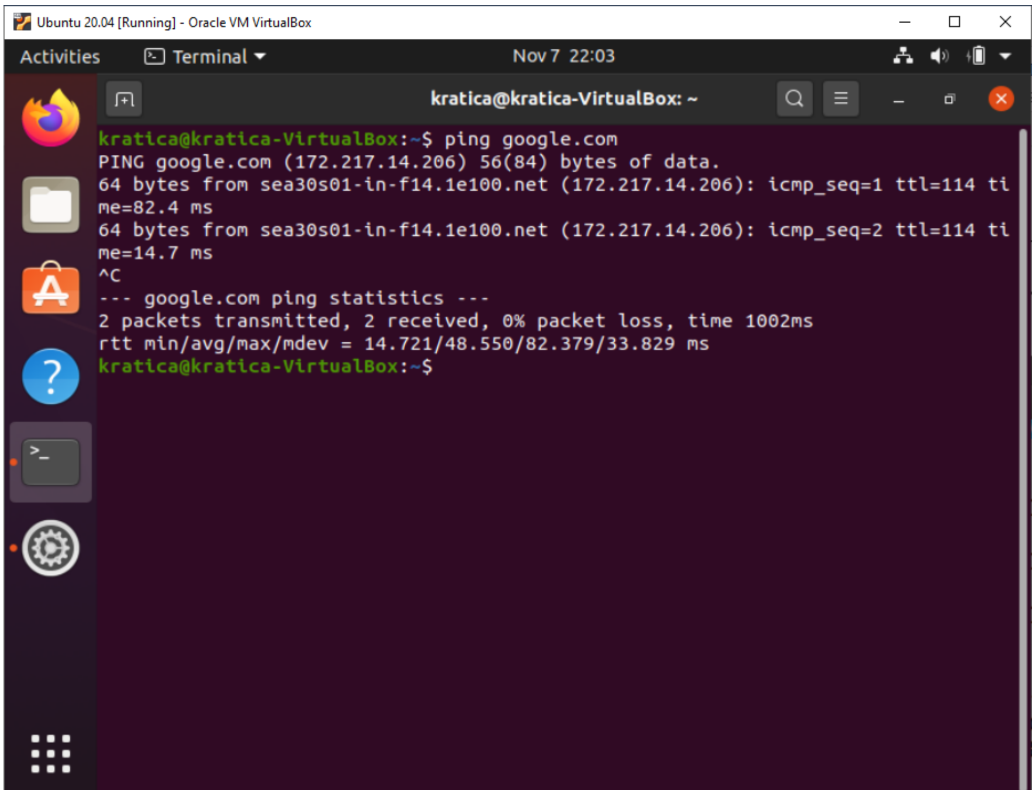
Cracking Linux Password Hashes with Hashcat

Part 1

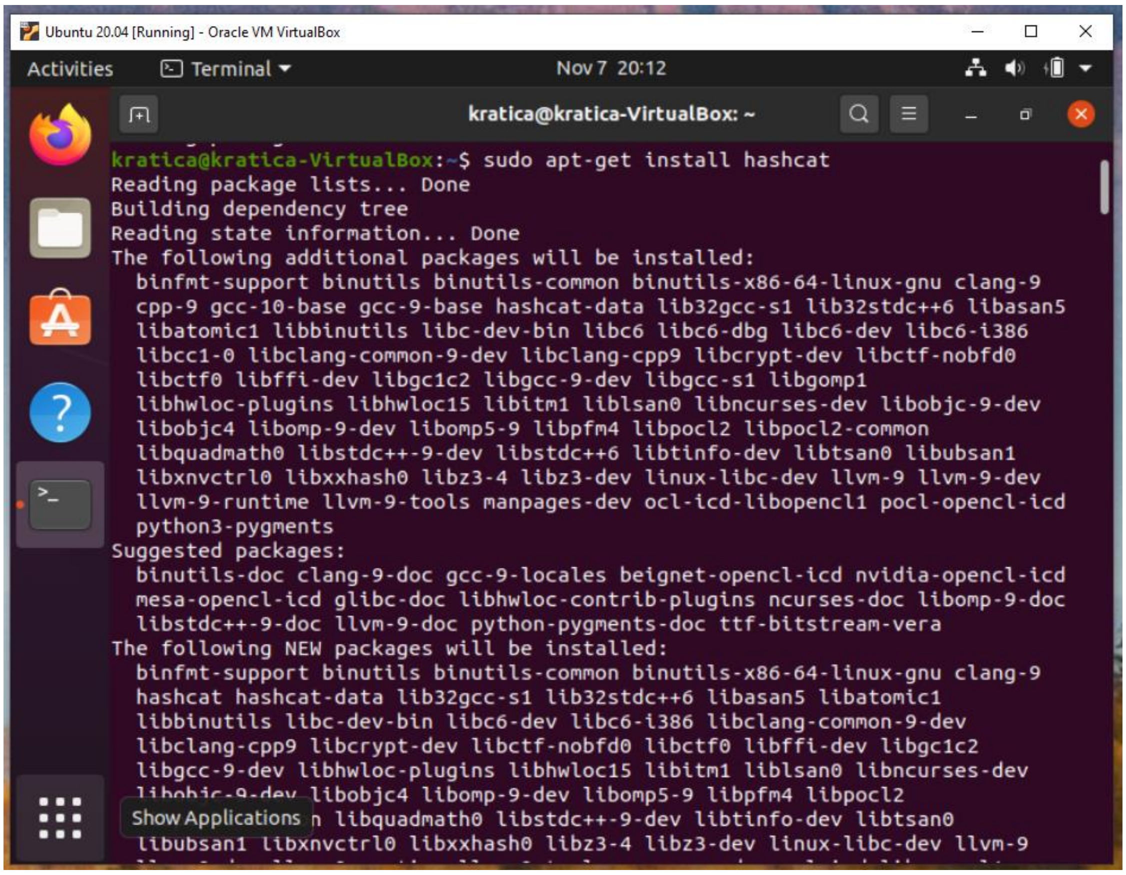
1. Updating Hashcat

- Executed "ping google.com" on terminal window. Getting 64 bytes… in response.

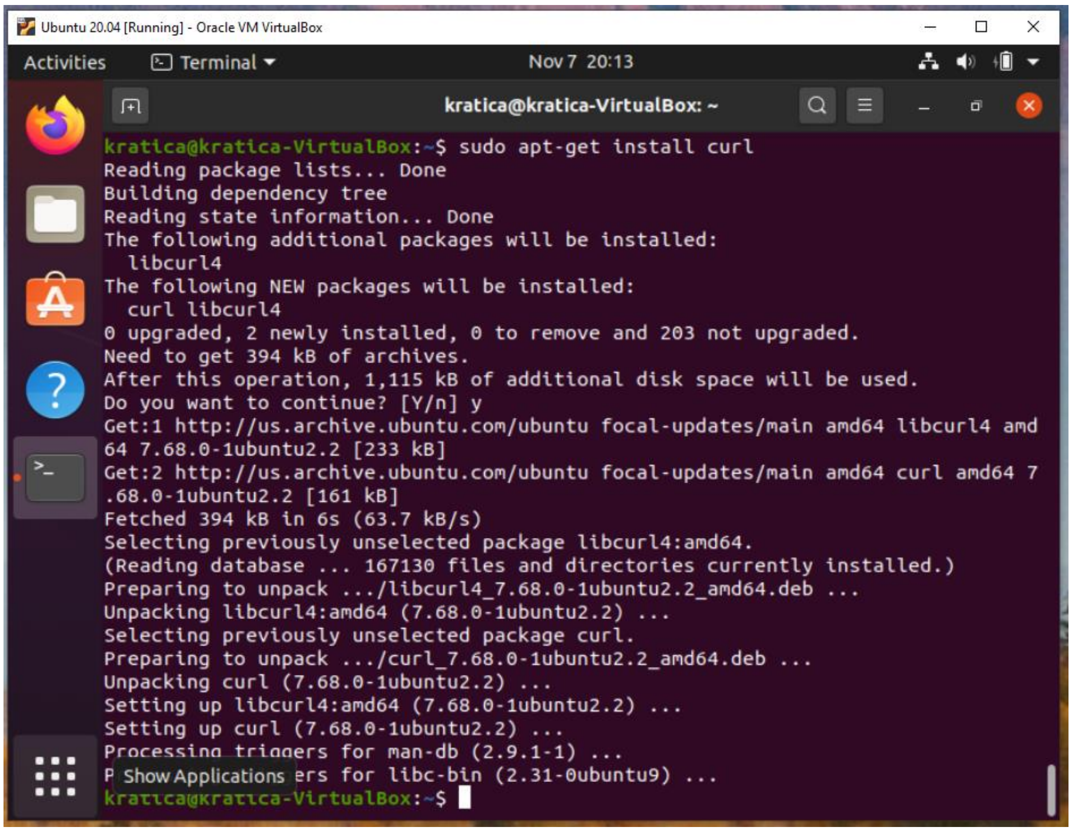

Ubuntu 20.04 [Running] - Oracle VM VirtualBox
Activities
Terminal
Nov 7 22:03
kratica@kratica-VirtualBox: ~

```
kratica@kratica-VirtualBox:~$ ping google.com
PING google.com (172.217.14.206) 56(84) bytes of data.
64 bytes from sea30s01-in-f14.1e100.net (172.217.14.206): icmp_seq=1 ttl=114 ti
me=82.4 ms
64 bytes from sea30s01-in-f14.1e100.net (172.217.14.206): icmp_seq=2 ttl=114 ti
me=14.7 ms
^C
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 14.721/48.550/82.379/33.829 ms
kratica@kratica-VirtualBox:~$
```

- Executed below commands:
  sudo apt-get update
  sudo apt install hashcat
  sudo apt install curl


Ubuntu 20.04 [Running] - Oracle VM VirtualBox
Activities
Terminal
Nov 7 20:12
kratica@kratica-VirtualBox: ~
Show Applications

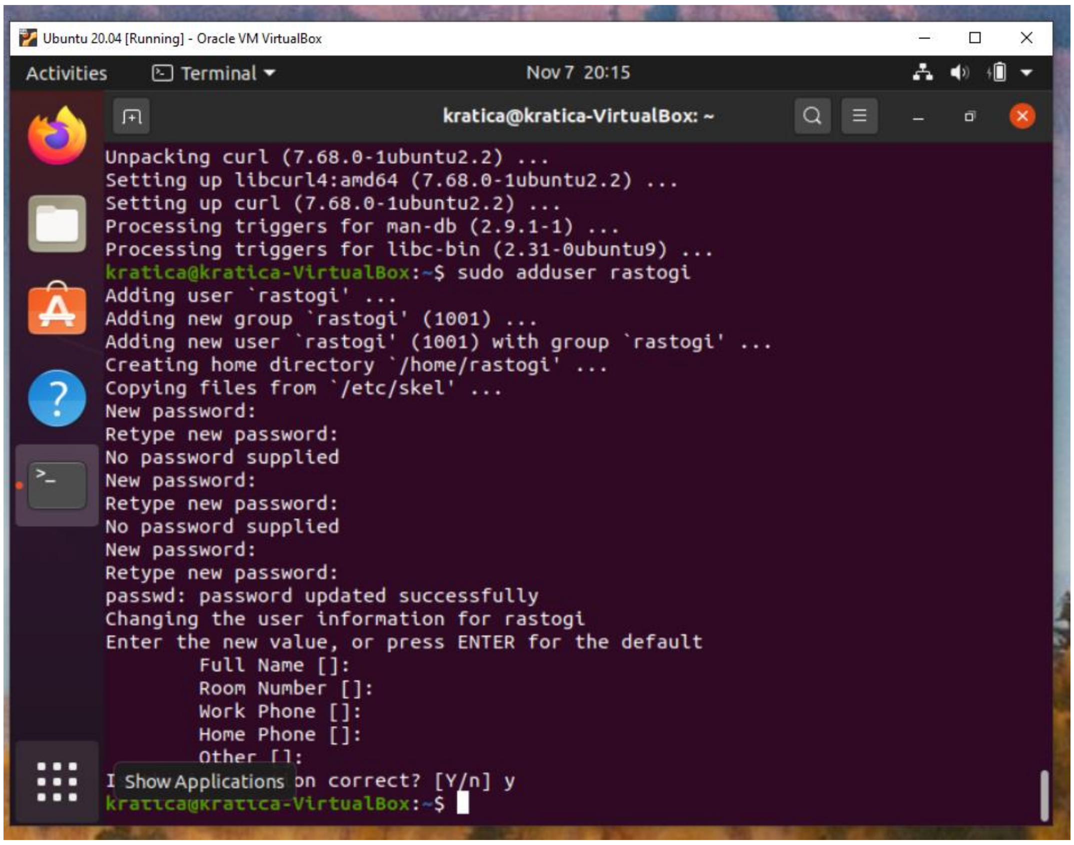
```
kratica@kratica-VirtualBox:~$ sudo apt-get install hashcat
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  binfmt-support binutils binutils-common binutils-x86-64-linux-gnu clang-9
  cpp-9 gcc-10-base gcc-9-base hashcat-data lib32gcc-s1 lib32stdc++6 libasan5
  libatomic1 libbinutils libc-dev-bin libc6 libc6-dbg libc6-dev libc6-i386
  libcc1-0 libclang-common-9-dev libclang-cpp9 libcrypt-dev libctf-nobfd0
  libctf0 libffi-dev libgcc1c2 libgcc-9-dev libgcc-s1 libgomp1
  libhwloc-plugins libhwloc15 libitm1 liblsan0 libncurses-dev libobjc-9-dev
  libobjc4 libomp-9-dev libomp5-9 libpfm4 libpocl2 libpocl2-common
  libquadmath0 libstdc++-9-dev libstdc++6 libtinfo-dev libtsan0 libubsan1
  libxnvctrl0 libxxhash0 libz3-4 libz3-dev linux-libc-dev llvm-9 llvm-9-dev
  llvm-9-runtime llvm-9-tools manpages-dev ocl-icd-libopencl1 pocl-opencl-icd
  python3-pygments
Suggested packages:
  binutils-doc clang-9-doc gcc-9-locales beignet-opencl-icd nvidia-opencl-icd
  mesa-opencl-icd glibc-doc libhwloc-contrib-plugins ncurses-doc libomp-9-doc
  libstdc++-9-doc llvm-9-doc python-pygments-doc ttf-bitstream-vera
The following NEW packages will be installed:
  binfmt-support binutils binutils-common binutils-x86-64-linux-gnu clang-9
  hashcat hashcat-data lib32gcc-s1 lib32stdc++6 libasan5 libatomic1
  libbinutils libc-dev-bin libc6-dev libc6-i386 libclang-common-9-dev
  libclang-cpp9 libcrypt-dev libctf-nobfd0 libctf0 libffi-dev libgc1c2
  libgcc-9-dev libhwloc-plugins libhwloc15 libitm1 liblsan0 libncurses-dev
  libobjc-9-dev libobjc4 libomp-9-dev libomp5-9 libpfm4 libpocl2
                    n libquadmath0 libstdc++-9-dev libtinfo-dev libtsan0
  libubsan1 libxnvctrl0 libxxhash0 libz3-4 libz3-dev linux-libc-dev llvm-9
```


Ubuntu 20.04 [Running] - Oracle VM VirtualBox
Activities
Terminal
Nov 7 20:13
kratica@kratica-VirtualBox: ~
Show Applications

```
kratica@kratica-VirtualBox:~$ sudo apt-get install curl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libcurl4
The following NEW packages will be installed:
  curl libcurl4
0 upgraded, 2 newly installed, 0 to remove and 203 not upgraded.
Need to get 394 kB of archives.
After this operation, 1,115 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 libcurl4 amd
64 7.68.0-1ubuntu2.2 [233 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 curl amd64 7
.68.0-1ubuntu2.2 [161 kB]
Fetched 394 kB in 6s (63.7 kB/s)
Selecting previously unselected package libcurl4:amd64.
(Reading database ... 167130 files and directories currently installed.)
Preparing to unpack .../libcurl4_7.68.0-1ubuntu2.2_amd64.deb ...
Unpacking libcurl4:amd64 (7.68.0-1ubuntu2.2) ...
Selecting previously unselected package curl.
Preparing to unpack .../curl_7.68.0-1ubuntu2.2_amd64.deb ...
Unpacking curl (7.68.0-1ubuntu2.2) ...
Setting up libcurl4:amd64 (7.68.0-1ubuntu2.2) ...
Setting up curl (7.68.0-1ubuntu2.2) ...
Processing triggers for man-db (2.9.1-1) ...
P                  ers for libc-bin (2.31-0ubuntu9) ...
kratica@kratica-VirtualBox:~$
```

2. Creating Test User

- Executed below command to create test user
  sudo adduser rastogi

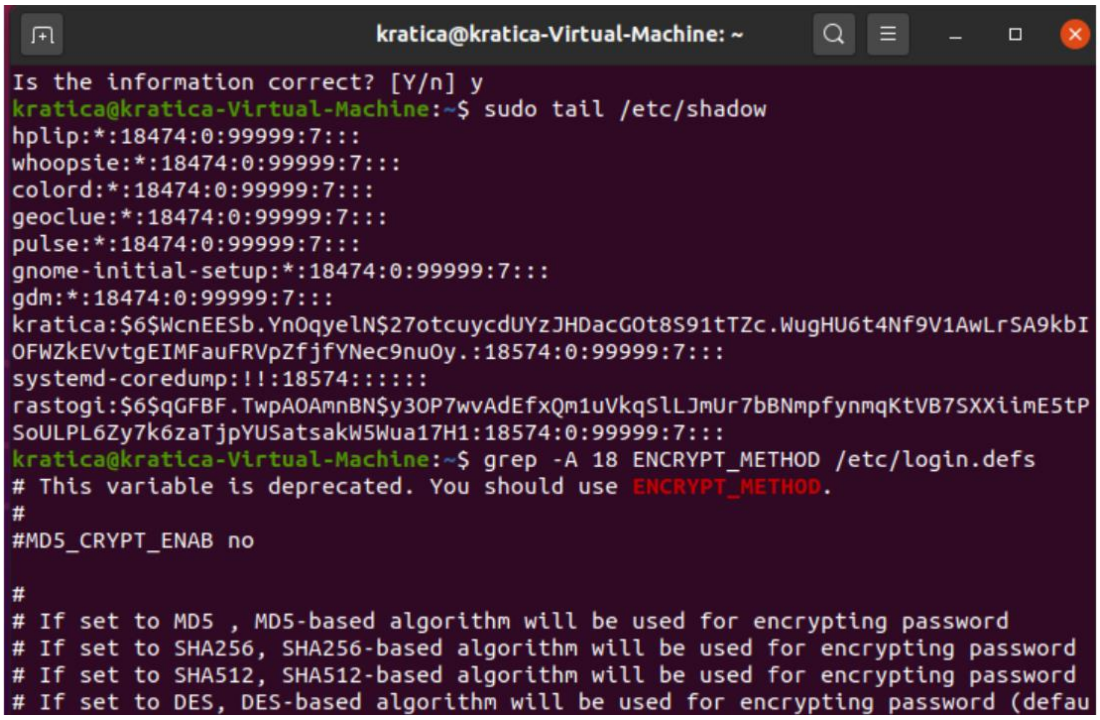


Ubuntu 20.04 [Running] - Oracle VM VirtualBox
Activities
Terminal
Nov 7 20:15
kratica@kratica-VirtualBox: ~
Show Applications

```
Unpacking curl (7.68.0-1ubuntu2.2) ...
Setting up libcurl4:amd64 (7.68.0-1ubuntu2.2) ...
Setting up curl (7.68.0-1ubuntu2.2) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9) ...
kratica@kratica-VirtualBox:~$ sudo adduser rastogi
Adding user `rastogi' ...
Adding new group `rastogi' (1001) ...
Adding new user `rastogi' (1001) with group `rastogi' ...
Creating home directory `/home/rastogi' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
No password supplied
New password:
Retype new password:
No password supplied
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for rastogi
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
I                   on correct? [Y/n] y
kratica@kratica-VirtualBox:~$
```

3. Viewing the Password Hash

- After executing below command, the last line shows the password for **rastogi**.
  sudo tail /etc/shadow



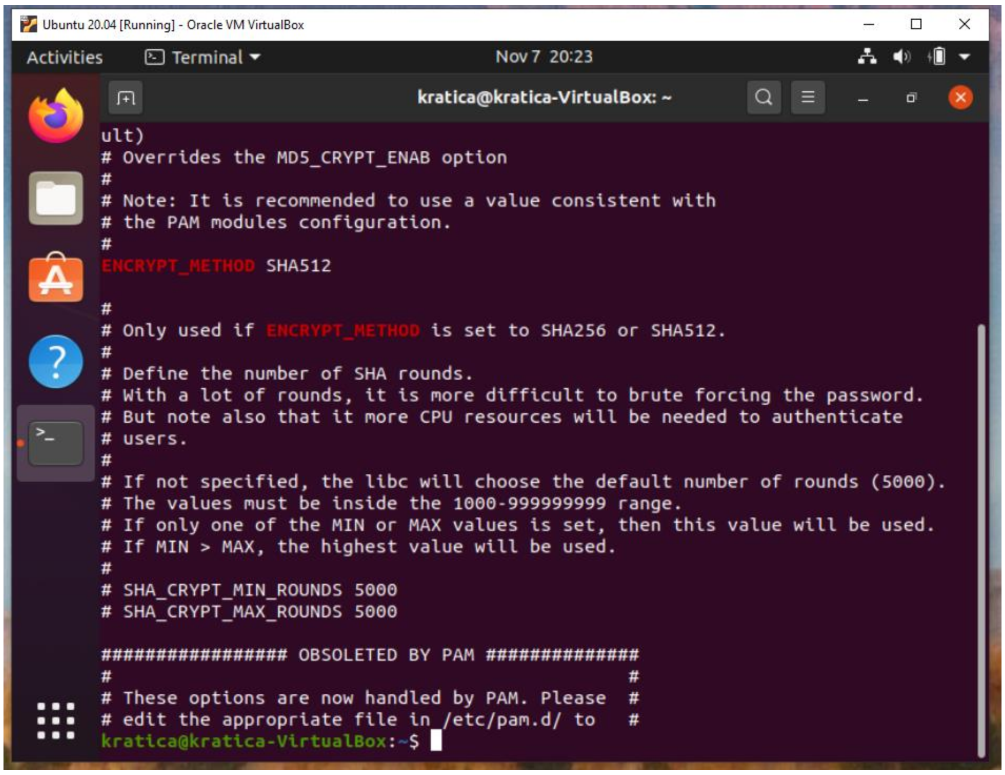
```
Is the information correct? [Y/n] y
kratica@kratica-Virtual-Machine:~$ sudo tail /etc/shadow
hplip:*:18474:0:99999:7:::
whoopsie:*:18474:0:99999:7:::
colord:*:18474:0:99999:7:::
geoclue:*:18474:0:99999:7:::
pulse:*:18474:0:99999:7:::
gnome-initial-setup:*:18474:0:99999:7:::
gdm:*:18474:0:99999:7:::
kratica:$6$WcnEESb.YnOqyelN$27otcuycdUYzJHDacGOt8S91tTZc.WugHU6t4Nf9V1AwLrSA9kbI
OFWZkEVvtgEIMFauFRVpZfjfYNec9nuOy.:18574:0:99999:7:::
systemd-coredump:!!:18574::::::
rastogi:$6$qGFBF.TwpAOAmnBN$y3OP7wvAdEfxQm1uVkqSlLJmUr7bBNmpfynmqKtVB7SXXiimE5tP
SoULPL6Zy7k6zaTjpYUSatsakW5Wua17H1:18574:0:99999:7:::
kratica@kratica-Virtual-Machine:~$ grep -A 18 ENCRYPT_METHOD /etc/login.defs
# This variable is deprecated. You should use ENCRYPT_METHOD.
#
#MD5_CRYPT_ENAB no

#
# If set to MD5 , MD5-based algorithm will be used for encrypting password
# If set to SHA256, SHA256-based algorithm will be used for encrypting password
# If set to SHA512, SHA512-based algorithm will be used for encrypting password
# If set to DES, DES-based algorithm will be used for encrypting password (defau
```

4. Finding the SALT VALUE

- The SALT value for username "rastogi" is **qGFBF.TwpAOAmnBN**

5. Understanding the Hash Algorithm

- Executed below command to see the portion of hash algorithm defined in /etc/login.defs file
  grep -A 18 ENCRYPT_METHOD /etc/login.defs



```
ult)
# Overrides the MD5_CRYPT_ENAB option
#
# Note: It is recommended to use a value consistent with
# the PAM modules configuration.
#
ENCRYPT_METHOD SHA512

#
# Only used if ENCRYPT_METHOD is set to SHA256 or SHA512.
#
# Define the number of SHA rounds.
# With a lot of rounds, it is more difficult to brute forcing the password.
# But note also that it more CPU resources will be needed to authenticate
# users.
#
# If not specified, the libc will choose the default number of rounds (5000).
# The values must be inside the 1000-999999999 range.
# If only one of the MIN or MAX values is set, then this value will be used.
# If MIN > MAX, the highest value will be used.
#
# SHA_CRYPT_MIN_ROUNDS 5000
# SHA_CRYPT_MAX_ROUNDS 5000

################# OBSOLETED BY PAM ##############
#                                               #
# These options are now handled by PAM. Please  #
# edit the appropriate file in /etc/pam.d/ to   #
kratica@kratica-VirtualBox:~$
```

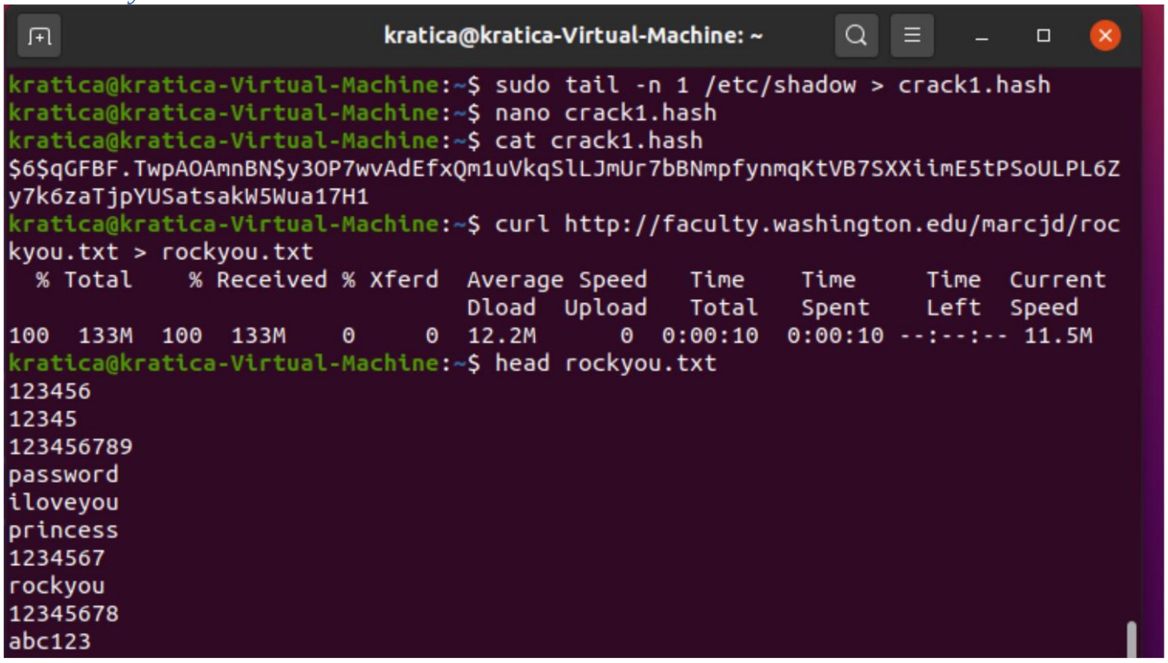
6. Making a Hash File

- Executed below command and then deleted colon and username **kratica** from crack1.hash file using nano
  tail -n 1 /etc/shadow > crack1.hash
  nano crack1.hash

7. Downloading a Wordlist

- Executed below curl command and then taking first ten passwords stored in rockyou.txt file.
  curl http://faculty.washington.edu/marcjd/rockyou.txt > rockyou.txt

  head rockyou.txt

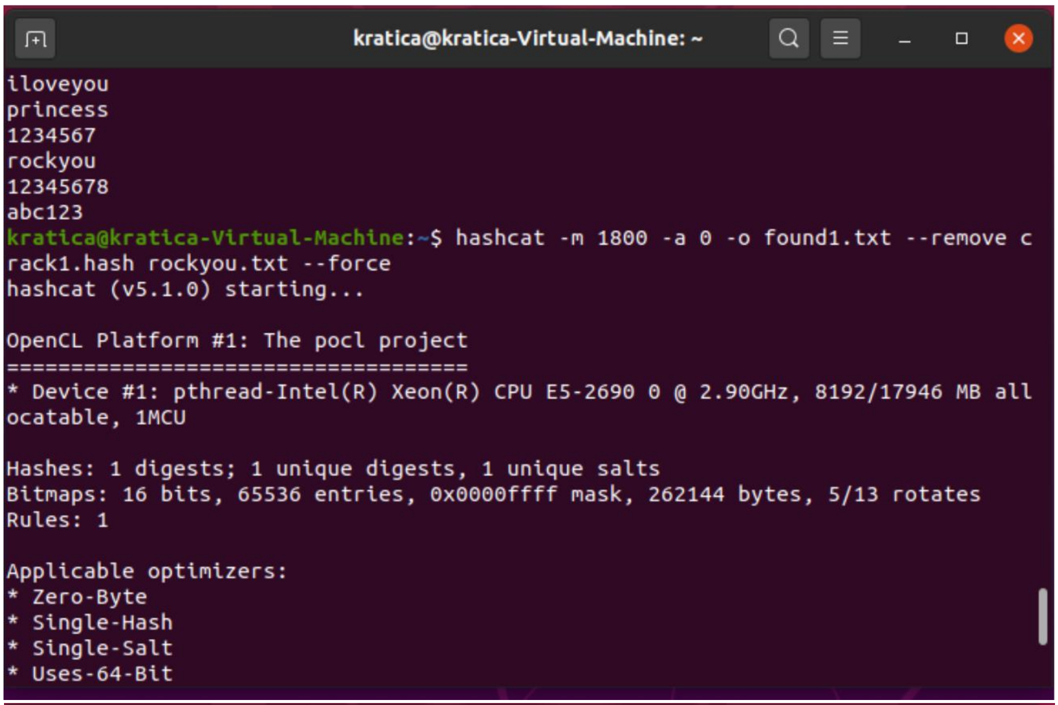


```
kratica@kratica-Virtual-Machine:~$ sudo tail -n 1 /etc/shadow > crack1.hash
kratica@kratica-Virtual-Machine:~$ nano crack1.hash
kratica@kratica-Virtual-Machine:~$ cat crack1.hash
$6$qGFBF.TwpAOAmnBN$y3OP7wvAdEfxQm1uVkqSlLJmUr7bBNmpfynmqKtVB7SXXiimE5tPSoULPL6Z
y7k6zaTjpYUSatsakW5Wua17H1
kratica@kratica-Virtual-Machine:~$ curl http://faculty.washington.edu/marcjd/roc
kyou.txt > rockyou.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  133M  100  133M    0     0  12.2M      0  0:00:10  0:00:10 --:--:-- 11.5M
kratica@kratica-Virtual-Machine:~$ head rockyou.txt
123456
12345
123456789
password
iloveyou
princess
1234567
rockyou
12345678
abc123
```

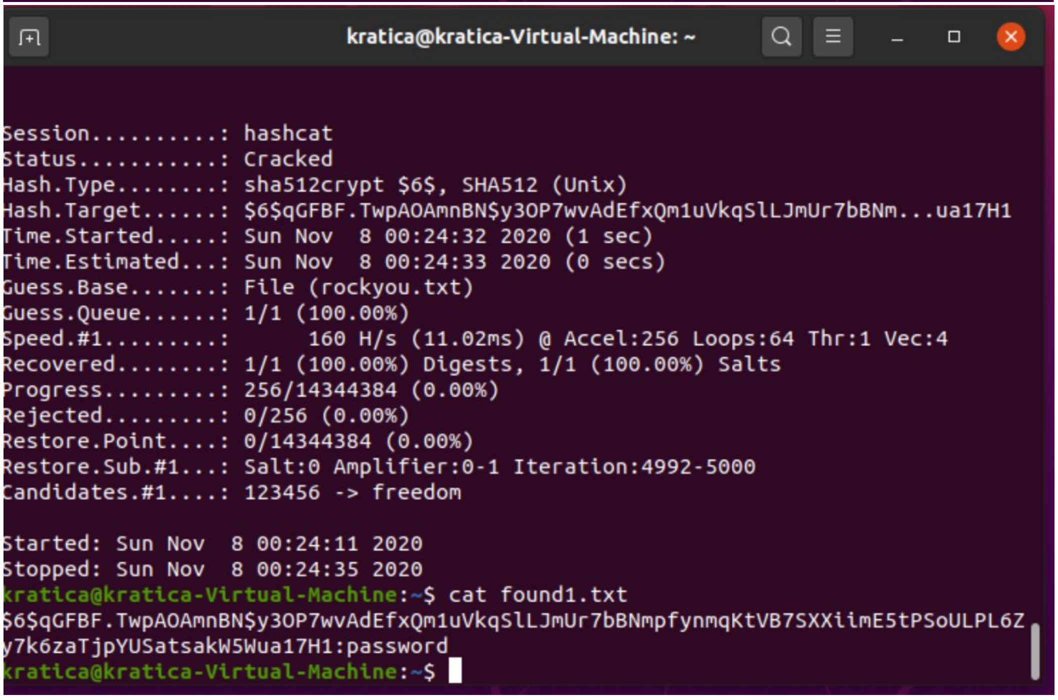
8. Cracking the Hash

- Execute below commands in terminal window:
  hashcat -m 1800 -a 0 -o found1.txt --remove crack1.hash rockyou.txt --force

  cat found1.txt



```
iloveyou
princess
1234567
rockyou
12345678
abc123
kratica@kratica-Virtual-Machine:~$ hashcat -m 1800 -a 0 -o found1.txt --remove c
rack1.hash rockyou.txt --force
hashcat (v5.1.0) starting...

OpenCL Platform #1: The pocl project
====================================
* Device #1: pthread-Intel(R) Xeon(R) CPU E5-2690 0 @ 2.90GHz, 8192/17946 MB all
ocatable, 1MCU

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Applicable optimizers:
* Zero-Byte
* Single-Hash
* Single-Salt
* Uses-64-Bit
```

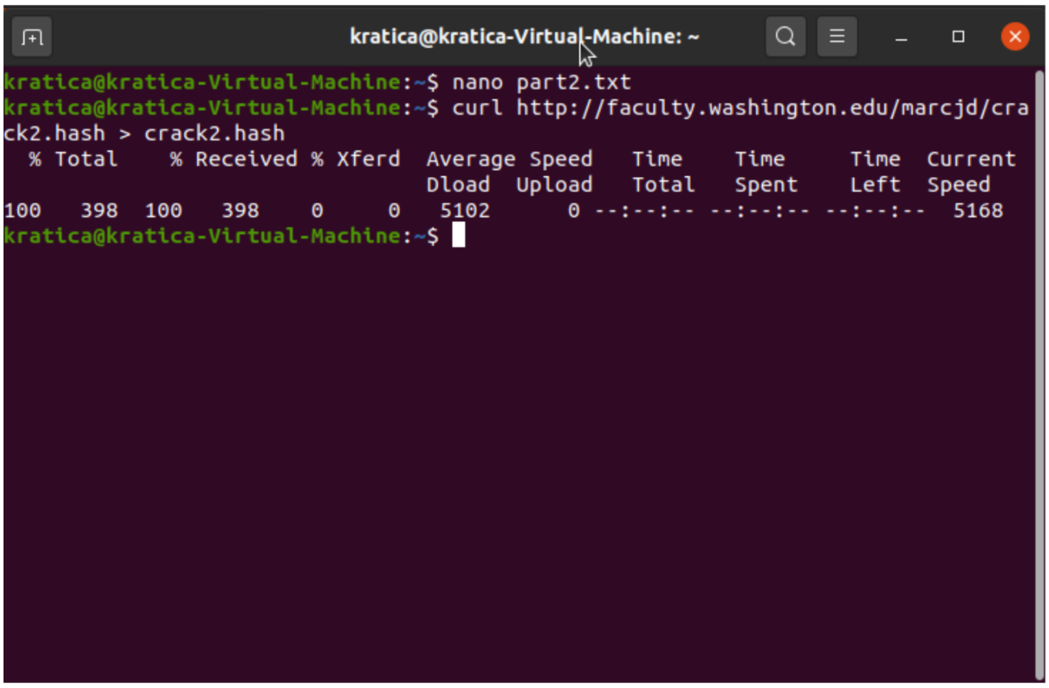


```
Session..........: hashcat
Status...........: Cracked
Hash.Type........: sha512crypt $6$, SHA512 (Unix)
Hash.Target......: $6$qGFBF.TwpAOAmnBN$y3OP7wvAdEfxQm1uVkqSlLJmUr7bBNm...ua17H1
Time.Started.....: Sun Nov  8 00:24:32 2020 (1 sec)
Time.Estimated...: Sun Nov  8 00:24:33 2020 (0 secs)
Guess.Base.......: File (rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:      160 H/s (11.02ms) @ Accel:256 Loops:64 Thr:1 Vec:4
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 256/14344384 (0.00%)
Rejected.........: 0/256 (0.00%)
Restore.Point....: 0/14344384 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:4992-5000
Candidates.#1....: 123456 -> freedom

Started: Sun Nov  8 00:24:11 2020
Stopped: Sun Nov  8 00:24:35 2020
kratica@kratica-Virtual-Machine:~$ cat found1.txt
$6$qGFBF.TwpAOAmnBN$y3OP7wvAdEfxQm1uVkqSlLJmUr7bBNmpfynmqKtVB7SXXiimE5tPSoULPL6Z
y7k6zaTjpYUSatsakW5Wua17H1:password
kratica@kratica-Virtual-Machine:~$
```

PART 2

Getting the crack.hash List

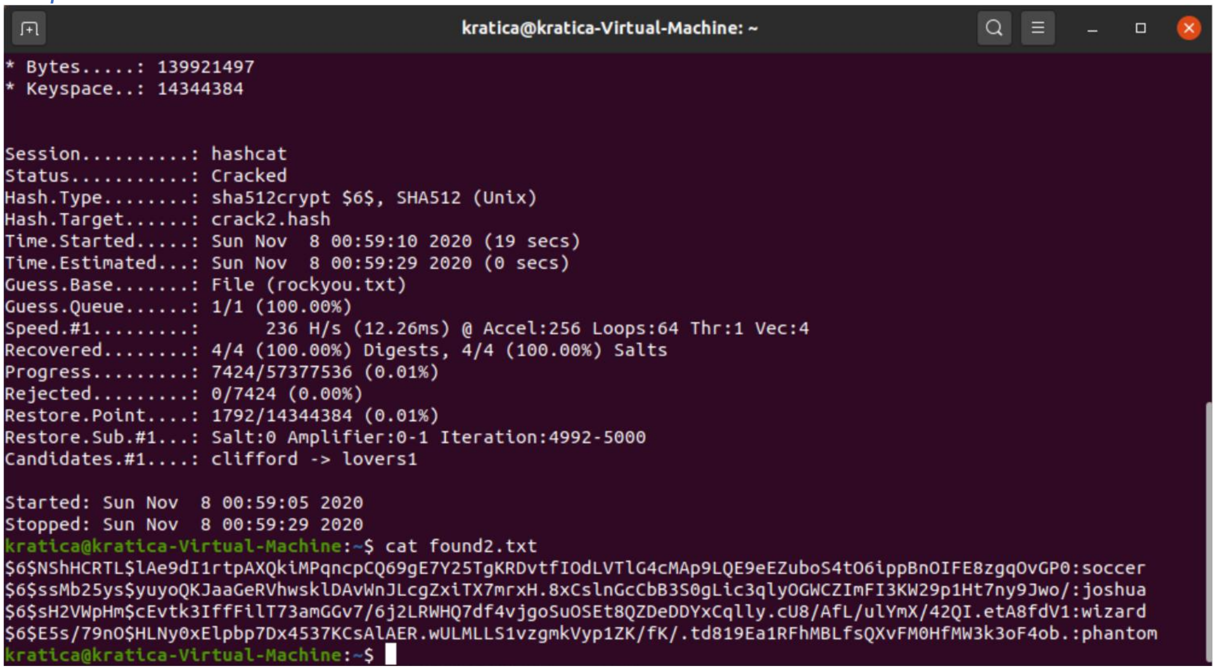
- Copied following hashes in part2.txt file using nano editor



```
kratica@kratica-Virtual-Machine:~$ nano part2.txt
kratica@kratica-Virtual-Machine:~$ curl http://faculty.washington.edu/marcjd/cra
ck2.hash > crack2.hash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   398  100   398    0     0   5102      0 --:--:-- --:--:-- --:--:--  5168
kratica@kratica-Virtual-Machine:~$ 
```

- Executed curl command as given below:
  curl http://faculty.washington.edu/marcjd/crack2.hash > crack2.hash
- Then, run below two hash commands:
  hashcat -m 1800 -a 0 -o found2.txt --remove crack2.hash rockyou.txt --force

  cat found2.txt



```
* Bytes.....: 139921497
* Keyspace..: 14344384


Session..........: hashcat
Status...........: Cracked
Hash.Type........: sha512crypt $6$, SHA512 (Unix)
Hash.Target......: crack2.hash
Time.Started.....: Sun Nov  8 00:59:10 2020 (19 secs)
Time.Estimated...: Sun Nov  8 00:59:29 2020 (0 secs)
Guess.Base.......: File (rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:      236 H/s (12.26ms) @ Accel:256 Loops:64 Thr:1 Vec:4
Recovered........: 4/4 (100.00%) Digests, 4/4 (100.00%) Salts
Progress.........: 7424/57377536 (0.01%)
Rejected.........: 0/7424 (0.00%)
Restore.Point....: 1792/14344384 (0.01%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:4992-5000
Candidates.#1....: clifford -> lovers1

Started: Sun Nov  8 00:59:05 2020
Stopped: Sun Nov  8 00:59:29 2020
kratica@kratica-Virtual-Machine:~$ cat found2.txt
$6$NShHCRTL$lAe9dI1rtpAXQkiMPqncpCQ69gE7Y25TgKRDvtfIOdLVTlG4cMAp9LQE9eEZuboS4tO6ippBnOIFE8zgqOvGP0:soccer
$6$ssMb25ys$yuyoQKJaaGeRVhwsklDAvWnJLcgZxiTX7mrxH.8xCslnGcCbB3S0gLic3qlyOGWCZImFI3KW29p1Ht7ny9Jwo/:joshua
$6$sH2VWpHm$cEvtk3IffFilT73amGGv7/6j2LRWHQ7df4vjgoSuOSEt8QZDeDDYxCqlly.cU8/AfL/ulYmX/42QI.etA8fdV1:wizard
$6$E5s/79nO$HLNy0xElpbp7Dx4537KCsAlAER.wULMLLS1vzgmkVyp1ZK/fK/.td819Ea1RFhMBLfsQXvFM0HfMW3k3oF4ob.:phantom
kratica@kratica-Virtual-Machine:~$ 
```

1. **How easy did you find it to crack passwords?**

   Answer-1:

   In this lab, I have used the username's password as a "password" which is simple and easy to detect if the user has root access. The password value can be converted to another value using the MD5 hash algorithm and is stored in the form of hashes. A plain text list of common dictionary words was obtained using the curl command. Since the user modifying the changes is the same then with the help of dictionary attack hash command hash was generated. Compare the hash result trying to crack to those of plain texts list. That's how one can easily crack a password.

2. **What challenges did you face?**

   Answer-2:

   Below are the challenges faced while doing this lab:

   i) Installing hashcat: I tried to install hashcat on ubuntu-16.04 but this version of ubuntu doesn't support hashcat binaries. So, I installed ubuntu 20.04 and with this version of ubuntu, I was able to install hashcat binaries successfully.

   ii) Identifying the hashcat mode.

   iii) Unable to execute hashcat command on the system. It may be because of less CPU.

3. **How might such an approach be employed to crack a password of an adversary?**

   Answer-3:

   Passwords are stored in hash values, called a one-way cryptosystem. These hash values can be acquired using various techniques, such as .dll injection in Windows systems or hash capture in transit, such as wireless cracking in WPA2. Once the hash value has been grabbed, then it becomes easy to find an efficient and effective way to crack the password. Hence, such an approach be employed to crack the password of an adversary.

4. **How difficult to you think this would be? Provide specific reasons why it might be difficult (or easy).**

   Answer-4:

   User-provided passwords are hashed first before they are stored in a database. Hashing is a data encryption method in one direction. An optimal methodology for hashing generates output that appears random. If the password is known then, it is easy to get the hash, but there is no clear way to get the password from the hash. An attacker must try to encrypt every possible password in order to determine the password, comparing the resulting hash with the hash that they want to crack.

5. **Do some quick research to identify other tools that can be used to crack passwords. What advantages do these other tools offer over the one used here?**

   Answer-5:

   Other tool that can be used to crack the passwords are:

   i) John the RipperGo

   ii) Pyrit

   iii) Mimikatz

   Advantages of these tools over hashcat are:

   i. John the RipperGo is useful for cracking passwords from some ZIP files for which hashcat lacks support.
   ii. Pyrit utilizes CPU capacity-sluggish calculation + slightly freezing system
   iii. Mimikatz commonly used for dumping hashes and clear text credentials directly from memory.

6. **During this research, also identify what tools allow you to create rainbow tables?**

   Answer-6:

   Rcracki_mt is a tool which allow us to create rainbow table.

7. **Finally, also find out approximately how large of a file would result for a rainbow table consisting of all possible passwords between 1 and 8 characters long, upper and lowercase, numbers, and special characters. How large of a file do you think would result for a rainbow table consisting of all possible passwords between 1 and 10 characters long?**

   Answer-7:

   Size of file with 1 to 8 characters long, upper and lowercase, numbers and special characters is below:

   Table size 4.50 TiB (4,949,612,114,592 Bytes)

   Total size 18.01 TiB (19,798,448,458,368 Bytes)

   Size of file with 1 to 10 characters long consisting of all possible password is below:

   Table size 212.19 PiB (238,909,566,292,697,696 Bytes)

Total size 848.78 PiB (955,638,265,170,790,784 Bytes)