Table of Contents:

Introduction

Passwords safeguard any system/machine/application/account from any external attacks/leaks.
In cryptanalysis and computer security, **password cracking** is the process of recovering passwords from data that has been stored in or transmitted by a computer system in scrambled form.

There are multiple methods as well as tools to crack a particular password.

Methods like:
- Brute Force Attack
- Dictionary Attack
- Password Spraying
- Credential Stuffing

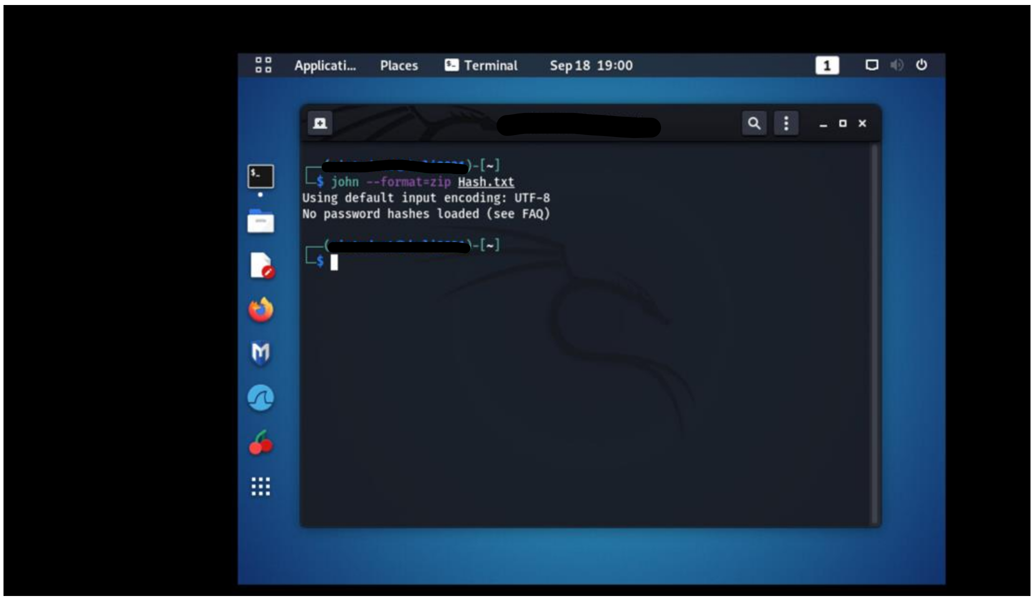
Tools like:
- Kali's in-house Tools
    - John The Ripper
    - Hashcat
    - Burp Suite
    - THC-Hydra
- Brutus
- RainbowCrack
- L0phtCrack

Tools Used

To crack the passwords, as part of the project, the below tools were used:

1. John the Ripper

- It is a well-known free open-source password-cracking tool for Linux, Unix, and Mac OS X.
- It supports a massive list of different password hash types.
- It works on the hash of the password, not the file itself.

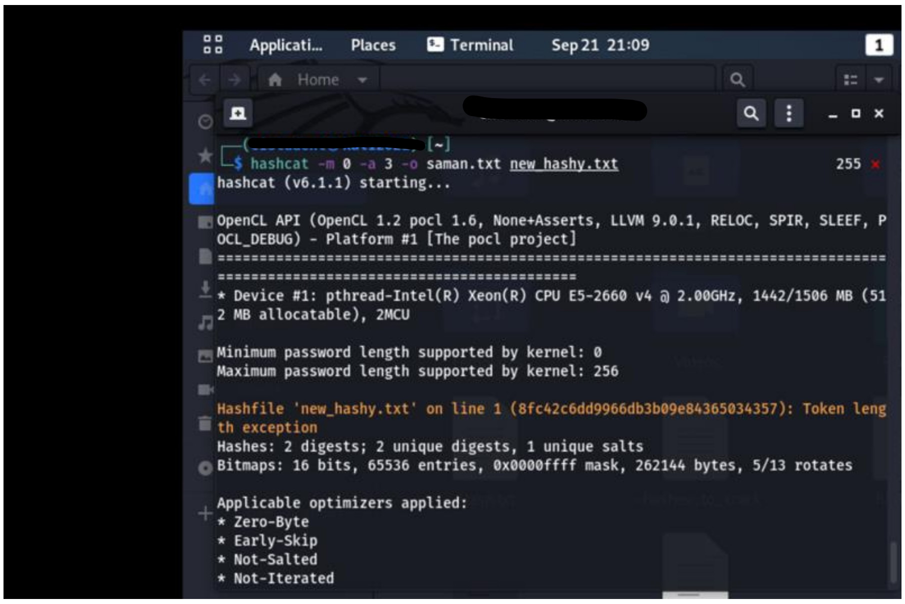


```
[redacted]
┌──([redacted])-[~]
└─$ john --format=zip Hash.txt
Using default input encoding: UTF-8
No password hashes loaded (see FAQ)

┌──([redacted])-[~]
└─$ 
```

John Ripper command

2. Hashcat

- World’s fastest password cracker.
- World’s first and only in-kernel rule engine.
- It is FREE and open-source and supports multiple OS & platforms.
- Supports several Algorithms.



```
[redacted]
┌──([redacted])-[~]
└─$ hashcat -m 0 -a 3 -o saman.txt new_hashy.txt                    255 ×
hashcat (v6.1.1) starting...

OpenCL API (OpenCL 1.2 pocl 1.6, None+Asserts, LLVM 9.0.1, RELOC, SPIR, SLEEF, P
OCL_DEBUG) - Platform #1 [The pocl project]
================================================================================
============================================
* Device #1: pthread-Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz, 1442/1506 MB (51
2 MB allocatable), 2MCU

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

Hashfile 'new_hashy.txt' on line 1 (8fc42c6dd9966db3b09e84365034357): Token leng
th exception
Hashes: 2 digests; 2 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates

Applicable optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
```

Hashcat starting

Experiment 1

Create 5-6 users and provide them passwords of different complexities and run the below attacks via **John the Ripper**:

- Brute Force Attack
- Dictionary Attack

Experiment 2

Create 5-6 users and provide them passwords of different complexities and run the below attacks via **HashCat**:

- Brute Force Attack
- Dictionary Attack

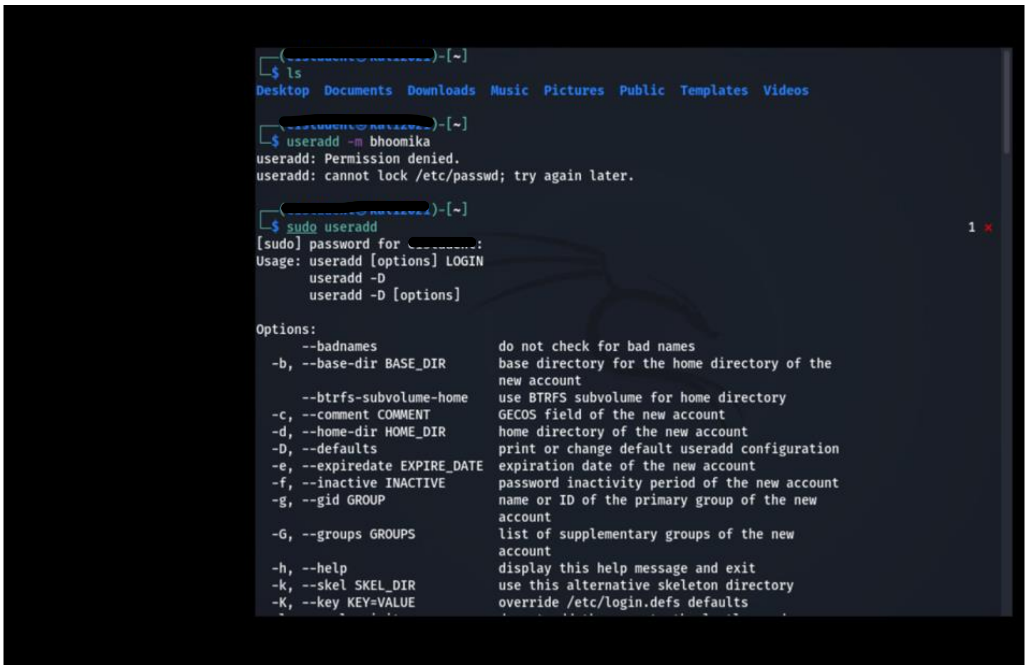
Results

As per the above experiments, the results variates between the above 2 tools.

- ### Steps for the Attacks

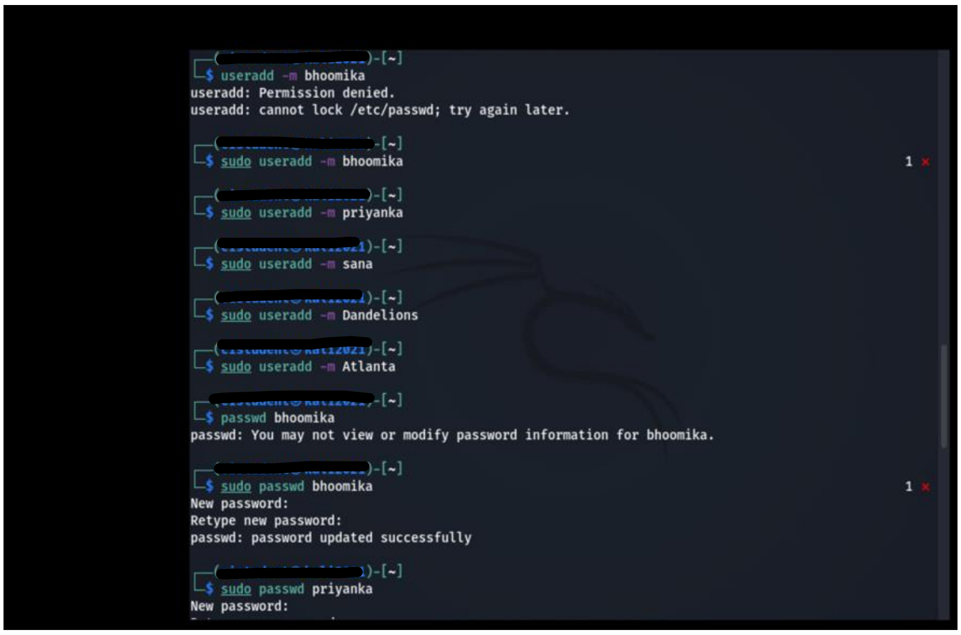
First, the Attacks were done via the Tool - **John the Ripper**:

Step 1: Create 5-6 new users and provide them passwords with different complexities:



```
┌─([redacted])-[~]
└─$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos

┌─([redacted])-[~]
└─$ useradd -m bhoomika
useradd: Permission denied.
useradd: cannot lock /etc/passwd; try again later.

┌─([redacted])-[~]
└─$ sudo useradd                                                              1 ✕
[sudo] password for [redacted]:
Usage: useradd [options] LOGIN
       useradd -D
       useradd -D [options]

Options:
      --badnames                do not check for bad names
  -b, --base-dir BASE_DIR       base directory for the home directory of the
                                new account
      --btrfs-subvolume-home    use BTRFS subvolume for home directory
  -c, --comment COMMENT         GECOS field of the new account
  -d, --home-dir HOME_DIR       home directory of the new account
  -D, --defaults                print or change default useradd configuration
  -e, --expiredate EXPIRE_DATE  expiration date of the new account
  -f, --inactive INACTIVE       password inactivity period of the new account
  -g, --gid GROUP               name or ID of the primary group of the new
                                account
  -G, --groups GROUPS           list of supplementary groups of the new
                                account
  -h, --help                    display this help message and exit
  -k, --skel SKEL_DIR           use this alternative skeleton directory
  -K, --key KEY=VALUE           override /etc/login.defs defaults
```

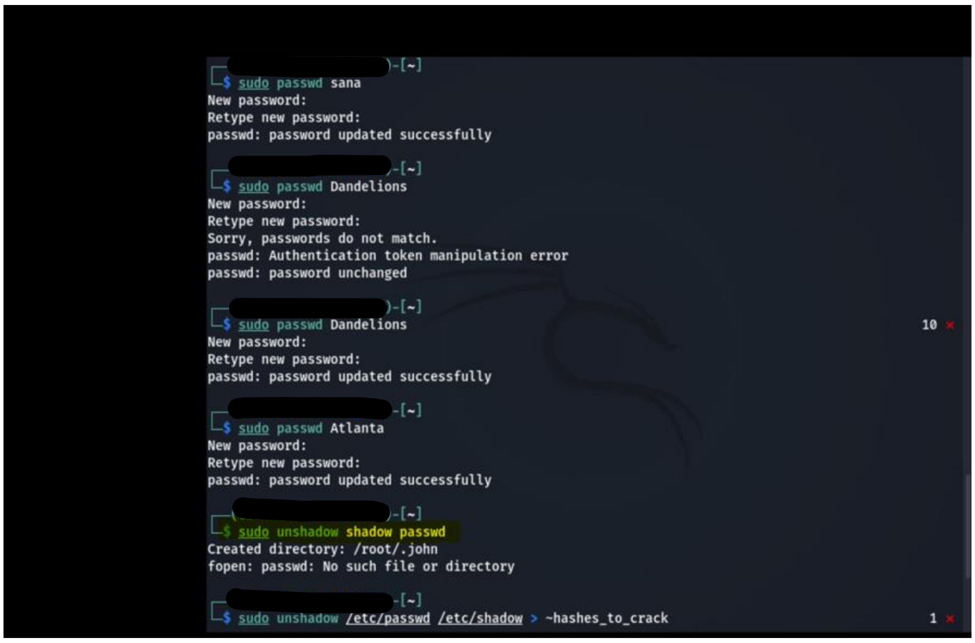
creating usernames by useradd command



```
┌──([redacted])-[~]
└─$ useradd -m bhoomika
useradd: Permission denied.
useradd: cannot lock /etc/passwd; try again later.

┌──([redacted]-[~]
└─$ sudo useradd -m bhoomika                                                        1 ×

┌──([redacted])-[~]
└─$ sudo useradd -m priyanka

┌──([redacted])-[~]
└─$ sudo useradd -m sana

┌──([redacted])-[~]
└─$ sudo useradd -m Dandelions

┌──([redacted])-[~]
└─$ sudo useradd -m Atlanta

┌──[redacted]-[~]
└─$ passwd bhoomika
passwd: You may not view or modify password information for bhoomika.

┌──([redacted])-[~]
└─$ sudo passwd bhoomika                                                            1 ×
New password:
Retype new password:
passwd: password updated successfully

┌──([redacted])-[~]
└─$ sudo passwd priyanka
New password:
```

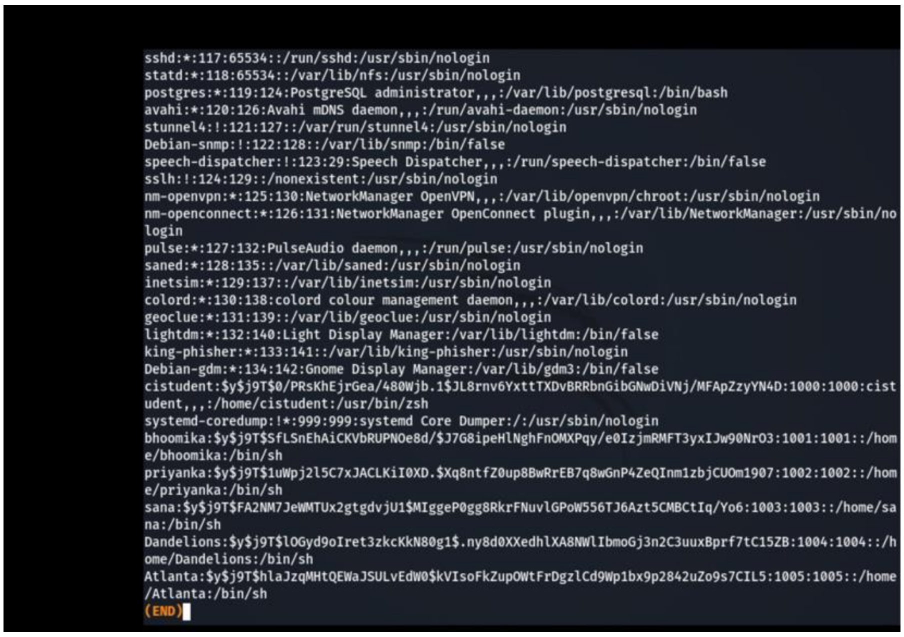
Creating passwords for the usernames created above

Step 2: Now unshadow the passwords file i.e. **passwd**



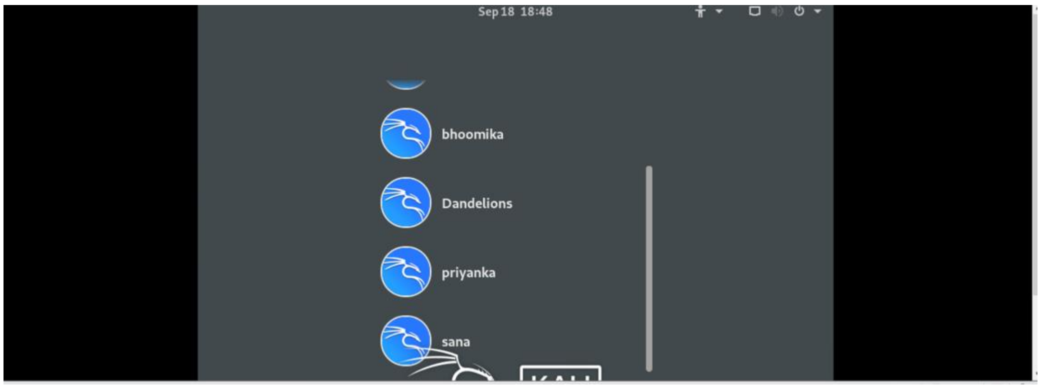
```
┌──[redacted])-[~]
└─$ sudo passwd sana
New password:
Retype new password:
passwd: password updated successfully

┌──[redacted]-[~]
└─$ sudo passwd Dandelions
New password:
Retype new password:
Sorry, passwords do not match.
passwd: Authentication token manipulation error
passwd: password unchanged

┌──[redacted])-[~]
└─$ sudo passwd Dandelions                                                          10 ×
New password:
Retype new password:
passwd: password updated successfully

┌──[redacted]-[~]
└─$ sudo passwd Atlanta
New password:
Retype new password:
passwd: password updated successfully

┌──[redacted]-[~]
└─$ sudo unshadow shadow passwd
Created directory: /root/.john
fopen: passwd: No such file or directory

┌──[redacted]-[~]
└─$ sudo unshadow /etc/passwd /etc/shadow > ~hashes_to_crack                        1 ×
```

Step 3: The new created usernames and passwords:



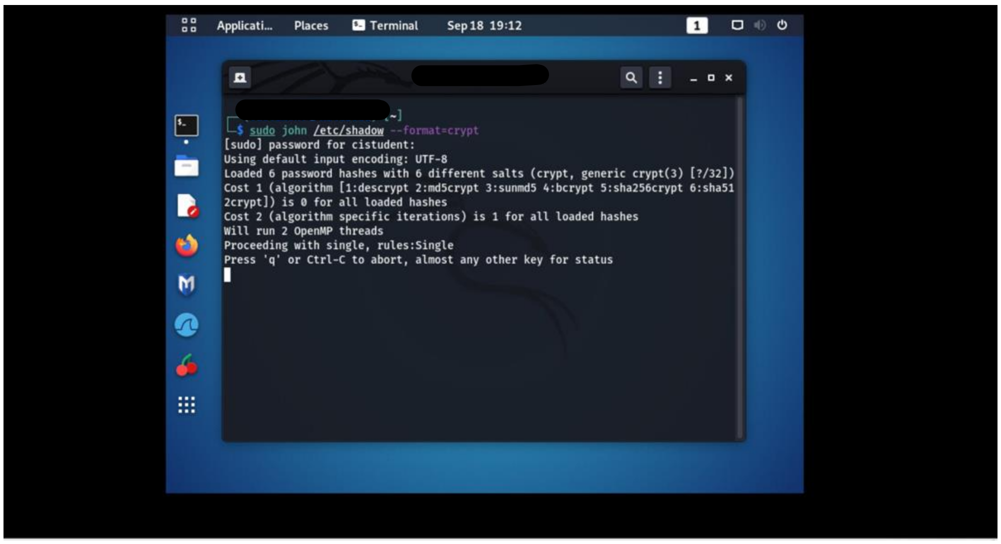
```
sshd:*:117:65534::/run/sshd:/usr/sbin/nologin
statd:*:118:65534::/var/lib/nfs:/usr/sbin/nologin
postgres:*:119:124:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
avahi:*:120:126:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin
stunnel4:!:121:127::/var/run/stunnel4:/usr/sbin/nologin
Debian-snmp:!:122:128::/var/lib/snmp:/bin/false
speech-dispatcher:!:123:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
sslh:!:124:129::/nonexistent:/usr/sbin/nologin
nm-openvpn:*:125:130:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
nm-openconnect:*:126:131:NetworkManager OpenConnect plugin,,,:/var/lib/NetworkManager:/usr/sbin/no
login
pulse:*:127:132:PulseAudio daemon,,,:/run/pulse:/usr/sbin/nologin
saned:*:128:135::/var/lib/saned:/usr/sbin/nologin
inetsim:*:129:137::/var/lib/inetsim:/usr/sbin/nologin
colord:*:130:138:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
geoclue:*:131:139::/var/lib/geoclue:/usr/sbin/nologin
lightdm:*:132:140:Light Display Manager:/var/lib/lightdm:/bin/false
king-phisher:*:133:141::/var/lib/king-phisher:/usr/sbin/nologin
Debian-gdm:*:134:142:Gnome Display Manager:/var/lib/gdm3:/bin/false
cistudent:$y$j9T$0/PRsKhEjrGea/480Wjb.1$JL8rnv6YxttTXDvBRRbnGibGNwDiVNj/MFApZzyYN4D:1000:1000:cist
udent,,,:/home/cistudent:/usr/bin/zsh
systemd-coredump:!*:999:999:systemd Core Dumper:/:/usr/sbin/nologin
bhoomika:$y$j9T$SfLSnEhAiCKVbRUPNOe8d/$J7G8ipeHlNghFnOMXPqy/e0IzjmRMFT3yxIJw90NrO3:1001:1001::/hom
e/bhoomika:/bin/sh
priyanka:$y$j9T$1uWpj2l5C7xJACLKiI0XD.$Xq8ntfZ0up8BwRrEB7q8wGnP4ZeQInm1zbjCUOm1907:1002:1002::/hom
e/priyanka:/bin/sh
sana:$y$j9T$FA2NM7JeWMTUx2gtgdvjU1$MIggeP0gg8RkrFNuvlGPoW556TJ6Azt5CMBCtIq/Yo6:1003:1003::/home/sa
na:/bin/sh
Dandelions:$y$j9T$lOGyd9oIret3zkcKkN80g1$.ny8d0XXedhlXA8NWlIbmoGj3n2C3uuxBprf7tC15ZB:1004:1004::/h
ome/Dandelions:/bin/sh
Atlanta:$y$j9T$hlaJzqMHtQEWaJSULvEdW0$kVIsoFkZupOWtFrDgzlCd9Wp1bx9p2842uZo9s7CIL5:1005:1005::/home
/Atlanta:/bin/sh
(END)
```

new usernames added to existing hash file



List of new users created

Step 4: Execute the **Brute Force Attack**:

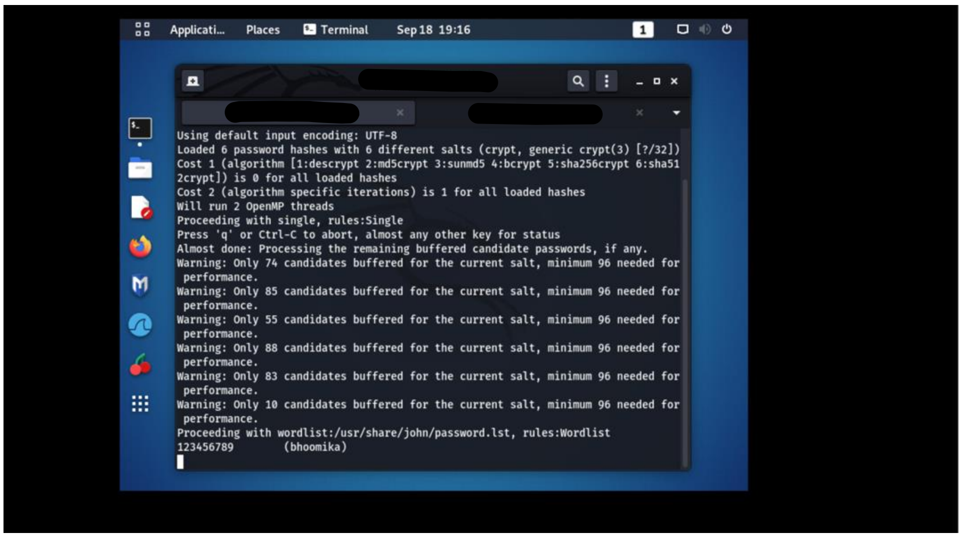


[redacted]

```
[redacted]~]
└─$ sudo john /etc/shadow --format=crypt
[sudo] password for cistudent:
Using default input encoding: UTF-8
Loaded 6 password hashes with 6 different salts (crypt, generic crypt(3) [?/32])
Cost 1 (algorithm [1:descrypt 2:md5crypt 3:sunmd5 4:bcrypt 5:sha256crypt 6:sha51
2crypt]) 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
```

Brute Force command

Step 5: The command got executed on **18th September 2022** at **11 PM EST** and the first password was cracked in **10 minutes**:



[redacted]

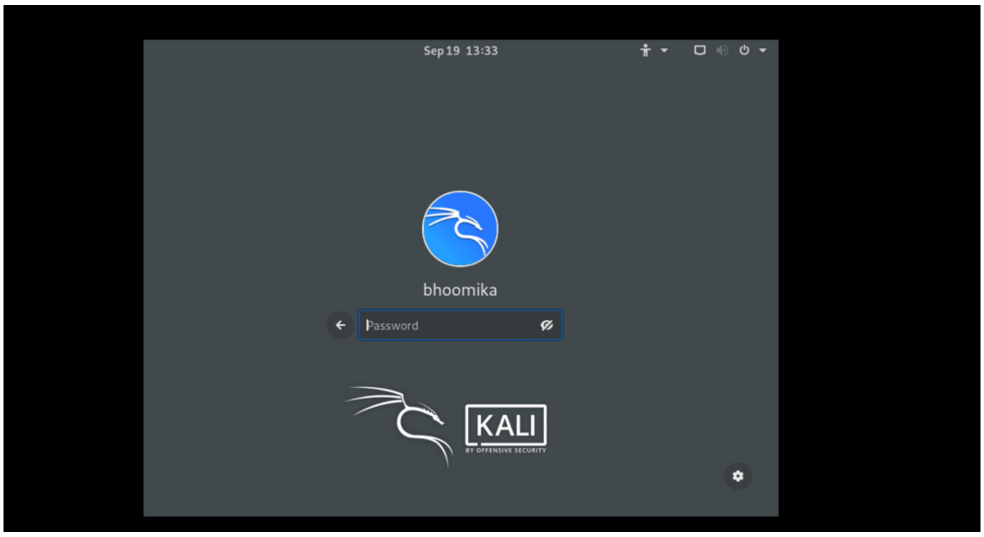
[redacted]

[redacted]

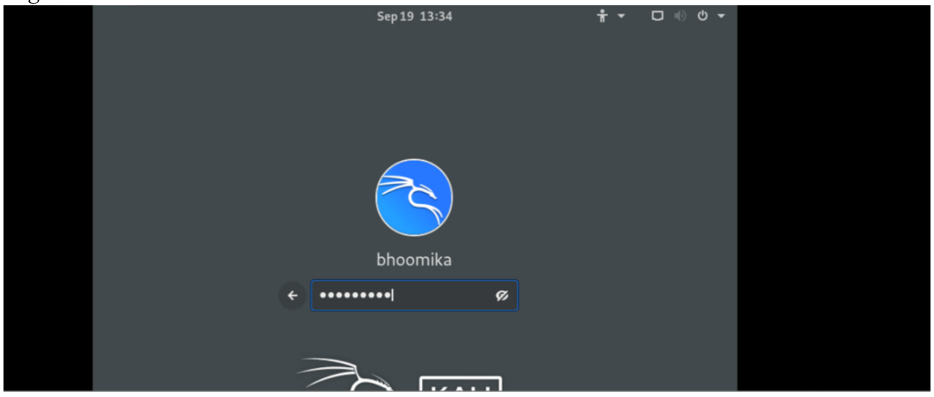
```
Using default input encoding: UTF-8
Loaded 6 password hashes with 6 different salts (crypt, generic crypt(3) [?/32])
Cost 1 (algorithm [1:descrypt 2:md5crypt 3:sunmd5 4:bcrypt 5:sha256crypt 6:sha51
2crypt]) is 0 for all loaded hashes
Cost 2 (algorithm specific iterations) is 1 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 74 candidates buffered for the current salt, minimum 96 needed for
 performance.
Warning: Only 85 candidates buffered for the current salt, minimum 96 needed for
 performance.
Warning: Only 55 candidates buffered for the current salt, minimum 96 needed for
 performance.
Warning: Only 88 candidates buffered for the current salt, minimum 96 needed for
 performance.
Warning: Only 83 candidates buffered for the current salt, minimum 96 needed for
 performance.
Warning: Only 10 candidates buffered for the current salt, minimum 96 needed for
 performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
123456789        (bhoomika)
```

The first password cracked

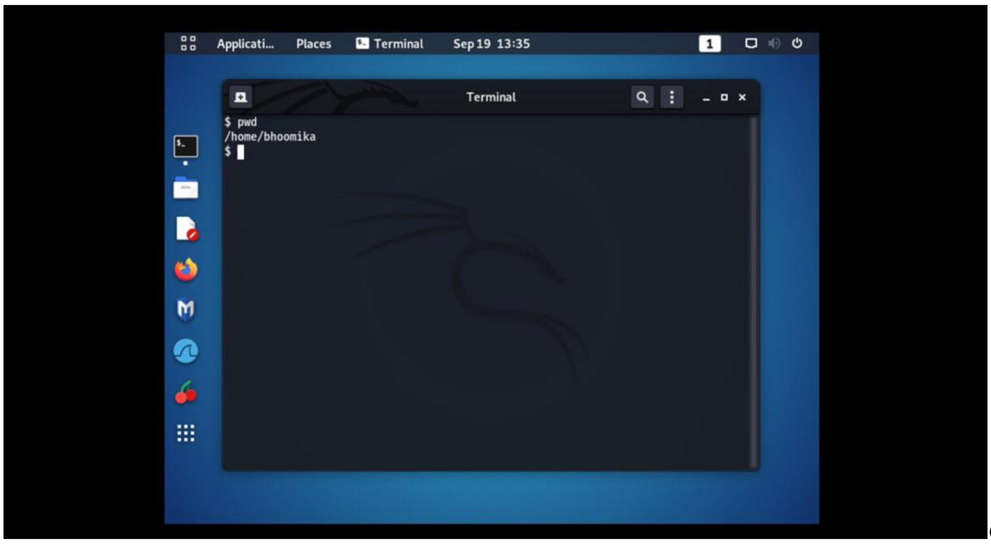
Step 6: Accessing the cracked password user **"bhoomika"** via Switch User:



Login in the user "bhoomika"

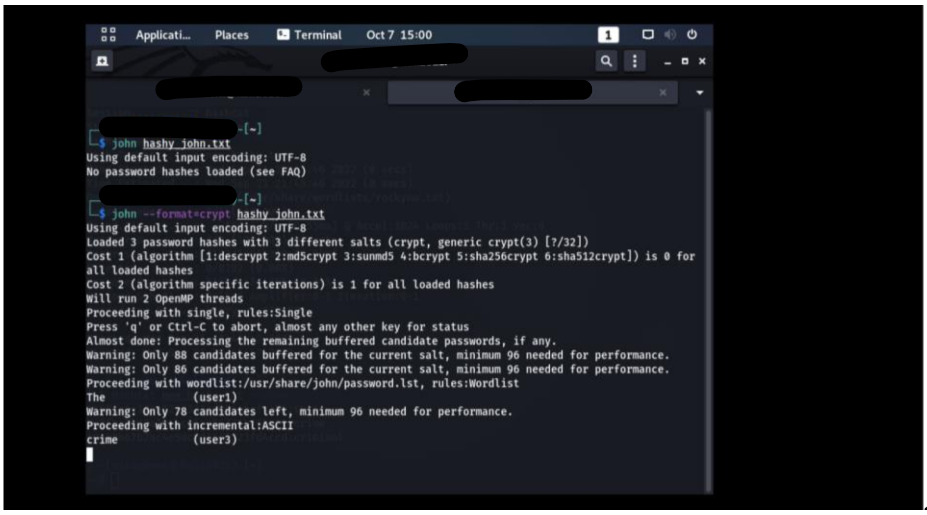


Provide the cracked password


```
$ pwd
/home/bhoomika
$ 
```

*Check where am I using **pwd** command*

Step 7: For 2 more users the task worked but did not stop and kept on running till date:


[redacted] [redacted] [redacted]

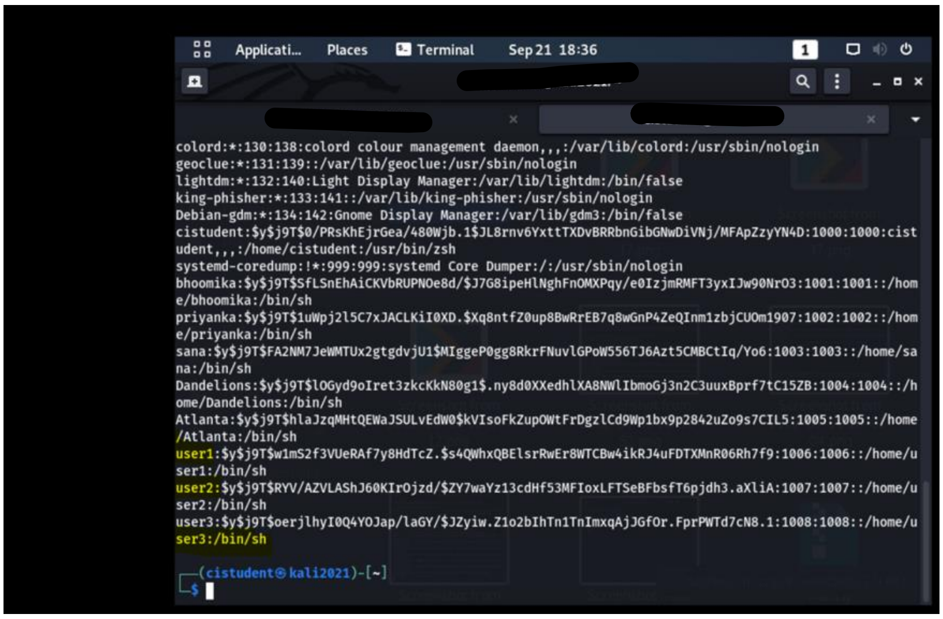
```
[redacted]-[~]
└─$ john hashy_john.txt
Using default input encoding: UTF-8
No password hashes loaded (see FAQ)

[redacted]-[~]
└─$ john --format=crypt hashy_john.txt
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (crypt, generic crypt(3) [?/32])
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
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 88 candidates buffered for the current salt, minimum 96 needed for performance.
Warning: Only 86 candidates buffered for the current salt, minimum 96 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
The              (user1)
Warning: Only 78 candidates left, minimum 96 needed for performance.
Proceeding with incremental:ASCII
crime            (user3)
```

2 more passwords cracked

Second, a **Dictionary Attack** was done using **John the Ripper**:

Steps for the Attack:

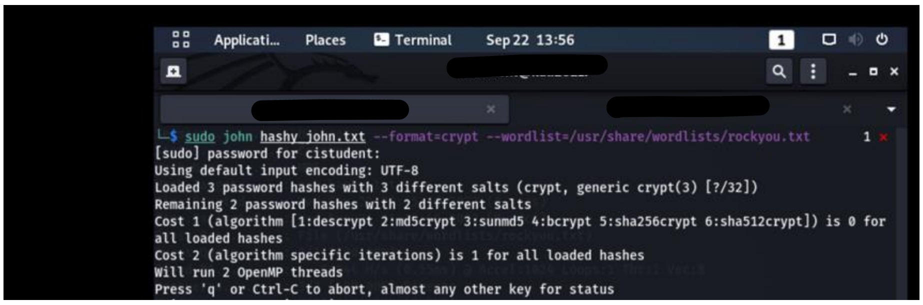
Step 1: To the existing hash file (with new users created above), 3 new users were added:



[redacted]

```
colord:*:130:138:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
geoclue:*:131:139::/var/lib/geoclue:/usr/sbin/nologin
lightdm:*:132:140:Light Display Manager:/var/lib/lightdm:/bin/false
king-phisher:*:133:141::/var/lib/king-phisher:/usr/sbin/nologin
Debian-gdm:*:134:142:Gnome Display Manager:/var/lib/gdm3:/bin/false
cistudent:$y$j9T$0/PRsKhEjrGea/480Wjb.1$JL8rnv6YxttTXDvBRRbnGibGNwDiVNj/MFApZzyYN4D:1000:1000:cist
udent,,,:/home/cistudent:/usr/bin/zsh
systemd-coredump:!*:999:999:systemd Core Dumper:/:/usr/sbin/nologin
bhoomika:$y$j9T$SfLSnEhAiCKVbRUPNOe8d/$J7G8ipeHlNghFnOMXPqy/e0IzjmRMFT3yxIJw90NrO3:1001:1001::/hom
e/bhoomika:/bin/sh
priyanka:$y$j9T$1uWpj2l5C7xJACLKiI0XD.$Xq8ntfZ0up8BwRrEB7q8wGnP4ZeQInm1zbjCUOm1907:1002:1002::/hom
e/priyanka:/bin/sh
sana:$y$j9T$FA2NM7JeWMTUx2gtgdvjU1$MIggeP0gg8RkrFNuvlGPoW556TJ6Azt5CMBCtIq/Yo6:1003:1003::/home/sa
na:/bin/sh
Dandelions:$y$j9T$lOGyd9oIret3zkcKkN80g1$.ny8d0XXedhlXA8NWlIbmoGj3n2C3uuxBprf7tC15ZB:1004:1004::/h
ome/Dandelions:/bin/sh
Atlanta:$y$j9T$hlaJzqMHtQEWaJSULvEdW0$kVIsoFkZupOWtFrDgzlCd9Wp1bx9p2842uZo9s7CIL5:1005:1005::/home
/Atlanta:/bin/sh
user1:$y$j9T$w1mS2f3VUeRAf7y8HdTcZ.$s4QWhxQBElsrRwEr8WTCBw4ikRJ4uFDTXMnR06Rh7f9:1006:1006::/home/u
ser1:/bin/sh
user2:$y$j9T$RYV/AZVLAShJ60KIrOjzd/$ZY7waYz13cdHf53MFIoxLFTSeBFbsfT6pjdh3.aXliA:1007:1007::/home/u
ser2:/bin/sh
user3:$y$j9T$oerjlhyI0Q4YOJap/laGY/$JZyiw.Z1o2bIhTn1TnImxqAjJGfOr.FprPWTd7cN8.1:1008:1008::/home/u
ser3:/bin/sh

┌──(cistudent㉿kali2021)-[~]
└─$ 
```

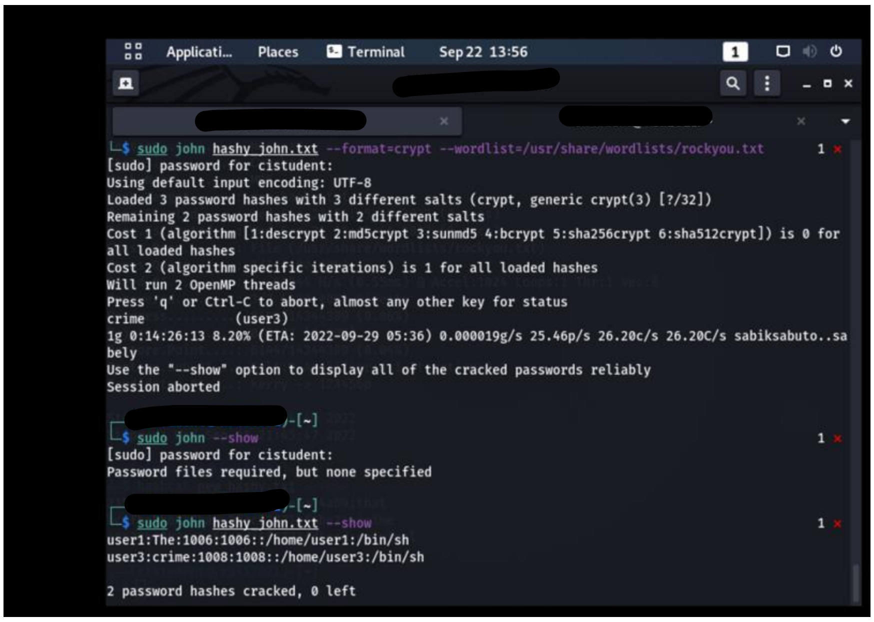
Step 2: Since many password lists exist in the system but we will work on the largest one – **“rockyou.txt”** and run the dictionary attack, the attack started on **19th September 2022** at **11 AM EST**:



[redacted]

```
└─$ sudo john hashy_john.txt --format=crypt --wordlist=/usr/share/wordlists/rockyou.txt        1 ×
[sudo] password for cistudent:
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (crypt, generic crypt(3) [?/32])
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
```

Step 3: The attack got completed on **22nd September 2022** at **5:56 PM EST**:



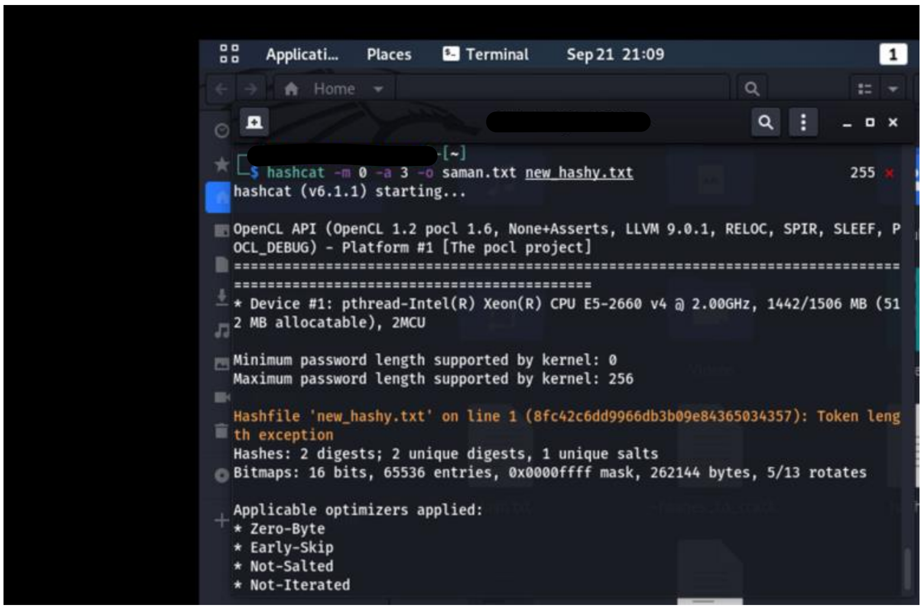
```
[redacted]
└─$ sudo john hashy_john.txt --format=crypt --wordlist=/usr/share/wordlists/rockyou.txt        1 ×
[sudo] password for cistudent:
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (crypt, generic crypt(3) [?/32])
Remaining 2 password hashes with 2 different 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
crime            (user3)
1g 0:14:26:13 8.20% (ETA: 2022-09-29 05:36) 0.000019g/s 25.46p/s 26.20c/s 26.20C/s sabiksabuto..sa
bely
Use the "--show" option to display all of the cracked passwords reliably
Session aborted

┌[redacted]-[~]
└─$ sudo john --show                                                                              1 ×
[sudo] password for cistudent:
Password files required, but none specified

┌[redacted]-[~]
└─$ sudo john hashy_john.txt --show                                                               1 ×
user1:The:1006:1006::/home/user1:/bin/sh
user3:crime:1008:1008::/home/user3:/bin/sh

2 password hashes cracked, 0 left
```

Use the –show command to view the cracked passwords

Second, the Attacks were done via the Tool - **Hashcat**:

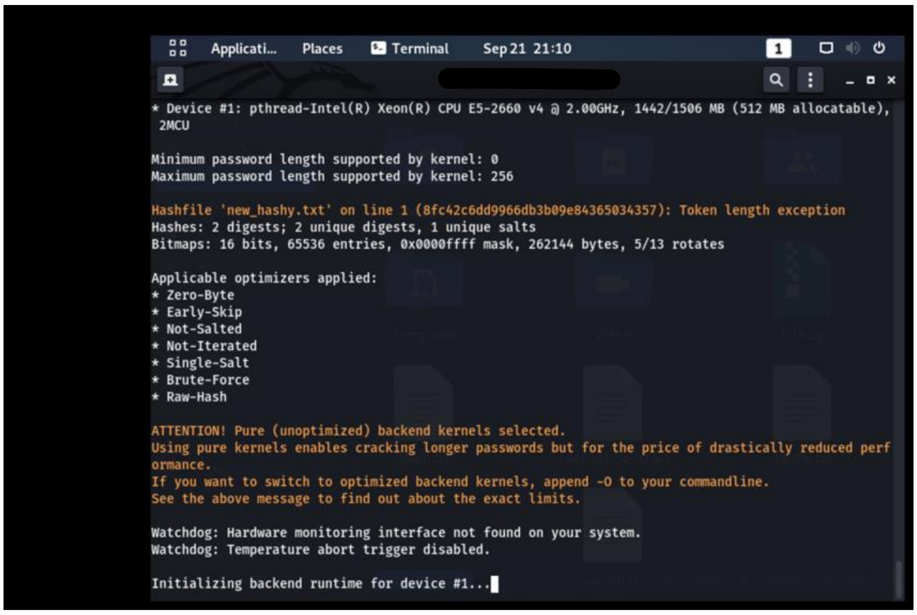
Note: The Task was executed on **22nd September 2022** at **1 AM EST:**

Step 1: Run the **Brute Force Attack** on the related users and passwords:



```
┌[redacted]-[~]
└─$ hashcat -m 0 -a 3 -o saman.txt new_hashy.txt                                  255 ×
hashcat (v6.1.1) starting...

OpenCL API (OpenCL 1.2 pocl 1.6, None+Asserts, LLVM 9.0.1, RELOC, SPIR, SLEEF, P
OCL_DEBUG) - Platform #1 [The pocl project]
================================================================================
===========================================
* Device #1: pthread-Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz, 1442/1506 MB (51
2 MB allocatable), 2MCU

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

Hashfile 'new_hashy.txt' on line 1 (8fc42c6dd9966db3b09e84365034357): Token leng
th exception
Hashes: 2 digests; 2 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates

Applicable optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
```

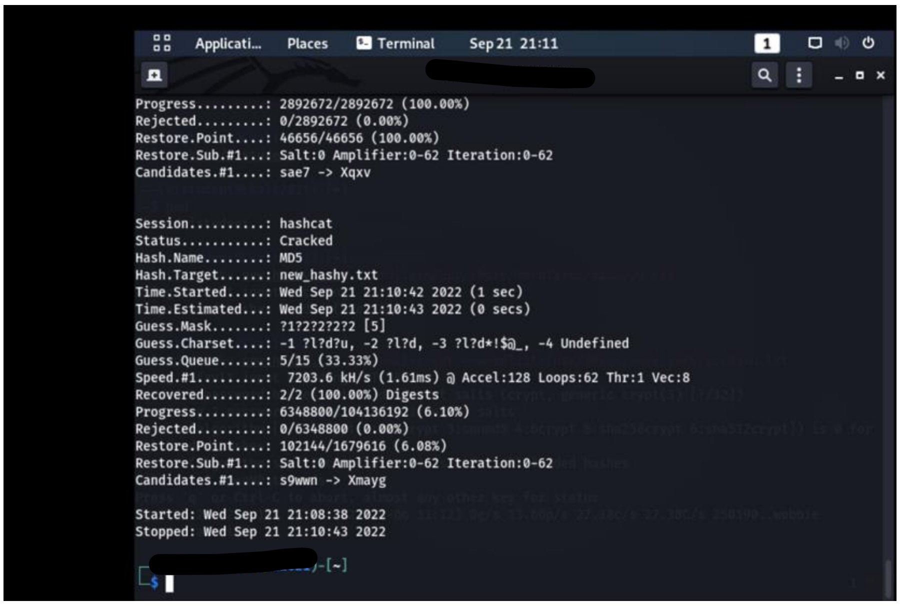
*Brute force attack is identified as **-a 3***



[redacted]

```
* Device #1: pthread-Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz, 1442/1506 MB (512 MB allocatable),
 2MCU

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

Hashfile 'new_hashy.txt' on line 1 (8fc42c6dd9966db3b09e84365034357): Token length exception
Hashes: 2 digests; 2 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates

Applicable optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Salt
* Brute-Force
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Using pure kernels enables cracking longer passwords but for the price of drastically reduced perf
ormance.
If you want to switch to optimized backend kernels, append -O to your commandline.
See the above message to find out about the exact limits.

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

Initializing backend runtime for device #1...
```

Password Cracking in progress

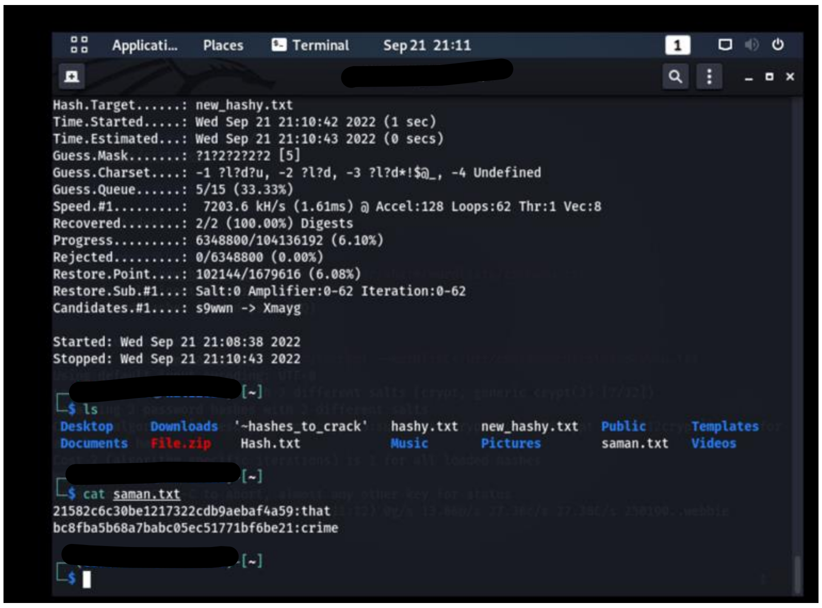


[redacted]

```
Progress.........: 2892672/2892672 (100.00%)
Rejected.........: 0/2892672 (0.00%)
Restore.Point....: 46656/46656 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-62 Iteration:0-62
Candidates.#1....: sae7 -> Xqxv

Session..........: hashcat
Status...........: Cracked
Hash.Name........: MD5
Hash.Target......: new_hashy.txt
Time.Started.....: Wed Sep 21 21:10:42 2022 (1 sec)
Time.Estimated...: Wed Sep 21 21:10:43 2022 (0 secs)
Guess.Mask.......: ?1?2?2?2?2 [5]
Guess.Charset....: -1 ?l?d?u, -2 ?l?d, -3 ?l?d*!$@_, -4 Undefined
Guess.Queue......: 5/15 (33.33%)
Speed.#1.........:  7203.6 kH/s (1.61ms) @ Accel:128 Loops:62 Thr:1 Vec:8
Recovered........: 2/2 (100.00%) Digests
Progress.........: 6348800/104136192 (6.10%)
Rejected.........: 0/6348800 (0.00%)
Restore.Point....: 102144/1679616 (6.08%)
Restore.Sub.#1...: Salt:0 Amplifier:0-62 Iteration:0-62
Candidates.#1....: s9wwn -> Xmayg

Started: Wed Sep 21 21:08:38 2022
Stopped: Wed Sep 21 21:10:43 2022

┌[redacted])-[~]
└$
```

Password successfully cracked

Step 2: The cracked passwords are stored in the text file called **saman.txt**:



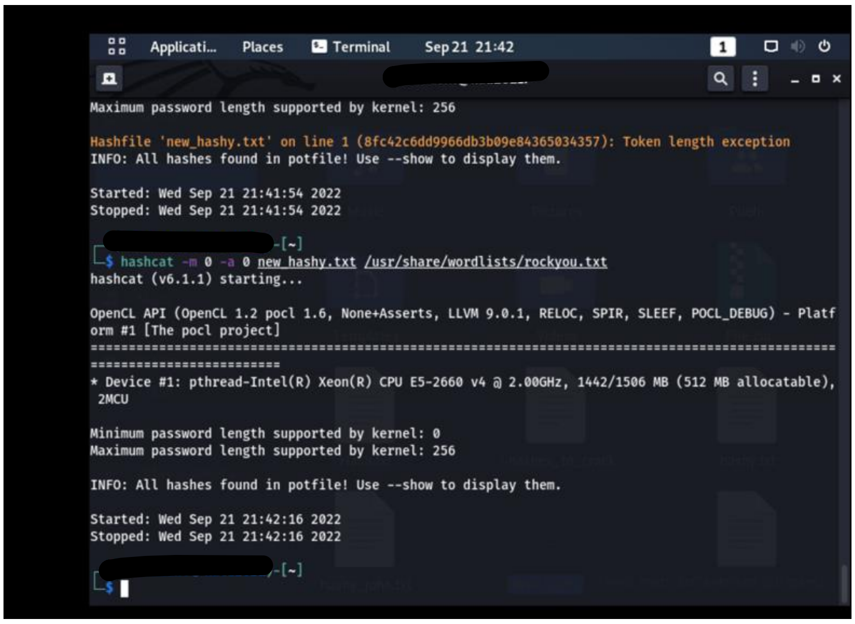
[redacted]

```
Hash.Target......: new_hashy.txt
Time.Started.....: Wed Sep 21 21:10:42 2022 (1 sec)
Time.Estimated...: Wed Sep 21 21:10:43 2022 (0 secs)
Guess.Mask.......: ?1?2?2?2?2 [5]
Guess.Charset....: -1 ?l?d?u, -2 ?l?d, -3 ?l?d*!$@_, -4 Undefined
Guess.Queue......: 5/15 (33.33%)
Speed.#1.........:  7203.6 kH/s (1.61ms) @ Accel:128 Loops:62 Thr:1 Vec:8
Recovered........: 2/2 (100.00%) Digests
Progress.........: 6348800/104136192 (6.10%)
Rejected.........: 0/6348800 (0.00%)
Restore.Point....: 102144/1679616 (6.08%)
Restore.Sub.#1...: Salt:0 Amplifier:0-62 Iteration:0-62
Candidates.#1....: s9wwn -> Xmayg

Started: Wed Sep 21 21:08:38 2022
Stopped: Wed Sep 21 21:10:43 2022

[redacted][~]
└─$ ls
Desktop    Downloads  '~hashes_to_crack'  hashy.txt  new_hashy.txt  Public     Templates
Documents  File.zip    Hash.txt           Music      Pictures       saman.txt  Videos

[redacted][~]
└─$ cat saman.txt
21582c6c30be1217322cdb9aebaf4a59:that
bc8fba5b68a7babc05ec51771bf6be21:crime

[redacted][~]
└─$ 
```

*--**show** command is run and the cracked passwords are stored in **saman.txt** (given in the initial command)*

Next, we will create a few more users (since the tool does not crack already cracked passwords) and give them passwords with different complexities and run the tool via **Dictionary Attack**:

Step 1: The same command is run but for Dictionary Attack the number is **-a 0**:



[redacted]

```
Maximum password length supported by kernel: 256

Hashfile 'new_hashy.txt' on line 1 (8fc42c6dd9966db3b09e84365034357): Token length exception
INFO: All hashes found in potfile! Use --show to display them.

Started: Wed Sep 21 21:41:54 2022
Stopped: Wed Sep 21 21:41:54 2022

[redacted]-[~]
└─$ hashcat -m 0 -a 0 new_hashy.txt /usr/share/wordlists/rockyou.txt
hashcat (v6.1.1) starting...

OpenCL API (OpenCL 1.2 pocl 1.6, None+Asserts, LLVM 9.0.1, RELOC, SPIR, SLEEF, POCL_DEBUG) - Platf
orm #1 [The pocl project]
==================================================================================================
=========================
* Device #1: pthread-Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz, 1442/1506 MB (512 MB allocatable),
 2MCU

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

INFO: All hashes found in potfile! Use --show to display them.

Started: Wed Sep 21 21:42:16 2022
Stopped: Wed Sep 21 21:42:16 2022

[redacted]-[~]
└─$ 
```

Note: The Attack was run on 22nd September 2022 at 1:42 AM EST and it was successful in 14 hours.

- ### Success/Failure

  Both the Tools and Methods proved to be working and none of them failed.
  Dictionary attacks were successful for both tools but Brute Force was only successful for Hashcat, for John the task is still running.

- ### Time Taken

  The below timestamps were recorded through the Project.

  Time taken by:

  - Hashcat (Dictionary) = 15 minutes
  - Hashcat (Brute Force) = 20 minutes

  - John (Dictionary) = 14 hours
  - John (Brute Force) = infinite (still running)

- ### Which Tool/Design/Method worked Better?

  As per my understanding, **Hashcat** Tool worked much better as compared to John because of:

  1. Less time is taken to crack the passwords.
  2. Lesser complexity involved to crack the passwords.

My Reflection on the Project

This was a good exercise for me to understand below things:

1. Different users have varied lengths of passwords.
2. Cracking every password is not easy/difficult.
3. There are zillions of tools to be learned and played with.
4. Patience! This leads to the cracking of passwords.

As part of the curriculum, understanding the complexity of different passwords, tagging them as weak or strong, and seeing how actually the tool and algorithm behind its work.
As part of the experiments done, Brute Force vs Dictionary Attacks - a lot came out in the picture while cracking the same passwords via different tools.

Dictionary Attack actually compares the hashes with a list (many) of already compromised passwords and runs through it and fetched the user's password whereas Brute Force runs an algorithm to search the password and is time-consuming.

This activity really made me understand the passwords and the importance of creating a password that none of these attacks could break (takes rather a lifetime).

Thank you Professor for letting me do this rigorous yet fun activity from which I have learned a lot.