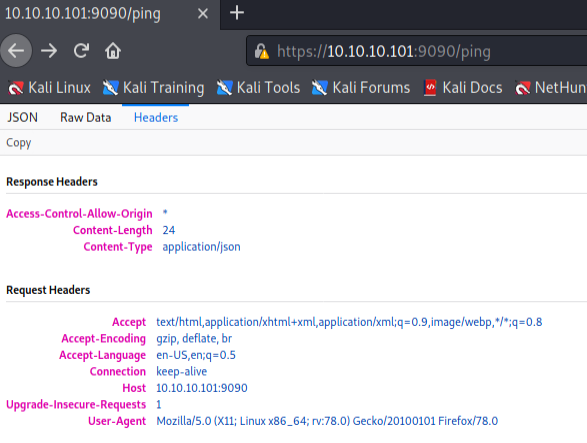
Easy and straight forward enough. Tried looking at the source code to find more additional information, but my current limited knowledge on web development tells me that there is nothing of interest left on this service.



Tried using **dirb** to do some additional recon for this service but didn’t get any promising hits.

~/ping returns a JSON application called cockpit that didn’t seem like I could exploit in anyway at the moment.





Script kiddie in me decided that trying SSLye was worth a shot, but didn’t return any promising hits as well. From some preliminary reading / YouTubing, vaguely recalled it was to scan for the Heartbleed bug that made headlines in 2014.

FLAG {There is no Zeus, in your face!} – 40/130 Points

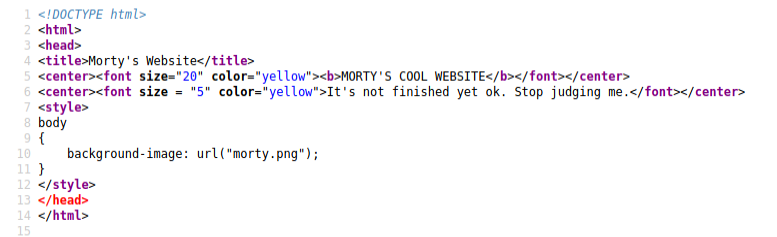
## Learning notes:

* Secure Sockets Layer (SSL) is a standard security technology for establishing an encrypted link between a server and a client—typically a web server (website) and a browser, or a mail server and a mail client (e.g., Outlook).
* Cockpit is an easy-to-use, lightweight and simple yet powerful remote manager for GNU/Linux servers, it’s an interactive server administration user interface that offers a live Linux session via a web browser.
* Heartbleed vulnerability link: <https://www.vox.com/2014/6/19/18076318/heartbleed>

# J. Port 80 (HTTP) – Part 1

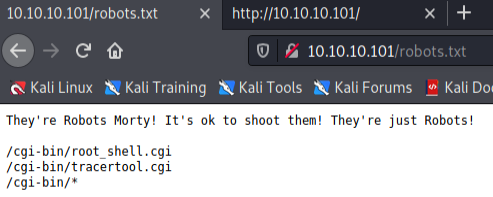
Initially the website and source code did not provide any useful information.



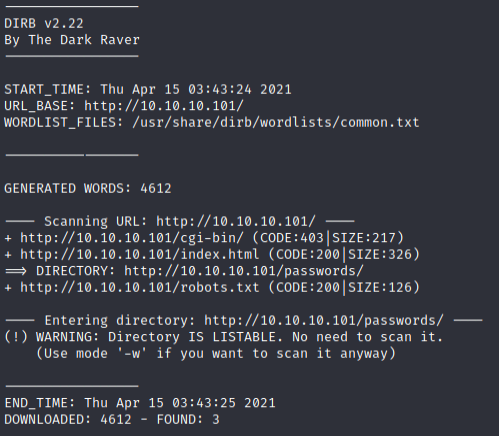


Additional reading and INE’s basic penetration testing course prior taught me that I should always attempt to find for hidden directories (spider), as those almost always yield additional information, especially in purposefully vulnerable machines. (Port 80 = HTTP service).

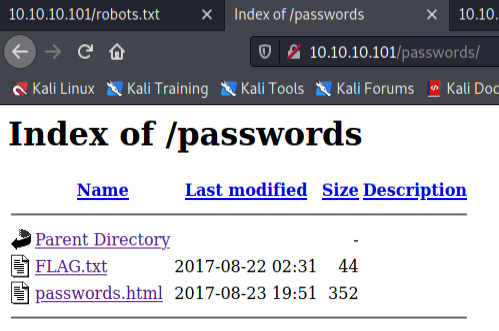
First, I went looking into ~/**robots.txt**, which yielded some directories.



Noted down kept the cgi-bin directory in mind first while performing **dirb** to search for additional hidden directories of interests.

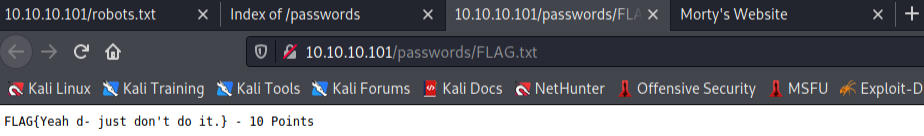


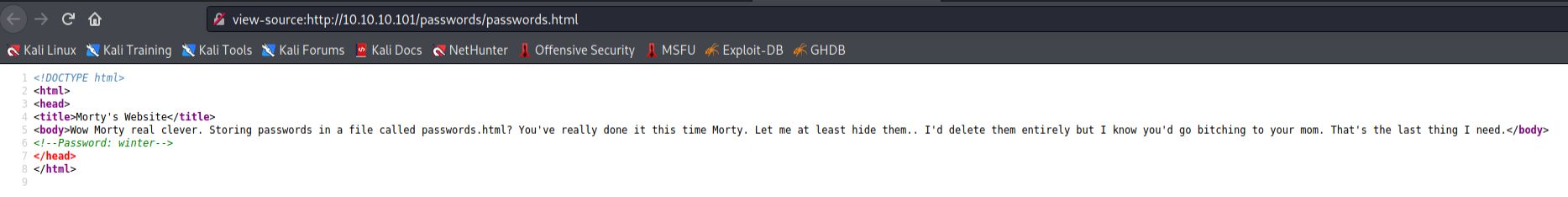
Noted the existence of ~/passwords directory now as well. Exploring it further since it is a dead giveaway.



FLAG.txt yielded another flag, while passwords.html didn’t return anything interesting initially until looking at the source code.

I now have a password called “winter”, but no other credentials to use it with. Noted it down for later.





FLAG{Yeah d- just don't do it.} – 50/130 Points

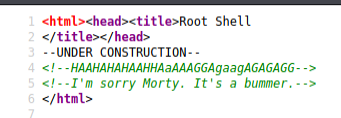
## Learning notes:

* Always look at source code for crucial information that may be purposefully hidden by the creators of vulnerable machines, or accidentally leftover by developers in real-world scenarios.

# K. Port 80 (HTTP) – Part 2

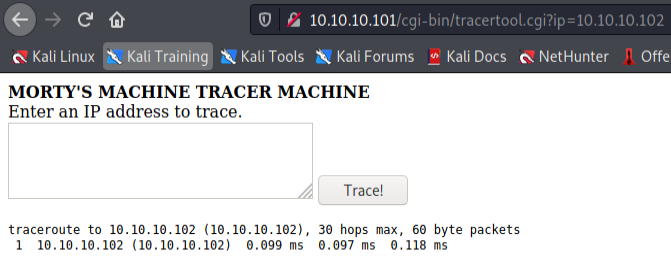
Attempting to figure out what functions the 2 cgi applications could do.

Root\_shell.cgi didn’t yield any promising results.



Tracertool.cgi on the other hand looked like it’s running a traceroute command from the server.

A little bit stumped here as to what else could be done using this app, so took the time to digest the walkthroughs to get a better understanding of what was happening, what additional things I could make happen, and the rationale behind it.



After reading additional explanation extracted from pentestsec’s medium post:

*What happened here? instead of using an IP as input I used a semicolon. Due to the fact that the site is not applying restrictions to the input we managed to run arbitrary commands on the machine.*

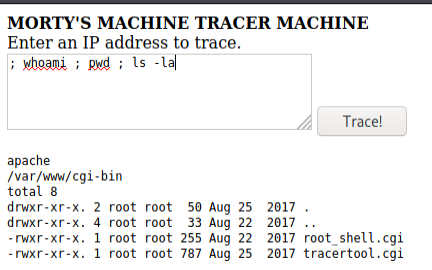
*Instead of the expected input: traceroute 192.168.1.5*

*We used ‘; whoami’ to execute the following command: traceroute ; whoami*

*The semicolon makes the second command to be executed after the first one finishes. Command concatenation can also be applied ussing ‘&&’ and ‘||’*

Able to vaguely understand that inputting the semicolon allowed for command injection or RCE, which makes the server output information as though I was on its terminal.

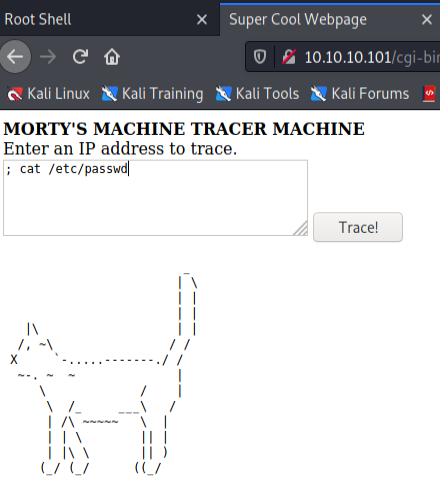
Tried by first using ; whoami ; pwd and ; ls -la to get my bearings of what can / could be done with this.

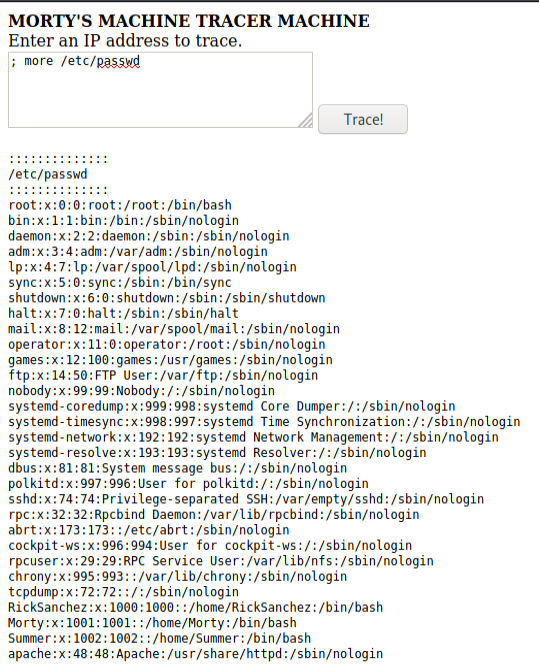


Remembered that users on a Linux system as stored in the /etc/passwd directory. I have a password (winter) from previously, now I need other credentials to attempt to move forward.

**Cat** command seems to have been replaced with an output of a literal cat ascii art, had to use **more** command to get the output I was seeking.

Now have 3 users: RickSanchez, Morty, and Summer.



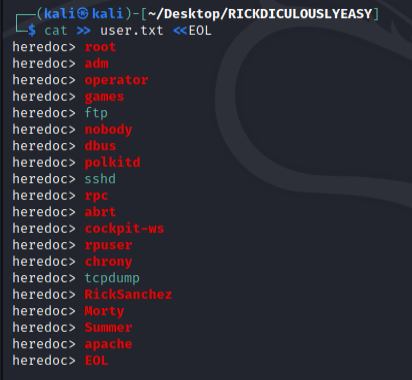


## Learning notes:

* CGI (Common Gateway Interface) is a standard way of running programs from a Web server. Often, CGI programs are used to generate pages dynamically or to perform some other action when someone fills out an HTML form and clicks the submit button.
* Remote code execution (RCE) is a class of software security flaws/vulnerabilities. RCE vulnerabilities will allow a malicious actor to execute any code of their choice on a remote machine over LAN, WAN, or internet.
* Every user on a Linux system, whether created as an account for a real human being or associated with a particular service or system function, is stored in a file called "/etc/passwd".
* The "/etc/passwd" file contains information about the users on the system. Each line describes a distinct user.
* **more** command lets you view text files or other output in a scrollable manner. It displays the text one screenful at a time, and lets you scroll backwards and forwards through the text, and even lets you search the text.

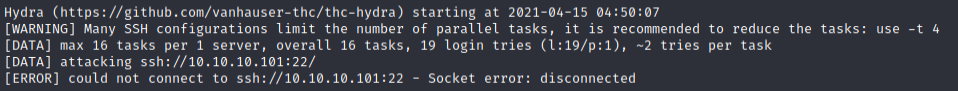
# L. Ports 22 and 22222 (SSH) – Part 1

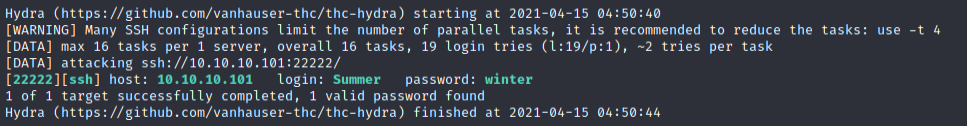
I have a list of users and 1 password. I create a user.txt file as a list for **hydra** to match with password “winter”.



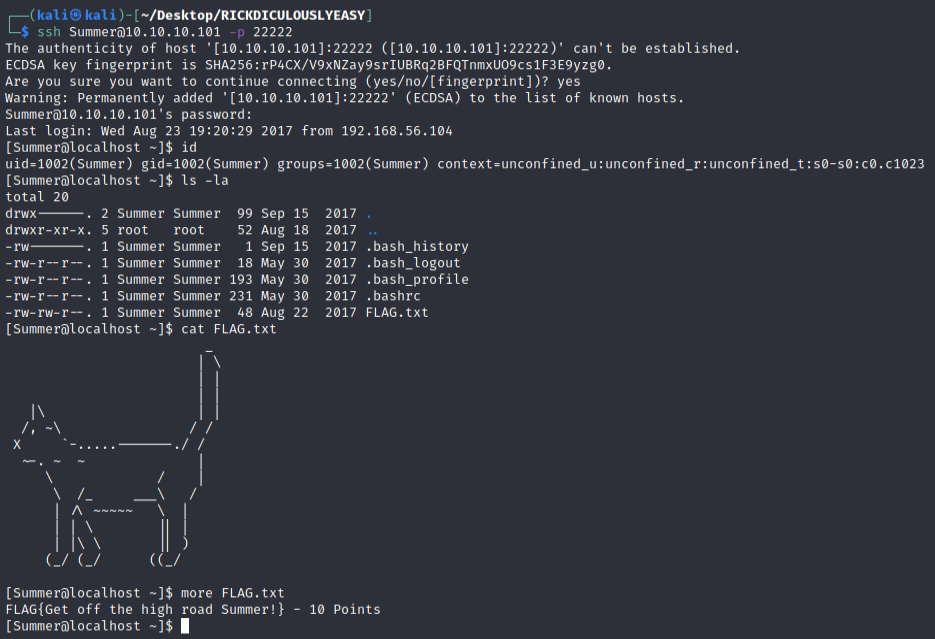
Used **hydra-wizard**. The step-by-step input feels beneficial to my current level of understanding when using tools.

Port 22 doesn’t seem to work, while 22222 matched user “Summer” with password “winter”. (which was to be expected)





With matched credentials and successful login, begin to start exploring the server’s internals and found another FLAG.txt file.



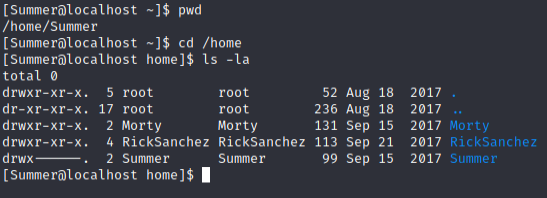
FLAG{Get off the high road Summer!} – 60/130 Points

## Learning notes:

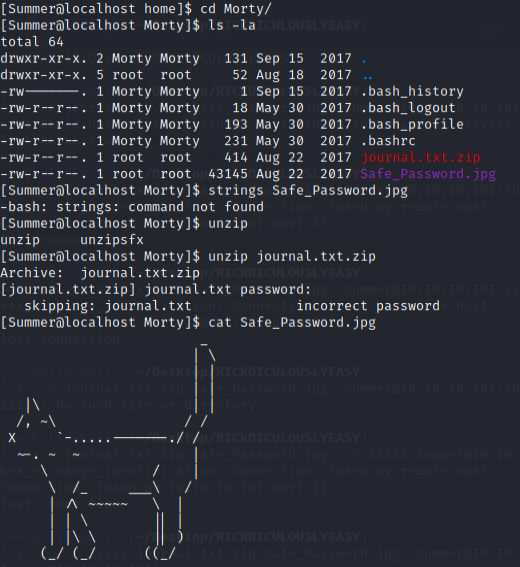
* Blocks of multiple lines of text can be written in a single command when using the <<EOL command (heredoc).
* There’s a somewhat obscure feature in Linux and Unix shells that allows you to open a sort of do-while loop for the cat command. It’s called the heredoc, and it enables you to have, more or less, a text editor no matter what shell you’re using.
* Using cut -d : -f 1 /etc/passwd allows to only get the users instead of the other information presented. -d represent the delimiter, while -f represents the position of the delimiter to stop at.
* Metasploit module “auxiliary/scanner/ssh/ssh\_login” can be used to brute force and open a shell as well. (More info at [here](https://www.offensive-security.com/metasploit-unleashed/scanner-ssh-auxiliary-modules/) and [here](https://null-byte.wonderhowto.com/how-to/get-root-with-metasploits-local-exploit-suggester-0199463/))

# M. Ports 22 and 22222 (SSH) – Part 2

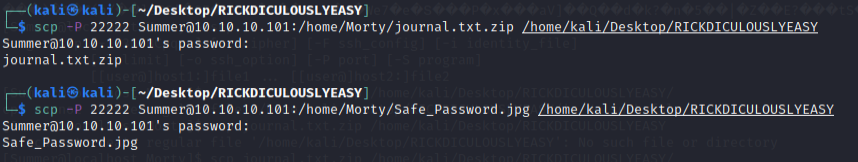
Further digging showed that Summer has access to Rick and Morty’s respective home directories.



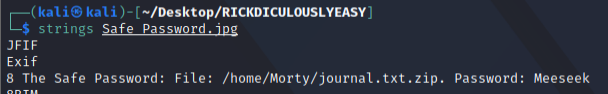
Within Morty directory, there’s a password protected .zip file and what appears to be a .jpg file with the needed password.



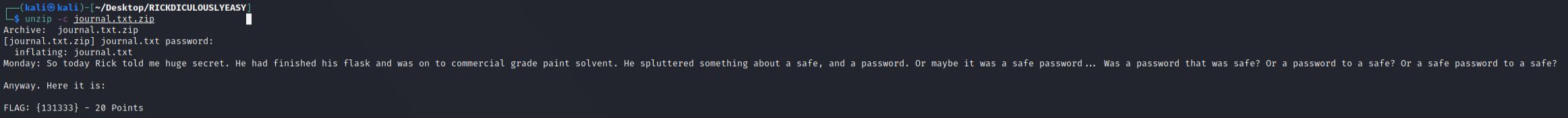
Downloaded the 2 files using **scp** to Kali machine to further investigate them.



.jpg file appears to be just a normal picture file. Further reading of the walkthroughs tells me that there are methods of inspecting the files, which yield the password required for the zip file “Meeseek” (using **head** or **strings** command).



Output of the file also states that the flag is a possible password for another objective. Have noted “131333” as something relevant for later use.



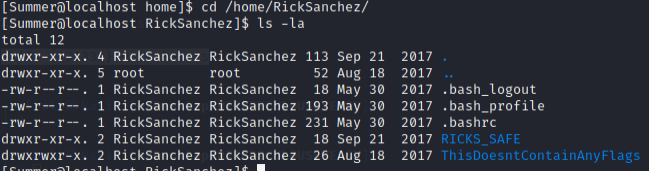
FLAG: {131333} – 80/130 Points

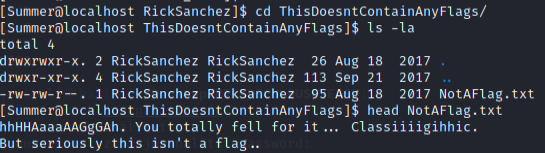
## Learning notes:

* SSH or Secure Shell is a protocol that allows a secure way to access remote computer. SSH implementation comes with scp utility for remote file transfer that utilises SSH protocol.
* The easiest of these are scp or secure copy. While cp is for copying local files, scp is for remote file transfer where both uses almost the same syntax. The main difference is that with scp you'll have to specify the remote host's DNS name or IP address and provide login credential for the command to work.
* Linux strings command is used to return the string characters into files. It primarily focuses on determining the contents of and extracting text from the binary files (non-text file).

# N. Ports 22 and 22222 (SSH) – Part 3

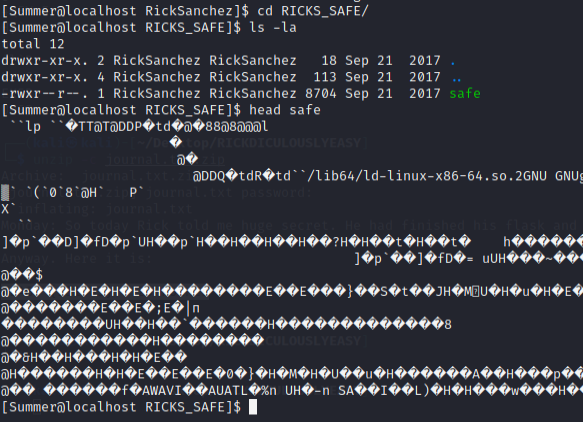
The directory that says it doesn’t contain any flags, surprisingly did not contain any flags.

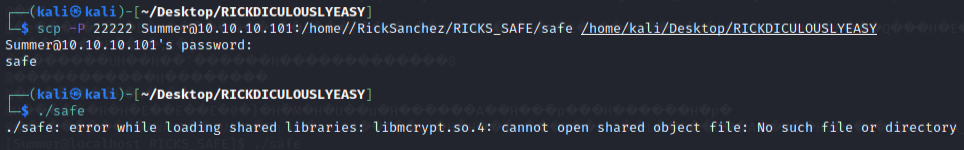




Dir RICKS\_SAFE on the other hand, seems to have a file and looks like an executable that we do not have permissions to run (No permission to e**x**ecute other than the owner).

Tried downloading it to the Kali machine using the scp command but resulted in an error instead. I chalk it up as the Kali machine just missing something.



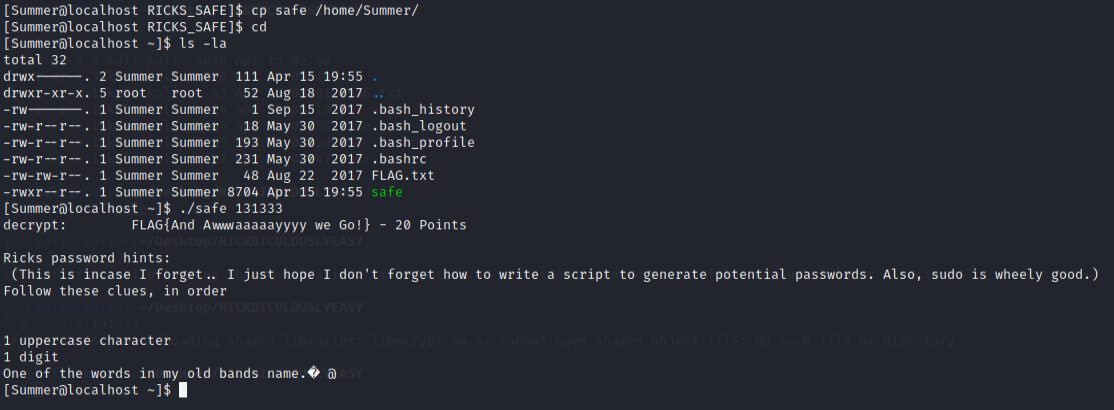


Consulted the walkthroughs to figure out what I was doing wrong instead.

From pentestsec’s medium write-up:

*Here we can see the file ‘safe’ is an executable but the user that we are currently using has only read privileges for it. The sticky bit is not set for that file so we are able to make a copy and execute it (because we are the owner of the copy). We use the password found in Morty’s journal as a parameter for the executable.*

Followed the steps and got another flag, followed by a clue for another password. Gut feeling for the line “sudo is wheely good” tells me that it is something I should note down at least.



FLAG{And Awwwaaaaayyyy we Go!} – 100/130 Points

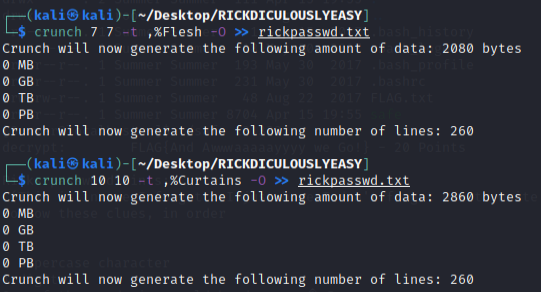
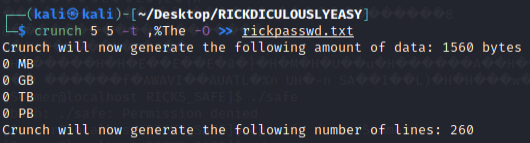
## Learning notes:

* Example: The permission in the command line is displayed as: \_rwxrwxrwx 1 owner:group
* The first character that I marked with an underscore is the special permission flag that can vary.
* The following set of three characters (rwx) is for the owner permissions.
* The second set of three characters (rwx) is for the Group permissions.
* The third set of three characters (rwx) is for the All Users permissions.
* Following that grouping since the integer/number displays the number of hardlinks to the file.
* The last piece is the Owner and Group assignment formatted as Owner:Group.

# O. Ports 22 and 22222 (SSH) – Part 4

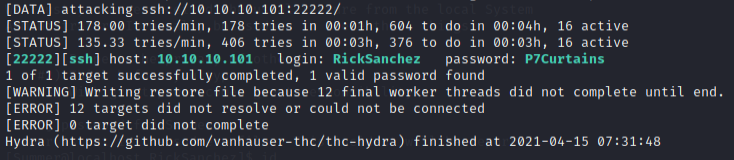
The previous clues indicated the password would be (Upper Case)(Digit)The/Flesh/Curtains.

Using hackingarticles’ walkthrough as a guideline for understanding, used **crunch** command to generate the password list to brute force for user RickSanchez with **hydra** again.



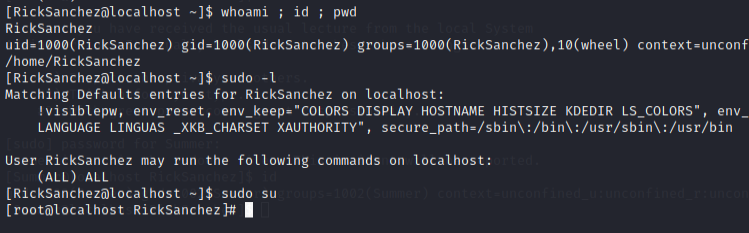


Have identified a valid login credential.

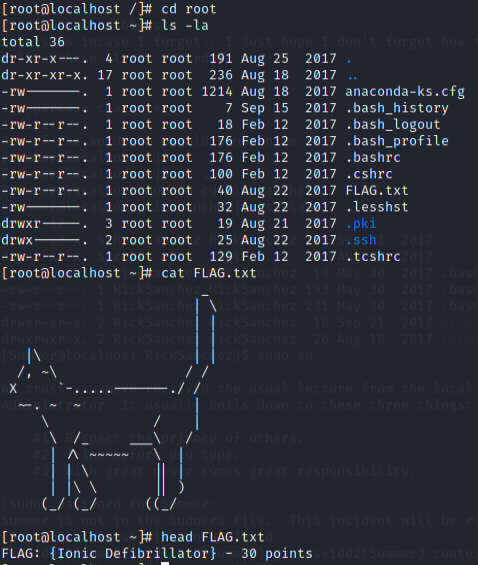


Started digging for what privileges the account had by using the id command, and was reminded of the note about “sudo is wheely good”.

Used sudo -l to list the user’s privileges and saw “ALL”, indicating that I could escalate to root privileges.



And with that, finally obtained the last flag for a full 130 points, and some PTSD about cat ascii art.



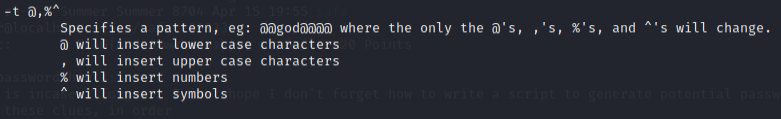
FLAG: {Ionic Defibrillator} – 130/130 points

## Learning notes:

* Crunch is a wordlist generator where you can specify a standard character set or a character set you specify. crunch can generate all possible combinations and permutations.







* The most easiest way to count the number of lines, words, and characters in text file is to use the Linux command “wc” in terminal.
* In Unix operating systems, the term wheel refers to a user account with a wheel bit, a system setting that provides additional special system privileges that empower a user to execute restricted commands that ordinary user accounts cannot access.
* The sudo -l list user's privileges or check a specific command. Using -l twice outputs a longer format.

# P. Conclusion

As my first CTF style machine, I found the pace to be just nice in grasping the basics for boot2root. There are still many commands and logics that I would like to commit to the common-sense portion of my brain. The entire process and documentation took about 6 hours in total.

I had read both the walkthroughs about 2 – 3 times each prior to attempting this machine to get a basic understanding of what to expect. The walkthroughs have been immensely helpful in explaining some basic concepts and I’d like to thank the authors for taking the time and effort in publishing them.

Moving forward, I think I should reenforce my Linux fundamentals, as well as read up more on how to use nmap effectively according to my needs.