

Linux Basics and Operating System Report

Install a Virtual Machine on a Personal Computer and Install Kali Linux

Steps to Install Kali Linux on a Virtual Machine:

1. Download VirtualBox or VMware Player and install it.
2. Download Kali Linux ISO image from the official site.
3. Open VirtualBox/VMware and create a new virtual machine.
4. Allocate memory (RAM) and disk space as recommended.
5. Attach the Kali Linux ISO image to the VM's virtual CD/DVD drive.
6. Start the VM and follow the on-screen Kali Linux installation steps.
7. Complete the installation and reboot the VM.



Fig: Linux Desktop

What is Linux?

Linux is an open-source operating system created in 1991. It is fast, reliable, small, and highly customizable, requiring minimal hardware resources. Linux is maintained by a global community of programmers and runs on a wide range of devices from wristwatches to supercomputers. Unlike proprietary OSs like Windows or macOS, Linux source code is freely available for anyone to inspect, modify, and redistribute.

A **Linux distribution** (distro) bundles the Linux kernel with tools and software. Popular distros include Debian, Ubuntu, Red Hat, CentOS, and SUSE.

The Value of Linux

- **Open Source:** Anyone can use and customize Linux for free.
- **Powerful Command Line Interface (CLI):** Allows direct and remote system control.
- **Greater Control:** Root (superuser) access lets administrators modify everything.
- **Superior Network Control:** Linux is designed for robust networking and supports many network applications.

Linux Tools

Linux systems in Security Operations Centers (SOC) often include penetration testing tools like packet generators, port scanners, and exploits. Kali Linux is a distribution dedicated to pen-testing, bundling numerous security tools categorized by their functions.

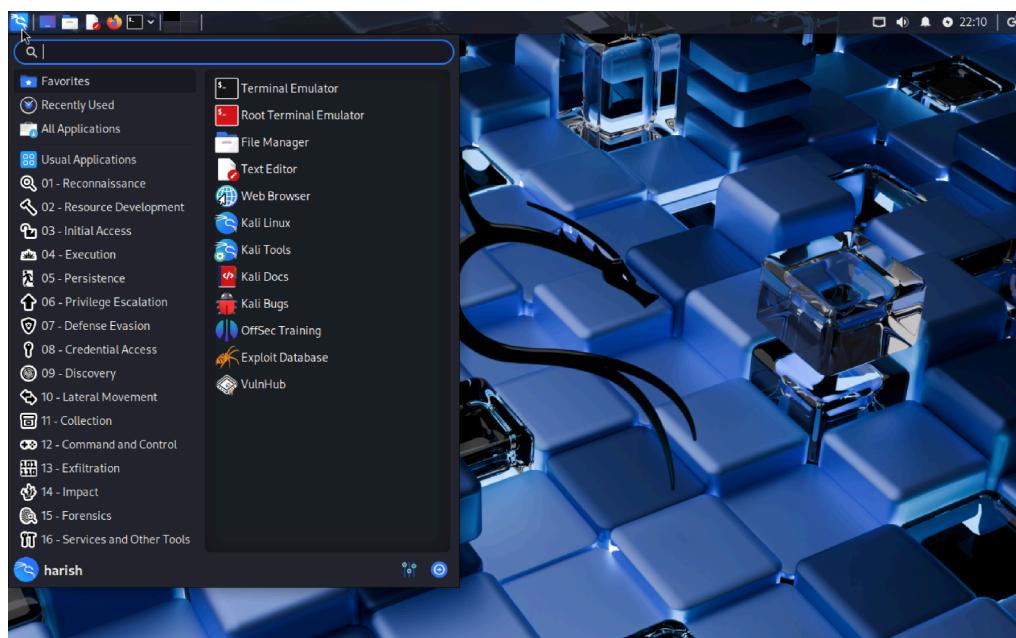


Fig: Pentesting Tools

Working in the Linux Shell

- Linux users interact via the CLI or GUI.
- The CLI is accessed through terminal emulators (e.g., `gnome-terminal`, `xterm`, `konsole`).
- The terms shell, console, CLI terminal, and terminal window often mean the same.

Basic Linux Commands

Command	Description
<code>ls</code>	Lists files and directories
<code>pwd</code>	Prints the current directory path
<code>cd</code>	Changes directory
<code>mkdir</code>	Creates a new directory
<code>rm</code>	Removes files or directories
<code>cp</code>	Copies files or directories
<code>mv</code>	Moves or renames files or directories

Note: Commands require proper permissions.

File and Directory Commands

Commands to navigate and manage files:

- `ls -l` : List with detailed info
- `stat filename` : Show file status and attributes
- `file filename` : Identify file type
- `find /path -name filename` : Search for files by name

Working with Text Files

Linux has many text editors:

- **Graphical editors:** Easy to use but require GUI.
- **Command-line editors:** Essential for remote management.

Common CLI text editors:

- **nano:** Easy to use; commands at bottom, e.g., **CTRL+O** to save, **CTRL+W** to search.
- **vim:** Powerful editor, steeper learning curve.

Example to open a file with nano:

```
nano /etc/hosts
```

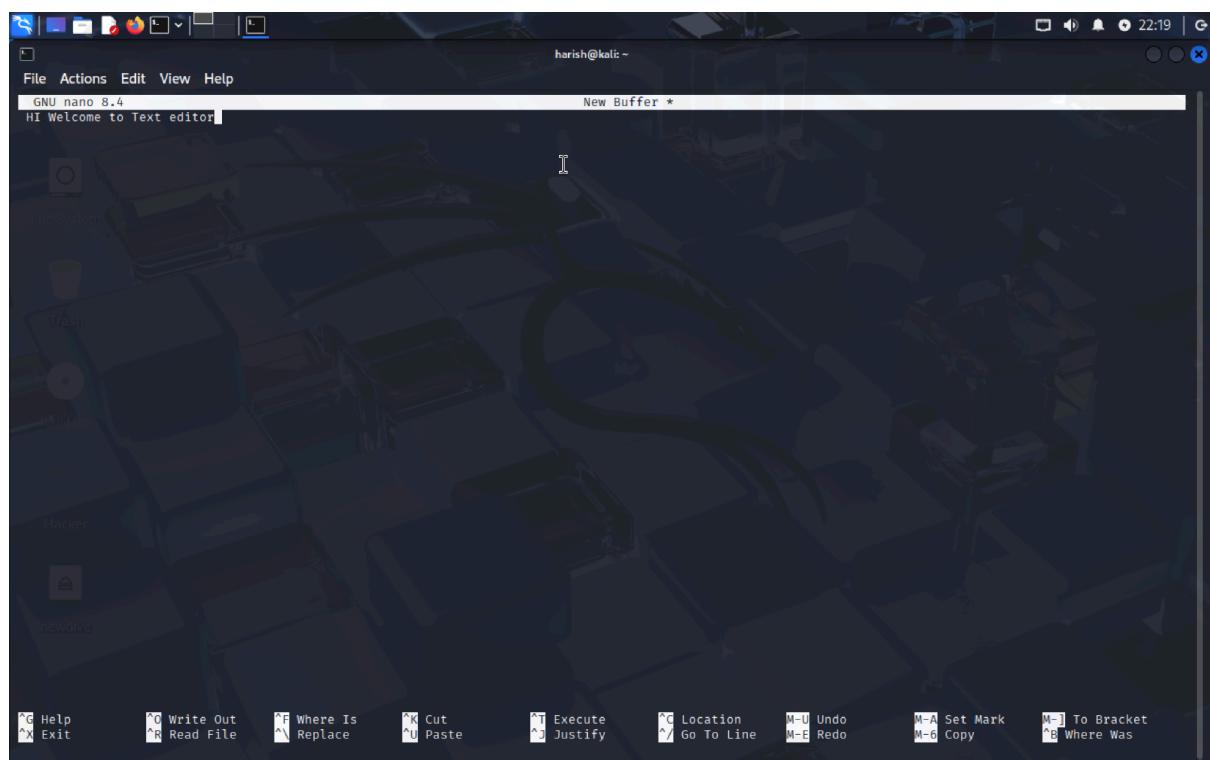


Fig: Text Editor (nano)

Linux Servers and Clients

- **Server:** A computer providing services like file sharing, email, or web hosting.
- **Client:** A computer or software that accesses server resources.

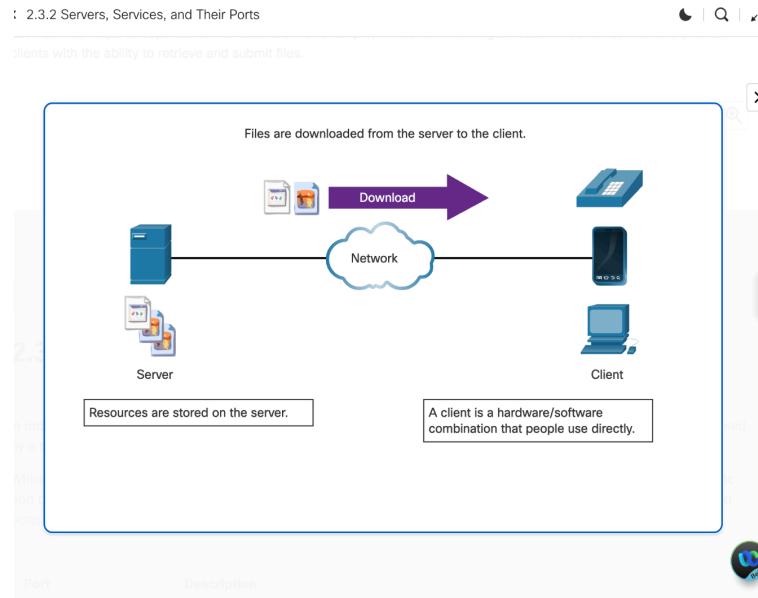


Fig: server to client

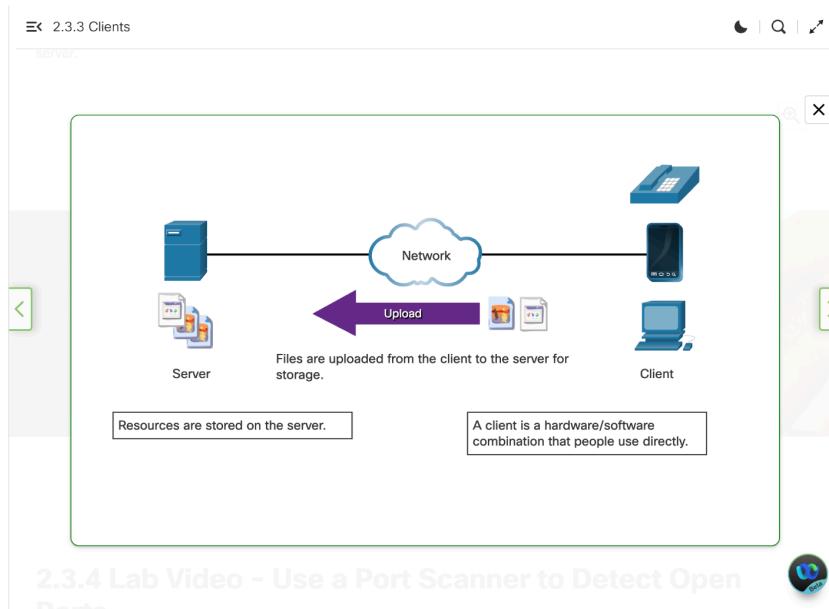


Fig: Client to Server

File System Types in Linux

Linux supports various file systems differing in speed, flexibility, and security.

ext2 (Second Extended File System)

- Default in many older Linux distros.
 - No journaling (which increases speed but risks corruption).
 - Preferred for flash storage to minimize writes.
-

ext3 (Third Extended File System)

- Adds journaling to ext2 for safer recovery.
 - Maximum file size up to 32 TB.
-

ext4 (Fourth Extended File System)

- Successor to ext3 with better performance.
- Supports larger file sizes and volumes.
- Can run with or without journaling.

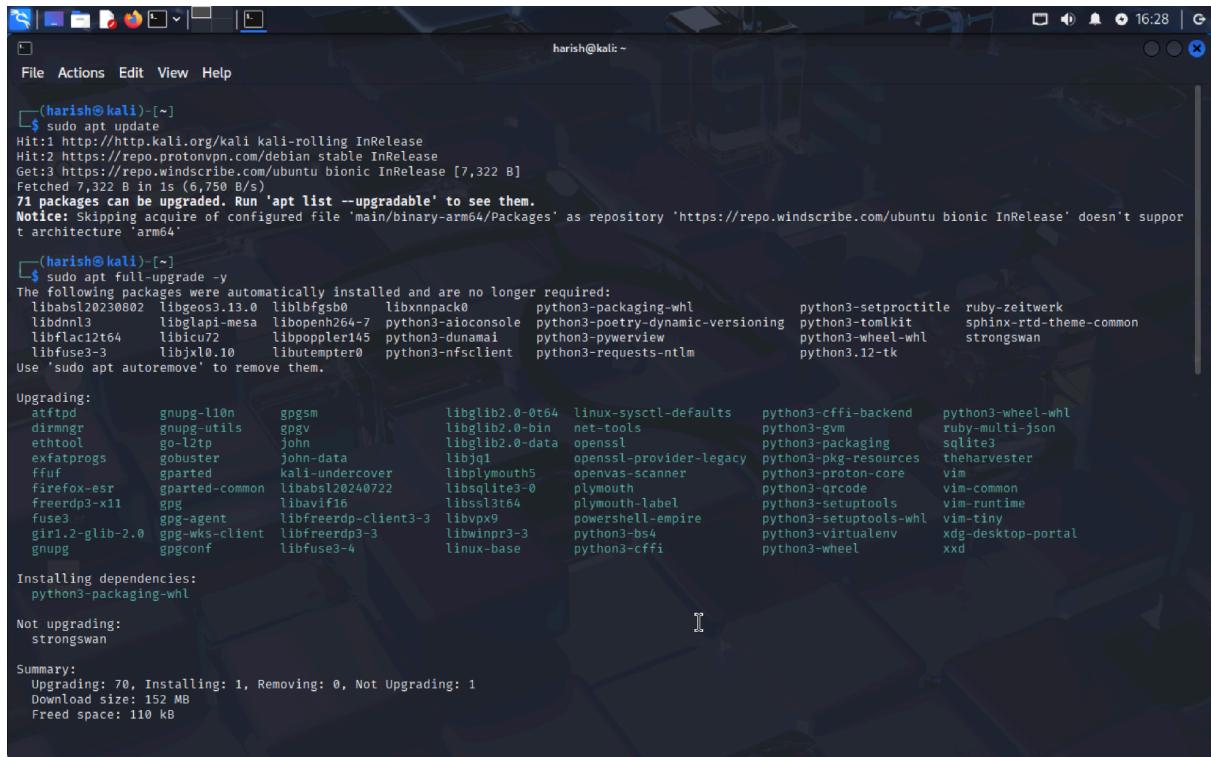
Installing and Running Applications on a Linux Host

Linux uses **package managers** to install and manage software.

- Debian/Ubuntu: `apt` and `dpkg`
- Arch Linux: `pacman`

Example commands for Debian-based systems:

```
sudo apt-get update # Update package list  
sudo apt-get upgrade # Upgrade installed packages  
sudo apt-get install package_name # Install a package
```



```
(harish@kali) [~]  
$ sudo apt update  
Hit:1 http://http.kali.org/kali kali-rolling InRelease  
Hit:2 https://repo.protonvpn.com/debian stable InRelease  
Get:3 https://repo.windscribe.com/ubuntu bionic InRelease [7,322 B]  
Fetched 7,322 B in 1s (6,750 B/s)  
71 packages can be upgraded. Run 'apt list --upgradable' to see them.  
Notice: Skipping acquire of configured file 'main/binary-arm64/Packages' as repository 'https://repo.windscribe.com/ubuntu bionic InRelease' doesn't support architecture 'arm64'.  
  
(harish@kali) [~]  
$ sudo apt full-upgrade -y  
The following packages were automatically installed and are no longer required:  
libabsl20230802 libgeos3.13.0 liblbbfsb0 liblxnnpack0 python3-packaging-whl python3-setproctitle ruby-zeitwerk  
libdmnl3 libglapi-mesa libopenh264-7 python3-aioconsole python3-poetry-dynamic-versioning python3-tomlkit sphinx-rtd-theme-common  
libfflac12t64 libicu72 libpoppler145 python3-dunamai python3-pywerView python3-wheel-wl  
libfuse3-3 libjxl0.10 libutempter0 python3-nfsclient python3-requests-ntlm python3.12-tk  
Use 'sudo apt autoremove' to remove them.  
  
Upgrading:  
atftpd gnupg-l10n gpgsm libglib2.0-0t64 linux-sysctl-defaults python3-cffi-backend python3-wheel-whl  
dirmngr gnupg-utils gpgv libglib2.0-bin net-tools python3-gvm ruby-multi-json  
ethtool go-l2tp john libglib2.0-data openssl python3-packaging sqlite3  
exfatprogs gobuster john-data libjq1 openssl-provider-legacy python3-pkg-resources theharvester  
ffuf gparted kali-undercover libplymouth5 opensvas-scanner python3-proton-core vim  
firefox-esr gparted-common libabsl20240722 libsqlite3-0 plymouth python3-qrcode vim-common  
freerdp3-x11 gpg libavif16 libss13t64 plymouth-label python3-setupools vim-runtime  
fuse3 gpg-agent libfreerdp-client3-3 libvpx9 powershell-empire python3-setuptools-wl vim-tiny  
gir1.2-glib-2.0 gpg-wks-client libfreerdp3-3 libwinpr3-3 python3-bs4 python3-virtualenv xdg-desktop-portal  
gnupg gpgconf libfuse3-4 linux-base python3-cffi python3-wheel xxd  
  
Installing dependencies:  
python3-packaging-whl  
  
Not upgrading:  
strongswan  
  
Summary:  
Upgrading: 70, Installing: 1, Removing: 0, Not Upgrading: 1  
Download size: 152 MB  
Freed space: 110 kB
```

Fig: Update && Upgrade

Processes

- A **process** is an executing program.
- Useful commands:
 - **ps** : List running processes
 - **top** : Interactive process viewer
 - **kill PID** : Terminate a process by ID

```
F S UID      PID  PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
4 S root      1    0  0 80  0 - 6171 - 13:01 ? 00:00:00 /sbin/init splash
1 S root      2    0  0 80  0 -     0 - 13:01 ? 00:00:00 [kthreadd]
1 S root      3    2  0 80  0 -     0 - 13:01 ? 00:00:00 [pool_workqueue_release]
1 I root      4    2  0 60 -20 -     0 - 13:01 ? 00:00:00 [kworker/R-kvfree_rcu_reclaim]
1 I root      5    2  0 60 -20 -     0 - 13:01 ? 00:00:00 [kworker/R-rcu_gp]
1 I root      6    2  0 60 -20 -     0 - 13:01 ? 00:00:00 [kworker/R-sync_wq]
1 I root      7    2  0 60 -20 -     0 - 13:01 ? 00:00:00 [kworker/R-slub_flushwq]
1 I root      8    2  0 60 -20 -     0 - 13:01 ? 00:00:00 [kworker/R-netns]
1 I root      11   2  0 60 -20 -     0 - 13:01 ? 00:00:00 [kworker/0:0H-events_highpri]
1 I root      13   2  0 60 -20 -     0 - 13:01 ? 00:00:00 [kworker/R-mm_percpu_wq]
1 I root      14   2  0 80  0 -     0 - 13:01 ? 00:00:00 [rcu_tasks_rude_kthread]
1 I root      15   2  0 80  0 -     0 - 13:01 ? 00:00:00 [rcu_tasks_trace_kthread]
1 S root      16   2  0 80  0 -     0 - 13:01 ? 00:00:00 [ksoftirqd/0]
1 I root      17   2  0 80  0 -     0 - 13:01 ? 00:00:00 [rcu_sched]
1 S root      18   2  0 80  0 -     0 - 13:01 ? 00:00:00 [rcu_exp_par_gp_kthread_worker/0]
1 S root      19   2  0 80  0 -     0 - 13:01 ? 00:00:00 [rcu_exp_gp_kthread_worker]
1 S root      20   2  0 -40 -     0 - 13:01 ? 00:00:00 [migration/0]
1 S root      21   2  0 80  0 -     0 - 13:01 ? 00:00:00 [cpuhp/0]
1 S root      22   2  0 80  0 -     0 - 13:01 ? 00:00:00 [cpuhp/1]
1 S root      23   2  0 -40 -     0 - 13:01 ? 00:00:00 [migration/1]
1 S root      24   2  0 80  0 -     0 - 13:01 ? 00:00:00 [ksoftirqd/1]
1 I root      26   2  0 60 -20 -     0 - 13:01 ? 00:00:00 [kworker/1:0H-events_highpri]
1 S root      27   2  0 80  0 -     0 - 13:01 ? 00:00:00 [cpuhp/2]
1 S root      28   2  0 -40 -     0 - 13:01 ? 00:00:00 [migration/2]
1 S root      29   2  0 80  0 -     0 - 13:01 ? 00:00:00 [ksoftirqd/2]
1 I root      31   2  0 60 -20 -     0 - 13:01 ? 00:00:00 [kworker/2:0H-events_highpri]
1 S root      32   2  0 80  0 -     0 - 13:01 ? 00:00:00 [cpuhp/3]
1 S root      33   2  0 -40 -     0 - 13:01 ? 00:00:00 [migration/3]
1 S root      34   2  0 80  0 -     0 - 13:01 ? 00:00:00 [ksoftirqd/3]
1 I root      36   2  0 60 -20 -     0 - 13:01 ? 00:00:00 [kworker/3:0H-events_highpri]
5 I root      38   2  0 80  0 -     0 - 13:01 ? 00:00:00 [kworker/u16:1-flush-254:0]
1 I root      39   2  0 80  0 -     0 - 13:01 ? 00:00:00 [kworker/u16:2-events_unbound]
5 S root      41   2  0 80  0 -     0 - 13:01 ? 00:00:00 [kdevtmpfs]
1 I root      42   2  0 60 -20 -     0 - 13:01 ? 00:00:00 [kworker/R-inet_frag_wq]
1 S root      43   2  0 80  0 -     0 - 13:01 ? 00:00:00 [kauditd]
1 S root      44   2  0 80  0 -     0 - 13:01 ? 00:00:00 [khungtaskd]
1 S root      45   2  0 80  0 -     0 - 13:01 ? 00:00:00 [oom_reaper]
1 I root      46   2  0 60 -20 -     0 - 13:01 ? 00:00:00 [kworker/R-writeback]
1 S root      47   2  0 80  0 -     0 - 13:01 ? 00:00:00 [kcompactd0]
```

Fig: Processes (ps or ps aux)

Rootkit Check

A **rootkit** is malware that hides unauthorized access by modifying the OS kernel.

- Rootkits are hard to detect and remove.
- Detection often uses trusted media and specialized tools.

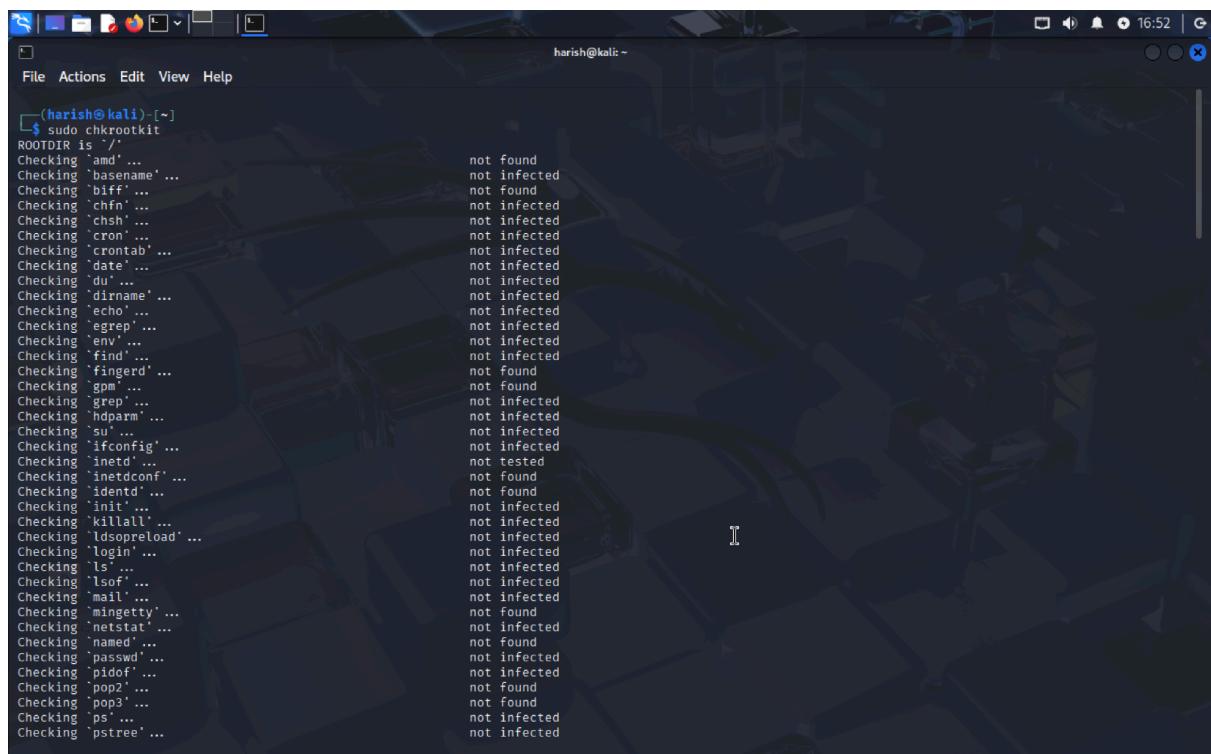
chkrootkit is a popular Linux tool to check for rootkits.

How to use chkrootkit:

Install:

```
sudo apt-get install chkrootkit
```

1. Run: sudo chkrootkit



```
(harish@kali)-[~]
$ sudo chkrootkit
ROOTDIR is '/'
Checking 'amd' ...
Checking 'basename' ...
Checking 'biff' ...
Checking 'chfn' ...
Checking 'chsh' ...
Checking 'cron' ...
Checking 'crontab' ...
Checking 'date' ...
Checking 'du' ...
Checking 'dirname' ...
Checking 'echo' ...
Checking 'egrep' ...
Checking 'env' ...
Checking 'find' ...
Checking 'fingerd' ...
Checking 'gpm' ...
Checking 'grep' ...
Checking 'hdparm' ...
Checking 'su' ...
Checking 'ifconfig' ...
Checking 'inetd' ...
Checking 'inetdconf' ...
Checking 'identd' ...
Checking 'init' ...
Checking 'killall' ...
Checking 'ldsopreload' ...
Checking 'login' ...
Checking 'ls' ...
Checking 'lsof' ...
Checking 'mail' ...
Checking 'mingetty' ...
Checking 'netstat' ...
Checking 'named' ...
Checking 'passwd' ...
Checking 'pidof' ...
Checking 'pop2' ...
Checking 'pop3' ...
Checking 'ps' ...
Checking 'pstree' ...
```

Fig: Checking RootKit

It scans the system for known rootkits but is not 100% reliable.

Conclusion

Linux is a powerful and versatile operating system widely used for security, networking, and server management. Understanding Linux basics, file systems, package management, processes, and security tools like rootkit detection are essential for effective administration and security. Tools such as Kali Linux enhance Linux's capabilities in penetration testing. Mastery of command-line tools and configuration files empowers users with precise control and flexibility.

Note: This report is based on material from Cisco Academy Operating Systems course in Information Technology.

Reference: <https://www.netacad.com/courses/operating-systems-basics?courseLang=en-US>