Linux Basics for Hackers - Notes

1) BINARIES

Located in -> /usr/bin and/or /usr/sbin

2) LINUX FILE SYSTEM

```
'/' -> The actual root system. The top most.
```

Inside it ->
/boot - Kernel image
/home - user dir
/proc - view of internal kernal data
/dev - special device files
/sbin - binaries
/root - SuperUser's Home Dir (different from '/')
/etc - Sys config
/mnt - GenPurpose Mount point
/sys - Kernel's view of HW
/bin - also binaries
/lib - libraries
/usr -> /sbin, /bin, /lib (more of the same stuff) /media - for ejectable media

3) cd Command

use '..' to move up 1 level
'.. ..' for 2 levels & '.. .. ..' for 3 levels and so on

4) Search-based commands

$ locate 'find_this' -> finds all occurences
$ whereis 'module_name' -> finds all BINARIES of the target (usually with man pages)
$ which aircrack-ng -> finds the binary file located in the PATH variable of the system
$ find -directory -option -targetExp -> finds literally everything [Eg: find /etc -type f -name apache2] -- altho apache2.* will find all file extensions but first name as apache2
Lastly, grep to filter
$ ps aux | grep apache2 -> will filter from all auxilliary processes containing apache2

5) 'cat' is versatile

$ cat file_name -> will spill the file contents.
$ cat > file_name -> will let you write in it BUT WILL REPLACE ALL EXISTING DATA.
$ cat >> file_name -> will actually let you append the text you enter.

6) Renaming doesn't exist in Linux

So we use
$ mv newfile newfile2 -> to essentially rename the file

7) REMOVING A DIRECTORY

Only the directory : $ rmdir
When that fails : $ rm -r (recursively delete everything in it)

8) TEXT MANIPULATION

head and tail ->
$ head file_name -> first 10 lines
$ head -n file_name -> n number of lines from the start
$ tail -> for bottom lines (+ specialized with a count)

$ nl path_to_file_or_file_name -> will number all the lines. // cat can and should be clubbed with grep as and when needed. Eg: cat snort.conf | grep output

Sed command - for find and replace ->
$ sed s/search_term/replace_term/g path_to_filename > newfile_name
-> 's/' will find the term, '/g' is for replacing globally. Rest is elementary.
-> Removing the '/g' will only replace the first occurrence.
Adding a number there can limit the number of occurrences to be changed. '/3' will only replace the first 3.
Eg: sed s/mysql/MySQL/g /etc/snort/snort.conf > snort2.conf

$ more file_name -> offers a scroll-able page if the file is to big.
$ less file_name -> "less is more" -- offers a filter to search for the term should you need to -- use the '/' key. Still scroll-able but with better functionality.

9) NETWORKING

'loopback' addr -- same as 'localhost' = 127.0.0.1
iwconfig - check for wireless adapter info -- good for getting power, the mode [monitor,

managed, promiscuous] etc.
Changing info -> MAC or IP Addr :

```
$ ifconfig eth0 192.1688.181.155
```

-> modifies what your router sees to redirect packets

Changing Netmask +/ Broadcast :

```
$ ifconfig eth0 192.168.181.155 netmask 255.255.0.0 broadcast 192.168.1.255
```

-> Netmask is the subnet mask -- determines the portion of the IP Addr to the NW and which refers to the host. Here, first 2 octets (16 bits) represent the network and last 2 show the hosts within that network.
-> Broadcast is the addr used to send packets to all hosts on the same network segment. Default (bcz of subnet) would become 192.168.255.255 but here overridden to 192.169.1.155 -- Basically packets sent to this will be broad-casted on that specific sub-network.

MAC Spoofing :

Take down the interface, change the Addr, restart

```
$ ifconfig eth0 down > ifconfig eth0 hw ether 00:11:22:33:44:55 > ifconfig eth0 up
```

Assigning new IP via DHCP Server :

Server runs of '*dhcpd*' - the daemon. Requested via '*dhclient*'. Requires a DHCP assigned IP addr. (Note : 'dhclient' is for Debian, different for all other distros.)

```
$ dhclient eth0
```

->DHCPDISCOVER req is sent by this command and then receives an offer from the DHCP i.e DHCPOFFER. Now, ifconfig will show a difft IP addr as given by the DHCP Server.

Manipulating DNS :

Use 'dig' :
Directly pass-on the domain and add the 'ns' tag to make the domain as the nameserver itself. 'mx' will fetch the mail-exchange server.

```
$ dig hackerarise.com ns OR $ dig hackerarise.com mx
```

-- Some Linux servers use BIND (Berkeley Internet Name Domain) which is just a fancy name for DNS.

Changing your DNS Server :

Edit a file stored in '/etc/resolv.conf'. There you will see the domain, search & nameserver fields. Swap the values here to switch your DNS server.
Other method to do the same (although this cleanly overwrites the file's content) ->

```
$ echo "nameserver 8.8.8.8" > /etc/resolv.conf
```

Mapping your own IP Addrs :
'hosts' file located in /etc/hosts. Useful for hijacking a TCP connection on your LAN to direct traffic to a malicious webserver by using a tool like *'dnsspoof'*
Usually this file has a mapping for your localhost only BUT you can map any website to any IP Address. Eg : "192.168.181.131 bankofamerica.com". Decent for local network attacks.
[dnsspoof & Ettercap can be used]

10) HANDLING SW PACKAGES

Search for package in local repo:
$ apt-cache search *'keyword'*
-> Eg: $ apt-cache search snort

Install, Remove, Purge, Update, Upgrade :

```
$ apt-get install --name--
$ apt-get remove --name--
$ sudo apt-get update
$ sudo apt-get upgrade
$ apt-get purge --name--
```

(purge removes the config as well)
or use a package manager like *'synaptic'* or *'gdebi'* like a normie.

Adding Repos to Sources.list file :
$ mousepad /etc/apt/sources.list -> will open the list
Categories : main (OSS), universe (community maintained OSS), multiverse (SW restricted by copyright), restricted (proprietary device drivers), backports (packages from later releases)
Format : "deb http:// ------ --package_name-- main non-free contrib" etc.

11) FILE DIRECTORIES & PERMISSION

r,w,x -> read, write and execute

Granting ownership :
$ chown *username file-path-name* -> Provides the ownership of that file to that user
$ chgrp *group-name package-or-module-name* -> Provides a user-group access to that module

Checking Permissions :
$ ls -l *file-or-path-to-it* -> will lay down the whole sheet. The type, permission on the file for

owner/groups/users, number of links, the owner, size in bytes, creation/mod date & its name.

Eg: "drwxr-xr-x" vs "-rw-r--r--". First letter denotes directory if 'd' or file if empty dash.

Followed by the permission values for 3 groups i.e owner then group then other_users.

Hence we observe 3 values at a time. Dash means no permission ofc.

In '-rw-r--r--' -> File, owner has read/write, group and other users only have read permissions.

2nd way : There is a proper calculation done in Octal terms as well.

```
001 : 1 : --x
010 : 2 : -w-
011 : 3 : -wx
100 : 4 : r--
101 : 5 : r-x
110 : 6 : rw-
111 : 7 : rwx
```

Total RWX is 7. Since we have 3 sets of permissions, giving a full read+write+execute permission to everyone, for example, would look like ->

$ chmod 777 hashcat.hcstat

3rd way : UGO Syntax

Here, '-' removes a permission, '+' adds and '=' sets a permission.

Eg: Remove the write (w) permission from user on a file

$ chmod u-w hashcat.hcstat -> Now -rw-r--r-- becomes -r-xr-xr--

Or for user and other users at once

$ chmod u+x, o+x hashcat.hcstat

Now, you can set execute permission for yourself on a newly downloaded tool/script bcz by default Linux won't set it

$ chmod 766 some_new_tool -> grants us (the owner) all permission including execute -- and everyone else only R/W permissions.

Masking can be done :

$ umask 007 -> set it so only the user and members of the user's group have permissions.

Special Permissions :

SUID - set user ID & SGID - set group ID

1) Granting Temp Root w/ SUID

/etc/shadow contains all user's password -- requires root privileges to execute. SUID requires an additional bit before the permission bit. So 644 becomes 4644 i.e

$ chmod 4644 *file_name*

2) Granting the Root user's Group permissions SGID
SGID works differently. Someone without execute permission can execute a file if the owner belongs to the group that has the permission to execute that file. When the bit is set on a directory -- t**he ownership of new files created in that directory goes to the directory's creator's group rather than the file creator's group.**
$ chmod 2644 *file_name* [SGID bit is represented by 2 and SUID uses 4]

Privilege Escalation :
One way - exploit the **SUID Bit** in the system. Eg: Scripts that need to change the password usually come with the SUID bit set already. Use that to gain temporary root priv - then do something shady like getting the file at /etc/shadow.
To proceed -> Use commands like 'find' to find the files and see their bit. Example :

```
$ find / -user root -perm -4000
```

Kali now starts at the top of the filesystem (because of '/') and looks everywhere below this -- the file that are owned by 'root' & specified with 'user root' + have the SUID bit set (-perm -4000). This command will give an output like ->
/usr/bin/chsh ; /usr/bin/gpasswd; /usr/bin/pkexec; /usr/bin/sudo; /usr/bin/passwd,.. etc.
Navigating to this directory, and observing, let's say "sudo" using ls-alh, you will see ->
-rwsr-xr-x root root 140944 *date* sudo
Here, the 's' in place of 'x' determines the SUID bit. Logically, anyone who runs the *sudo* file has the priv of a root user -- which becomes an attack vector IF an application -- which needs access to /etc/shadow file to successfully complete their task -- can be hijacked.

12) PROCESS MANAGEMENT

To view - use -> $ ps
Every process ofc has a PID or process ID. You can use -> $ kill *PID_value* to kill any process.
Issue ? 'ps' command won't give you much info either ways. We have another command for that ->

```
$ ps aux
```

It shows the USER, PID, %CPU, VSZ, RSS, TTY, STAT, START, TIME & COMMAND.

Filtering by Process Name
For instance, try running *msfconsole* command to have its process running. Then use *grep* to filter it. This way you can filter all the processes running/attached to it.

```
$ ps aux | grep msfconsole
```

You might see a few, such as the attached DB running, the ruby script, etc, and finally the program itself.

We also have commands like "*top*" to monitor the processes sorted by their resource usage. It's active i.e refreshes on its own (every 3-4 seconds)

Managing Processes

We can alter the affinity/priority of any process by using the *"nice"* command (by passing a numeric value to its argument '-n'). Kernel always has the final say, we're just suggesting.
The value ranges from -20 to +19. Sadly, **HIGH value means LOW priority and vice versa**. So -20 is most likely to receive priority, 0 is default ofc, and +19 is least likely.
Usually, any process inherits the *nice* value of its parent process.
Unsurprisingly, you can alter the priority by using the *"renice"* command.
DIFFERENCE !!
nice is relative. Its adds/subtracts the priority value given what you pass to it. A process with a priority of 15, when asked 'nicely' to be -10 will have a priority of 5 now. OR when asked to be +5, it will now be 20. 'nice' can use the process via its location as well.
renice is absolute. Requires a fixed value b/w -20 and +19. BUT it sets the process to that level, cuz you've altered the deal and it prays you don't alter it any further. It also requires the PID.
Examples :
$ nice -n 10 /bin/some_slow_process [lowers it]
or
$ nice -n -9 /bin/some_slow_process [improves it]
and
$ renice 20 6996 [6996 is the PID of some_slow_process, and 20 is setting it]
NOTE: 'top' can also be used to alter these values.

Killing Processes

'kill' command is your friend. Just pass the PID and pass the required kill signal. There are 64 of them. Default is SIGTERM (n=15) i.e termination. Ofc they are optional. Use them as a flag arg while using kill command.
$ kill -n *PID*
Example - $ kill -9 6887
Signal Interrupts for kill ->

```
SIGHUP (1) : Hangup - stops and then restarts with the same PID
SIGIN (2) : Interrupt - weak kill signal not guaranteed to work but does work mostly.
SIGQUIT (3) : Quit/Core dump - terminates but saves the process info in memory + inside pwd.
SIGKILL (9) : absolute kill signal. Forces the process to stop by sending the process's resources to a special device -- /dev/null
```

Basically : to restart - use '-1' ; for zombie/malicious - use '-9'

Running Processes in the Background
Everything runs from within the shell and the shell waits for the task/command to run/finish. It waits for this whole sequence -- hence busy & won't allow any new commands. To prevent this we can essentially detach the process from the shell. Use the *'&'* right after the task.

```
$ mousepad someDoc &
```

Moving to foreground
Use the *'fg'* command followed by the PID. Fetch the PID if needed.

```
$ fg 1273
```

Process Scheduling
Either use *'at'* or *'crond'*.
'at' is useful for scheduling a job to run once at some point in the future -- execute 1 or many commands in the future, passed with time as argument.

```
Eg: $ msfconsole at 7:20PM June 13
OR $ msfconsole at now + 20 minutes
```

// If you just write 'at' followed by a time, then the "at>" console will open asking you to map the file/process path for it.

'crond' is best for scheduling tasks to occur everyday/week/month etc. [SEPARATE CHAPTER LATER]

13) MANAGING USER ENVIRONMENT VARIABLE

Always 2 there are. Environment and Shell variables.
EnV - always uppercase, system wide, controls the way the system acts/looks/feels + inherited by child shells or processes.
ShV - usually lowercase + valid only in the shell they are set in.
Format -> KEY=value1 OR KEY=value1:value2:value3...

To see the EnV's, use

```
$ env
```

View All
To see vars of all types (including shell, local + shell functions) use "set" (and preferably filter it with 'more' to have a scrollable feed)

```
$ set | more
```

OUTPUT :

```
BASH=/bin/bash
BASHOPTS=check........:.....
BASH_ALIASES=()
BASH_ARGC=()
```

....etc

Grep can be used to filter
$ set | grep HISTSIZE -> filtering via grep. HISTSIZE is one such var that contains the maximum number of commands your command history file will store. It does not store the commands themselves just the number of them that can be stored.
Modify these vars
Simply set them. Eg: $ HISTSIZE=0

Making Var Changes Permanent