**Red Hat Certification System Administrator**

# Linux Architecture

The kernel is linked directly to the hardware components. The shell is a higher level than the kernel and we have the highest level that is the closest to the user which is the user interface.

In linux, everything is a file. It is known as flat file so for example in linux a directory is literally a file with the type ‘d’. So if a virus is spread, it won’t be able to spread across files inside of the directory.

# Linux overvirew

In linux, everything is a file. Linux is known as flat file so for example a directory is literally a file with the type ‘d’ which means that if a virus is spread, It won’t be able to spread across files inside

# Basic directories present in Linux OS

- Under the ‘/’ directory which is the root we have multiple directories like home directory, root directory for the root, dev, etc, bin etc…

- /dev : Contains devices that are attached to the system

- /etc : Contains config files. Does not contain binary files like scripts

- /bin : Contains essential user programs and commands like cat, cd, etc…

- /sbin : Like bin but contains also commands intended to be executed by root administrator

- /home : Contain all user profiles

- /lib : Contains libraries needed by /bin

- /mnt : It’s where system admins mounted temporary file systems while using them

- /media : Contains subdirectories where removable media device inserted into the computer are mounted

- /opt : Contains subdirectories for optional software packages. For example, if we install chrome browser, it will be stored here

- /proc : Contains special files that represent system and process info

- /run : Fairly new directory. Gives apps a standard place to store transient files they require like sockets and process IDs.

- /srv : Contains data for services provided by the system. So app data can be stored here

- /sys : Virtual file system which stores info about devices connected to the system

- /usr : Contains apps and files used by users as opposed to apps and files used by the system

- /var : Contains logs, mail, temps files… This directory is expected to grow

- /tmp : Contains temporary files. Accessible by root and user. When reboot happens, data in /tmp will be deleted

# Check status of services

**systemctl** **status** sshd : This command will check the status of the service sshd which is the service responsible for remote connections on servers (OpenSSH)

# RPM and YUM

Understand RPM package :

bash-completion-2.7-5.el8.noarch

Package name

Version

Update

Patch nb

Linux version

Package type

♣ noarch means no Architecture so it works on 32 and 64 bits.

● RPM : Red hat package manager

Does not resolve dependencies

Manual updates

Difficult to manage installing/upgrading packages

● YUM : Red hat package manager

Does not resolve dependencies

# su **vs** sudo

With **su**, we must enter the ‘root’ password so the administrator needs to give his root password to other users. However, with **sudo**, the user must be in the sudoers file so that he won’t need the root password of the admin.

# Parameters of sudo

**root ALL=(ALL) ALL**

**Username MachineName = (EffectiveUser) Command**

*User\_name: This is the name of ‘sudo‘ user.*

*Machine\_name: This is the host name, in which ‘sudo‘ command is valid. Useful when you have lots of host machines.*

*(Effective\_user): The ‘Effective user’ that are allowed to execute the commands. This column lets you allows users to execute System Commands.*

*Command: command or a set of commands which user may run.*

# Check ip address

**ip addr** : To see the interfaces and ip addresses.

# Relative path vs Absolute path

Absolute path is when we start from the root directory and pass to another directory (/home/elie)

# Difference between soft and hard link

Soft link connects to the original file and the original file is connected to the data on the HDD so if the file to which the soft link points is removed, the soft link will no longer have access to the data. In hard link, the link is connected to the original file but have another link also which is linked to the data in the HDD. In softlink, we will see the symbol of the file which is not the case with hardlink where we can not find the difference between linked and original. With softlink, we can create links across file systems which is not the case with hardlink. With soft link we have different inodes so the link will have an inode value and the original file will have different value of inode which is not the case with hardlink.

# Soft link

**ln -s** softlinksource softlinkdest : The ‘softlinkdest’ is the shortcut for the ‘softlinksource’ file

**stat** softlinkdest : We will see that size is equal to 0

**stat** softlinksource : We will see the actual size of the file to which we are pointing.

If we do changes on dest file we will have same content on source file and vice versa

**rm -rf** softlinksource

**cat** softlinkdest : It will not work because the original file is no longer accessible.

♣ Same thing with directories

**ln -s** Kenlm/LICENSE : Create link to that file with same name

# Hard link

**ln** hardlink destlink : Now when I execute the command ‘stats’ on the destlink we will see the field ‘links’ having a value equal to 2 which means that it’s linking to the source (hardlink) and to the data. So when we delete the source, we will still have the data.

♣ Commands are case sensitive

♣ Hard link does not work on directories like soft link does

♣ Hard link does not have the typle ‘l’ that means link when we run the command ‘ll’ but soft link does show us that it is of type ‘l’. So Hard link is of type file because it has ‘-‘

# Open a file and write in it

**cat >** firstFile

Here we can write some text but if there were some info in the file before it will be overridden.

**cat >>** firstFile

Here we will not override we will just add.

**cat** /etc/sudoers **>>** /opt/sudoers : Backing up to /opt/sudoers

**cat -n >>** firstFile

Will add the number for each line

# Time execution

**time ls -la** : Shows the time execution for ‘ls-la’

# ns lookup

**nslookup** google.com : Show us DNS server and other details (Resolves name to IP address)

**nslookup** 218.58.137.46 : Resolves IP address to name

In fact with nslookup we don’t have a lot of details. We can use the command :

**dig -x** google.com : Will show us mail server of google.com

# Tree command

**tree** /root/ : Will show us all subdirectories

**tree /**L1 : See the ‘/’ on level 1

# Traverse between directories

**cd -** : Go to the last directory.

**cd ~** : Go to home directory.

# Work with directories

**mkdir -p** 1/2/3/4 : Here we created 4 direcotires nested in each other so the the directory 1 is the first parent.

**mkdir** {2008..2021}-{01..12}

**rmdir 6/ :** Removes the directory 6 but in condition that the directory is empty otherwise we should add the parameter **-f**

**rmdir -rf** 4/ : ‘r’ means recursif so here it will delete nested directories. We can also add the **-i** that will ask for confirmation before deleting

# Check the history of commands

**history** : Show all executed commands and each command will be listed on a line with the number of the command, let’s say I want to execute the command of the line 71 I can do : **!71**

**!!** : Will execute the last command

**history -c** : Will delete the history

♣ The bash history file of the root user is saved in /root/.bash\_history We can also check while visualizing the environment variable : echo $HISTFILE

# Check info about machine and os

**uname -a**

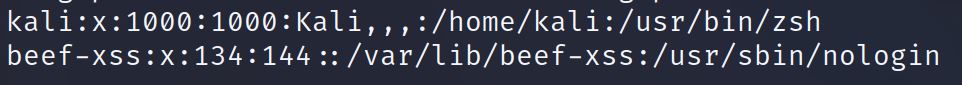
**cat** /etc/os-release

# File ownership

- File owner can only be changed by **root** user but access permissions can be changed by both **root** and file owner

- Shadow passwords improve system security by making passwords encrypted from

**/etc/passwd** to **/etc/shadow** and it is readable only by the **root** user.



So this file will show us the following : **user Name**, **password** **user ID**, **group ID**, **Description,** **Home directory** and **shell**. The group ID is stored in the **/etc/group**

In fact the root user has an ID of 0 for the user ID and group ID.

Owner : Group : World : Path

4 read (r)

2 write (w)

1 execute (x)

7 = 4+2+1

6 = 4+2

**ls -l** file ou **ll** file:

- rw- rw- r--

The - can be either ‘d’ which stands for directory or ‘l’ that stands for link or ‘-‘ that means file.

**chmod** 554 file : Will give the following access rules for the file.

**chmod +x** : Add execute permission to owner and group of file.

**chmod u+rw** : Add r/w to user.

**chmod -R+x** : Add execute permissions to files and directories permissions recursively.

**chmod g-rw** : Removes read and write for group

**chmod ugo+rwx** : Adds r/w/x for user,group and others.

**chmod -R 770** Downloads/ : Changes recursively the permissions for this directory.

**chown** root file : Change the owner group of file so root is owner of file.

**chown** root : root file : Changes the owner and group of file to root.

**chgrp** root file : Change the group to root for the file.

# Work with files

**touch** file-‘date +%d-%M-%Y’

Result : file-4-12-2021

**touch** file{1..100}.txt

Here we will create files going from 1 to 100

# Copy some files

**cp** .txt Downloads

Will copy all text files to the Downloads directory.

**cp -pv** file Downloads/

The ‘-p’ is used to preserve parameters so for example the modification date of the copied file will be same as before copying. The ‘-v’ means verbose which will show us a sentence explaining what happened.

**cp -r** Desktop/ Downloads/

The ‘-r’ is used to copy a directory.

**scp** backup.tar.gz root@192.168.2.140:/opt/ : Will copy file to opt directory on the remote server.

# Move some files

**mv** file othernamefile : Renames the file.

**mv** file Downloads/ : Moves to Downloads.

# List some files

**ls -m** /home/elie : ‘-m’ is used to list files separated with a comma. Usually used when exporting.

**ls -R** : list subdirectories also.

**less** file.txt : Display file content 1 page at a time and we can jump to other page with space button.

**more** file.txt : Do the same thing but here we can see the percentage so when we jump to another page we can see how much pages left we got to see

# Difference between less and more

More has some limitations, in fact with more we can not scroll back so once we jump to another page we can not go back, which is not the case with thes less command. But more give us the functionality of percentage.

# Compare two files

**diff** file1 file2 : Verify if two files are equal

# Output some data

**head** : By default, prints first 10 lines of file

**head -n** 15 : First 15 lines

**tail** : last 10 lines

**tail -n** 20 /var/log/messages **>** /home/aravi/logfile

# Arithmetic expression

**echo $**((7+3)) : Will output 10

# Brace expansion

**echo** abc{elie,joe}xyz : Will output 2 things : abceliexyz and abcjoexyz

**echo** {090..100} : 090 091 092 … 100

**echo** a{A{1,2},B{3,4}}b : aA1b aA2b aB3b aB4b

# Work with echo

**echo $**{USER} : Will output the value of the variable USER

**echo \**$USER : Outputs \USER

**echo** my name is \\$USER : Will output my name is elie because the \\ will cancel the one the other.

# Work with partitions

**df -h** : Will show us partitions and directories with some details.

**du -sh** /etc/\* : Shows directories size inside of etc directory in human readable format.

**df -x** tmpfs : List the actual file system details, not the temporary file systems.

**df -T -x** tmpfs : To see type of file systems also

# Work with users

**who** : See the users logged to the machine.

**whoami** : Show the actual user that is logged.

**w** : Similar but gives more details like when user was logged in and out…

# Work with processes

**ps -aux** : Show all running processes.

**ps -U** elie : Show the processes running by the user elie.

# Uptime command

**uptime** : See when machine (server) was rebooted and how many user are logged in…

# Get help for a certain command

**ls --help**

**man ls**

# Search for files

**locate** or **find** : Locate is faster but need database update. Find is used when we know where the file is located.

**updatedb** : Will do a database update

**find** /home/ **-i -name** “file1” : The ‘-i’ will ignore the case sensitive.

**locate -i** fil : If you know a fragment of file

**locate -ib** fil : Without the ‘-b’ it will take the path as a search parameter so it will output everything in it. The ‘-b’ will force to use the last thing we wrotes in the command.

# Archieve and compress

Archieving is different than compressing. Archieving means putting files into one file without reducing size so without compressing.

We have 3 types of zip methods :

- gzip

- bzip

- zip

We can archeive+zip in one method :

- tar

**tar -cvf** archieve.tar varlog.tar.gz anothervar.rpm : The “-c” means create, the “-v” means archeive and the “-f” means specify files. So here we archieve files into one file without compressing.

**tar -cvzf** archieve.tar varlog.tar.gz anothervar.rpm : The “-z” will zip the resulting file.

**tar -uvf** archieve.tar \*.txt : We can update the contents of the archieve file so for example if we made changes in the folder and want to recompress it we can just update. Here we are saying to check all text files that has been changed to update the zip folder in the end.

**tar -tf** archieve.tar : The “-t” will list the files that are present in the archieve folder.

**tar -xvf**  archeive.tar : Extracts files from archieve.

**zip** -9 **-r** test1.zip /home/aravi : Here we used zip so it is another method and “-9” let us do high compression and here we are compressing all subfiles into one archieve which is tes1.zip

**zip -d** test1.zip home/aravi/ioen/testing.txt : The “-d” is used to delete specific file from the archieve.

**zip -fr** test1.zip \* : Here the test1.zip will be updated.

**unzip** test1.zip : To unzip file

# Redirection of errors

Standard input : 0

Standard output : 1

Standard error : 2

**ls** /root 2**>** /temp/errors : Will insert the error line into errors file and the ‘2’ refers to the standard error value.

**find** /etc **-name** passwd **>** /tmp/output 2 **>** /tmp/error : Will redirect the output file to /tmp/output and if there is an error, the /tmp/error will be filled with the error line.

**ls** /home/aravi **>>** /tmp/errors 2 **>** &1 : The **>>** will let insert multiple things instantly so here we insert the output and the error if there is one.

# Pass output as input for a command

**grep** aravi **<** /tmp/output : Will give the output of /tmp/output to the command as an input

# Grep command

**grep -e** “First” **-e** “last” demofile : So the ‘-e’ must be used if we want to search multiple strings

**grep -v** “First” demofile : The ‘-v’ is used to unmatch the word so we will output everything except the word “First”.

**grep** “First” **-A** 2 file : Will output line containing the word “First” and 2 lines after.

**grep** “First” **-B** 2 file : Will output line containing the word “First” and 2 lines before.

**grep** “All” **-C** 1 file : Will output 1 line before and 1 line after.

**grep -ril** /home/aravi/ : The “-l” is used when searching files or directories. The ‘-r’ stands for recursvie and the ‘i’ means ignore case sensitivity.

**grep -l** demo\* : Will output the files having “demo” included in their name.

**grep -c** first demofile : Will output the number of times the word “first” is found in demofile.

**grep** “demo**$**” demofie : Will output the line that ends with the word “demo”.

**cat** /var/log/messages **| grep** “^Sep 26 03:26:41” : Will search for lines beginning (^) with date.

# Vi&Vim editor

Vi is the best editor to use in Linux. It is used to modify config files.

The difference between Vi and Vim is that Vim is an imoproved version of Vi where output is colored.

# Nano editor

Vi is more complicated than nano with its key bindings. Nano is like windows notepad.

- Provides more features than pico

- Colored text for writing scripts

- Smooth in scrolling

- Simple control keys

- Regular expression support to search text in file

**nano** file1

# Manual page

**man -s** 5 passwd : Will give us passwd(5) which is the 5th sub category.

**man -k** printf : Shows us how many related command are there. So all commands related to printing.

**man -s** 5 **-k** passwd : Show the command linked to passwd in the 5th sub category.

♣ In the manual page I can enter / and search.

**pinfo** ls : Similar to man command but different the way and content.

**info** ls : Exactly similar to man command.

# whereis and whatis command

**whereis** python : Show us where it is located

**whatis** ls : Brief description of command

# SSH Servers

Here we are going to explore OpenSSH server&client configuration so how to securely log in to the server and how the client is authentified. OpenSSH use port 22 by default for communicating between client and server. Public and private keys are used for cryptographics purposes.

- Firstly we must install the OpenSSH package :

**yum install** openssh

♣ Difference between yum and rpm : Yum is a package manager and rpms are the actual packages. With yum you can add or remove software. The software itself comes within a rpm.

**rpm -qa | grep** openssh : Will show us the installed version of openssh, so here if we are on the server side we will see the installed package.

- Now we have to configure the config files and SSH keys on server side and authenticate the client.

**systemctl status** sshd : Will show us the status of the service openssh.

**systemctl enable** sshd : To enable the service openssh if it is not enabled.

- Add the service to the firewall :

**firewall-cmd --permanent --add-service=ssh**

- Now we must verify the config files :

**vi** /etc/ssh/sshd-config : Now when we do access of the file we can add some code

a) Add the protocol type responsible for a secured environnement :

Protocol 2

UsePAM yes

AllowGroups sshusers (This group only will have access to the ssh server)

- Now we will create the group

**groupadd** sshusers

**usermod -Ag** sshusers aravi : Add user aravi to group sshsuers

**id** aravi : To verify that aravi is part of the group sshusers.

- Now we will try to connect to the server

**ssh** aravi@192.168.2.140 : Here he will ask for passwd of aravi and we will be able to connect to ssh server.

♣ I can also use Putty and choose the connection type as SSH and he will also ask for user’s password.

# SCP (Secured copy)

In this section, we will learn how to securely copy files from a source to another.

- Source IP Server : 192.168.2.140

- Destination IP Server : 192.168.2.42

**scp** file1 root@192.168.2.42:/root/ : The ip address and location in orange is the ip address of the destination server and the location where we want to store the file.

We can add multiple files simulatenously

**scp -r** directoryName root@192.168.2.42:/root/ : Here we used the “-r” to copy entire directory.

**scp -C** filename root@192.168.2.42:/root/ : The “-C” is used to compress and send data.

In fact, in this situation the timestamp will be the actual one when we make the copy that’s why we can use the parameter “-p” to preserve parameters.

**scp -vp** filenameroot@192.168.2.42:/root/ : To preserve parameters

**scp -l** 500 filename root@192.168.2.42:/root/ : To limit the bandwidth allocated for the copy.

# Linux Processes

**sleep** 300 **&**

output : [1] 2798 : This is the process ID.

**ps -aux | grep** 2798

In fact, each process has a specific state :

A) Running : Running or ready to run

B) Waiting : Process is waiting for an event or for a resource (Like waiting to be allocated a certain part of CPU or Ram.

C) Stopped : Received a stop signal

D) Orphaned : Process exists while children still running, thoses childrens are orphanes.

E) Zombie : It is a halted process. Still has a task\_struct data structure in task vector. .Here the process is dead but still found in Process table. To find these processes:

**ps -aux | grep** Z

**pstree** : Shows us the process tree containing father and children processes.

**ps -ef** : Show us not only processes like **ps -a** but also sleeping processes and all processes.

**ps -aux --sort = -pcpu,+pmem**: To sort process with utilization rate.

**ps -e -o** : Filter with , so we will see pid,uname,pcpu,pmem…

**jobs** : [1] Running

**fg %**1 : Here the ‘fg’ means foreground so it will kick the process having id ‘1’.

**kill -l** : Will show us all parameters that I can give to the kill command and each parameter is identified with a specific ID or number.

**kill** -9 3298 : Here we will kill this process. (-9) means kill process.

**pgrep -u** root ssh: See all executed commands with root processes about ssh.

Change process priroty : A normal user can only decrease a process priority but not increase it. The ‘-20’ is the highest priority value and the ‘20’ is the lowest.

**nice** -10 **sleep** 500 **&** : Will give a value of ‘-10’ as priority for this process.

# fdisk vs df -h

**df -h** shows us only mounted partitions however with **fdisk -l** we can see all partitions that exists on our drive (mounted or not)

# Disk partitioning

**sudo fdisk -l** : To see the hardrives or partitions.

♣ You can make only 4 partitions per hardrive (for Windows or Linux). In fact, we use the 4th partition for example as an extended one where inside of it we can use multiple partitions (locial partitions)

**sudo -s** : Login with root and go to home directory of present user.

**fdisk** /dev/sdb : Then enter ‘m’ for help then we will see multiple parameters to choose from and we will type ‘n’ to add a new partition. Then we will choose ‘p’ for primary then we will choose a partition number (1🡪4). Then we will choose the size of the 1st partition, we will put +5G then we will enter ‘wq’ to write and quit.

**partprobe** /dev/sdb : To update the partition table to this hardrive.

Now we will create or directory and add it (mount it) to a specific partition.

**mkdir** /part1

**mount** /dev/sdb1 /part1/ : It will not work and return an error, we need to add a file system to the partition.

**mkfs**.ext4 /dev/sdb1

**mount** /dev/sdb1 /part1/

♣ Difference between ‘df -h’ and ‘fdisk’ :

df -h : Displays all the mounted partitions.

fdisk -l : Displays all partitions that exist on your disk. Apparently, there are some partitions which exist but which are not mounted

**df -h** : To see all mounted partitions. Now here we will notice that the mount didn’t actually work, we should add something in /etc/fstab

**vi** /etc/fstab

Then we will add the following line :

/dev/sdb1 /part1 ext4 defaults 0 0

**mount -a**

**df -h** : Now we will see that part1 is mounted to sdb1

# Delete partitions

**vi** /etc/fstab

Then we will delete the following line :

/dev/sdb1 /part1 ext4 defaults 0 0

**unmount** /part1

**fdisk** /dev/sdb : Then ‘p’ to print partitions then ‘d’ to delete and choose the partition number, we will choose ‘1’ which refers to sdb1 then ‘wq’ to save and quit.

**partprobe** /dev/sdb

# Create LVM

We used standard partitions in the previous section. One of the disadvantages is that if we want to increase a partition size we need to stop the users using the partitions so the partition will be shut down until we increase it and finish and remount it again. LVM or Logical Volume Manager combines multiple hardrives so if we want to write a file instead of writing in on a single partition which will take specific time, with LVM the time will be split on the number of hardrives so the file will be written in equal manner on the hardrives and this volume group that combines multiple hardrives can be splitted on as many slices that we want. Each slice is called logical volume. So we will have logicalvolume1, logicalvolume2, etc..

In our example, we create 2 partitions. One of them is sdb1 and the other is sdc1. Sdb1 is the partition in sdb hdd and sdc1 is the partition in sdc hdd. In fact, these partitions are standard partitions. We need to convert them to LVM partitions. To do so :

**fdisk** /dev/sdb : Then ‘m’ then ‘t’ then ‘L’ to list all codes then ‘8e’ which is the code of Linux LVM. Now we converted the partition from standard to Linux LVM. Now we will pass to sdc.

**fdisk** /dev/sdc : Then ‘m’ then ‘t’ then ‘L’ to list all codes then ‘8e’ which is the code of Linux LVM. Now we converted the partition from standard to Linux LVM. Now we will pass to sdc.

**partprobe** /dev/sdb

**partprobe** /dev/sdc

Now we have to create the physical volumes inside of each partition

**pvcreate** /dev/sdb1

**pvcreate** /dev/sdc1

**pvs** : To see the created pv’s.

Now we will create the volume group

**vgcreate** VG0/dev/sdb1 /dev/sdc1

**vgs** : To see created vg’s.

**vgdisplay** VG0 : Will output further details on VG0.

**lvcreate -n** lv0 **-L** 4G VG0 : Here we created the logical volume containing the volume group. Now we will create another logical volume.

**lvcreate -n** lv1 **-L** 2G VG0

Now as we created pv’s then vg then lv’s we will create file systems and mount some things to them.

**mkfs**.ext4 /dev/VG0/lv0

**mkfs**.xfs /dev/VG0/lv1

**mkdir** /ext4part

**mkdir** /xfspart

**vi** /etc/fstab

We need to add the following lines :

/dev/VG0/lv0 /ext4part ext4 defaults 1 2

/dev/VG0/lv1 /xfspart xfs defaults 1 2

**mount -a**

**df -h**: To check

# Extend LVM

Here we will show how to extend filesystem without interrupting applications.

FREE SPACE

**LV0**  **Extend LV0** **VG0**

In fact, to be able to extend without interrupting user’s applications we need to resize the file system to this extended value.

**lvextend -L** +1G /dev/VG0/lv0

**resize2fs** /dev/VG0/lv0

♣ For xfs, the below command will change

**lvextend -L** +1G /dev/VG0/lv1

**xfs\_growfs** /dev/VG0/lv1

# Access Linux

● What is swap file system and how it works?

In fact, swap file is used to optimize the load on the physical memory. if we have some apps that are opened we may fill our memory capacities. In order to optimize that we will remove the apps that are inactive for a certain time and put them on the hardisk so we will always have the info but in another location.

Swap memory = Ram X2

Now we will create the partition as swap.

**fdisk** /dev/sdc : Then ‘m’ then ‘n’ then ‘p’ then ‘+2G’ then ‘t’ then ‘1’ then ’82’ which will convert the file system of sdc1 partition from Linux to Linux swap/Solaris. Then ‘wq’ to save.

**partprobe** /dev/sdc

**mkswap** /dev/sdc1

**vi** /etc/fstab

Then add the following :

/dev/sdc1 swap swap defaults 0 0

**swapon -a**

**swapoff -a** : To turn off swap

**swapon -s** : To see the swap partition.

# Create users

There are 3 types of users :

* Super User (Root/Administrator)
* SystemUser : UID : 1 999 (Appear when downloading a package service for example)
* Normal Users : UID : 1000 65 000

Useradd command will create by default multiple properties like Home directory, login shell details, by default there is also a group that is created with the same name of the user and there are some files that are created also like .bash…

And each time we add a user, we will have some files that will be modified like /etc/passwd and /etc/shadow (Contain the password that we added for this user but here it will be encrypted, and /etc/gshadow file will be modified if the group for which we added a user has a password allocated to it (By default, the file is not modified). /etc/skel

♣ A user can only have 1 primary group but can have multiple secondary groups.

**sudo -s** : To pass to the administrator

**cat** /etc/passwd : We will some entries that contains all users (We will see system users that are created for each and every service)

Now we will explain in details each field of a user (normal user) entry.

**cat** /etc/passwd **| grep** aravi

Output : aravi : x : 1000 : 1000 : Ravi Kumor :/home/aravi :/bin/bash

The field ‘aravi’ is the login name user used to log to the system. The ‘x’ means that password is encrypted and added to /etc/shadow file. The first ‘1000’ field is the UID (User Id). The other ‘1000’ field is the Primary Group Id. After that if there is some secondary groups, they will be listed here. We see ‘Rami Kumor’ after that which is the description of the user. ‘/home/aravi’ is the home directory for this user so when a user logs on he will be transferred directly to this directory. The last field is the shell used which is bash in this scenario.

There are some default config that has been allocated to usedadd command and we can see it with the following command :

**cat** /etc/default/useradd

We have also a file containing user group details and login expiry password age etc… in the following path :

**cat** /etc/login.defs

When we create a new user, by default a group with the same name will be created in /etc/group. The user is created in the home directory. By default, there will be some files in the user directory that are copied from /etc/skel directory.

Now, we create a user with some customization :

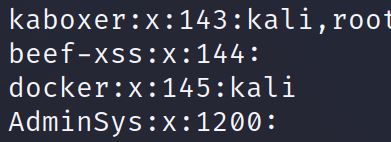
**useradd -u** 2000 **-g** 2000 **-G** aravi,test2 **-c** “Some user” **-s** /bin/bash **-d** /opt/rhesa **-e** 2018-08-28

**-p** elie@123 rhesa

‘-u’ defines UID. ‘-g’ defines primary group id. ‘-G’ defines secondary groups. ‘-c’ is for comments. ‘-s’ is for shell. ‘-d’ designes the home path for user. ‘-e’ is the user expiry date, after that he will be deleted. ‘-p’ is the password chosen and the last field is the name of user.

# Understand group file

The /etc/group file is divided with the following values :



For example, we see in the first entry the name of the group which is ‘kaboxer’ then the second value is the password. The third value is the Group ID and the fourth value is normally for the users that are existant in a supplementary group so for example if I create a new user and append him to a primary group this user will not be shown in this group in /etc/group

# Modify user properties

Here we used usermod to modify

**usermod -l** ravi test : Will change the login name from test to ravi.

Now we will add the user to same groups

**usermod -aG** aravi,root ravi : ‘-a’ means append so we are appending those groups to ‘ravi’.

Now we will change the primary group of ‘ravi’ user :

**usermod -g** aravi ravi

Now we will log the user, in that way he won’t need to enter the password when he tries to log.

**usermod -L** ravi : Locks a user

**usermod -U** ravi : Unlocks a user

In fact, when a user is logged, we can see in /etc/shadow a ‘!’ that will appear before the encrypted password.

**usermod -f** ravi : Sets the user inactive after password expiration :

To change a user’s password from another user, only the super user can do that so pass to root :

**passwd** aravi

**userdel** aravi : To delete a user (Here home directory will not be deleted)

To delete all user’s data :

**userdel -r** aravi

To delete a user from a group :

**gpasswd -d** aravi finance

To delete a group :

**groupdel** finance

To change group name :

**groupmod -n** newname oldname

Set expiry date for a user :

**usermod -e** 2018-08-22 ravi

To see password parameters, use the following command :

**chage -l** ravi

Here we will see the password expiry date

**chage -m** 0 **-M** 90 **-W** 10 **-E** 2018-09-28 ravi

The ‘-m’ is the minimum number of days between password change. The ‘-M’ is the maximum number of days. The ‘-W’ provide warning before 10 days of password expiration. The ‘-E’ is the expiry date for the password.

# Modify group properties

In this scenario, we will create a group, 3 users and we will add the users to the group.

**groupadd** finusers

**useradd** user1

**useradd** user2

**useradd** user3

**usermod -G** finusers user1

**usermod -G** finusers user2

**usermod -G** finusers user3

We can protect our group using a password, in that case if another administrator wants to modify our group he must know the password.

**groupmod -p** testfinusers

Now we will try to insert a new user in that group by another administrator. In fact, it will return permission denied and to fix that we will make the other administrator as the owner of the group.

**gpasswd -A** user4 finusers : The ‘-A’ means administrators.

We can add multiple users as admins at same time.

**gpasswd -M** user1,user2,user3 finusers

To remove a specific user as being the admin of group :

**gpasswd -d** user4 finusers

To check id and to what groups a user belongs to :

**id** user1

To modify group properties :

**groupmod -g** 1015 finusers : Will change group Id to 1015

**groupmod -n** engusers finusers : Change name of the group

To delete a user can go to /home/directory and execute the command :

**userdel** user1

But here we will still see the folder of user in /home/ because with this command, data will not be erased.

To delete all user data :

**userdel -r -f** user1

# Permissions

The defaullt permissions are :

File : 644

Directory : 755

**ls -ltr**

Here we will see the permissions for directories or files. After that we have a field which is a count and show us how many files are there in a directory, then we will see the owner of the user and the group which have permissions and then we have the size field, creation date and name of directory.

Output : d rwx r-x r-x 2 aravi aravi 6 Jun 2 16:30 Videos

If the first field ‘d’ was ‘-‘ it means that it’s a file. Here it is a directory and then we see the permissions for the user, the group and others successively.

♣ If you do not have a read permission on a directory then you can not go inside it.

When using the numerical value with chmod, always enter the 3 digits. If you do chmod 7 file for example, the permissions of user and group will be removed and others will take the value 7 which refers to rwx so output will be --- --- rwx.

In fact, when we create a file for example with the touch command, the file will get some permissions by default. Actually, these permissions are received because of the umask command.

**umask**

Output : 0002

The default value of a file is 666 and the default value of a directory is 777. We will now explain how and why the umask value is 0002. The output of ls -ltr for the file that we created with touch by default gets the following permissions : -rw-rw-r-- which is equivalent to 664. And we said that the default value of a file is 666 s the calculation will confirm the umask value : 666-664 = 002

If we want to change the default permission giveb for a file after creation, we need to modify our umask value. For example if we want the following values : rw-rw-rw- We will make the calculation so see what value of umask we should use : 666-666=000. So umask will be : 0000

**umask** 000 : To change the umask

Here we modified the umask value on the user. If we want to change that on a global scale we need to go to as user:

**vim** /etc/profile

In fact, we forgot to explain the role of the first digit here. Here there can be 3 values but each value has a differebt meaning than normal 1,2 and 4

1 → Sticky

2 → SUID

4 → SGID

● Sticky is used to assign the files permissions or directories permissions so that other users except the owner can not delete the files accidentely to prevent some problems.

● SUID works as following : I am root user but I do want to give the access to users to execute the files but what happen here is that I will give the access to them by assigning the SUID to that file. So if I assign a special SUID, whenever a user is executing that files, the file will be executed as the ownership of that file.

● SGID : If we created a directory and 3 users need to access it and do some changes then we need to assign the value 4 so the value of SGID. In that way the users will be in the group of the directory.

# Sudo Access Config

In this part, we will learn how to provide for a user root access without giving them root password so we could execute root commands. In fact, sudo means subtitue user do or super user do.

**cat** /etc/sudoers

*ou*

**visudo** : Now we will give permissions to a particular user

**cat** /dev/null **>** /etc/sudoers : Here we made the values of sudoers completely null. So now when we visudo we will have nothing inside of it.

**visudo** : The file is empty so we will add a user

Host\_Alias SERVERS = localhost, server (Here the server is the @server in root@server)

#User Alias (If I want to make any group)

User\_Alias ADMINS = aravi, user1, user2 (Here the users becomes the ADMINS group)

#Command Alias

Cmnd\_Alias CUSTOM = /sbin/mount, /sbin/fdisk, /sbin/parted

Cmnd\_Alias ADMINSTRATORS **=** /sbin/\*

Cmnd\_Alias CHMOD = /bin/chmod

Cmnd\_Alias CHOWN = /bin/chown

Now we will define some policies so for example if a user typed a wrong command then we should get email alert and other things..

#Defaults

Defaults syslog = auth, insults, syslog\_goodpri=alert

Defaults logfile = /var/log/sudo.log

Defaults timestamp\_timeout = 0, log\_year, tty\_tickets

Defaults mailto = [aravikumar@gmail.com](mailto:aravikumar@gmail.com), mail\_always, mail\_no\_user, mail\_badpass

#Allow users to run commands

root All = (ALL) ALL

aravi All = NOPASSWDl ALL (Here it means no password is needed when running sudo commands so he will be able to execute all commands)

#Group names

%engineers ALL = NOPASSWD: CUSTOM (Here we gave permission to the group ‘engineers’ to all commands in the alias CUSTOM.

%admins ALL = NOPASSWD: ADMINISTRATORS, !CHMOD, !CHOWN (This group can execute all commands defined in the ADMINISTRATORS group except CHMOD and CHOWN,

Now if i switch to aravi user :

**su –** aravi

**sudo cat** /etc/sudoers : The command will work without demanding a password of root to enter.

♣ Instead of each time we create a user we modify /sudoers files, we can directly add the user to the group that we defined in the /sudoers having the right permissions.

# Networking IP

In this section, we will explore the NMCLI tool (Network manager command line interface)

**nmcli device status** : Here we will see the interfaces with its status.

We can create multiple profiles having different configuration for a specific interface so that if I change location and network I can directly activate the corresponding profile. To see the profile used for a connection :

**nmcli connection show**

output : Name UUID TYPE DEVICE

ens33 ffgbe-e23dd.. 802.3\_ethernet ens33

**sudo nmcli connection add type** ethernet **conn-name** home **ifname** ens33 : Now if we execute the command ‘nmcli connection show’ we will see the same output as the ligne before but the field ‘DEVICE’ will be empty because this interface is already used for another connexion.

**nmcli general status** : To see the connectivity running

**nmcli general logging** : To see different protocols, etc…

**nmcli device status** : To see NFC cards

**nmcli device show** ens33 : Show us more details about interface like MAC etc…

**nmcli connection delete** home’

- Now we will configure a connection profile so we will assign an IP Adress, etc…

**nmcli connection modify** home **ipv4.addresses** 192.168.2.141/24 **ipv4.gateway** 192.168.2.2 **ipv4.dns** 192.168.2.2 **+ipv4.dns** 4.4.4.4 **connection.autoconnect** yes **ipv4.method** manual : Here, manual means static ip and do not use DHCP.

**cat** /etc/sysconfig/network-scripts/ifcfg-home : Here we will see all info config about this profile. There are some parameters that will be defined by default.

**nmcli connection up** home

**nmcli connection show --active** : It will show us only active profiles (active connection)

**nmcli device monitor** : To see packets dropping, etc… on specific interface.

- I can give the permissions for a specific user not root to be able to shut down a connection

**nmcli connection modify** aravi **connection.permissions user:**aravi

**nmcli connection down** techarkit : Work from user without root priveleges.

- Now we will learn how we can do all these things without having a lot of commands to write we will use the ‘nmtui’ tool which is a text based user interface.

**nmtui edit** home

**nmtui -hostname** : To directly pass to the hostname in the interface

**nmtui -connect** : To directly pass to the connection section for profiles

# Firewall config

**firewall-config** : It opens a firewall config window. When we launch this windows we will see that there is 2 types of configuration : Runtime config and Permanent config. If we do a runtime config, when we reboot the system, the config will go off.

If I want to use the command line :

**firewall-cmd --help** : To see all rules

**systemctl status** firewall : To see if It is activated.

**firewall-cmd --get-default-zone**

Output : public

**firewall-cmd --set-default-zone =** public

**firewall-cmd --get-active-zones**

**firewall-cmd --zone =** public **--list-interfaces** : To see interfaces added to the public zone

**firewall-cmd --add-interface =** eth0 **--zone =** home: To add interface to a zone

**firewall-cmd --get-services** : To see all services on firewall

We can also make firewall changes by accessing a xml file and modify it :

**cat** /etc/firewalld/zones/public.xml

# SELinux context

Security enhanced Linux is used in addition to a firewall to insure more secured environement. For example on the edge level of our machine we can secure with firewall and on system level we can secure with SELinux.

It comes with offering security on multiple levels :

1) Port level security

2) Service level security

3) File level security

In contrast, firewall only provides security on port level. However, with SELinux, le’s say we allocated the port 80 we can still secure a service from this source.

SELinux has 3 modes :

- Enforcing (Enabled)

- Permissive (Not disabled)

- Disabled

**cat** /etc/selinux/config : We will see the different modes and which is actually used. In fact, permissive just saves the logs and does not restrict like enforcing.

**ls -lZ** : To see context of SELinux on a file for example

Output : drwxr-xr-x aravi aravi unconfined\_u : object\_r : user\_home\_t : s0

unconfined\_u : Mean it is an SELinux user based context.

object\_r : Role based access control (RBAC)

user\_home\_t : Desktop (Type of context)

s0 : Which level of SELinux is provided.

**cat** /etc/selinux/targeted/setrans.conf : To see system levels

**sudo sestauts** : To see status of SELinux

**getenforce** : Will return enforcing if it is the case.

**setenforce** 0

**getenforce**

Output : Permissive

**setenforce 1**

**getenforce**

Output : Enforcing

**ls -ldZ** /root/

Output : drwxr-xr-x aravi aravi system\_u : object\_r : admin\_home\_t : s0 /root/

**ls -ldZ** /usr/

Output : drwxr-xr-x aravi aravi system\_u : object\_r : usr\_t : s0 /usr/

**ls -ldZ** /var/

Output : drwxr-xr-x aravi aravi system\_u : object\_r : var\_t : s0 /root/

So here, each directory has its type pf context. So that the service can only access based on the context so if it’s matching.

Now we will take an example with the service httpd

**systemctl start** httpd

**ls -ldZ** /var/www/html/

Output : drwxr-xr-x aravi aravi system\_u : object\_r : httpd\_sys\_content\_t : s0 /var/www/html/

Here, we see that the context is httpd so only a service matching this context will work. So a connection from httpd to a file in this directory will work because the SELinux context is defined as httpd context.

To change the context type :

**chcon** unconfined\_u : object\_r : etc\_t : s0 index.html

**ls -lZ** index.html

Output : drwxr-xr-x root root unconfined\_u: object\_r : etc\_t : s0 index.html

Now, the httpd will not work because the service is different than the context type of the directory. If we set the enforce to 0 he will not put restrictions but will log because it is the mode permissive.

If I want to disable SELinux, I must pass to the file

**vim** /etc/selinux/config

SELinux = disabled

♣ We need to reboot the machine

# YUM Repository

(1) AppStream : Used for installing the applications

(2) BaseOS : Used for installing Base OS required packages

(3) YUM Repository : Stores both of AppStream and BaseOS repositories

**yum install** httpd

# NTP Server and Client

Send timestamp

Receive timestamp(UDP)

Client Server

It is mandatory that the time is synchronized between server and client

● Commands to execute on NTP Server side :

**dnf install chrony**

**systemctl enable chronyd**

**vi** /etc/chrony.conf

**allow** 192.168.45.0/24

**systemctl restart chronyd**

**firewall-cmd --permanent --add-service =** ntp

**firewall-cmd -reload**

**ntpupdate <**NTP SERVER ADDRESS**>**

● Commands to execute on NTP Client side :

**dnf install chrony**

**systemctl enable chronyd**

**Server <**NTP SERVER ADDRESS**>**

**vi** /etc/chrony.conf

**server** 192.168.175.128

**systemctl restart chronyd**

# Scheduling jobs

**cat** /etc/crontab : To see the entries description

Now we will make an example of doing a job every 1 minute (A script was written to be executed)

**crontab -e** : Will open a file to edit

\* \* \* \* \* : This means it will be executed everytime (Every minute, houre, everyday..)

\* \* \* \* \* **sh** /root/binbash/quotes.sh

So here we are saying to execute this shell everytime. Now to check if that thing worked, we need to check the log file where we can see the last time the shell was executed.

**cat** /var/log/cron **| grep** quotes

Another example :

● If I want to execute it every 5 minutes :

\*/5 \* \* \* \* **sh** /root/binbash/quotes.sh

● If I want to execute it between 2 values I use (-)

\*/5 21-23 \* \* \* **sh** /root/binbash/quotes.sh

So here we are executing every 5 mintes between hour 21 and 23

● If I want to execute it between at values I use (-)

\*/5 21,23 \* \* \* **sh** /root/binbash/quotes.sh

So here we are executing every 5 mintes between at hour 21 and 23

**crontab -l** : To see the entries I added

# Introduction to bash scripting

- When we executes a script, we are not launching a new process but it is the role of the interpreter which is ‘bash’ to execute the script. There are 3 ways to execute a bash script

**./**script.sh

**bash** script.sh

**sh** script.sh

- In a script the following operators means the following :

**$#** : Number of arguments

**$0** : Name of actual shell script

**$1** : First argument passed when calling the script

**$2** : Second argument passed when calling the script