

02. Linux Fundamentals

Philosophy

The Linux philosophy centers on simplicity, modularity, and openness. It advocates for building small, single-purpose programs that perform one task well. These programs can be combined in various ways to accomplish complex operations, promoting efficiency and flexibility. Linux follows five core principles:

Principle	Description
Everything is a file	All configuration files for the various services running on the Linux operating system are stored in one or more text files.
Small, single-purpose programs	Linux offers many different tools that we will work with, which can be combined to work together.
Ability to chain programs together to perform complex tasks	The integration and combination of different tools enable us to carry out many large and complex tasks, such as processing or filtering specific data results.
Avoid captive user interfaces	Linux is designed to work mainly with the shell (or terminal), which gives the user greater control over the operating system.
Configuration data stored in a text file	An example of such a file is the <code>/etc/passwd</code> file, which stores all users registered on the system.

Components

Component	Description
Bootloader	A piece of code that runs to guide the booting process to start the operating system. Parrot Linux uses the GRUB Bootloader.
OS Kernel	The kernel is the main component of an operating system. It manages the resources for system's I/O devices at the hardware level.
Daemons	Background services are called "daemons" in Linux. Their purpose is to ensure that key functions such as scheduling, printing, and multimedia are working correctly. These small programs load after we booted or log into the computer.
OS Shell	The operating system shell or the command language interpreter (also known as the command line) is the interface between the OS and the user. This interface allows the user to tell the OS what to do. The most commonly used shells are Bash, Tcsh/Csh, Ksh, Zsh, and Fish.
Graphics server	This provides a graphical sub-system (server) called "X" or "X-server" that allows graphical programs to run locally or remotely on the X-windowing system.
Window Manager	Also known as a graphical user interface (GUI). There are many options, including GNOME, KDE, MATE, Unity, and Cinnamon. A desktop environment usually has several applications, including file and web browsers. These allow the user to access and manage the essential and frequently accessed features and services of an operating system.
Utilities	Applications or utilities are programs that perform particular functions for the user or another program.

Linux Architecture

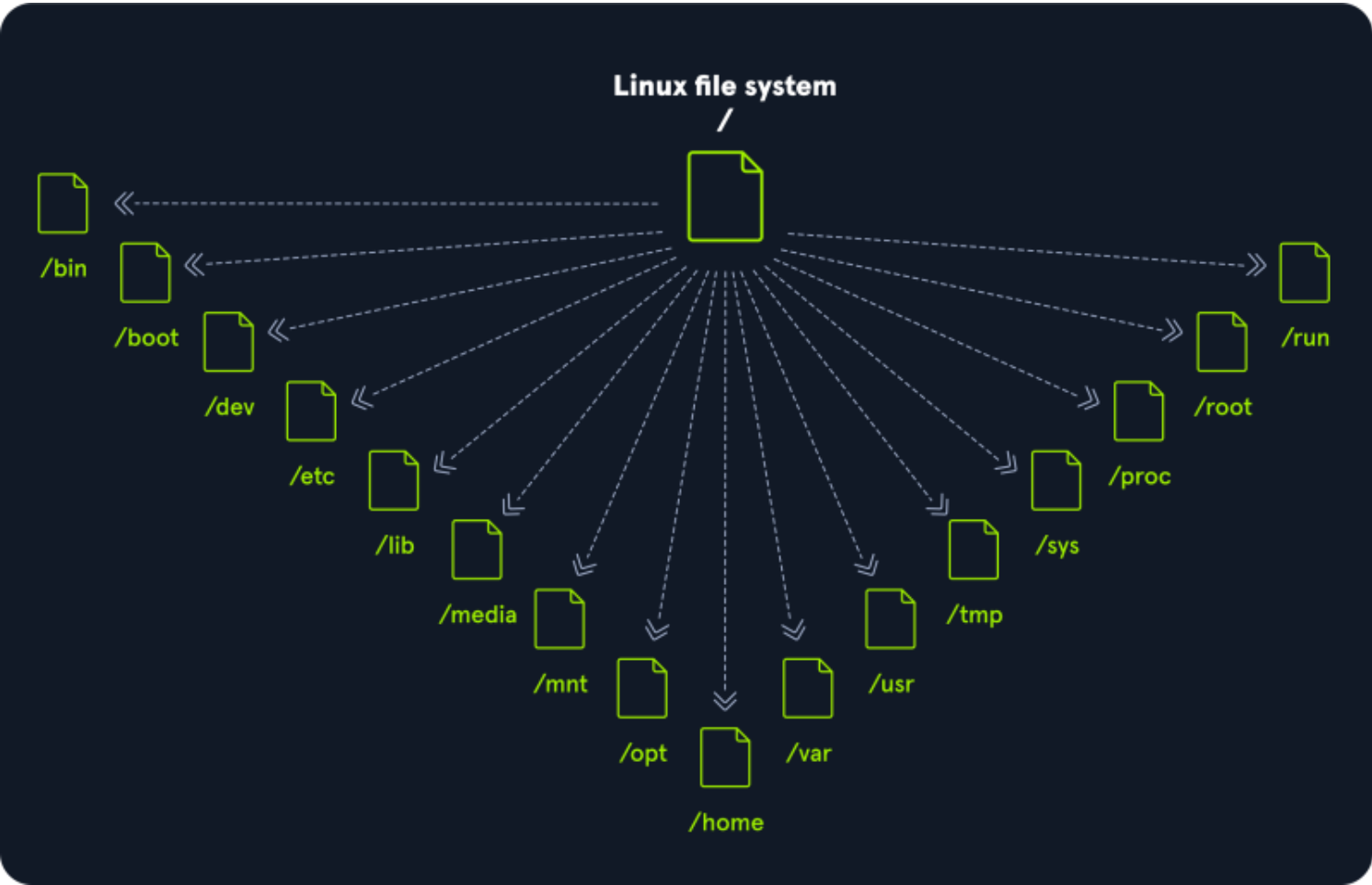
The Linux operating system can be broken down into layers:

Layer	Description
Hardware	Peripheral devices such as the system's RAM, hard drive, CPU, and others.
Kernel	The core of the Linux operating system whose function is to virtualize and control common computer hardware resources like CPU, allocated memory, accessed data, and others. The kernel gives each process its own virtual resources and prevents/mitigates conflicts between different processes.
Shell	A command-line interface (CLI), also known as a shell that a user can enter commands into to execute the kernel's functions.
System Utility	Makes available to the user all of the operating system's functionality.

File System Hierarchy

The Linux operating system is structured in a tree-like hierarchy and is documented in the [Filesystem Hierarchy](#) Standard (FHS). Linux is structured with

the following standard top-level directories:



Path	Description
/	The top-level directory is the root filesystem and contains all of the files required to boot the operating system before other filesystems are mounted, as well as the files required to boot the other filesystems. After boot, all of the other filesystems are mounted at standard mount points as subdirectories of the root.
/bin	Contains essential command binaries.
/boot	Consists of the static bootloader, kernel executable, and files required to boot the Linux OS.
/dev	Contains device files to facilitate access to every hardware device attached to the system.
/etc	Local system configuration files. Configuration files for installed applications may be saved here as well.
/home	Each user on the system has a subdirectory here for storage.
/lib	Shared library files that are required for system boot.
/media	External removable media devices such as USB drives are mounted here.
/mnt	Temporary mount point for regular filesystems.
/opt	Optional files such as third-party tools can be saved here.
/root	The home directory for the root user.
/sbin	This directory contains executables used for system administration (binary system files).
/tmp	The operating system and many programs use this directory to store temporary files. This directory is generally cleared upon system boot and may be deleted at other times without any warning.
/usr	Contains executables, libraries, man files, etc.
/var	This directory contains variable data files such as log files, email in-boxes, web application related files, cron files, and more.

/bin = contains the commands used in the terminal (cd, cat, pwd, ls, etc..)

/boot = files required to boot linux

/dev = hardware devices on linux

/etc = system configuration files, and installed applications configuration files (wireshark, proxychains, python, vim, etc...)

/home = storage for the user

/lib = files required for system boot

/media = portable storage devices, such as USB drives, external hard drives, memory cards, and optical discs are in here

/mnt = a standard, empty directory that serves as a mount point for temporarily attaching or mounting other filesystems and storage devices, such as CD-ROMs, floppy disks, USB drives, or network shares

/opt = It is primarily reserved for the installation of "add-on" or "optional" application software packages that are not part of the core operating system or managed by the distribution's package manager

Difference Between /opt and /usr

◆ Summary Table		
Feature	/usr	/opt
🔧 Purpose	System and user applications	Optional or third-party applications
📦 Package Manager	Managed by system package manager	Often used for manually installed apps
🏠 Structure	Shared among many packages	Self-contained app directories
🧹 Clean Removal	Harder; files are scattered	Easier; everything is in one folder
👤 Used by	OS, package maintainers	Users, vendors, or custom software

- **is for the OS and standard tools.**

- **is for "optional" extras**, often self-contained or vendor-managed.

/root = home directory of the root user, also known as the superuser or administrator. This directory serves as the primary location for the root user's files, including configuration files and personal data.

- you need sudo privileges to open /root, type "sudo su" to access as root user

•:

This command stands for "superuser do" or "substitute user do." It allows a permitted user to execute a command as another user, typically the superuser (root), by providing their own password for authentication.

•:

This command stands for "switch user" or "substitute user." It allows a user to switch to another user account, including the root account. When used without a username, it defaults to switching to the root user.

/sbin = directory that holds essential binary executables and command-line tools used for system administration and booting the operating system, primarily by the root user.

/tmp = The operating system and many programs use this directory to store temporary files. This directory is generally cleared upon system boot and may be deleted at other times without any warning.

/usr = stands for User System Resources and contains essential user-level software, libraries, documentation, and binaries,

/var = This directory contains variable data files such as log files, email in-boxes, web application related files, cron files, and more.

Linux Distributions

Linux distributions - or distros - are operating systems based on the Linux kernel. They are used for various purposes, from servers and embedded devices to desktop computers and mobile phones. Linux distributions are like different branches or franchises of the same company, each tailored to serve specific markets or customer preferences. While they all share the same dedicated employees (components), organizational structure (architecture), and corporate culture (philosophy), each distribution offers its own unique products and

services (software packages and configurations), customizing the experience to meet diverse needs. All while operating under the unified brand and values of Linux. Each Linux distribution is different, with its own set of features, packages, and tools. Some popular examples include:

- [Ubuntu](#)
- [Fedora](#)
- [CentOS](#)
- [Debian](#)
- [Red Hat Enterprise Linux](#)

Many users choose Linux for their desktop computers because it is free, open source, and highly customizable. Ubuntu and Fedora are two popular choices for desktop Linux and beginners. It is also widely used as a server operating system because it is secure, stable, and reliable and comes with frequent and regular updates. Finally, we, as cybersecurity specialists, often prefer Linux because it is open source, meaning its source code is available for scrutiny and customization. Because of such customization, we can optimize and customize our Linux distribution the way we want and configure it for specific use cases only if necessary.

We can use those distros everywhere, including (web) servers, mobile devices, embedded systems, cloud computing, and desktop computing. For cyber security specialists, some of the most popular Linux distributions are but are not limited to:

- [ParrotOS](#)
- [Raspberry Pi OS](#)
- [BlackArch](#)
- [Ubuntu](#)
- [CentOS](#)
- [Pentoo](#)

- [Debian](#)
- [BackBox](#)
- [Kali](#)

The main differences between the various Linux distributions are the included packages, the user interface, and the tools available. [Kali Linux](#) is the most popular distribution for cyber security specialists, including a wide range of security-focused tools and packages. Ubuntu is widespread for desktop users, while Debian is popular for servers and embedded systems. Finally, red Hat Enterprise Linux and CentOS are popular for enterprise-level computing.

Debian

Debian is a widely used and well-respected Linux distribution known for its stability and reliability. It is used for various purposes, including desktop computing, servers, and embedded system. It uses an Advanced Package Tool (APT) package management system to handle software updates and security patches. The package management system helps keep the system up-to-date and secure by automatically downloading and installing security updates as soon as they are available. This can be executed manually or set up automatically.

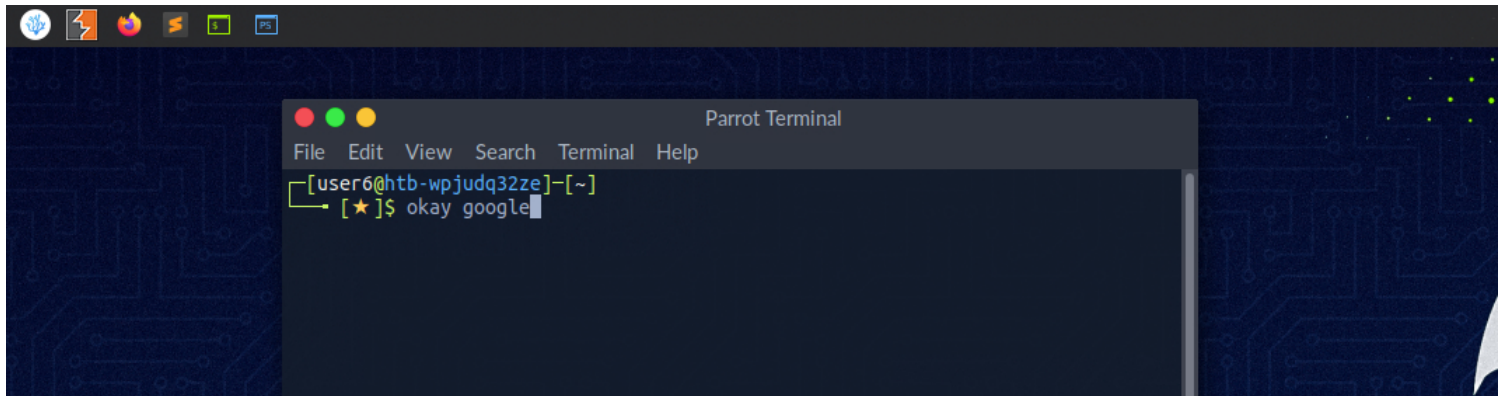
Debian can have a steeper learning curve than other distributions, but it is widely regarded as one of the most flexible and customizable Linux distros. The configuration and setup can be complex, but it also provides excellent control over the system, which can be good for advanced users. The more control we have over a Linux system, the more complex it feels to become. However, it just feels that way compared to the options and possibilities we get. Without learning it with the required depth, we might spend way more time configuring “easy” tasks and processes than when we would learn to use a few commands and tools more in-depth.

Stability and reliability are key strengths of Debian. The distribution is known for its long-term support releases, which can provide updates and security patches for up to five years. This can be especially important for servers and other systems that must be up and running 24/7. It has had some vulnerabilities, but the development community has quickly released patches and security updates. In addition, Debian has a strong commitment to security and privacy, and the distribution has a well-established security track record. Debian is a versatile and

reliable Linux distribution that is widely used for a range of purposes. Its stability, reliability, and commitment to security make it an attractive choice for various use cases, including cyber security.

Introduction to Shell

Web servers are often based on Linux. Knowing how to use the operating system to control it effectively requires understanding and mastering Linux's essential part, the . When we first switched from Windows to Linux, does it look something like this:



A Linux terminal, also called a or command line, provides a text-based input/output (I/O) interface between users and the kernel for a computer system. The term console is also typical but does not refer to a window but a screen in text mode. In the terminal window, commands can be executed to control the system.

We can think of a shell as a text-based GUI in which we enter commands to perform actions like navigating to other directories, working with files, and obtaining information from the system but with way more capabilities.

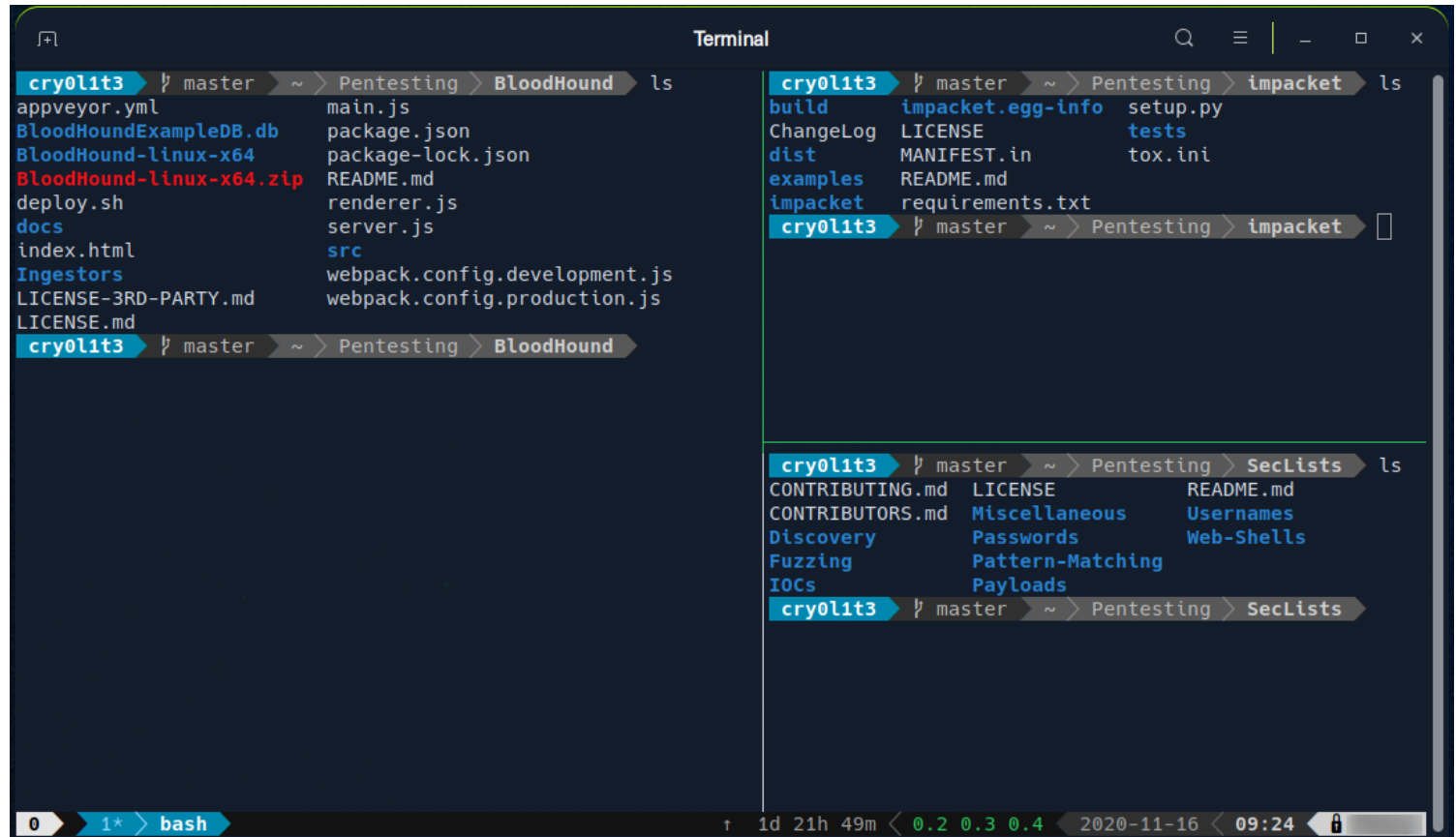
Terminal Emulators

Terminal emulation is software that emulates the function of a terminal. It allows the use of text-based programs within a graphical user interface (). There are also so-called command-line interfaces () that run as additional terminals in one terminal. In short, a terminal serves as an interface to the shell interpreter.

Additionally, () that run as additional terminals within one terminal are like having multiple virtual receptionist desks open on your screen simultaneously. Each one allows you to send different instructions to the server room independently, but through the same main interface. In essence, the terminal

serves as your gateway to communicate with and control the core operations managed by the shell.

Terminal emulators and multiplexers are beneficial extensions for the terminal. They provide us with different methods and functions to work with the terminal, such as splitting the terminal into one window, working in multiple directories, creating different workspaces, and much more. An example of the use of such a multiplexer called Tmux could look something like this:



```
cry01t3 ~ master > Pentesting > BloodHound > ls
appveyor.yml      main.js
BloodHoundExampleDB.db  package.json
BloodHound-linux-x64  package-lock.json
BloodHound-linux-x64.zip  README.md
deploy.sh          renderer.js
docs               server.js
index.html         src
Ingestors          webpack.config.development.js
LICENSE-3RD-PARTY.md  webpack.config.production.js
LICENSE.md

cry01t3 ~ master > Pentesting > BloodHound

cry01t3 ~ master > Pentesting > impacket > ls
build      impacket.egg-info  setup.py
ChangeLog  LICENSE              tests
dist       MANIFEST.in   tox.ini
examples   README.md
impacket    requirements.txt

cry01t3 ~ master > Pentesting > impacket

cry01t3 ~ master > Pentesting > SecLists > ls
CONTRIBUTING.md  LICENSE      README.md
CONTRIBUTORS.md  Miscellaneous  Usernames
Discovery         Passwords     Web-Shells
Fuzzing           Pattern-Matching
IOCs              Payloads

cry01t3 ~ master > Pentesting > SecLists
```

Shell

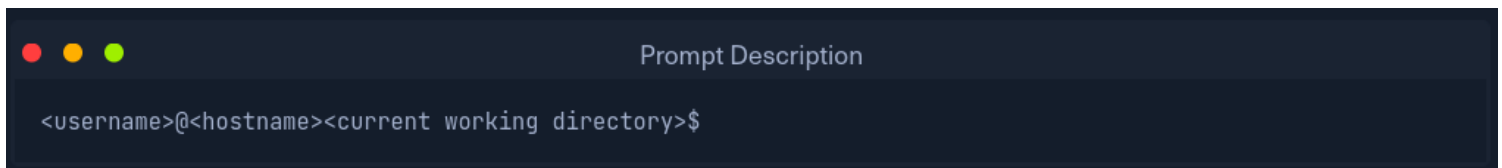
The most commonly used shell in Linux is the `()`, and is part of the GNU project. Everything we do through the GUI we can do with the shell. The shell gives us many more possibilities to interact with programs and processes to get information faster. Besides, many processes can be easily automated with smaller or larger scripts that make manual work much easier.

Besides Bash, there also exist other shells like [Tcsh/Csh](#), [Ksh](#), [Zsh](#), [Fish](#) shell and others.

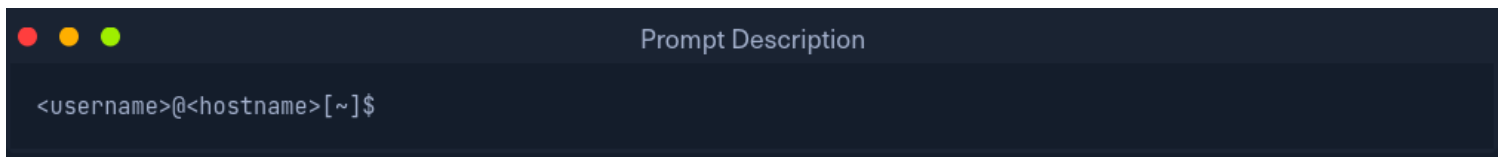
Prompt Description

The bash prompt is simple to understand. By default, it shows information like your username (who you are), your computer's name (hostname), and the folder/directory you're currently working in. It's a line of text that appears on the screen to let you know the system is ready for you. The prompt appears on a new line, and the cursor (the blinking line or box) is placed right after it, waiting for you to type a command.

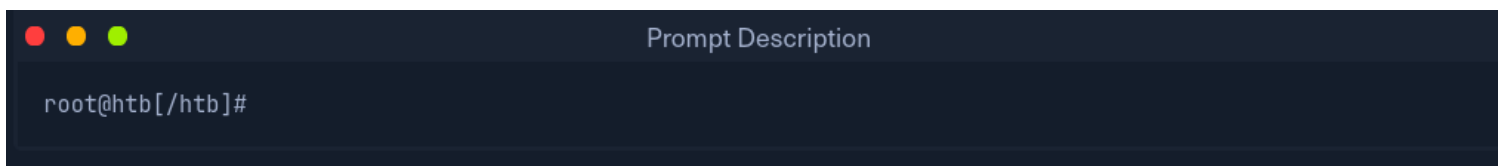
It can be customized to provide useful information to the user. The format can look something like this:



The home directory for a user is marked with a tilde `<~>` and is the default folder when we log in.



The dollar sign, in this case, stands for a user. As soon as we log in as , the character changes to a `<#>` and looks like this:



For example, when we upload and run a shell on the target system, we may not see the username, hostname, and current working directory. This may be due to the **PS1 variable** in the environment not being set correctly. In this case, we would see the following prompts:

Unprivileged - User Shell Prompt



Prompt Description

\$

Privileged - Root Shell Prompt



Prompt Description

#

The `PROMPT` variable in Linux systems controls how your command prompt looks in the terminal. It's like a template that defines the text you see each time the system is ready for you to type a command. By customizing the `PS1` variable, you can change the prompt to display information such as your username, your computer's name, the current folder you're in, or even add colors and special characters. This allows you to personalize the command-line interface to make it more informative or visually appealing.

In addition to displaying basic information like your username and current folder, you can customize the command prompt to show other useful details such as the IP address, date, time, and the success or failure of the last command. This customization is especially helpful during penetration tests because it allows you to keep track of your actions more effectively. For instance, you can set the prompt to show the full path of the current working directory instead of just its name, and even include the target's IP address if needed. Using tools like `history` or reviewing the `~/.bash_history` file (located in the user's home directory), you can record all the commands you've used and organize them by date and time, which aids in documentation and analysis.

The prompt can be customized using special characters and variables in the shell's configuration file (`~/.bashrc` for the Bash shell). For example, we can use: the `%u` character to represent the current username, `%h` for the hostname, and `%P` for the current working directory.

Special Character	Description
\d	Date (Mon Feb 6)
\D{%Y-%m-%d}	Date (YYYY-MM-DD)
\H	Full hostname
\j	Number of jobs managed by the shell
\n	Newline
\r	Carriage return
\s	Name of the shell
\t	Current time 24-hour (HH:MM:SS)
\T	Current time 12-hour (HH:MM:SS)
\@	Current time
\u	Current username
\w	Full path of the current working directory

- is located in the home directory. Use “ls -a” to find it.

Customizing the prompt can be a useful way to make your terminal experience more personalized and efficient. It can also be a helpful tool for troubleshooting and problem-solving, as it can provide important information about the system’s state at any given time.

In addition to customizing the prompt, we can customize their terminal environment with different color schemes, fonts, and other settings to make their work environment more visually appealing and easier to use.

However, we see the same as when working on the Windows GUI here. We are logged in as a user on a computer with a specific name, and we know which directory we are in when we navigate through our system. Bash prompt can also be customized and changed to our own needs. The adjustment of the bash prompt is outside the scope of this module. However, we can look at the [bash-prompt-generator](#) and [powerline](#), which gives us the possibility to adapt our prompt to our needs.

Getting Help

We will always stumble across tools whose optional parameters we do not know from memory or tools we have never seen before. Therefore it is vital to know how we can help ourselves to get familiar with those tools. The first two ways are the **man pages** and the **help functions**. It is always a good idea to familiarize ourselves with the tool we want to try first. We will also learn some possible tricks with some of the tools that we thought were not possible. In the man pages, we will find the detailed manuals with detailed explanations.

```
First Command:

Getting Help

m0nocl3@htb[/htb]$ ls

cacert.der  Documents  Music      Public     Videos
Desktop     Downloads  Pictures   Templates
```

The `ls` command in Linux and Unix systems is used to list the files and directories within the current folder or any specified directory, allowing you to see what's inside and manage files more effectively. Like most Linux commands, `ls` comes with additional options and features that help you filter or format the output to display exactly what you want. To discover which options a tool or command offers, there are several ways to get help. One such method is using the `man` command, which displays the manual pages for commands and provides detailed information about their usage.

```
Syntax:

Getting Help

m0nocl3@htb[/htb]$ man <tool>
```

Let us have a look at an example and get help for the `ls` command:

Example:

```
m0noc13@htb[/htb]$ man ls
```

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

After looking at some examples, we can also quickly look at the optional parameters without browsing through the complete documentation. We have several ways to do that.

Syntax:

```
m0noc13@htb[/htb]$ <tool> --help
```

Example:

```
Getting Help

m0n0cl3@htb[/htb]$ ls --help

Usage: ls [OPTION]... [FILE]...
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

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
      --block-size=SIZE    with -l, scale sizes by SIZE when printing them;
                           e.g., '--block-size=M'; see SIZE format below

  -B, --ignore-backups     do not list implied entries ending with ~
  -c                       with -lt: sort by, and show, ctime (time of last
                           modification of file status information);
                           with -l: show ctime and sort by name;
                           otherwise: sort by ctime, newest first

  -C                       list entries by columns
<SNIP>
```

Some tools or commands like provide a short version of help by using instead of:

Syntax:

```
Getting Help

m0n0cl3@htb[/htb]$ <tool> -h
```

Example:

```
Getting Help

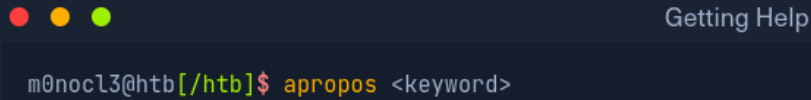
m0n0cl3@htb[/htb]$ curl -h

Usage: curl [options...] <url>
  --abstract-unix-socket <path> Connect via abstract Unix domain socket
  --anyauth              Pick any authentication method
  -a, --append           Append to target file when uploading
  --basic                Use HTTP Basic Authentication
  --cacert <file>        CA certificate to verify peer against
  --capath <dir>         CA directory to verify peer against
  -E, --cert <certificate[:password]> Client certificate file and password
<SNIP>
```

As we can see, the results from each other do not differ in this example. Another

tool that can be useful in the beginning is . Each manual page has a short description available within it. This tool searches the descriptions for instances of a given keyword.

Syntax:



```
m0noc13@htb[/htb]$ apropos <keyword>
```

Example:



```
m0noc13@htb[/htb]$ apropos sudo

sudo (8)                - execute a command as another user
sudo.conf (5)            - configuration for sudo front end
sudo_plugin (8)          - Sudo Plugin API
sudo_root (8)            - How to run administrative commands
sudedit (8)              - execute a command as another user
sudoers (5)              - default sudo security policy plugin
sudoreplay (8)           - replay sudo session logs
visudo (8)               - edit the sudoers file
```

Another useful resource to get help if we have issues to understand a long command is: <https://explainshell.com/>

System Information

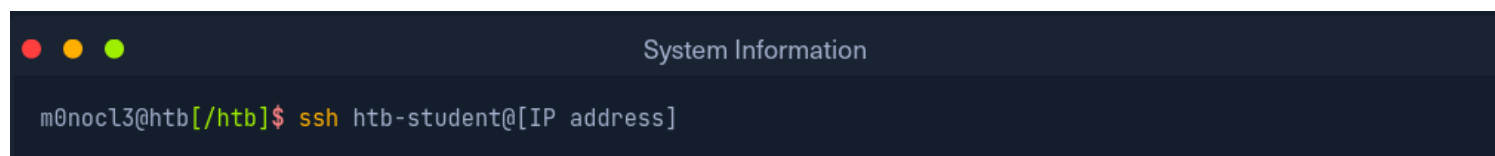
Since we'll be working with various Linux systems, it's important to understand their structure, including system details, processes, network configurations, users/user settings, and directories, along with their related parameters. Below is a list of essential tools to help gather this information. Most of these tools come pre-installed. However, this knowledge is not only crucial for routine Linux tasks, but also plays a key role when assessing security configurations, identifying vulnerabilities, or preventing potential security risks in Linux operating systems.

Command	Description
<code>whoami</code>	Displays current username.
<code>id</code>	Returns users identity
<code>hostname</code>	Sets or prints the name of current host system.
<code>uname</code>	Prints basic information about the operating system name and system hardware.
<code>pwd</code>	Returns working directory name.
<code>ifconfig</code>	The ifconfig utility is used to assign or to view an address to a network interface and/or configure network interface parameters.
<code>ip</code>	Ip is a utility to show or manipulate routing, network devices, interfaces and tunnels.
<code>netstat</code>	Shows network status.
<code>ss</code>	Another utility to investigate sockets.
<code>ps</code>	Shows process status.
<code>who</code>	Displays who is logged in.
<code>env</code>	Prints environment or sets and executes command.
<code>lsblk</code>	Lists block devices.
<code>lsusb</code>	Lists USB devices
<code>lsdf</code>	Lists opened files.
<code>lspci</code>	Lists PCI devices.

Logging In via SSH

(`ssh`) refers to a protocol that allows clients to access and execute commands or actions on remote computers. On Linux-based hosts and servers, as well as other Unix-like operating systems, SSH is one of the permanently installed standard tools and is the preferred choice for many administrators to configure and maintain a computer through remote access. It is an older and very proven protocol that does not require or offer a graphical user interface (GUI). For this reason, it works very efficiently and occupies very few resources. We use this type of connection in the following sections, and in most of the other module lab exercises, to offer the possibility to try out the learned commands and actions in a safe environment.

We can connect to our targets with the following command:

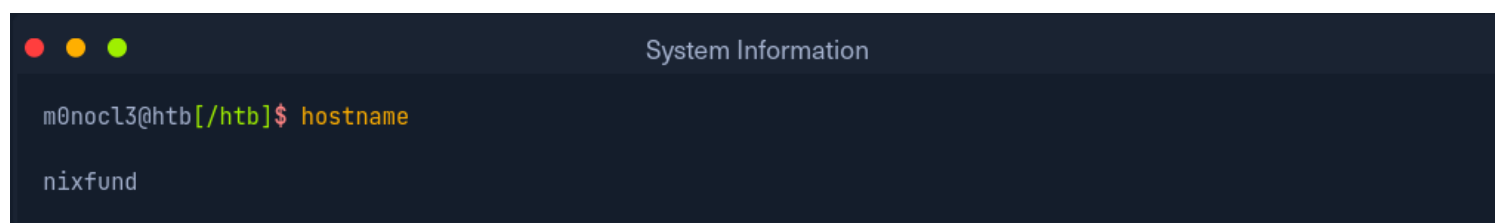
A terminal window titled "System Information" with three colored window control buttons (red, yellow, green) in the top-left corner. The terminal shows a command prompt where the user has executed an SSH command to log into a host named "htb-student".

```
m0noc13@htb[/htb]$ ssh htb-student@[IP address]
```

Now, let us look at a few examples on the machine we have just logged in.

Hostname

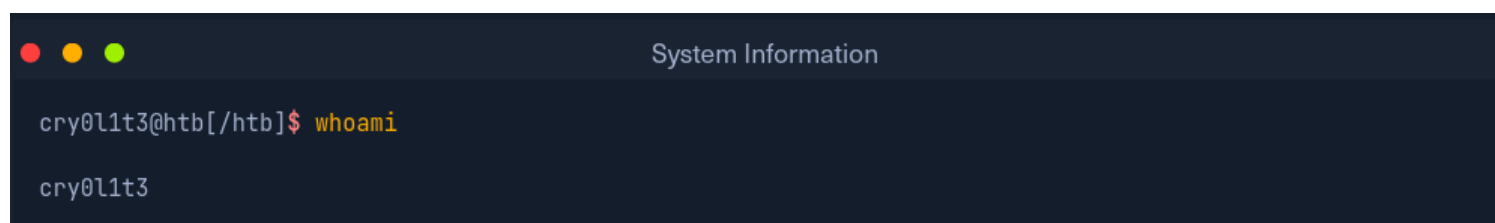
The `hostname` command is pretty self-explanatory and will just print the name of the computer that we are logged into

A terminal window titled "System Information" with three colored window control buttons in the top-left corner. The terminal shows the user executing the `hostname` command, which returns the output "nixfund".

```
m0noc13@htb[/htb]$ hostname  
nixfund
```

Whoami

This quick and easy command can be used on both Windows and Linux systems to get our current username. During a security assessment, we obtain reverse shell access on a host, and one of the first bits of situational awareness we should do is figuring out what user we are running as. From there, we can figure out if the user has any special privileges/access.

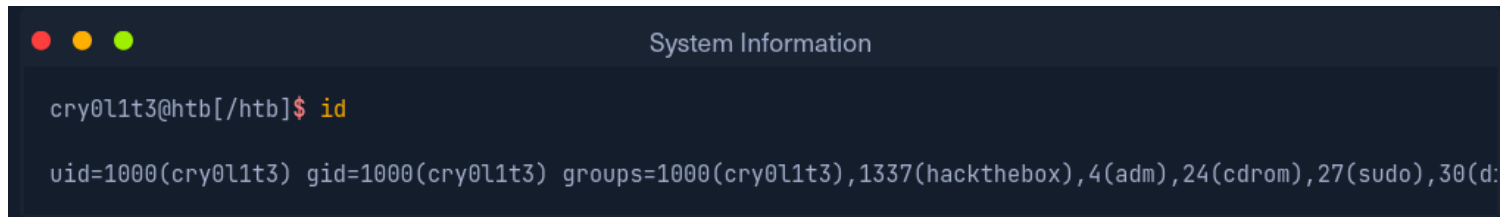
A terminal window titled "System Information" with three colored window control buttons in the top-left corner. The terminal shows the user executing the `whoami` command, which returns the output "cry0l1t3".

```
cry0l1t3@htb[/htb]$ whoami  
cry0l1t3
```

Id

The `id` command expands on the `whoami` command and prints out our effective group membership and IDs. This can be of interest to penetration testers looking to see what access a user may have and sysadmins looking to audit account permissions and group membership. In this output, the group is of interest

because it is non-standard, the `group` means that the user can read log files in and could potentially gain access to sensitive information, membership in the group is of particular interest as this means our user can run some or all commands as the all-powerful `root` user. Sudo rights could help us escalate privileges or could be a sign to a sysadmin that they may need to audit permissions and group memberships to remove any access that is not required for a given user to carry out their day-to-day tasks.

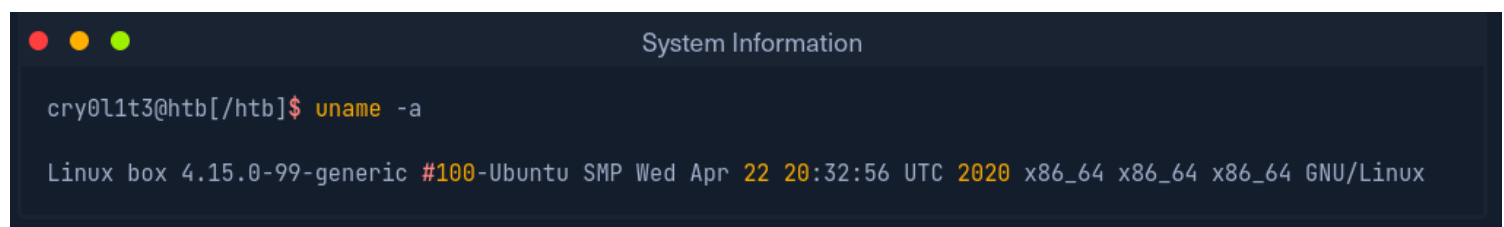
A terminal window titled "System Information" with a dark background. The prompt is "cry0l1t3@htb[/htb]\$". The command "id" has been entered and executed. The output is "uid=1000(cry0l1t3) gid=1000(cry0l1t3) groups=1000(cry0l1t3),1337(hackthebox),4(adm),24(cdrom),27(sudo),30(d:...)".

```
System Information  
cry0l1t3@htb[/htb]$ id  
uid=1000(cry0l1t3) gid=1000(cry0l1t3) groups=1000(cry0l1t3),1337(hackthebox),4(adm),24(cdrom),27(sudo),30(d:...
```

Uname

Let's dig into the `command` a bit more. If we type `man command` in our terminal, we will bring up the man page for the command, which will show the possible options we can run with the command and the results.

Running `uname -a` will print all information about the machine in a specific order: kernel name, hostname, the kernel release, kernel version, machine hardware name, and operating system. The `-a` flag will omit `-p` (processor type) and `-i` (hardware platform) if they are unknown.

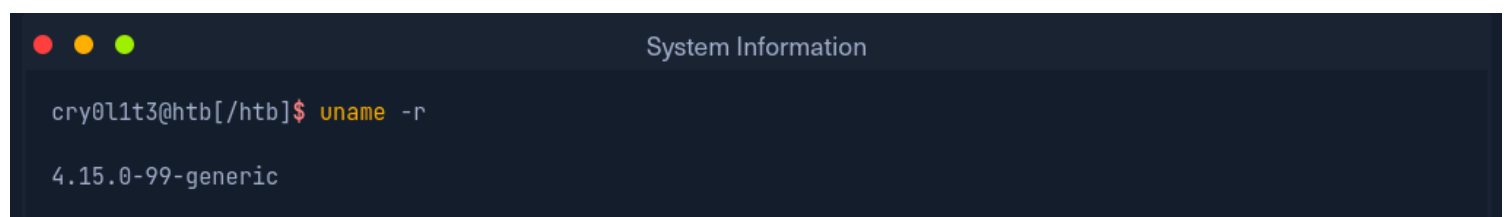
A terminal window titled "System Information" with a dark background. The prompt is "cry011t3@htb[/htb]\$". The command "uname -a" has been entered and executed. The output is "Linux box 4.15.0-99-generic #100-Ubuntu SMP Wed Apr 22 20:32:56 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux".

```
System Information  
cry011t3@htb[/htb]$ uname -a  
Linux box 4.15.0-99-generic #100-Ubuntu SMP Wed Apr 22 20:32:56 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
```

From the above command, we can see that the kernel name is Linux, the hostname is box, the kernel release is 4.15.0-99-generic, the kernel version is #100-Ubuntu SMP Wed Apr 22 20:32:56 UTC 2020, and so on. Running any of these options on their own will give us the specific bit output we are interested in.

Uname to Obtain Kernel Release

Suppose we want to print out the kernel release to search for potential kernel exploits quickly. We can type `uname -r` to obtain this information.

A terminal window titled "System Information" with a dark background. The prompt is "cry011t3@htb[/htb]\$". The command "uname -r" has been entered and executed. The output is "4.15.0-99-generic".

```
System Information  
cry011t3@htb[/htb]$ uname -r  
4.15.0-99-generic
```

With this info, we could go and search for "4.15.0-99-generic exploit," and the first [result](#) immediately appears useful to us.

It is highly recommended to study the commands and understand what they are for and what information they can provide. Though a bit tedious, we can learn much from studying the manpages for common commands. We may even find out things that we did not even know were possible with a given command. This information is not only used for working with Linux. However, it will also be used later to discover vulnerabilities and misconfigurations on the Linux system that may contribute to privilege escalation. Here are a few optional exercises that we can solve for practice purposes, which will help us become familiar with some of the commands.

About Linux Exercises

The exercises provided for studying the Linux OS and its commands might not always be immediately clear in terms of what you need to do, and that's perfectly fine—it's even unavoidable. As you've learned from the Learning Process module, learning something new can feel uncomfortable and may cause stress. You can think of it like the first time you sat behind the wheel of a car and had to drive on your own. It was stressful because there were many things you had to focus on at once. But now, with experience, driving is easier, though you're not learning as much anymore. Similarly, in this module, you may find yourself in situations where you're unsure of what to do, but that's okay. In your cybersecurity journey, you'll often face such moments, and they're a positive indicator that you're learning something new. Overcoming these challenges helps you improve, even if you haven't fully solved the exercise yet. That's the ultimate goal—progress through learning.

The exercises are intentionally designed to gradually push you out of your current knowledge and into unfamiliar territory. This progression is deliberate and ensures that as you continue practicing, your experience and knowledge will naturally expand. While it may feel uncomfortable at times, this process is essential for growth. With each new challenge, you'll stretch beyond what you already know, and with consistent effort, you'll find that your understanding and skills develop almost automatically. Keep practicing, and you'll steadily grow more confident and capable in navigating the unknown.



Questions

Answer the question(s) below to complete this Section and earn cubes!


Questions

Answer the question(s) below to complete this Section and earn cubes!

Target(s): 10.129.25.99 (ACADEMY-NIXFUND) 

Life Left: 111 minute(s)  **Terminate** 

 SSH to 10.129.25.99 (ACADEMY-NIXFUND) with user "htb-student" and password "HTB_@cademy_stdnt!"

 + 0 Find out the machine hardware name and submit it as the answer.

x86_64



Cheat Sheet



Download VPN
Connection File



Submit



Hint

First, connect to the vpn of hackthebox by doing
sudo openvpn academy-regular.ovpn

Session Actions Edit View Help

```
2025-09-10 22:48:49 ROUTE6_GATEWAY fe80::2 IFACE=eth0
2025-09-10 22:48:49 TUN/TAP device tun0 opened
2025-09-10 22:48:49 net_iface_mtu_set: mtu 1500 for tun0
2025-09-10 22:48:49 net_iface_up: set tun0 up
2025-09-10 22:48:49 net_addr_v4_add: 10.10.16.8/23 dev tun0
2025-09-10 22:48:49 net_iface_mtu_set: mtu 1500 for tun0
2025-09-10 22:48:49 net_iface_up: set tun0 up
2025-09-10 22:48:49 net_addr_v6_add: dead:beef:4::1006/64 dev tun0
2025-09-10 22:48:49 net_route_v4_add: 10.10.10.0/23 via 10.10.16.1 dev [
NULL] table 0 metric -1
2025-09-10 22:48:49 net_route_v4_add: 10.129.0.0/16 via 10.10.16.1 dev [
NULL] table 0 metric -1
2025-09-10 22:48:49 add_route_ipv6(dead:beef::/64 → dead:beef:4::1 metr
ic -1) dev tun0
2025-09-10 22:48:49 net_route_v6_add: dead:beef::/64 via :: dev tun0 tab
le 0 metric -1
2025-09-10 22:48:49 Initialization Sequence Completed
2025-09-10 22:48:49 Data Channel: cipher 'AES-256-CBC', auth 'SHA256', p
eer-id: 15, compression: 'lzo'
2025-09-10 22:48:49 Timers: ping 10, ping-restart 120
2025-09-10 22:48:49 Protocol options: protocol-flags cc-exit tls-ekm dyn
-tls-crypt
```

Once you see Initialization Sequence Completed you are ready to go, do not close

the terminal tab as this will kill your connection, open a new tab and start working from it.

Then ssh to the target machine by

```
ssh username@your_server_ip
```

In this case it was

```
ssh htb-student@10.129.25.99
```

Then it will ask for authenticity and password. After entering the password, you will now be able to access the target machine.

The password for this target machine was **HTB_@cademy_stdnt!**

The screenshot shows the HTB Academy interface. On the left, under the 'Questions' section, it says 'Answer the question(s) below to complete this Section and earn cubes!'. The target is '10.129.25.99 (ACADEMY-NIXFUND)'. The life left is '111 minute(s)'. There are buttons for '+', 'Terminate', and 'X'. Below this, it says 'SSH to 10.129.25.99 (ACADEMY-NIXFUND) with user "htb-student" and password "HTB_@cademy_stdnt!"'. The question is '+ 0 Find out the machine hardware name and submit it as the answer.' The answer field contains 'x86_64'. On the right, there are buttons for 'Cheat Sheet' and 'Download VPN Connection File'. At the bottom right, there are 'Submit' and 'Hint' buttons.

The first question was to find the machine hardware.

What I did was I typed

```
uname -m
```

```
htb-student@nixfund:~$ uname -m
x86_64
```

The next question was asking the path for the target machine's home directory.

+ 1 📦 What is the path to htb-student's home directory?

/home/htb-student

Submit

What I did was I typed

`pwd`

```
htb-student@nixfund:~$ pwd
/home/htb-student
```

The next question was asking the path to the target machine's email.

+ 0 📦 What is the path to the htb-student's mail?

Submit your answer here...

+10 Streak pts

Submit

On a typical Linux system, local user mail is stored in one of two standard locations:

`/var/mail/<username>`

`/var/spool/mail/<username>`

So I searched it on `/var/mail/<username>` first:

```
htb-student@nixfund:~$ ls /var/mail
root
```

So it wasn't in there. Lets check `/var/spool/mail/<username>`:

```
htb-student@nixfund:~$ ls /var/spool/mail
root
```


Huh, both of them does not contain the mail for htb-student.

Even if the file doesn't exist yet, `/var/mail/htb-student` is still considered the "working directory" for their mail by the system.

+ 0

What is the path to the htb-student's mail?

`/var/mail/htb-student`

Submit

The next question asks which shell is specified for the target machine.

+ 0

Which shell is specified for the htb-student user?

Submit your answer here...

+10 Streak pts Submit

It's asking: What command interpreter (shell program) is set as the default for that user?

A "shell" is what runs when you log in — like bash, sh, zsh, etc. Each user in Linux has a default shell assigned.

Where this information is stored

Linux keeps basic account info in `/etc/passwd`.

Each line represents one user, with several fields separated by colons :

```
htb-student@nixfund:/etc$ cat passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
lxd:x:105:65534::/var/lib/lxd:/bin/false
uidd:x:106:110::/run/uidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1::/var/cache/pollinate:/bin/false
sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
```

To filter and find htb-student specifically, we use **grep**

Why use grep htb-student /etc/passwd

/etc/passwd contains entries for all users (system accounts, daemons, etc.).

Using grep filters out only the line for htb-student, so you don't have to scroll through everything.

That way, you immediately see which shell is set for that specific user.

```
htb-student@nixfund:/etc$ grep htb-student /etc/passwd
htb-student:x:1002:1002::/home/htb-student:/bin/bash
```

And there we go, we found the shell that htb-student has.

+ 0

Which shell is specified for the htb-student user?

/bin/bash

Submit

Next question:

+ 0

Which kernel release is installed on the system? (Format: 1.22.3)

4.15.0

Submit

To find the kernel release on the system:

```
htb-student@nixfund:/etc$ uname -r  
4.15.0-123-generic
```

Last Question:

+ 1

What is the name of the network interface that MTU is set to 1500?

Submit your answer here...

+10 Streak pts

Submit

What MTU means

MTU (Maximum Transmission Unit) is the largest packet size (in bytes) that a network interface can send in one piece, without fragmentation.

For example:

Ethernet networks usually use **1500 bytes** as default MTU.

VPNs, tunnels, or special networks might use smaller values (like 1400, 576, etc.)

So, MTU is basically a limit on packet size per interface.

So in finding MTU, we either use the **ip a** command or the **ip link** command.

Why ip a or ip link shows MTU

The **ip** command (part of the iproute2 package) lets you inspect and configure network interfaces.

ip a (short for ip addr show) displays interfaces, addresses, and MTU values.

ip link (short for ip link show) focuses on link-layer info, including MTU, state (UP/DOWN), and MAC address.

```
htb-student@nixfund:/etc$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
  default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP g
  roup default qlen 1000
    link/ether 00:50:56:b0:33:63 brd ff:ff:ff:ff:ff:ff
    inet 10.129.25.99/16 brd 10.129.255.255 scope global ens192
        valid_lft forever preferred_lft forever
    inet6 dead:beef::250:56ff:feb0:3363/64 scope global dynamic mngtmpadd
  r
        valid_lft 86398sec preferred_lft 14398sec
    inet6 fe80::250:56ff:feb0:3363/64 scope link
        valid_lft forever preferred_lft forever
```

Here we see **mtu 1500** in **ens192**

```
htb-student@nixfund:/etc$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode
DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP m
ode DEFAULT group default qlen 1000
    link/ether 00:50:56:b0:33:63 brd ff:ff:ff:ff:ff:ff
```

The same goes for `ip link`

+ 1

What is the name of the network interface that MTU is set to 1500?

ens192

Submit

Navigation

Navigation is essential, like working with the mouse as a standard Windows user. With it, we move across the system and work in directories and with files, we need and want. Therefore, we use different commands and tools to print out information about a directory or a file and can use advanced options to optimize the output to our needs.

One of the best ways to learn something new is to experiment with it. Here we cover the sections on navigating through Linux, creating, moving, editing, and deleting files and folders, finding them on the operating system, different types of redirects, and what file descriptors are. We will also find shortcuts to make our work with the shell much easier and more comfortable. We recommend experimenting on our locally hosted VM. Ensure we have created a snapshot for our VM in case our system gets unexpectedly damaged.

Let us start with the navigation. Before we move through the system, we have to find out in which directory we are. We can find out where we are with the command `pwd`.

```
Navigation

cry0l1t3@htb[~]$ pwd

/home/cry0l1t3
```

Only the **ls** command is needed to list all the contents inside a directory. It has many additional options that can complement the display of the content in the current folder.

```
Navigation

cry0l1t3@htb[~]$ ls

Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
```

Using it without any additional options will display the directories and files only. However, we can also add the **-l** option to display more information on those directories and files.

```
Navigation

cry0l1t3@htb[~]$ ls -l

total 32
drwxr-xr-x 2 cry0l1t3 htbacademy 4096 Nov 13 17:37 Desktop
drwxr-xr-x 2 cry0l1t3 htbacademy 4096 Nov 13 17:34 Documents
drwxr-xr-x 3 cry0l1t3 htbacademy 4096 Nov 15 03:26 Downloads
drwxr-xr-x 2 cry0l1t3 htbacademy 4096 Nov 13 17:34 Music
drwxr-xr-x 2 cry0l1t3 htbacademy 4096 Nov 13 17:34 Pictures
drwxr-xr-x 2 cry0l1t3 htbacademy 4096 Nov 13 17:34 Public
drwxr-xr-x 2 cry0l1t3 htbacademy 4096 Nov 13 17:34 Templates
drwxr-xr-x 2 cry0l1t3 htbacademy 4096 Nov 13 17:34 Videos
```

First, we see the total amount of blocks (1024-byte) used by the files and directories listed in the current directory, which indicates the total size used. That means it used $32 \text{ blocks} * 1024 \text{ bytes/block} = 32,768 \text{ bytes}$ (or 32 KB) of disk space. Next, we see a few columns that are structured as follows:

Column Content	Description
<code>drwxr-xr-x</code>	Type and permissions
<code>2</code>	Number of hard links to the file/directory
<code>cry0l1t3</code>	Owner of the file/directory
<code>htbacademy</code>	Group owner of the file/directory
<code>4096</code>	Size of the file or the number of blocks used to store the directory information
<code>Nov 13 17:37</code>	Date and time
<code>Desktop</code>	Directory name

However, we will not see everything that is in this folder. A directory can also have hidden files that start with a dot at the beginning of its name (e.g., `.bashrc` or `.bash_history`). Therefore, we need to use the command `ls -la` to list all files of a directory:

```

Navigation

cry0l1t3@htb[~]$ ls -la

total 403188
drwxr-xr-x 2 cry0l1t3 htbacademy 4096 Nov 13 17:37 .bash_history
drwxr-xr-x 2 cry0l1t3 htbacademy 4096 Nov 13 17:37 .bashrc
...SNIP...
drwxr-xr-x 2 cry0l1t3 htbacademy 4096 Nov 13 17:37 Desktop
drwxr-xr-x 2 cry0l1t3 htbacademy 4096 Nov 13 17:34 Documents
drwxr-xr-x 3 cry0l1t3 htbacademy 4096 Nov 15 03:26 Downloads
drwxr-xr-x 2 cry0l1t3 htbacademy 4096 Nov 13 17:34 Music
drwxr-xr-x 2 cry0l1t3 htbacademy 4096 Nov 13 17:34 Pictures
drwxr-xr-x 2 cry0l1t3 htbacademy 4096 Nov 13 17:34 Public
drwxr-xr-x 2 cry0l1t3 htbacademy 4096 Nov 13 17:34 Templates
drwxr-xr-x 2 cry0l1t3 htbacademy 4096 Nov 13 17:34 Videos

```

To list the contents of a directory, we do not necessarily need to navigate there first. We can also use "`ls`" to specify the path where we want to know the contents.

```
Navigation

cry011t3@htb[~]$ ls -l /var/

total 52
drwxr-xr-x  2 root root    4096 Mai 15 18:54 backups
drwxr-xr-x 18 root root    4096 Nov 15 16:55 cache
drwxrwsrwt  2 root whoopsie 4096 Jul 25 2018 crash
drwxr-xr-x 66 root root    4096 Mai 15 03:08 lib
drwxrwsr-x  2 root staff   4096 Nov 24 2018 local
<SNIP>
```

We can do the same thing to navigate to the directory. To move through the directories, we use the command `cd`. Let us change to the `/dev/shm` directory. Of course, we can go to the `/dev` directory first and then `/shm`. Nevertheless, we can also enter the full path and jump there.

```
Navigation

cry011t3@htb[~]$ cd /dev/shm

cry011t3@htb[/dev/shm]$
```

Since we were in the home directory before, we can quickly jump back to the directory we were last in.

```
Navigation

cry011t3@htb[/dev/shm]$ cd -

cry011t3@htb[~]$
```

The shell also offers us the auto-complete function, which makes navigation easier. If we now type `cd /dev/s` and press `[TAB]` twice, we will get all entries starting with the letter `"s"` in the directory of `/dev/`.

If we add the letter `"h"` to the letter `"s,"` the shell will complete the input since otherwise there will be no folders in this directory beginning with the letters `"sh"`. If we now display all contents of the directory, we will only see the following contents.


```
Navigation

cry011t3@htb[/dev/shm]$ ls -la /dev/shm

total 0
drwxrwxrwt  2 root root   40 Mai 15 18:31 .
drwxr-xr-x 17 root root 4000 Mai 14 20:45 ..
```

The first entry with a single dot (.) indicates the current directory we are currently in. The second entry with two dots (..) represents the parent directory /dev. This means we can jump to the parent directory with the following command.

```
Navigation

cry011t3@htb[/dev/shm]$ cd ..

cry011t3@htb[/dev]$
```

Since our shell is filled with some records, we can clean the shell with the command **clear**. First, however, let us return to the directory /dev/shm before and then execute the clear command to clean up our terminal.

```
Navigation

cry011t3@htb[/dev]$ cd shm && clear
```

Another way to clean up our terminal is to use the shortcut [Ctrl] + [L]. We can also use the arrow keys (↑ or ↓) to scroll through the command history, which will show us the commands that we have used before. But we also can search through the command history using the shortcut [Ctrl] + [R] and type some of the text that we are looking for.

Questions

Answer the question(s) below to complete this Section and earn cubes!

[Cheat Sheet](#)[Download VPN
Connection File](#)

Target(s): Target(s) are spawning...

SSH to with user "htb-student" and password "HTB_@cademy_stdnt!"

+ 0 What is the name of the hidden "history" file in the htb-user's home directory?

Submit your answer here...

+10 Streak pts

Submit

+ 1 What is the index number of the "sudoers" file in the "/etc" directory?

Submit your answer here...

+10 Streak pts

Submit

To go through these tasks again, first connect to the vpn of Hackthebox, then ssh to the target machine.

Then to find the hidden files, type `ls -a`.

```
htb-student@nixfund:~$ ls -a
.      .bash_history  .bashrc      .gnupg
..     .bash_logout  .cache       .profile
htb-student@nixfund:~$
```

The answer is `.bash_history`

SSH to 10.129.2.219 (ACADEMY-NIXFUND) with user "htb-student" and password "HTB_@cademy_stdnt!"

+ 0 What is the name of the hidden "history" file in the htb-user's home directory?

.bash_history

Submit

Next Question. To find the index of a file, simply type `ls -i`

```
htb-student@nixfund:~$ ls -i /etc/sudoers
147627 /etc/sudoers
htb-student@nixfund:~$
```

Here, the index is 147627.

+ 1 What is the index number of the "sudoers" file in the "/etc" directory?

147627

Submit

Working with Files and Directories

The primary difference between working with files in Linux, as opposed to Windows, lies in how we access and manage those files. In Windows, we typically use graphical tools like Explorer to find, open, and edit files. However, in Linux, the terminal offers a powerful alternative where files can be accessed and edited directly using commands. This method is not only faster, but also more efficient, as it allows you to edit files interactively without even needing editors like `vim` or `nano`.

The terminal's efficiency stems from its ability to access files with just a few commands, and it allows you to modify files selectively using regular expressions (`regex`). Additionally, you can run multiple commands at once, redirecting output

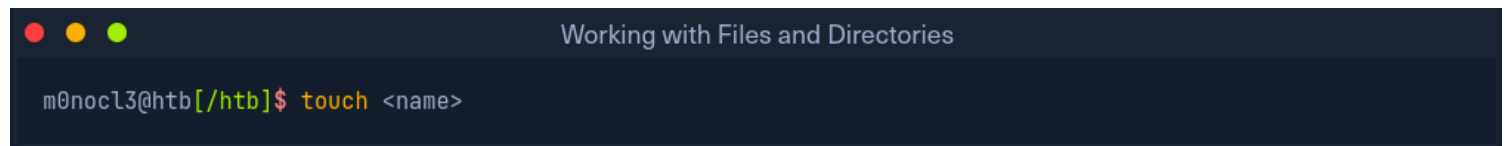
to files and automating batch editing tasks, which is a major time-saver when working with numerous files simultaneously. This command-line approach streamlines workflow, making it an invaluable tool for tasks that would be more time-consuming through a graphical interface.

Next, we will explore working with files and directories to effectively manage the content on our operating system.

Create, Move, and Copy

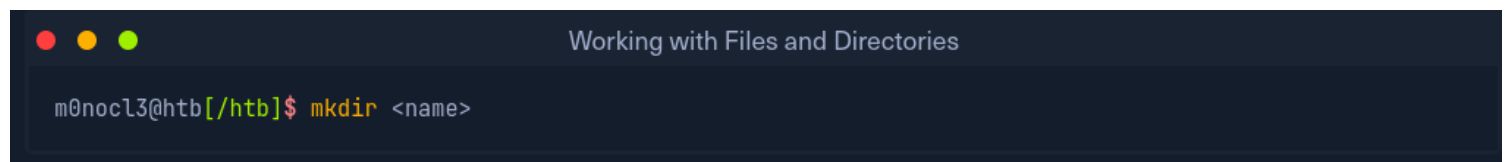
Let us begin by learning how to perform key operations like creating, renaming, moving, copying, and deleting files. Before we execute the following commands, we first need to SSH into the target (using the connection instructions at the bottom of the section). Now, let's say we want to create a new file or directory. The syntax for this is the following:

Syntax - touch

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The title bar reads "Working with Files and Directories". The terminal shows a prompt "m0noc13@htb[/htb]" followed by the command "touch <name>".

```
m0noc13@htb[/htb]$ touch <name>
```

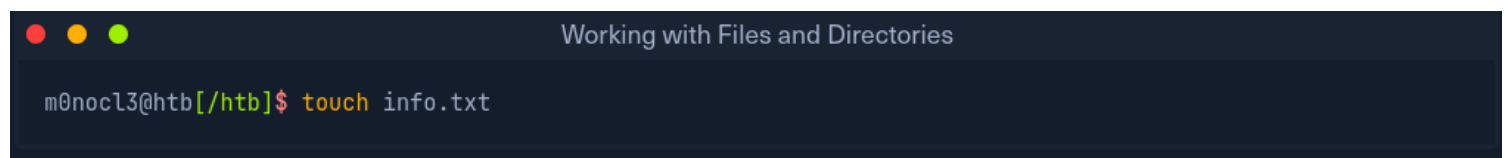
Syntax - mkdir

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The title bar reads "Working with Files and Directories". The terminal shows a prompt "m0noc13@htb[/htb]" followed by the command "mkdir <name>".

```
m0noc13@htb[/htb]$ mkdir <name>
```

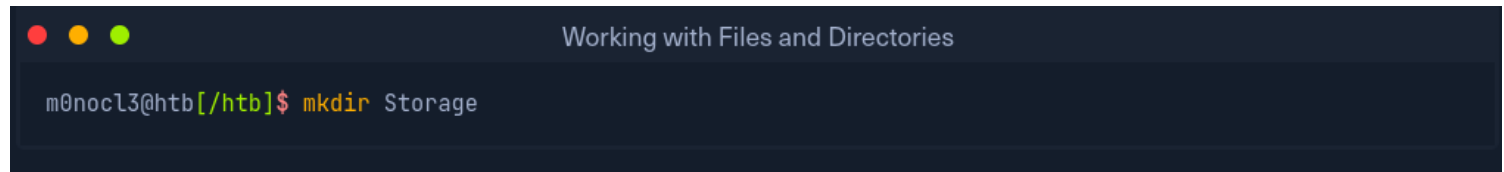
In the next example, we will create a file called info.txt and a directory called Storage. To create these, we follow the commands and their syntax as shown above.

Create an Empty File

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The title bar reads "Working with Files and Directories". The terminal shows a prompt "m0noc13@htb[/htb]" followed by the command "touch info.txt".

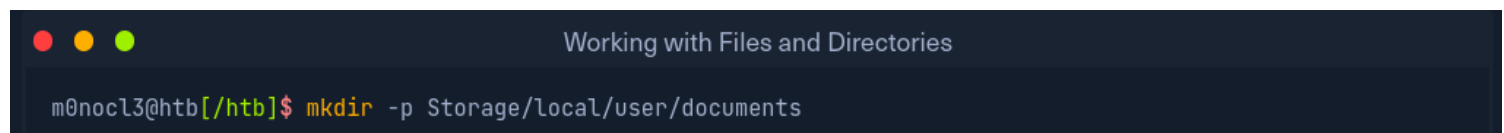
```
m0noc13@htb[/htb]$ touch info.txt
```

Create a Directory

A terminal window titled "Working with Files and Directories" with a dark background. It shows a command prompt where the user has entered the command to create a directory named "Storage".

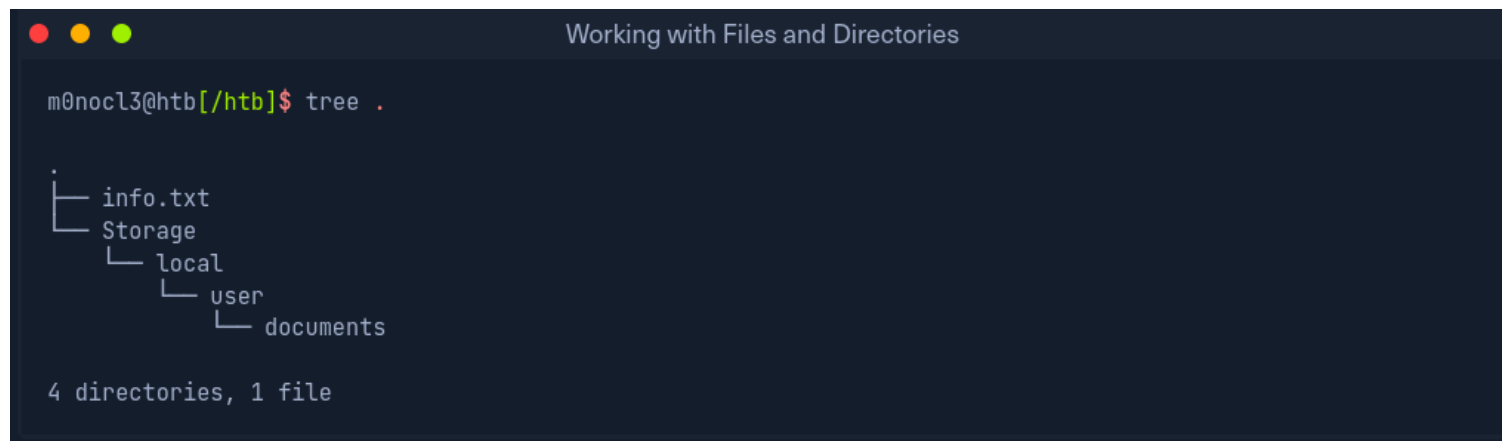
```
m0noc13@htb[/htb]$ mkdir Storage
```

When organizing your system, you may need to create multiple directories within other directories. Manually running the `mkdir` command for each one would be time-consuming. Fortunately, the `mkdir` command has the `-p` (parents) option, which allows you to create parent directories automatically.

A terminal window titled "Working with Files and Directories" with a dark background. It shows a command prompt where the user has entered the command to create a nested directory structure using the -p option.

```
m0noc13@htb[/htb]$ mkdir -p Storage/local/user/documents
```

We can look at the whole structure after creating the parent directories with the tool `tree`.

A terminal window titled "Working with Files and Directories" with a dark background. It shows a command prompt where the user has entered the command to display the directory tree. The output shows a tree structure with a file 'info.txt' and a nested directory structure 'Storage/local/user/documents'.

```
m0noc13@htb[/htb]$ tree .  
  
├── info.txt  
└── Storage  
    ├── local  
    │   ├── user  
    │   └── documents  
    └──   
  
4 directories, 1 file
```

You can create files directly within specific directories by specifying the path where the file should be stored, and you can use the single dot (`.`) to indicate that you want to start from the current directory. This is a convenient way to work within your current location, without needing to type the full path. Therefore, the command for creating another empty file looks like this:

Create `userinfo.txt`

```
Working with Files and Directories

m0nocl3@htb[/htb]$ touch ./Storage/local/user/userinfo.txt
```

```
Working with Files and Directories

m0nocl3@htb[/htb]$ tree .

├── info.txt
└── Storage
    ├── local
    │   └── user
    │       ├── documents
    │       └── userinfo.txt
    └── user

4 directories, 2 files
```

With the command **mv**, we can move and also rename files and directories. The syntax for this looks like this:

Syntax - mv

```
Working with Files and Directories

m0nocl3@htb[/htb]$ mv <file/directory> <renamed file/directory>
```

First, let us rename the file info.txt to information.txt and then move it to the directory Storage.

```
Working with Files and Directories

m0nocl3@htb[/htb]$ mv info.txt information.txt
```

Now let us create a file named readme.txt in the current directory and then copy the files information.txt and readme.txt into the Storage/ directory.

Create readme.txt

```
m0nocl3@htb[/htb]$ touch readme.txt
```

Move Files to Specific Directory

```
m0nocl3@htb[/htb]$ mv information.txt readme.txt Storage/
```

```
m0nocl3@htb[/htb]$ tree .
```

```
.
├── Storage
│   ├── information.txt
│   ├── local
│   │   └── user
│   │       ├── documents
│   │       └── userinfo.txt
│   └── readme.txt
```

```
4 directories, 3 files
```

Let us assume we want to have the `readme.txt` in the `local/` directory. Then we can copy them there with the paths specified.

Copy `readme.txt`

```
m0nocl3@htb[/htb]$ cp Storage/readme.txt Storage/local/
```

Now we can check if the file is thereby using the tool `tree` again.

```
Working with Files and Directories

m0noc13@htb[/htb]$ tree .
.
├── Storage
│   ├── information.txt
│   ├── local
│   │   ├── readme.txt
│   │   └── user
│   │       ├── documents
│   │       └── userinfo.txt
│   └── readme.txt
└──
```

4 directories, 4 files

In addition to basic file management commands, there are many other powerful ways to work with files in Linux, such as using redirection and text editors. Redirection allows you to manipulate the flow of input and output between commands and files, making tasks like creating or modifying files faster and more efficient. You can also use popular text editors like vim and nano for more interactive editing.

We will explore and discuss these methods in greater detail in later sections. As you become familiar with these techniques, you will gain more flexibility in how you create, edit, and manage files on your system.

? Optional Exercise:

Use the tools we've already learned to figure out how to delete files and directories. Keep in mind that online research is a valuable part of the learning process—it's not cheating. You're not being tested right now, but rather building your knowledge. Searching for solutions online can expose you to different approaches and alternative methods, giving you a broader understanding of how things work and helping you discover the most efficient ways to solve problems.

Questions

Answer the question(s) below to complete this Section and earn cubes!

[Cheat Sheet](#)[Download VPN
Connection File](#)

Target(s): [Click here to spawn the target system!](#)

SSH to with user "**htb-student**" and password "**HTB_@cademy_stdnt!**"

+ 0 What is the name of the last modified file in the `/var/backups` directory?

Submit your answer here...

+10 Streak pts

Submit

+ 1 What is the inode number of the `shadow.bak` file in the `/var/backups` directory?

Submit your answer here...

+10 Streak pts

Submit

The first question is asking how we can find the last modified file in the `/var/backups` directory.

To do this, we need to list all the files inside that directory by using the `ls` command. And then we will use its parameters `-l` and `-t`.

`-l`

Use a long listing format, showing permissions, number of hard links, owner, group, size, and modification time.

`-t`

Sort by modification time, newest first.

```
htb-student@nixfund:~$ ls -lt /var/backups
total 2160
-rw-r--r-- 1 root root 41872 Nov 12 2020 apt.extended_states.0
-rw-r--r-- 1 root root 4437 Nov 12 2020 apt.extended_states.1.
gz
-rw-r--r-- 1 root root 742750 Nov 11 2020 dpkg.status.0
-rw-r--r-- 1 root root 206270 Nov 11 2020 dpkg.status.1.gz
-rw-r--r-- 1 root root 206270 Nov 5 2020 dpkg.status.2.gz
-rw-r--r-- 1 root root 206270 Nov 5 2020 dpkg.status.3.gz
-rw-r--r-- 1 root root 206270 Nov 5 2020 dpkg.status.4.gz
-rw-r--r-- 1 root root 206270 Nov 5 2020 dpkg.status.5.gz
-rw-r--r-- 1 root root 206270 Nov 5 2020 dpkg.status.6.gz
-rw-r--r-- 1 root root 51200 Oct 29 2020 alternatives.tar.0
-rw-r--r-- 1 root root 4623 Oct 22 2020 apt.extended_states.2.
gz
-rw-r--r-- 1 root root 2497 Oct 16 2020 alternatives.tar.1.gz
-rw-r--r-- 1 root root 4601 Oct 15 2020 apt.extended_states.3.
gz
-rw-r--r-- 1 root root 2492 Sep 24 2020 alternatives.tar.2.gz
```

but we can also not use `-lt` and instead just `-t`. The `-l` parameter just gives more details.

```
htb-student@nixfund:~$ ls -t /var/backups
apt.extended_states.0      dpkg.statoverride.3.gz
apt.extended_states.1.gz  dpkg.statoverride.4.gz
dpkg.status.0             dpkg.statoverride.5.gz
dpkg.status.1.gz         dpkg.statoverride.6.gz
dpkg.status.2.gz         apt.extended_states.4.gz
dpkg.status.3.gz         passwd.bak
dpkg.status.4.gz         shadow.bak
dpkg.status.5.gz         gshadow.bak
dpkg.status.6.gz         group.bak
alternatives.tar.0       dpkg.diversions.0
apt.extended_states.2.gz  dpkg.diversions.1.gz
alternatives.tar.1.gz    dpkg.diversions.2.gz
apt.extended_states.3.gz  dpkg.diversions.3.gz
alternatives.tar.2.gz    dpkg.diversions.4.gz
dpkg.statoverride.0      dpkg.diversions.5.gz
dpkg.statoverride.1.gz   dpkg.diversions.6.gz
dpkg.statoverride.2.gz
htb-student@nixfund:~$
```

SSH to 10.129.27.193 (ACADEMY-NIXFUND) with user "htb-student" and password "HTB_@cademy_stdnt!"

+ 0 What is the name of the last modified file in the "/var/backups" directory?

apt.extended_states.0

Submit

The next question is asking for the inode number of the shadow.bak file in /var/backups.

To do this, we are gonna simply use the parameter `-i` from the `ls` command.

```
htb-student@nixfund:~$ ls -i /var/backups/shadow.bak
265293 /var/backups/shadow.bak
htb-student@nixfund:~$ ^C
htb-student@nixfund:~$
```

+ 1 What is the inode number of the "shadow.bak" file in the "/var/backups" directory?

265293

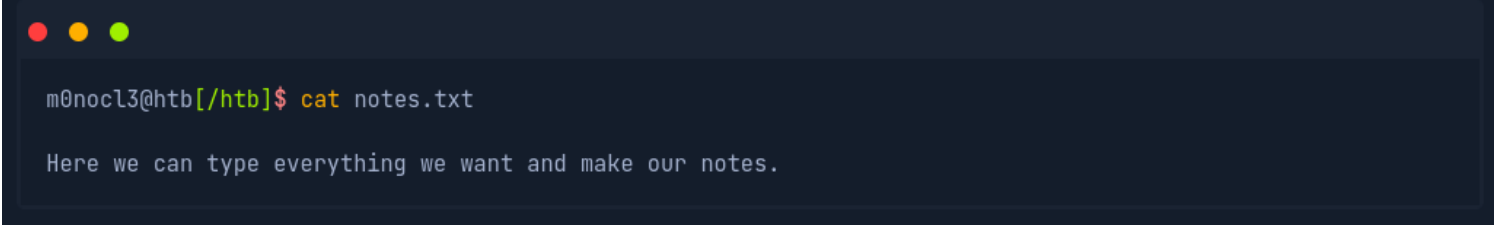
Submit

Editing Files

You can edit stuff on nano.

```
m0nocl3@htb[/htb]$ nano notes.txt
```

To view the file you edited on nano, use `cat`



```
m0noc13@htb[/htb]$ cat notes.txt
```

```
Here we can type everything we want and make our notes.
```

On Linux systems, there are several files that can be tremendously beneficial for penetration testers, due to misconfigured permissions or insufficient security settings by the administrators. One such important file is the `/etc/passwd` file. This file contains essential information about the users on the system, such as their usernames, user IDs (UIDs), group IDs (GIDs), and home directories.

Historically, the `/etc/passwd` file also stored password hashes, but now those hashes are typically stored in `/etc/shadow`, which has stricter permissions. However, if the permissions on `/etc/passwd` or other critical files are not set correctly, it may expose sensitive information or lead to privilege escalation opportunities.

As penetration testers, identifying files with improper rights or permissions can provide key insights into potential vulnerabilities that might be exploited, such as weak user accounts or misconfigured file access that should otherwise be restricted. Understanding these files is vital when assessing the security posture of a system.

VIM

Vim is an open-source editor for all kinds of ASCII text, just like Nano. It is an improved clone of the previous Vi. It is an extremely powerful editor that focuses on the essentials, namely editing text. For tasks that go beyond that, Vim provides an interface to external programs, such as `grep`, `awk`, `sed`, etc., which can handle their specific tasks much better than a corresponding function directly implemented in an editor usually can. This makes the editor small and compact, fast, powerful, flexible, and less error-prone.

Vim follows the Unix principle here: many small specialized programs that are well tested and proven, when combined and communicating with each other, resulting in a flexible and powerful system.


```
Find Files and Directories

m0nocl3@htb[/htb]$ which python

/usr/bin/python
```

If the program we search for does not exist, no results will be displayed.

Find

Another handy tool is find. Besides the function to find files and folders, this tool also contains the function to filter the results. We can use filter parameters like the size of the file or the date. We can also specify if we only search for files or folders.

Syntax - find

```
Find Files and Directories

m0nocl3@htb[/htb]$ find <location> <options>
```

Let us look at an example of what such a command with multiple options would look like.

```
Find Files and Directories

m0nocl3@htb[/htb]$ find / -type f -name *.conf -user root -size +20k -newermt 2020-03-03 -exec ls -al {} \;

-rw-r--r-- 1 root root 136392 Apr 25 20:29 /usr/src/linux-headers-5.5.0-1parrot1-amd64/include/config/auto.conf
-rw-r--r-- 1 root root 82290 Apr 25 20:29 /usr/src/linux-headers-5.5.0-1parrot1-amd64/include/config/tristate.conf
-rw-r--r-- 1 root root 95813 May 7 14:33 /usr/share/metasploit-framework/data/jtr/repeats32.conf
-rw-r--r-- 1 root root 60346 May 7 14:33 /usr/share/metasploit-framework/data/jtr/dynamic.conf
-rw-r--r-- 1 root root 96249 May 7 14:33 /usr/share/metasploit-framework/data/jtr/dumb32.conf
-rw-r--r-- 1 root root 54755 May 7 14:33 /usr/share/metasploit-framework/data/jtr/repeats16.conf
-rw-r--r-- 1 root root 22635 May 7 14:33 /usr/share/metasploit-framework/data/jtr/korelogic.conf
-rwxr-xr-x 1 root root 108534 May 7 14:33 /usr/share/metasploit-framework/data/jtr/john.conf
-rw-r--r-- 1 root root 55285 May 7 14:33 /usr/share/metasploit-framework/data/jtr/dumb16.conf
-rw-r--r-- 1 root root 21254 May 2 11:59 /usr/share/doc/sqlmap/examples/sqlmap.conf
-rw-r--r-- 1 root root 25086 Mar 4 22:04 /etc/dnsmasq.conf
-rw-r--r-- 1 root root 21254 May 2 11:59 /etc/sqlmap/sqlmap.conf
```

Now let us take a closer look at the options we used in the previous command. If we hover the mouse over the respective options, a small window will appear with

an explanation. These explanations will also be found in other modules, which should help us if we are not yet familiar with one of the tools.

Option	Description
<code>-type f</code>	Hereby, we define the type of the searched object. In this case, 'f' stands for 'file'.
<code>-name *.conf</code>	With '-name', we indicate the name of the file we are looking for. The asterisk (*) stands for 'all' files with the '.conf' extension.
<code>-user root</code>	This option filters all files whose owner is the root user.
<code>-size +20k</code>	We can then filter all the located files and specify that we only want to see the files that are larger than 20 KIB.
<code>-newermt 2020-03-03</code>	With this option, we set the date. Only files newer than the specified date will be presented.
<code>-exec ls -al {} \;</code>	This option executes the specified command, using the curly brackets as placeholders for each result. The backslash escapes the next character from being interpreted by the shell because otherwise, the semicolon would terminate the command and not reach the redirection.
<code>2>/dev/null</code>	This is a STDERR redirection to the 'null device', which we will come back to in the next section. This redirection ensures that no errors are displayed in the terminal. This redirection must not be an option of the 'find' command.

Locate

It will take much time to search through the whole system for our files and directories to perform many different searches. The command locate offers us a quicker way to search through the system. In contrast to the find command, locate works with a local database that contains all information about existing files and folders. We can update this database with the following command.

```
Find Files and Directories
m0noc13@htb[/htb]$ sudo updatedb
```

If we now search for all files with the ".conf" extension, you will find that this search produces results much faster than using find.

```
Find Files and Directories

m0n0cl3@htb[/htb]$ locate *.conf

/etc/GeoIP.conf
/etc/NetworkManager/NetworkManager.conf
/etc/UPower/UPower.conf
/etc/adduser.conf
<SNIP>
```

However, this tool does not have as many filter options that we can use. So it is always worth considering whether we can use the locate command or instead use the find command. It always depends on what we are looking for.

Optional Exercise:

Questions

Answer the question(s) below to complete this Section and earn cubes!

Target(s): **10.129.24.11** (ACADEMY-NIXFUND)

Life Left: 118 minute(s) + Terminate ×

SSH to 10.129.24.11 (ACADEMY-NIXFUND) with user "**htb-student**" and password "**HTB_@cademy_stdnt!**"

Cheat Sheet

Download VPN Connection File

+ 1 What is the name of the config file that has been created after 2020-03-03 and is smaller than 28k but larger than 25k?

Submit your answer here...

+10 Streak pts Submit

+ 1 How many files exist on the system that have the ".bak" extension?

Submit your answer here...

+10 Streak pts Submit

+ 0 Submit the full path of the "xxd" binary.

Submit your answer here...

+10 Streak pts Submit

SSH to 10.129.24.11 (ACADEMY-NIXFUND) with user "htb-student" and password "HTB_@cademy_stdnt!"

+ 1 📦 What is the name of the config file that has been created after 2020-03-03 and is smaller than 28k but larger than 25k?

00-mesa-defaults.conf

Submit

The first question asked to find a **config file**, that has been created after **2020-03-03** and is smaller than **28k** but larger than **25k**

To do this, we will use the **find** command

We will start finding at the root directory, so we type:

find / [options]

We apply these tests to filter results:

-type f -name "*.conf" -size +25k -size -28k -newermt 2020-03-03 2>/dev/null

-type f → only regular files

-name "*.conf" → filename ends with .conf

-size +25k → larger than 25 KB

-size -28k → smaller than 28 KB

-newermt 2020-03-03 → modified after 2020-03-03

We add **2>/dev/null** to discard error messages from directories we cannot read.

```
htb-student@nixfund:~$ find / -type f -name "*.conf" -size +25k -size -28k -newermt 2020-03-03 2>/dev/null
/usr/share/dirc.d/00-mesa-defaults.conf
```

so the answer is **00-mesa-defaults.conf**

+ 1 📦 How many files exist on the system that have the ".bak" extension?

Submit your answer here...

+10 Streak pts

Submit

The question asks to count all existing files with the .bak extension.
We use find starting at the root directory `/`.
We apply these tests:

`-type f` → only regular files

`-name "*.bak"` → filenames ending with .bak

We add `2>/dev/null` to discard permission errors.

Finally, we pipe the output to `wc -l` to count the number of matching files.

```
type -f -name "*.bak" 2>/dev/null | wc -l
```

+ 0 📦 Submit the full path of the "xxd" binary.


Submit your answer here...

+10 Streak pts

Submit

Simply use the `which` command and find `xxd`

```
htb-student@nixfund:~$ which xxd
/usr/bin/xxd
htb-student@nixfund:~$
```

+ 0  Submit the full path of the "xxd" binary.

```
/usr/bin/xxd
```

 Submit

- **find** searches the filesystem anywhere you tell it.
- **which** searches only the directories listed in \$PATH (places the shell looks for commands).

