ETHICAL HACKING V2
LAB SERIES

Lab 08: Password Cracking with John the Ripper and Hashcat

Document Version: 2020-08-24

Material in this Lab Aligns to the Following	
Books/Certifications	**Chapters/Modules/Objectives**
All-In-One CEH Chapters ISBN-13: 978-1260454550	5: Attacking a System
EC-Council CEH v10 Domain Modules	5: Vulnerability Analysis 6: System Hacking
CompTIA Pentest+ Objectives	4.2: Compare and contrast various use cases of tools 4.3: Given a scenario, analyze tool output or data related to a penetration test
CompTIA All-In-One PenTest+ Chapters ISBN-13: 978-1260135947	4: Vulnerability Scanning and Analysis 7: Network-Based Attacks 9: Web and Database Attacks 10: Attacking Local Host Vulnerabilities

Contents

Introduction 3
Objective 3
Pod Topology 4
Lab Settings 5
1 Generating Password Lists for Password Cracking 6
2 Create User Accounts to be Cracked 9
3 Password Cracking Using John the Ripper 11
4 Password Cracking Using Hashcat 13

Introduction

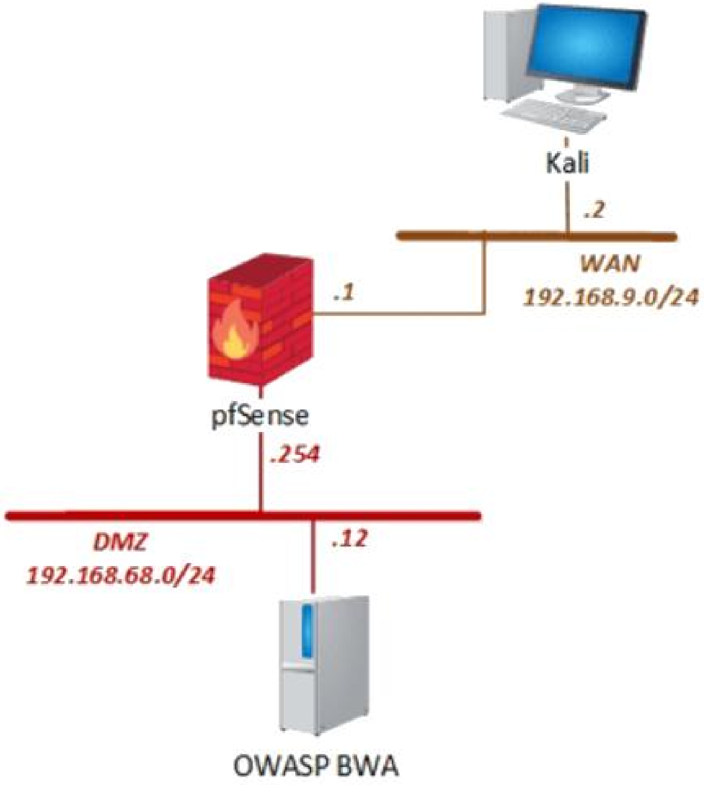
This lab introduces the methodologies used for cracking Linux passwords, using two different tools. In addition, this lab examines how to create supporting wordlists in order to create dictionaries for the tools.

Objective

In this lab, you will be conducting ethical hacking practices using various tools. You will be performing the following tasks:

1. Generating Password Lists for Password Cracking
2. Create User Accounts to be Cracked
3. Password Cracking Using John the Ripper
4. Password Cracking Using Hashcat

Pod Topology



Lab Settings

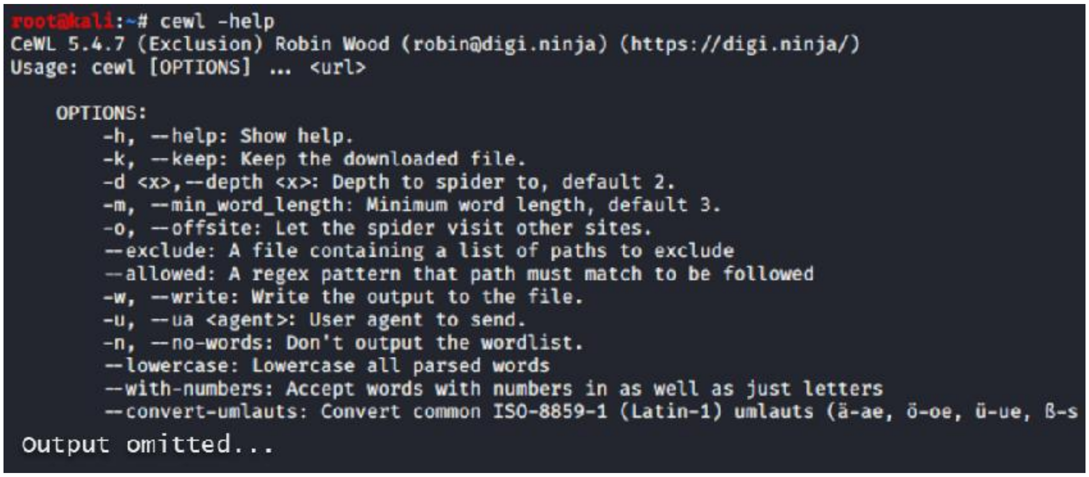
The information in the table below will be needed in order to complete the lab. The task sections below provide details on the use of this information.

Virtual Machine	**IP Address**	**Account** (if needed)	**Password** (if needed)
Kali Linux	192.168.9.2 192.168.0.2	root	toor
pfSense	192.168.0.254 192.168.68.254 192.168.9.1	admin	pfsense
OWASP Broken Web App	192.168.68.12	root	owaspbwa

1 Generating Password Lists for Password Cracking

1. Click on the **Kali** tab.
2. Click within the console window and press **Enter** to display the login prompt.
3. Enter **`root`** as the *username*. Press **Tab**.
4. Enter **`toor`** as the *password*. Click **Log In**.
5. Open a new terminal by clicking on the **Terminal** icon located at the top of the page, if the terminal is not already opened.
6. In the new *Terminal* window, observe the options available for the *cewl* tool. This tool is used to gather text from a website. Type the command below, followed by pressing the **Enter** key.

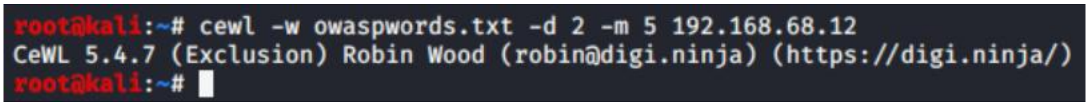
```
cewl -help
```



```
root@kali:~# cewl -help
CeWL 5.4.7 (Exclusion) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
Usage: cewl [OPTIONS] ... <url>

    OPTIONS:
        -h, --help: Show help.
        -k, --keep: Keep the downloaded file.
        -d <x>,--depth <x>: Depth to spider to, default 2.
        -m, --min_word_length: Minimum word length, default 3.
        -o, --offsite: Let the spider visit other sites.
        --exclude: A file containing a list of paths to exclude
        --allowed: A regex pattern that path must match to be followed
        -w, --write: Write the output to the file.
        -u, --ua <agent>: User agent to send.
        -n, --no-words: Don't output the wordlist.
        --lowercase: Lowercase all parsed words
        --with-numbers: Accept words with numbers in as well as just letters
        --convert-umlauts: Convert common ISO-8859-1 (Latin-1) umlauts (ä-ae, ö-oe, ü-ue, ß-s
 Output omitted...
```

7. Target the *OWASP* VM using the *cewl* tool and copy the text to a file named *owaspwords.txt*. Enter the command below.

```
cewl –w owaspwords.txt –d 2 –m 5 192.168.68.12
```

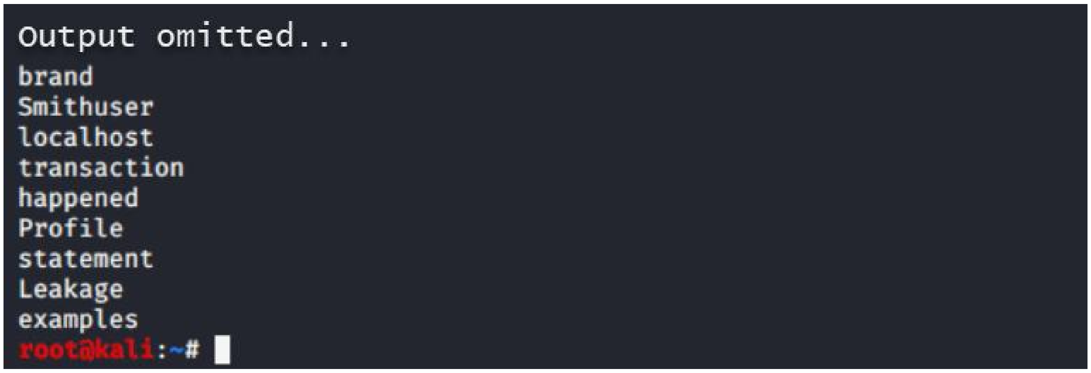


```
root@kali:~# cewl -w owaspwords.txt -d 2 -m 5 192.168.68.12
CeWL 5.4.7 (Exclusion) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
root@kali:~# 
```

Wait about 1-2 minutes until the prompt reappears before continuing to the next step.

8. Once the prompt appears back on the screen, view the contents of the *owaspwords.txt* file. Enter the command below.

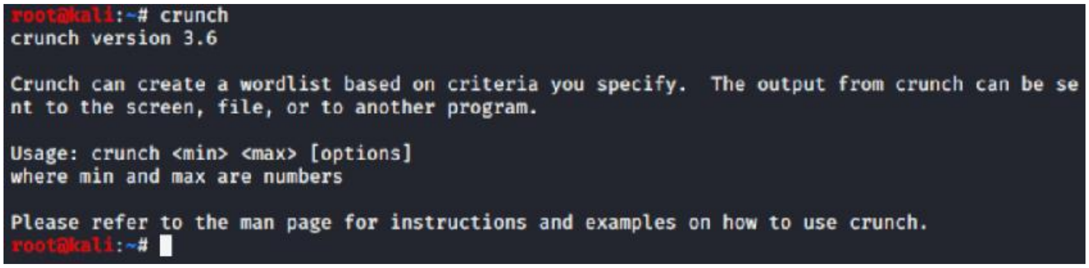
```
cat owaspwords.txt
```



```
Output omitted...
brand
Smithuser
localhost
transaction
happened
Profile
statement
Leakage
examples
root@kali:~#
```

9. *Crunch* is also another utility that can be used to generate passwords. At the prompt, type the command below, followed by pressing the **Enter** key to receive generalized information about the *crunch* utility.

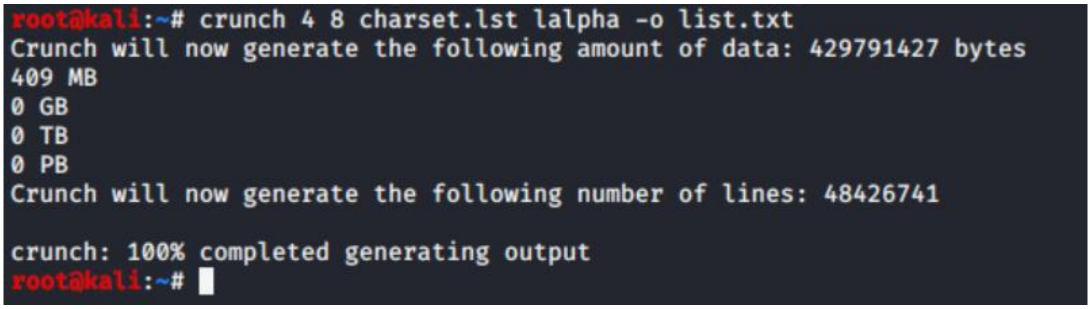
```
crunch
```



```
root@kali:~# crunch
crunch version 3.6

Crunch can create a wordlist based on criteria you specify.  The output from crunch can be se
nt to the screen, 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.
root@kali:~#
```

10. Using *crunch*, generate a set of passwords that are a minimum of **4** characters and a maximum of **8** characters in length, made up of all the **letters of the alphabet** in **lowercase** only. Enter the command below.

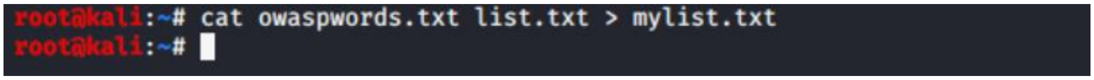
```
crunch 4 8 charset.lst lalpha –o list.txt
```



```
root@kali:~# crunch 4 8 charset.lst lalpha -o list.txt
Crunch will now generate the following amount of data: 429791427 bytes
409 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 48426741

crunch: 100% completed generating output
root@kali:~#
```

11. Once *crunch* is finished generating passwords, combine the wordlist from the *OWASP* web server and the randomly generated *crunch* wordlist.

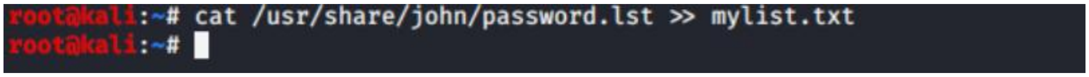
```
cat owaspwords.txt list.txt > mylist.txt
```



```
root@kali:~# cat owaspwords.txt list.txt > mylist.txt
root@kali:~# 
```

12. Add the password list from *John the Ripper* into the combined **mylist.txt** file.

```
cat /usr/share/john/password.lst >> mylist.txt
```



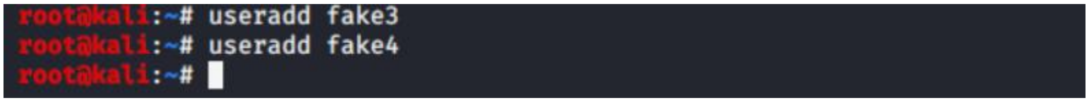
```
root@kali:~# cat /usr/share/john/password.lst >> mylist.txt
root@kali:~# 
```

2 Create User Accounts to be Cracked

1. Create two fake user accounts. Enter the two commands below separately.

```
useradd fake3
```

```
useradd fake4
```

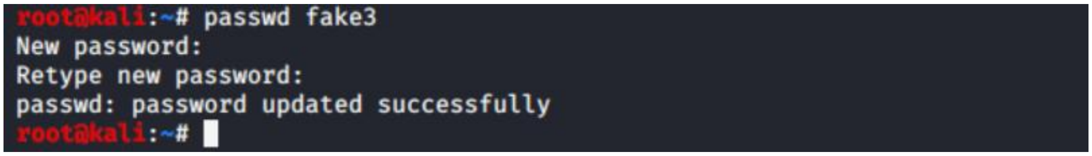


```
root@kali:~# useradd fake3
root@kali:~# useradd fake4
root@kali:~# █
```

2. Assign the user **fake3** a new password by entering the command below.

```
passwd fake3
```

   a. When prompted for a password, type **`123456`**. Press **Enter**.
   b. When prompted to retype the password, type **`123456`** once more. Press **Enter**.

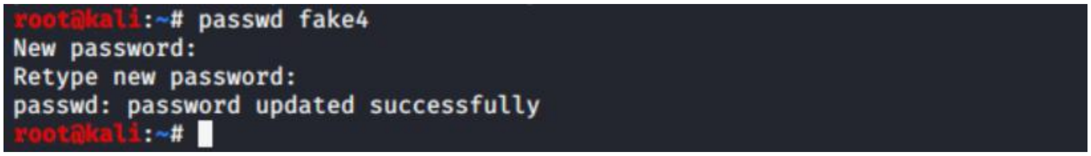


```
root@kali:~# passwd fake3
New password:
Retype new password:
passwd: password updated successfully
root@kali:~# █
```

3. Assign the user **fake4** a new password by entering the command below.

```
passwd fake4
```

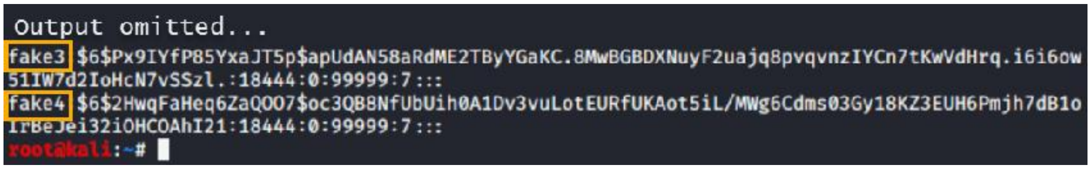
   a. When prompted for a password, type **`password`**. Press **Enter**.
   b. When prompted to retype the password, type **`password`** once more. Press **Enter**.



```
root@kali:~# passwd fake4
New password:
Retype new password:
passwd: password updated successfully
root@kali:~# █
```

4. Confirm that the new users have passwords assigned to them in the **/etc/shadow** file.

```
cat /etc/shadow
```

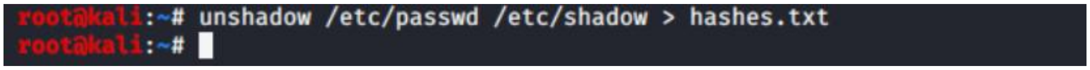


```
Output omitted...
fake3 $6$Px9IYfP85YxaJT5p$apUdAN58aRdME2TByYGaKC.8MwBGBDXNuyF2uajq8pvqvnzIYCn7tKwVdHrq.i6i6ow
51IW7d2IoHcN7vSSzl.:18444:0:99999:7:::
fake4 $6$2HwqFaHeq6ZaQOO7$oc3QB8NfUbUih0A1Dv3vuLotEURfUKAot5iL/MWg6Cdms03Gy18KZ3EUH6Pmjh7dB1o
IrBeJei32iOHCOAhI21:18444:0:99999:7:::
root@kali:~# 
```

Notice the two names with hashed passwords.

5. Combine the two files that make updates to the user's credentials; both **/etc/passwd** and **/etc/shadow**.

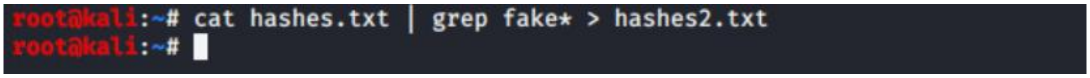
```
unshadow /etc/passwd /etc/shadow > hashes.txt
```



```
root@kali:~# unshadow /etc/passwd /etc/shadow > hashes.txt
root@kali:~# 
```

6. Edit the **hashes.txt** file so that it is narrowed down to only two accounts that need to be cracked.

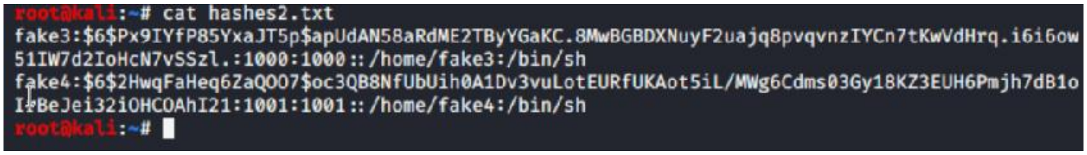
```
cat hashes.txt | grep fake* > hashes2.txt
```



```
root@kali:~# cat hashes.txt | grep fake* > hashes2.txt
root@kali:~# 
```

7. View the contents of the **hashes2.txt** file.

```
cat hashes2.txt
```

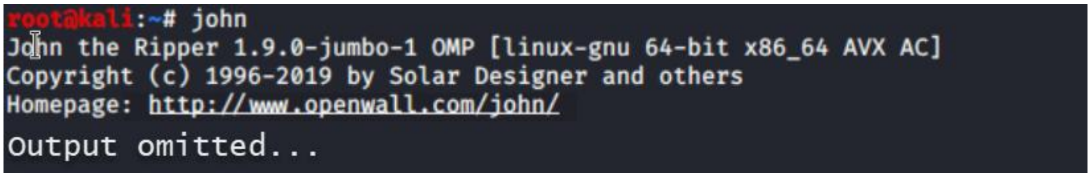


```
root@kali:~# cat hashes2.txt
fake3:$6$Px9IYfP85YxaJT5p$apUdAN58aRdME2TByYGaKC.8MwBGBDXNuyF2uajq8pvqvnzIYCn7tKwVdHrq.i6i6ow
51IW7d2IoHcN7vSSzl.:1000:1000::/home/fake3:/bin/sh
fake4:$6$2HwqFaHeq6ZaQOO7$oc3QB8NfUbUih0A1Dv3vuLotEURfUKAot5iL/MWg6Cdms03Gy18KZ3EUH6Pmjh7dB1o
IrBeJei32iOHCOAhI21:1001:1001::/home/fake4:/bin/sh
root@kali:~# 
```

3 Password Cracking Using John the Ripper

1. *John the Ripper* can be used either through the command line or the GUI version. Start by typing the command below to become familiarized with the command line version. Press **Enter**.

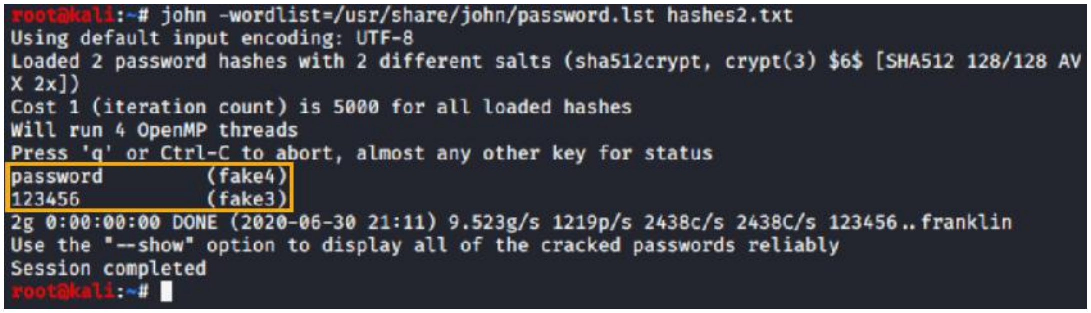
```
john
```



```
root@kali:~# john
John the Ripper 1.9.0-jumbo-1 OMP [linux-gnu 64-bit x86_64 AVX AC]
Copyright (c) 1996-2019 by Solar Designer and others
Homepage: http://www.openwall.com/john/
Output omitted...
```

2. Use *John's* default password list to try to crack the passwords in the *hashes2.txt* file.

```
john –wordlist=/usr/share/john/password.lst hashes2.txt
```



```
root@kali:~# john -wordlist=/usr/share/john/password.lst hashes2.txt
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (sha512crypt, crypt(3) $6$ [SHA512 128/128 AV
X 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
password         (fake4)
123456           (fake3)
2g 0:00:00:00 DONE (2020-06-30 21:11) 9.523g/s 1219p/s 2438c/s 2438C/s 123456..franklin
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~# 
```

Once finished, notice the successful password crack attempt given from the *john* output. Note you could use the mylist.txt password file you created from above. Using this list will take longer as there are many more passwords to try.

Since there is one password per line in these files, you can use the following command to see the password count:

wc -l /root/mylist.txt

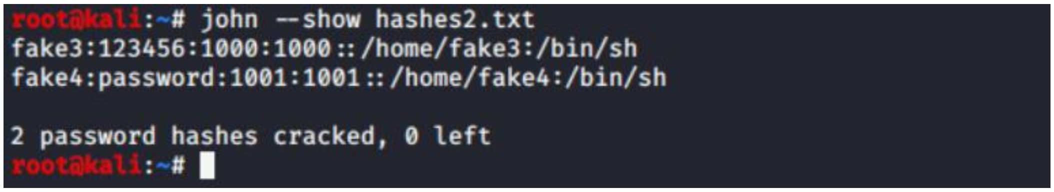
Notice there is 48,432,715 results. Compare that to the default list:

wc -l /usr/share/john/password.lst

Notice there are 3,559 results. This is significantly less and will take less time to run.

3. View the cracked passwords using *john*.

```
john --show hashes2.txt
```



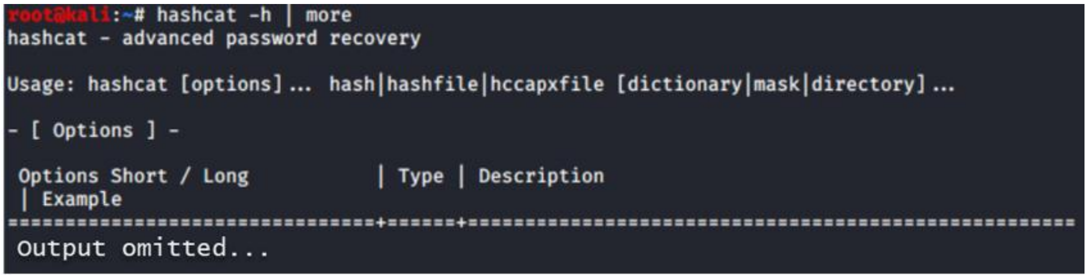
```
root@kali:~# john --show hashes2.txt
fake3:123456:1000:1000::/home/fake3:/bin/sh
fake4:password:1001:1001::/home/fake4:/bin/sh

2 password hashes cracked, 0 left
root@kali:~# 
```

4 Password Cracking Using Hashcat

Hashcat is a powerful password cracking tool that uses an OpenCL library. This allows the tool to use not only the CPU, but GPUs (graphics cards), and other compatible OpenCL hardware. In this lab, you will use CPU only.

1. *Hashcat* can be used for advanced password cracking. Enter the command below to become familiarized with the options made available.

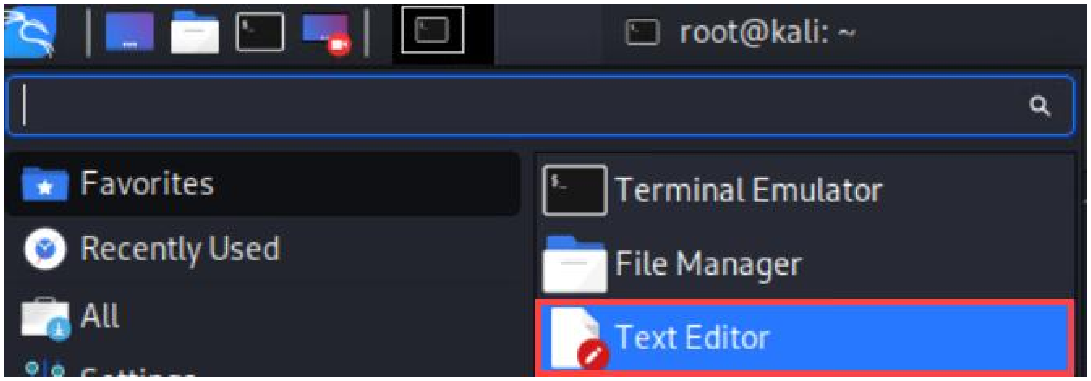
```
hashcat –h | more
```



```
root@kali:~# hashcat -h | more
hashcat - advanced password recovery

Usage: hashcat [options]... hash|hashfile|hccapxfile [dictionary|mask|directory]...

- [ Options ] -

 Options Short / Long           | Type | Description
 | Example
================================+======+==================================================
Output omitted...
```

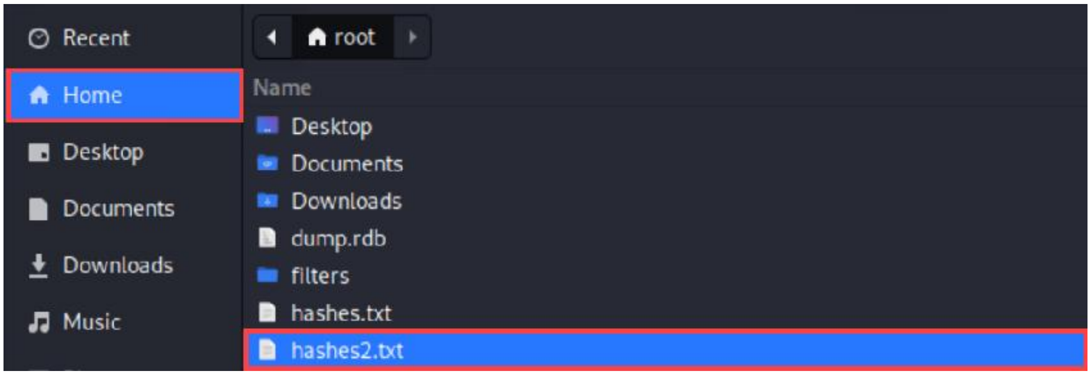
Press the **Spacebar** to skip to the next page or the **Enter** key to skip by each line. Press **Q** to quit the help screen at any given time.

2. Click on the **Application Menu > Text Editor**.



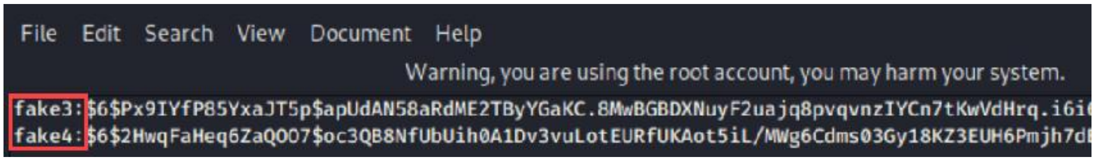
3. On the *Mousepad* window, click on the **File** tab and select **Open**.
4. Click on the **Home** file directory underneath in the left pane.

5. In the right pane, select the **hashes2.txt** file and click **Open**.



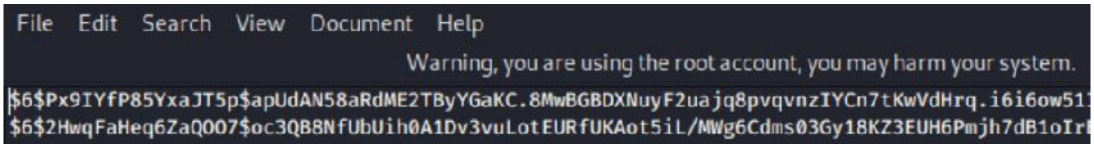
6. Using the file editor, remove the account names, **fake3** and **fake4**, along with the **colon,** as shown below.
   a. Using the arrow keys, move the cursor towards the right and remove all the information to the left of the first colon ":" and the colon itself as shown below. Do this for both lines.

Remove the selected:


File Edit Search View Document Help
Warning, you are using the root account, you may harm your system.

```
fake3:$6$Px9IYfP85YxaJT5p$apUdAN58aRdME2TByYGaKC.8MwBGBDXNuyF2uajq8pvqvnzIYCn7tKwVdHrq.i6i
fake4:$6$2HwqFaHeq6ZaQOO7$oc3QB8NfUbUih0A1Dv3vuLotEURfUKAot5iL/MWg6Cdms03Gy18KZ3EUH6Pmjh7d
```

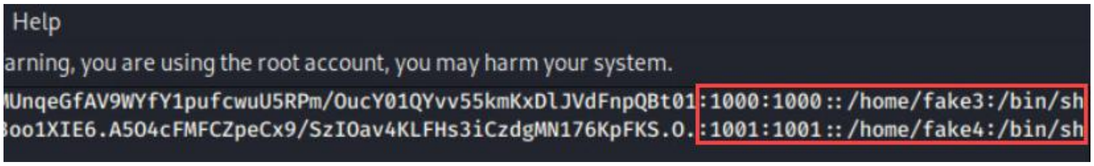
Finalized:


File Edit Search View Document Help
Warning, you are using the root account, you may harm your system.

```
$6$Px9IYfP85YxaJT5p$apUdAN58aRdME2TByYGaKC.8MwBGBDXNuyF2uajq8pvqvnzIYCn7tKwVdHrq.i6i6ow51
$6$2HwqFaHeq6ZaQOO7$oc3QB8NfUbUih0A1Dv3vuLotEURfUKAot5iL/MWg6Cdms03Gy18KZ3EUH6Pmjh7dB1oIr
```

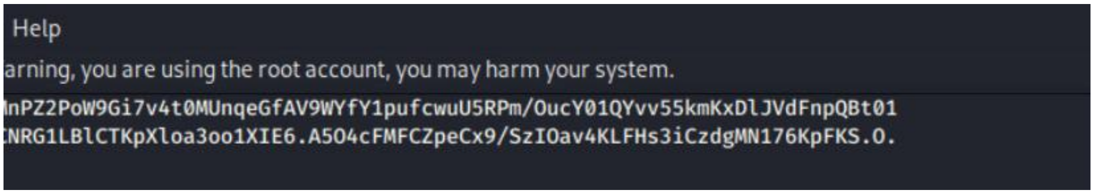
b. Using the arrow keys, move the cursor to the end with the first colon ":" and remove everything to the right, including the colon itself, as shown below. Do this for both lines.

Remove the selected:



```
Help
arning, you are using the root account, you may harm your system.
MUnqeGfAV9WYfY1pufcwuU5RPm/OucY01QYvv55kmKxDlJVdFnpQBt01:1000:1000::/home/fake3:/bin/sh
3oo1XIE6.A5O4cFMFCZpeCx9/SzIOav4KLFHs3iCzdgMN176KpFKS.O.:1001:1001::/home/fake4:/bin/sh
```

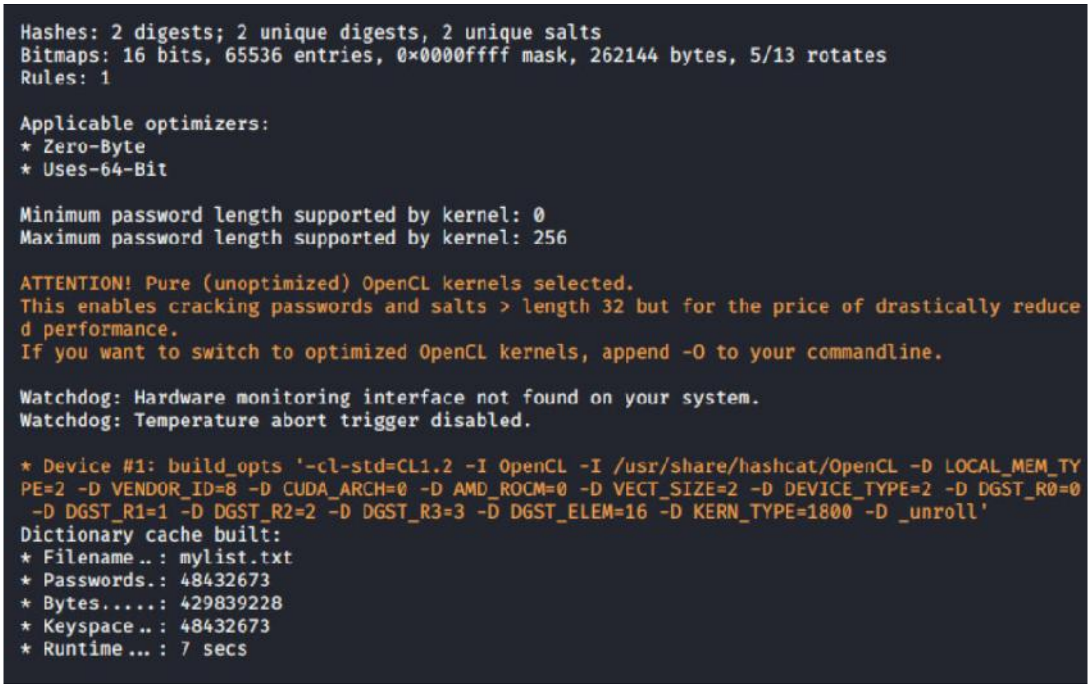
Finalized:



```
Help
arning, you are using the root account, you may harm your system.
nPZ2PoW9Gi7v4t0MUnqeGfAV9WYfY1pufcwuU5RPm/OucY01QYvv55kmKxDlJVdFnpQBt01
NRG1LBlCTKpXloa3oo1XIE6.A5O4cFMFCZpeCx9/SzIOav4KLFHs3iCzdgMN176KpFKS.O.
```

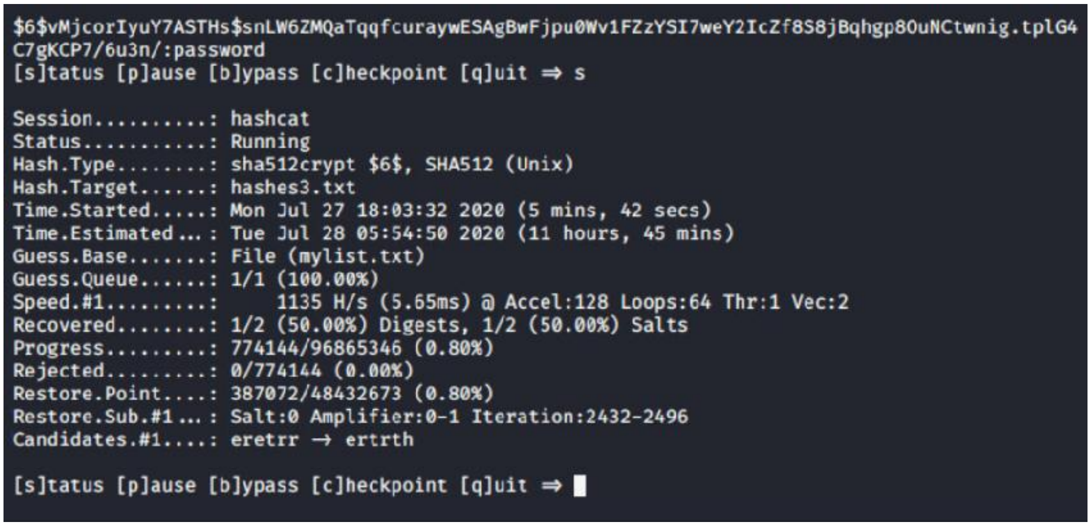
7. After the modifications, click the **File** menu option and select **Saves As**.
8. In the *Save As* window, enter **`hashes3.txt`** in the *Name* text field and have the **root** file directory selected. Click **Save**.
9. Close the **Mousepad** window.
10. Navigate back to the **Terminal** window.
11. Enter the command below to use *Hashcat* in an attempt to password crack the two fake usernames in the **hashes3.txt** file using *John's* password dictionary.

```
hashcat ––force –m 1800 –a 0 hashes3.txt mylist.txt
```



```
Hashes: 2 digests; 2 unique digests, 2 unique salts
Bitmaps: 16 bits, 65536 entries, 0×0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Applicable optimizers:
* Zero-Byte
* Uses-64-Bit

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

ATTENTION! Pure (unoptimized) OpenCL kernels selected.
This enables cracking passwords and salts > length 32 but for the price of drastically reduce
d performance.
If you want to switch to optimized OpenCL kernels, append -O to your commandline.

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

* Device #1: build_opts '-cl-std=CL1.2 -I OpenCL -I /usr/share/hashcat/OpenCL -D LOCAL_MEM_TY
PE=2 -D VENDOR_ID=8 -D CUDA_ARCH=0 -D AMD_ROCM=0 -D VECT_SIZE=2 -D DEVICE_TYPE=2 -D DGST_R0=0
 -D DGST_R1=1 -D DGST_R2=2 -D DGST_R3=3 -D DGST_ELEM=16 -D KERN_TYPE=1800 -D _unroll'
Dictionary cache built:
* Filename..: mylist.txt
* Passwords.: 48432673
* Bytes.....: 429839228
* Keyspace..: 48432673
* Runtime...: 7 secs
```

12. The program will compile this large list and then begin cracking through the iterations. Press the **s** key to check the status.



```
$6$vMjcorIyuY7ASTHs$snLW6ZMQaTqqfcuraywESAgBwFjpu0Wv1FZzYSI7weY2IcZf8S8jBqhgp8OuNCtwnig.tplG4
C7gKCP7/6u3n/:password
[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit ⇒ s

Session..........: hashcat
Status...........: Running
Hash.Type........: sha512crypt $6$, SHA512 (Unix)
Hash.Target......: hashes3.txt
Time.Started.....: Mon Jul 27 18:03:32 2020 (5 mins, 42 secs)
Time.Estimated...: Tue Jul 28 05:54:50 2020 (11 hours, 45 mins)
Guess.Base.......: File (mylist.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:     1135 H/s (5.65ms) @ Accel:128 Loops:64 Thr:1 Vec:2
Recovered........: 1/2 (50.00%) Digests, 1/2 (50.00%) Salts
Progress.........: 774144/96865346 (0.80%)
Rejected.........: 0/774144 (0.00%)
Restore.Point....: 387072/48432673 (0.80%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:2432-2496
Candidates.#1....: eretrr → ertrth

[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit ⇒ █
```

Notice the following fields:

Hash Type – The type of hash. This is set from the -m 1800 argument above.
Time Estimated – This is the estimated completion time. Note the time remaining is high.
Speed #1 – This is the number of hashes/second. This number will vary based the system providing this lab.

13. As this will take a long time, press the **q** key to quit. *Hashcat* will save your progress and allow you to continue where you left off.
13. You may now end your reservation.