Challenge name: Manini-Case

Challenge description:

> An employee's machine has been brought to your notice for analysis.

> The employee is suspected of engaging in equipment abuse by contravening the company's policies.

> You are requested to carry out an inquiry into the staff device to identify any evidence of improper usage.

> Connect to the server below and answer all the questions correctly to get the flag at the end.

Author: @Houssem0x1

Writeup:

The Challenge is about analyzing a disk image and answering the questions correctly to get the flag, after downloading and decompressing the Attached file, we can see 03 files: SusManini.E01 SusManini.E02 SusManini.E03. After quick research on google about the file's extension we find that is EWF "Expert witness format" which is very common in forensics Disk images acquisition tools, and in our case the image is splitted into 03 parts which may be confusing. Now, we need somehow to view the content of this disk image, various methods can be followed, we can use popular GUI Disk analysis utilities like Autopsy, FTK imager … but in this writeup I will show you a more efficient and professional way using Linux CLI and the sleuth-kit utility.

To not waste time lets see the 1st question and start with it:

1. What is the sha1 hash of the disk image:
   Most of people may go directly and run the Sha1sum command against the image which is wrong due to EWF files structure. In linux, we have the sleuth kit utility which comes pre-installed in most cybersecurity distros like KALI, and we need to install ewf-tools to work properly with the EWF files.

```
└─$ sudo apt install sleuthkit
[sudo] password for kali:
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following packages were automatically installed and are no longer required:
  aardvark-dns buildah conmon fuse-overlayfs golang-github-containers-common golang-github-containers-image libslirp0 libsubid4 netavark passt
  podman slirp4netns uidmap
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libtsk19
Suggested packages:
  mac-robber
The following packages will be upgraded:
  libtsk19 sleuthkit
2 upgraded, 0 newly installed, 0 to remove and 324 not upgraded.
Need to get 763 kB of archives.
After this operation, 3,072 B of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

```
└─$ sudo apt install ewf-tools
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
ewf-tools is already the newest version (20140814-1).
The following packages were automatically installed and are no longer required:
  aardvark-dns buildah conmon fuse-overlayfs golang-github-containers-common golang-github-containers-image libslirp0 libsubid4 netavark passt
  podman slirp4netns uidmap
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 326 not upgraded.
```

Ok, to answer Q1, there is a command part of ewf-tools called ewfinfo or we can use enwverify to get the coorect hash which actually stored in the footer of the image.

```
└─$ sudo ewfinfo SusMan1n1.E01
ewfinfo 20140814

Acquiry information
	Case number:			001
	Description:			Nexus CTF
	Examiner name:			@Houssem0×1
	Evidence number:		001
	Notes:				Nexus club rights.
	Acquisition date:		Tue Jan 16 22:32:18 2024
	System date:			Tue Jan 16 22:32:18 2024
	Operating system used:		5.15.0-91-generic
	Software version used:		guymager 0.8.13-1
	Password:			N/A
	Model:				VMware_Virtual_S

EWF information
	File format:			EnCase 6
	Sectors per chunk:		64
	Error granularity:		1
	Compression method:		deflate
	Compression level:		good (fast) compression

Media information
	Media type:			fixed disk
	Is physical:			yes
	Bytes per sector:		512
	Number of sectors:		62914560
	Media size:			30 GiB (32212254720 bytes)

Digest hash information
	MD5:				8c3143f4c8909dba0a2b371afe89f65e
	SHA1:				312fcd401047b7767a7410ce2464ad5a8af346b2
```

As you can see you can see I run the command against part 1 of the image only which sufficient because the result will be the same with E02 E03 files. Notice the MD5 and SHA1 hashes info at the bottom which give us the answer to Q1

Another thing to observe is The Operating System used line says it its 5.15.0-91-generic which give us a hint that we are dealing with a linux capture, this is very useful info in our further analysis.
Q1 answer: 312fcd401047b7767a7410ce2464ad5a8af346b2

2. What is the user's computer name ?
   Now, we need to start our investigation, this info can be obtained quite easily using gui tools like Autopsy, but we're using linux and the CLI, so mounting the disk image would be the best idea, the thing that the image is splitted into 03 parts is confusing but in fact it is easy using the proper tools I will show you. Let's go to our files directory and create two folders in the same directory, name them as you want for instance phy1 log1, then we need to mount the image to one of the folders we already created for ex phy1, we can use ewfmount command which is part of ewf-tools, so:
   "sudo ewfmount SusManini.E01 ./phy1"

```
└─$ sudo ewfmount SusManini.E01 ./phy1
ewfmount 20140814
```

Notice that I gave the command Only part one of the image SusManini.E01 because it is smart enough to detect the other parts, re-assemble them, decompress them to get the actual physical disk and mount it to phy1 directory. This will create a file named "ewf1" under the phy1 dir. To confirm that is actually a disk, let's run file command against it:

```
└─$ sudo file ./phy1/ewf1
./phy1/ewf1: DOS/MBR boot sector, extended partition table (last)
```

perfect. Now we need to access it, lets use "mmls" which is part of the sleuth kit to get info about the disk partitions:

```
└─$ sudo mmls ./phy1/ewf1
GUID Partition Table (EFI)
Offset Sector: 0
Units are in 512-byte sectors

     Slot      Start        End          Length       Description
000:  Meta     0000000000   0000000000   0000000001   Safety Table
001:  ------   0000000000   0000002047   0000002048   Unallocated
002:  Meta     0000000001   0000000001   0000000001   GPT Header
003:  Meta     0000000002   0000000033   0000000032   Partition Table
004:  000      0000002048   0000004095   0000002048
005:  001      0000004096   0001054719   0001050624   EFI System Partition
006:  002      0001054720   0062912511   0061857792
007:  ------   0062912512   0062914559   0000002048   Unallocated
```

Ok, notice that the mmls output says that Units are in 512-byte sectors, remember this. So basically, here we are looking for the largest partition which is the one that has the biggest length (probably it is the one that contains user's data), here we need the start offset of this partition which is **1054720** and remember that Units are in 512-byte sectors we can do a quick math: 1054720 * 512

```
└─$ echo 1054720 *\ 512 | bc
540016640
```

and we get 540016640, bc is just a calculator.

Now, we can use the built-in linux command 'mount' to mount our physical disk located in phy1 to the other folder we created which is log1 in my case, the command would be:

"sudo mount -o ro,norecovery,loop,offset=540016640 ./phy1/ewf1 ./log1/"

You should check the command manual for the options used.

```
└─$ sudo mount -o ro,norecovery,loop,offset=540016640 ./phy1/ewf1 ./log1/
```

Perfect, now if we go to our log1 folder and see what is there!

```
└─$ ls log1
bin   cdrom  etc   lib    lost+found  mnt  proc  run   srv   sys  usr
boot  dev    home  lib64  media       opt  root  sbin  swapfile  tmp  var
```

And as you can see, we get the system files of a Linux machine. We can navigate between them and read their content easily. But in this and with the magic of linux, I will use "chroot" command to get a native view of the system files as I am in the actual machine that the image was taken from, watch this. Make sure you are a step above the mounting dir which is log1 in my case and run: "sudo chroot ./log1 /usr/binbash", so here we are changing our root directory to the mount point and interacting with it using bash.

```
└─$ sudo chroot ./log1 /usr/bin/bash
root@kali:/#
```

and as you can see, we are root, wait wait this is forensics not pwn xD!

Ok, now we are done with the mounting lets go back to our question, to get the answer simply we type: "cat /etc/hostname"

```
root@kali:/# cat /etc/hostname
manini-desktop
root@kali:/#
```

3. What is the login password of this user ?
   Here, clearly, we need to do some cracking. And for performance purposes we are given a password list file attached with challenge files. We all know that the password hash is stored in /etc/shadow file, lets take a look:

```
manini:$y$j9T$WAaXEsxn2XMOxQhYFAJ9t0$7PKOLnhZVjKDbkEZ3WvkvbTK2HLMGWDo3EmJG6JuOr8:19739:0:99
99:7:::
```

   after a quick google search we find that is a Yescrypt hash and not a casual Sha512, ok after another google search we find out that JohnTheriper can do the mission and since we have the password list it should be easy, we need to give john all a copy of the /etc/shadow file (the entire file and not only the hash), the command would be (of course in your host machine):

   "john –format=crypt CopyOf/etc/shadow –wordlist=ThePathToTheGivenPasswordList"

```
└─$ sudo john yescrypt_hash --show
manini:2024:19739:0:99999:7:::

1 password hash cracked, 0 left
```

   I already did that and the password is: 2024

4. What is the ID of the last boot ?
   This info can be obtained using the following:

```
root@kali:/# journalctl --list-boots
-4 1e29ad9887ca4e5286bbd83ac974fd72 Tue 2024-01-16 19:38:21 CST—Tue 20
-3 5315f18b30db492f8593e7fc31beec9a Tue 2024-01-16 19:42:42 CST—Tue 20
-2 6a0a5d92e64d4874b68341ed0009bca1 Tue 2024-01-16 19:56:31 CST—Tue 20
-1 811095976caf4dd8a2ee7edf2b65912f Tue 2024-01-16 21:14:18 CST—Tue 20
 0 62c2a4fe86254123a4a73629ea60fee0 Tue 2024-01-16 21:27:28 CST—Tue 20
```

   The second column is the ID and based on the time the last one is the answer: 62…
   This is a privilege for us, because if we were using a gui software in windows like autopsy we would have to extract this journal file and and parse with a linux system.

5. How did he install google-chrome (full command) ?
   By checking the bash history we can find the answer:

```
sudo apt update
sudo apt upgrade
sudo apt --fix-broken install
clear
sudo dpkg -i google-chrome-stable_current_amd64.deb
clear
```

6. When exactly he did install google-chrome (YYYY-MM-DD_HH:MM:SS) ?
   Since he used dpkg to install chrome, we need to investigate dpkg log file for this info.
   Most of linux log files are stored in /var/log/ directory, so:

```
root@kali:/# cat /var/log/dpkg.log | grep -i chrome
2024-01-16 21:01:36 install google-chrome-stable:amd64 <none> 120.0.609
9.224-1
2024-01-16 21:01:36 status half-installed google-chrome-stable:amd64 12
0.0.6099.224-1
2024-01-16 21:01:48 status unpacked google-chrome-stable:amd64 120.0.60
99.224-1
2024-01-16 21:05:10 upgrade google-chrome-stable:amd64 120.0.6099.224-1
 120.0.6099.224-1
```

   and voila! Of course I used grep -i to filter out. Format: 2024-01-16_21:01:36

7. Which interesting website he usually visits ?
   Back to the bash history we observe that he edited the /etc/hosts file which is interesting, let's take a look:

```
clear
nano /etc/hosts
sudo nano /etc/hosts
```

```
root@kali:/# cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       manini-desktop
10.10.10.1      freepalestine-foundation.dz
# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

8. What is the UUID of the main root volume ?
   Such info, if we were in a live system we can run:

   “sudo lsblk -o UUID,PARTUUID,NAME,MOUNTPOINT”

   But since we have only an image of the disk and we are dealing with the data partition we can retrieve this info from various log files such as: dmesg, kernel.log …

```
root@kali:/# cat /var/log/dmesg | grep -i uuid
[    0.000000] kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-5.15.0-91
-generic root=UUID=a3389150-6669-478a-8bfb-059dd1718175 ro quiet splash
[    0.218829] kernel: Kernel command line: BOOT_IMAGE=/boot/vmlinuz-5.
15.0-91-generic root=UUID=a3389150-6669-478a-8bfb-059dd1718175 ro quiet
 splash
```

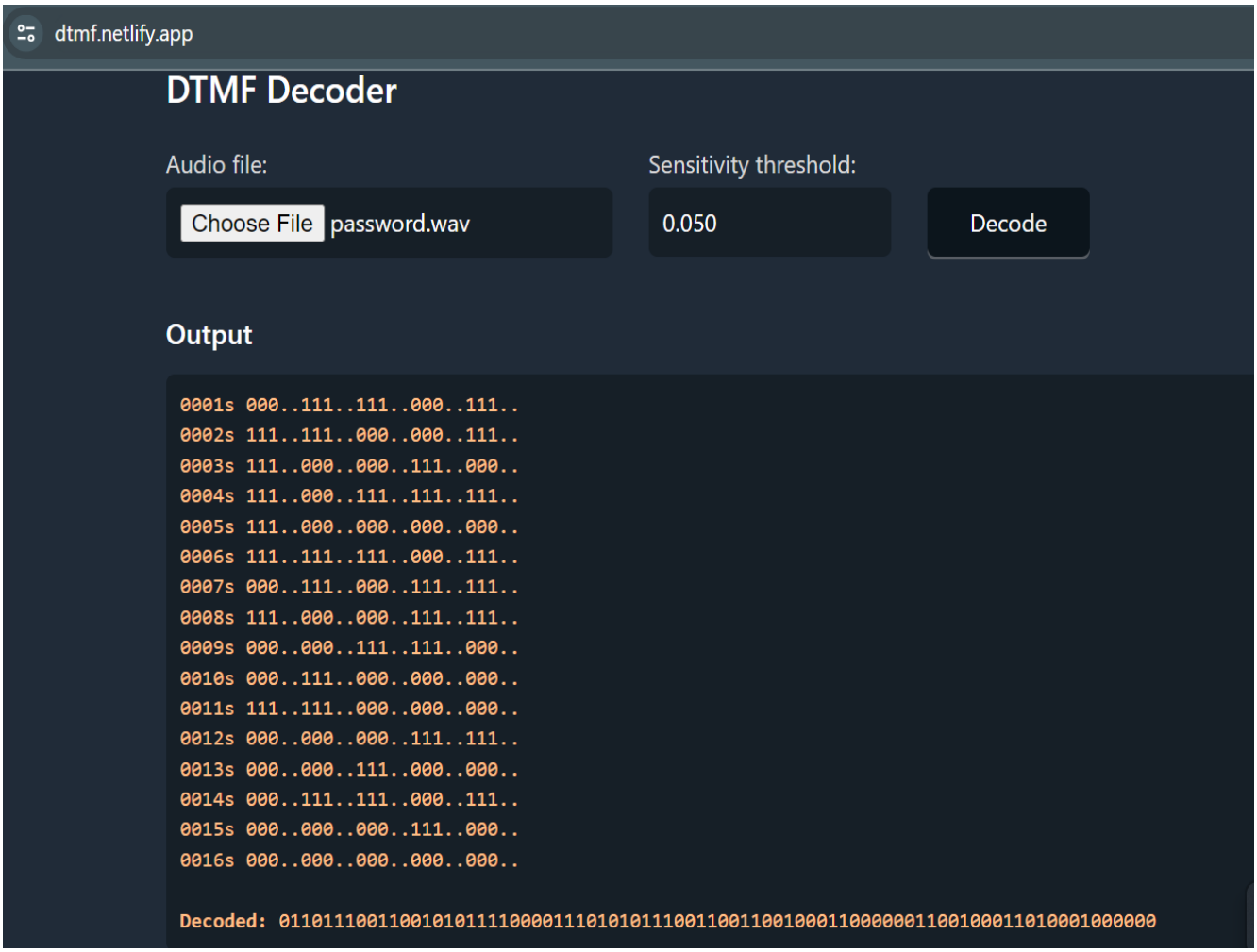
9. The user is suspected that he was hiding a secret in a non-casual place, can you recover his ultimate secret ?
   Back to bash history we see that he created a secret folder under the documents folder, navigating to the document folder we cannot find anything, maybe he moved it or delete it using GUI, but the question says that he is hiding it in a non-casual place hmm, maybe he just moved to recycle bin and se know that the recycle bin does not delete files permanently, after a quick google search, we know that recycle bin content is stored in the user’s home directory in a hidden place, exactly: ‘.local/share/Trash/files/’

```
root@kali:/home/manini# ls .local/share/
applications/       gvfs-metadata/      nano/
evolution/          icc/                recently-used.xbel
flatpak/            keyrings/           Trash/
root@kali:/home/manini# ls .local/share/Trash/
expunged/ files/    info/
root@kali:/home/manini# ls .local/share/Trash/
expunged/ files/    info/
root@kali:/home/manini# ls .local/share/Trash/files/
secret
```

   And yes. Let’s see what is inside.

```
root@kali:/home/manini/.local/share/Trash/files# ls
secret
root@kali:/home/manini/.local/share/Trash/files# cd secret/
root@kali:/home/manini/.local/share/Trash/files/secret# ls
password.wav  uncrackable-secret.7z
root@kali:/home/manini/.local/share/Trash/files/secret#
```

Hmm, a 7zip password protected file an a password.wav file, so clearly that we need to use the password.wav file somehow to get the 7zip archive password. After playing the password.wav it sounds familiar as it looks like someone trying to call a phone number but it is a sequence of two numbers hmmm (binary for sure!), after some googling we understand that is DTMF (google it) and we can use an online DTMF decoder to decode it:



The output looks like binary, let's convert it to ASCII and yes, the pass is: nexus2024@

Let's use it to decompress that archive and finally we have a text file:

```
This is my ultimate secret, i hope no one can find it as I am a cyber geek Pro Max: FreePalestine
```

And yes, the secret is: FreePalestine

Thanks.