Name: Miftahul Huq

Course: Network Security

Course Prefix: CSEC 744

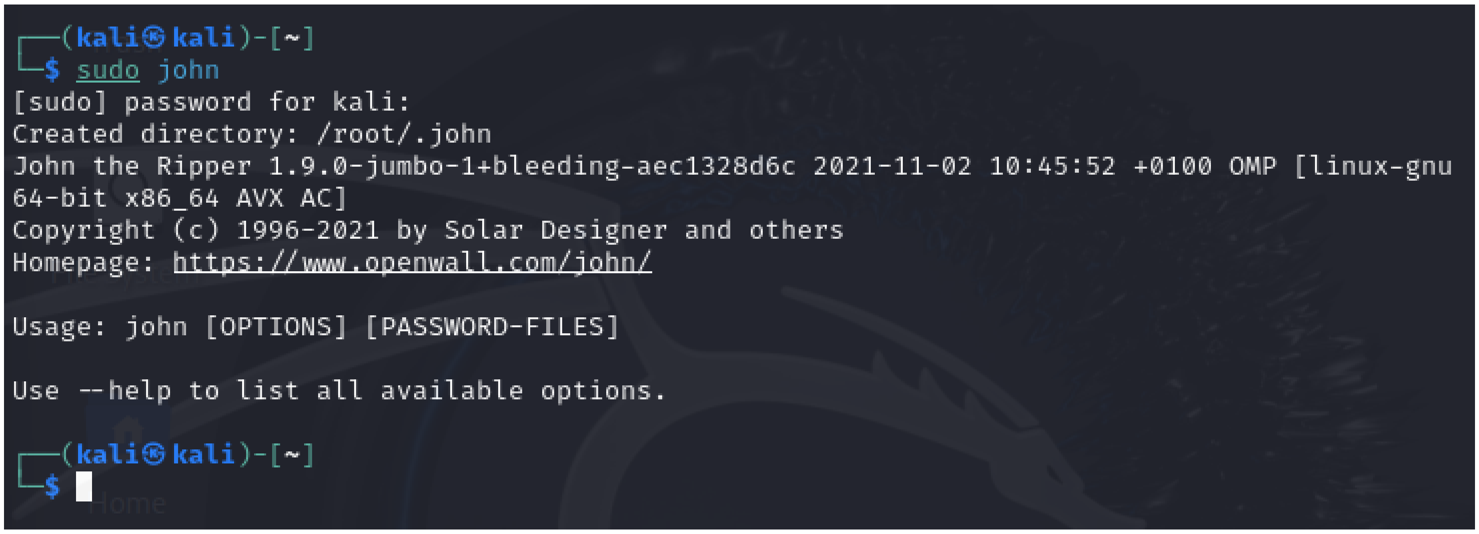
Section: 01

Chapter 11: Authentication and Remote Access

Date: 04/08/2024

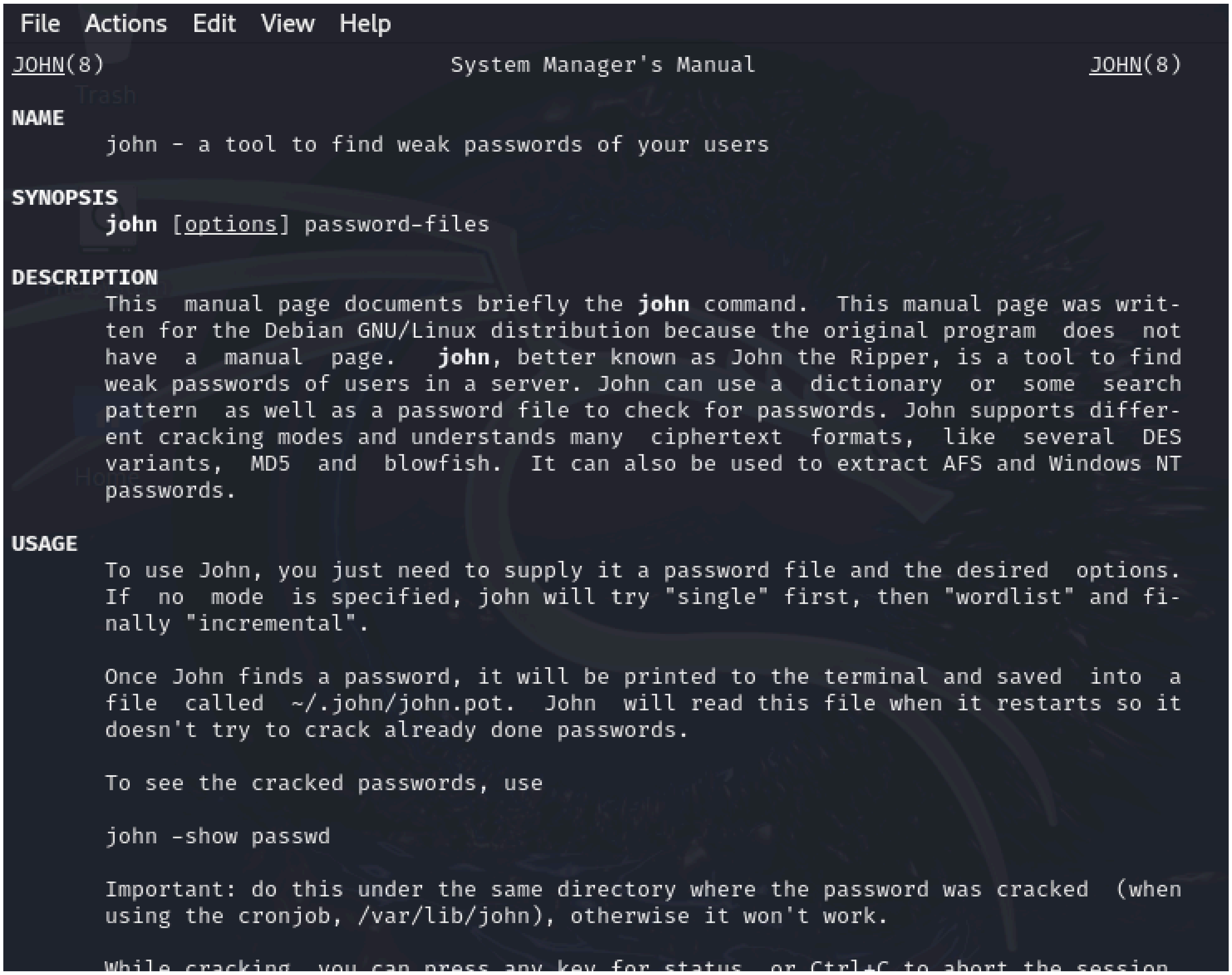
Lab Exercise 11.01: Dictionary Attacks on Linux Passwords with John the Ripper

Step 1a:



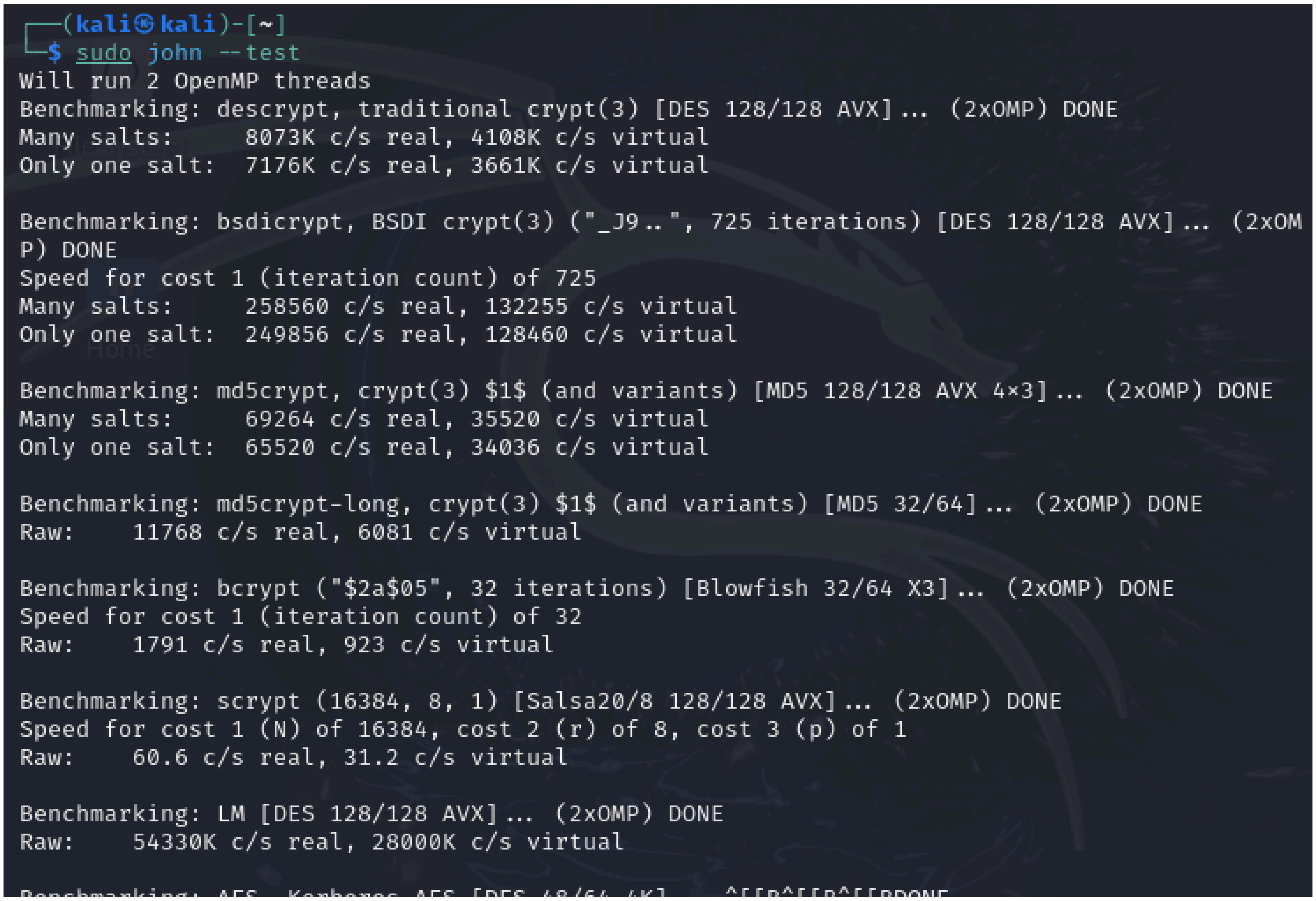
```
┌──(kali㉿kali)-[~]
└─$ sudo john
[sudo] password for kali:
Created directory: /root/.john
John the Ripper 1.9.0-jumbo-1+bleeding-aec1328d6c 2021-11-02 10:45:52 +0100 OMP [linux-gnu
64-bit x86_64 AVX AC]
Copyright (c) 1996-2021 by Solar Designer and others
Homepage: https://www.openwall.com/john/

Usage: john [OPTIONS] [PASSWORD-FILES]

Use --help to list all available options.

┌──(kali㉿kali)-[~]
└─$ 
```

Step 1b:



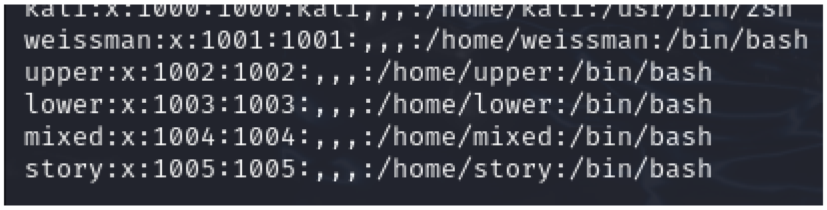
```
JOHN(8)                          System Manager's Manual                          JOHN(8)

NAME
       john - a tool to find weak passwords of your users

SYNOPSIS
       john [options] password-files

DESCRIPTION
       This  manual page documents briefly the john command.  This manual page was writ-
       ten for the Debian GNU/Linux distribution because the original program  does  not
       have  a  manual  page.   john, better known as John the Ripper, is a tool to find
       weak passwords of users in a server. John can use a  dictionary  or  some  search
       pattern  as well as a password file to check for passwords. John supports differ-
       ent cracking modes and understands many ciphertext  formats,  like  several  DES
       variants,  MD5  and  blowfish.  It can also be used to extract AFS and Windows NT
       passwords.

USAGE
       To use John, you just need to supply it a password file and the desired  options.
       If  no  mode  is specified, john will try "single" first, then "wordlist" and fi-
       nally "incremental".

       Once John finds a password, it will be printed to the terminal and saved  into  a
       file  called  ~/.john/john.pot.  John  will read this file when it restarts so it
       doesn't try to crack already done passwords.

       To see the cracked passwords, use

       john -show passwd

       Important: do this under the same directory where the password was cracked  (when
       using the cronjob, /var/lib/john), otherwise it won't work.

       While cracking, you can press any key for status, or Ctrl+C to abort the session,
```

Step 1c:

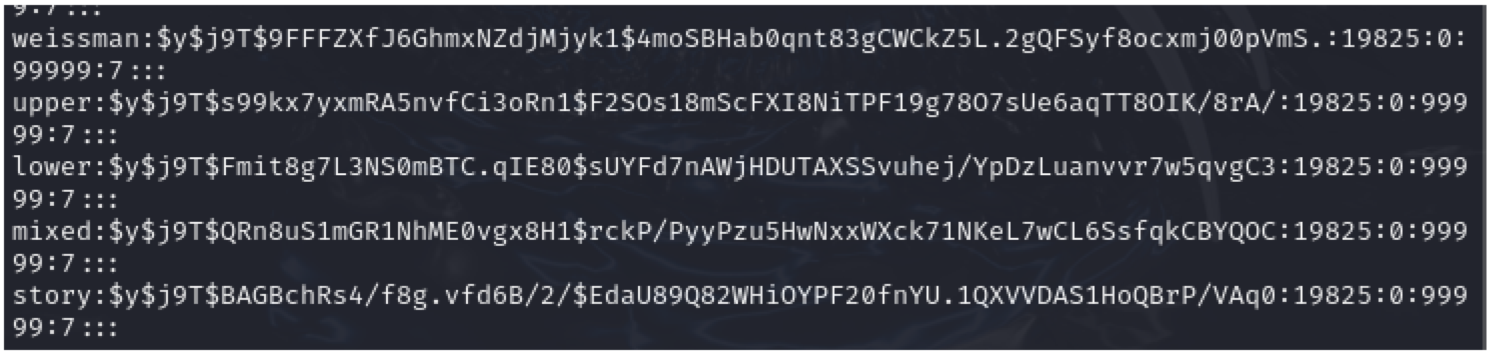


```
┌──(kali㉿kali)-[~]
└─$ sudo john --test
Will run 2 OpenMP threads
Benchmarking: descrypt, traditional crypt(3) [DES 128/128 AVX]... (2xOMP) DONE
Many salts:     8073K c/s real, 4108K c/s virtual
Only one salt:  7176K c/s real, 3661K c/s virtual

Benchmarking: bsdicrypt, BSDI crypt(3) ("_J9..", 725 iterations) [DES 128/128 AVX]... (2xOM
P) DONE
Speed for cost 1 (iteration count) of 725
Many salts:     258560 c/s real, 132255 c/s virtual
Only one salt:  249856 c/s real, 128460 c/s virtual

Benchmarking: md5crypt, crypt(3) $1$ (and variants) [MD5 128/128 AVX 4×3]... (2xOMP) DONE
Many salts:     69264 c/s real, 35520 c/s virtual
Only one salt:  65520 c/s real, 34036 c/s virtual

Benchmarking: md5crypt-long, crypt(3) $1$ (and variants) [MD5 32/64]... (2xOMP) DONE
Raw:    11768 c/s real, 6081 c/s virtual

Benchmarking: bcrypt ("$2a$05", 32 iterations) [Blowfish 32/64 X3]... (2xOMP) DONE
Speed for cost 1 (iteration count) of 32
Raw:    1791 c/s real, 923 c/s virtual

Benchmarking: scrypt (16384, 8, 1) [Salsa20/8 128/128 AVX]... (2xOMP) DONE
Speed for cost 1 (N) of 16384, cost 2 (r) of 8, cost 3 (p) of 1
Raw:    60.6 c/s real, 31.2 c/s virtual

Benchmarking: LM [DES 128/128 AVX]... (2xOMP) DONE
Raw:    54330K c/s real, 28000K c/s virtual

Benchmarking: AFS, Kerberos AFS [DES 48/64 4K]... ^[[B^[[B^[[BDONE
```

Step 1d - f:

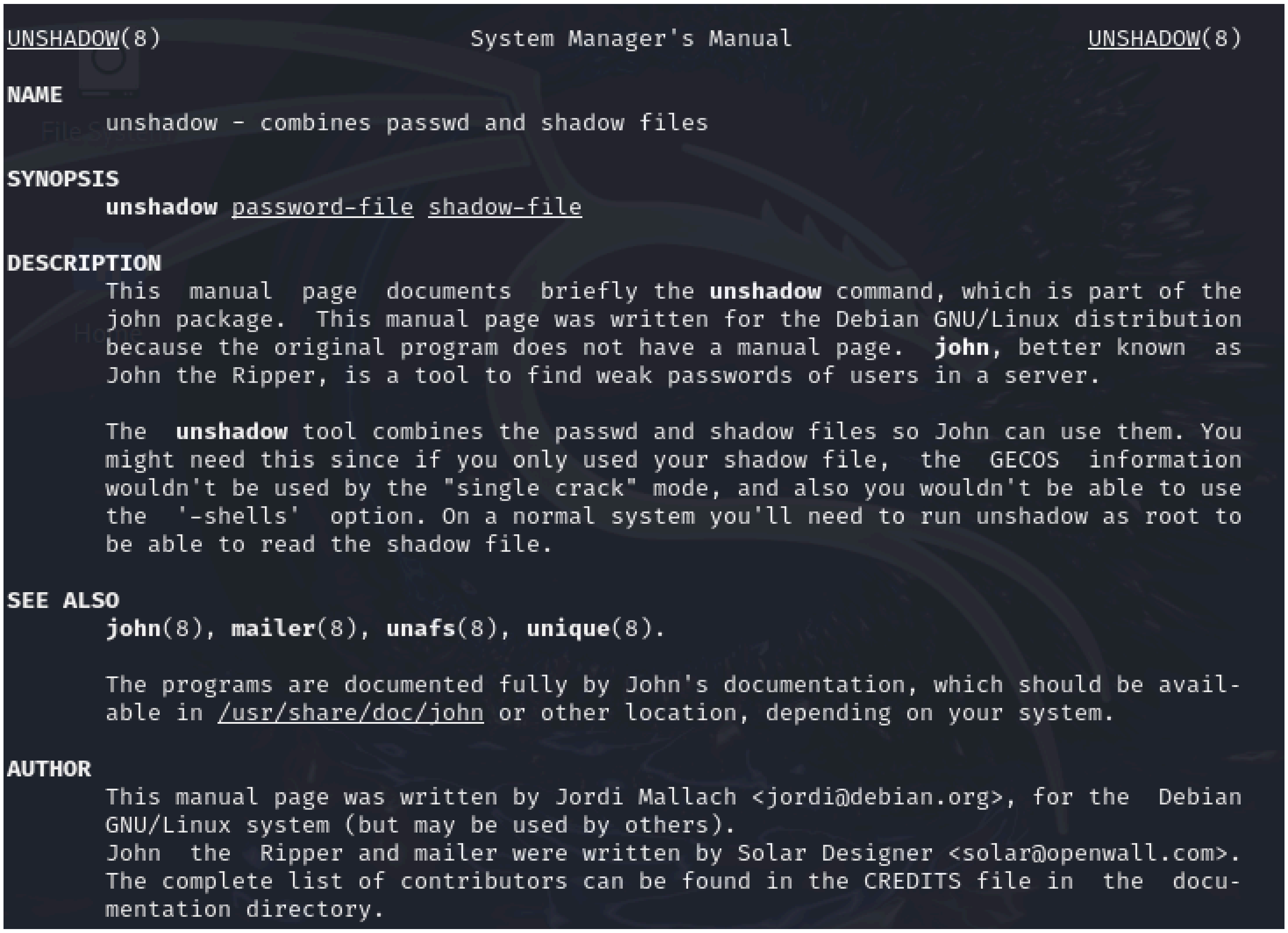


```
kali:x:1000:1000:kali,,,:/home/kali:/usr/bin/zsh
weissman:x:1001:1001:,,,:/home/weissman:/bin/bash
upper:x:1002:1002:,,,:/home/upper:/bin/bash
lower:x:1003:1003:,,,:/home/lower:/bin/bash
mixed:x:1004:1004:,,,:/home/mixed:/bin/bash
story:x:1005:1005:,,,:/home/story:/bin/bash
```



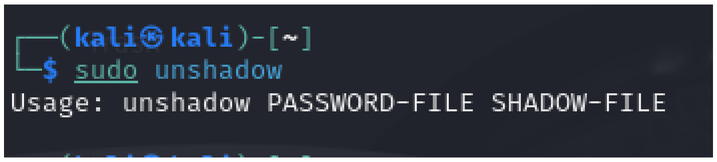
```
9:7:::
weissman:$y$j9T$9FFFZXfJ6GhmxNZdjMjyk1$4moSBHab0qnt83gCWCkZ5L.2gQFSyf8ocxmj00pVmS.:19825:0:
99999:7:::
upper:$y$j9T$s99kx7yxmRA5nvfCi3oRn1$F2SOs18mScFXI8NiTPF19g78O7sUe6aqTT8OIK/8rA/:19825:0:999
99:7:::
lower:$y$j9T$Fmit8g7L3NS0mBTC.qIE80$sUYFd7nAWjHDUTAXSSvuhej/YpDzLuanvvr7w5qvgC3:19825:0:999
99:7:::
mixed:$y$j9T$QRn8uS1mGR1NhME0vgx8H1$rckP/PyyPzu5HwNxxWXck71NKeL7wCL6SsfqkCBYQOC:19825:0:999
99:7:::
story:$y$j9T$BAGBchRs4/f8g.vfd6B/2/$EdaU89Q82WHiOYPF20fnYU.1QXVVDAS1HoQBrP/VAq0:19825:0:999
99:7:::
```

Step 1g:



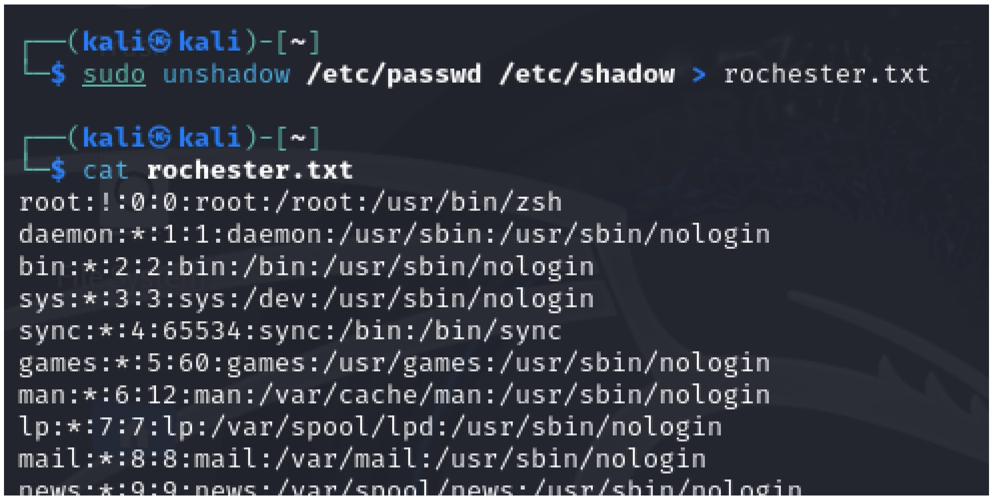
```
UNSHADOW(8)                  System Manager's Manual                 UNSHADOW(8)

NAME
       unshadow - combines passwd and shadow files

SYNOPSIS
       unshadow password-file shadow-file

DESCRIPTION
       This  manual  page  documents  briefly the unshadow command, which is part of the
       john package.  This manual page was written for the Debian GNU/Linux distribution
       because the original program does not have a manual page.  john, better known  as
       John the Ripper, is a tool to find weak passwords of users in a server.

       The  unshadow tool combines the passwd and shadow files so John can use them. You
       might need this since if you only used your shadow file,  the  GECOS  information
       wouldn't be used by the "single crack" mode, and also you wouldn't be able to use
       the  '-shells'  option. On a normal system you'll need to run unshadow as root to
       be able to read the shadow file.

SEE ALSO
       john(8), mailer(8), unafs(8), unique(8).

       The programs are documented fully by John's documentation, which should be avail-
       able in /usr/share/doc/john or other location, depending on your system.

AUTHOR
       This manual page was written by Jordi Mallach <jordi@debian.org>, for the  Debian
       GNU/Linux system (but may be used by others).
       John  the  Ripper and mailer were written by Solar Designer <solar@openwall.com>.
       The complete list of contributors can be found in the CREDITS file in  the  docu-
       mentation directory.
```

Step 1h:



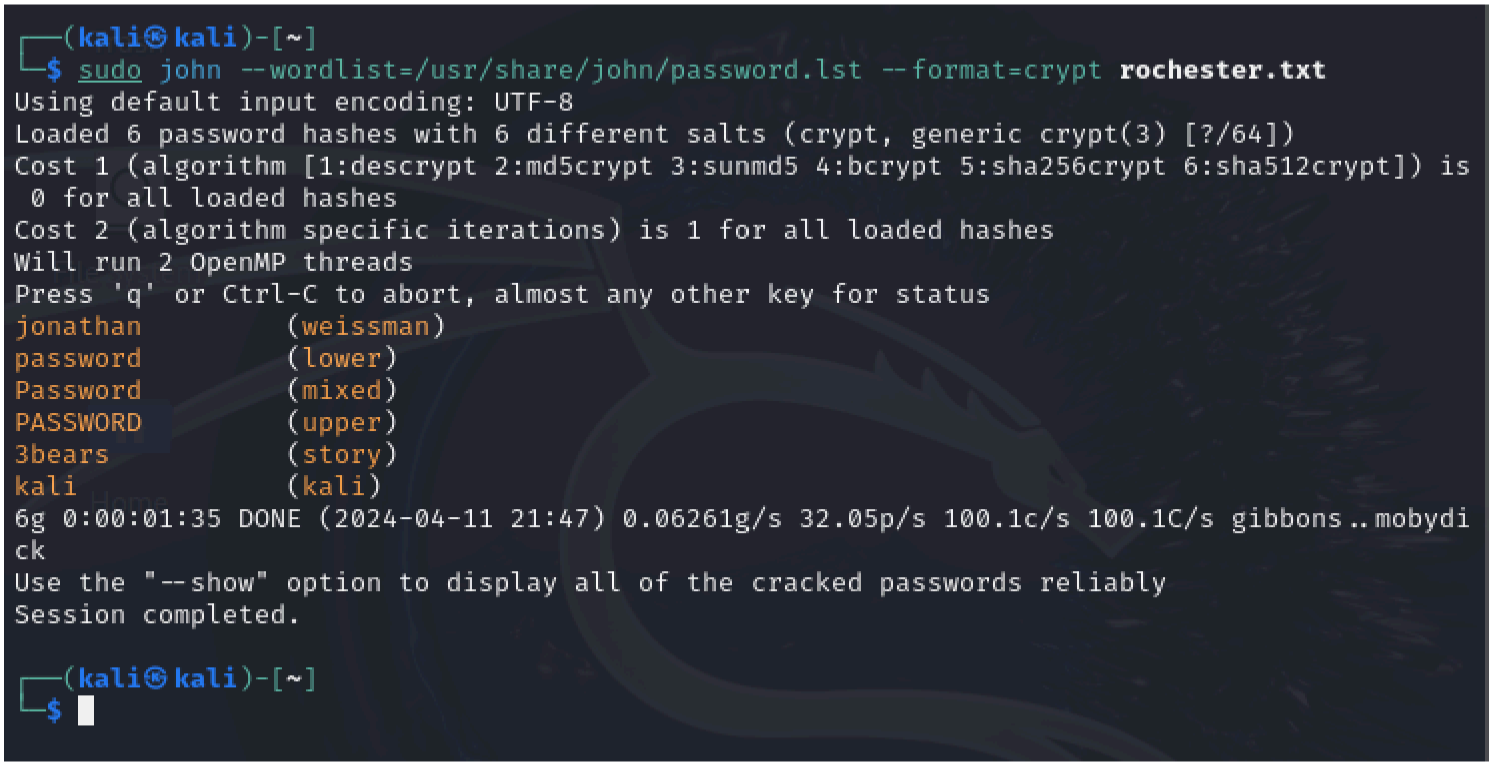
```
┌──(kali㉿kali)-[~]
└─$ sudo unshadow
Usage: unshadow PASSWORD-FILE SHADOW-FILE
```

Step 1i - j:



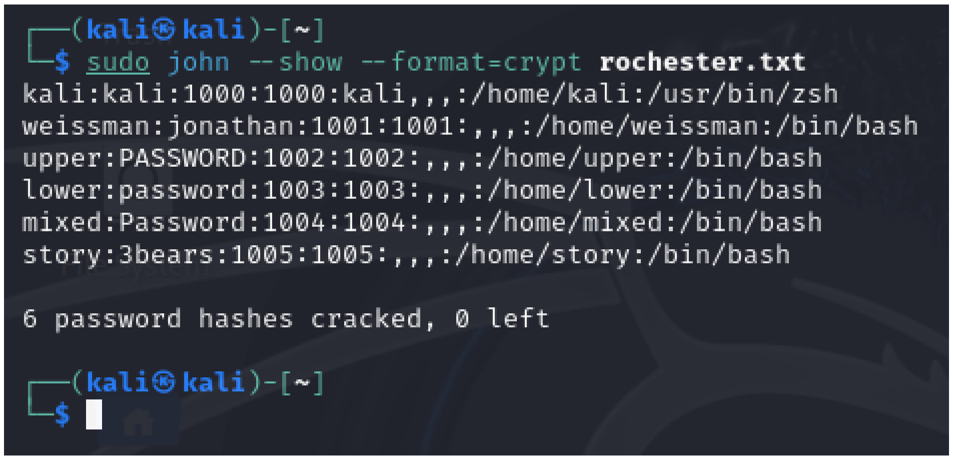
```
┌──(kali㉿kali)-[~]
└─$ sudo unshadow /etc/passwd /etc/shadow > rochester.txt

┌──(kali㉿kali)-[~]
└─$ cat rochester.txt
root:!:0:0:root:/root:/usr/bin/zsh
daemon:*:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:*:2:2:bin:/bin:/usr/sbin/nologin
sys:*:3:3:sys:/dev:/usr/sbin/nologin
sync:*:4:65534:sync:/bin:/bin/sync
games:*:5:60:games:/usr/games:/usr/sbin/nologin
man:*:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:*:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:*:8:8:mail:/var/mail:/usr/sbin/nologin
news:*:9:9:news:/var/spool/news:/usr/sbin/nologin
```

Step 1k:



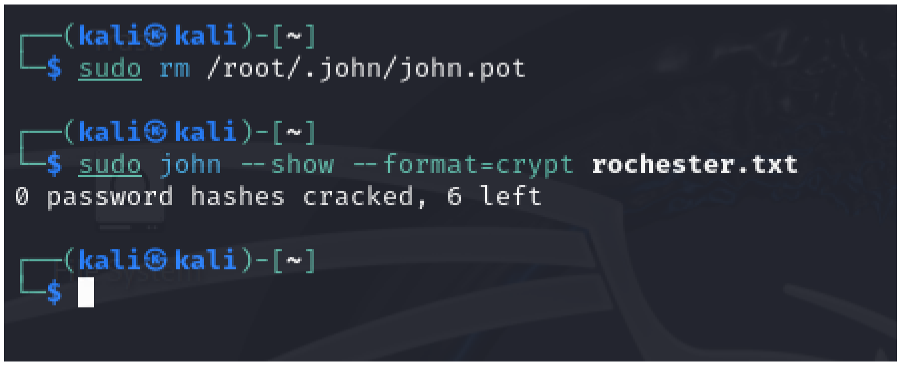
```
┌──(kali㉿kali)-[~]
└─$ sudo john --wordlist=/usr/share/john/password.lst --format=crypt rochester.txt
Using default input encoding: UTF-8
Loaded 6 password hashes with 6 different salts (crypt, generic crypt(3) [?/64])
Cost 1 (algorithm [1:descrypt 2:md5crypt 3:sunmd5 4:bcrypt 5:sha256crypt 6:sha512crypt]) is
 0 for all loaded hashes
Cost 2 (algorithm specific iterations) is 1 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
jonathan         (weissman)
password         (lower)
Password         (mixed)
PASSWORD         (upper)
3bears           (story)
kali             (kali)
6g 0:00:01:35 DONE (2024-04-11 21:47) 0.06261g/s 32.05p/s 100.1c/s 100.1C/s gibbons..mobydi
ck
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

┌──(kali㉿kali)-[~]
└─$ 
```

Step 1l:



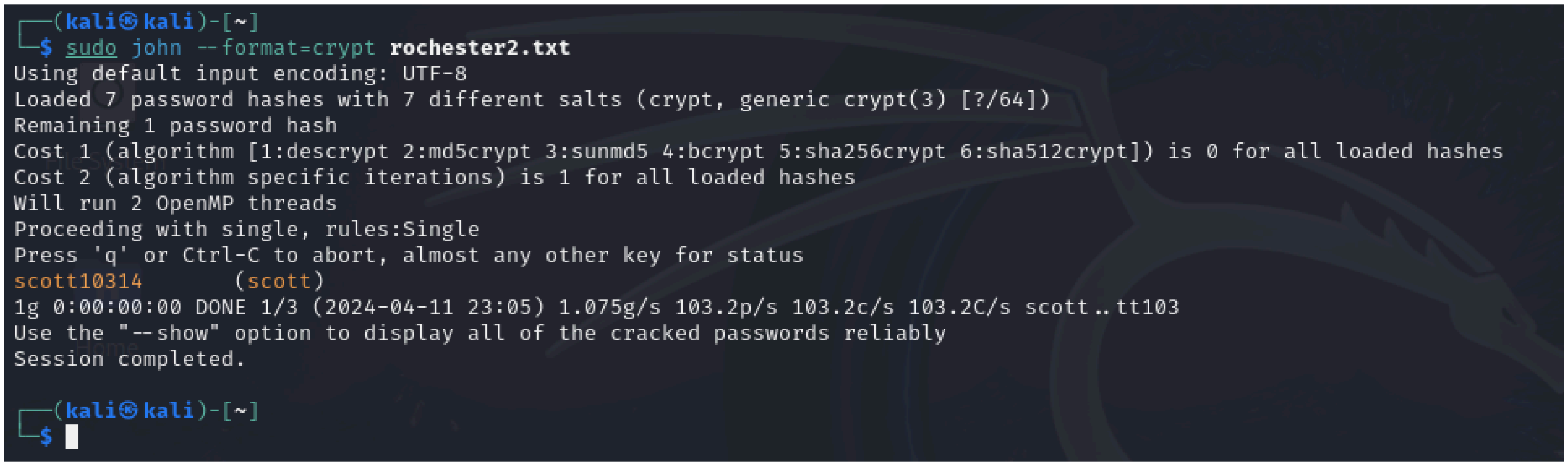
```
┌──(kali㉿kali)-[~]
└─$ sudo john --show --format=crypt rochester.txt
kali:kali:1000:1000:kali,,,:/home/kali:/usr/bin/zsh
weissman:jonathan:1001:1001:,,,:/home/weissman:/bin/bash
upper:PASSWORD:1002:1002:,,,:/home/upper:/bin/bash
lower:password:1003:1003:,,,:/home/lower:/bin/bash
mixed:Password:1004:1004:,,,:/home/mixed:/bin/bash
story:3bears:1005:1005:,,,:/home/story:/bin/bash

6 password hashes cracked, 0 left

┌──(kali㉿kali)-[~]
└─$ 
```

Step 1m:



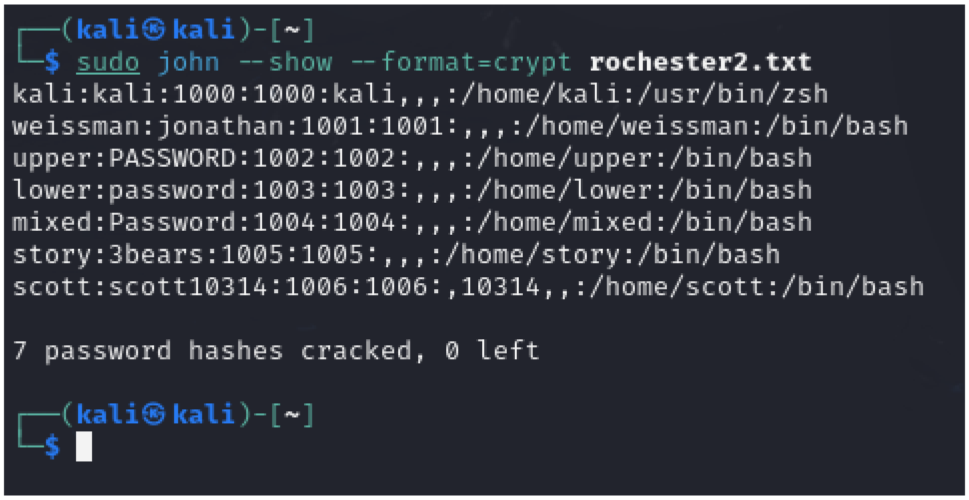
```
┌──(kali㉿kali)-[~]
└─$ sudo rm /root/.john/john.pot

┌──(kali㉿kali)-[~]
└─$ sudo john --show --format=crypt rochester.txt
0 password hashes cracked, 6 left

┌──(kali㉿kali)-[~]
└─$ 
```

Step n - p:



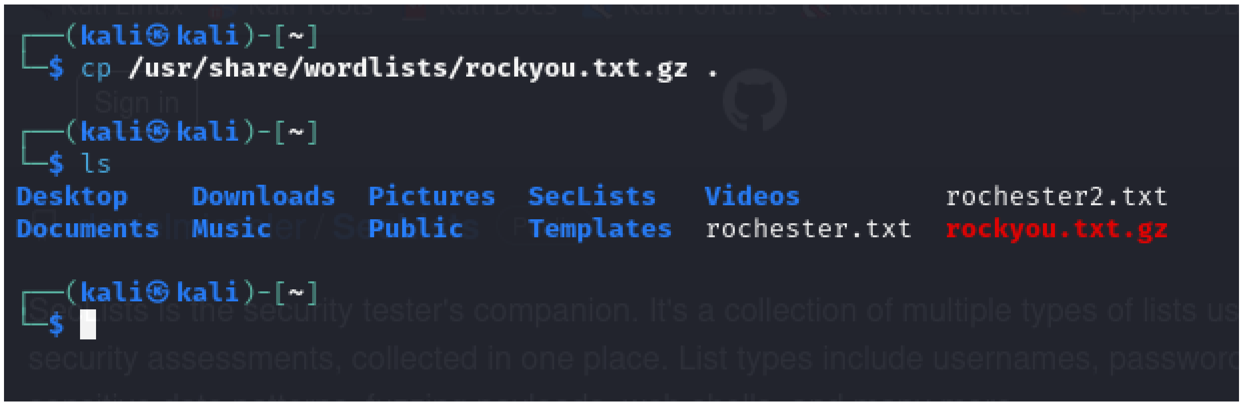
```
┌──(kali㉿kali)-[~]
└─$ sudo john --format=crypt rochester2.txt
Using default input encoding: UTF-8
Loaded 7 password hashes with 7 different salts (crypt, generic crypt(3) [?/64])
Remaining 1 password hash
Cost 1 (algorithm [1:descrypt 2:md5crypt 3:sunmd5 4:bcrypt 5:sha256crypt 6:sha512crypt]) is 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
scott10314       (scott)
1g 0:00:00:00 DONE 1/3 (2024-04-11 23:05) 1.075g/s 103.2p/s 103.2c/s 103.2C/s scott..tt103
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

┌──(kali㉿kali)-[~]
└─$ 
```

Step q:



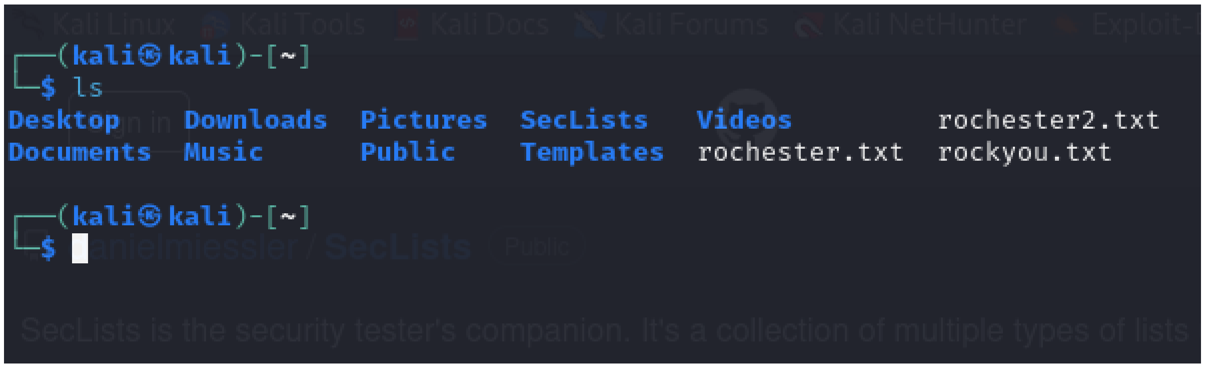
```
┌──(kali㉿kali)-[~]
└─$ sudo john --show --format=crypt rochester2.txt
kali:kali:1000:1000:kali,,,:/home/kali:/usr/bin/zsh
weissman:jonathan:1001:1001:,,,:/home/weissman:/bin/bash
upper:PASSWORD:1002:1002:,,,:/home/upper:/bin/bash
lower:password:1003:1003:,,,:/home/lower:/bin/bash
mixed:Password:1004:1004:,,,:/home/mixed:/bin/bash
story:3bears:1005:1005:,,,:/home/story:/bin/bash
scott:scott10314:1006:1006:,10314,,:/home/scott:/bin/bash

7 password hashes cracked, 0 left

┌──(kali㉿kali)-[~]
└─$ 
```

Step 2a:



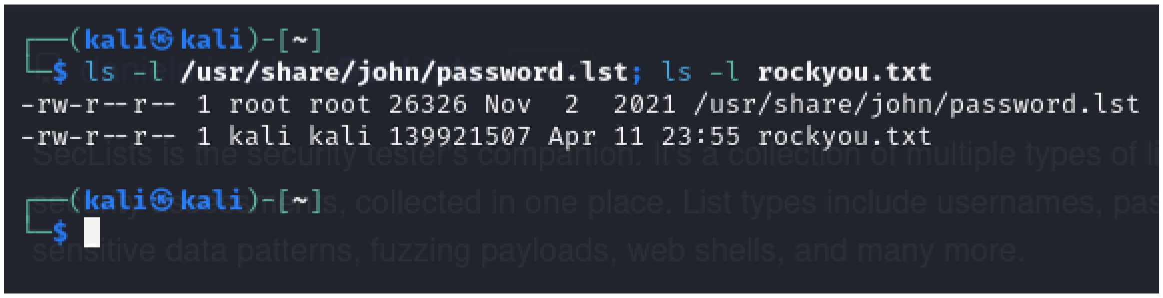
```
┌──(kali㉿kali)-[~]
└─$ cp /usr/share/wordlists/rockyou.txt.gz .

┌──(kali㉿kali)-[~]
└─$ ls
Desktop    Downloads  Pictures  SecLists   Videos        rochester2.txt
Documents  Music      Public    Templates  rochester.txt rockyou.txt.gz

┌──(kali㉿kali)-[~]
└─$ 
```

Step 2b:



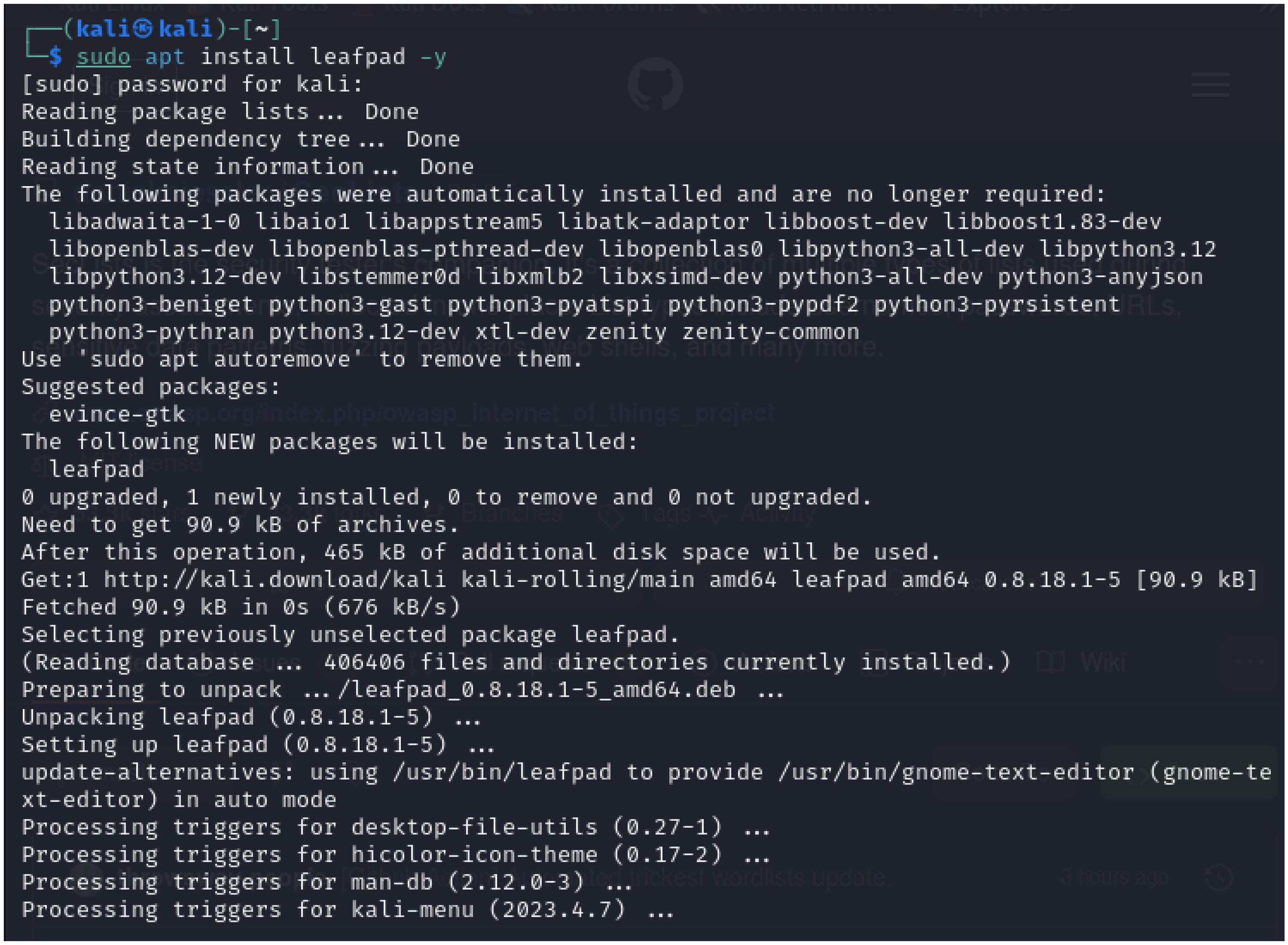
```
┌──(kali㉿kali)-[~]
└─$ ls
Desktop    Downloads  Pictures  SecLists   Videos        rochester2.txt
Documents  Music      Public    Templates  rochester.txt rockyou.txt

┌──(kali㉿kali)-[~]
└─$ 
```

Step 2c:



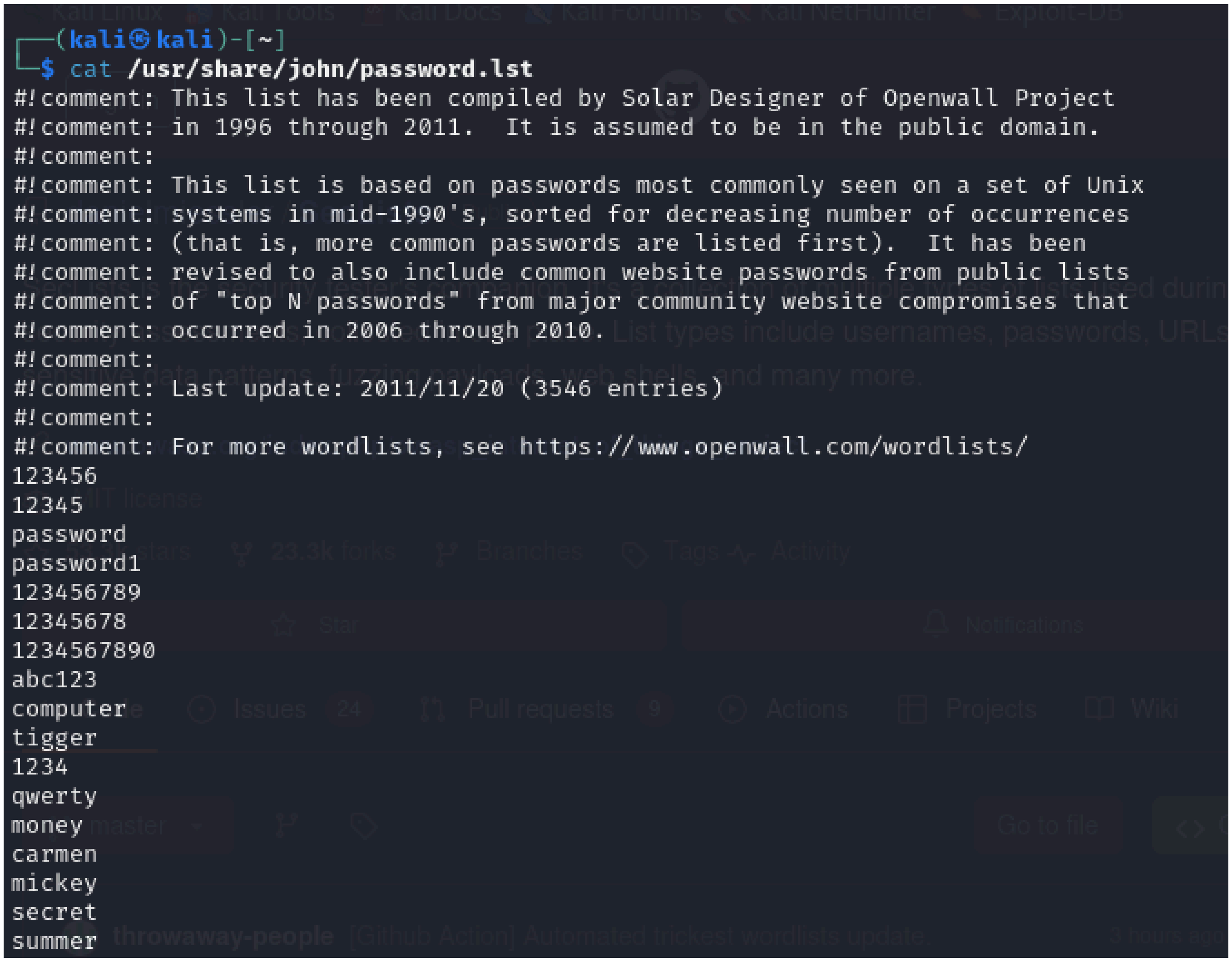
```
┌──(kali㉿kali)-[~]
└─$ ls -l /usr/share/john/password.lst; ls -l rockyou.txt
-rw-r--r-- 1 root root 26326 Nov  2  2021 /usr/share/john/password.lst
-rw-r--r-- 1 kali kali 139921507 Apr 11 23:55 rockyou.txt

┌──(kali㉿kali)-[~]
└─$ 
```

Step 2d:

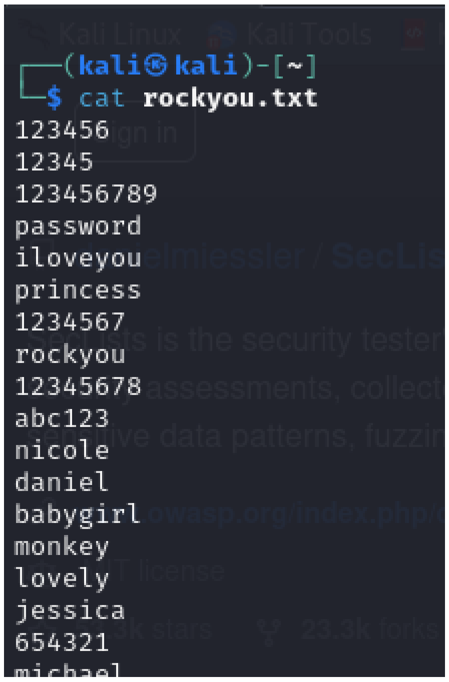


```
┌──(kali㉿kali)-[~]
└─$ sudo apt install leafpad -y
[sudo] password for kali:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libadwaita-1-0 libaio1 libappstream5 libatk-adaptor libboost-dev libboost1.83-dev
  libopenblas-dev libopenblas-pthread-dev libopenblas0 libpython3-all-dev libpython3.12
  libpython3.12-dev libstemmer0d libxmlb2 libxsimd-dev python3-all-dev python3-anyjson
  python3-beniget python3-gast python3-pyatspi python3-pypdf2 python3-pyrsistent
  python3-pythran python3.12-dev xtl-dev zenity zenity-common
Use 'sudo apt autoremove' to remove them.
Suggested packages:
  evince-gtk
The following NEW packages will be installed:
  leafpad
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 90.9 kB of archives.
After this operation, 465 kB of additional disk space will be used.
Get:1 http://kali.download/kali kali-rolling/main amd64 leafpad amd64 0.8.18.1-5 [90.9 kB]
Fetched 90.9 kB in 0s (676 kB/s)
Selecting previously unselected package leafpad.
(Reading database ... 406406 files and directories currently installed.)
Preparing to unpack .../leafpad_0.8.18.1-5_amd64.deb ...
Unpacking leafpad (0.8.18.1-5) ...
Setting up leafpad (0.8.18.1-5) ...
update-alternatives: using /usr/bin/leafpad to provide /usr/bin/gnome-text-editor (gnome-te
xt-editor) in auto mode
Processing triggers for desktop-file-utils (0.27-1) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for man-db (2.12.0-3) ...
Processing triggers for kali-menu (2023.4.7) ...
```

Step 2e:

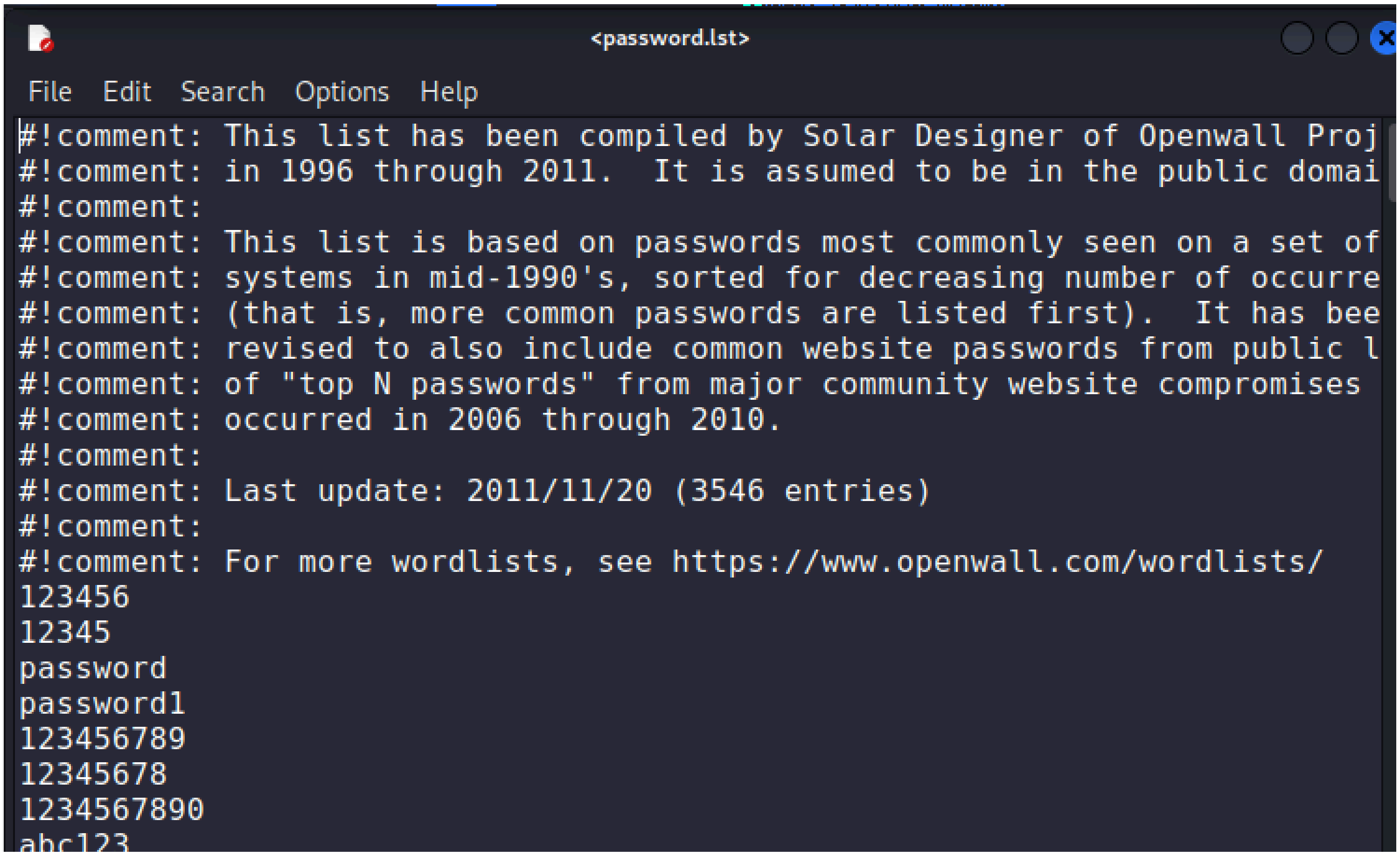


```
┌──(kali㉿kali)-[~]
└─$ cat /usr/share/john/password.lst
#!comment: This list has been compiled by Solar Designer of Openwall Project
#!comment: in 1996 through 2011.  It is assumed to be in the public domain.
#!comment:
#!comment: This list is based on passwords most commonly seen on a set of Unix
#!comment: systems in mid-1990's, sorted for decreasing number of occurrences
#!comment: (that is, more common passwords are listed first).  It has been
#!comment: revised to also include common website passwords from public lists
#!comment: of "top N passwords" from major community website compromises that
#!comment: occurred in 2006 through 2010.
#!comment:
#!comment: Last update: 2011/11/20 (3546 entries)
#!comment:
#!comment: For more wordlists, see https://www.openwall.com/wordlists/
123456
12345
password
password1
123456789
12345678
1234567890
abc123
computer
tigger
1234
qwerty
money
carmen
mickey
secret
summer
```

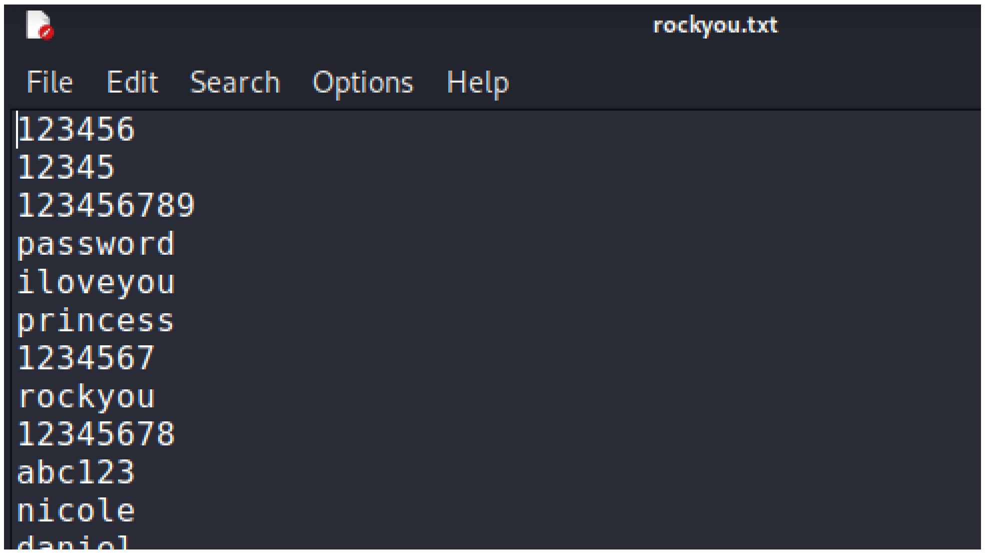


```
┌──(kali㉿kali)-[~]
└─$ cat rockyou.txt
123456
12345
123456789
password
iloveyou
princess
1234567
rockyou
12345678
abc123
nicole
daniel
babygirl
monkey
lovely
jessica
654321
michael
```

Step 2f:

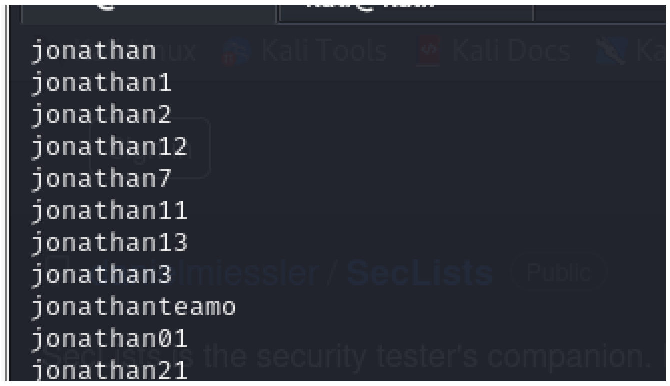


```
#!comment: This list has been compiled by Solar Designer of Openwall Proj
#!comment: in 1996 through 2011.  It is assumed to be in the public domai
#!comment:
#!comment: This list is based on passwords most commonly seen on a set of
#!comment: systems in mid-1990's, sorted for decreasing number of occurre
#!comment: (that is, more common passwords are listed first).  It has bee
#!comment: revised to also include common website passwords from public l
#!comment: of "top N passwords" from major community website compromises
#!comment: occurred in 2006 through 2010.
#!comment:
#!comment: Last update: 2011/11/20 (3546 entries)
#!comment:
#!comment: For more wordlists, see https://www.openwall.com/wordlists/
123456
12345
password
password1
123456789
12345678
1234567890
abc123
```



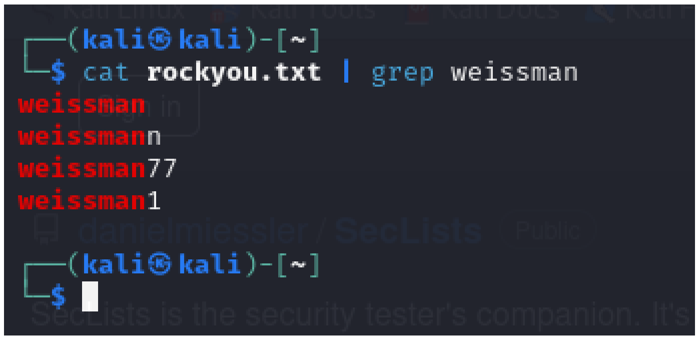
```
123456
12345
123456789
password
iloveyou
princess
1234567
rockyou
12345678
abc123
nicole
daniel
```

Step 2g:



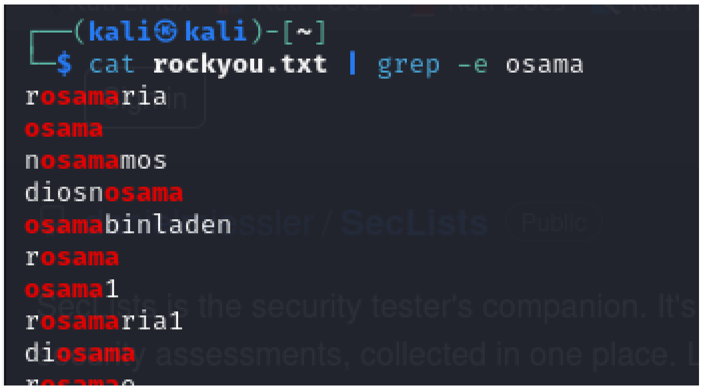
```
jonathan
jonathan1
jonathan2
jonathan12
jonathan7
jonathan11
jonathan13
jonathan3
jonathanteamo
jonathan01
jonathan21
```

Step 2h:

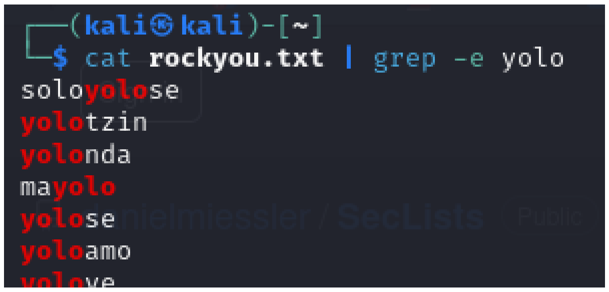


```
┌──(kali㉿kali)-[~]
└─$ cat rockyou.txt | grep weissman
weissman
weissmann
weissman77
weissman1

┌──(kali㉿kali)-[~]
└─$ 
```

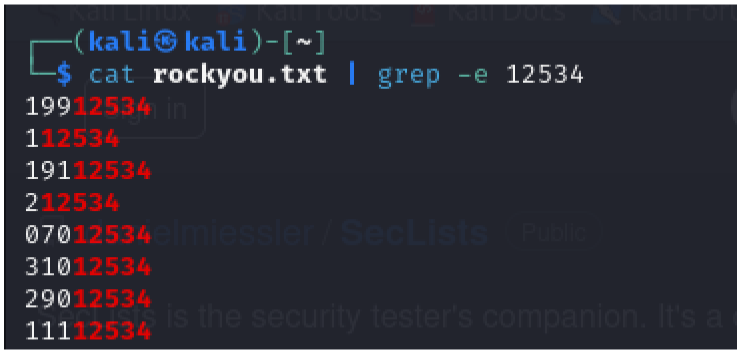
Step 2i:



```
┌──(kali㉿kali)-[~]
└─$ cat rockyou.txt | grep -e osama
rosamaria
osama
nosamamos
diosnosama
osamabinladen
rosama
osama1
rosamaria1
diosama
```

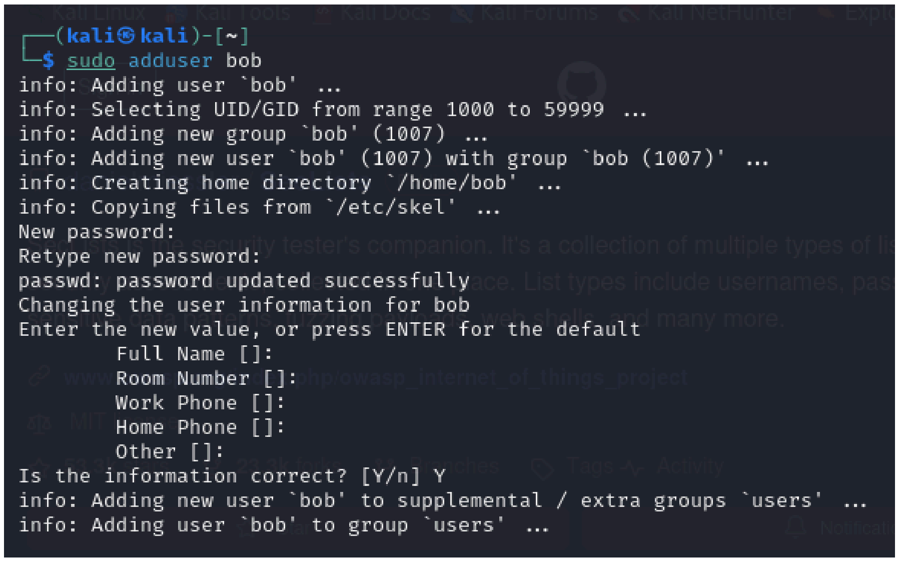


```
┌──(kali㉿kali)-[~]
└─$ cat rockyou.txt | grep -e yolo
soloyolose
yolotzin
yolonda
mayolo
yolose
yoloamo
yolove
```

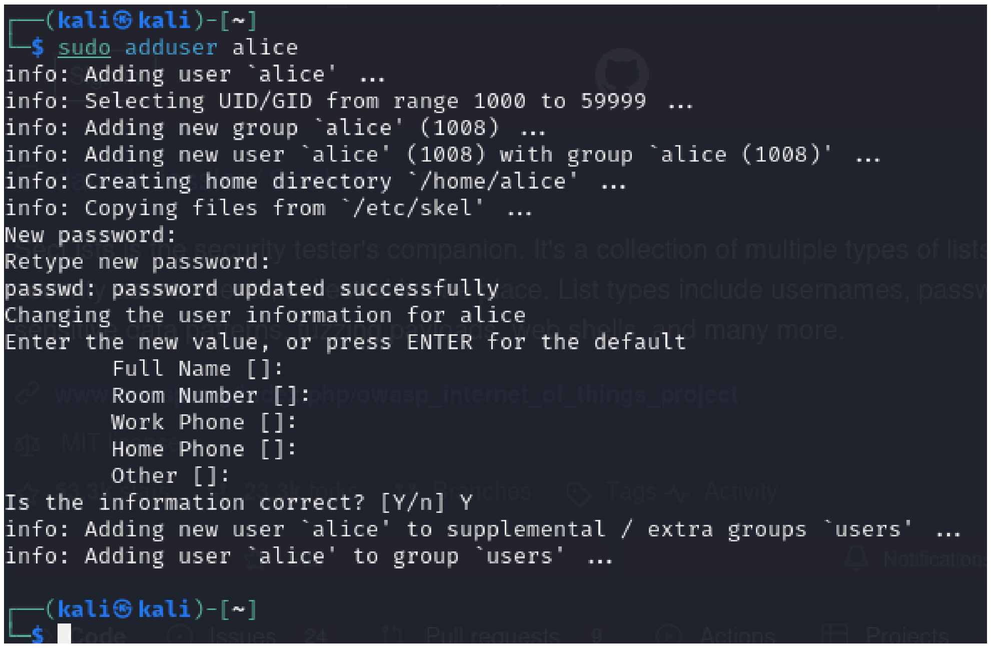


```
┌──(kali㉿kali)-[~]
└─$ cat rockyou.txt | grep -e 12534
19912534
112534
19112534
212534
07012534
31012534
29012534
11112534
```

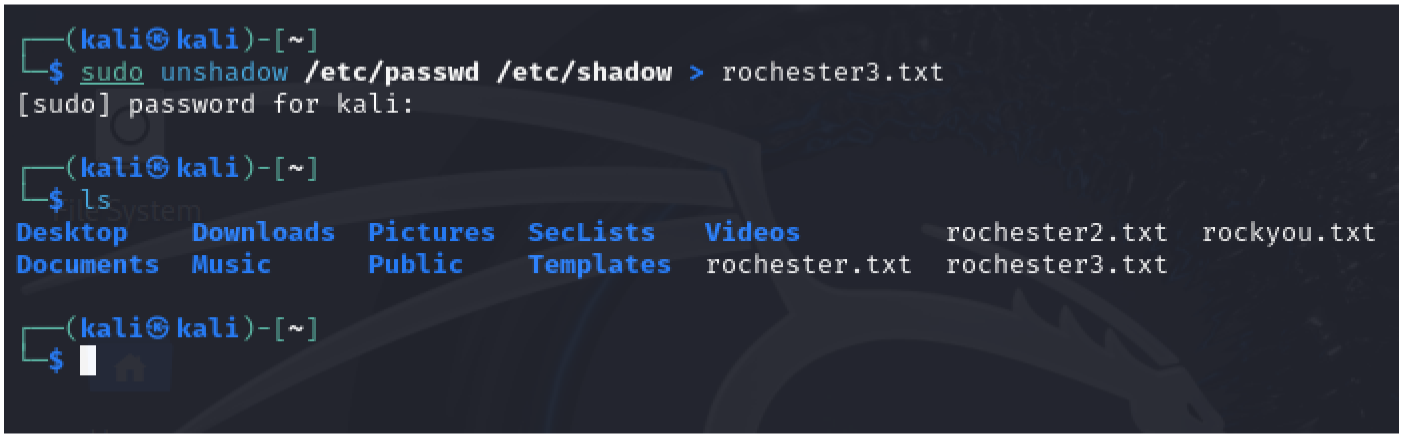
Step 2j:



```
┌──(kali㉿kali)-[~]
└─$ sudo adduser bob
info: Adding user `bob' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `bob' (1007) ...
info: Adding new user `bob' (1007) with group `bob (1007)' ...
info: Creating home directory `/home/bob' ...
info: Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for bob
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
info: Adding new user `bob' to supplemental / extra groups `users' ...
info: Adding user `bob' to group `users' ...
```

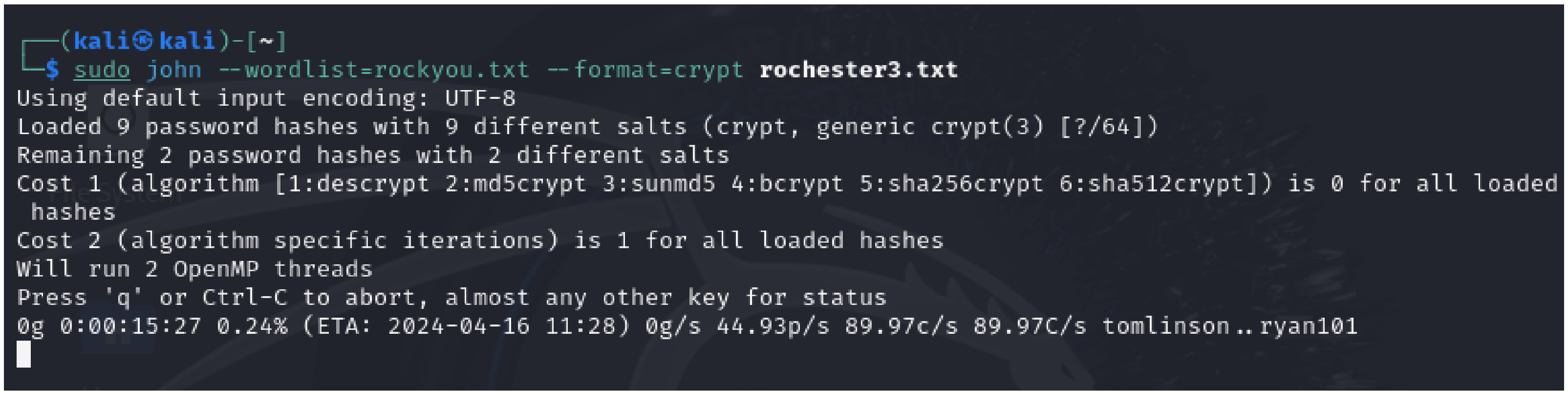


```
┌──(kali㉿kali)-[~]
└─$ sudo adduser alice
info: Adding user `alice' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `alice' (1008) ...
info: Adding new user `alice' (1008) with group `alice (1008)' ...
info: Creating home directory `/home/alice' ...
info: Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for alice
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
info: Adding new user `alice' to supplemental / extra groups `users' ...
info: Adding user `alice' to group `users' ...

┌──(kali㉿kali)-[~]
└─$ 
```



```
┌──(kali㉿kali)-[~]
└─$ sudo unshadow /etc/passwd /etc/shadow > rochester3.txt
[sudo] password for kali:

┌──(kali㉿kali)-[~]
└─$ ls
Desktop    Downloads  Pictures  SecLists   Videos         rochester2.txt  rockyou.txt
Documents  Music      Public    Templates  rochester.txt  rochester3.txt

┌──(kali㉿kali)-[~]
└─$ 
```

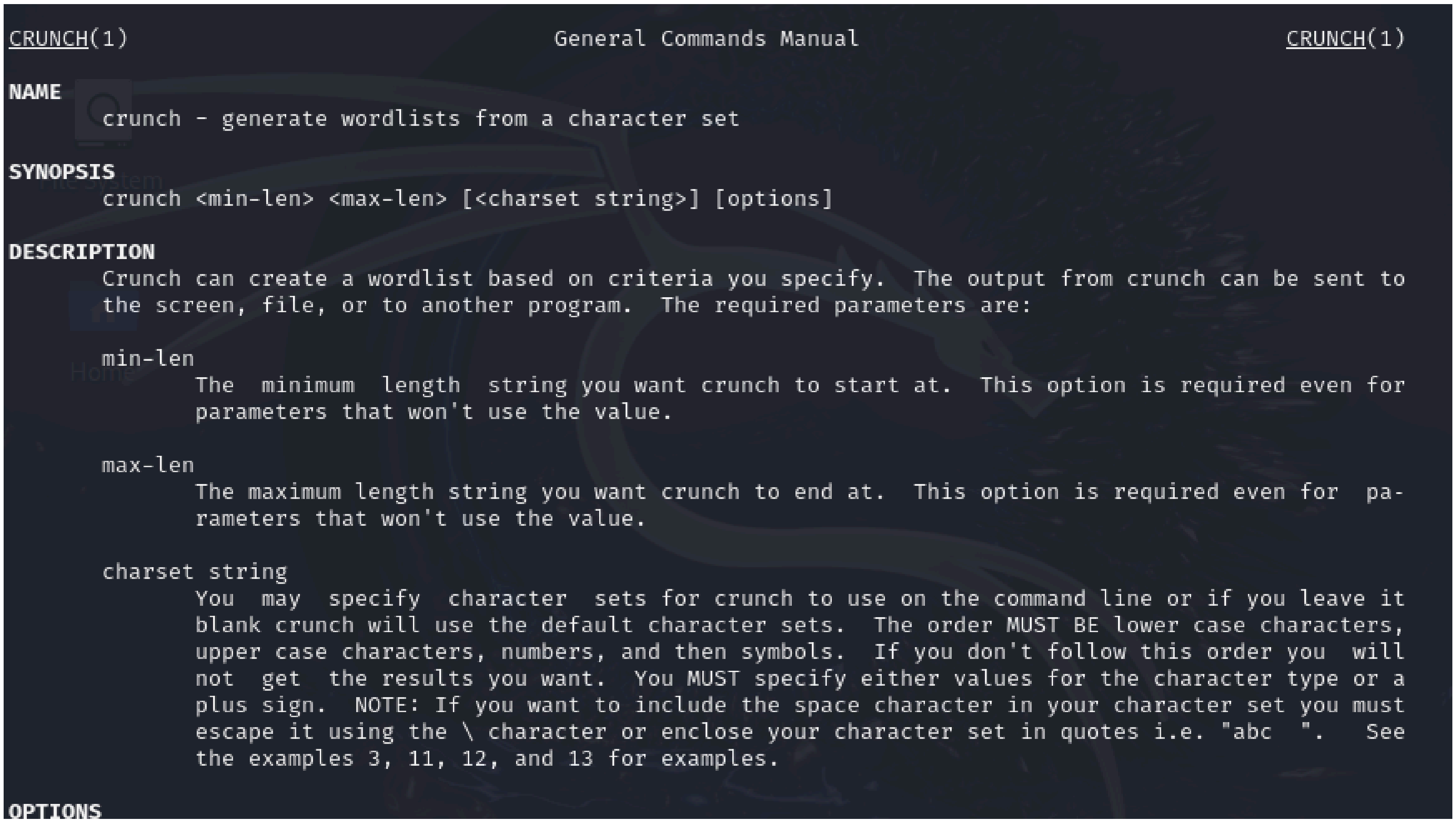
Step 2k: It was not able to crack any of the new users, or it would've taken along time.



```
┌──(kali㉿kali)-[~]
└─$ sudo john --wordlist=rockyou.txt --format=crypt rochester3.txt
Using default input encoding: UTF-8
Loaded 9 password hashes with 9 different salts (crypt, generic crypt(3) [?/64])
Remaining 2 password hashes with 2 different salts
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
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:15:27 0.24% (ETA: 2024-04-16 11:28) 0g/s 44.93p/s 89.97c/s 89.97C/s tomlinson..ryan101
```

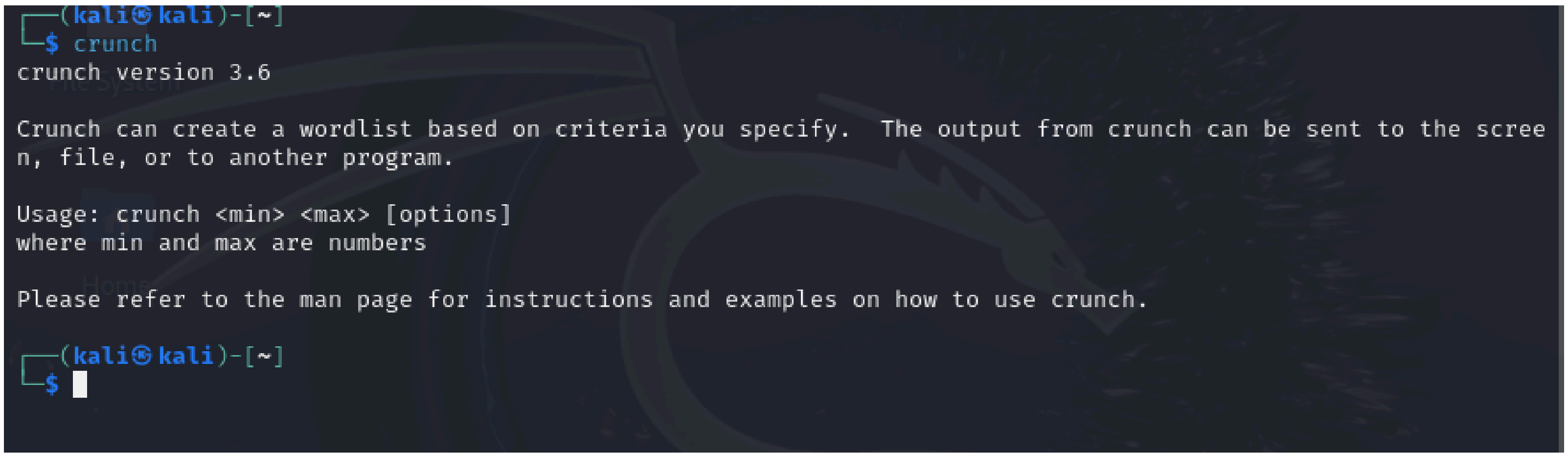
Lab Exercise 11.02: Brute Force Attacks on Linux Passwords with crunch and John the Ripper

Step 1a:



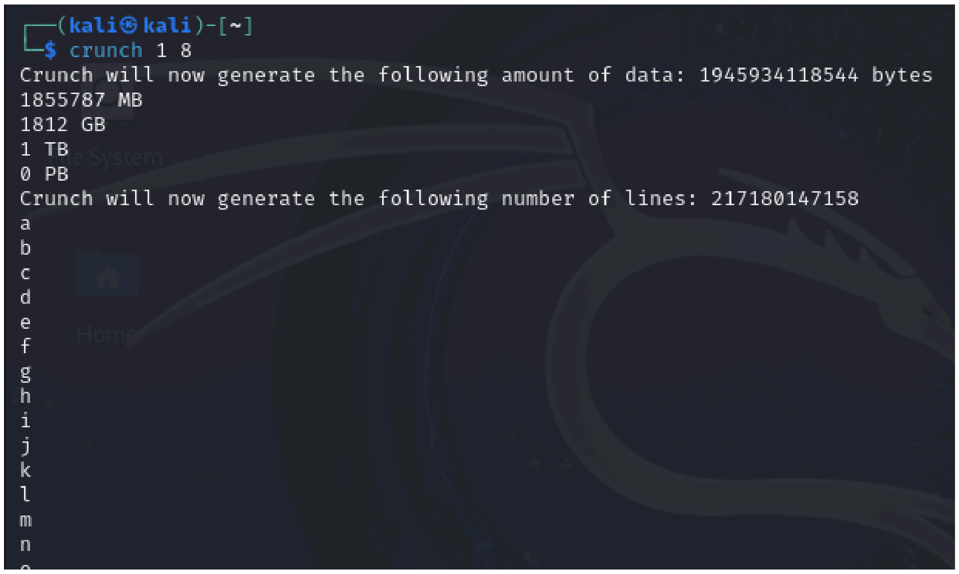
```
CRUNCH(1)                         General Commands Manual                         CRUNCH(1)

NAME
       crunch - generate wordlists from a character set

SYNOPSIS
       crunch <min-len> <max-len> [<charset string>] [options]

DESCRIPTION
       Crunch can create a wordlist based on criteria you specify.  The output from crunch can be sent to
       the screen, file, or to another program.  The required parameters are:

       min-len
              The  minimum  length  string you want crunch to start at.  This option is required even for
              parameters that won't use the value.

       max-len
              The maximum length string you want crunch to end at.  This option is required even for  pa-
              rameters that won't use the value.

       charset string
              You  may  specify  character  sets for crunch to use on the command line or if you leave it
              blank crunch will use the default character sets.  The order MUST BE lower case characters,
              upper case characters, numbers, and then symbols.  If you don't follow this order you  will
              not  get  the results you want.  You MUST specify either values for the character type or a
              plus sign.  NOTE: If you want to include the space character in your character set you must
              escape it using the \ character or enclose your character set in quotes i.e. "abc  ".   See
              the examples 3, 11, 12, and 13 for examples.

OPTIONS
```

Step 1b:



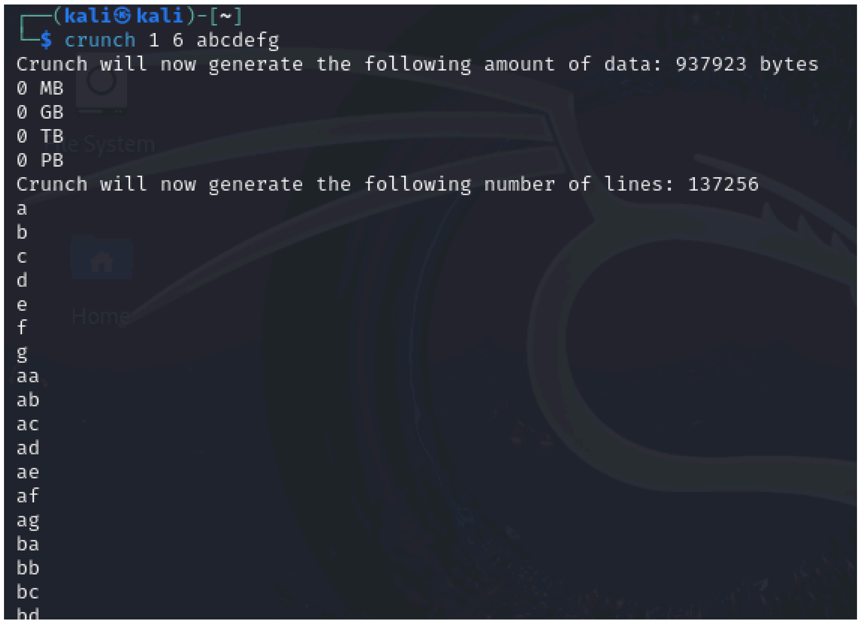
```
┌──(kali㉿kali)-[~]
└─$ crunch
crunch version 3.6

Crunch can create a wordlist based on criteria you specify.  The output from crunch can be sent to the scree
n, file, or to another program.

Usage: crunch <min> <max> [options]
where min and max are numbers

Please refer to the man page for instructions and examples on how to use crunch.

┌──(kali㉿kali)-[~]
└─$ 
```

Step 1c:



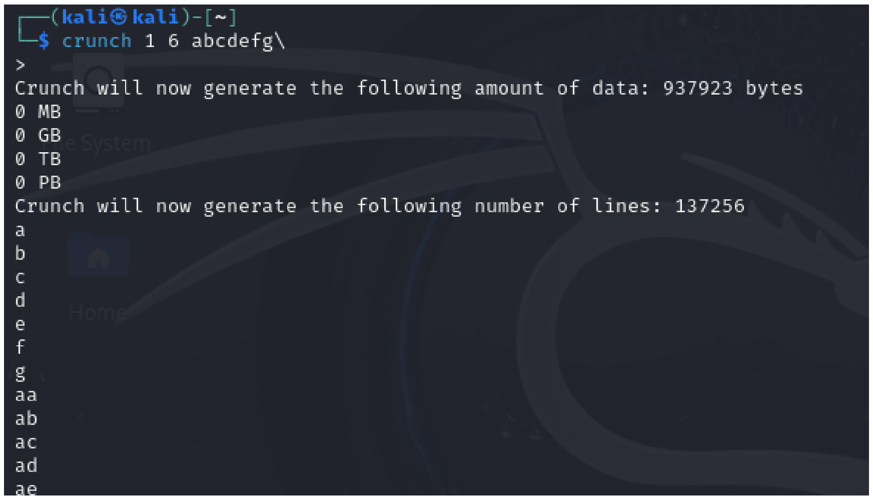
```
┌──(kali㉿kali)-[~]
└─$ crunch 1 8
Crunch will now generate the following amount of data: 1945934118544 bytes
1855787 MB
1812 GB
1 TB
0 PB
Crunch will now generate the following number of lines: 217180147158
a
b
c
d
e
f
g
h
i
j
k
l
m
n
```

Step 1d:



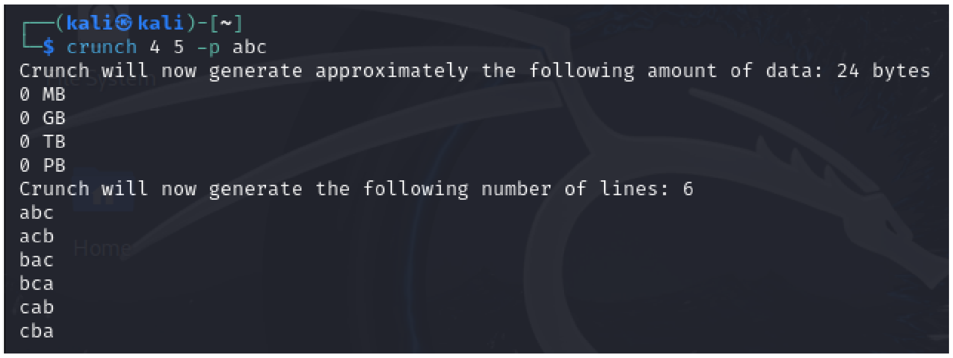
```
┌──(kali㉿kali)-[~]
└─$ crunch 1 6 abcdefg
Crunch will now generate the following amount of data: 937923 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 137256
a
b
c
d
e
f
g
aa
ab
ac
ad
ae
af
ag
ba
bb
bc
bd
```

Step 1e:



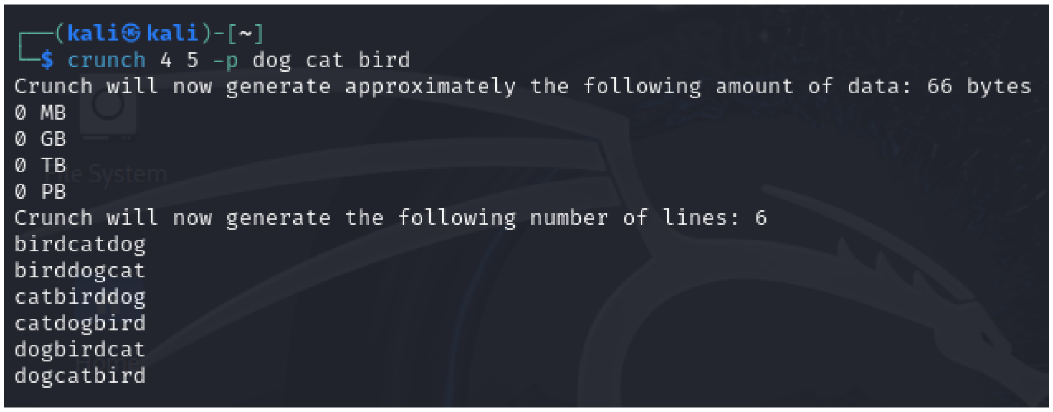
```
┌──(kali㉿kali)-[~]
└─$ crunch 1 6 abcdefg\
>
Crunch will now generate the following amount of data: 937923 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 137256
a
b
c
d
e
f
g
aa
ab
ac
ad
ae
```

Step 1f:



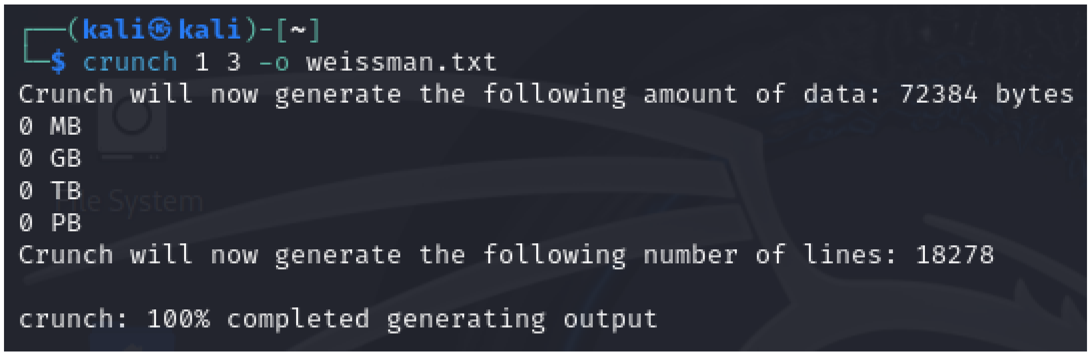
```
┌──(kali㉿kali)-[~]
└─$ crunch 4 5 -p abc
Crunch will now generate approximately the following amount of data: 24 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 6
abc
acb
bac
bca
cab
cba
```

Step 1g:



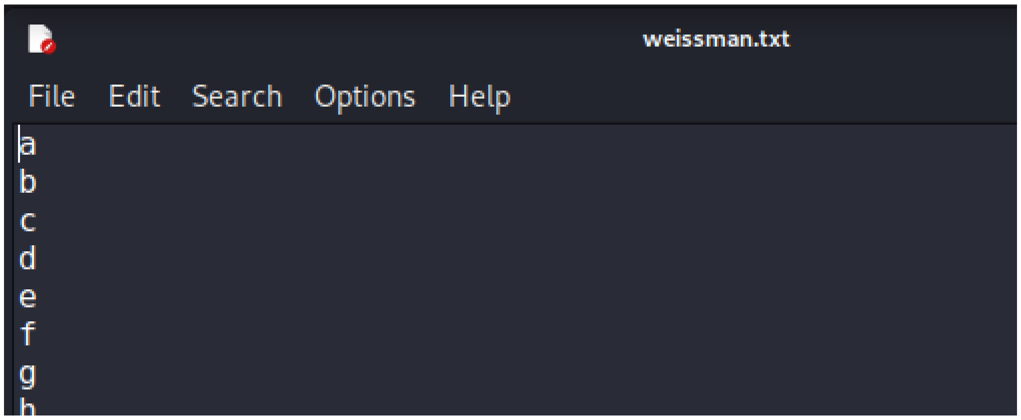
```
┌──(kali㉿kali)-[~]
└─$ crunch 4 5 -p dog cat bird
Crunch will now generate approximately the following amount of data: 66 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 6
birdcatdog
birddogcat
catbirddog
catdogbird
dogbirdcat
dogcatbird
```

Step 2a:



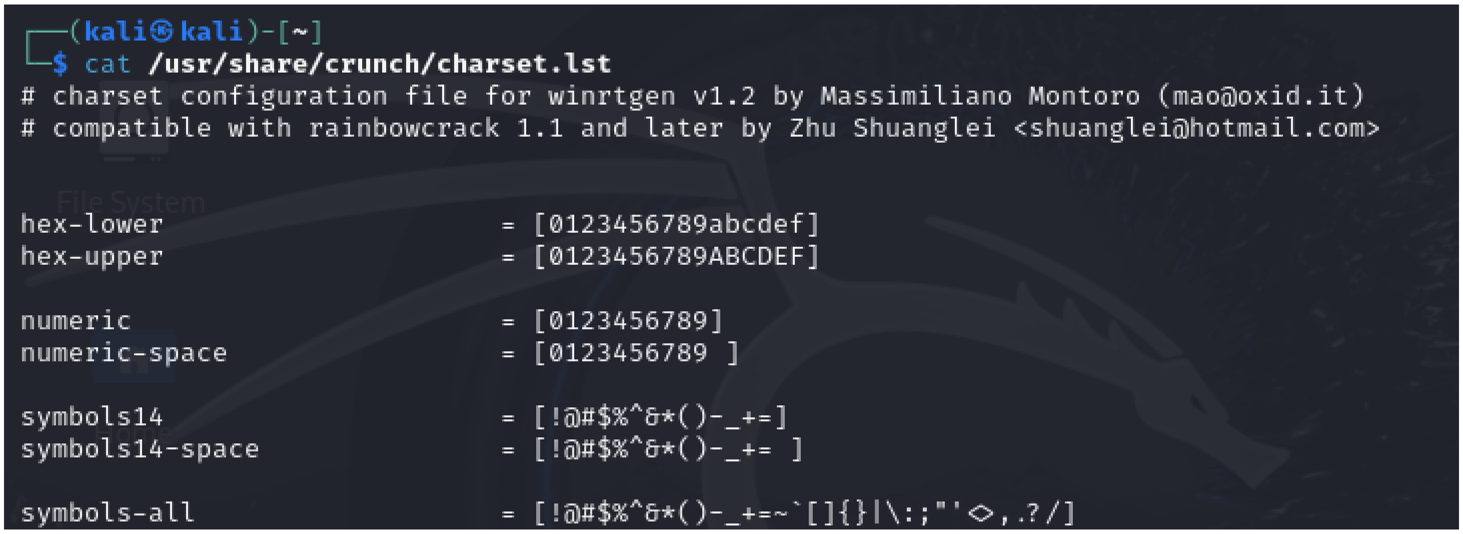
```
┌──(kali㉿kali)-[~]
└─$ crunch 1 3 -o weissman.txt
Crunch will now generate the following amount of data: 72384 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 18278

crunch: 100% completed generating output
```

Step 2b:



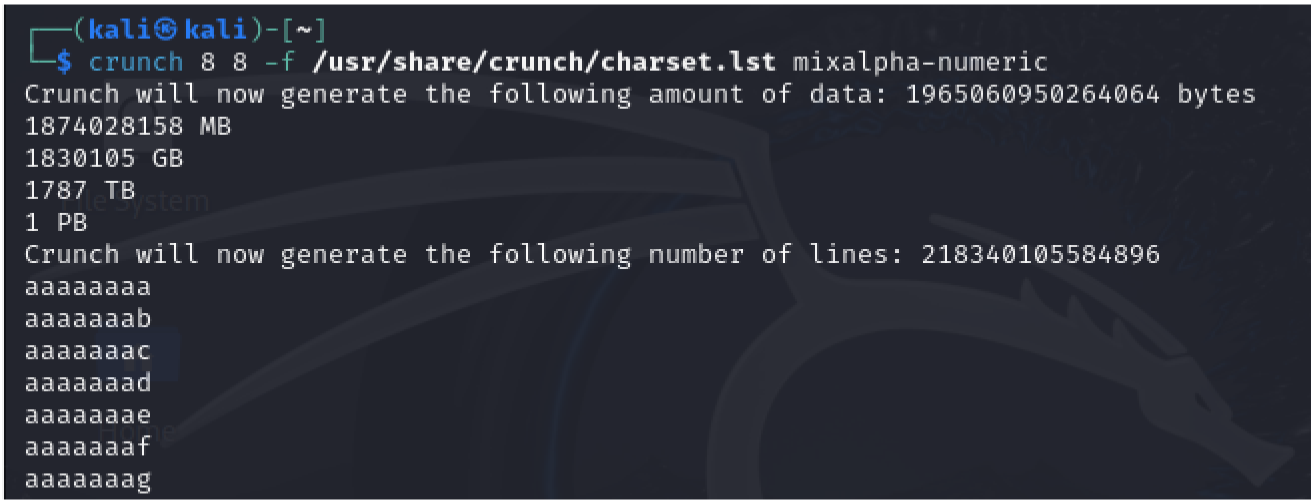
```
a
b
c
d
e
f
g
h
```

Step 2c:



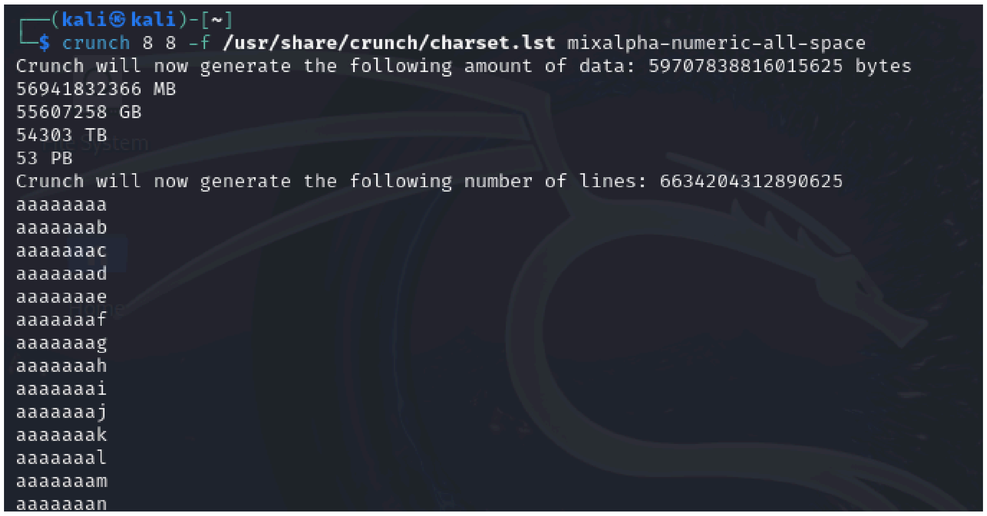
```
┌──(kali㉿kali)-[~]
└─$ cat /usr/share/crunch/charset.lst
# charset configuration file for winrtgen v1.2 by Massimiliano Montoro (mao@oxid.it)
# compatible with rainbowcrack 1.1 and later by Zhu Shuanglei <shuanglei@hotmail.com>


hex-lower                      = [0123456789abcdef]
hex-upper                      = [0123456789ABCDEF]

numeric                        = [0123456789]
numeric-space                  = [0123456789 ]

symbols14                      = [!@#$%^&*()-_+=]
symbols14-space                = [!@#$%^&*()-_+= ]

symbols-all                    = [!@#$%^&*()-_+=~`[]{}|\:;"'<>,.?/]
```

Step 2d:



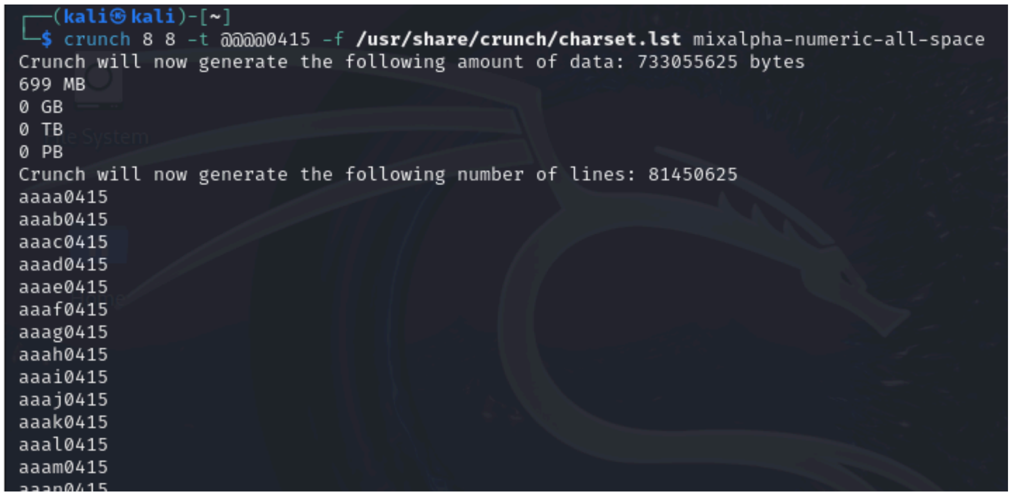
```
┌──(kali㉿kali)-[~]
└─$ crunch 8 8 -f /usr/share/crunch/charset.lst mixalpha-numeric
Crunch will now generate the following amount of data: 1965060950264064 bytes
1874028158 MB
1830105 GB
1787 TB
1 PB
Crunch will now generate the following number of lines: 218340105584896
aaaaaaaa
aaaaaaab
aaaaaaac
aaaaaaad
aaaaaaae
aaaaaaaf
aaaaaaag
```

Step 2e:



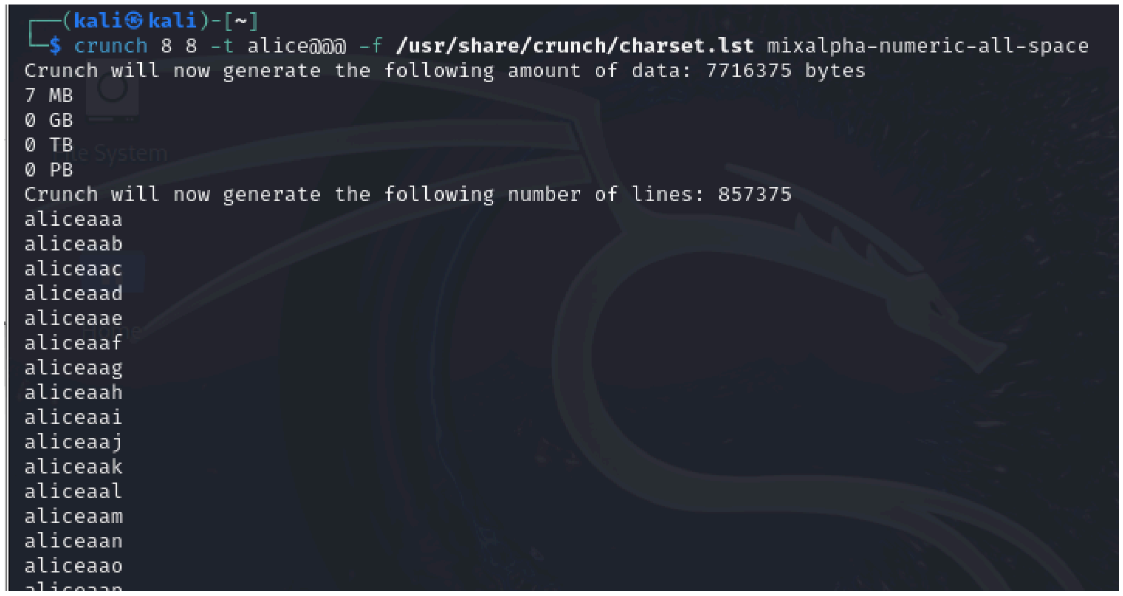
```
┌──(kali㉿kali)-[~]
└─$ crunch 8 8 -f /usr/share/crunch/charset.lst mixalpha-numeric-all-space
Crunch will now generate the following amount of data: 59707838816015625 bytes
56941832366 MB
55607258 GB
54303 TB
53 PB
Crunch will now generate the following number of lines: 6634204312890625
aaaaaaaa
aaaaaaab
aaaaaaac
aaaaaaad
aaaaaaae
aaaaaaaf
aaaaaaag
aaaaaaah
aaaaaaai
aaaaaaaj
aaaaaaak
aaaaaaal
aaaaaaam
aaaaaaan
```

Step 2f:



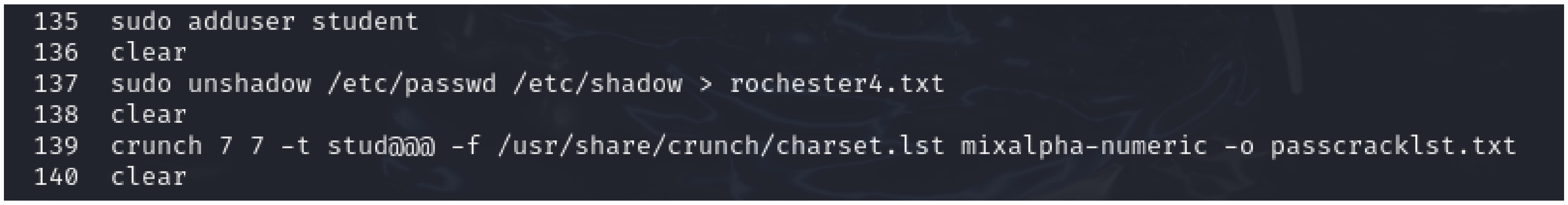
```
┌──(kali㉿kali)-[~]
└─$ crunch 8 8 -t @@@@0415 -f /usr/share/crunch/charset.lst mixalpha-numeric-all-space
Crunch will now generate the following amount of data: 733055625 bytes
699 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 81450625
aaaa0415
aaab0415
aaac0415
aaad0415
aaae0415
aaaf0415
aaag0415
aaah0415
aaai0415
aaaj0415
aaak0415
aaal0415
aaam0415
```

Step 2g:



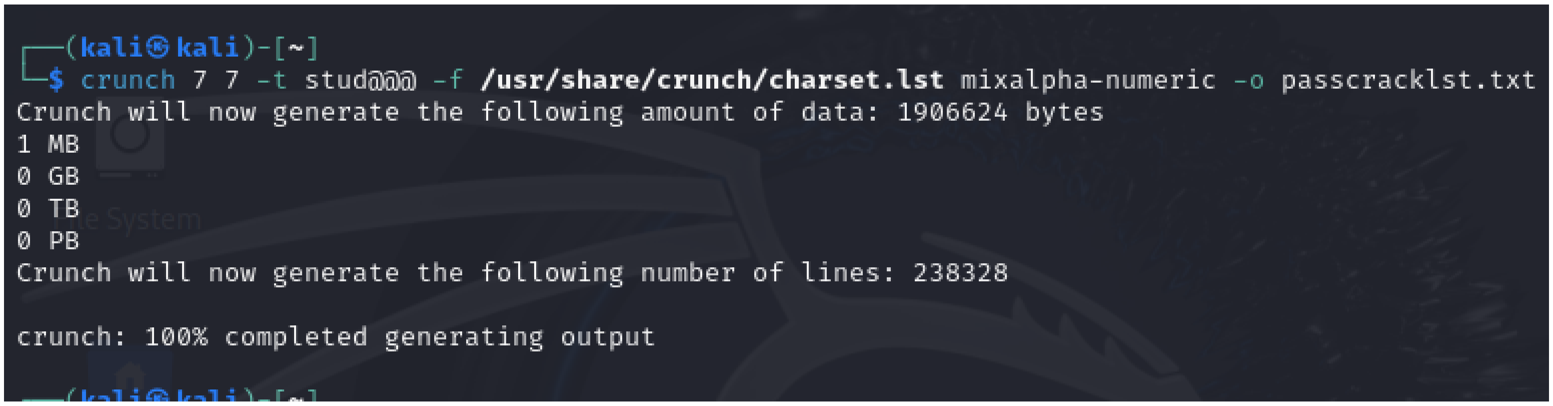
```
┌──(kali㉿kali)-[~]
└─$ crunch 8 8 -t alice@@@ -f /usr/share/crunch/charset.lst mixalpha-numeric-all-space
Crunch will now generate the following amount of data: 7716375 bytes
7 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 857375
aliceaaa
aliceaab
aliceaac
aliceaad
aliceaae
aliceaaf
aliceaag
aliceaah
aliceaai
aliceaaj
aliceaak
aliceaal
aliceaam
aliceaan
aliceaao
```

Step 3a - b: (Using history command)



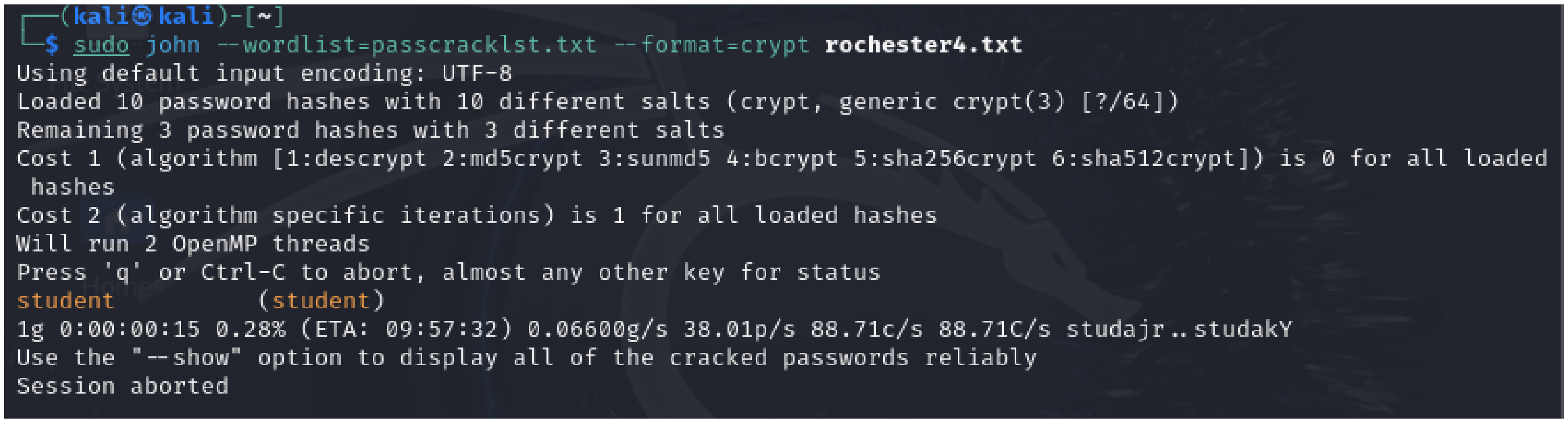
```
 135  sudo adduser student
 136  clear
 137  sudo unshadow /etc/passwd /etc/shadow > rochester4.txt
 138  clear
 139  crunch 7 7 -t stud@@@ -f /usr/share/crunch/charset.lst mixalpha-numeric -o passcracklst.txt
 140  clear
```

Step 3c:



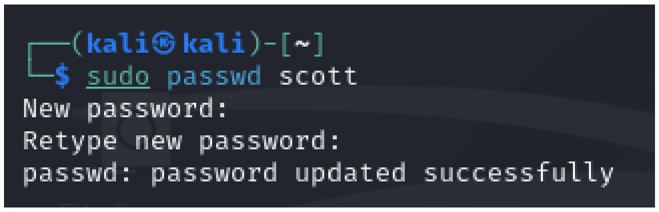
```
┌──(kali㉿kali)-[~]
└─$ crunch 7 7 -t stud@@@ -f /usr/share/crunch/charset.lst mixalpha-numeric -o passcracklst.txt
Crunch will now generate the following amount of data: 1906624 bytes
1 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 238328

crunch: 100% completed generating output

┌──(kali㉿kali)-[~]
```

Step 3d:



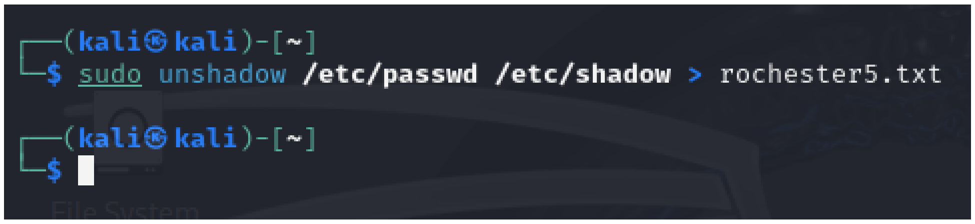
```
┌──(kali㉿kali)-[~]
└─$ sudo john --wordlist=passcracklst.txt --format=crypt rochester4.txt
Using default input encoding: UTF-8
Loaded 10 password hashes with 10 different salts (crypt, generic crypt(3) [?/64])
Remaining 3 password hashes with 3 different salts
Cost 1 (algorithm [1:descrypt 2:md5crypt 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
Press 'q' or Ctrl-C to abort, almost any other key for status
student          (student)
1g 0:00:00:15 0.28% (ETA: 09:57:32) 0.06600g/s 38.01p/s 88.71c/s 88.71C/s studajr..studakY
Use the "--show" option to display all of the cracked passwords reliably
Session aborted
```

Step 4a:



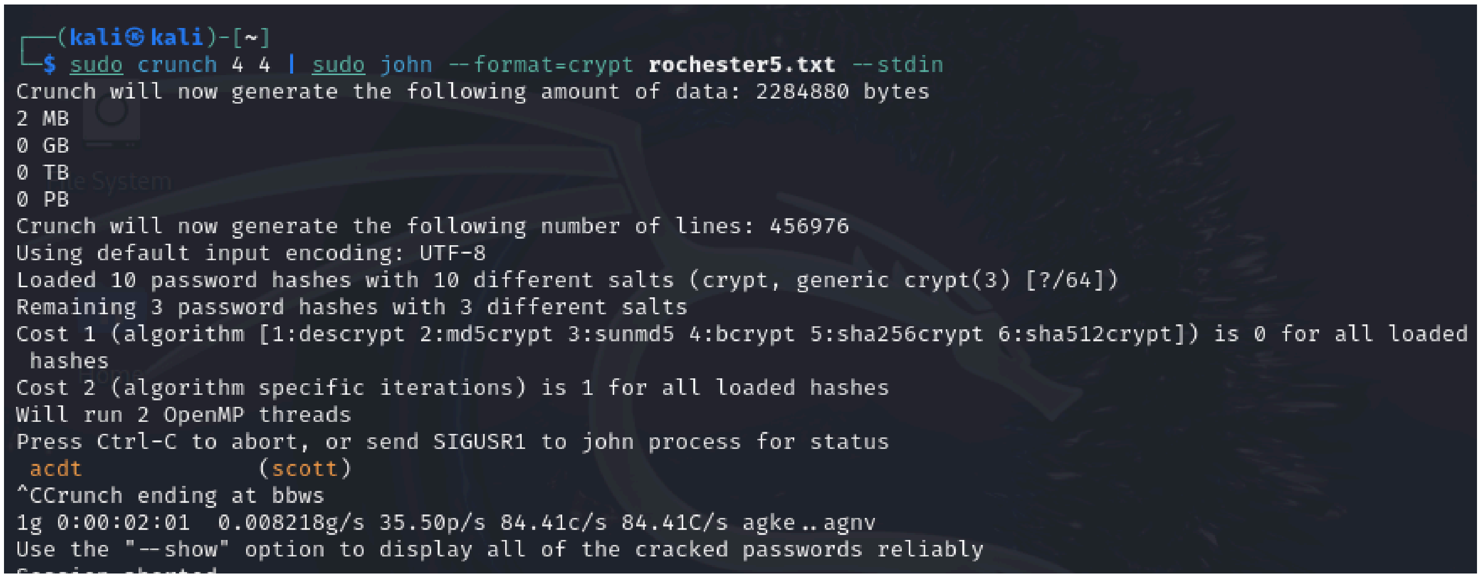
```
┌──(kali㉿kali)-[~]
└─$ sudo passwd scott
New password:
Retype new password:
passwd: password updated successfully
```

Step 4b:



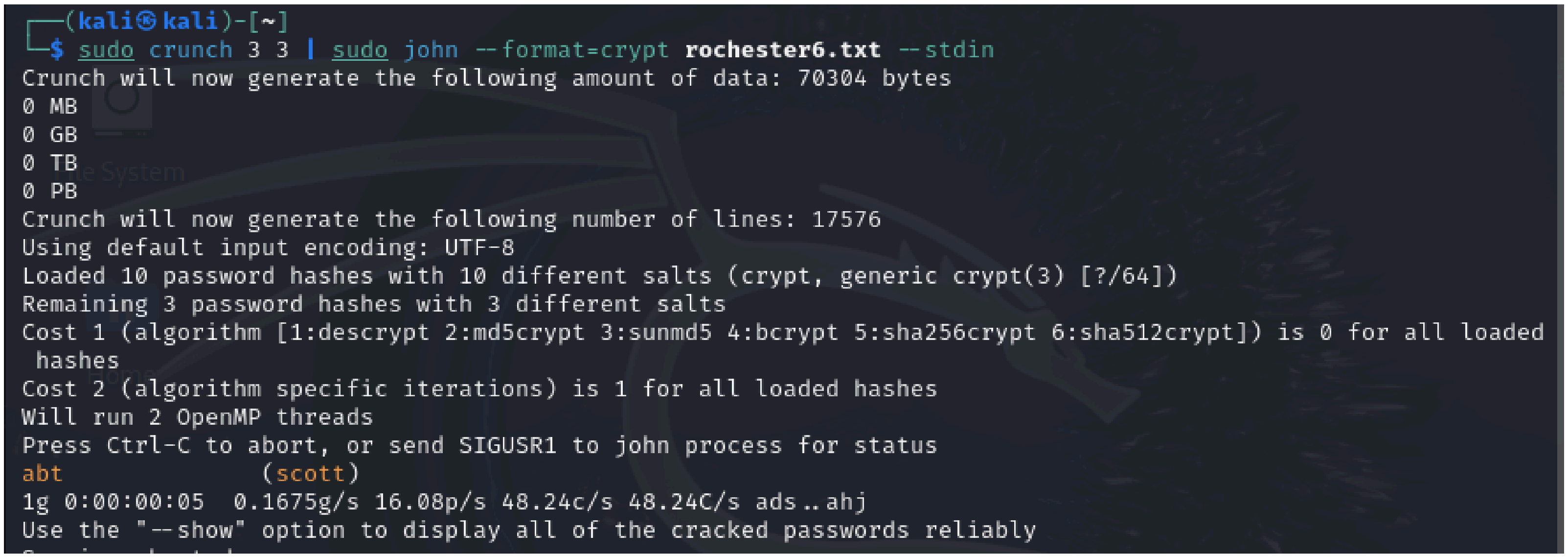
```
┌──(kali㉿kali)-[~]
└─$ sudo unshadow /etc/passwd /etc/shadow > rochester5.txt

┌──(kali㉿kali)-[~]
└─$ 
```

Step 4c:



```
┌──(kali㉿kali)-[~]
└─$ sudo crunch 4 4 | sudo john --format=crypt rochester5.txt --stdin
Crunch will now generate the following amount of data: 2284880 bytes
2 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 456976
Using default input encoding: UTF-8
Loaded 10 password hashes with 10 different salts (crypt, generic crypt(3) [?/64])
Remaining 3 password hashes with 3 different salts
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
Press Ctrl-C to abort, or send SIGUSR1 to john process for status
 acdt            (scott)
^CCrunch ending at bbws
1g 0:00:02:01  0.008218g/s 35.50p/s 84.41c/s 84.41C/s agke..agnv
Use the "--show" option to display all of the cracked passwords reliably
Session aborted
```

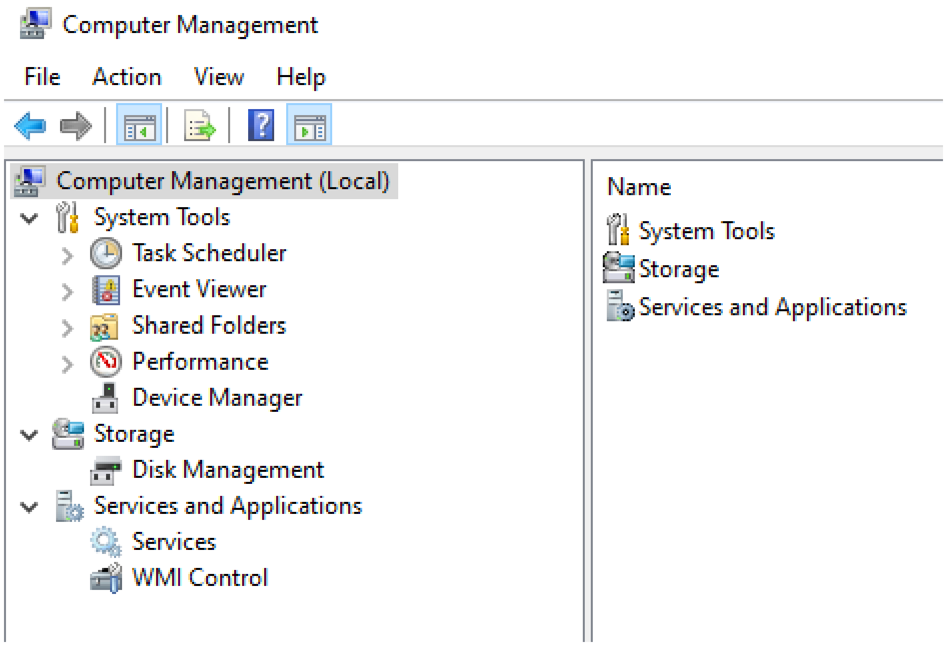
Step 4d:

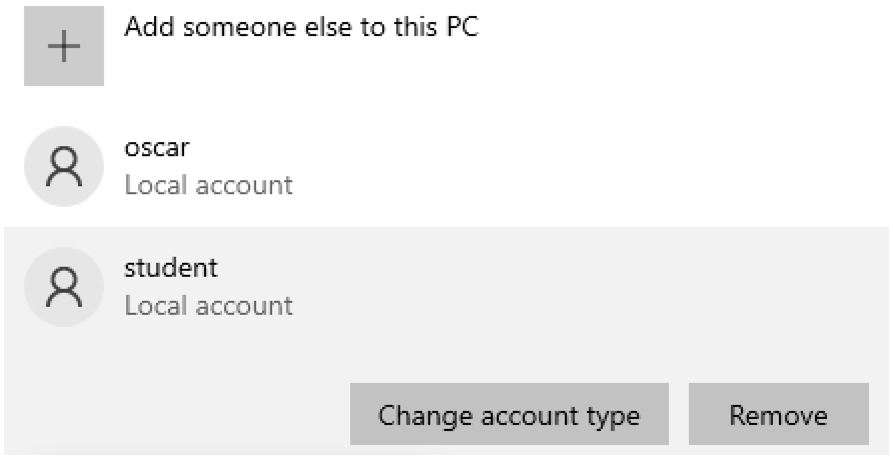


```
┌──(kali㉿kali)-[~]
└─$ sudo crunch 3 3 | sudo john --format=crypt rochester6.txt --stdin
Crunch will now generate the following amount of data: 70304 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 17576
Using default input encoding: UTF-8
Loaded 10 password hashes with 10 different salts (crypt, generic crypt(3) [?/64])
Remaining 3 password hashes with 3 different salts
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
Press Ctrl-C to abort, or send SIGUSR1 to john process for status
abt              (scott)
1g 0:00:00:05  0.1675g/s 16.08p/s 48.24c/s 48.24C/s ads..ahj
Use the "--show" option to display all of the cracked passwords reliably
```

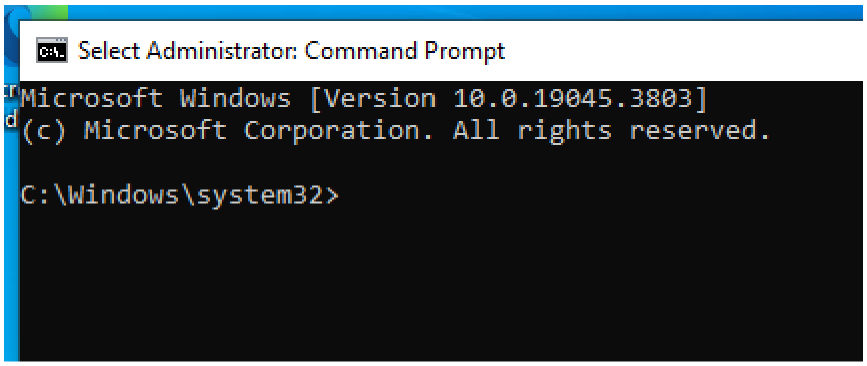
Lab Exercise 11.03: Dictionary Attacks and Brute Force Attacks on Windows Passwords with Mimikatz, crunch, and John the Ripper

Step 2f - g:
Local user and groups is not available in computer management, and had to do it through another account setting



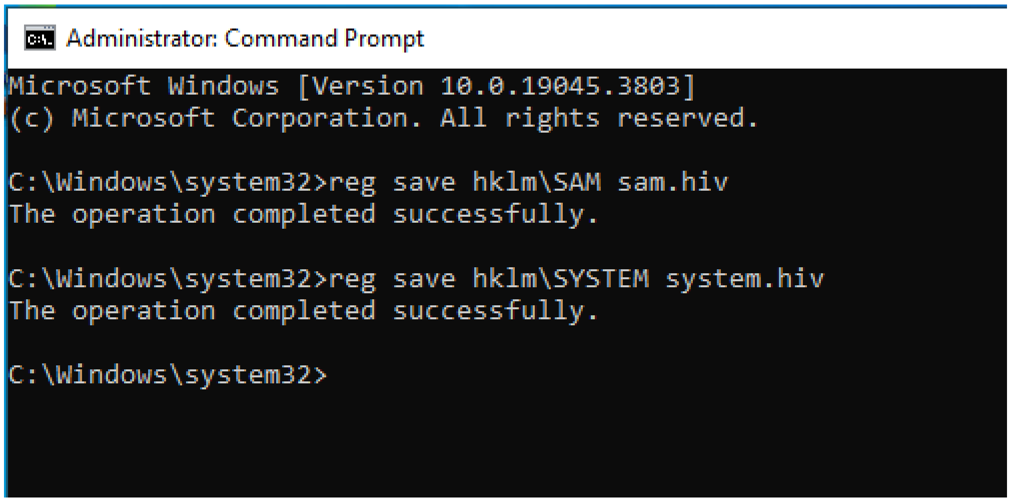


Step 3a:



```
Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

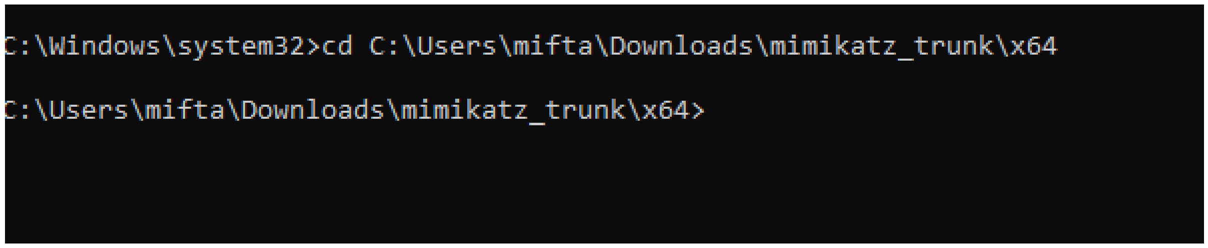
Step 3b:



Administrator: Command Prompt

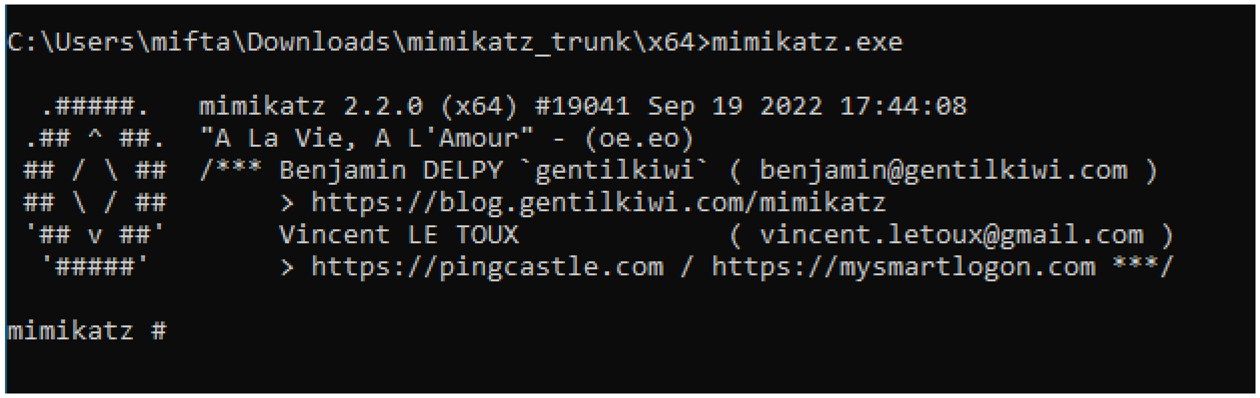
```
Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>reg save hklm\SAM sam.hiv
The operation completed successfully.

C:\Windows\system32>reg save hklm\SYSTEM system.hiv
The operation completed successfully.

C:\Windows\system32>
```

Step 3c:



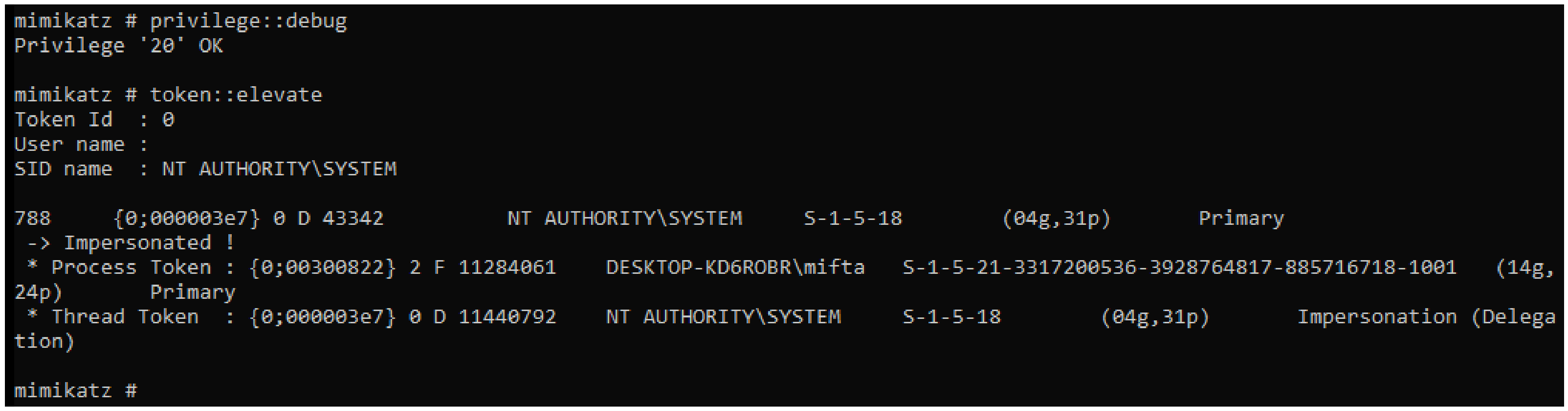
```
C:\Windows\system32>cd C:\Users\mifta\Downloads\mimikatz_trunk\x64

C:\Users\mifta\Downloads\mimikatz_trunk\x64>
```

Step 3d:



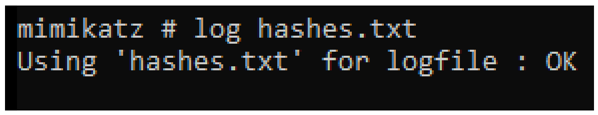
```
C:\Users\mifta\Downloads\mimikatz_trunk\x64>mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz #
```

Step 3e:



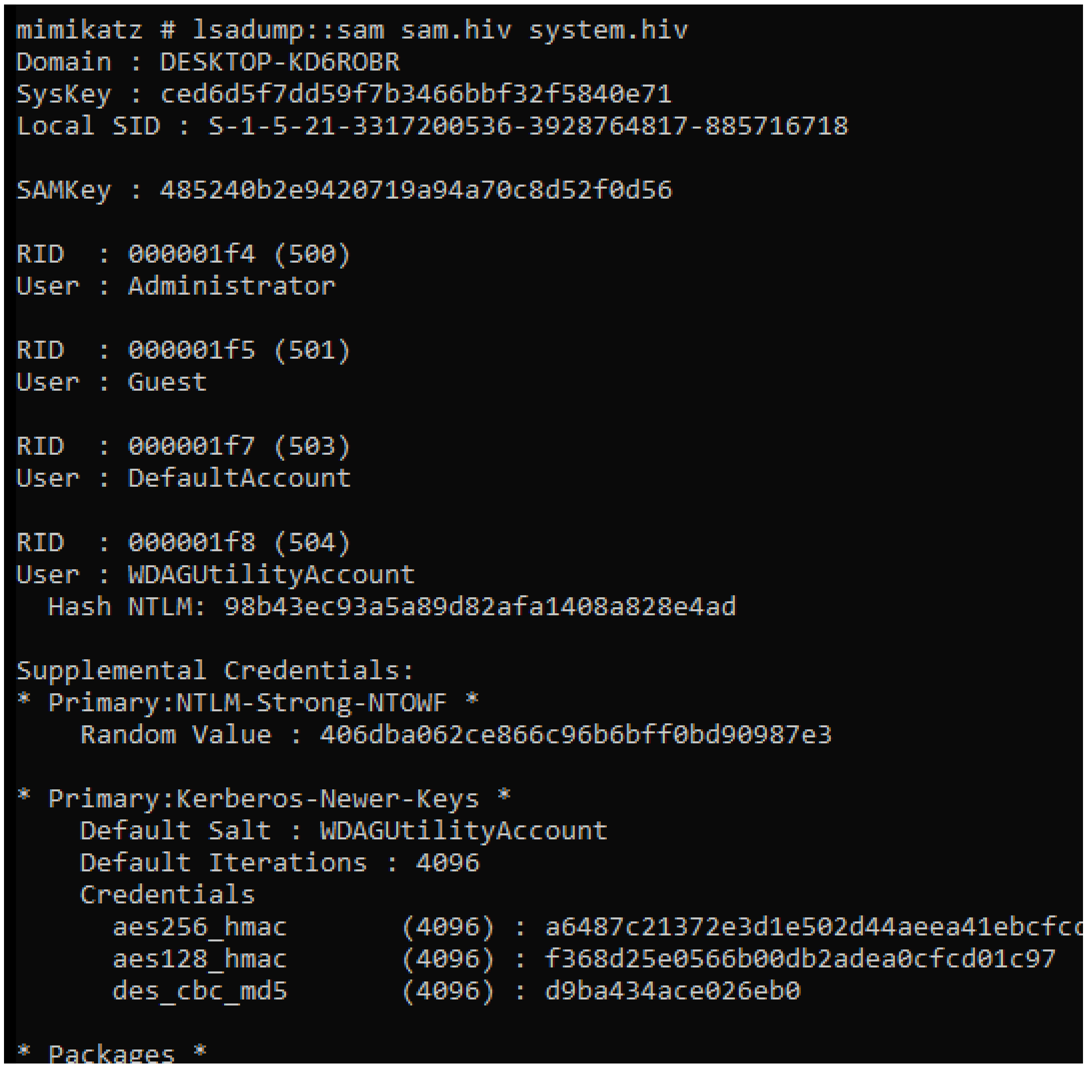
```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

788     {0;000003e7} 0 D 43342          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,31p)       Primary
 -> Impersonated !
 * Process Token : {0;00300822} 2 F 11284061    DESKTOP-KD6ROBR\mifta   S-1-5-21-3317200536-3928764817-885716718-1001   (14g,
24p)       Primary
 * Thread Token  : {0;000003e7} 0 D 11440792    NT AUTHORITY\SYSTEM     S-1-5-18        (04g,31p)       Impersonation (Delega
tion)

mimikatz #
```

Step 3f:



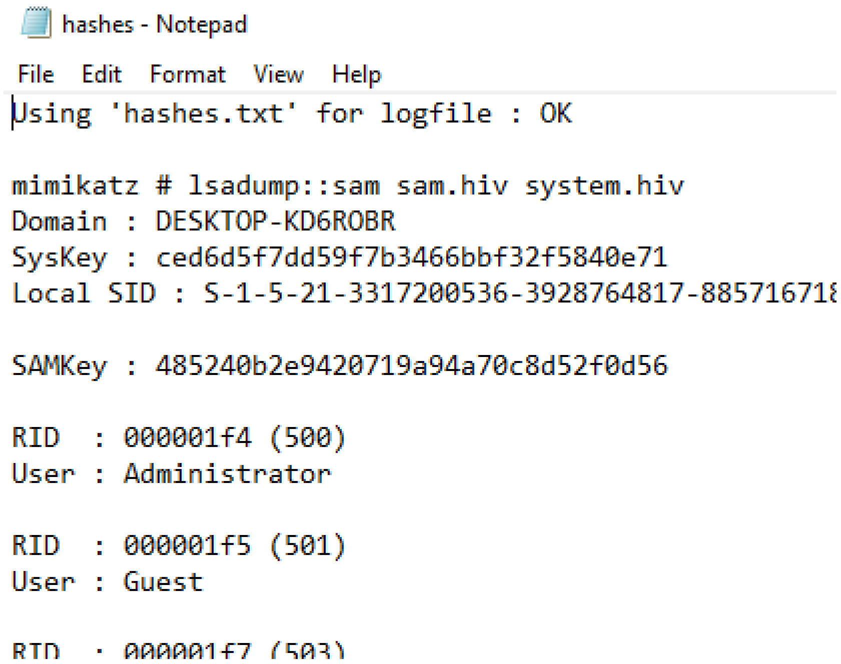
```
mimikatz # log hashes.txt
Using 'hashes.txt' for logfile : OK
```

Step 3g:



```
mimikatz # lsadump::sam sam.hiv system.hiv
Domain : DESKTOP-KD6ROBR
SysKey : ced6d5f7dd59f7b3466bbf32f5840e71
Local SID : S-1-5-21-3317200536-3928764817-885716718

SAMKey : 485240b2e9420719a94a70c8d52f0d56

RID  : 000001f4 (500)
User : Administrator

RID  : 000001f5 (501)
User : Guest

RID  : 000001f7 (503)
User : DefaultAccount

RID  : 000001f8 (504)
User : WDAGUtilityAccount
  Hash NTLM: 98b43ec93a5a89d82afa1408a828e4ad

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 406dba062ce866c96b6bff0bd90987e3

* Primary:Kerberos-Newer-Keys *
    Default Salt : WDAGUtilityAccount
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : a6487c21372e3d1e502d44aeea41ebcfc
      aes128_hmac       (4096) : f368d25e0566b00db2adea0cfcd01c97
      des_cbc_md5       (4096) : d9ba434ace026eb0

* Packages *
```

Step 3h:

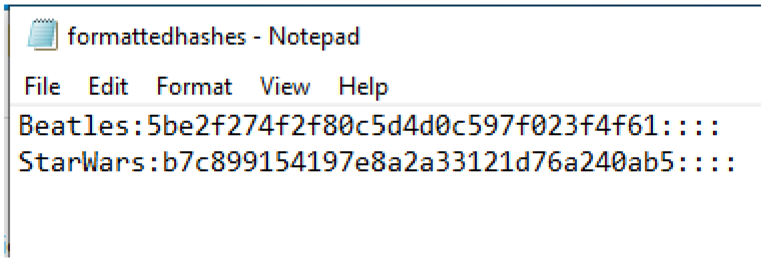


hashes - Notepad

File Edit Format View Help

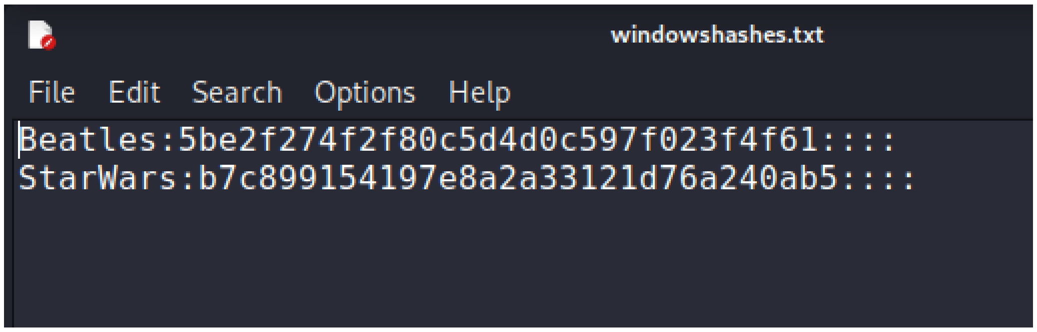
```
Using 'hashes.txt' for logfile : OK

mimikatz # lsadump::sam sam.hiv system.hiv
Domain : DESKTOP-KD6ROBR
SysKey : ced6d5f7dd59f7b3466bbf32f5840e71
Local SID : S-1-5-21-3317200536-3928764817-88571671

SAMKey : 485240b2e9420719a94a70c8d52f0d56

RID  : 000001f4 (500)
User : Administrator

RID  : 000001f5 (501)
User : Guest

RID  : 000001f7 (503)
```

Step 3i:



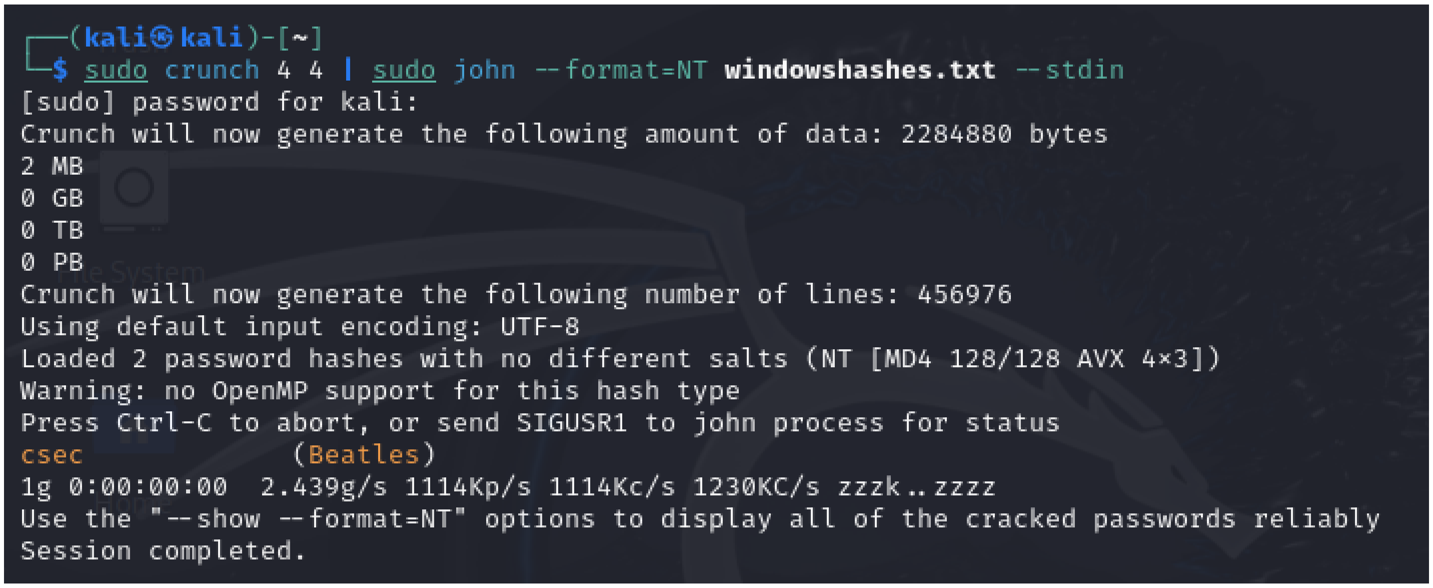
```
formattedhashes - Notepad
File  Edit  Format  View  Help
Beatles:5be2f274f2f80c5d4d0c597f023f4f61:::
StarWars:b7c899154197e8a2a33121d76a240ab5:::
```

Step 3j - k:



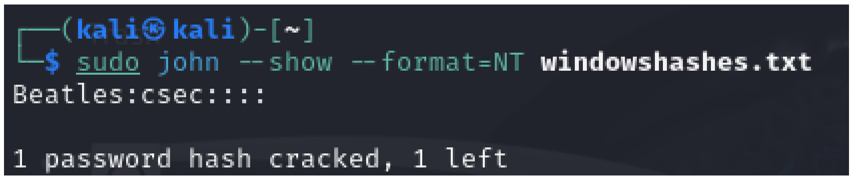
```
windowshashes.txt
File  Edit  Search  Options  Help
Beatles:5be2f274f2f80c5d4d0c597f023f4f61:::
StarWars:b7c899154197e8a2a33121d76a240ab5:::
```

Step 3l:



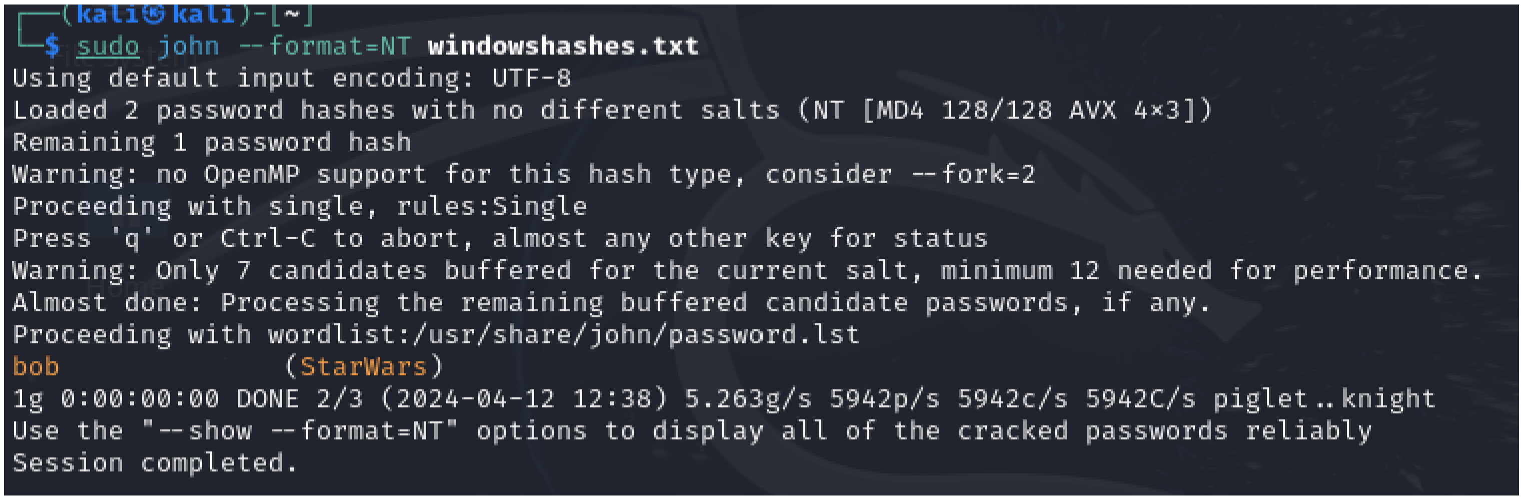
```
┌──(kali㉿kali)-[~]
└─$ sudo crunch 4 4 | sudo john --format=NT windowshashes.txt --stdin
[sudo] password for kali:
Crunch will now generate the following amount of data: 2284880 bytes
2 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 456976
Using default input encoding: UTF-8
Loaded 2 password hashes with no different salts (NT [MD4 128/128 AVX 4×3])
Warning: no OpenMP support for this hash type
Press Ctrl-C to abort, or send SIGUSR1 to john process for status
csec             (Beatles)
1g 0:00:00:00  2.439g/s 1114Kp/s 1114Kc/s 1230KC/s zzzk..zzzz
Use the "--show --format=NT" options to display all of the cracked passwords reliably
Session completed.
```

Step 3m:



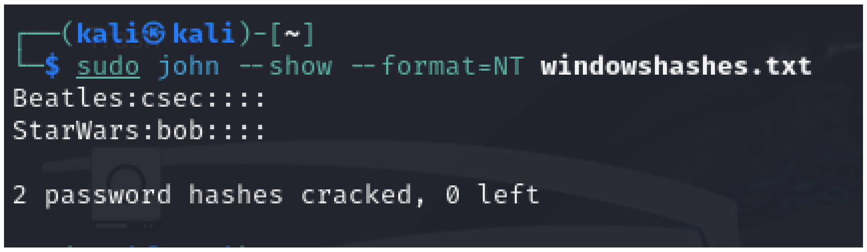
```
┌──(kali㉿kali)-[~]
└─$ sudo john --show --format=NT windowshashes.txt
Beatles:csec::::

1 password hash cracked, 1 left
```

Step 3n:



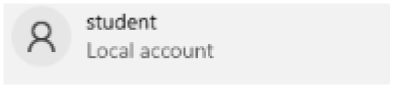
```
┌──(kali㉿kali)-[~]
└─$ sudo john --format=NT windowshashes.txt
Using default input encoding: UTF-8
Loaded 2 password hashes with no different salts (NT [MD4 128/128 AVX 4×3])
Remaining 1 password hash
Warning: no OpenMP support for this hash type, consider --fork=2
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 7 candidates buffered for the current salt, minimum 12 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
bob             (StarWars)
1g 0:00:00:00 DONE 2/3 (2024-04-12 12:38) 5.263g/s 5942p/s 5942c/s 5942C/s piglet..knight
Use the "--show --format=NT" options to display all of the cracked passwords reliably
Session completed.
```

Step 3o:



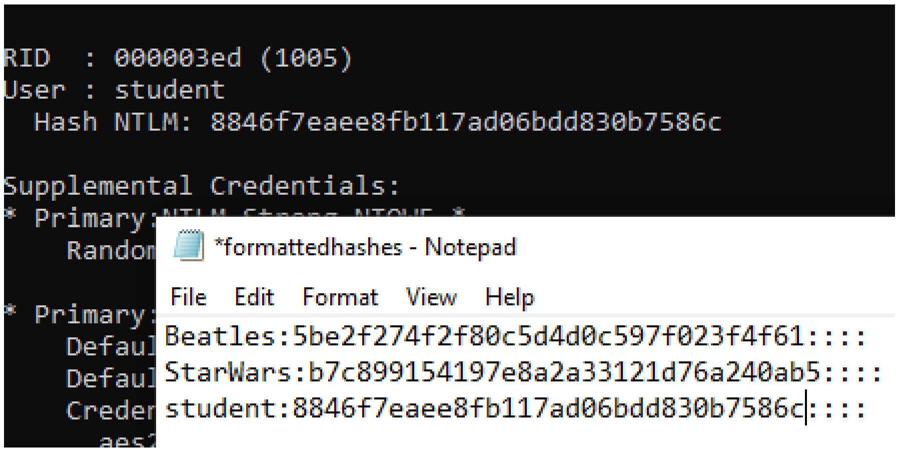
```
┌──(kali㉿kali)-[~]
└─$ sudo john --show --format=NT windowshashes.txt
Beatles:csec::::
StarWars:bob::::

2 password hashes cracked, 0 left
```

Step 4a:
For another user I will using the account that I have created in the beginning of this lab section.



student
Local account

Step 4b:



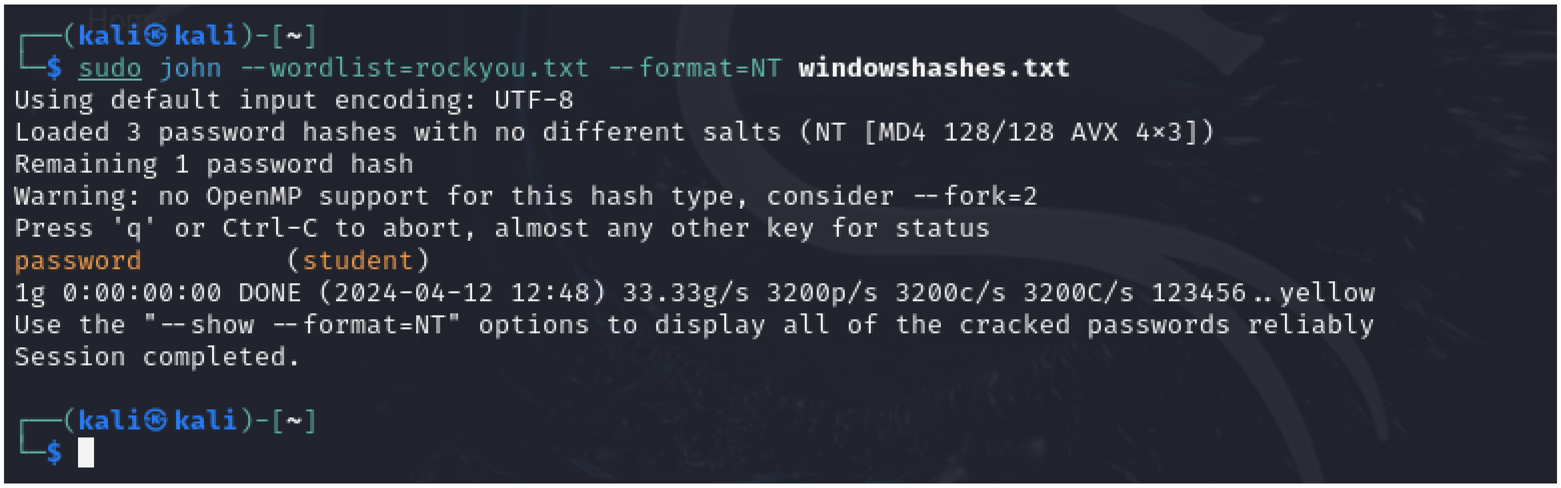
```
RID  : 000003ed (1005)
User : student
  Hash NTLM: 8846f7eaee8fb117ad06bdd830b7586c

Supplemental Credentials:
* Primary:
    Randon

* Primary:
    Defau
    Defau
    Creder
```

*formattedhashes - Notepad

File Edit Format View Help

```
Beatles:5be2f274f2f80c5d4d0c597f023f4f61::::
StarWars:b7c899154197e8a2a33121d76a240ab5::::
student:8846f7eaee8fb117ad06bdd830b7586c::::
```

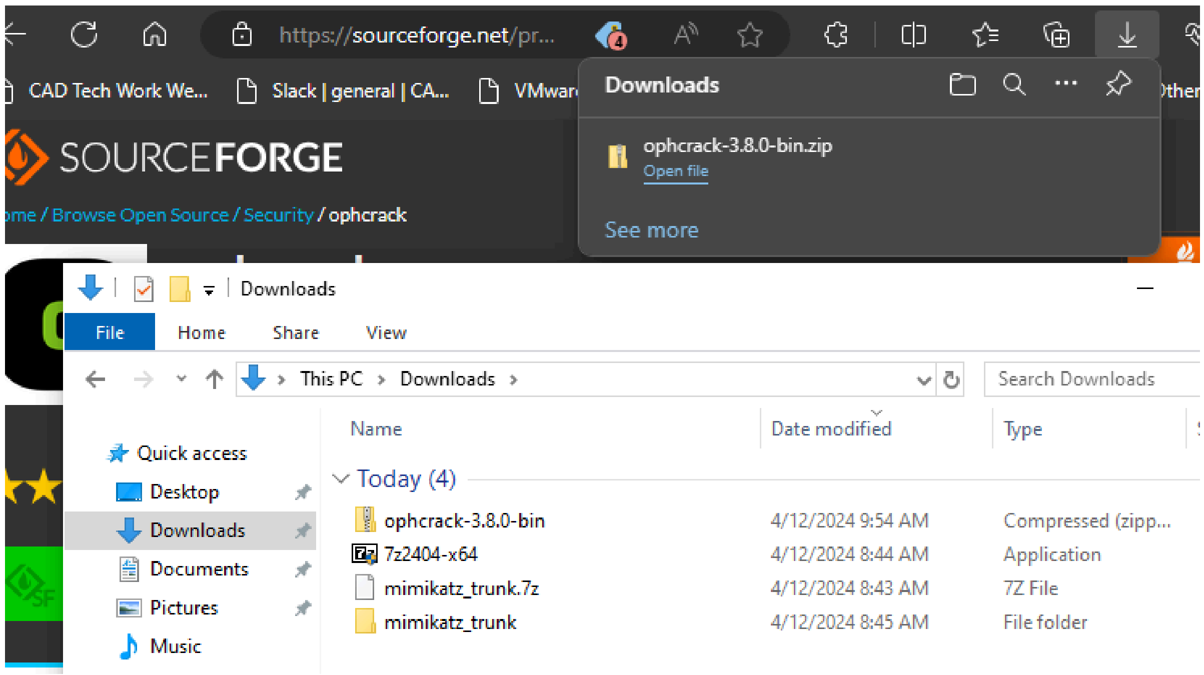
Step 4c:



```
┌──(kali㉿kali)-[~]
└─$ sudo john --wordlist=rockyou.txt --format=NT windowshashes.txt
Using default input encoding: UTF-8
Loaded 3 password hashes with no different salts (NT [MD4 128/128 AVX 4×3])
Remaining 1 password hash
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
password         (student)
1g 0:00:00:00 DONE (2024-04-12 12:48) 33.33g/s 3200p/s 3200c/s 3200C/s 123456..yellow
Use the "--show --format=NT" options to display all of the cracked passwords reliably
Session completed.

┌──(kali㉿kali)-[~]
└─$ 
```

Lab Exercise 11.04: Rainbow Table Attacks on Windows Passwords with ophcrack

Step 1a:



https://sourceforge.net/pr...

CAD Tech Work We... Slack | general | CA... VMware

SOURCEFORGE

ome / Browse Open Source / Security / ophcrack

Downloads

ophcrack-3.8.0-bin.zip
Open file

See more

Downloads

File Home Share View

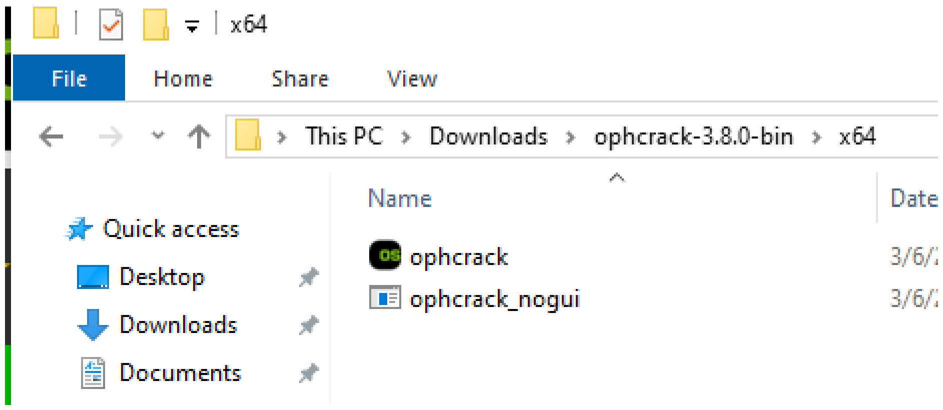
This PC > Downloads >

Search Downloads

Quick access
Desktop
Downloads
Documents
Pictures
Music

Name	Date modified	Type
Today (4)		
ophcrack-3.8.0-bin	4/12/2024 9:54 AM	Compressed (zipp...
7z2404-x64	4/12/2024 8:44 AM	Application
mimikatz_trunk.7z	4/12/2024 8:43 AM	7Z File
mimikatz_trunk	4/12/2024 8:45 AM	File folder

Step 1b - c:



x64

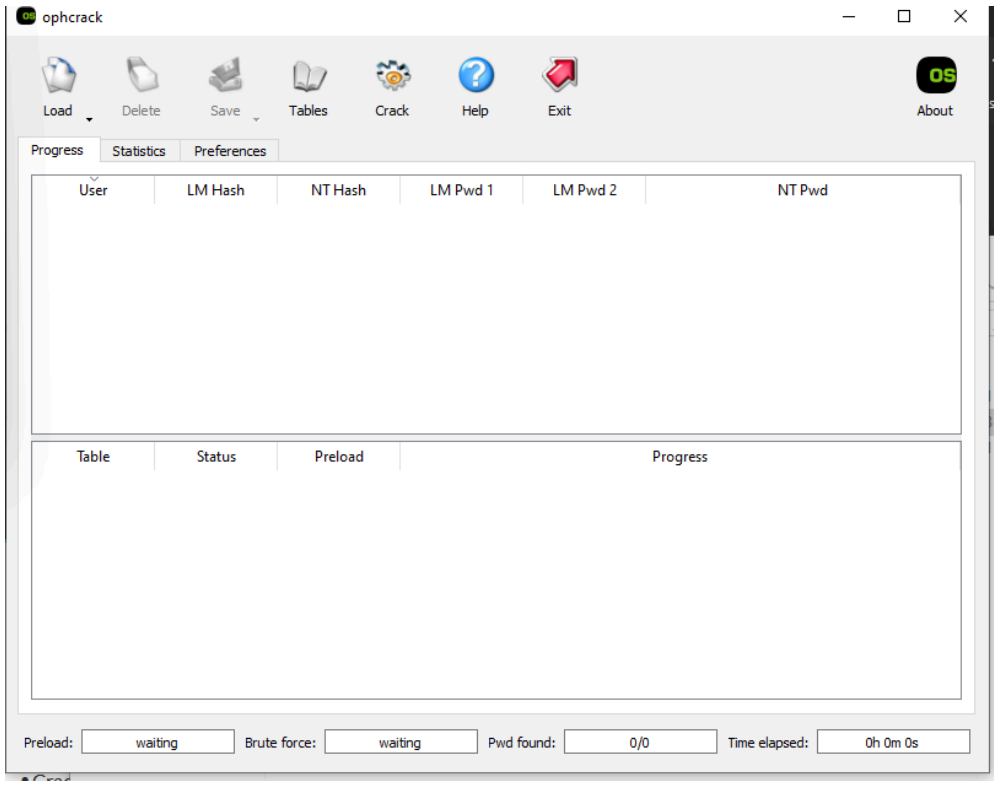
File Home Share View

This PC > Downloads > ophcrack-3.8.0-bin > x64

Quick access
Desktop
Downloads
Documents

Name	Date
ophcrack	3/6/
ophcrack_nogui	3/6/

Step 1d:



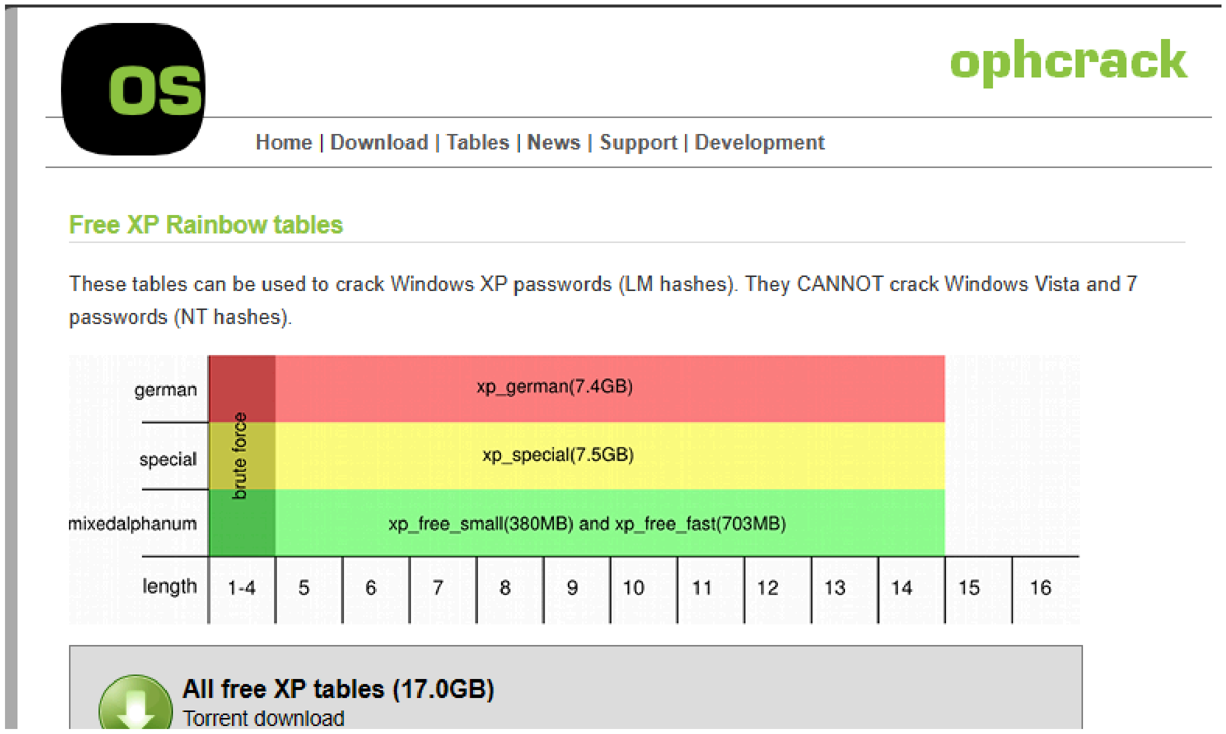
ophcrack

User	LM Hash	NT Hash	LM Pwd 1	LM Pwd 2	NT Pwd

Table	Status	Preload	Progress

Preload: waiting Brute force: waiting Pwd found: 0/0 Time elapsed: 0h 0m 0s

Step 1e:



ophcrack

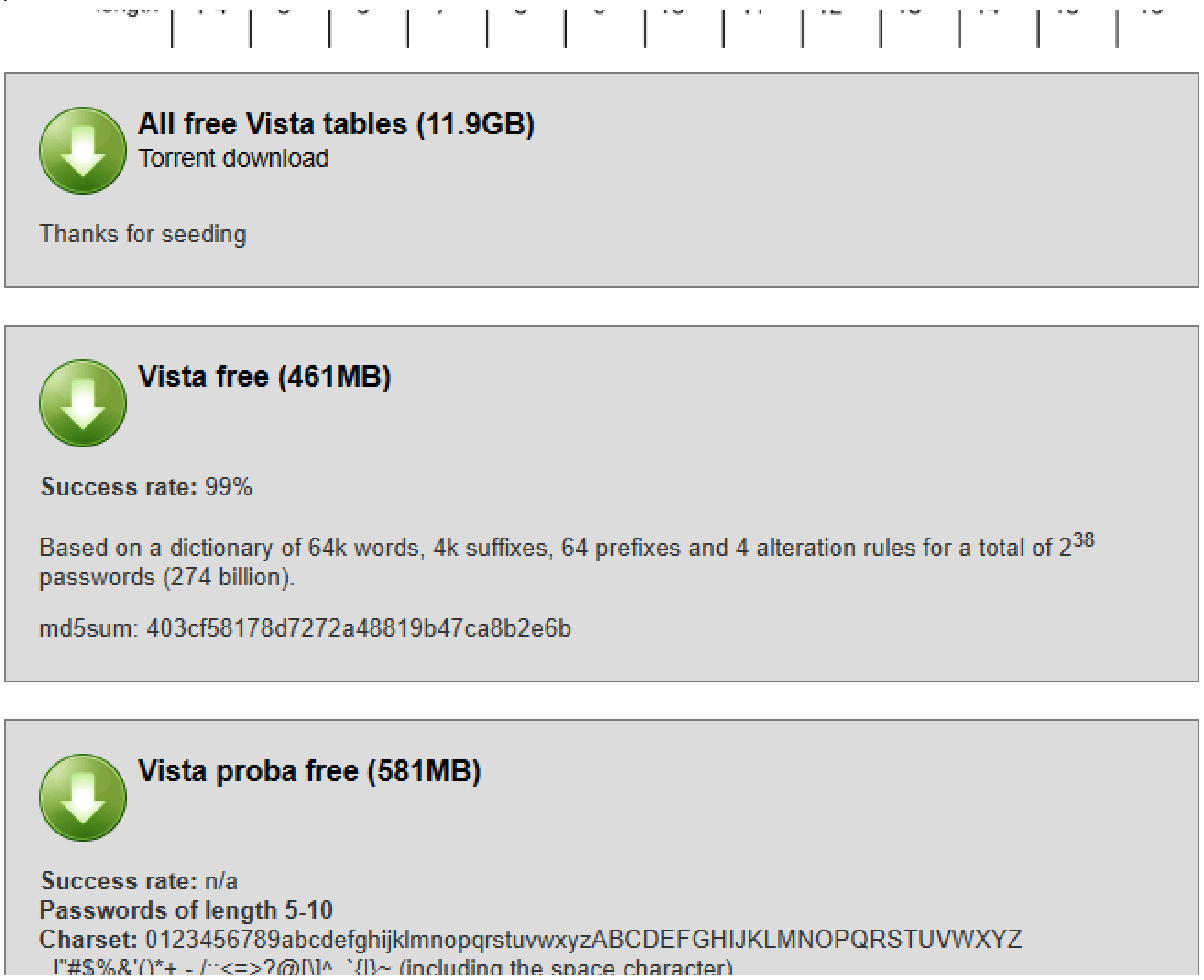
Home | Download | Tables | News | Support | Development

Free XP Rainbow tables

These tables can be used to crack Windows XP passwords (LM hashes). They CANNOT crack Windows Vista and 7 passwords (NT hashes).

All free XP tables (17.0GB)
Torrent download

Step 1f:



All free Vista tables (11.9GB)
Torrent download

Thanks for seeding

Vista free (461MB)

Success rate: 99%

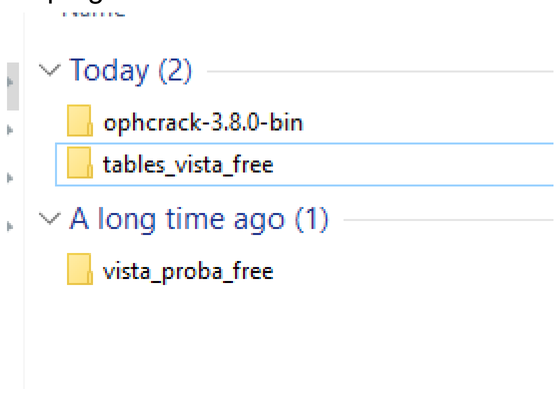
Based on a dictionary of 64k words, 4k suffixes, 64 prefixes and 4 alteration rules for a total of 2^{38} passwords (274 billion).

md5sum: 403cf58178d7272a48819b47ca8b2e6b

Vista proba free (581MB)

Success rate: n/a
Passwords of length 5-10
Charset: 0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
!"#$%&'()*+ - /:;<=>?@[\]^_`{|}~ (including the space character)

Step 1g - h:



Today (2)

ophcrack-3.8.0-bin

tables_vista_free

A long time ago (1)

vista_proba_free

Step 1i:

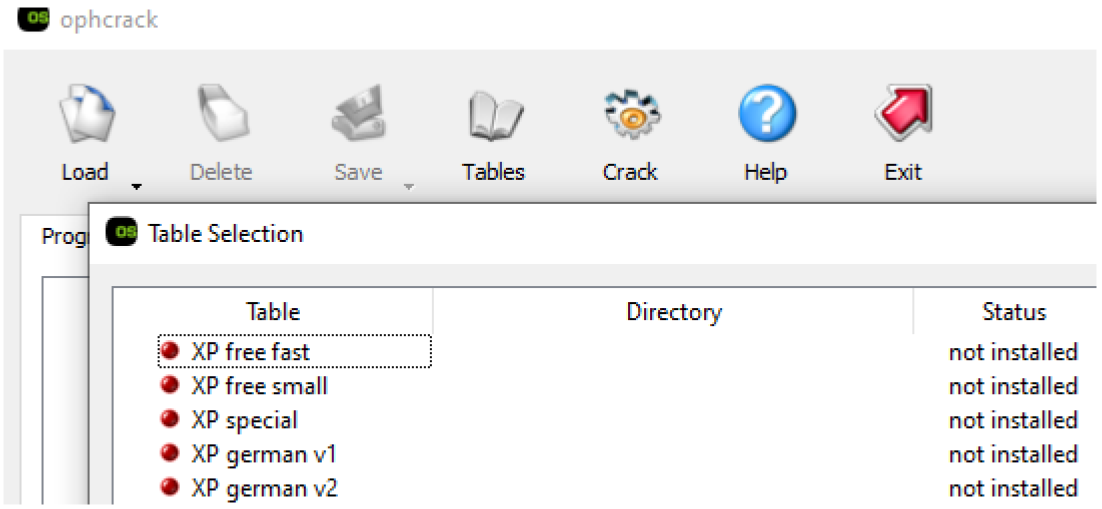
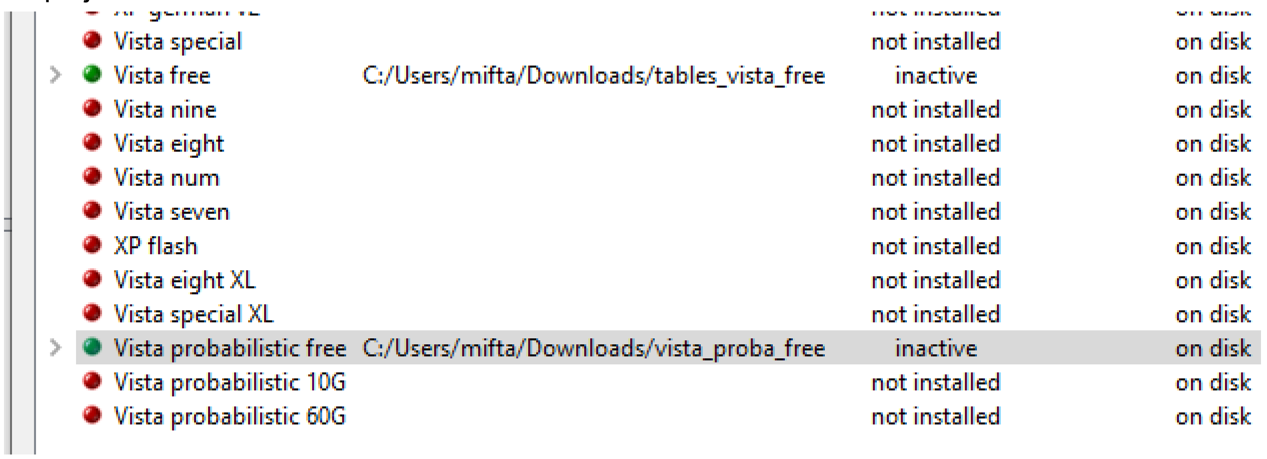


Table Selection

Table	Directory	Status
● XP free fast		not installed
● XP free small		not installed
● XP special		not installed
● XP german v1		not installed
● XP german v2		not installed

Step 1j - k:



● Vista special		not installed	on disk
● Vista free	C:/Users/mifta/Downloads/tables_vista_free	inactive	on disk
● Vista nine		not installed	on disk
● Vista eight		not installed	on disk
● Vista num		not installed	on disk
● Vista seven		not installed	on disk
● XP flash		not installed	on disk
● Vista eight XL		not installed	on disk
● Vista special XL		not installed	on disk
● Vista probabilistic free	C:/Users/mifta/Downloads/vista_proba_free	inactive	on disk
● Vista probabilistic 10G		not installed	on disk
● Vista probabilistic 60G		not installed	on disk

Step 1n:

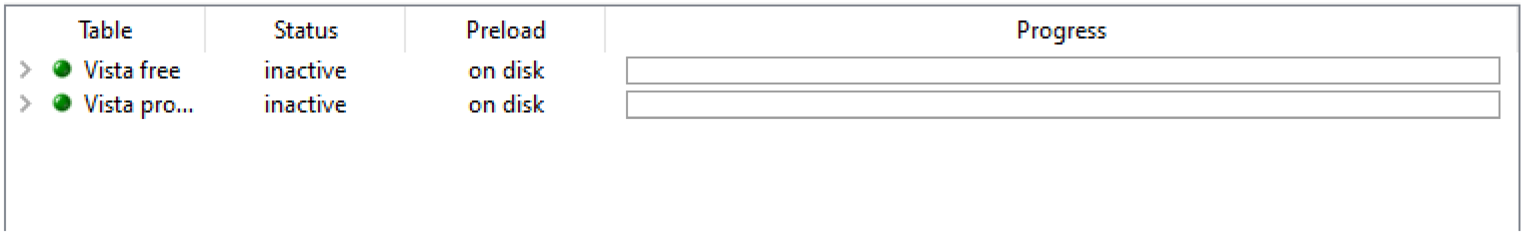
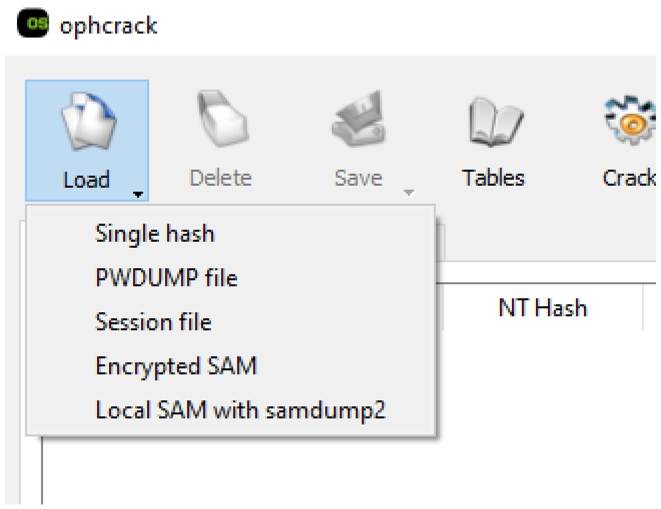
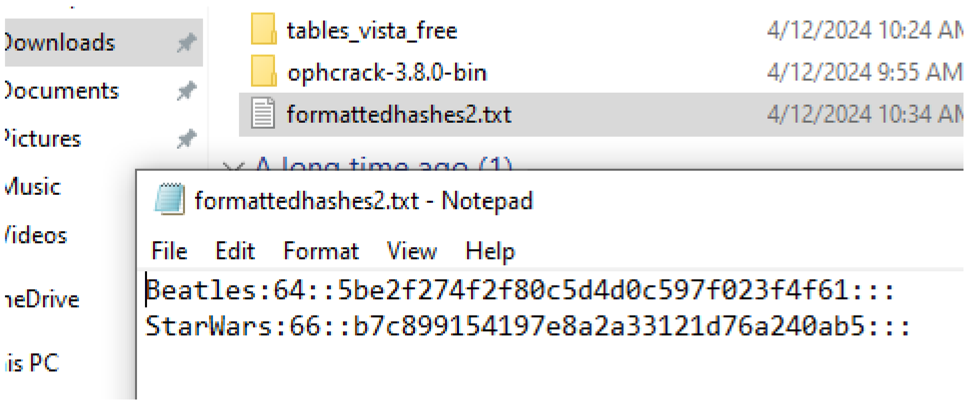


Table	Status	Preload	Progress
● Vista free	inactive	on disk	
● Vista pro...	inactive	on disk	

Step 2a:

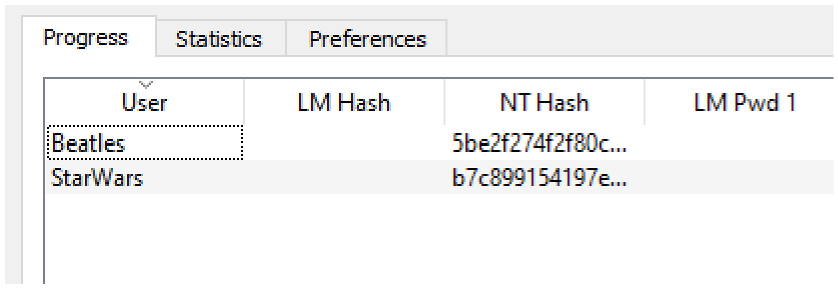


Step 2b:


Downloads
Documents
Pictures
Music
Videos
OneDrive
This PC
tables_vista_free 4/12/2024 10:24 AM
ophcrack-3.8.0-bin 4/12/2024 9:55 AM
formattedhashes2.txt 4/12/2024 10:34 AM
A long time ago (1)
formattedhashes2.txt - Notepad
File Edit Format View Help

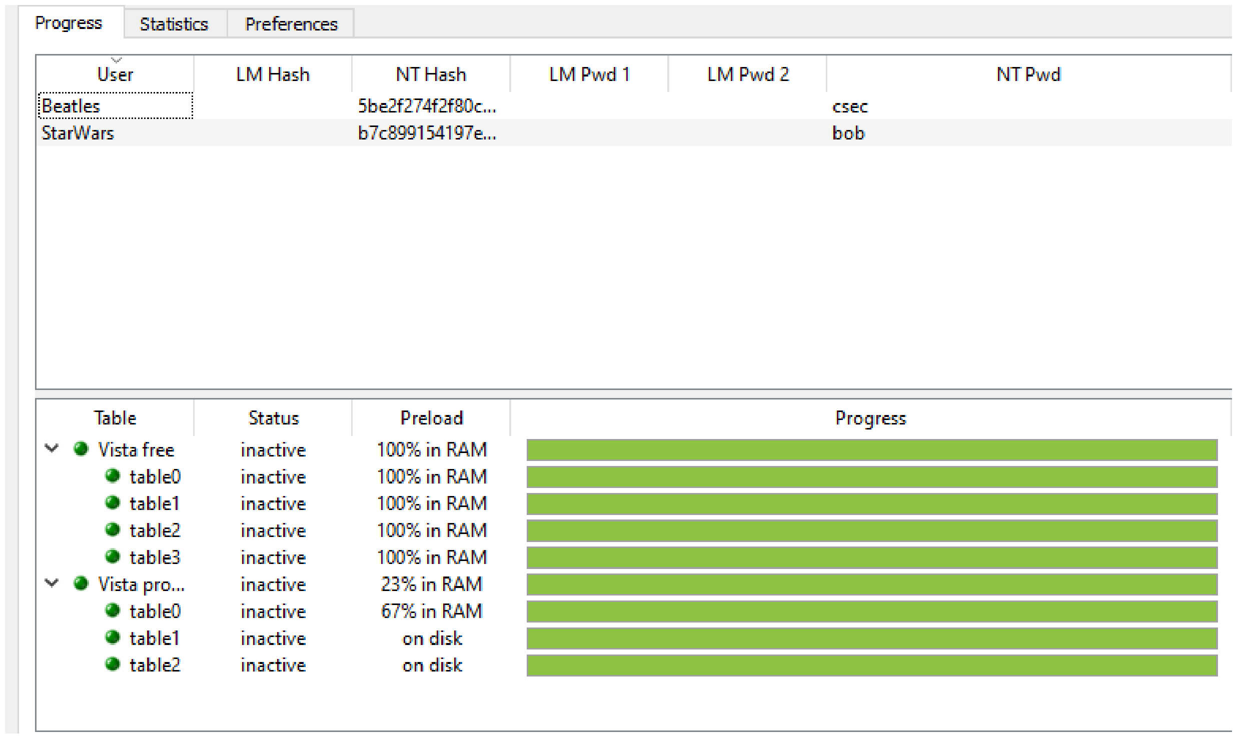
```
Beatles:64::5be2f274f2f80c5d4d0c597f023f4f61:::
StarWars:66::b7c899154197e8a2a33121d76a240ab5:::
```

Step 2c:



User	LM Hash	NT Hash	LM Pwd 1
Beatles		5be2f274f2f80c...	
StarWars		b7c899154197e...	

Step 2d - e:



User	LM Hash	NT Hash	LM Pwd 1	LM Pwd 2	NT Pwd
Beatles		5be2f274f2f80c...			csec
StarWars		b7c899154197e...			bob

Table	Status	Preload	Progress
Vista free	inactive	100% in RAM	
table0	inactive	100% in RAM	
table1	inactive	100% in RAM	
table2	inactive	100% in RAM	
table3	inactive	100% in RAM	
Vista pro...	inactive	23% in RAM	
table0	inactive	67% in RAM	
table1	inactive	on disk	
table2	inactive	on disk	

Lab Analysis:

1. A dictionary attack is using every password in a file of passwords or a dictionary to guess a potential password in the file to figure out the password for the target account.
2. A bruteforce attack is when every combination of passwords is used to guess the password of the target user account.
3. A rainbow table attack is a type of cryptographic attack that uses a precomputed table of hash values for every possible combination of characters to figure out a potential password for the target account.
4. It is essential to store passwords in a hashed format since hashing is a one-way function. This means that even if an attacker gains access to the hashed passwords, they cannot obtain the user's original password directly.

Key Term quiz:

1. The use of **salt** nullifies any rainbow table instantly.
2. Windows hashes are stored in the **SAM** file.
3. The Linux **/etc/shadow** file contains password hashes.
4. One of the most renowned wordlists is **rockyou.txt**.
5. A common hacking tool, used in the wild, is known as **Mimikatz**.
6. Dictionary attacks and brute force attacks can be done with a Linux tool known as **John the Ripper**.
7. Rainbow table attacks can be done with a Windows tool known as **ophcrack**.