

Module 2

Linux For Hackers

History of LINUX

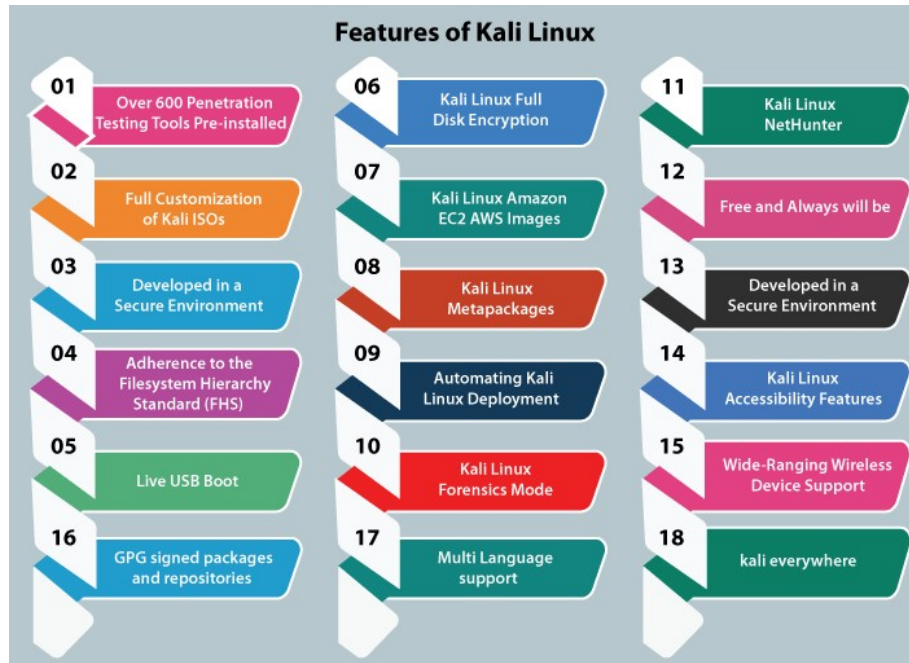
- In 1969, Ken Thompson & Dennis Ritchie of Bell Laboratories created a UNIX OS. It was written in assembly language & later translated into C to make it portable & eventually it became a widely used OS.
- After UNIX, some researchers created UNIX-like systems such as BSD, MINX etc.. But they were lacking one important thing - kernel
- The kernel is a computer program at the core of the computer's OS with complete control over everything in the system. It is an integral part of OS. Kernel is a layer between hardware and software.
- The Linux OS was developed by Linus Torvalds in 1991, which sprouted as an idea to improve the UNIX OS. He suggested improvements but was rejected by UNIX designers. Therefore, he thought of launching an OS, designed in a way that could be modified by its users.
- Nowadays, Linux is the fastest-growing OS. It is used from phones to supercomputers by almost all major hardware devices.
- On top of Linux kernel developers around the globe have created a lot of other distributions like Debian, redhat, mint, android etc.

KALI LINUX

- Kali Linux is a Debian-based Linux distribution that is designed for digital forensics and penetration testing.
- It is funded and maintained by Offensive Security, an information training company.
- Kali Linux was developed through the rewrite of BackTrack by Mati Aharoni and Devon Kearns of Offensive Security.
- Kali Linux comes with a large number of tools that are well suited to a variety of information security tasks, including penetration testing, computer forensics, security research, and reverse engineering.
- **BackTrack** was their previous information security operating system. Kali Linux's first version, **Kali 1.0.0**, was released in **March 2013**.
- Kali Linux is now funded and supported by **Offensive Security**. Today, if we went to Kali's website (www.kali.org), we'd notice a giant banner that states, "**Our Most Advanced Penetration Testing Distribution, Ever.**"
- A very bold statement that ironically has yet to be disproven. There are over 600 **penetration-testing applications** preconfigured on Kali Linux for us to explore. Each program has its own set of capabilities and applications.

Kali Linux performs a fantastic job of categorizing these important tools into the following groups:

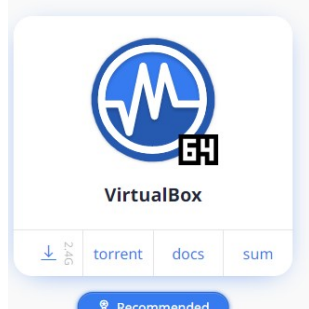
1. Information Gathering
2. Vulnerability Analysis
3. Wireless Attacks
4. Web Application
5. Exploitation Tools
6. Stress Testing
7. Forensics Tools
8. Sniffing & Spoofing
9. Password Attacks
10. Maintaining Access
11. Reverse Engineering
12. Reporting Tools
13. Hardware Hacking



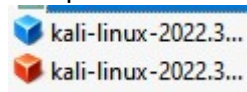
KALI LINUX INSTALLATION IN VIRTUAL BOX

Download and install the Virtual Box : <https://www.virtualbox.org/>

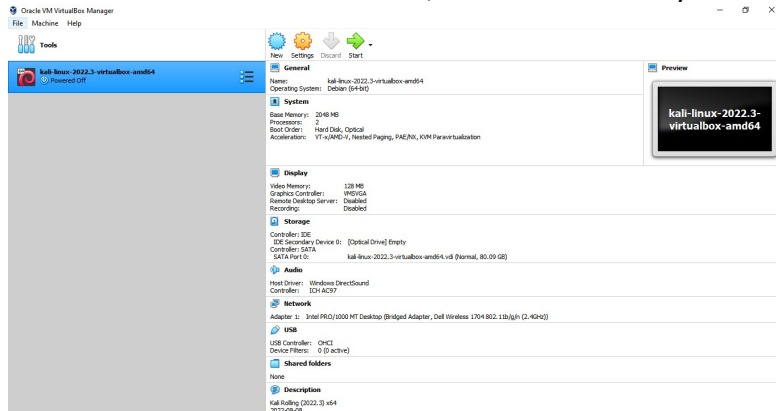
Download Kali for virtualbox: <https://www.kali.org/>



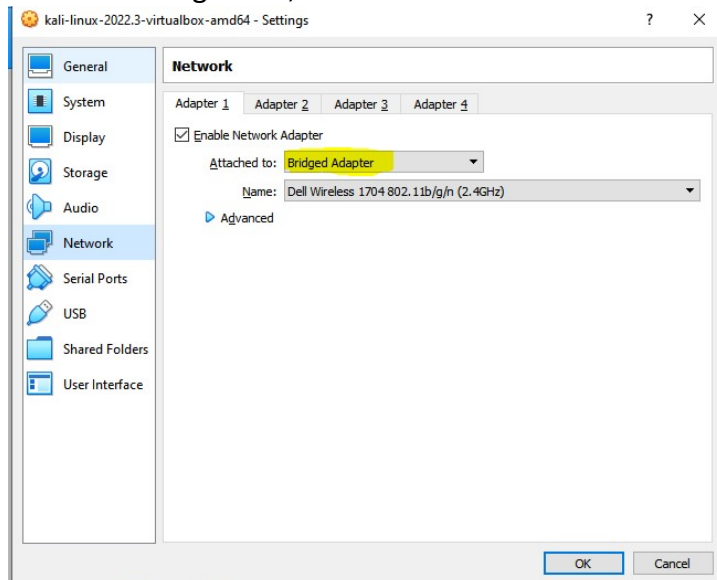
Unzip the downloaded Kali file using WINRAR.



Double click the blue icon of kali,it will automatically installed in virtual box.



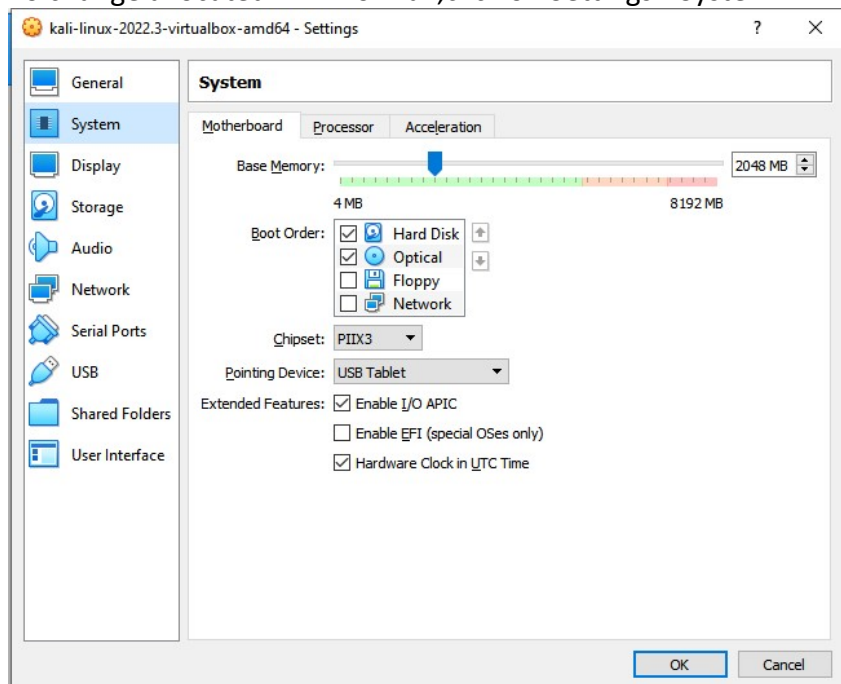
Click on Settings icon ,and click Network -> choose bridged adapter



Click OK.

Choose Kali Linx->Click start to open kali linux.

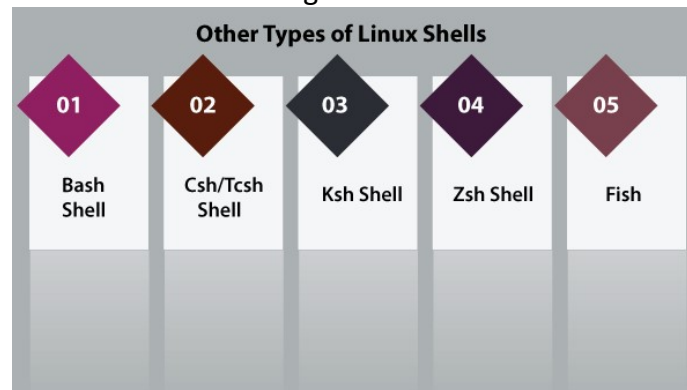
To change allocated RAM for Kali,click on Settings->System



SHELL

- The shell is basically a program that takes your commands from the keyboard and sends them to the OS to perform.
- The shell can be defined as a command interpreter within an operating system like Linux/GNU or Unix. It is a program that runs other programs. The shell facilitates every user of the computer as an interface to the Unix/GNU Linux system. Hence, the user can execute different tools/utilities or commands with a few input data.
- The shell sends the result to the user over the screen when it has completed running a program which is the common output device. That's why it is known as "**command interpreter**".

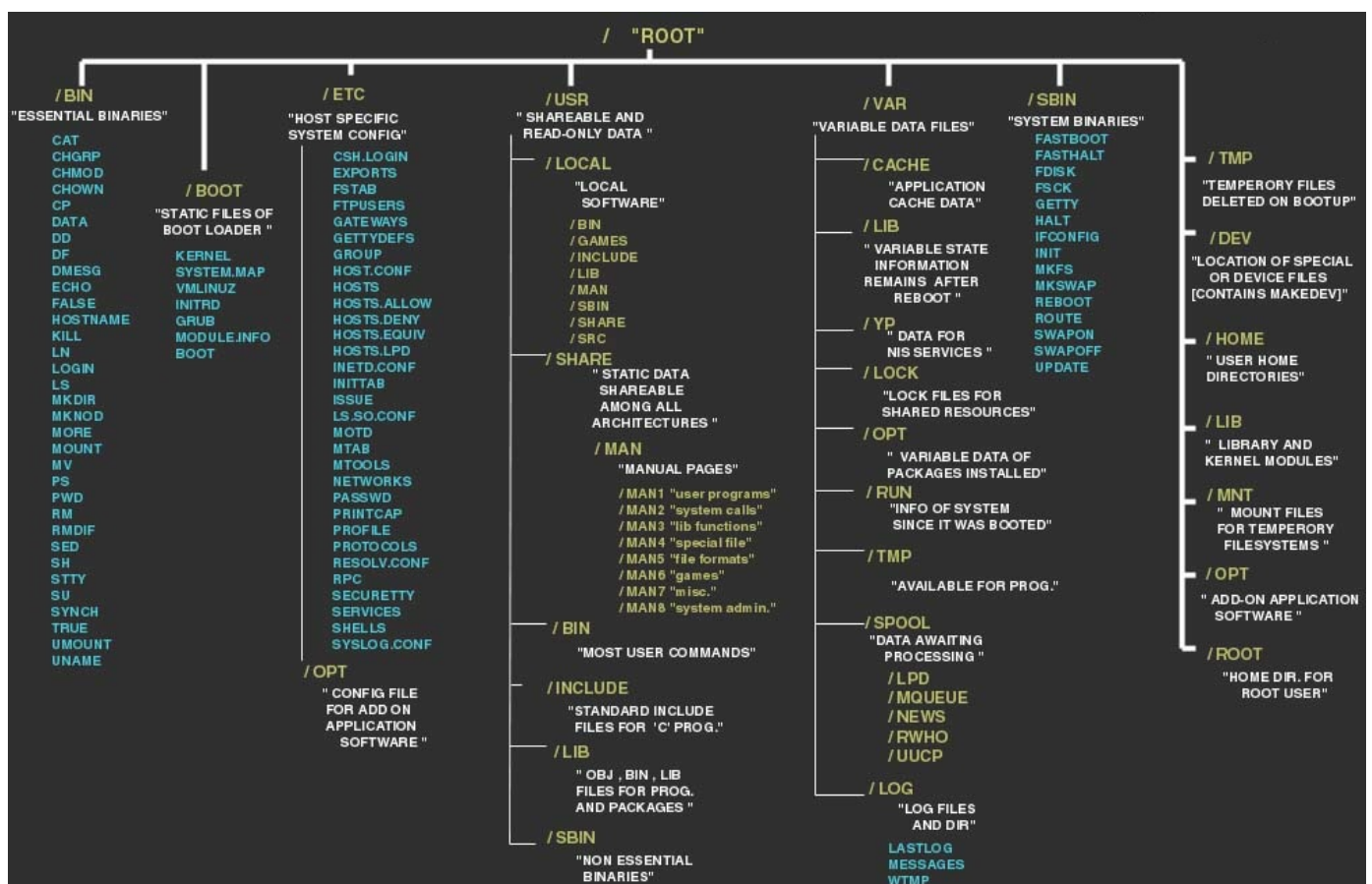
- The shell is not just a command interpreter. Also, the shell is a programming language with complete constructs of a programming language such as **functions, variables, loops, conditional execution**, and many others.
- For this reason, GNU/Unix Linux Shell is stronger than the Windows shell.



Bash Shell

In the bash shell, bash means Bourne Again Shell. It is a default shell over several distributions of Linux today. It is a sh-compatible shell. It could be installed over Windows OS.

File System in LINUX



- Root(/) - The root directory of the entire file system hierarchy, everything is nested under this directory.
- /bin - Essential to run program(binaries), includes the most basic commands such as ls and cp.
- /boot - contains kernel boot loader files.
- /dev - device files
- /etc - core system configuration directories, should hold only configuration files & not any binaries.
- /home - personal directories for users, holds documents files etc etc.
- /lib - hold library files that binaries can use.

- /media - used as an attachment point for removable medias like USB.
- /mnt - temporarily mounted file system
- /opt - optical application system packages
- /proc - have information about currently running process
- /root - the root user's directory(admin)
- /run - information about the running system since the last boot
- /sbin - contain essential system binaries,usually can only be run by root
- /tmp - storage for temporary files
- /usr - it didn't contains user files.it include user installed software's files & utilities
- /var - variable directory,it's used for system logging,user tracking caches etc. Files subjected to change at the time is stored here.

Commands in Kali Linux

- To print message :

```
(kali㉿kali)-[~]
$ echo hello world
hello world
```

- To print current working directory :

```
(kali㉿kali)-[~]
$ pwd
/home/kali
```

- To change directory :

```
(kali㉿kali)-[~]
$ cd Desktop

(kali㉿kali)-[~/Desktop]
$
```

- To view all files and directory :

```
(kali㉿kali)-[~]
$ ls
Desktop  Documents  Downloads
```

- To go to the previous directory :

```
(kali㉿kali)-[~/Desktop]
$ cd ..

(kali㉿kali)-[~]
$
```

- To clear the terminal:

```
(kali㉿kali)-[~]
$ clear
```


- To create a new directory:

```
(kali㉿kali)-[~/Desktop]
$ mkdir demodir

(kali㉿kali)-[~/Desktop]
$ ls
beef  demodir
```

- To create a new file:

```
(kali㉿kali)-[~/Desktop/demodir]
$ touch abc.txt

(kali㉿kali)-[~/Desktop/demodir]
$ ls
abc.txt
```

- To write contents to a file:

```
(kali㉿kali)-[~/Desktop/demodir]
$ echo hello all > abc.txt
```

- To view contents in file:

```
(kali㉿kali)-[~/Desktop/demodir]
$ cat abc.txt
hello all
```

- To replace a text in the file:

```
(kali㉿kali)-[~/Desktop/demodir]
$ echo the conetn is replaced > abc.txt

(kali㉿kali)-[~/Desktop/demodir]
$ cat abc.txt
the conetn is replaced
```

- To add new lines to file without replacing:

```
(kali㉿kali)-[~/Desktop/demodir]
$ echo hello all,not replaced >> abc.txt

(kali㉿kali)-[~/Desktop/demodir]
$ cat abc.txt
the conetn is replaced
hello all,not replaced
```

- To find the type:

```
(kali㉿kali)-[~/Desktop/demodir]
$ file abc.txt
abc.txt: ASCII text
```

- To concat two files and view as single file:

```
(kali㉿kali)-[~/Desktop/demodir]
$ ls
abc.txt  a.txt

(kali㉿kali)-[~/Desktop/demodir]
$ cat abc.txt a.txt
the conetn is replaced
hello all,not replaced
this is fiel2
```

- To view all commands used till now:

```
(kali㉿kali)-[~/Desktop/demodir]
$ history
1  whoami
2  pwd
3  ls
4  ls -l
5  cd Downloads
6  ls
7  ls -l
```

- To find the current user

```
(kali㉿kali)-[~/Desktop/demodir]
$ whoami
kali
```

- To copy file to another directory:

```
(kali㉿kali)-[~/Desktop/demodir]
$ ls
abc.txt  a.txt  demodir  demodir1

(kali㉿kali)-[~/Desktop/demodir]
$ cp abc.txt demodir1

(kali㉿kali)-[~/Desktop/demodir]
$ cd demodir1

(kali㉿kali)-[~/Desktop/demodir/demodir1]
$ ls
abc.txt
```

- To move file to another directory:

```
(kali㉿kali)-[~/Desktop/demodir]
$ ls
abc.txt  a.txt  demodir  demodir1

(kali㉿kali)-[~/Desktop/demodir]
$ mv a.txt demodir1

(kali㉿kali)-[~/Desktop/demodir]
$ ls
abc.txt  demodir  demodir1

(kali㉿kali)-[~/Desktop/demodir]
$ cd demodir1

(kali㉿kali)-[~/Desktop/demodir/demodir1]
$ ls
abc.txt  a.txt
```

- To rename file :

```
(kali㉿kali)-[~/Desktop/demodir/demodir1]
$ ls
abc.txt  a.txt

(kali㉿kali)-[~/Desktop/demodir/demodir1]
$ mv a.txt hello.txt

(kali㉿kali)-[~/Desktop/demodir/demodir1]
$ ls
abc.txt  hello.txt
```

- To remove (empty) directory :


```

(kali㉿kali)-[~/Desktop/demodir]
$ cd demodir1

(kali㉿kali)-[~/Desktop/demodir/demodir1]
$ mkdir thisisemptydir

(kali㉿kali)-[~/Desktop/demodir/demodir1]
$ ls
hello.txt  thisisemptydir

(kali㉿kali)-[~/Desktop/demodir/demodir1]
$ rmdir thisisemptydir

(kali㉿kali)-[~/Desktop/demodir/demodir1]
$ ls
hello.txt

```

This command can only use when directory is empty.

```

(kali㉿kali)-[~/Desktop/demodir]
$ ls
abc.txt  demodir  demodir1

(kali㉿kali)-[~/Desktop/demodir]
$ rmdir demodir1
rmdir: failed to remove 'demodir1': Directory not empty

```

- To remove directory(not empty): we can use recursive deletion

```

(kali㉿kali)-[~/Desktop/demodir/demodir1]
$ ls
hello.txt

(kali㉿kali)-[~/Desktop/demodir/demodir1]
$ cd ..

(kali㉿kali)-[~/Desktop/demodir]
$ ls
abc.txt  demodir  demodir1

(kali㉿kali)-[~/Desktop/demodir]
$ rm -r demodir1

(kali㉿kali)-[~/Desktop/demodir]
$ ls
abc.txt  demodir

```

- To remove file:

```

(kali㉿kali)-[~/Desktop/demodir/demodir1]
$ ls
abc.txt  hello.txt

(kali㉿kali)-[~/Desktop/demodir/demodir1]
$ rm abc.txt

(kali㉿kali)-[~/Desktop/demodir/demodir1]
$ ls
hello.txt

```

- To view manual pages of program:

```

(kali㉿kali)-[~/Desktop/demodir]
$ man rm

```

- To view all files included hidden files and their details ,permission etc ..:

```

(kali㉿kali)-[~/Desktop/demodir]
$ ls -l
total 8
-rw-r--r-- 1 kali kali 46 Oct  4 22:46 abc.txt
-rw-r--r-- 1 kali kali 46 Oct  4 22:52 demodir

(kali㉿kali)-[~/Desktop/demodir]
$ ls -a
.  ..  abc.txt  demodir

(kali㉿kali)-[~/Desktop/demodir]
$ ls -la
total 16
drwxr-xr-x 2 kali kali 4096 Oct  4 23:01 .
drwxr-xr-x 4 kali kali 4096 Oct  4 22:43 ..
-rw-r--r-- 1 kali kali  46 Oct  4 22:46 abc.txt
-rw-r--r-- 1 kali kali  46 Oct  4 22:52 demodir

```

- To edit file:

```
(kali@kali)-[~/Desktop/demodir]
$ nano abc.txt
```

- To create new file and edit it:

```
(kali@kali)-[~/Desktop/demodir]
$ ls
abc.txt  demodir

(kali@kali)-[~/Desktop/demodir]
$ nano xyz.txt

(kali@kali)-[~/Desktop/demodir]
$ cat xyz.txt
hello all
```

- To find a file

```
(kali@kali)-[~]
$ find -name demo.txt
./Desktop/demodir/abc/xyz/mno/demo.txt
```

We can use this command in different ways.

Ex:

```
find -name *.password -user bandit7 -user bandit6 -size 32c
```

Here *.password, means the file name ends with that.

Bandit 7 is the name of user group

Bandit 6 is the name of another user

32c is the size in byte.

```
find -type f -size 1033c
```

F is the type

1033c is the size in byte

Check the manual page for more details

- Wordcount

```
tryhackme@linux1:~$ wc -l access.log
302 access.log
tryhackme@linux1:~$
```

- To find the line count

```
(kali@kali)-[~/../demodir/abc/xyz/mno]
$ cat demo.txt

(kali@kali)-[~/../demodir/abc/xyz/mno]
$ wc -l demo.txt
0 demo.txt

(kali@kali)-[~/../demodir/abc/xyz/mno]
$ echo hello all > demo.txt

(kali@kali)-[~/../demodir/abc/xyz/mno]
$ wc -l demo.txt
1 demo.txt
```

- To find particular line of word from file

```
(kali@kali)-[~/.../demodir/abc/xyz/mno]
$ cat demo.txt
hello all
hello all second line

(kali@kali)-[~/.../demodir/abc/xyz/mno]
$ grep "second" demo.txt
hello all second line
```

- To use alias name for commands

```
(kali@kali)-[~/Desktop]
$ alias nimisha="ls -l"

(kali@kali)-[~/Desktop]
$ nimisha
total 4
drwxr-xr-x 3 kali kali 4096 Oct  4 23:07 demodir
```

- To find the usage of a command

```
(kali@kali)-[~/Desktop]
$ whatis rm
rm (1) - remove files or directories
```

- To find IP

```
(kali@kali)-[~/Desktop]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING>
    inet 192.168.83.230 netmask 255.255.255.0
```

- To install software

```
(kali@kali)-[~/Desktop]
$ sudo apt-get install beef-xss
[sudo] password for kali:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

- To shutdown
power off
- To read dash files
cat ./-
- To view content of file with spaces in file name (file name :spaces in this filename)
cat spaces\ in\ this\ filename
- To sort content in files & find unique contents in file
cat data.txt | sort | uniq -u
- To convert decode to base64
echo VGhlIHh3b3JkUTkVViVVBSCg== | base64 --decode
- List all files
ls -la
- To find particular word or string in a file
grep 'millionth' data.txt
- To reverse Hexdump
xxd -r xyz.txt > abc.bin
- To reverse gzip
zcat abc.bin > xyz
- To reverse bzip2
bzcat xyz > pqr
- To reverse POSIX tar

tar -xvf <filename>

- To download a file from web via HTTP
wget <https://www.domain.com/abc.txt>
- Transferring Files From Your Host - SCP (SSH)
scp important.txt ubuntu@192.168.1.30:/home/ubuntu/transferred.txt
- Wordlist

```
(kali㉿kali)-[/usr/share/wordlists]
$ ls
amass  dirbuster  fern-wifi  legion  nmap.lst  sqlmap.txt  wifite.txt
dirb   fasttrack.txt  john.lst  metasploit  rockyou.txt.gz  wfuzz
```

PERMISSIONS

```
drwxr-xr-x 3 kali kali 4096 Oct 4 23:07 demodir
-rw-r--r-- 1 kali kali 0 Oct 4 23:24 demo.txt
```

Description

3	Hard-link count
kali kali	<owner user> <group user name>
4096	size of the file
Oct 4 23:07	Last modified date and time
demodir	directory name
Demo.txt	filename

If it starts with d, then it is directory.

If it starts with -, then it is a file.

Permissions

R	read
W	write
X	execute

It is divided to 3 users.

- Root user/owner
- Group permission/other person than owner
- Other users like remote access

We can identify directory with color. If any file starts with dot(.) it is hidden file

Change Permissions

Method 1: Symbolic method

chmod +x <filename> -----set execute(x) permission to all user/group/others

Chmod u+rw <filename> -----set rw permission to owner user(g:group,u:owner,o:others)

Chmod u-r <filename>-----to remove read(x) permission for owner user

Method 2 : Numeric method

rwX

000	---	0
001	--x	1
010	-w-	2
011	-wx	3

```
100  r--    4
101  r-x    5
110  rw-    6
111  rwx    7
chmod 444 demo.txt -----setting read permission to all 3 user
```

CTF challenges : <https://overthewire.org/wargames/>
Command : Ssh <username>@<domainname> -p <port no>

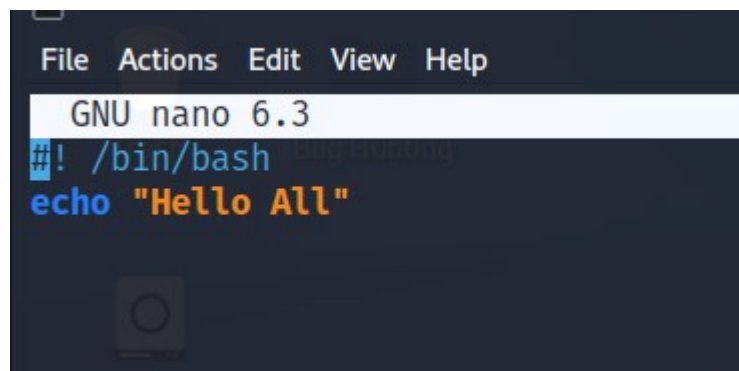
TryhackMe : Fundamentals part 1,2,3

Bash Scripting

BASH is an acronym for Bourne Again Shell, a punning name, which is a tribute to Bourne Shell (i.e., invented by Steven Bourne). Bash is a shell program written by Brian Fox as an upgraded version of Bourne Shell program '**sh**'. It is an open source GNU project. It was released in 1989 as one of the most popular shell distribution of GNU/Linux operating systems. It provides functional improvements over Bourne Shell for both programming and interactive uses. It includes command line editing, key bindings, command history with unlimited size, etc. In basic terms, Bash is a command line interpreter that typically runs in a text window where user can interpret commands to carry out various actions. The combination of these commands as a series within a file is known as a Shell Script. Bash can read and execute the commands from a Shell Script.

Step 1: type "nano hello.sh"

Step 2 : type below commands



#! (shebang) and specifies the bash shell location.

Step 3 : press,Cntrl +O ,to save

Step 4 : press,Cntrl + X ,to exit

Step 5 : To Run ,

Method 1

```
bash hello.sh
```

Method 2

```
chmod +x hello.sh
./hello.sh
```

Single Line Comments

```
#this is comment
```

MultiLine Comments

Method 1

```
<<COMMENTS
    This is the first comment
    This is the second comment
    This is the third comment
COMMENTS
```

Method 2

```
: '
    This is the first comment
    This is the second comment
    This is the third comment
'
```

Arithmetic Program

```
#!/bin/bash

x=8
y=2
echo "x=8, y=2"
echo "Addition of x & y"
echo $(( $x + $y ))
echo "Subtraction of x & y"
echo $(( $x - $y ))
echo "Multiplication of x & y"
echo $(( $x * $y ))
echo "Division of x by y"
echo $(( $x / $y ))
echo "Exponentiation of x,y"
echo $(( $x ** $y ))
echo "Modular Division of x,y"
echo $(( $x % $y ))
echo "Incrementing x by 5, then x= "
(( x += 5 ))
echo $x
echo "Decrementing x by 5, then x= "
(( x -= 5 ))
echo $x
echo "Multiply of x by 5, then x="
(( x *= 5 ))
echo $x
echo "Dividing x by 5, x= "
(( x /= 5 ))
echo $x
echo "Remainder of Dividing x by 5, x="
(( x %= 5 ))
echo $x
```



```
File Actions Edit View Help
GNU nano 6.3
#!/bin/bash
d=`date +%m-%d-%Y`
echo "Date in format MM-DD-YYYY"
echo $d #MM-DD-YYYY
```

```
File Actions Edit View Help
GNU nano 6.3
#!/bin/bash
echo "What is your name?"
read a
echo "my name is $a"
sleep 3
echo "what is your age?"
read b
echo "my age is $b"
```

```
GNU nano 6.3
#!/bin/bash
echo "enter first number?"
read a
echo "enter second number?"
read b
((sum=a+b))
echo sum
```

```
File Actions Edit View Help
GNU nano 6.3
#!/bin/bash
echo "what is your name?"
read a
echo "$a,you are in $(pwd),using the user $(whoami)"
```

```
GNU nano 6.3
#!/bin/bash
echo "what is your name?"
read a
echo "$a,you are in $(pwd),using the user $(whoami)"
echo "your files are $(ls -la)"
read -p "enter your username:" b
echo "my name is $b"
```

```
File Actions Edit View Help
GNU nano 6.3
#!/bin/bash
echo "enter 2 num"
read a
read b
if [[ a -gt b ]]; then
echo " $a "
else
echo " $b "
fi
```

```

GNU nano 6.3
#!/bin/bash
read -p "Enter a number of quantity:" num
if [ $num -gt 100 ];
then
echo "Eligible for 10% discount"
elif [ $num -lt 100 ];
then
echo "Eligible for 5% discount"
else
echo "Lucky Draw Winner"
echo "Eligible to get the item for free"
fi

```

```

GNU nano 6.3
#!/bin/bash
echo "1:print hello"
echo "2:print hai"
echo "3:print hii"
read -p "select an option :" opt
case $opt in
1)
echo "hello";;
2)
echo "hai";;
3)
echo "hii";;
esac

```

```

GNU nano 6.3
#!/bin/bash
echo "Enter the limit:"
read N
for((i=1;i<=N;i++))
do
echo "$i"
done

```

```

GNU nano 6.3
#!/bin/bash
echo "enter a number"
read n
factorial=1
for((i=1;i<=n;i++))
do
factorial=$((factorial*i))
done
echo "the factorial is $factorial"

```

```

GNU nano 6.3
#!/bin/bash
n=1
while [[ n -le 10 ]];do
echo $n
n=$((n+1))
done

```

Until Loop : the loop executes when the condition is false;it will exit when the condition is true

```

GNU nano 6.3
#!/bin/bash
n=1
until [[ n -gt 10 ]];do
echo $n
n=$((n+1))
done

```

Functions without parameter

```

GNU nano 6.3
#!/bin/bash
demo()
{
echo "hello"
}
demo

```

Function with parameter

```
GNU nano 6.3
#!/bin/bash
#Script to pass and access arguments
function_arguments()
{
    echo $1
    echo $2
    echo $3
    echo $4
    echo $5
}
#Calling function_arguments
function_arguments "Hello" "am" "here" "to" "explain."
```

```
(kali@kali)-[~/]
$ bash abc.sh
Hello
am
here
to
explain.
```

Function with return values

```
GNU nano 6.3
#!/bin/bash
#Setting up a return status for a function
print_it () {
    echo Hello $1
    return 5
}
print_it User
print_it Reader
echo The previous function returned a value of $?
```

```
(kali@kali)-[~/Desktop/Bug Hunting/Bash]
$ bash abc1.sh
Hello User
Hello Reader
The previous function returned a value of 5
```

Array

```
#!/bin/bash
echo "Enter names : "
read -a names
echo "The entered names are : ${names[0]}, ${names[1]}."
```

```
(kali@kali)-[~/Desktop/Bug Hunting/Bash]
$ ./program3.sh
Enter names :
nimisha archana
The entered names are : nimisha, archana.
```

Seminar Topic - Linux Boot Process