Spring 2025 Cyber Security Fundamentals (INFT-3508 - 20332)

Aysel Panahova

Quiz 3

Key Learnings from Chapter 3 of the CTF Primer

In these chapters, I learned about Forensics, commands for searching for strings and filenames, and Sleuthkit tools. Forensics, in simple terms, is a process of solving a mystery after an event happens, such as a crime. In cybersecurity, it is called Digital Forensics. It is about investigating computers to determine how a security violation occurred. This helps in two ways: Law enforcement (finding out if someone is guilty or innocent) and Security improvement (Learning from past attacks to prevent them in the future).

Moreover, I learned several useful commands to find specific words in a file or locate a file by name. For instance, if I need to search for a specific word in a file, I should first download the file using the “wget” command and copy the file’s address. Then, we need to check the file type to see what kind of file it is with the” file” command. To show the contents of the particular file we can use the “cat” command. Additionally, we can use the “grep” command to search for the needed word in a file.

If I want to find a file by name, I should also first download the zipped file using the “wget” command and unzip it with the “gunzip.” If I am looking for a file named “cybersecurity.txt” I should write the command as “find. -name cybersecuriyu.txt”. Then I need to go to the file’s folder using “cd” and view the contents that this file contains using the “cat” command again.

In section 3.3, I learned that disk analysis is one of the most essential skills in Digital Forensics. Analysts examine disks, which can be physical hardware or disk images that are stored as files. The command-line tools from Sleuthkit ensure that forensic analysis gain a deeper understanding of disk structures rather than relying on GUI tools that might hide some major details.

I learned several useful commands from this chapter. The first one is the “nc” (nectat) command which is used to communicate with a remote server. The structure of this command is like “nc [server name] [port number]”.

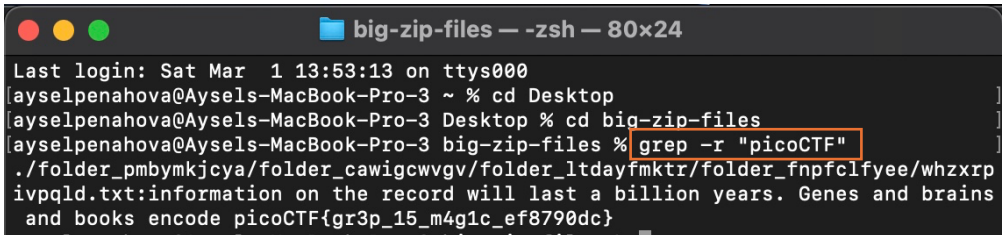
Moreover, because disk images are large files, navigating them manually is quite complicated. Therefore, Sleuthkit tools are needed to interpret it. Disk images are divided into four layers: Media Layer, Block Layer, Inode Layer, and Filename Layer. The Media Layer manages storage layout, and tools in this layer start with “mm” (for instance, “mmls”), which shows partition tables. The Block Layer breaks data into blocks, and tools here start with “blk.” The Inode Layer maps file locations. The tools in this layer start with “i” (for example, “icat”), which extract files by inode. Lastly, the Filename Layer shows files or directories. Tools in this layer start with “f” (for example, “fls”), which lists file names and locations.

In conclusion, digital forensics involves analyzing data to uncover important information, and disk analysis plays a key role in this process. By using different useful tools to break down disk images into manageable layers, analysts can understand and extract data better to improve security.

PicoCTF Assignment

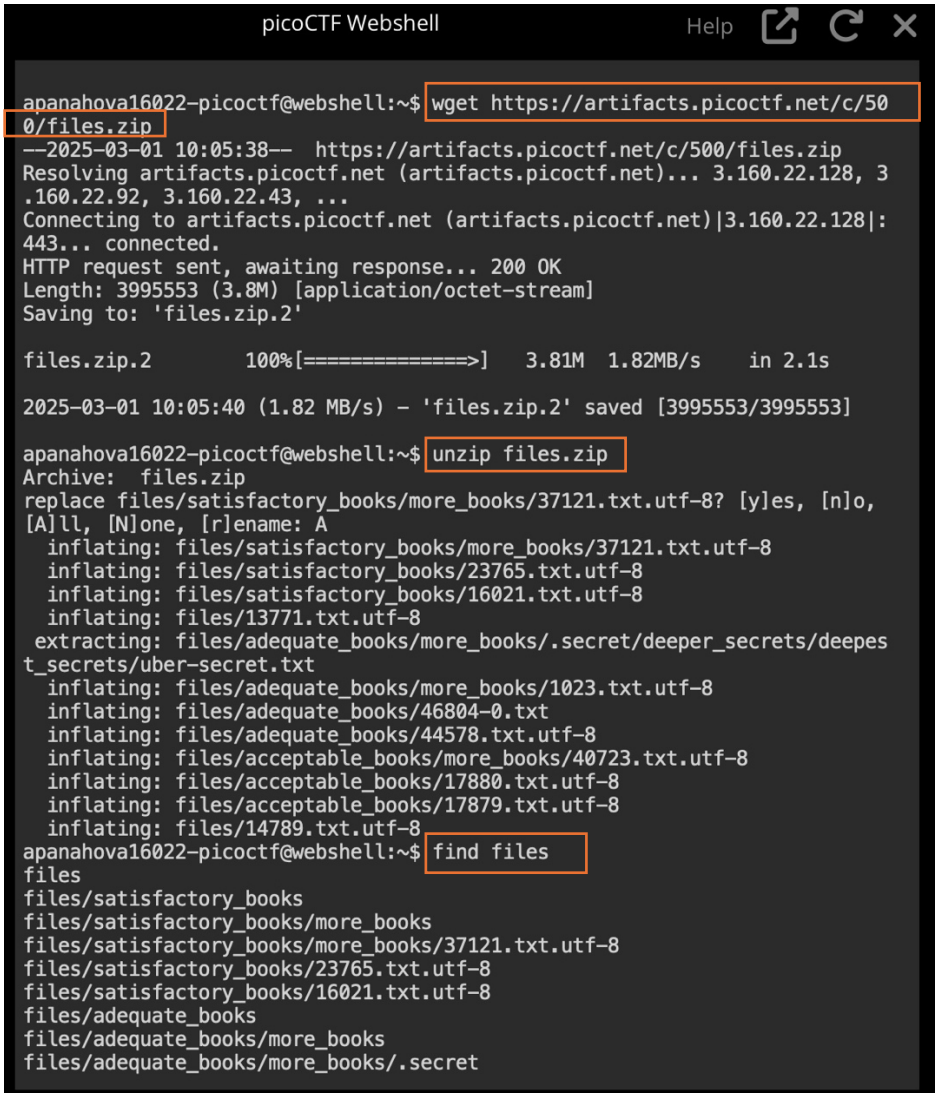
1. Big Zip

In this task, I used my terminal to find the flag, not the picoCTF’s webshell. Firstly, I copied the address by right-clicking on the indicated zipped file “Download zip file.” Then, I navigated to the Desktop (where the file is located) and then into the “big-zip-files” (the name of the zipped file). After these steps, I run the “grep -r” command, which is needed to search for the word “picoCTF” inside all the files within that folder. Inside the file, we can see the flag “picoCTF{gr3p_15_m4g1c_ef8790dc}”.



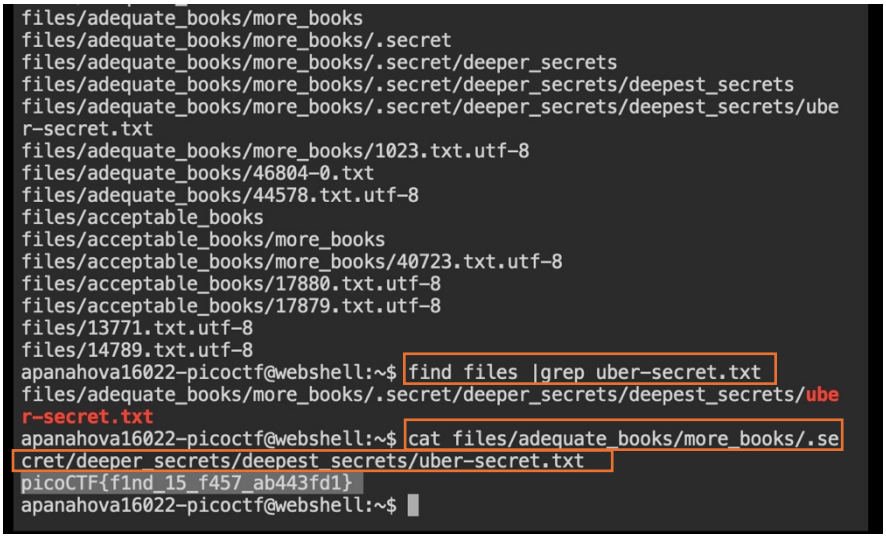
2. First Find

In this problem, we need to unzip the provided file and find the file named “uber-secret.txt.” The first step that we need to take is downloading the zipped file in the webshell. We need to do it using the “wget” command and copy and paste the address of the zipped file. As the task requires unzipping the file, we can use the “unzip [name of the file]” command to access its contents. After unzipping, we check what files and folders are located using the “find” command.



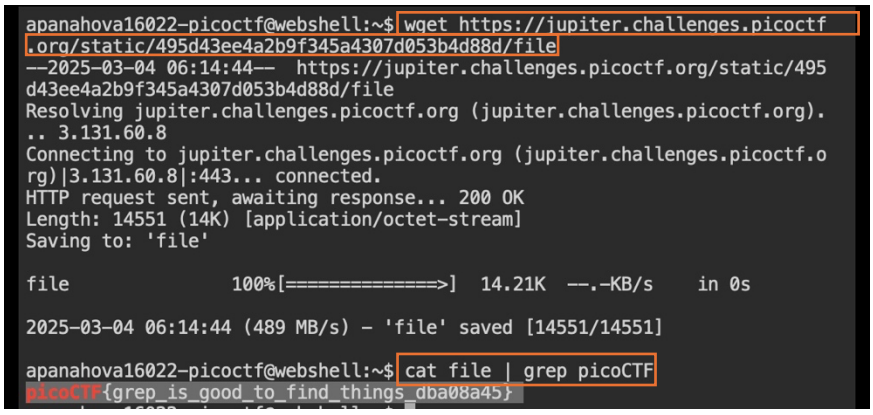
After listing all the files, the task requires finding a file named “uber-secret.txt.” To find this specific file, we need to run “find files | grep uber-secret.txt.” This command works in two parts. First, “find files” searches for all the files and folders in the finds directory and lists them. Since there are many folders, manually searching for a specific file would take too much

time. Therefore, we use “grep uber-secret.txt,” which filters the output of find and shows the specific file name, which is “uber-secret.txt” in our case. After running this command, we can see that the file is hidden inside multiple directories: “files/adequate_books/more_books/.secret/deeper_secrets/uber-secret.txt”. The last step is to read its contents. To do this step, we use the “cat” command. When we run this, we can finally see the hidden flag inside.



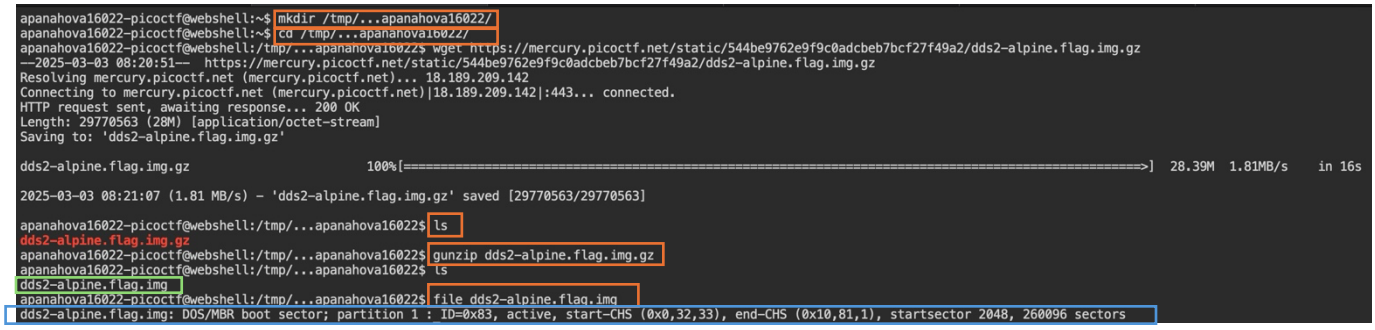
3. First Grep

This problem requires us to find the flag in the provided file. First, we use the “wget” command to download the file from the website to the webshell. Next, we use the “cat” command to display the contents of the file but filter by using the “grep” command to search for the keyword “picoCTF.” This helps to find the flag very quickly without searching manually.

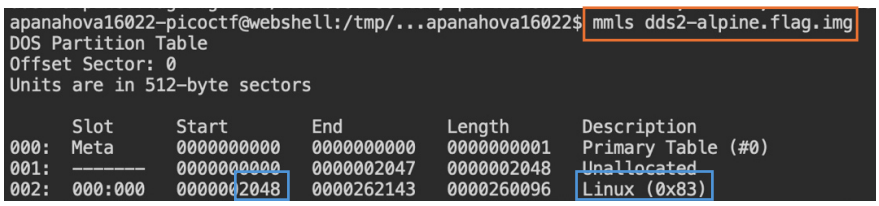


4. Disk, disk, sleuth! II

In this task first, I created an additional directory for this task because our home directory does not have enough disk space. To create a new directory, we write “mkdir /tmp/…[any directory name]/”. “mkdir” stands for “make directory”, which is used to create a new folder. “tmp” is a temporary directory in Linux operating system, where files are stored for short-term use. After creating a directory, we are going to enter that directory with “cd” (change directory) to move into the new directory that we created. This is crucial because it helps us to perform operations. Then, we download the compressed disk image file “dds2-alpine.flag.img.gz” by copying the address and using the “wget” command. Then, we can run the “ls” command to make sure that the file was successfully downloaded and is present in the current directory. After downloading it in the webshell, we need to decompress it using the “gunzip” command, which extracts the disk image “dds2-alpine.flag.img” (highlighted with green). Next, we use the “file” command to check what type of file it is. It proved that the file is a disk image with a DOS/MBR boot sector and contains a Linux partition (highlighted with blue).



Then we run the “mmls” command from the Sleuth Kit, which shows the partition table of the disk image. We can see three sections: the primary partition table, an allocated space, and a Linux partition (the main partition because it contains the actual file). Here we see that the Linux partition starts at sector 2048. This means that if we want to explore the contents of the system, we should start from the sector 2048.



Then, we use the “fls” command to list all the files inside the Linux partition, starting from sector 2048, and the “grep” command to filter out the output. The structure of the command must be like “fls -r -p -o 2048 dds2-alpine.flag.img | grep down-at-the-bottom.txt”. We can break down the command to understand its function better. “-r” tells “fls” to list all files inside directories. “-p” shows the full path of each file. “-o 2048” tells “fls” to start analyzing from sector 2048, where Linux partition starts. “dds2-alpine.flag.img” is basically the disk image. Finally, “grep down-at-the-bottom.txt” filters the output to find the file named “down-at-the-bottom.txt”.

This command shows that the file exists and its inode number (the unique identifier assigned to a file or a directory) is 18291.

Since normal tools (like “cat”) do not work on disk images directly, we use the “icat” command to extract the file using its inode number. The command should look like “icat -o 2028 dds2-alpine.flag.img 18291”. As a result, we can see the flag.

5. Operation Orchid

In this task, we have a similar structure. First, we again create a new directory with “mkdir /tmp/…[any directory name]/”. Then, we are going to enter that directory with “cd.” After this step, decompress the file with the “gunzip,” which extracts the disk image “disk.flag.img”. After this, we check the file type with “file disk.flag.img”, which confirms it is a DOM/MBR disk image. Then, we use the “mmls” command to show the partition layout of a disk image. We can see several sections: the primary partition table, an allocated space, a Linux partition (2048), Linux partition (411648), and a Solaris x86 partition (206848).

```
apanahova16022-picoctf@webshell:~$ mkdir /tmp/...ays/
apanahova16022-picoctf@webshell:~$ cd /tmp/...ays/
apanahova16022-picoctf@webshell:/tmp/...ays$ wget https://artifacts.picoctf.net/c/213/disk.flag.img.gz
--2025-03-03 08:40:48--  https://artifacts.picoctf.net/c/213/disk.flag.img.gz
Resolving artifacts.picoctf.net (artifacts.picoctf.net)... 3.160.22.92, 3.160.22.16, 3.160.22.43, ...
Connecting to artifacts.picoctf.net (artifacts.picoctf.net)|3.160.22.92|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 44360922 (42M) [application/octet-stream]
Saving to: 'disk.flag.img.gz'

disk.flag.img.gz                    100%[===========================================================================================>]  42.31M  1.81MB/s    in 23s

2025-03-03 08:41:11 (1.81 MB/s) - 'disk.flag.img.gz' saved [44360922/44360922]

apanahova16022-picoctf@webshell:/tmp/...ays$ gunzip disk.flag.img.gz
apanahova16022-picoctf@webshell:/tmp/...ays$ ls
disk.flag.img
apanahova16022-picoctf@webshell:/tmp/...ays$ file disk.flag.img
disk.flag.img: DOS/MBR boot sector; partition 1 : ID=0x83, active, start-CHS (0x0,32,33), end-CHS (0xc,223,19), startsector 2048, 204800 sectors; partition 2 : ID=0x82, start-CHS (0xc,223,
20), end-CHS (0x19,159,6), startsector 206848, 204800 sectors; partition 3 : ID=0x83, start-CHS (0x19,159,7), end-CHS (0x32,253,11), startsector 411648, 407552 sectors
apanahova16022-picoctf@webshell:/tmp/...ays$ mmls disk.flag.img
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

      Slot      Start        End          Length       Description
000:  Meta      0000000000   0000000000   0000000001   Primary Table (#0)
001:  -------   0000000000   0000002047   0000002048   Unallocated
002:  000:000   0000002048   0000206847   0000204800   Linux (0x83)
003:  000:001   0000206848   0000411647   0000204800   Linux Swap / Solaris x86 (0x82)
004:  000:002   0000411648   0000819199   0000407552   Linux (0x83)
apanahova16022-picoctf@webshell:/tmp/...ays$
```

Now, we need to list all the files and directories from a file system inside a disk image by using the “fls” command for each offset. Firstly, we investigate the offset 2048 by running the “fls disk.flag.img 2048 -o”. “-o 2048” specifies the offset where the file system starts within the image.

```
apanahova16022-picoctf@webshell:/tmp/...ays$ fls disk.flag.img 2048 -o
fls: option requires an argument -- 'o'
Invalid argument: (null)
usage: fls [-adDFlhpruvV] [-f fstype] [-i imgtype] [-b dev_sector_size] [-m dir/] [-o imgoffset] [-z ZONE] [-s seconds] image [images] [inode]
        If [inode] is not given, the root directory is used
        -a: Display "." and ".." entries
        -d: Display deleted entries only
        -D: Display only directories
        -F: Display only files
        -l: Display long version (like ls -l)
        -i imgtype: Format of image file (use '-i list' for supported types)
        -b dev_sector_size: The size (in bytes) of the device sectors
        -f fstype: File system type (use '-f list' for supported types)
        -m: Display output in mactime input format with
             dir/ as the actual mount point of the image
        -h: Include MD5 checksum hash in mactime output
        -o imgoffset: Offset into image file (in sectors)
        -P pooltype: Pool container type (use '-P list' for supported types)
        -B pool_volume_block: Starting block (for pool volumes only)
        -S snap_id: Snapshot ID (for APFS only)
        -p: Display full path for each file
        -r: Recurse on directory entries
        -u: Display undeleted entries only
        -v: verbose output to stderr
        -V: Print version
        -z: Time zone of original machine (i.e. EST5EDT or GMT) (only useful with -l)
        -s seconds: Time skew of original machine (in seconds) (only useful with -l & -m)
        -k password: Decryption password for encrypted volumes
apanahova16022-picoctf@webshell:/tmp/...ays$
```

We repeat the same command two more times because we have to investigate 206848 (Solaris x86) and 411648 (Linux).

```
apanahova16022-picoctf@webshell:/tmp/...ays$ fls disk.flag.img -o 206847
Cannot determine file system type
apanahova16022-picoctf@webshell:/tmp/...ays$
```

Because the output says “cannot determine file system” we do not consider this part and ignore it.

```
apanahova16022-picoctf@webshell:/tmp/...ays$ fls disk.flag.img -o 411648
d/d 460:        home
d/d 11: lost+found
d/d 12: boot
d/d 13: etc
d/d 81: proc
d/d 82: dev
d/d 83: tmp
d/d 84: lib
d/d 87: var
d/d 96: usr
d/d 106:        bin
d/d 120:        sbin
d/d 466:        media
d/d 470:        mnt
d/d 471:        opt
d/d 472:        root
d/d 473:        run
d/d 475:        srv
d/d 476:        sys
d/d 2041:       swap
V/V 51001:      $OrphanFiles
apanahova16022-picoctf@webshell:/tmp/...ays$
```

Now, here, we can see a lot of different directories. We will look at the ROOT directory because it is needed to find the flag. The root directory’s inode is 472. Therefore, we will use it to list the files, particularly on that inode. The command for this step must be “fls disk.flag.img 472 -o 411648”.

```
apanahova16022-picoctf@webshell:/tmp/...ays$ fls disk.flag.img 472 -o 411648
r/r 1875:       .ash_history
r/r * 1876(realloc):    flag.txt
r/r 1782:       flag.txt.enc
```

In this inode, we can see three files. Then, we use the “icat” command, which is going to let us read specific inodes at different offsets.

```
apanahova16022-picoctf@webshell:/tmp/...ays$ icat -o 411648 disk.flag.img 1876
          -0.881573            34.311733
apanahova16022-picoctf@webshell:/tmp/...ays$ icat -o 411648 disk.flag.img 1782
Salted__ eBJc$QE&$4jMKGeE^Z7 _$'%apanahova16022-picoctf@webshell:/tmp/...ays$ icat -o 411648 disk.flag.img 1875
touch flag.txt
nano flag.txt
apk get nano
apk --help
apk add nano
nano flag.txt
openssl
openssl aes256 -salt -in flag.txt -out flag.txt.enc -k unbreakablepassword1234567
shred -u flag.txt
ls -al
halt
```

Now because we have access to the encoded version of the flag, we can decode it with the opposite operation of "openSSL" line. Before that we need to run "icat -o 411648 disk.flag.img 1782 > flag.txt.enc". This command extracts the file with inode 1782 from the disk.flag.img disk image, considering that the file system starts at sector 411648. The extracted content is then saved into flag.txt.enc.

```
apanahova16022-picoctf@webshell:/tmp/...ays$ icat -o 411648 disk.flag.img 1782 > flag.txt.enc
apanahova16022-picoctf@webshell:/tmp/...ays$ ls
disk.flag.img  flag.txt.enc
```

Now, we run the opposite operation with "openssl" in order to decode the flag.

```
apanahova16022-picoctf@webshell:/tmp/...ays$ openssl aes256 -d -salt -in flag.txt.enc -out flag.txt -k unbreakablepassword1234567
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
bad decrypt
80CB6E54017F0000:error:1C800064:Provider routines:ossl_cipher_unpadblock:bad decrypt:../providers/implementations/ciphers/ciphercommon_block.c:124:
apanahova16022-picoctf@webshell:/tmp/...ays$ ls
disk.flag.img  flag.txt  flag.txt.enc
apanahova16022-picoctf@webshell:/tmp/...ays$ file flag.txt
flag.txt: data
apanahova16022-picoctf@webshell:/tmp/...ays$ cat flag.txt
picoCTF{h4un71ng_p457_5113beab}apanahova16022-picoctf@webshell:/tmp/...ays$
```

As we can see, it may display a warning message saying "bad decrypt", but this does not cause any issues. We can verify this by running the ls command to list the files, and we can see that it successfully contains a flag.txt file. When we read the file by running the "cat" command, it will show the flag itself.

6. Sleuthkit Apprentice

In this task, we complete all the steps that are written up to the point of determining the file system layout using the ”mmls [file name]” command. When we run the “mmls disk.flag.img” command we can see that there are three potential partitions to look at.

```
apanahova16022-picoctf@webshell:~$ mkdir /tmp/...apanahova/
apanahova16022-picoctf@webshell:~$ cd /tmp/...apanahova/
apanahova16022-picoctf@webshell:/tmp/...apanahova$ wget https://artifacts.picoctf.net/c/138/disk.flag.img.gz
--2025-03-03 09:05:40--  https://artifacts.picoctf.net/c/138/disk.flag.img.gz
Resolving artifacts.picoctf.net (artifacts.picoctf.net)... 3.160.22.43, 3.160.22.92, 3.160.22.16, ...
Connecting to artifacts.picoctf.net (artifacts.picoctf.net)|3.160.22.43|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 47534528 (45M) [application/octet-stream]
Saving to: 'disk.flag.img.gz'

disk.flag.img.gz                              100%[=========================================================

2025-03-03 09:06:05 (1.80 MB/s) - 'disk.flag.img.gz' saved [47534528/47534528]

apanahova16022-picoctf@webshell:/tmp/...apanahova$ gunzip disk.flag.img.gz
apanahova16022-picoctf@webshell:/tmp/...apanahova$ ls
disk.flag.img
apanahova16022-picoctf@webshell:/tmp/...apanahova$ mmls disk.flag.img
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

     Slot      Start        End          Length       Description
000:  Meta      0000000000   0000000000   0000000001   Primary Table (#0)
001:  -------   0000000000   0000002047   0000002048   Unallocated
002:  000:000   0000002048   0000206847   0000204800   Linux (0x83)
003:  000:001   0000206848   0000360447   0000153600   Linux Swap / Solaris x86 (0x82)
004:  000:002   0000360448   0000614399   0000253952   Linux (0x83)
```

Now we need to use the “fsstat” command, which is going to help us to get information about these partitions. The command structure should be like “fsstat -o [offset] [name of the file]”. In our case first, we should investigate offset 2048 (fsstat -o 2048 disk.flag.img). The output should give a lot of information but we are particularly interested in the line that specifies the "File System Type: Ext4".

```
--------------------------------------------
File System Type: Ext4
Volume Name:
Volume ID: 3c054136f02898b3224bd632cbd6c255
```

We are going to use this in our enumeration commands. Now, we enumerate the other partitions.

```
apanahova16022-picoctf@webshell:/tmp/...apanahova$ fsstat -o 206848 disk.flag.img
Cannot determine file system type
apanahova16022-picoctf@webshell:/tmp/...apanahova$
```

As we can see, a message appears stating “cannot determine file system.” This indicates that we will not be using this partition.

However, when we execute the command “fsstat -o 360448 disk.flag.img”, we observe that the output contains valid information about the partition. We can determine that the partition is formatted using the 'Ext4' file system by carefully looking at the details. This confirms that the partition is usable, and we will proceed to use this information for further enumeration and analysis of the partition.

In the next step, we need to list the all files and directories within the “Ext4” file system of the disk image “disk.flag.img”, starting from offset 2048 by using the “fls -f ext4 -o 2048 -r disk.flag.img” command.

```
apanahova16022-picoctf@webshell:/tmp/...apanahova$ fls -f ext4 -o 2048 -r disk.flag.img
d/d 11: lost+found
r/r 12: ldlinux.sys
r/r 13: ldlinux.c32
r/r 15: config-virt
r/r 16: vmlinuz-virt
r/r 17: initramfs-virt
l/l 18: boot
r/r 20: libutil.c32
r/r 19: extlinux.conf
r/r 21: libcom32.c32
r/r 22: mboot.c32
r/r 23: menu.c32
r/r 14: System.map-virt
r/r 24: vesamenu.c32
V/V 25585:      $OrphanFiles
```

Since there are not many files in this partition, we will examine the other partition starting at offset 360448.

```
apanahova16022-picoctf@webshell:/tmp/...apanahova$ fls -f ext4 -o 360448 -r disk.flag.img
```

In the output, there will be a lot of files. However, if we look further up, we can see three useful files located under the ROOT directory. Why root directory? It is because ROOT is a top-level directory in the file system.

```
d/d 1992:       media
+ d/d 457:      cdrom
+ d/d 458:      floppy
+ d/d 459:      usb
d/d 1993:       mnt
d/d 1994:       opt
d/d 1995:       root
+ r/r 2363:     .ash_history
+ d/d 3981:     my_folder
++ r/r * 2082(realloc): flag.txt
++ r/r 2371:    flag.uni.txt
d/d 1996:       run
d/d 1997:       srv
d/d 1998:       sys
d/d 2358:       swap
V/V 31745:      $OrphanFiles
+ -/r * 2374:   OrphanFile-2374
+ -/l * 2375:   OrphanFile-2375
+ -/r * 2376:   OrphanFile-2376
+ -/l * 2377:   OrphanFile-2377
+ -/r * 2378:   OrphanFile-2378
```

After this step, we must examine the inode numbers in order to find the flag. Firstly, we can examine “flag.txt’ (inode number - 2082) by using “icat -f ext4 -o 360448 disk.flag.img 2082”

```
apanahova16022-picoctf@webshell:/tmp/...apanahova$ icat -f ext4 -o 360448 disk.flag.img 2082
            3.449677            13.056403
```

There is no flag here. Therefore, we will examine “flag.uni.txt” (inode number - 2371).

```
apanahova16022-picoctf@webshell:/tmp/...apanahova$ icat -f ext4 -o 360448 disk.flag.img 2371
picoCTF{by73_5urf3r_2f22df38}
```

As a result, we found the flag.

7. Sleuthkit Intro

In this task we again create a new repository, then enter that directory with “cd” to move into the new directory that we created. After that download the file with “wget’ command and unzip the file by using “gunzip [unzipped file name]” command. Then run “file [file name] to determine the file type. In our case, it is “disk.img”.

```
apanahova16022-picoctf@webshell:~$ mkdir /tmp/...apanahova/
apanahova16022-picoctf@webshell:~$ cd /tmp/...apanahova/
apanahova16022-picoctf@webshell:/tmp/...apanahova$ wget https://artifacts.picoctf.net/c/164/disk.img.gz
--2025-03-03 17:15:08--  https://artifacts.picoctf.net/c/164/disk.img.gz
Resolving artifacts.picoctf.net (artifacts.picoctf.net)... 3.160.22.43, 3.160.22.92, 3.160.22.16, ...
Connecting to artifacts.picoctf.net (artifacts.picoctf.net)|3.160.22.43|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 29714372 (28M) [application/octet-stream]
Saving to: 'disk.img.gz'

disk.img.gz                                   100%[=============================================================================================

2025-03-03 17:15:24 (1.81 MB/s) - 'disk.img.gz' saved [29714372/29714372]

apanahova16022-picoctf@webshell:/tmp/...apanahova$ gunzip disk.img.gz
apanahova16022-picoctf@webshell:/tmp/...apanahova$ ls
disk.img
apanahova16022-picoctf@webshell:/tmp/...apanahova$ file disk.img
disk.img: DOS/MBR boot sector; partition 1 : ID=0x83, active, start-CHS (0x0,32,33), end-CHS (0xc,190,50), startsector 2048, 202752 sectors
apanahova16022-picoctf@webshell:/tmp/...apanahova$ mmls disk.img
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

      Slot      Start        End          Length       Description
000:  Meta      0000000000   0000000000   0000000001   Primary Table (#0)
001:  -------   0000000000   0000002047   0000002048   Unallocated
002:  000:000   0000002048   0000204799   0000202752   Linux (0x83)
apanahova16022-picoctf@webshell:/tmp/...apanahova$
```

As we see from the picture, we have one usable disk partition, which is Linux (0x83). As the task requires the size of the partition, we must look at the length. Here, we see that the length of the Linux partition is ‘202752”.

After clicking the 'Launch Instance' button in the picoCTF task, we can see the access checker program to connect to the saturn.picoctf.net server on port 57134 using the “nc” (Netcat) command. Netcat is a tool used for tasks such as network connections, port scanning, and data transfer between computers. We simply need to copy the command (nc saturn.picoctf.net 57134) and paste it into the webshell. Once connected, it will ask for the length in sectors. We just need to enter the length of the Linux partition, and there, we will find the flag.

```
apanahova16022-picoctf@webshell:/tmp/...apanahova$ nc saturn.picoctf.net 55489
What is the size of the Linux partition in the given disk image?
Length in sectors: 202752
202752
Great work!
picoCTF{mm15_f7w!}
```