

Quark Cybersecurity

Linux Week and Cyber Security Module 1:

-Harshwardhan Solanki



Index

1. [What is Linux and Command Line Interface](#)
2. [Commands Syntax](#)
3. [File Permissions](#)
4. [Ownership of Files](#)
5. [Obtaining Assistance](#)
6. [Adding Text To The File](#)
7. [Pipes](#)
8. [Standard Output To a File](#)
9. [File Maintenance Commands](#)
10. [File Display Commands](#)
11. [Filters and Text Processing Commands](#)
12. [Compare Files](#)
13. [Compress and Uncompress Files and Directories](#)
14. [Truncate File Size](#)
15. [Combining and Splitting Files](#)

What is Linux and Command Line Interface

- Linux is a free and open-source operating system based on Unix Operating System, it is the most popular operating system in servers, desktops, laptops, and mobiles.
- As it is open source it is not licensed by anyone and can be modified and distributed by anyone.
- It has many popular distributions like Ubuntu, Debian, Kali, Fedora, etc.
- Kali is used for penetration testing, forensics and security auditing. It is based on Debian and has many tools and utilities for testing the security of networks, applications and systems.
- A Command Line Interface is a text-based interface similar to GUI but in text form and is used to interact with Operating System.



cyberops.in/blog



Commands Syntax

- The Basic Linux command Syntax is as follows:

```
command [options] [arguments]
```

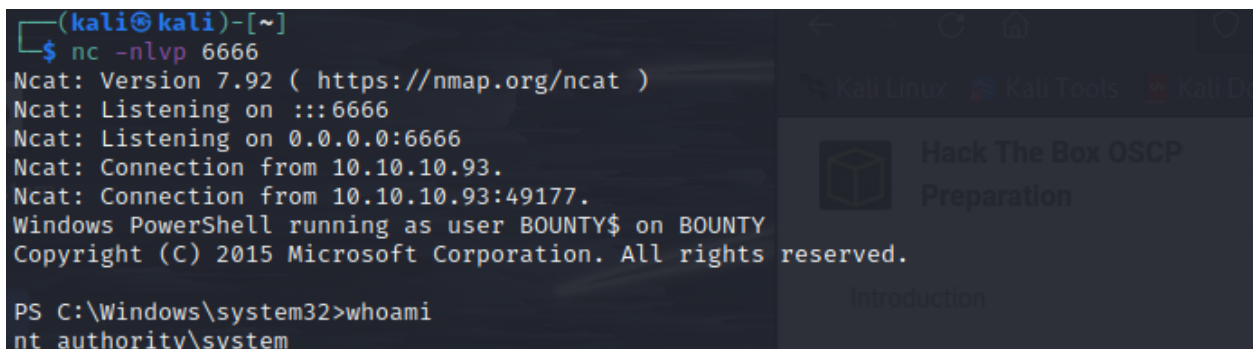
- Command:** The command here is the name of the command you want to execute eg. 'ls', 'sudo', etc.
- Options:** Flags that modify the behaviour of the command and usually start with a hyphen '-'
- Arguments:** The input you need your command to perform on like the name of a directory or a file.
- Here are some examples:

```
mv [options] [source] [destination]
```

```
chmod [options] [permissions] [file/directory]
```

```
ssh [options] [user@]hostname
```

Example Image



```
(kali㉿kali)-[~]  
$ nc -nlvp 6666  
Ncat: Version 7.92 ( https://nmap.org/ncat )  
Ncat: Listening on :::6666  
Ncat: Listening on 0.0.0.0:6666  
Ncat: Connection from 10.10.10.93.  
Ncat: Connection from 10.10.10.93:49177.  
Windows PowerShell running as user BOUNTY$ on BOUNTY.  
Copyright (C) 2015 Microsoft Corporation. All rights reserved.  
  
PS C:\Windows\system32>whoami  
nt authority\system
```

File Permissions

- File Permissions in Linux are defined as who can access the file and to what level.

```
chmod permissions filename
```

- There are three types of permissions: Read (r), Write (w) and Execute (x)
- There are also three types of users: Owner, Group, and everyone else.
- The permission is represented by a series of nine characters divided into groups of three, the first group is of the owner, the second of the group and the third of everyone else.
- The format looks like 'rwxrwxrwx' for better understanding we've highlighted the groups (orange-owner, green-group, blue-everyone else).
- The 'r', 'w', and 'x' can be replaced with a hyphen '-' if permission is not granted. For example, if the group does not have permission to write and everyone else does not have to execute and write the code should look like this 'rwxr-xr-'.
- To modify the permission of a file or directory we use 'chmod' which stands for change mode followed by a three-digit number and then the file or directory name. The first digit represents the owner, then the group and the last is everyone else.
- Each of these digits is calculated by adding the values of desired permission i.e. 4 for read, 2 for write and 1 for execute. Example. 'chmod 752 file.txt' the code sets the permissions for the owner as read, write and execute, for the group it is read and execute and for everyone else it is write.
- We can see the default set permissions by using the 'umask' command. By default for security and safety reasons in Unix systems, the execute permissions are not provided to newly created files.

Ownership of Files

- In Linux, every file and directory is owned by a user and a group. The owner is the user who created the file or directory, and the group is a set of users who have specific permissions to access the file or directory.
- The ownership can be viewed by using the 'ls -lrt' command and can be modified using the 'chown' command.

```
(kali㉿kali)-[~]  
$ ls -lrt CTF1.txt  
-rw-r--r-- 1 kali kali 371 Apr  1 13:40 CTF1.txt
```

- The owner is usually the one who created the file or directory and the group is a set of users who have specified permissions.
- You can change the ownership of the file by using the 'chown' command

```
chown user filename
```

- The ownership is passed using chown then writing the name of the user you want to transfer the ownership then the file name.

```
(kali㉿kali)-[~]  
$ sudo chown harsh CTF1.txt  
  
(kali㉿kali)-[~]  
$ ls -lrt CTF1.txt  
-rwxrwx-w- 1 harsh kali 371 Apr  1 13:40 CTF1.txt
```

- The 'chgrp' command is used to change the group ownership of a file or directory.

```
chgrp [OPTIONS] NEW_GROUP FILE...
```

- Here NEW-GROUP is the group name you want to transfer the ownership to and FILE is the file name, there are some options available with 'chgrp' they



are:

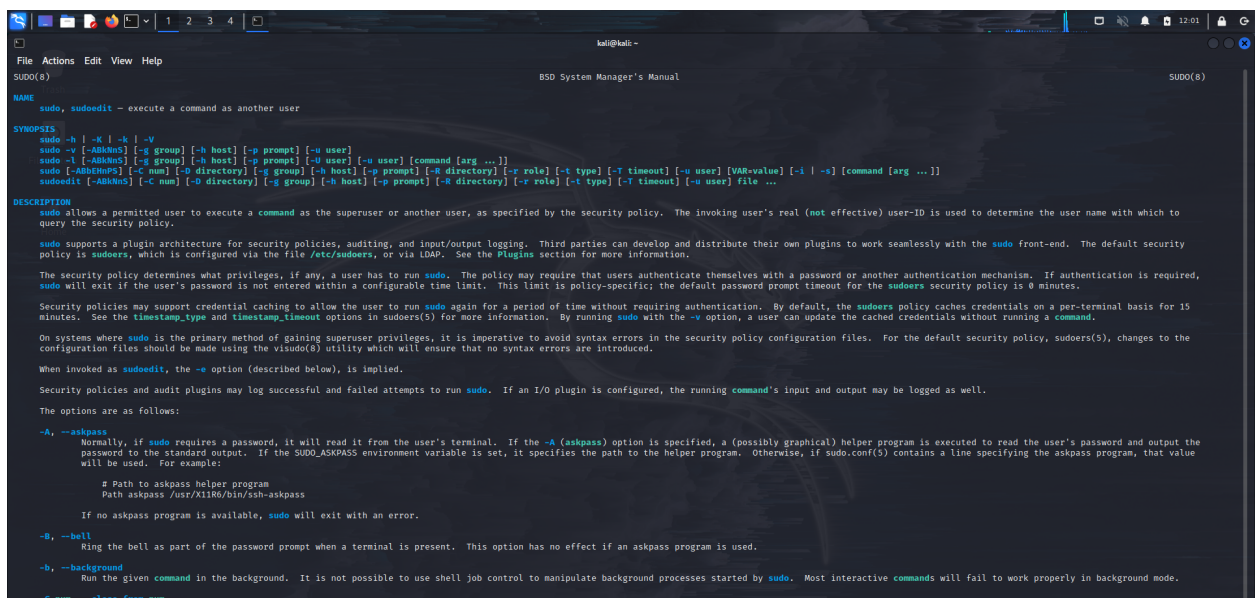
-R: Recursively changes the group ownership of all files and directories under the specified directory.

--reference=file: Set the group ownership of the specified files or directories to match the group ownership of the file.

Obtaining Assistance

- There are many ways to obtain assistance in Linux for commands and programs here are some of them:
- **Man:** It stands for manual and displays manual pages for the specified command or program.
- **–help:** It is used after you've written your command as it is an argument, it displays a brief help message.
- **whatis:** Is used to provide a small description of the command or program.

Manual for 'sudo' command



```
sudo(8)                                BSD System Manager's Manual                                sudo(8)

NAME
  sudo, sudoedit - execute a command as another user

SYNOPSIS
  sudo -h | -K | -k | -V
  sudo -v [-ABbHns] [-g group] [-h host] [-p prompt] [-u user]
  sudo -l [-ABbHns] [-g group] [-h host] [-p prompt] [-U user] [command [arg ...]]
  sudo [-ABbHnPS] [-C num] [-D directory] [-g group] [-h host] [-p prompt] [-R role] [-t type] [-T timeout] [-u user] [VAR=value] [-i | -s] [command [arg ...]]
  sudoedit [-ABbHns] [-C num] [-D directory] [-g group] [-h host] [-p prompt] [-R directory] [-r role] [-t type] [-T timeout] [-u user] file ...

DESCRIPTION
  sudo allows a permitted user to execute a command as the superuser or another user, as specified by the security policy. The invoking user's real (not effective) user-ID is used to determine the user name with which to query the security policy.

  sudo supports a plugin architecture for security policies, auditing, and input/output logging. Third parties can develop and distribute their own plugins to work seamlessly with the sudo front-end. The default security policy is sudoers, which is configured via the file /etc/sudoers, or via LDAP. See the Plugins section for more information.

  The security policy determines what privileges, if any, a user has to run sudo. The policy may require that users authenticate themselves with a password or another authentication mechanism. If authentication is required, sudo will exit if the user's password is not entered within a configurable time limit. This limit is policy-specific; the default password prompt timeout for the sudoers security policy is 5 minutes.

  Security policies may support credential caching to allow the user to run sudo again for a period of time without requiring authentication. By default, the sudoers policy caches credentials on a per-terminal basis for 15 minutes. See the timestamp_type and timestamp_timeout options in sudoers(5) for more information. By running sudo with the -s option, a user can update the cached credentials without running a command.

  On systems where sudo is the primary method of gaining superuser privileges, it is imperative to avoid syntax errors in the security policy configuration files. For the default security policy, sudoers(5), changes to the configuration files should be made using the visudo(8) utility which will ensure that no syntax errors are introduced.

  When invoked as sudoedit, the -e option (described below), is implied.

  Security policies and audit plugins may log successful and failed attempts to run sudo. If an I/O plugin is configured, the running command's input and output may be logged as well.

  The options are as follows:

  -A, --askpass
      Normally, if sudo requires a password, it will read it from the user's terminal. If the -A (askpass) option is specified, a (possibly graphical) helper program is executed to read the user's password and output the password to the standard output. If the SUDO_ASKPASS environment variable is set, it specifies the path to the helper program. Otherwise, if sudo.conf(5) contains a line specifying the askpass program, that value will be used. For example:

          # Path to askpass helper program
          Path askpass /usr/X11R6/bin/ssh-askpass

      If no askpass program is available, sudo will exit with an error.

  -B, --bell
      Ring the bell as part of the password prompt when a terminal is present. This option has no effect if an askpass program is used.

  -b, --background
      Run the given command in the background. It is not possible to use shell job control to manipulate background processes started by sudo. Most interactive commands will fail to work properly in background mode.

  -C num --close-from-num
```


‘sudo’ –help

```
(kali@kali)-[~]
└─$ sudo --help
sudo - execute a command as another user

usage: sudo -h | -K | -k | -V
usage: sudo -v [-ABkNns] [-g group] [-h host] [-p prompt] [-u user]
usage: sudo -l [-ABkNns] [-g group] [-h host] [-p prompt] [-u user] [command [arg ...]]
usage: sudo [-ABEknPS] [-r role] [-t type] [-C num] [-D directory] [-g group] [-h host] [-p prompt] [-R directory] [-T timeout] [-u user] [VAR=value] [-i | -s] [command [arg ...]]
usage: sudo -e [-ABkNns] [-r role] [-t type] [-C num] [-D directory] [-g group] [-h host] [-p prompt] [-R directory] [-T timeout] [-u user] file ...

Options:
-A, --askpass          use a helper program for password prompting
-B, --background      run command in the background
-b, --bell             ring bell when prompting
-C, --close-from-num   close all file descriptors >= num
-D, --chdir=directory  change the working directory before running command
-E, --preserve-env      preserve user environment when running command
    --preserve-env=list preserve specific environment variables
-e, --edit             edit files instead of running a command
-g, --group=group       run command as the specified group name or ID
-H, --set-home          set HOME variable to target user's home dir
-h, --help             display help message and exit
-H, --host=host         run command on host (if supported by plugin)
-i, --login             run login shell as the target user; a command may also be specified
-K, --remove-timestamp remove timestamp file completely
-k, --reset-timestamp  invalidate timestamp file
-l, --list             list user's privileges or check a specific command; use twice for longer format
-n, --non-interactive  non-interactive mode, no prompts are used
-P, --preserve-groups   preserve group vector instead of setting to target's
-p, --prompt=prompt    use the specified password prompt
-R, --chroot=directory change the root directory before running command
-r, --role=role         create SELinux security context with specified role
-S, --stdin            read password from standard input
-s, --shell            run shell as the target user; a command may also be specified
-t, --type=type        create SELinux security context with specified type
-T, --command-timeout=timeout terminate command after the specified time limit
-U, --other-user=user   in list mode, display privileges for user
-u, --user=user         run command (or edit file) as specified user name or ID
-V, --version          display version information and exit
-v, --validate         update user's timestamp without running a command
-                      stop processing command line arguments
```

‘Whatis’ sudo

```
(kali@kali)-[~]
└─$ whatis sudo
sudo (8) - execute a command as another user
```

Adding Text To The File

- There are many ways to add text to a file and here are some of them:
 - a.) echo
 - b.) printf
- You can also use editors in Linux like Nano, Vi, or Emacs
 - a.) nano is a simple and beginner-friendly text editor.
 - b.) vi is an advanced level text editor and complex but once mastered it can be a very powerful tool.
 - c.) emacs is used by many programmers and is highly customizable and also supports many programming languages.

Echo

```
(kali㉿kali)-[~]  
$ echo "hello world" >> CTF1.txt  
  
(kali㉿kali)-[~]  
$ open CTF1.txt
```

```
1 # Nmap 7.93 scan initiated Sat Apr  1 13:40:22 2023 as: nmap -p 80 -oG CTF1.txt 10.10.10.10  
2 Host: 10.10.10.10 (ec2-10-10-10-10.ap-south-1.compute.amazonaws.com) Status: Up  
3 Host: 10.10.10.10 (ec2-10-10-10-10.ap-south-1.compute.amazonaws.com) Ports: 80/open/tcp/http///  
4 # Nmap done at Sat Apr  1 13:40:22 2023 -- 1 IP address (1 host up) scanned in 0.13 seconds  
5 hello world
```

Printf

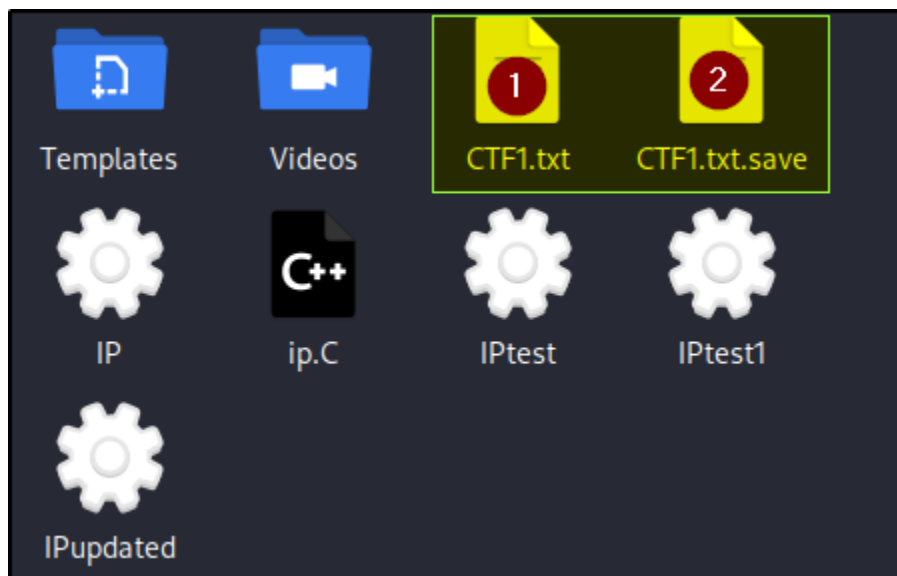
```
(kali㉿kali)-[~]  
$ printf "FSociety" >> CTF1.txt  
  
(kali㉿kali)-[~]  
$ open CTF1.txt
```

```
1 # Nmap 7.93 scan initiated Sat Apr  1 13:40:22 2023 as: nmap -p 80 -oG CTF1.txt
2 Host: 10.10.10.10 (10.10.10.10) Status: Up
3 Host: 10.10.10.10 (10.10.10.10) Ports: 80/open/tcp//http//
4 # Nmap done at Sat Apr  1 13:40:22 2023 -- 1 IP address (1 host up) scanned in 0.13 seconds
5 hello world
6 FSociety
```

Pipes

- Pipes (|) are used to connect the output of one command to the input of another this allows us to chain multiple commands and perform more complex tasks.
- Just as the name suggests it can be also called a pipeline as we take one input and feed it to another and so on so it is like a continuous pipeline.

```
(kali㉿kali)-[~]  
$ ls | grep "CTF1.txt"  
CTF1.txt  
CTF1.txt.save  
  
(kali㉿kali)-[~]  
$ open CTF1.txt
```



- As we can see above the 'ls' lists all the files in the directory and 'grep' searches for the file or line named CTF1.txt, 'ls' lists the file CTF1.txt in all the files the grep takes that file and stores it named as CTF1.txt.save.

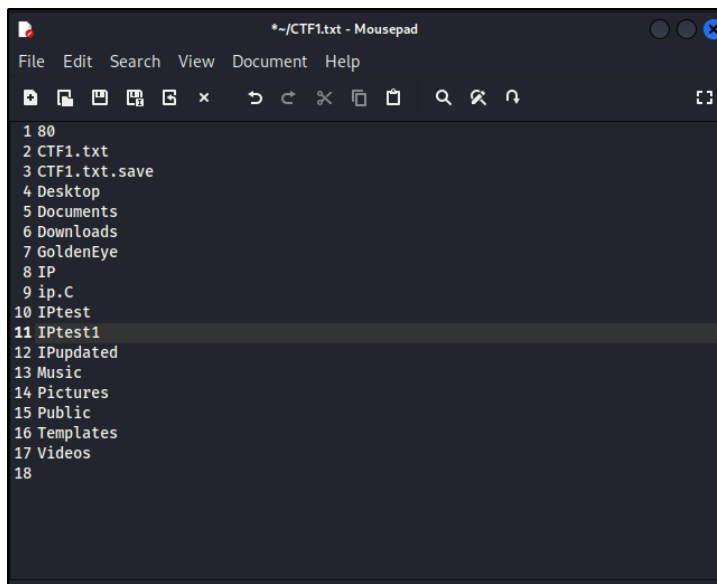
Standard Output To a File

- The tee command in Linux is used to read from standard input and write to standard output and one or more files simultaneously. It allows you to both display and store the output of a command or series of commands.

```
command | tee filename
```

- This is the basic syntax of the tee command, the 'command' is the command you want to insert then we use a pipe with 'tee' to write that output to the terminal and save it in a file.

```
(kali㉿kali)-[~]  
$ ls | tee "CTF1.txt"  
80  
CTF1.txt  
CTF1.txt.save  
Desktop  
Documents  
Downloads  
GoldenEye  
IP  
ip.C  
IPtest  
IPtest1  
IPupdated  
Music  
Pictures  
Public  
Templates  
Videos
```





File Maintenance Commands

- There are many file maintenance commands here are a few of them
- cp: The cp command is used to copy files and directories from one location to another.
- mkdir: The mkdir command is used to create a new directory
- rmdir: The rmdir command is used to remove a directory. Note that the directory must be empty before it can be removed
- find: The find command is used to search for files and directories based on various criteria, such as name, size, or date modified
- mv: The mv command is used to move or rename files and directories
- rm: The rm command is used to remove files and directories. Be careful when using this command, as it permanently deletes the specified files and directories

Etc.

File Display Commands

In Linux, several file display commands can be used to view the contents of files. Here are some of the most commonly used file display commands:

- **cat:** The cat command is used to display the contents of one or more files to the standard output
- **less:** The less command is used to display the contents of a file one page at a time. This command is useful for viewing large files
- **more:** The more command is similar to less, but it displays the contents of a file one screen at a time
- **head:** The head command is used to display the first few lines of a file. By default, it displays the first 10 lines
- **tail:** The tail command is used to display the last few lines of a file. By default, it displays the last 10 lines
- **awk:** The awk command is a powerful text-processing tool that can be used to manipulate and display the contents of a file

```
(kali@kali)-[~]
$ head GoldenEye/README.md
# THIS PROJECT IS NO LONGER MAINTAINED

# GoldenEye

GoldenEye is a Python 3 app for SECURITY TESTING PURPOSES ONLY!

GoldenEye is an HTTP DoS Test Tool.

Attack Vector exploited: HTTP Keep Alive + NoCache
```

```
(kali@kali)-[~]
$ tail GoldenEye/README.md
## To-do
* Change from getopt to argparse
* Change from string.format() to printf-like

## License
This software is distributed under the GNU General Public License version 3 (GPLv3)

## LEGAL NOTICE
THIS SOFTWARE IS PROVIDED FOR EDUCATIONAL USE ONLY! IF YOU ENGAGE IN ANY ILLEGAL ACTIVITY THE AUTHOR DOES NOT TAKE ANY RESPONSIBILITY FOR IT. BY USING THIS SOFTWARE YOU AGREE WITH THESE TERMS.
```


Filters and Text Processing Commands

In Linux, there are several filters and text processing commands that can be used to manipulate the contents of files.

- **grep:** The grep command is used to search for a specific pattern in a file and display only the matching lines
- **awk:** The awk command is a powerful text-processing tool that can be used to manipulate and display the contents of a file. It can be used to perform operations such as filtering, sorting, and formatting
- **cut:** The cut command is used to extract specific columns or fields from a file
- **sort:** The sort command is used to sort the contents of a file alphabetically or numerically
- **Wc:** It is used to display info about the number of lines, words and characters in a file.

```
(kali㉿kali)-[~]  
$ wc GoldenEye/README.md  
45  268 2147 GoldenEye/README.md
```

Here 45 is the number of lines, 268 is the number of words and 2147 is the number of characters.

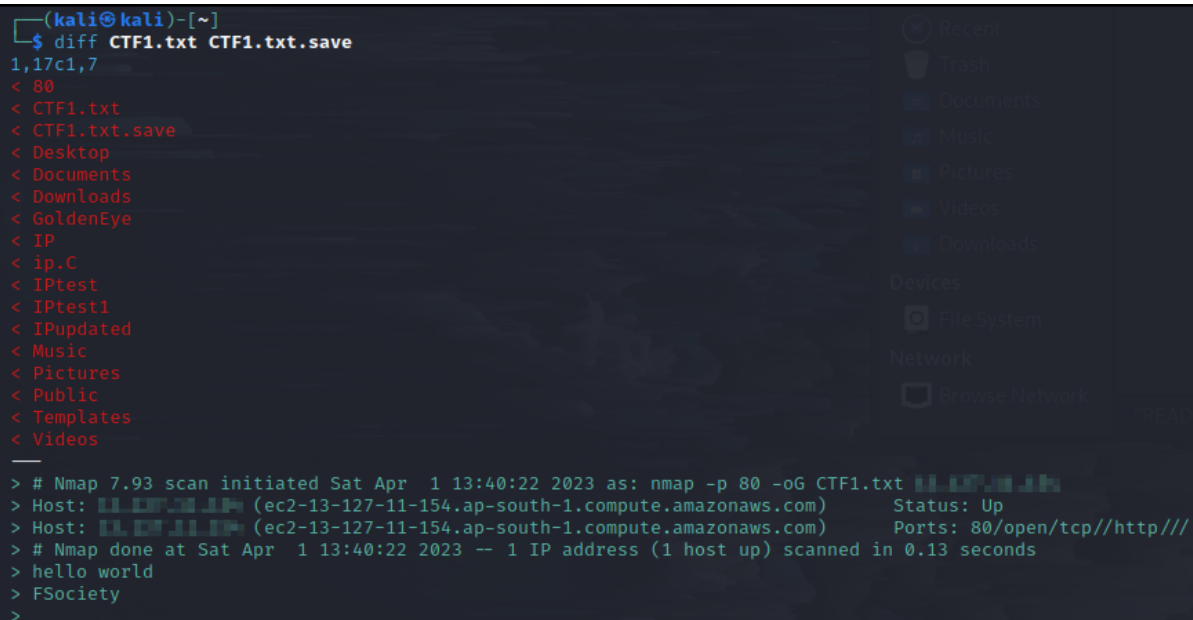
```
(kali㉿kali)-[~]  
$ grep -E "hello" CTF1.txt.save  
hello world
```

Here method '-E' is used to find the string in the file and the command grep finds it and prints it on the terminal.

Compare Files

In Linux, several commands can be used to compare the contents of two files. Here are some of the most commonly used commands:

- diff: The diff command is used to display the differences between two files. It displays the lines that are different in each file and indicates the changes that were made
- cmp: The cmp command is used to compare two files byte by byte and display the first byte and line number where they differ
- comm: The comm command is used to compare two sorted files line by line and display the lines that are unique to each file and the lines that are common to both files
- sdiff: The sdiff command is used to display the differences between two files side by side

A terminal window on a Kali Linux system. The prompt is (kali@kali)-[~]. The user enters \$ diff CTF1.txt CTF1.txt.save. The output shows a diff between the two files, indicating a change at line 1, column 17. The user then enters \$ nmap -p 80 -oG CTF1.txt. The output shows the results of an Nmap scan on the host ec2-13-127-11-154.ap-south-1.compute.amazonaws.com, showing that port 80 is open and the status is up. The user then enters > hello world and > FSociety, which are echoed back. The terminal also shows a sidebar with file system navigation options like Recent, Trash, Documents, Music, Pictures, Videos, Downloads, Devices, File system, Network, and Browse Network.

```
(kali@kali)-[~]
$ diff CTF1.txt CTF1.txt.save
1,17c1,7
< 80
< CTF1.txt
< CTF1.txt.save
< Desktop
< Documents
< Downloads
< GoldenEye
< IP
< ip.C
< IPtest
< IPtest1
< IPupdated
< Music
< Pictures
< Public
< Templates
< Videos

> # Nmap 7.93 scan initiated Sat Apr  1 13:40:22 2023 as: nmap -p 80 -oG CTF1.txt
> Host:  (ec2-13-127-11-154.ap-south-1.compute.amazonaws.com) Status: Up
> Host:  (ec2-13-127-11-154.ap-south-1.compute.amazonaws.com) Ports: 80/open/tcp///http///
> # Nmap done at Sat Apr  1 13:40:22 2023 -- 1 IP address (1 host up) scanned in 0.13 seconds
> hello world
> FSociety
>
```

Compress and Uncompress Files and Directories

In Linux, several commands can be used to compress and uncompress files and directories. Here are some of the most commonly used commands:

- tar: The tar command is used to create, manipulate, and extract tar archives, which are commonly used to package files and directories
- gzip: The gzip command is used to compress files, typically reducing the file size by 50-70%
- bzip2: The bzip2 command is used to compress files, typically reducing the file size by 50-75% while providing better compression than gzip
- zip: The zip command is used to compress and package files into a ZIP archive
- unzip: The unzip command is used to extract files from a ZIP archive

```
(kali㉿kali)-[~]  
$ zip myfiles.zip CTF1.txt  
adding: CTF1.txt (deflated 97%)
```

Truncate File Size

- In Linux, you can use the truncate command to change the size of a file. The truncate command is used to shrink or extend the size of a file to a specified size.

```
truncate [OPTION]... FILENAME
```

- This is the basic syntax
- To reduce a file size we use the '-c' argument
- To create a new file with a specified size we can use '-s' after '-c'.

```
(kali㉿kali)-[~]
$ truncate --size 200 CTF1.txt

(kali㉿kali)-[~]
$ ls -lh CTF1.txt
-rwxrwx-w- 1 harsh kali 200 Apr  3 15:47 CTF1.txt

(kali㉿kali)-[~]
$ truncate --size 1 CTF1.txt

(kali㉿kali)-[~]
$ ls -lh CTF1.txt
-rwxrwx-w- 1 harsh kali 1 Apr  3 15:48 CTF1.txt
```

Combining and Splitting Files

- To combine files we use the 'cat' command followed by the file1 name then the file2 name and then to store the output we use the '>' operator followed by the name of our new file.
- For Splitting we use the 'split' command in Linux followed by the file name without any arguments it will by default split the file into 1,00 line chunks.
- The smaller files then will be named by default using the prefix 'x'.
- To split the file into specific bytes we can use '-b' arguments with a max size of 10MB.

Cat

```
(kali㉿kali)-[~]  
$ cat CTF1.txt CTF1.txt.save > hello  
  
(kali㉿kali)-[~]  
$ open hello
```

```
1 |C# Nmap 7.93 scan initiated Sat Apr  1 13:40:22 2023 as: nmap -p 80 -oG  
   CTF1.txt 15.127.11.154  
2 Host: 15.127.11.154 (ec2-13-127-11-154.ap-south-1.compute.amazonaws.com)  
   Status: Up  
3 Host: 15.127.11.154 (ec2-13-127-11-154.ap-south-1.compute.amazonaws.com)  
   Ports: 80/open/tcp///http///  
4 # Nmap done at Sat Apr  1 13:40:22 2023 -- 1 IP address (1 host up) scanned  
   in 0.13 seconds  
5 hello world  
6 FSociety  
7  
8
```

Split

```
(kali㉿kali)-[~]  
$ split CTF1.txt  
  
(kali㉿kali)-[~]  
$ split -b 100 CTF1.txt.save
```

