

Target Machine NOOB 1 VM

Machine Information

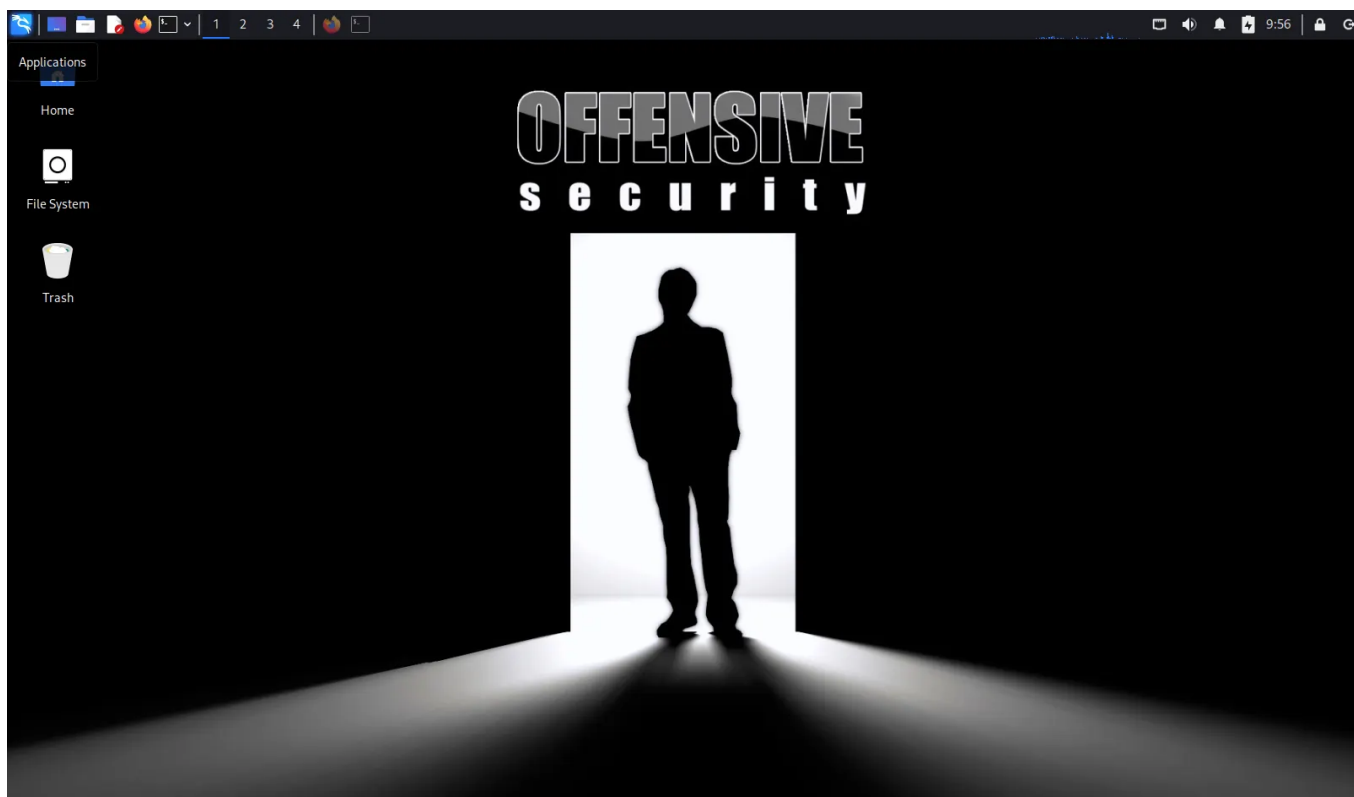
Author: Shantanu Ramdas Kakade

- Platform: VulnHub
- Machine Name: **NOOB:1**
- Difficulty: Beginner
- Goal: Gain **root access** and capture the flag.

Lab Setup

Attacker Machine

- Kali Linux



Target Machine

- NOOB:1 VM



[VIRTUAL MACHINES](#)
[HELP](#)
[RESOURCES](#)
[ABOUT](#)

[SUBMIT MACHINE](#)
[CONTACT US](#)

[Back](#)
[About Release](#)
[Download](#)
[Description](#)
[File Information](#)
[Virtual Machine](#)
[Networking](#)
[Screenshot\(s\)](#)

NOOB: 1

[About Release](#)
[Back to the Top](#)

Name: Noob: 1

Date release: 22 Sep 2021

Author: [VIEH Group](#)

Series: Noob

[Download](#)
[Back to the Top](#)

Please remember that VulnHub is a free community resource so we are unable to check the machines that are provided to us. Before you download, please read our FAQs sections dealing with the dangers of running unknown VMs and our suggestions for "protecting yourself and your network. If you understand the risks, please download!

Noob.ova (Size: 3.1 GB)

Download (Mirror): <https://download.vulnhub.com/noob/Noob.ova>

[Description](#)
[Back to the Top](#)

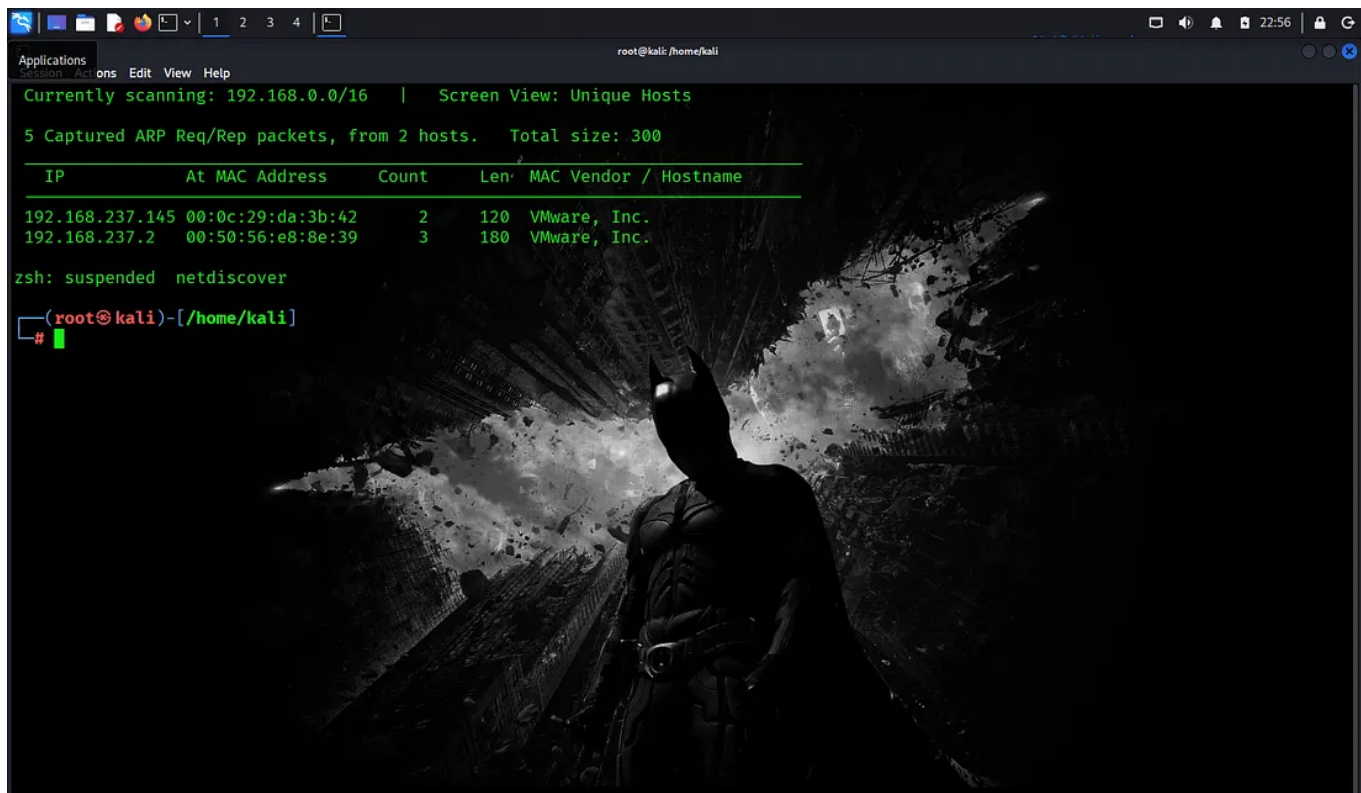
N/A

Both machines connected on the **same network (NAT or Host-Only)**.

Network Discovery

First, identify the target IP address

```
netdiscover
```



```

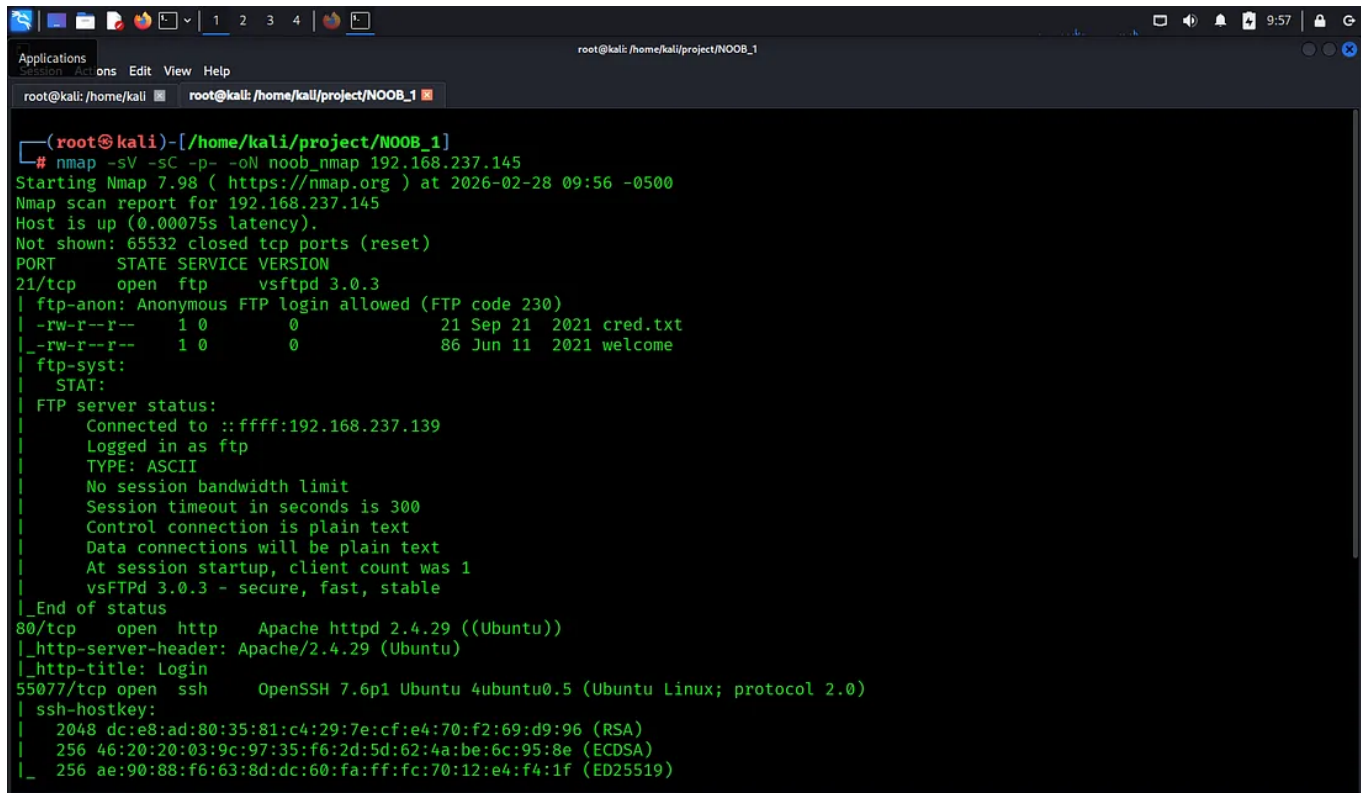
root@kali: ~/home/kali
Applications  Session Actions Edit View Help
Currently scanning: 192.168.0.0/16 | Screen View: Unique Hosts
5 Captured ARP Req/Rep packets, from 2 hosts. Total size: 300
+-----+-----+-----+-----+-----+-----+
| IP           | At MAC Address | Count | Len | MAC Vendor / Hostname |
+-----+-----+-----+-----+-----+-----+
| 192.168.237.145 | 00:0c:29:da:3b:42 | 2     | 120 | VMware, Inc.          |
| 192.168.237.2   | 00:50:56:e8:8e:39 | 3     | 180 | VMware, Inc.          |
+-----+-----+-----+-----+-----+-----+
zsh: suspended netdiscover
root@kali: ~/home/kali
#

```

Port Scanning and Service Enumeration

After identifying the target IP (**192.168.237.145**), I performed a detailed scan using **Nmap** to discover open ports and running services.

```
nmap -sV -sC -p- -oN noob_nmap 192.168.237.145
```



```
(root@kali)-[/home/kali/project/NOOB_1]
# nmap -sV -sC -p- -oN noob_nmap 192.168.237.145
Starting Nmap 7.98 ( https://nmap.org ) at 2026-02-28 09:56 -0500
Nmap scan report for 192.168.237.145
Host is up (0.00075s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_-rw-r--r--  1 0      0      21 Sep 21  2021 cred.txt
|_-rw-r--r--  1 0      0      86 Jun 11  2021 welcome
| ftp-syst:
|   STAT:
|   FTP server status:
|     Connected to ::ffff:192.168.237.139
|     Logged in as ftp
|     TYPE: ASCII
|     No session bandwidth limit
|     Session timeout in seconds is 300
|     Control connection is plain text
|     Data connections will be plain text
|     At session startup, client count was 1
|     vsFTPD 3.0.3 - secure, fast, stable
|_End of status
80/tcp    open  http     Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Login
55077/tcp open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 dc:e8:ad:80:35:81:c4:29:7e:cf:e4:70:f2:69:d9:96 (RSA)
|   256 46:20:20:03:9c:97:35:f6:2d:5d:62:4a:be:6c:95:8e (ECDSA)
|_  256 ae:90:88:f6:63:8d:dc:60:fa:ff:fc:70:12:e4:f4:1f (ED25519)
```

FTP Enumeration

After discovering that **anonymous FTP login was allowed** during the Nmap scan, I attempted to connect to the FTP service. The FTP service was running **vsftpd 3.0.3** on port **21**.

Connecting to FTP

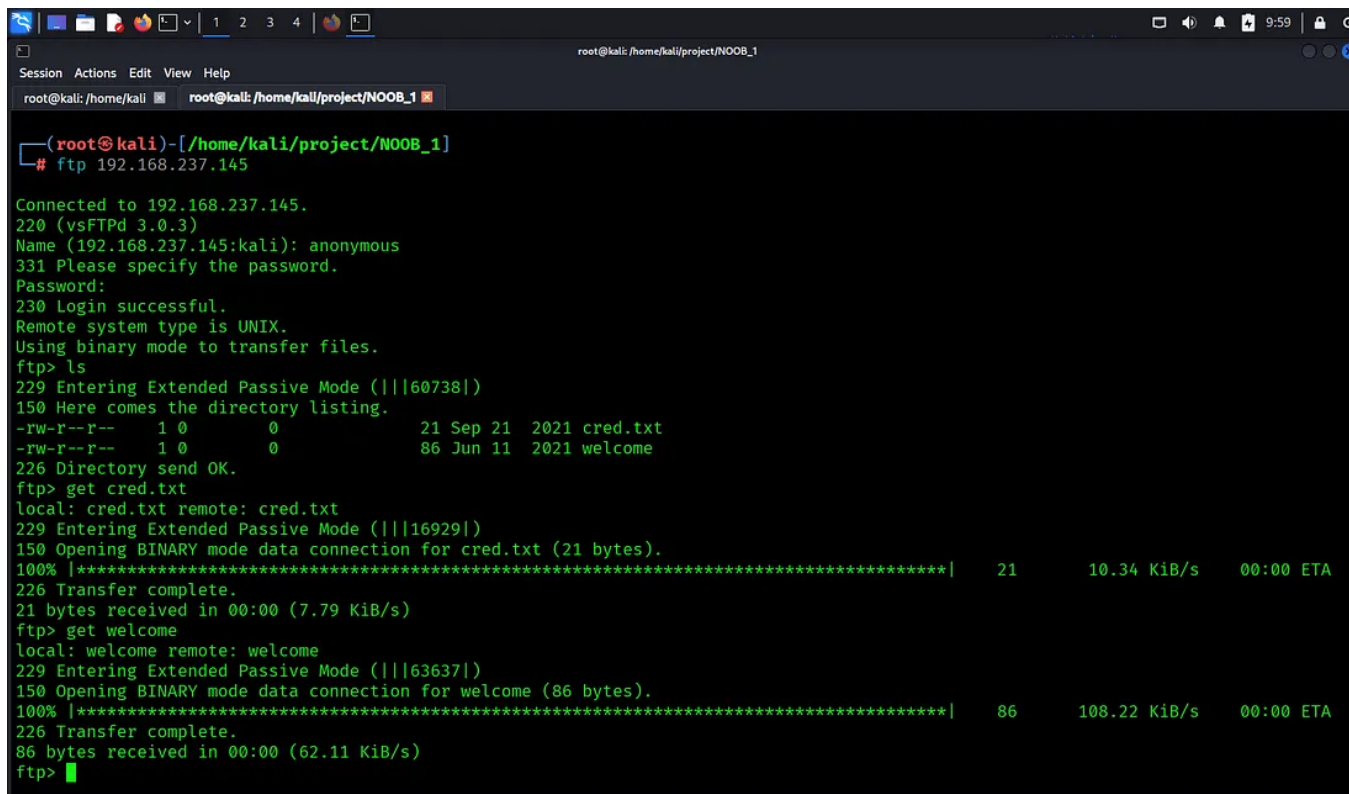
I used the following command to connect to the FTP server:

```
ftp 192.168.237.145
```

I entered **anonymous** as the username and **anonymous** as the password and successfully logged into the FTP server.

```
Username: anonymous
Password: anonymous
```

The login was successful, confirming that the FTP server allowed **unauthenticated access**. After logging into the FTP server, I used the `ls` command to list the available files. I found two files: **cred.txt** and **welcome**. I then used the `get` command to download these files to my local machine.



```
(root@kali)-[/home/kali/project/NOOB_1]
# ftp 192.168.237.145

Connected to 192.168.237.145.
220 (vsFTPD 3.0.3)
Name (192.168.237.145:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||60738|)
150 Here comes the directory listing.
-rw-r--r--  1 0      0          21 Sep 21  2021 cred.txt
-rw-r--r--  1 0      0          86 Jun 11  2021 welcome
226 Directory send OK.
ftp> get cred.txt
local: cred.txt remote: cred.txt
229 Entering Extended Passive Mode (|||16929|)
150 Opening BINARY mode data connection for cred.txt (21 bytes).
100% |*****| 21      10.34 KiB/s   00:00 ETA
226 Transfer complete.
21 bytes received in 00:00 (7.79 KiB/s)
ftp> get welcome
local: welcome remote: welcome
229 Entering Extended Passive Mode (|||63637|)
150 Opening BINARY mode data connection for welcome (86 bytes).
100% |*****| 86      108.22 KiB/s   00:00 ETA
226 Transfer complete.
86 bytes received in 00:00 (62.11 KiB/s)
ftp>
```

Analyzing cred.txt

After downloading the files from the FTP server, I examined the **cred.txt** file to check if it contained any useful information.

To read the file, I used the following command:

```
cat cred.txt
```

The file contained the following encoded string:

```
Y2hhbXA6cGFzc3dvcmQ=
```

The format suggested that the string was **Base64 encoded**, so I decoded it using the following command:

```
echo Y2hhbXA6cGFzc3dvcmQ= | base64 -d
```

The decoded result was:

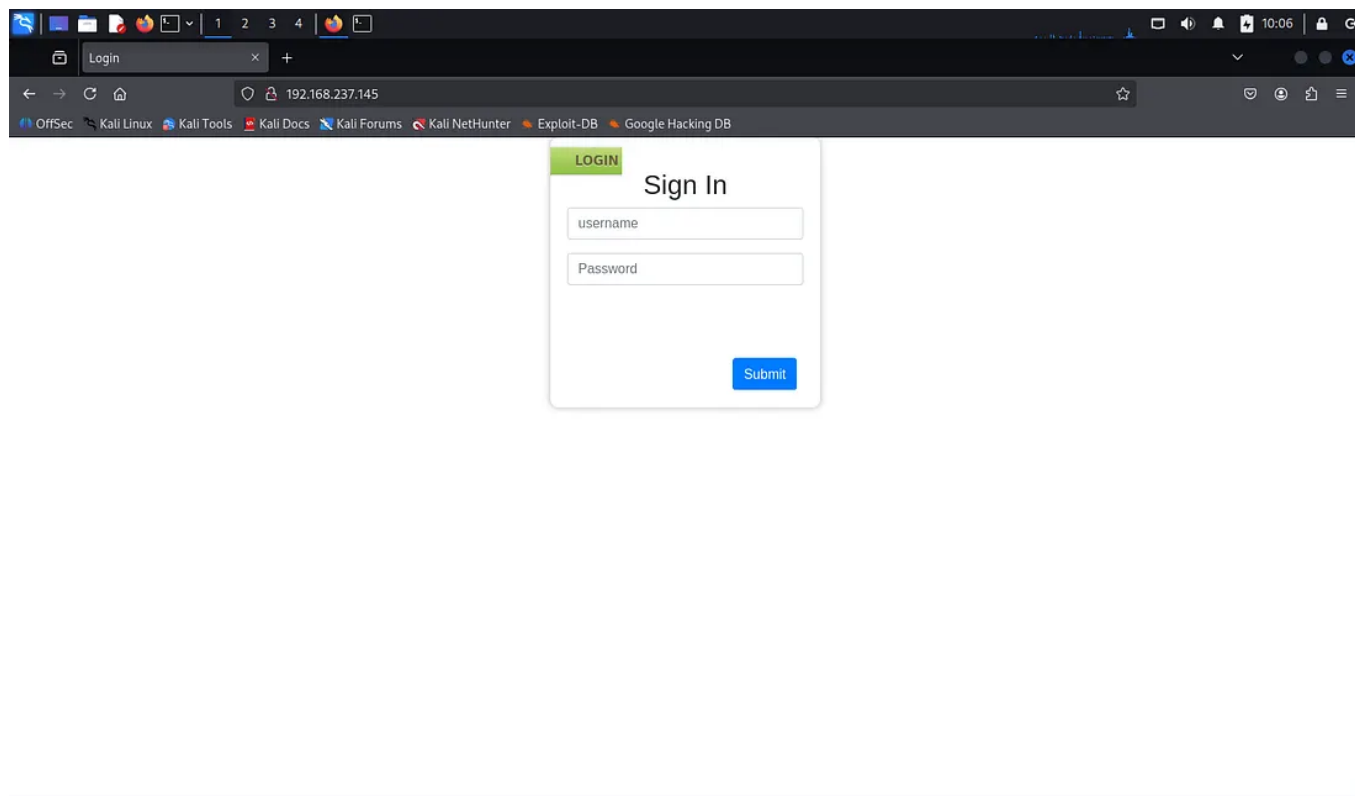
```
champ:password
```

Web Login Page

During enumeration, I visited the web service running on port **80**.

```
http://192.168.237.145
```

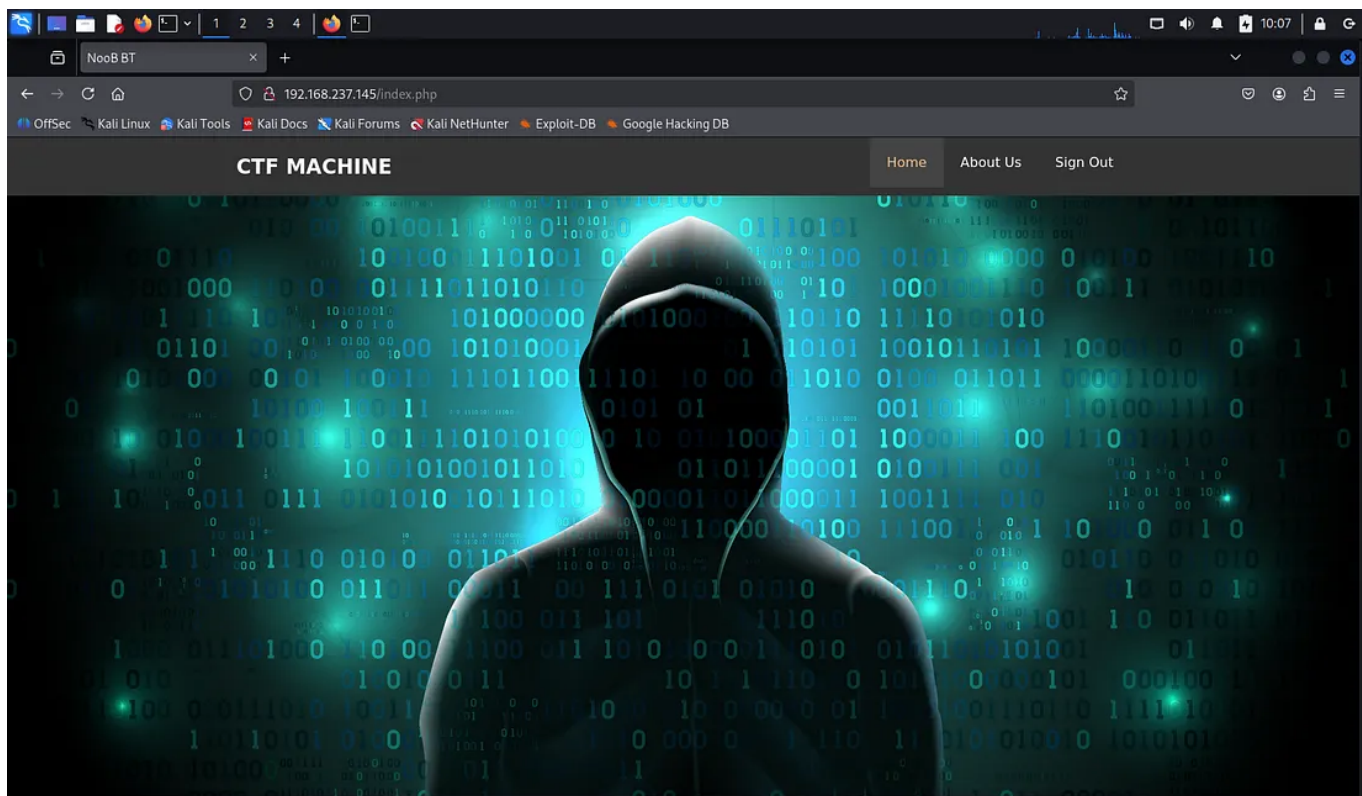
The page displayed a **login interface** with fields for a username and password. This indicates that the target machine hosts a **web application with an authentication system**. Since credentials were discovered earlier in the **cred.txt** file, they can be tested on this login page.



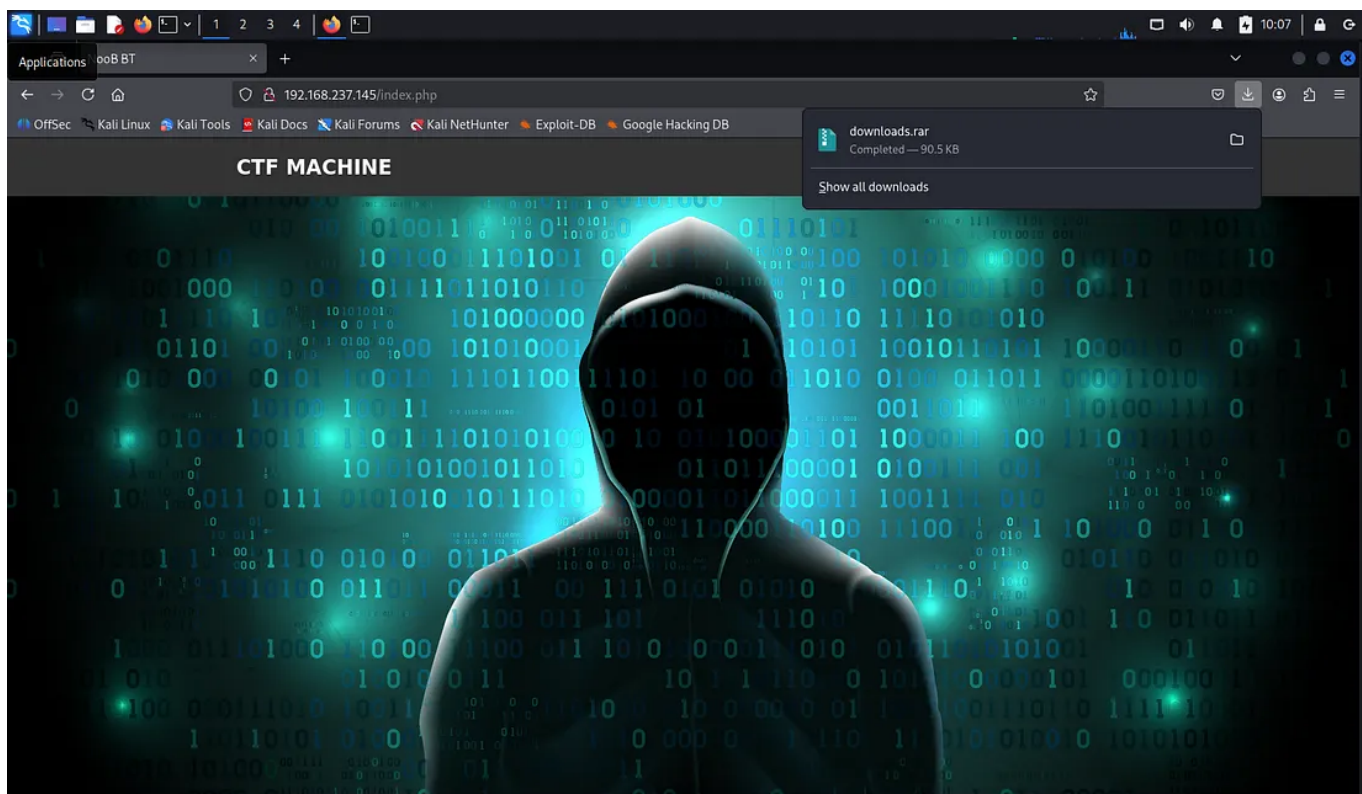
Credentials Found Earlier

```
Username champ  
Password password
```

Using that credentials I successfully logged into the web application.



After logging into the web application, I clicked the **About Us** section. A file named **downloads.rar** was automatically downloaded. This file was then examined for further information.



Extracting `downloads.rar`

After downloading the **downloads.rar** file, I extracted it to examine its contents.

```
unrar x downloads.rar
```

After extracting the archive, I found two image files:

```
funny.bmp  
funny.jpg
```

Since images in CTF challenges often contain hidden data, I used **Steghide** to check if any information was embedded inside the images.

Extracting Hidden Data

I attempted to extract hidden data from **funny.jpg** using the following command:

```
steghide extract -sf funny.jpg
```

Analyzing the Extracted File

I then examined the contents of the file:

```
cat hint.py
```

The file contained the following message:

```
This is not a python file but you are revolving around.  
well, try to rotate some words too.
```

I used `steghide` to extract hidden data from **funny.jpg**. This produced a file named **hint.py**. After reading the file, it suggested trying a **rotation-based cipher** to decode the hidden message.

```
root@kali: /home/kali/Downloads/downloads (2)
Session Actions Edit View Help
root@kali: /home/kali/Downloads/downloads (2)

(root@kali)-[/home/kali/Downloads/downloads (2)]
# ls
funny.bmp funny.jpg sudo

(root@kali)-[/home/kali/Downloads/downloads (2)]
# steghide extract -sf funny.jpg
Enter passphrase:
wrote extracted data to "hint.py".

(root@kali)-[/home/kali/Downloads/downloads (2)]
# ls
funny.bmp funny.jpg hint.py sudo

(root@kali)-[/home/kali/Downloads/downloads (2)]
# cat hint.py
This is_not a python file but you are revolving around.
well, try_ to rotate some words too.

(root@kali)-[/home/kali/Downloads/downloads (2)]
#
```

Extracting Data from `funny.bmp`

Since the previous hint suggested trying rotations, I checked the second image file **funny.bmp** for hidden data using **Steghide**.

```
steghide extract -sf funny.bmp
```

After running the command, hidden data was successfully extracted and saved to a file named:

```
user.txt
```

Reading the Extracted File

I then viewed the contents of the file using:

```
cat user.txt
```

The file contained the following encoded message:

```
jgs:guvf bar vf n fvzcyr bar
```

```
root@kali: /home/kali/Downloads/downloads (2)
Session Actions Edit View Help
root@kali: /home/kali/Downloads/downloads (2)

(root@kali)-[/home/kali/Downloads/downloads (2)]
# ls
funny.bmp funny.jpg hint.py sudo

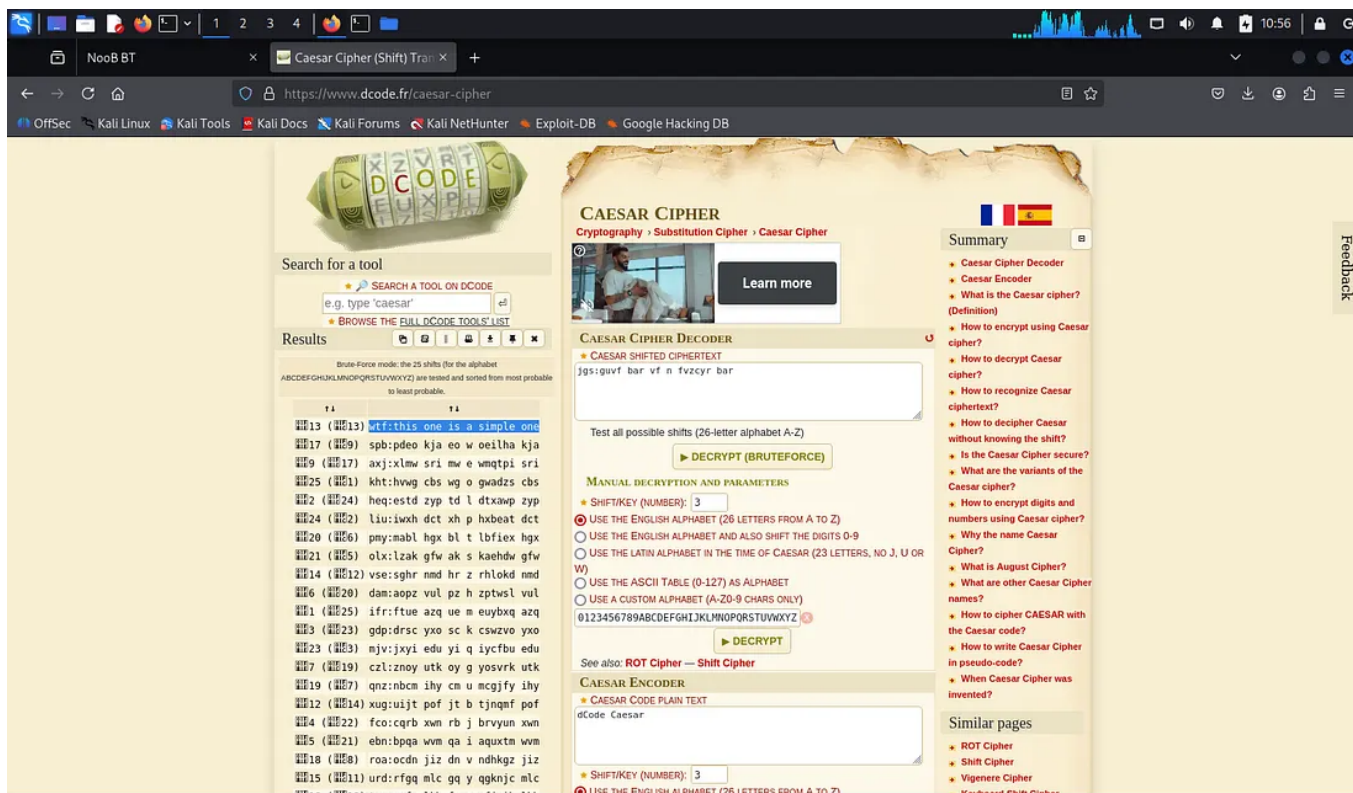
(root@kali)-[/home/kali/Downloads/downloads (2)]
# steghide extract -sf funny.bmp
Enter passphrase:
wrote extracted data to "user.txt".

(root@kali)-[/home/kali/Downloads/downloads (2)]
# cat user.txt
jgs:guvf bar vf n fvzcyr bar

(root@kali)-[/home/kali/Downloads/downloads (2)]
#
```

Based on the earlier hint suggesting **word rotation**, I suspected that the message was encoded using a **Caesar cipher (ROT13)**

To decode the message, I used an online Caesar cipher decoder:



After decoding the text using **ROT13**, the message revealed

```
wtf : this one is a simple one
```

The message in **user.txt** appeared to be encoded using a ROT cipher. After decoding it using ROT13, the result was “**wtf:this one is a simple one.**”

SSH Login

After decoding the message, I obtained the credentials:

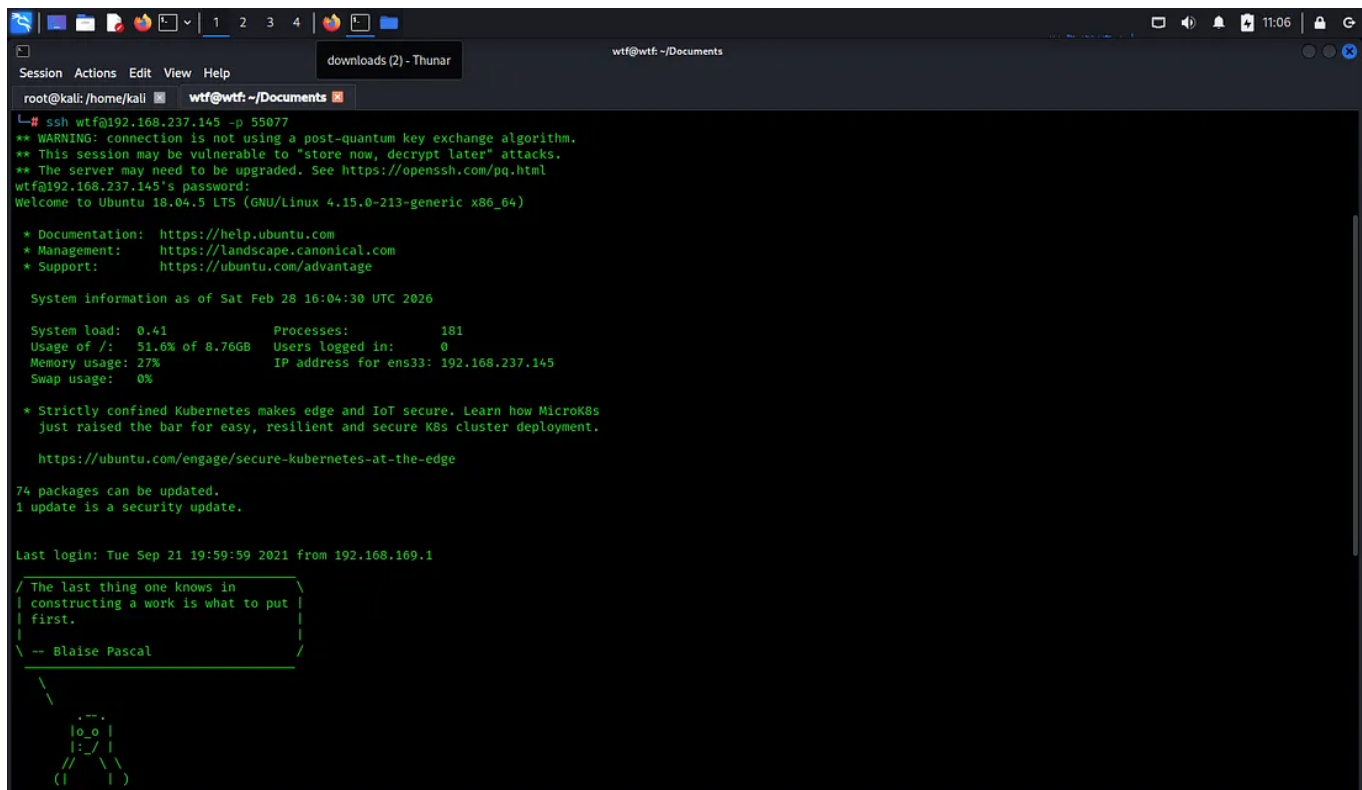
```
Username : wtf
Password : this one is a simple one
```

Since the **OpenSSH** service was running on port **55077**, I attempted to log in using these credentials.

Command Used

```
ssh wtf@192.168.237.145 -p 55077
```

Using the discovered credentials, I connected to the machine via SSH on port **55077** and successfully obtained a user shell.



```
Session Actions Edit View Help
root@kali: /home/kali wtf@wtf: ~/Documents
ssh wtf@192.168.237.145 -p 55077
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
wtf@192.168.237.145's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-213-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sat Feb 28 16:04:30 UTC 2026

System load:  0.41          Processes:    181
Usage of /:   51.6% of 8.76GB Users logged in:  0
Memory usage: 27%          IP address for ens33: 192.168.237.145
Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

74 packages can be updated.
1 update is a security update.

Last login: Tue Sep 21 19:59:59 2021 from 192.168.169.1

/ The last thing one knows in
| constructing a work is what to put |
| first.                             |
-- Blaise Pascal

  /
 /
|o_o|
|:~|
(  )
```

Privilege Escalation

Step 1 — Check sudo permissions

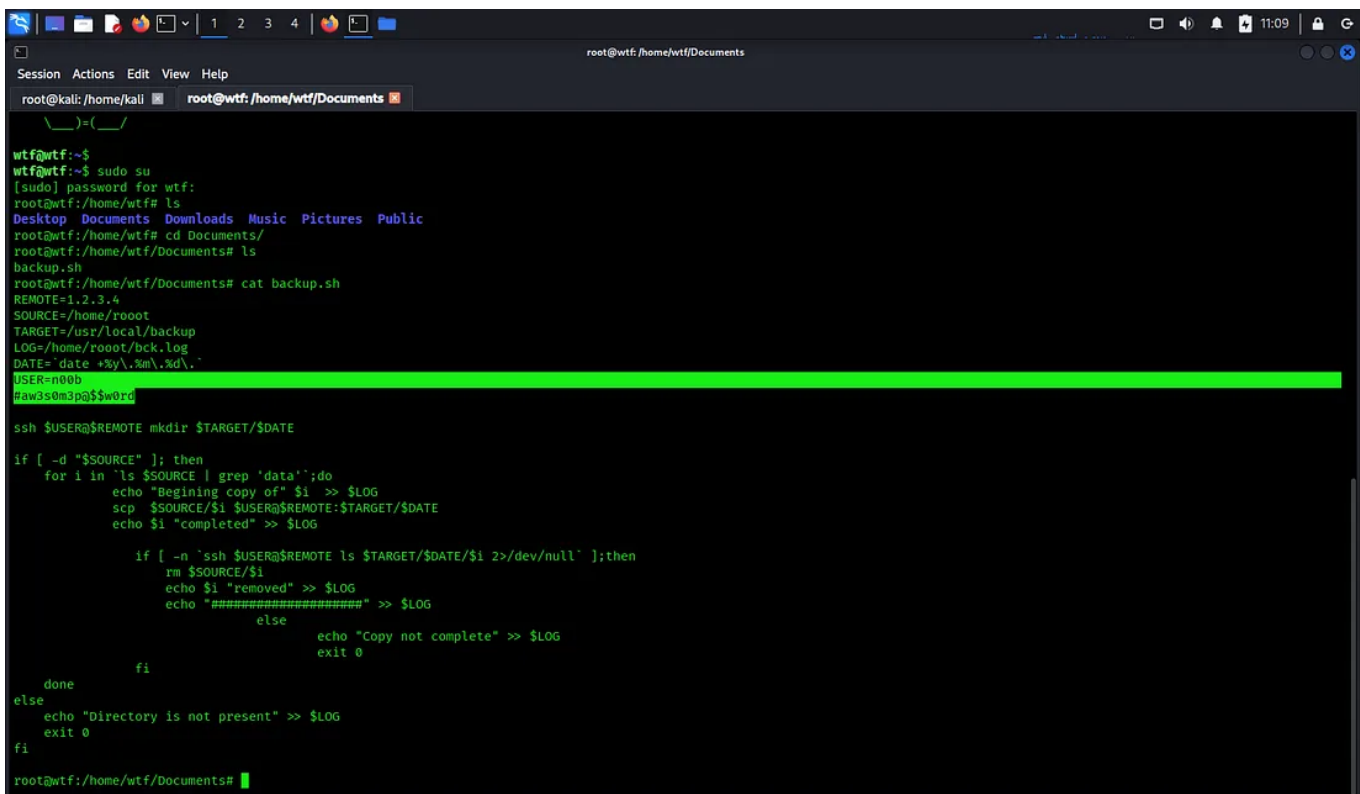
After logging in through SSH as the **wtf** user, I checked the sudo permissions. The output showed that the user could run **any command with sudo privileges**, which means privilege escalation is possible.

Step 2 — Switch to Root

Since sudo access was allowed, I switched to the root user.

```
sudo su
```

After entering the password, I successfully obtained **root access**.



```
root@kali: /home/kali | root@wtf: /home/wtf/Documents
wtf@wtf:~$
wtf@wtf:~$ sudo su
[sudo] password for wtf:
root@wtf: /home/wtf# ls
Desktop  Documents  Downloads  Music  Pictures  Public
root@wtf: /home/wtf# cd Documents/
root@wtf: /home/wtf/Documents# ls
backup.sh
root@wtf: /home/wtf/Documents# cat backup.sh
REMOTE=1,2,3,4
SOURCE=/home/rootoot
TARGET=/usr/local/backup
LOG=/home/rootoot/bck.log
DATE=$(date +%y%\m%\d\%)
USER=n00b
#aws0m3p@$$w0rd

ssh $USER@$REMOTE mkdir $TARGET/$DATE
if [ -d "$SOURCE" ]; then
  for i in $(ls $SOURCE | grep 'data');do
    echo "Beginning copy of" $i >> $LOG
    scp $SOURCE/$i $USER@$REMOTE:$TARGET/$DATE
    echo $i "completed" >> $LOG

    if [ -n "$(ssh $USER@$REMOTE ls $TARGET/$DATE/$i 2>/dev/null)" ];then
      rm $SOURCE/$i
      echo $i "removed" >> $LOG
      echo "#####" >> $LOG
    else
      echo "Copy not complete" >> $LOG
      exit 0
    fi
  done
else
  echo "Directory is not present" >> $LOG
  exit 0
fi
root@wtf: /home/wtf/Documents#
```

Post Exploitation and Flag Retrieval

Step 1 — Check User Directories

After gaining **root privileges**, I started exploring the user directories.

```
cd /home/wtf
ls
```

The directory contained several folders:

Desktop Documents Downloads Music Pictures Public

Step 2 — Locate the First Flag

I navigated to the **Downloads** directory.

```
cd Downloads
ls
```

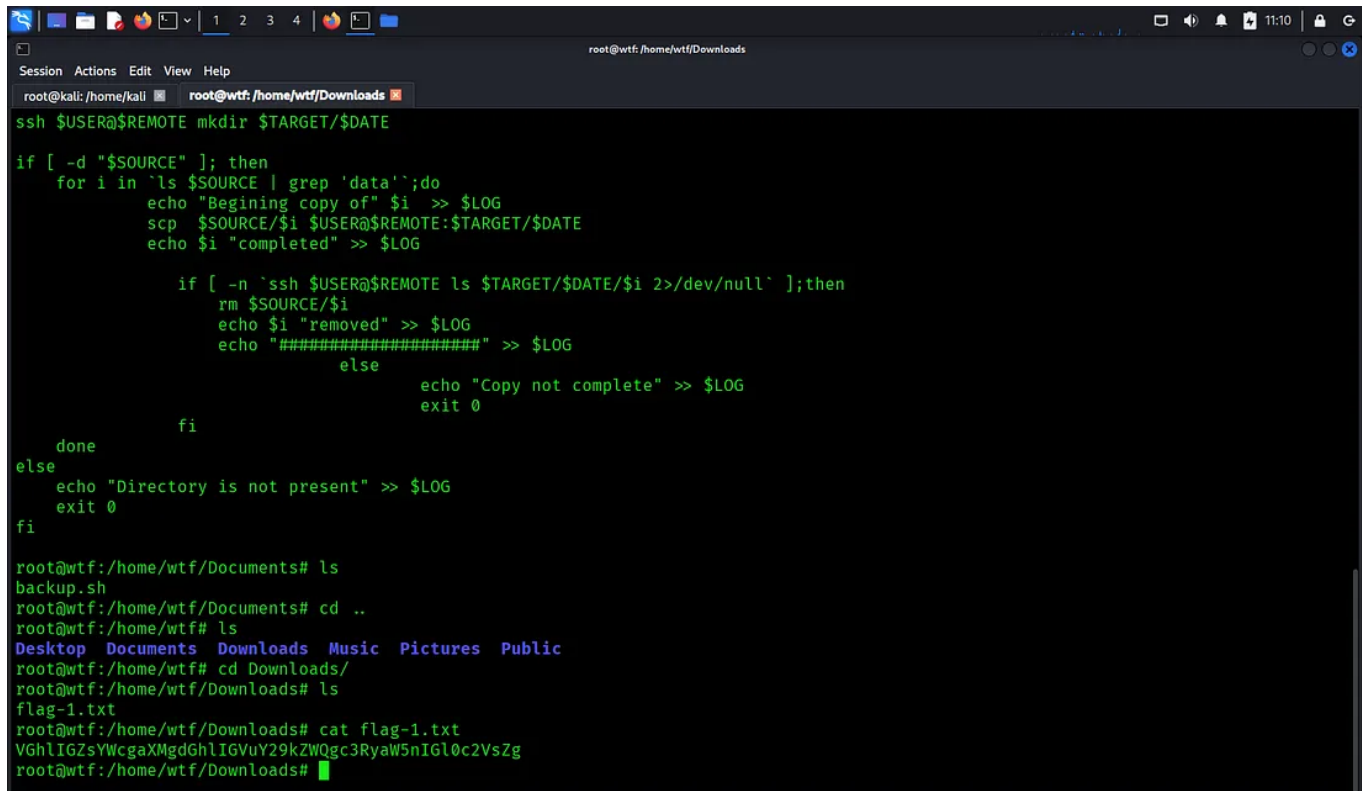
A flag file was found:

flag-1.txt

Output:

VGhlIGZsYWcgaXMgdGhIIIGVuY29kZWQgR3RyaW5nIGI0c2VsZg

This is the **first flag of the machine**.



```
root@wtf: /home/wtf/Downloads
ssh $USER@$REMOTE mkdir $TARGET/$DATE
if [ -d "$SOURCE" ]; then
  for i in `ls $SOURCE | grep 'data'`;do
    echo "Beginning copy of" $i >> $LOG
    scp $SOURCE/$i $USER@$REMOTE:$TARGET/$DATE
    echo $i "completed" >> $LOG
  fi
  if [ -n `ssh $USER@$REMOTE ls $TARGET/$DATE/$i 2>/dev/null` ];then
    rm $SOURCE/$i
    echo $i "removed" >> $LOG
    echo "#####" >> $LOG
  else
    echo "Copy not complete" >> $LOG
    exit 0
  fi
done
else
  echo "Directory is not present" >> $LOG
  exit 0
fi

root@wtf:/home/wtf/Documents# ls
backup.sh
root@wtf:/home/wtf/Documents# cd ..
root@wtf:/home/wtf# ls
Desktop Documents Downloads Music Pictures Public
root@wtf:/home/wtf# cd Downloads/
root@wtf:/home/wtf/Downloads# ls
flag-1.txt
root@wtf:/home/wtf/Downloads# cat flag-1.txt
VGhlIGZsYWcgaXMgdGhIIIGVuY29kZWQgR3RyaW5nIGI0c2VsZg
root@wtf:/home/wtf/Downloads#
```

Step 3 — Search for Additional Clues

Next, I explored another directory:

```
cd /home/rooooot  
ls
```

I found another file:

```
flag.txt
```

Reading the file:

```
cat flag.txt
```

Output

```
A Rabbit Hole?  
Not sure!  
  
Well, look for the thing you want.  
It's just 2 steps ahead :)
```

After retrieving the first flag, I continued exploring the system to find more clues.

To move two directories back, I used:

```
cd ../../
```

This command takes the current directory **two levels up** in the directory structure. After moving back, I listed the directories to see what else was available.

```
root@wtf:/home/root
Session Actions Edit View Help
root@kali:/home/kali root@wtf:/home/root

done
fi
else
echo "Directory is not present" >> $LOG
exit 0
fi

root@wtf:/home/wtf/Documents# ls
backup.sh
root@wtf:/home/wtf/Documents# cd ..
root@wtf:/home/wtf# ls
Desktop Documents Downloads Music Pictures Public
root@wtf:/home/wtf# cd Downloads/
root@wtf:/home/wtf/Downloads# ls
flag-1.txt
root@wtf:/home/wtf/Downloads# cat flag-1.txt
VGhIGZsYWcgaXMgdGhlIGVuY29kZWQgc3RyaW5nIGl0c2VsZg
root@wtf:/home/wtf/Downloads# cd ..
root@wtf:/home/wtf# ls
Desktop Documents Downloads Music Pictures Public
root@wtf:/home/wtf# cd ..
root@wtf:/home# ls
root wtf
root@wtf:/home# cd rooot/
root@wtf:/home/rooot# ls
flag.txt
root@wtf:/home/rooot# cat flag.txt
A Rabbit Hole?
Not sure!

Well, look for the thing you want.
It's just 2 steps ahead :)
root@wtf:/home/rooot#
```

```
root@wtf:~
Session Actions Edit View Help
root@kali:/home/kali root@wtf:~

root@wtf:/home/wtf# cd ..
root@wtf:/home# ls
root wtf
root@wtf:/home# cd rooot/
root@wtf:/home/rooot# ls
flag.txt
root@wtf:/home/rooot# cat flag.txt
A Rabbit Hole?
Not sure!

Well, look for the thing you want.
It's just 2 steps ahead :)
root@wtf:/home/rooot# cd ../..
root@wtf:/# ls
bin cdrom etc initrd.img lib lost+found mnt proc run snap swap.img tmp var vmlinuz.old
boot dev home initrd.img.old lib64 media opt root sbin srv sys usr vmlinuz
root@wtf:/# cd root/
root@wtf:/root# ls
root.txt snap
root@wtf:/root# cat root.txt
RW5kb3JzZSBtZSBvbiBsaW5rZWRpbiA9PiBodHRwczovL3d3dy5saW5rZWRpbi5jb20vaW4vZGVlcGFrLWFoZWVvCg==

Follow me on Twitter https://www.twitter.com/Deepakhr9

TryHackMe → https://www.tryhackme.com/p/Malwre99
Github → https://www.github.com/Deepak-Aheer
(the flag is my LinkedIn username)

THANK YOU for PLAYING THIS CTF

But REMEMBER we're still N00bs ;)
root@wtf:/root#
```

Then I navigated to the root user's directory.

```
cd /root
ls
```

Here I found the final flag file:

root.txt

Conclusion

This machine was compromised using the following steps:

1. Network discovery
2. Port scanning and service enumeration
3. Anonymous FTP access
4. Steganography analysis
5. Cipher decoding
6. SSH login
7. Privilege escalation using sudo
8. Retrieving user and root flags

NOOB:1 machine successfully completed.