CS695 – Lab 1

Introduction to Kali Linux and Basic CLI Commands

Ryan Christopher

Table of Contents:

Part 1 – Set up VirtualBox and Explore Kali

1) What is a virtual machine? Why use a virtual machine?

A virtual machine is an application that functions as a computer. This allows us to run another operating system that does not have access to the data or resources outside of the virtual machine, allowing us to have a “sandbox” like computer to use.

2) What is a snapshot of a virtual machine?

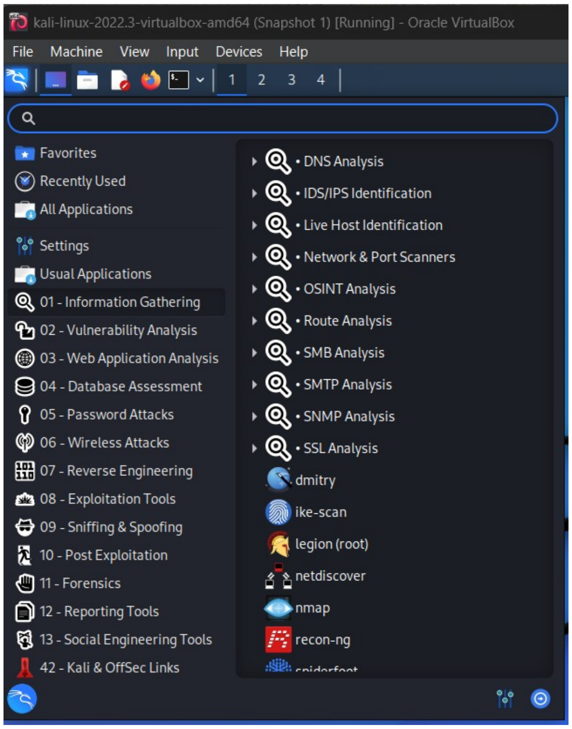
A snapshot of a virtual machine is a saved state at the specific point in time you take the snapshot. Similar to a save file of a video game, it allows us to “go backwards” to that point in the event we want to undo a large number of actions or changes that occurred in the virtual machine.

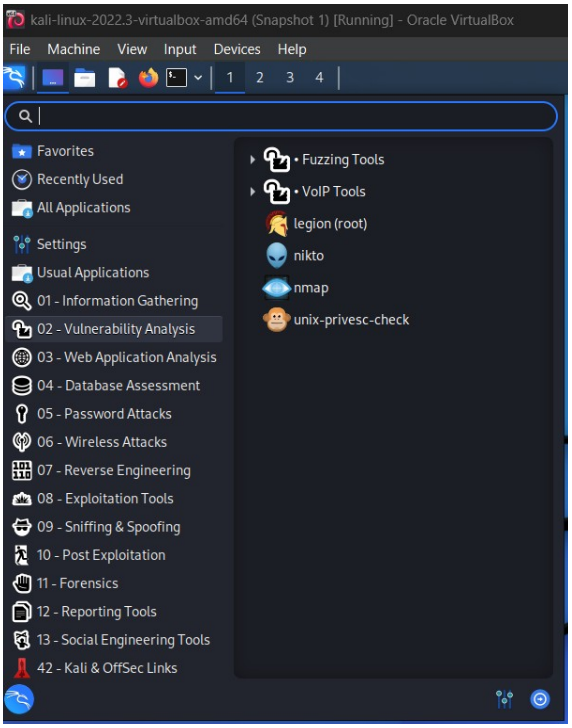
3) What is Kali Linux? (What distribution, based OS, open source?). Why use it instead of a basic Linux distribution?

Kali Linux is an open source Debian-based Linux distribution founded and maintained by the cybersecurity company Offensive Security. Using Kali Linux instead of a basic Linux distribution is beneficial for this class since it comes pre-installed with applications used for cybersecurity, specifically penetration testing.

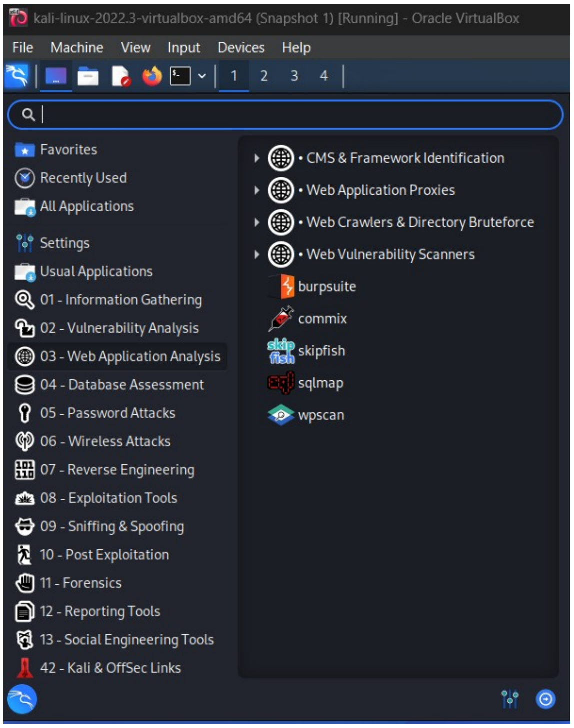
4) Tools in the Kali menu:

Nmap, standing for “Network Mapper”, is a tool for scanning IP addresses and ports of a network. It can be used for tasks like finding which devices are connected to a network, the applications running on a network, and determining what types of firewalls a network is using. It is found in the “Information Gathering” and “Vulnerability Analysis” categories.

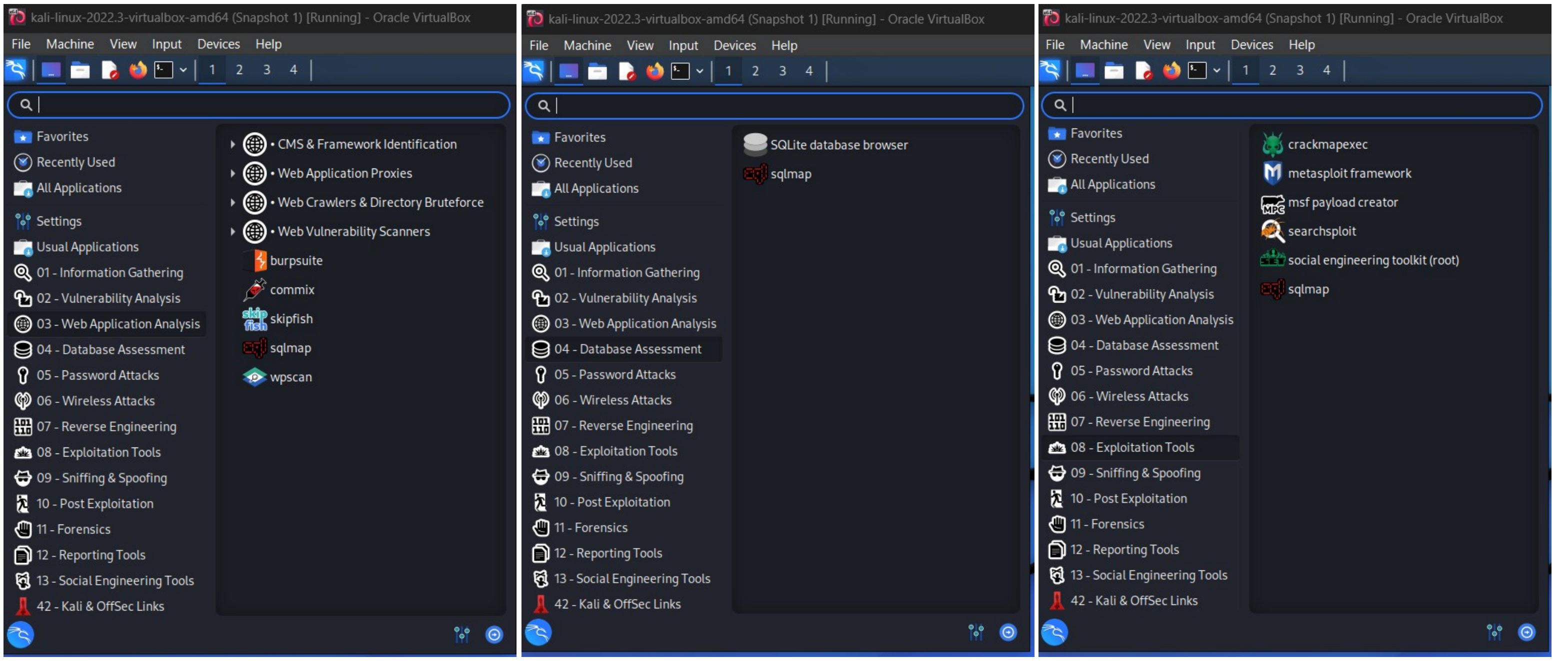




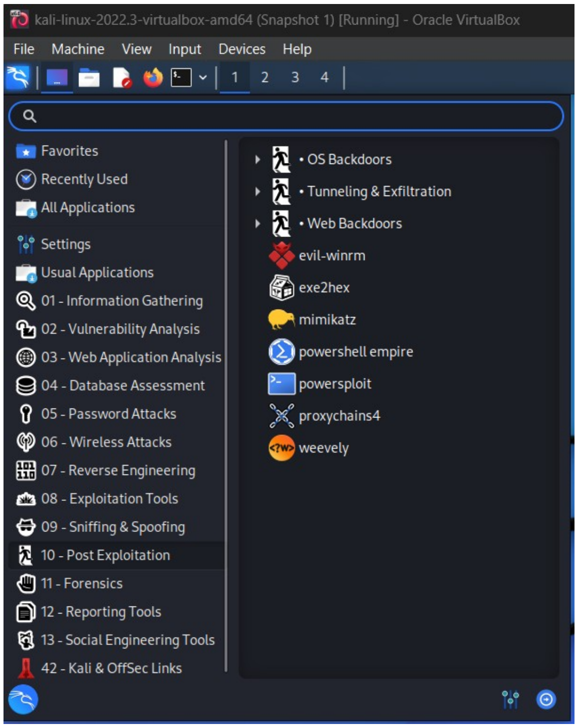
Burpsuite is a tool used for penetration testing web applications. It can be used to find vulnerabilities web apps through tactics like manipulated requests. It is found in the “Web Application Analysis” category.


kali-linux-2022.3-virtualbox-amd64 (Snapshot 1) [Running] - Oracle VirtualBox
File Machine View Input Devices Help
1 2 3 4
Favorites
Recently Used
All Applications
Settings
Usual Applications
01 - Information Gathering
02 - Vulnerability Analysis
03 - Web Application Analysis
04 - Database Assessment
05 - Password Attacks
06 - Wireless Attacks
07 - Reverse Engineering
08 - Exploitation Tools
09 - Sniffing & Spoofing
10 - Post Exploitation
11 - Forensics
12 - Reporting Tools
13 - Social Engineering Tools
42 - Kali & OffSec Links
• CMS & Framework Identification
• Web Application Proxies
• Web Crawlers & Directory Bruteforce
• Web Vulnerability Scanners
burpsuite
commix
skipfish
sqlmap
wpscan

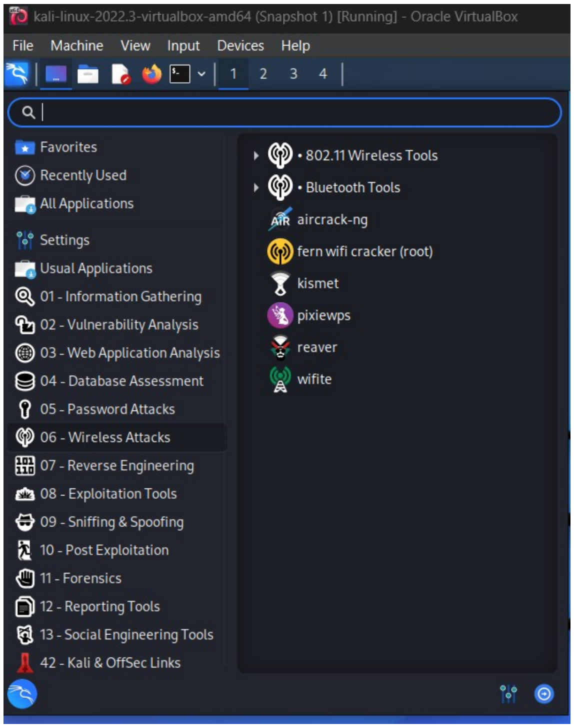
SQLMap is a tool used to find vulnerabilities in databases of web applications, often through SQL injection. It is found in the “Web Application Analysis”, “Database Assessment”, and “Exploitation Tools” categories.


kali-linux-2022.3-virtualbox-amd64 (Snapshot 1) [Running] - Oracle VirtualBox
03 - Web Application Analysis
• CMS & Framework Identification
• Web Application Proxies
• Web Crawlers & Directory Bruteforce
• Web Vulnerability Scanners
burpsuite
commix
skipfish
sqlmap
wpscan
04 - Database Assessment
SQLite database browser
sqlmap
08 - Exploitation Tools
crackmapexec
metasploit framework
msf payload creator
searchsploit
social engineering toolkit (root)
sqlmap

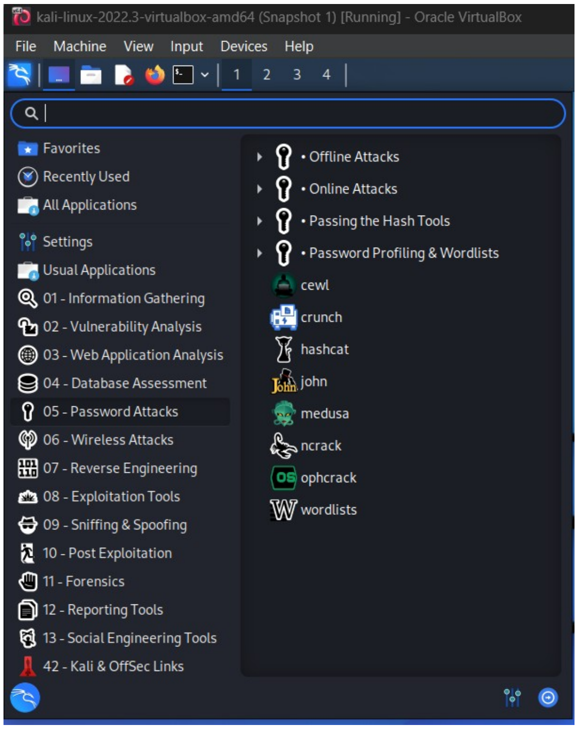
Mimikatz is a tool for extracting data like passwords and login information specifically from devices using Windows as their operating system. It is found in the “Post Exploitation” category.



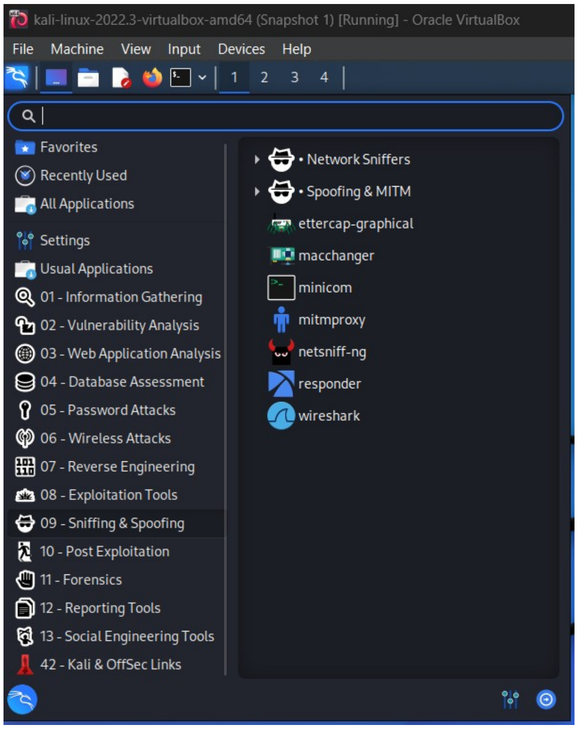
Aircrack-ng is a tool that is used to assess WiFi network security through methods like monitoring, attacking, testing, and cracking on a wireless network. It is found in the “Wireless Attacks” category.



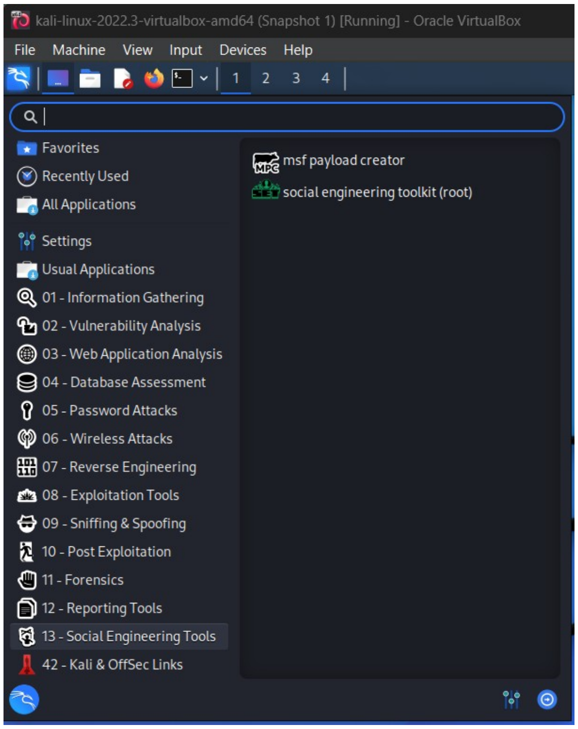
John, also known as **John the Ripper**, is a tool used for password cracking that can run on multiple operating systems such as Windows, macOS, and Unix. It is found in the "Password Attacks" category.



Wireshark is a network packet analyzer tool. It can be used to measure the traffic of a network in real-time. It is found in the "Sniffing & Spoofing" category.

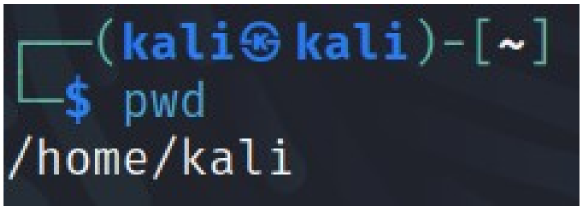


SET (Social Engineering Toolkit) is a tool used for penetration testing through Social-Engineering tactics like spear-phishing, QR codes, and websites. It is found in the "Social Engineering Tools" category.



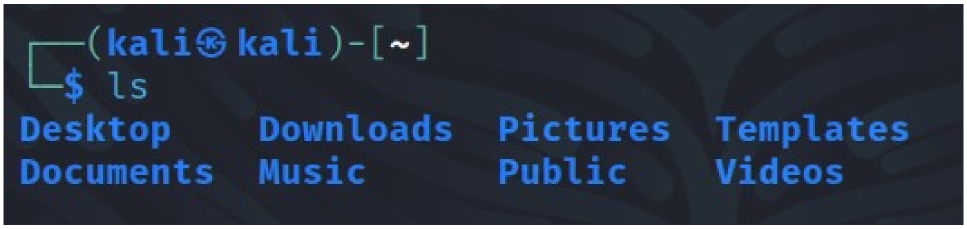
Part 2 – Basic Linux Commands

1) *pwd*



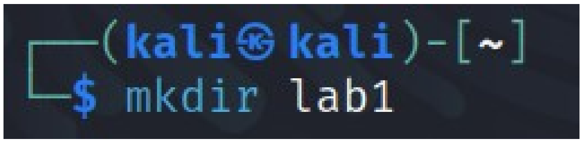
```
┌──(kali㉿kali)-[~]
└─$ pwd
/home/kali
```

2) *ls*



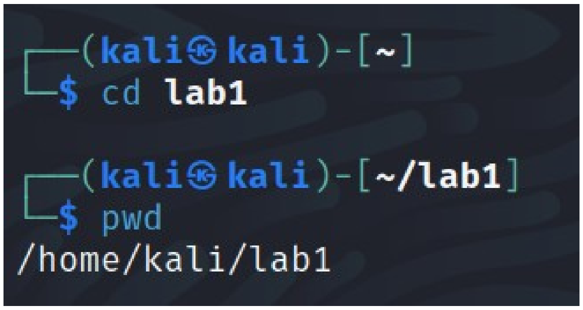
```
┌──(kali㉿kali)-[~]
└─$ ls
Desktop    Downloads  Pictures  Templates
Documents  Music      Public    Videos
```

3) *mdkir*



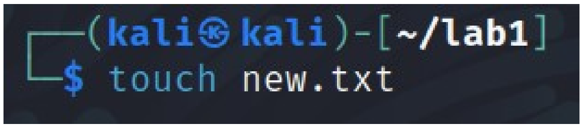
```
┌──(kali㉿kali)-[~]
└─$ mkdir lab1
```

4) *cd*



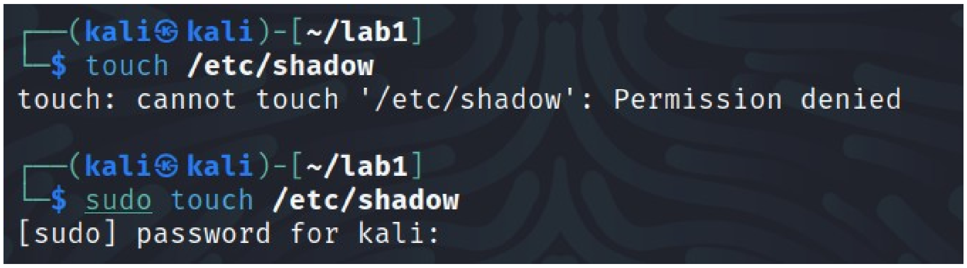
```
┌──(kali㉿kali)-[~]
└─$ cd lab1

┌──(kali㉿kali)-[~/lab1]
└─$ pwd
/home/kali/lab1
```

5) *touch*



```
┌──(kali㉿kali)-[~/lab1]
└─$ touch new.txt
```

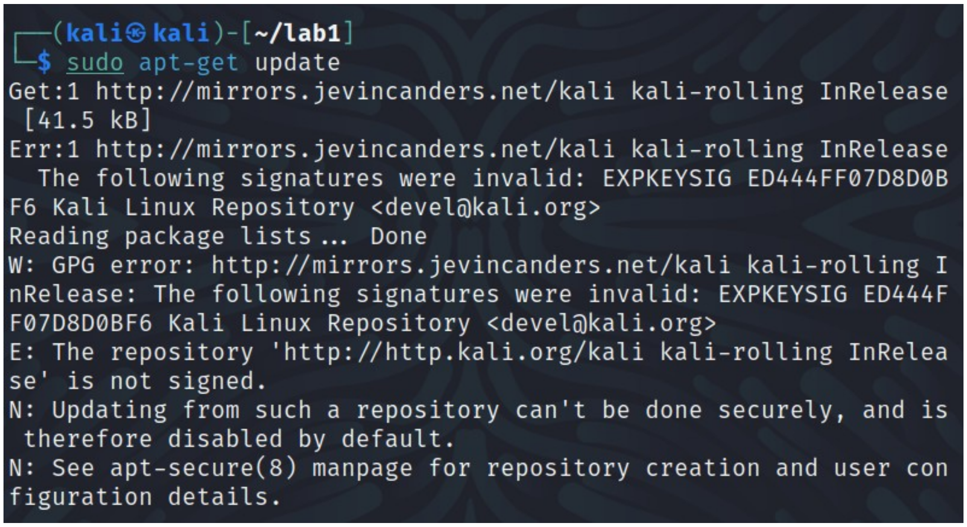
6) *sudo*



```
┌──(kali㉿kali)-[~/lab1]
└─$ touch /etc/shadow
touch: cannot touch '/etc/shadow': Permission denied

┌──(kali㉿kali)-[~/lab1]
└─$ sudo touch /etc/shadow
[sudo] password for kali:
```

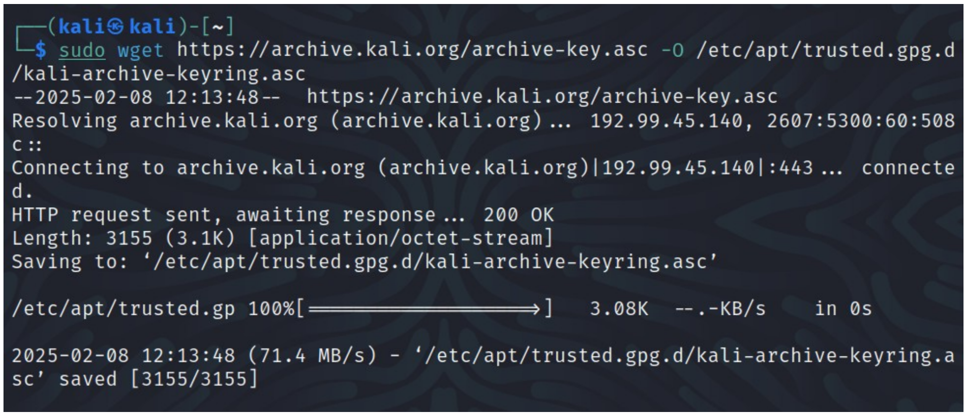
7) *apt-get*

This step I encountered a mild roadblock as the “sudo apt-get update” command returned an error each time I tried running it. I eventually ended up on the Kali Linux 2018.1 Release page which included a solution to the GPG error. Unfortunately, I could not run that command as it required me to be the root user, however I found this StackExchange post that included a command to retrieve the latest key.

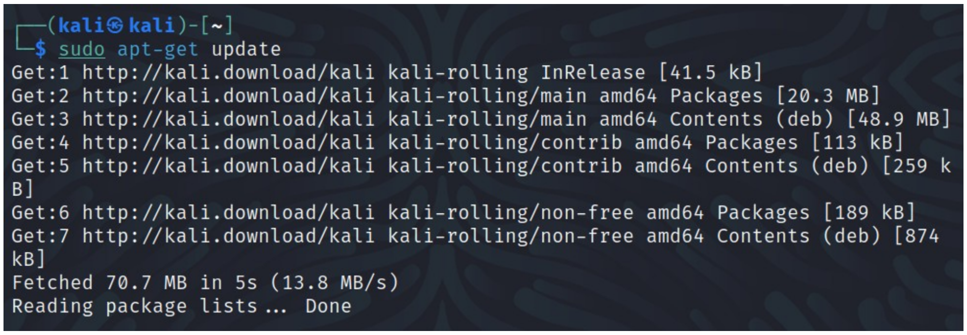


```
┌──(kali㉿kali)-[~/lab1]
└─$ sudo apt-get update
Get:1 http://mirrors.jevincanders.net/kali kali-rolling InRelease
 [41.5 kB]
Err:1 http://mirrors.jevincanders.net/kali kali-rolling InRelease
  The following signatures were invalid: EXPKEYSIG ED444FF07D8D0B
F6 Kali Linux Repository <devel@kali.org>
Reading package lists... Done
W: GPG error: http://mirrors.jevincanders.net/kali kali-rolling I
nRelease: The following signatures were invalid: EXPKEYSIG ED444F
F07D8D0BF6 Kali Linux Repository <devel@kali.org>
E: The repository 'http://http.kali.org/kali kali-rolling InRelea
se' is not signed.
N: Updating from such a repository can't be done securely, and is
 therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user con
figuration details.
```

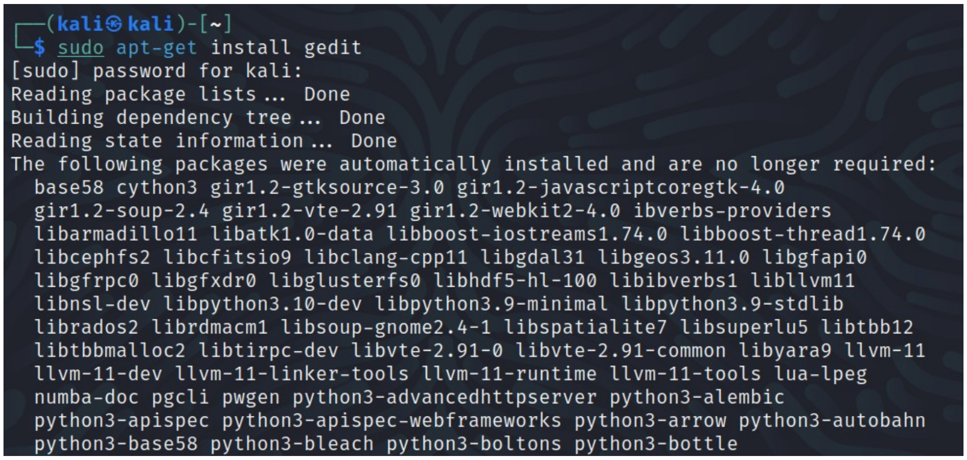
After running the command from the above post, I was able to use the “sudo apt get-update” command without any errors.



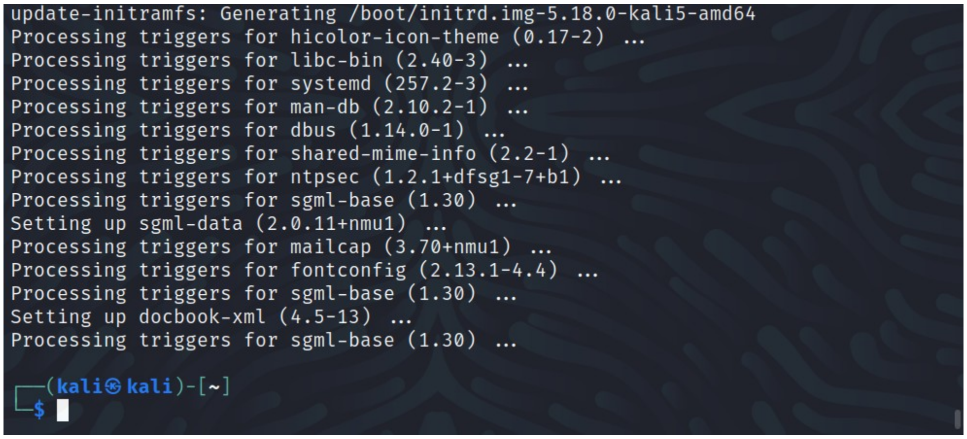
```
┌──(kali㉿kali)-[~]
└─$ sudo wget https://archive.kali.org/archive-key.asc -O /etc/apt/trusted.gpg.d
/kali-archive-keyring.asc
--2025-02-08 12:13:48--  https://archive.kali.org/archive-key.asc
Resolving archive.kali.org (archive.kali.org)... 192.99.45.140, 2607:5300:60:508
c::
Connecting to archive.kali.org (archive.kali.org)|192.99.45.140|:443... connecte
d.
HTTP request sent, awaiting response... 200 OK
Length: 3155 (3.1K) [application/octet-stream]
Saving to: ‘/etc/apt/trusted.gpg.d/kali-archive-keyring.asc’

/etc/apt/trusted.gp 100%[===================>]   3.08K  --.-KB/s    in 0s

2025-02-08 12:13:48 (71.4 MB/s) - ‘/etc/apt/trusted.gpg.d/kali-archive-keyring.a
sc’ saved [3155/3155]
```



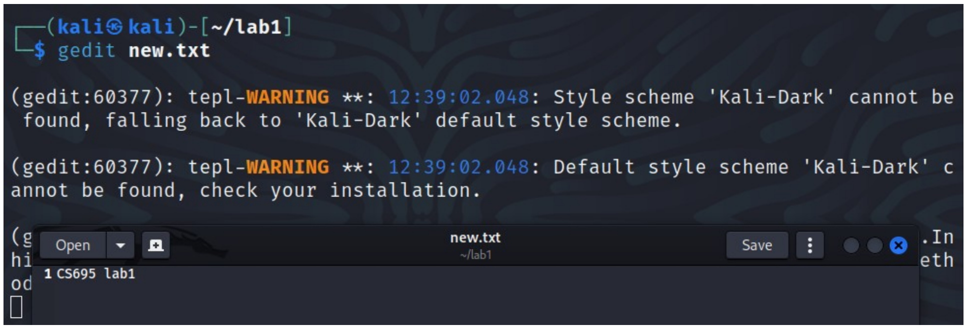
```
┌──(kali㉿kali)-[~]
└─$ sudo apt-get update
Get:1 http://kali.download/kali kali-rolling InRelease [41.5 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [20.3 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [48.9 MB]
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [113 kB]
Get:5 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [259 k
B]
Get:6 http://kali.download/kali kali-rolling/non-free amd64 Packages [189 kB]
Get:7 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb) [874
kB]
Fetched 70.7 MB in 5s (13.8 MB/s)
Reading package lists... Done
```



```
┌──(kali㉿kali)-[~]
└─$ sudo apt-get install gedit
[sudo] password for kali:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  base58 cython3 gir1.2-gtksource-3.0 gir1.2-javascriptcoregtk-4.0
  gir1.2-soup-2.4 gir1.2-vte-2.91 gir1.2-webkit2-4.0 ibverbs-providers
  libarmadillo11 libatk1.0-data libboost-iostreams1.74.0 libboost-thread1.74.0
  libcephfs2 libcfitsio9 libclang-cpp11 libgdal31 libgeos3.11.0 libgfapi0
  libgfrpc0 libgfxdr0 libglusterfs0 libhdf5-hl-100 libibverbs1 libllvm11
  libnsl-dev libpython3.10-dev libpython3.9-minimal libpython3.9-stdlib
  librados2 librdmacm1 libsoup-gnome2.4-1 libspatialite7 libsuperlu5 libtbb12
  libtbbmalloc2 libtirpc-dev libvte-2.91-0 libvte-2.91-common libyara9 llvm-11
  llvm-11-dev llvm-11-linker-tools llvm-11-runtime llvm-11-tools lua-lpeg
  numba-doc pgcli pwgen python3-advancedhttpserver python3-alembic
  python3-apispec python3-apispec-webframeworks python3-arrow python3-autobahn
  python3-base58 python3-bleach python3-boltons python3-bottle
```



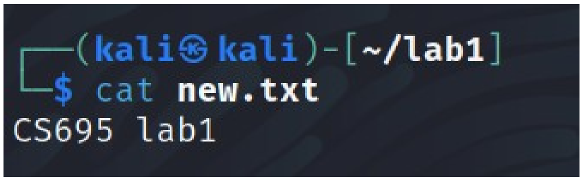
```
update-initramfs: Generating /boot/initrd.img-5.18.0-kali5-amd64
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for libc-bin (2.40-3) ...
Processing triggers for systemd (257.2-3) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for dbus (1.14.0-1) ...
Processing triggers for shared-mime-info (2.2-1) ...
Processing triggers for ntpsec (1.2.1+dfsg1-7+b1) ...
Processing triggers for sgml-base (1.30) ...
Setting up sgml-data (2.0.11+nmu1) ...
Processing triggers for mailcap (3.70+nmu1) ...
Processing triggers for fontconfig (2.13.1-4.4) ...
Processing triggers for sgml-base (1.30) ...
Setting up docbook-xml (4.5-13) ...
Processing triggers for sgml-base (1.30) ...

┌──(kali㉿kali)-[~]
└─$ 
```



```
┌──(kali㉿kali)-[~/lab1]
└─$ gedit new.txt

(gedit:60377): tepl-WARNING **: 12:39:02.048: Style scheme 'Kali-Dark' cannot be
 found, falling back to 'Kali-Dark' default style scheme.

(gedit:60377): tepl-WARNING **: 12:39:02.048: Default style scheme 'Kali-Dark' c
annot be found, check your installation.

(g                                                                            .In
hi                                                                            eth
od
```

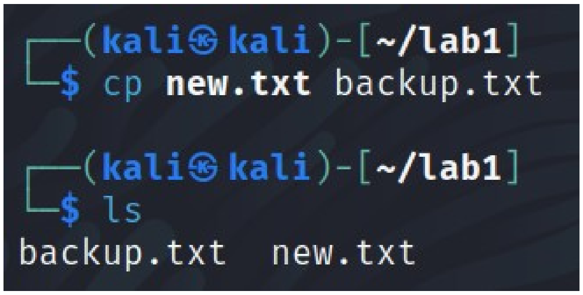
```
CS695 lab1
```

8) *cat*



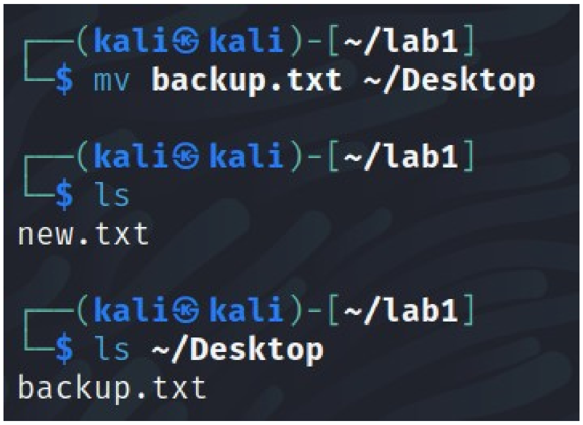
```
┌──(kali㉿kali)-[~/lab1]
└─$ cat new.txt
CS695 lab1
```

9) *cp*



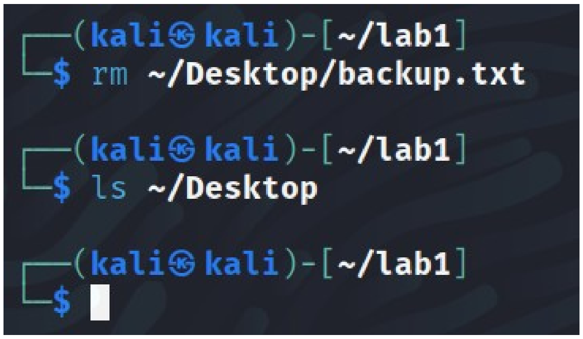
```
┌──(kali㉿kali)-[~/lab1]
└─$ cp new.txt backup.txt

┌──(kali㉿kali)-[~/lab1]
└─$ ls
backup.txt  new.txt
```

10) *mv*



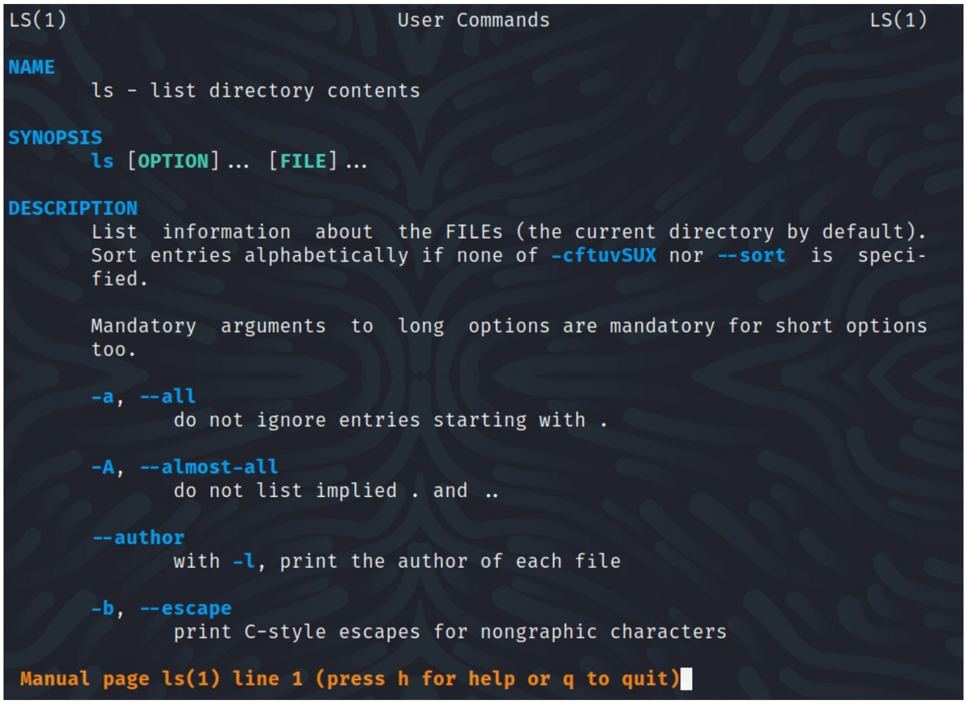
```
┌──(kali㉿kali)-[~/lab1]
└─$ mv backup.txt ~/Desktop

┌──(kali㉿kali)-[~/lab1]
└─$ ls
new.txt

┌──(kali㉿kali)-[~/lab1]
└─$ ls ~/Desktop
backup.txt
```

11) *rm*

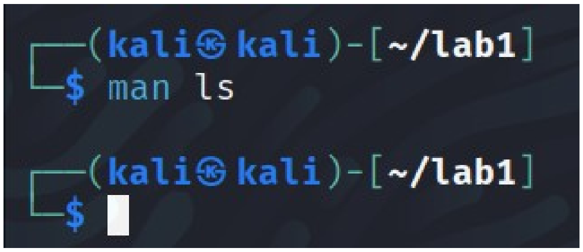


```
┌──(kali㉿kali)-[~/lab1]
└─$ rm ~/Desktop/backup.txt

┌──(kali㉿kali)-[~/lab1]
└─$ ls ~/Desktop

┌──(kali㉿kali)-[~/lab1]
└─$ ▌
```

12) *man*

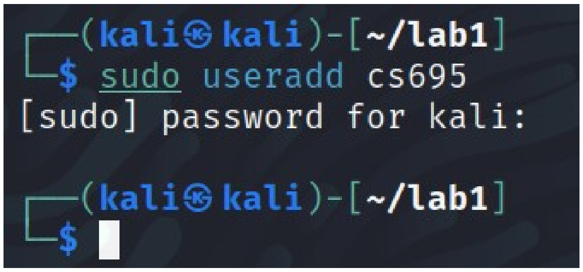


```
LS(1)                          User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort  is  speci-
       fied.

       Mandatory  arguments  to  long  options are mandatory for short options
       too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

       -b, --escape
              print C-style escapes for nongraphic characters

 Manual page ls(1) line 1 (press h for help or q to quit)
```



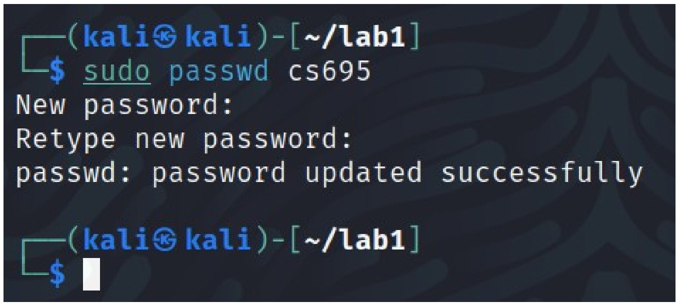
```
┌──(kali㉿kali)-[~/lab1]
└─$ man ls

┌──(kali㉿kali)-[~/lab1]
└─$
```

13) *useradd*



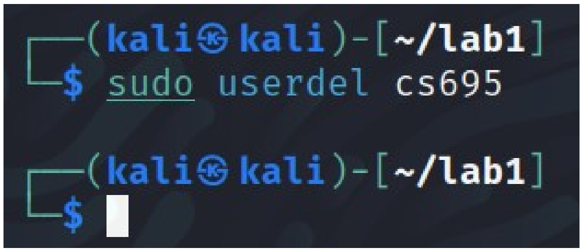
```
┌──(kali㉿kali)-[~/lab1]
└─$ sudo useradd cs695
[sudo] password for kali:

┌──(kali㉿kali)-[~/lab1]
└─$
```

14) *passwd*



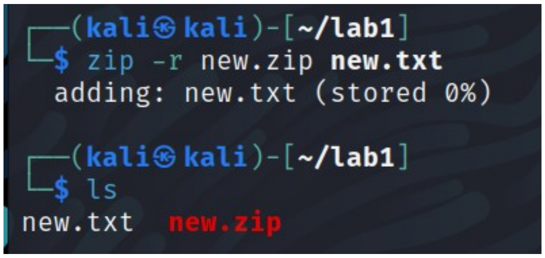
```
┌──(kali㉿kali)-[~/lab1]
└─$ sudo passwd cs695
New password:
Retype new password:
passwd: password updated successfully

┌──(kali㉿kali)-[~/lab1]
└─$ 
```

15) *userdel*

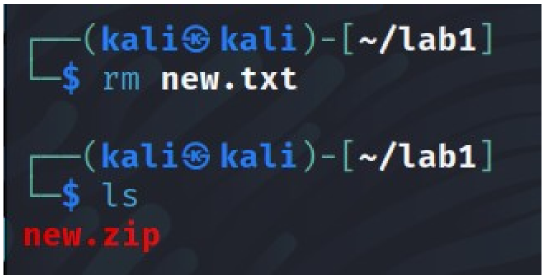


```
┌──(kali㉿kali)-[~/lab1]
└─$ sudo userdel cs695

┌──(kali㉿kali)-[~/lab1]
└─$ 
```

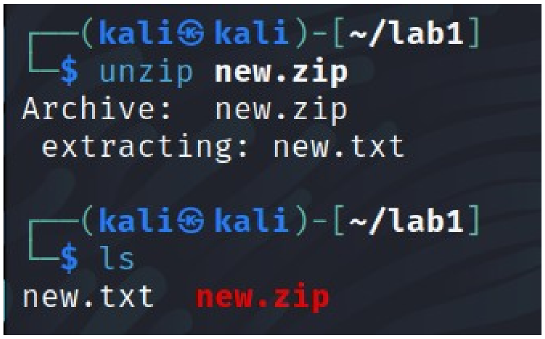
16) *zip/unzip*



```
┌──(kali㉿kali)-[~/lab1]
└─$ zip -r new.zip new.txt
  adding: new.txt (stored 0%)

┌──(kali㉿kali)-[~/lab1]
└─$ ls
new.txt  new.zip
```

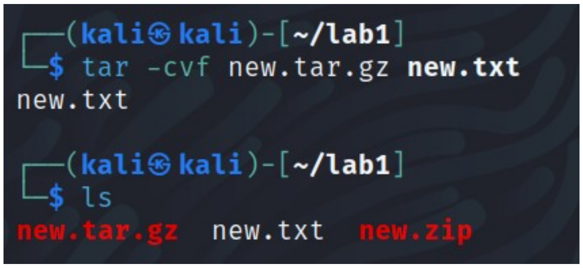


```
┌──(kali㉿kali)-[~/lab1]
└─$ rm new.txt

┌──(kali㉿kali)-[~/lab1]
└─$ ls
new.zip
```

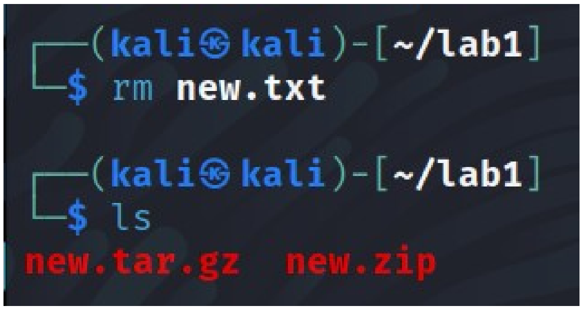


```
┌──(kali㉿kali)-[~/lab1]
└─$ unzip new.zip
Archive:  new.zip
 extracting: new.txt

┌──(kali㉿kali)-[~/lab1]
└─$ ls
new.txt  new.zip
```

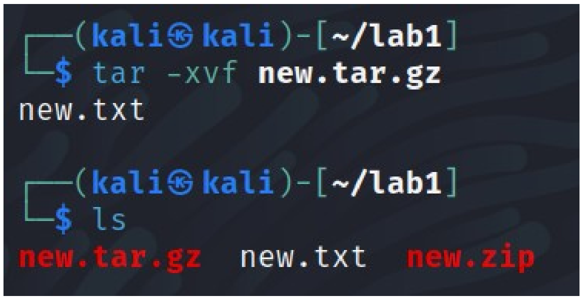
17) *tar*



```
┌──(kali㉿kali)-[~/lab1]
└─$ tar -cvf new.tar.gz new.txt
new.txt

┌──(kali㉿kali)-[~/lab1]
└─$ ls
new.tar.gz  new.txt  new.zip
```

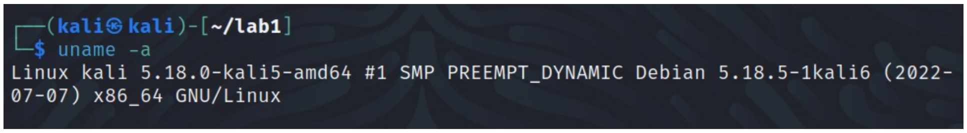


```
┌──(kali㉿kali)-[~/lab1]
└─$ rm new.txt

┌──(kali㉿kali)-[~/lab1]
└─$ ls
new.tar.gz  new.zip
```



```
┌──(kali㉿kali)-[~/lab1]
└─$ tar -xvf new.tar.gz
new.txt

┌──(kali㉿kali)-[~/lab1]
└─$ ls
new.tar.gz  new.txt  new.zip
```

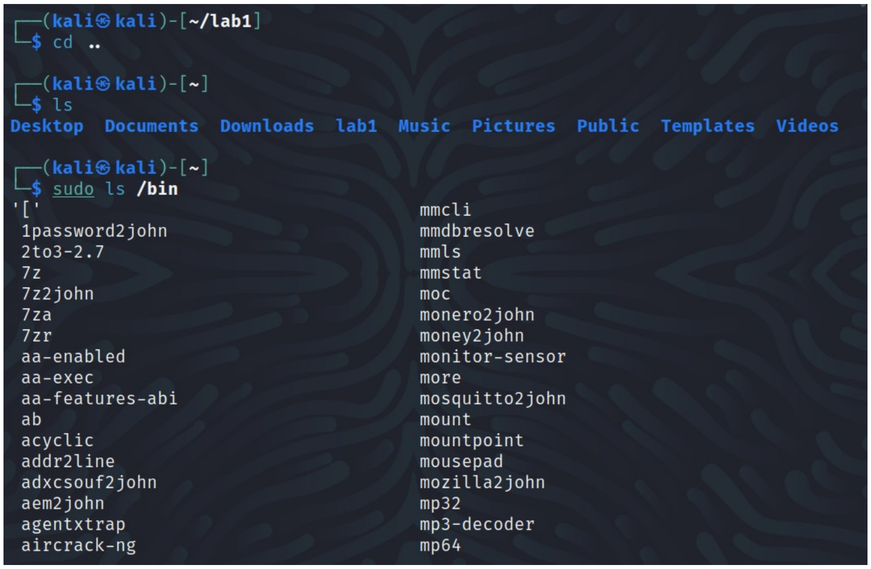
18) *uname*



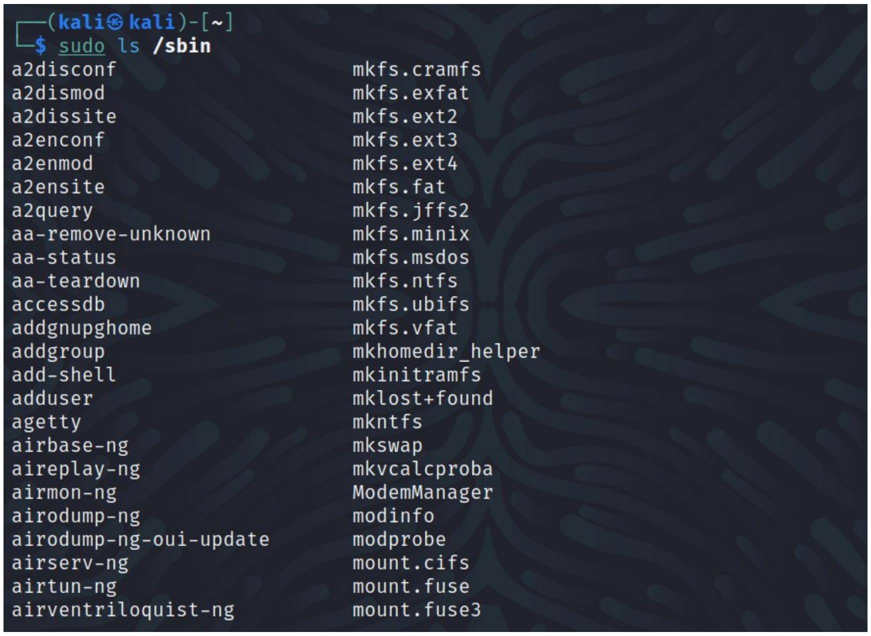
```
┌──(kali㉿kali)-[~/lab1]
└─$ uname -a
Linux kali 5.18.0-kali5-amd64 #1 SMP PREEMPT_DYNAMIC Debian 5.18.5-1kali6 (2022-
07-07) x86_64 GNU/Linux
```

Questions

a) Find the /bin and /sbin folder and show their contents (provide a screenshot of the used commands). Explain what you have found in the /bin and /sbin folder and the difference between these two folders.



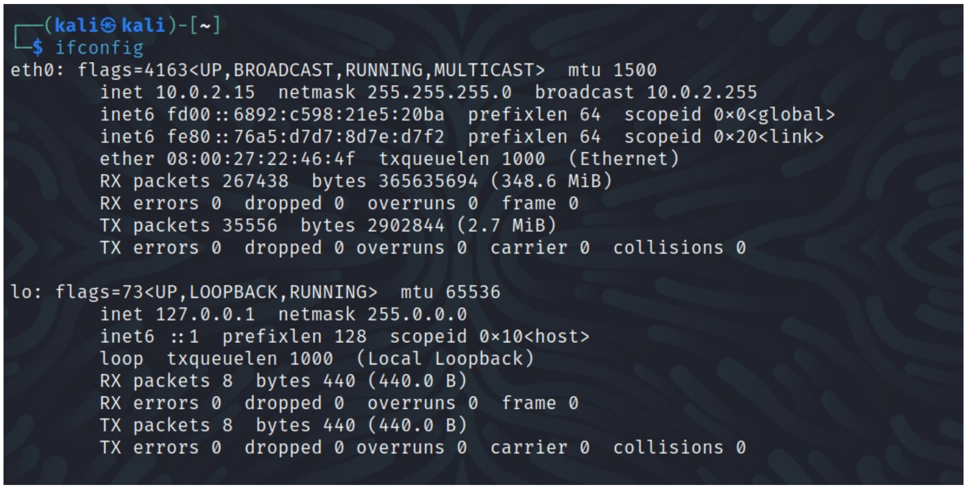
```
┌──(kali㉿kali)-[~/lab1]
└─$ cd ..

┌──(kali㉿kali)-[~]
└─$ ls
Desktop  Documents  Downloads  lab1  Music  Pictures  Public  Templates  Videos

┌──(kali㉿kali)-[~]
└─$ sudo ls /bin
'['                                   mmcli
 1password2john                       mmdbresolve
 2to3-2.7                             mmls
 7z                                   mmstat
 7z2john                              moc
 7za                                  monero2john
 7zr                                  money2john
 aa-enabled                           monitor-sensor
 aa-exec                              more
 aa-features-abi                      mosquitto2john
 ab                                   mount
 acyclic                              mountpoint
 addr2line                            mousepad
 adxcsouf2john                        mozilla2john
 aem2john                             mp32
 agentxtrap                           mp3-decoder
 aircrack-ng                          mp64
```



```
┌──(kali㉿kali)-[~]
└─$ sudo ls /sbin
a2disconf                    mkfs.cramfs
a2dismod                     mkfs.exfat
a2dissite                    mkfs.ext2
a2enconf                     mkfs.ext3
a2enmod                      mkfs.ext4
a2ensite                     mkfs.fat
a2query                      mkfs.jffs2
aa-remove-unknown            mkfs.minix
aa-status                    mkfs.msdos
aa-teardown                  mkfs.ntfs
accessdb                     mkfs.ubifs
addgnupghome                 mkfs.vfat
addgroup                     mkhomedir_helper
add-shell                    mkinitramfs
adduser                      mklost+found
agetty                       mkntfs
airbase-ng                   mkswap
aireplay-ng                  mkvcalcproba
airmon-ng                    ModemManager
airodump-ng                  modinfo
airodump-ng-oui-update       modprobe
airserv-ng                   mount.cifs
airtun-ng                    mount.fuse
airventriloquist-ng          mount.fuse3
```

The /bin and /sbin folders contain the list of binaries (or executables) that can be performed. While they both contain a list of binaries, /bin is accessible for a normal user to run, whereas /sbin (standing for system binary) contains the binaries where superuser privileges are required.

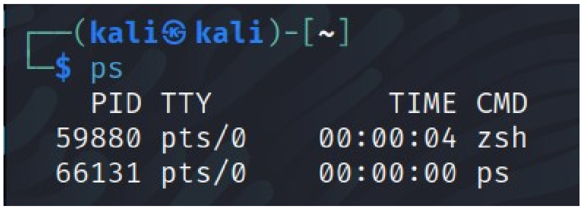
b) Get the virtual machine network information using *ifconfig* (provide a screenshot of the used command). What is your IP address? Are you connected to the public network?



```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fd00::6892:c598:21e5:20ba  prefixlen 64  scopeid 0×0<global>
        inet6 fe80::76a5:d7d7:8d7e:d7f2  prefixlen 64  scopeid 0×20<link>
        ether 08:00:27:22:46:4f  txqueuelen 1000  (Ethernet)
        RX packets 267438  bytes 365635694 (348.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 35556  bytes 2902844 (2.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 440 (440.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 440 (440.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

My IP address in the above screenshot is 10.0.2.15. This address falls within the Class A network IP range, and is a private network connection. The IP showing a private network connection is because the virtual machine is using the computer running it as a host and is forwarding traffic to the internal network being used by VirtualBox.

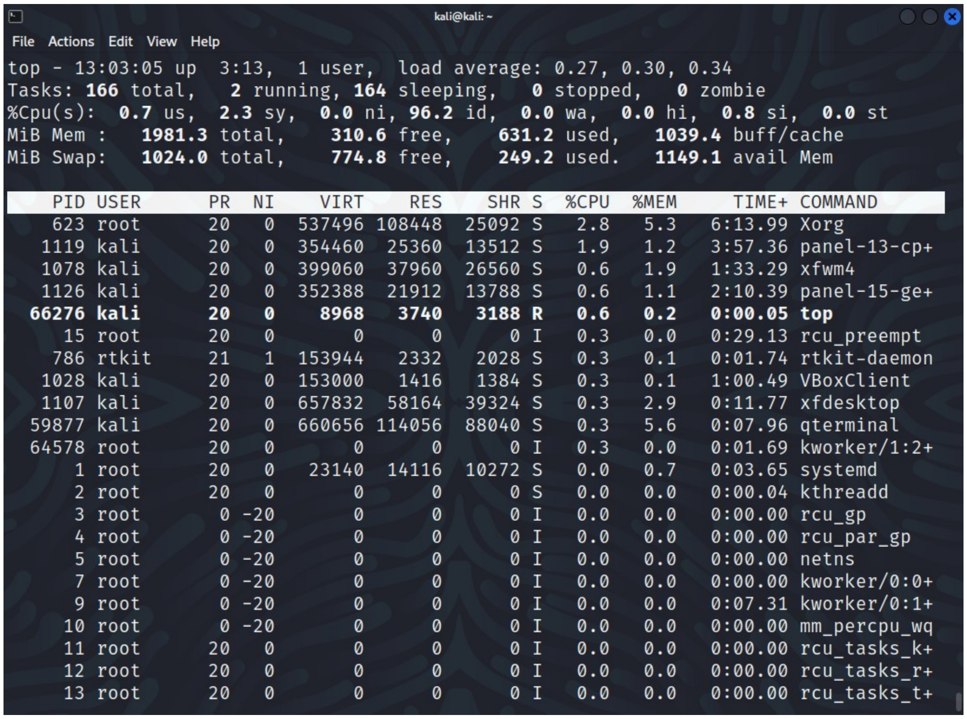
c) Get process information using the *ps* command. What are the 4 information printed by the *ps* command?



```
┌──(kali㉿kali)-[~]
└─$ ps
    PID TTY          TIME CMD
  59880 pts/0    00:00:04 zsh
  66131 pts/0    00:00:00 ps
```

The four pieces of information printed by *ps* are:
PID   – Process ID
TTY   – Terminal Type
TIME – The amount of CPU time that the process has used
CMD – The command that launched the process

d) Get Linux processes information using the *top* command (type q to return to the terminal from the output of the *top* command). What is the information in the header of the result of the command?



```
top - 13:03:05 up  3:13,  1 user,  load average: 0.27, 0.30, 0.34
Tasks: 166 total,   2 running, 164 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.7 us,  2.3 sy,  0.0 ni, 96.2 id,  0.0 wa,  0.0 hi,  0.8 si,  0.0 st
MiB Mem :   1981.3 total,    310.6 free,    631.2 used,   1039.4 buff/cache
MiB Swap:   1024.0 total,    774.8 free,    249.2 used.   1149.1 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    623 root      20   0  537496 108448  25092 S   2.8   5.3   6:13.99 Xorg
   1119 kali      20   0  354460  25360  13512 S   1.9   1.2   3:57.36 panel-13-cp+
   1078 kali      20   0  399060  37960  26560 S   0.6   1.9   1:33.29 xfwm4
   1126 kali      20   0  352388  21912  13788 S   0.6   1.1   2:10.39 panel-15-ge+
  66276 kali      20   0    8968   3740   3188 R   0.6   0.2   0:00.05 top
     15 root      20   0       0      0      0 I   0.3   0.0   0:29.13 rcu_preempt
    786 rtkit     21   1  153944   2332   2028 S   0.3   0.1   0:01.74 rtkit-daemon
   1028 kali      20   0  153000   1416   1384 S   0.3   0.1   1:00.49 VBoxClient
   1107 kali      20   0  657832  58164  39324 S   0.3   2.9   0:11.77 xfdesktop
  59877 kali      20   0  660656 114056  88040 S   0.3   5.6   0:07.96 qterminal
  64578 root      20   0       0      0      0 I   0.3   0.0   0:01.69 kworker/1:2+
      1 root      20   0   23140  14116  10272 S   0.0   0.7   0:03.65 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.04 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
      5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 netns
      7 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0+
      9 root       0 -20       0      0      0 I   0.0   0.0   0:07.31 kworker/0:1+
     10 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
     11 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_k+
     12 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_r+
     13 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_t+
```

In the header op the result of the *top* command, we are given the following information:

top – the uptime information (up), how long the system has been running (3:13 in this case), number of users (user), and load averages for the last minute, five minutes, and fifteen minutes (load average)

Tasks – total number of processes (total), how many processes are currently executing (running), processes awaiting resources (sleeping), processes exiting (stopped), and processes waiting for a parent process to release it (zombie)

%Cpu(s) – CPU time spent running user processes (us), kernel processes (sy), processes with a nice value (ni), idle processes (id), waiting for I/O operations to complete (wa), hardware interrupts (hi), software interrupts (si), and stolen time from the virtual machine by the hypervisor (st)

MiB Mem – total installed memory (total), amount of available memory (free), amount of used memory (used), and amount of memory buffered and cached (buff/cache)

MiB Swap – total amount of virtual memory (total), amount of available virtual memory (free), amount of used virtual memory (used), and amount of virtual memory available for starting applications without swapping (avail Mem)

e) What is the difference between the *ps* and the *top* commands?

The *ps* and *top* commands both provide information on processes running, however *ps* provides a snapshot of the information on processes running at that specific time whereas *top* provides dynamic data that updates as the processes are running.

Reflection

a) What is the purpose of the lab in your own words?

I think that the purpose of this lab is setting us up for the future assignments and labs in this course using Kali Linux and the suite of software it comes with. Prior to this class, I have only used Ubuntu and handful of times, however I have quite enjoyed setting up Kali and seeing a brief preview of the tools that can be used in it.

b) What did you learn? Did you achieve the objectives?

Up until this lab I was not aware Kali Linux existed, so for starters I learned that Kali is a thing! I am also now aware of the different tools that Kali has to offer that were discussed in Part 1.4. Becoming familiar with and getting to use the tools such as Nmap, John, and Mimikatz is something that I am particularly looking forward to now that I have the virtual machine set up. I believe I have achieved the objectives, and that the virtual machine I am using is configured properly for the upcoming work in this class.

c) Is this lab hard or easy? Are the lab instructions clear?

I would say that this lab was on the easier to follow side due to the clear instructions. I appreciate that each section contains steps that are well written and easily followed.
The only difficulty I had was on step 7 of Part 2 where I encountered the GPG error for updating. Fortunately, the Kali blog contained the solution to the problem.

d) Other feedback

Having the lab session recording available was a great resource for this lab and it was very helpful for me at times when I wanted to make sure what I was doing was correct. I'm eager to work more in this virtual machine as it is the first time I have encountered most of the applications listed and I am curious what kinds of information and data can be gathered.
The only detail I would suggest for this lab is including a note for the GPG error if others have encountered the same message when working through part 2 of the lab.