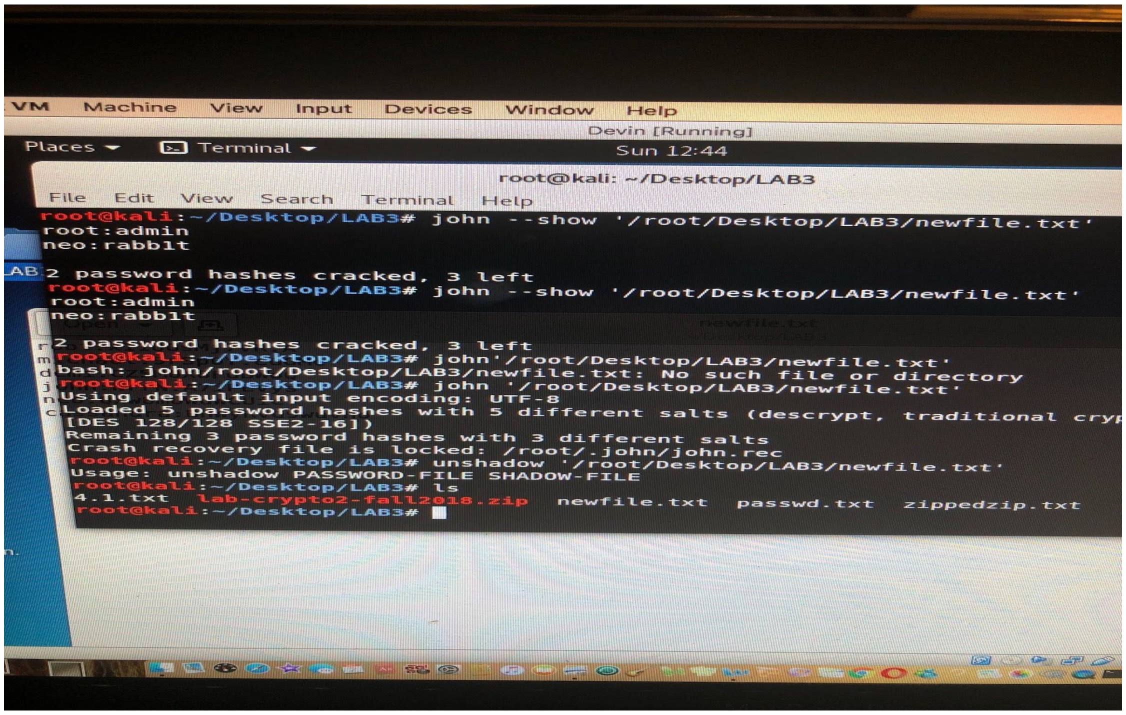
CSCI 400-01 Cryptography Lab Topic: Hashes, Stashes, and Rainbows

Roles: Devin Polichetti 4.1, Andy Bui 4.2, Michael Gonzalez 4.3, Michael Gonzalez 4.4.1, Mohammed Mamun/Michael Gonzalez 4.4.3, 4.5.1, Michael Gonzalez 5

4.1

Finding the Unix passwords that generated each of the following Unix crypt was at first tough to start due to which tool would work best. I ended up using John the Ripper since it performed best to crack the Unix hash entries for me compared to the other tools available. I began with setting up root privileges to be able to read the /etc/passwd files and then putting the files into a shadow file. I then unshadowed the files together to merge the passwd file and shadow file together. Then proceeded to put the hash entries into .txt file named Newfile.txt stored in my /Desktop/Lab3 folder. Having the .txt file allows John the ripper to begin cracking the passwords. The lines used in Kali Linux to start was "john newfile.txt" which created a directory of the hash entries. This at first only gave me 2 of the passwords and this was because the remaining passwords were shadowed. Now what I had to do was go in and unshadow everything. The line used was " unshadow passwd >newfile.txt. I ran John the ripper again and used line " john --show '/root/desktop/lab3/newfile.txt" and then this printed out the remaining hashes that were shadowed. My kali System crashed while I finished so I was unable to retrieve the remaining screenshots due to the fan on my laptop overheating.


VM Machine View Input Devices Window Help
Devin [Running]
Places Terminal
Sun 12:44
root@kali: ~/Desktop/LAB3
File Edit View Search Terminal Help

```
root@kali:~/Desktop/LAB3# john --show '/root/Desktop/LAB3/newfile.txt'
root:admin
neo:rabb1t

2 password hashes cracked, 3 left
root@kali:~/Desktop/LAB3# john --show '/root/Desktop/LAB3/newfile.txt'
root:admin
neo:rabb1t

2 password hashes cracked, 3 left
root@kali:~/Desktop/LAB3# john'/root/Desktop/LAB3/newfile.txt'
bash: john/root/Desktop/LAB3/newfile.txt: No such file or directory
root@kali:~/Desktop/LAB3# john '/root/Desktop/LAB3/newfile.txt'
Using default input encoding: UTF-8
Loaded 5 password hashes with 5 different salts (descrypt, traditional cryp
[DES 128/128 SSE2-16])
Remaining 3 password hashes with 3 different salts
Crash recovery file is locked: /root/.john/john.rec
root@kali:~/Desktop/LAB3# unshadow '/root/Desktop/LAB3/newfile.txt'
Usage: unshadow PASSWORD-FILE SHADOW-FILE
root@kali:~/Desktop/LAB3# ls
4.1.txt  lab-crypto2-fall2018.zip  newfile.txt  passwd.txt  zippedzip.txt
root@kali:~/Desktop/LAB3#
```

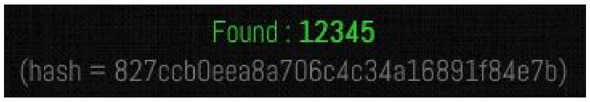
CRACKED HASHES:

- root:x11UGsMj7vR.A
  -admin
- michael:9rSp0Y.eGGDMc
  -leachim
- demi:CZ5MOX8FdBWoQ
  -jolie197
- john:NrrN.A7iPEVys
  -rippe
- neo:Qxw8mUWuWCUic
  -rabb1t
- cassandra:BHWyRvG9wuvHo
  -@p0110

4.2
An MD5 hash is a cryptographic security feature, which basically takes data like an email address, legal documents or images etc…. and converts it into a 32 hexa string. In this case we will decrypt the character hash into words. The only people that will see it is the people that know it and want to reveal it.

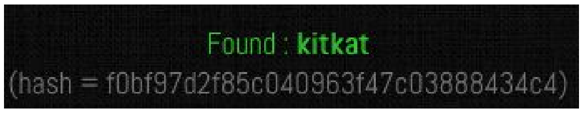
The goal is to find the preimage of an MD5 hash using a simple tool, such as john the ripper, hashcat, and other uncommon tools like mdcrack.

I will being going with a more simple approach by using a MD5 Hash generator also known as Dan's tools. Using md5online.org the results are below.



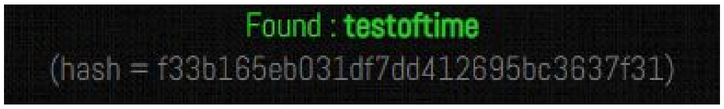
```
Found : 12345
(hash = 827ccb0eea8a706c4c34a16891f84e7b)
```

827ccb0eea8a706c4c34a16891f84e7b



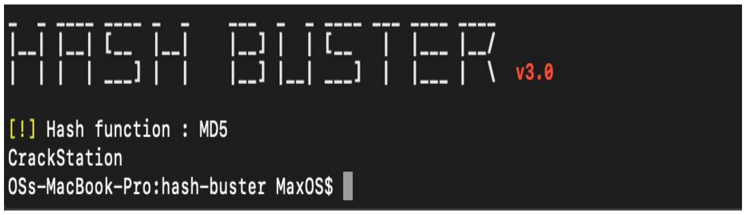
```
Found : kitkat
(hash = f0bf97d2f85c040963f47c03888434c4)
```

f0bf97d2f85c040963f47c03888434c4



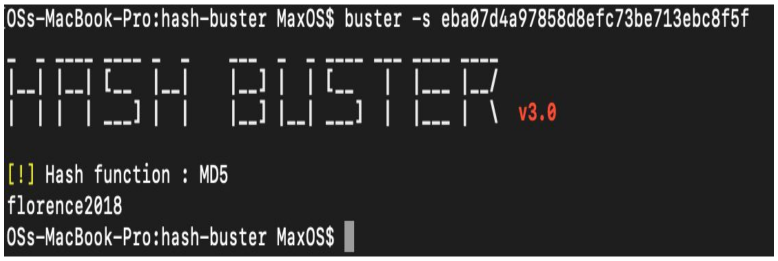
```
Found : testoftime
(hash = f33b165eb031df7dd412695bc3637f31)
```

f33b165eb031df7dd412695bc3637f31



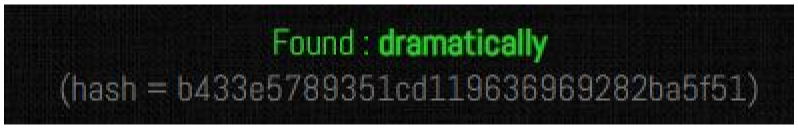
```
 _  _ ____ ____ _  _   ___  _  _ ____ ___ ____ ____
|__| |__| [__  |__|   |__] |  | [__   |  |___ |__/
|  | |  | ___] |  |   |__] |__| ___]  |  |___ |  \  v3.0

[!] Hash function : MD5
CrackStation
OSs-MacBook-Pro:hash-buster MaxOS$
```

05c73fd8d7e6db8d3dffd41ecb24a99b



```
OSs-MacBook-Pro:hash-buster MaxOS$ buster -s eba07d4a97858d8efc73be713ebc8f5f
 _  _ ____ ____ _  _   ___  _  _ ____ ___ ____ ____
|__| |__| [__  |__|   |__] |  | [__   |  |___ |__/
|  | |  | ___] |  |   |__] |__| ___]  |  |___ |  \  v3.0

[!] Hash function : MD5
florence2018
OSs-MacBook-Pro:hash-buster MaxOS$
```

eba07d4a97858d8efc73be713ebc8f5f



```
Found : dramatically
(hash = b433e5789351cd119636969282ba5f51)
```

b433e5789351cd119636969282ba5f51

Final thought: As you can see out of the six words four came out with actual words and two didn't. This normally mean two of the six hash isn't verified for file integrity, which are proven in different ways such as using pictures and even digital files. A specific program I have used a while back was md5com and does exactly that.

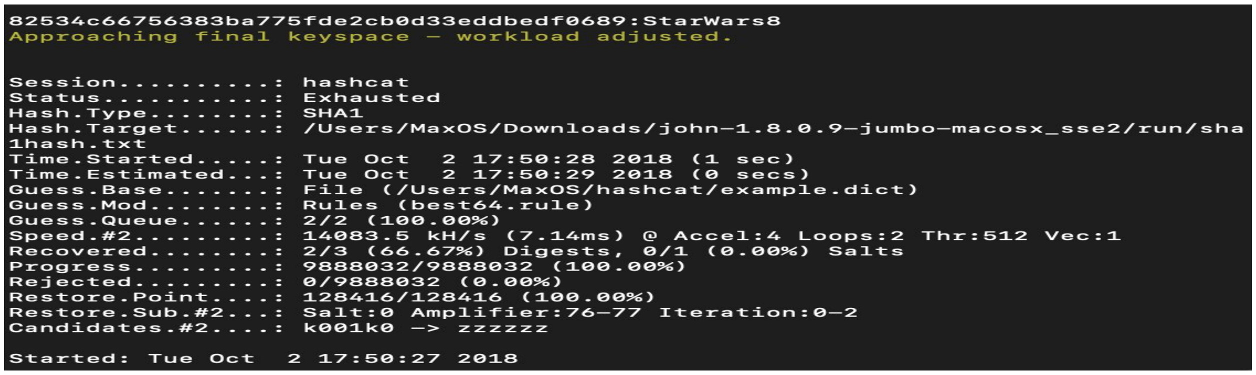
4.3 Finding a preimage to a SHA-1 hash

Cracking the sha1 hashes provided turned out to be a simple task, I just often found the misfortune of tools failing on me. Originally I started off with John the Ripper, however after the second hash was cracked, the iterations seemed to go on forever to find the third hash. I then decided to go about installing hashcat to my machine, at first getting the program set up proved to be a challenge due to some missing dependencies however, they were flushed out and I was able to finish in a timely manner.

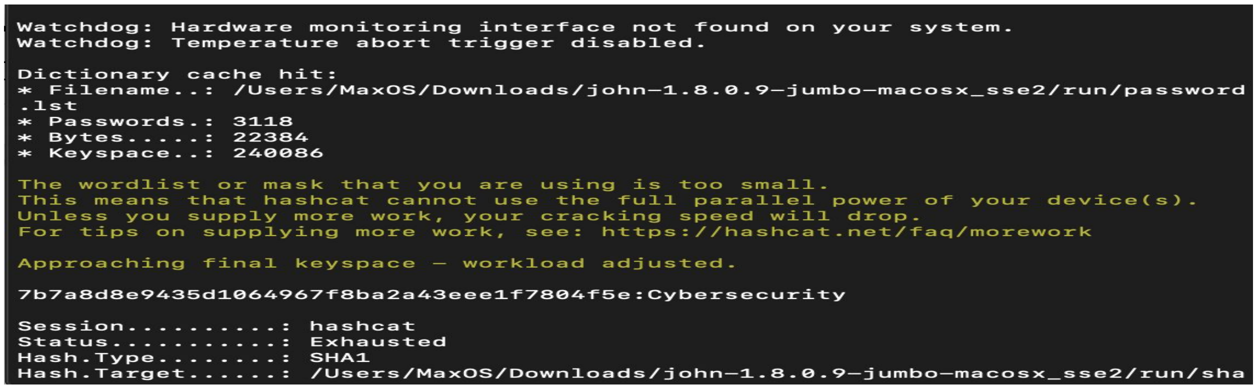
For John the Ripper my two solves were,
$dynamic_26$157b87e62bea5db4c8a3de54d8d795032e5ddb7c:jjc2018
$dynamic_26$bd7939b88afb88507676259e19ec8ae5f3b7cd40:csci4001

Due to my usage of only two word lists in hashcat, I was only able to find two words as well when using hashcat, these were the hashes for:
82534c66756383ba775fde2cb0d33eddbedf0689:StarWars8

7b7a8d8e9435d1064967f8ba2a43eee1f7804f5e:Cybersecurity

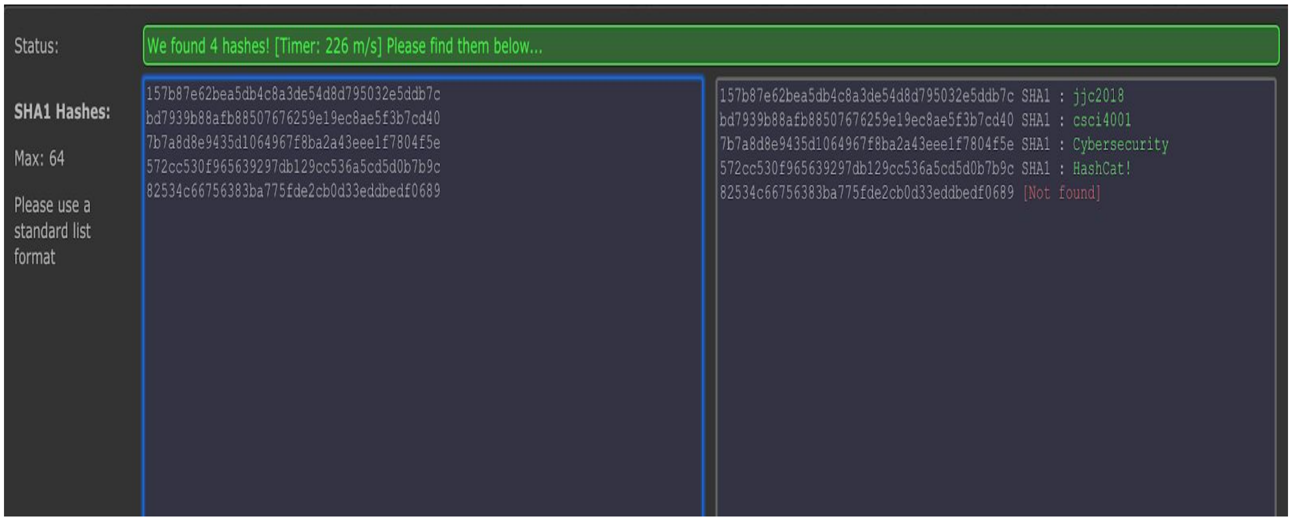


```
82534c66756383ba775fde2cb0d33eddbedf0689:StarWars8
Approaching final keyspace - workload adjusted.


Session..........: hashcat
Status...........: Exhausted
Hash.Type........: SHA1
Hash.Target......: /Users/MaxOS/Downloads/john-1.8.0.9-jumbo-macosx_sse2/run/sha
1hash.txt
Time.Started.....: Tue Oct  2 17:50:28 2018 (1 sec)
Time.Estimated...: Tue Oct  2 17:50:29 2018 (0 secs)
Guess.Base.......: File (/Users/MaxOS/hashcat/example.dict)
Guess.Mod........: Rules (best64.rule)
Guess.Queue......: 2/2 (100.00%)
Speed.#2.........: 14083.5 kH/s (7.14ms) @ Accel:4 Loops:2 Thr:512 Vec:1
Recovered........: 2/3 (66.67%) Digests, 0/1 (0.00%) Salts
Progress.........: 9888032/9888032 (100.00%)
Rejected.........: 0/9888032 (0.00%)
Restore.Point....: 128416/128416 (100.00%)
Restore.Sub.#2...: Salt:0 Amplifier:76-77 Iteration:0-2
Candidates.#2....: k001k0 -> zzzzzz

Started: Tue Oct  2 17:50:27 2018
```



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

Dictionary cache hit:
* Filename..: /Users/MaxOS/Downloads/john-1.8.0.9-jumbo-macosx_sse2/run/password
.lst
* Passwords.: 3118
* Bytes.....: 22384
* Keyspace..: 240086

The wordlist or mask that you are using is too small.
This means that hashcat cannot use the full parallel power of your device(s).
Unless you supply more work, your cracking speed will drop.
For tips on supplying more work, see: https://hashcat.net/faq/morework

Approaching final keyspace - workload adjusted.

7b7a8d8e9435d1064967f8ba2a43eee1f7804f5e:Cybersecurity

Session..........: hashcat
Status...........: Exhausted
Hash.Type........: SHA1
Hash.Target......: /Users/MaxOS/Downloads/john-1.8.0.9-jumbo-macosx_sse2/run/sha
```

The last image of this set contains an online hash cracking tool which provided me with a confirmation for the four I cracked using the tools, and one hash could not be solved which was “HashCat!”.



Status: We found 4 hashes! [Timer: 226 m/s] Please find them below...

SHA1 Hashes:

Max: 64

Please use a standard list format

```
157b87e62bea5db4c8a3de54d8d795032e5ddb7c
bd7939b88afb88507676259e19ec8ae5f3b7cd40
7b7a8d8e9435d1064967f8ba2a43eee1f7804f5e
572cc530f965639297db129cc536a5cd5d0b7b9c
82534c66756383ba775fde2cb0d33eddbedf0689
```

```
157b87e62bea5db4c8a3de54d8d795032e5ddb7c SHA1 : jjc2018
bd7939b88afb88507676259e19ec8ae5f3b7cd40 SHA1 : csci4001
7b7a8d8e9435d1064967f8ba2a43eee1f7804f5e SHA1 : Cybersecurity
572cc530f965639297db129cc536a5cd5d0b7b9c SHA1 : HashCat!
82534c66756383ba775fde2cb0d33eddbedf0689 [Not found]
```

4.4.1 Unlabeled hashes, single form

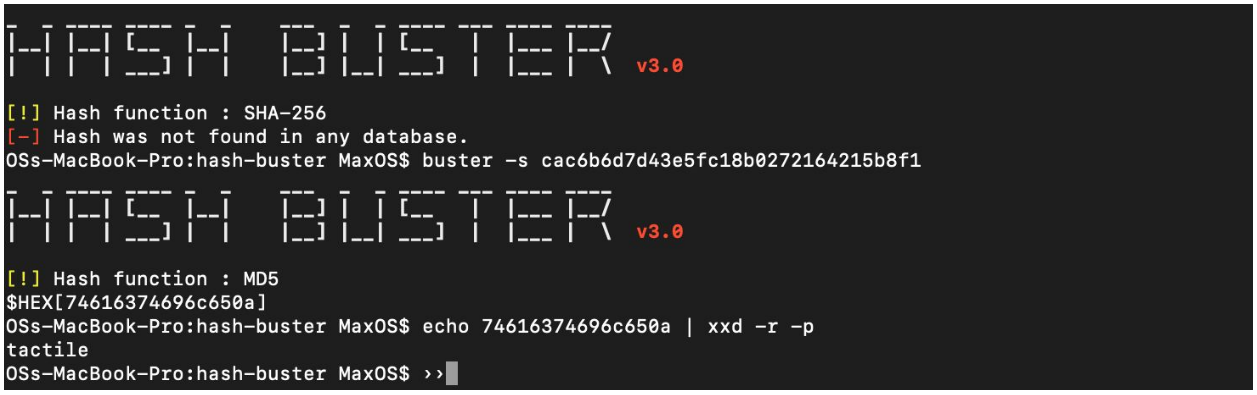
Hash 1: f573b7dd8369faa58368bb167539106ff0e1e027:challenged -> RIPEMD-160
Hash 2:0a4628c5ae0771b1d27ac71e2d2574dec4174f35fa0a6c5e7e0f98871fbcc657:Boom! -> SHA256 using rockyou mask, and a larger wordlist...
Hash 3: $2y$05$xrbOovgynQ9Cm/5CzX.POOyP80t1sxQkvRiYbuEpPwZaTTbdzdGFy: could not compute (limited gpu power) -> Blowfish (bcrypt)
Hash 4: 282e0c4b0434d019f360047296f00fb0ec519f9cd658967f694bc82a7966f17bab5511 0bc711b56b2f4d46ba0d18ea4066cb5427831d889479e90e793f9f3653 (size too high could not compute) -> unknown hash found however analyzing the size it is probably SHA-512.

4.4.2 Keyed and nested hashes

Unable to complete... Tried web scraping a database of female names in order to place into a word list however the list was hard to implement and merge into an existing one. Based on the operations of HMAC I likely would have used the md5 hash with names attached in order to crack the hash. Once I found a match, I would then use the same concept to break the next hash 7b3d979ca8330a94fa7e9e1b466d8b99e0bcdea1ec90596c0dcc8d7ef6b4300c by creating a wordlist, and appending each line to openssl and do a compare of hashes to the hash needed to find the original file.

4.4.3 Something much harder: unusual hashes

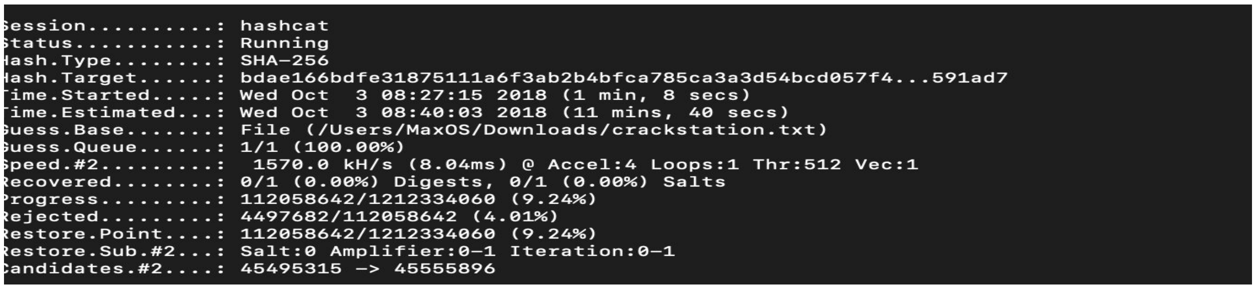
OSs-MacBook-Pro:masks MaxOS$ echo mike | openssl md5
(stdin)= 2ff8557a16f674e466ee4ae619f22758
OSs-MacBook-Pro:masks MaxOS$ echo -n mike | openssl md5
(stdin)= 18126e7bd3f84b3f3e4df094def5b7de
md5 : cac6b6d7d43e5fc18b0272164215b8f1



```
HASH BUSTER v3.0

[!] Hash function : SHA-256
[-] Hash was not found in any database.
OSs-MacBook-Pro:hash-buster MaxOS$ buster -s cac6b6d7d43e5fc18b0272164215b8f1
HASH BUSTER v3.0

[!] Hash function : MD5
$HEX[74616374696c650a]
OSs-MacBook-Pro:hash-buster MaxOS$ echo 74616374696c650a | xxd -r -p
tactile
OSs-MacBook-Pro:hash-buster MaxOS$ ››
```

SHA-256 : bdae166bdfe31875111a6f3ab2b4bfca785ca3a3d54bcd057f448ef762591ad7

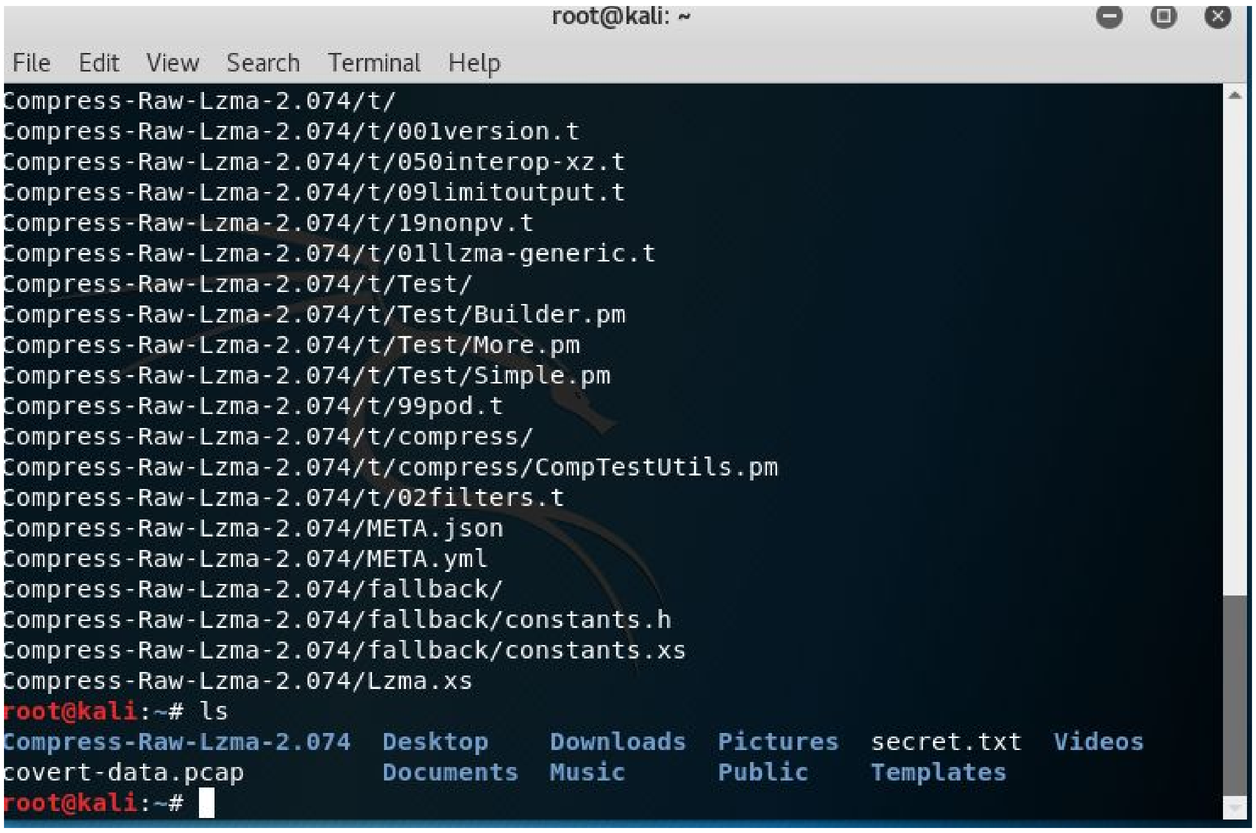
I attempted running in hashcat with a larger wordlist file that I bought from crackstation however the keyspace is so massive that it would take an extended period of time....



```
ession..........: hashcat
tatus...........: Running
ash.Type........: SHA-256
ash.Target......: bdae166bdfe31875111a6f3ab2b4bfca785ca3a3d54bcd057f4...591ad7
ime.Started.....: Wed Oct  3 08:27:15 2018 (1 min, 8 secs)
ime.Estimated...: Wed Oct  3 08:40:03 2018 (11 mins, 40 secs)
uess.Base.......: File (/Users/MaxOS/Downloads/crackstation.txt)
uess.Queue......: 1/1 (100.00%)
peed.#2.........:  1570.0 kH/s (8.04ms) @ Accel:4 Loops:1 Thr:512 Vec:1
ecovered........: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
rogress.........: 112058642/1212334060 (9.24%)
ejected.........: 4497682/112058642 (4.01%)
estore.Point....: 112058642/1212334060 (9.24%)
estore.Sub.#2...: Salt:0 Amplifier:0-1 Iteration:0-1
andidates.#2....: 45495315 -> 45555896
```

4.5 Encrypted archive files

4.5.2



```
Compress-Raw-Lzma-2.074/t/
Compress-Raw-Lzma-2.074/t/001version.t
Compress-Raw-Lzma-2.074/t/050interop-xz.t
Compress-Raw-Lzma-2.074/t/09limitoutput.t
Compress-Raw-Lzma-2.074/t/19nonpv.t
Compress-Raw-Lzma-2.074/t/01llzma-generic.t
Compress-Raw-Lzma-2.074/t/Test/
Compress-Raw-Lzma-2.074/t/Test/Builder.pm
Compress-Raw-Lzma-2.074/t/Test/More.pm
Compress-Raw-Lzma-2.074/t/Test/Simple.pm
Compress-Raw-Lzma-2.074/t/99pod.t
Compress-Raw-Lzma-2.074/t/compress/
Compress-Raw-Lzma-2.074/t/compress/CompTestUtils.pm
Compress-Raw-Lzma-2.074/t/02filters.t
Compress-Raw-Lzma-2.074/META.json
Compress-Raw-Lzma-2.074/META.yml
Compress-Raw-Lzma-2.074/fallback/
Compress-Raw-Lzma-2.074/fallback/constants.h
Compress-Raw-Lzma-2.074/fallback/constants.xs
Compress-Raw-Lzma-2.074/Lzma.xs
root@kali:~# ls
Compress-Raw-Lzma-2.074  Desktop    Downloads  Pictures  secret.txt  Videos
covert-data.pcap         Documents  Music      Public    Templates
root@kali:~#
```

Kali died and lost the rest of the screenshots.
Ran john /root/desktop/challenge.hash
And got the password Optimistic
The .hash file will be included in the .zip file

Word Problems

1. Summarize the attack techniques used by the tools.

The attack techniques used by the password cracking tools were brute force, modified brute force, and rainbow table attacks. Brute force was a generalized attack from a dictionary. A modified brute force approach would perform a ruleset in order to significantly lesson the computational time require, as it turns out; hashcat proved to be the fairer option in terms of rulesets and matches. Plaintext attacks uses the fact that a hacker would know a plaintext. Rainbow tables would attempt to reverse cryptographic functions.

2. Looking at the rainbow table approach, does it sound like a viable alternative?

I would say the Rainbow Table is a rather situational usage but I can see its appeal when it comes to cracking authentication (handshake based hashes), and also combines the usage cryptographic tables. It does come in handy when attempting to crack built in cryptographic systems such as Windows' LM hashes.

3. How does JTR (or the others) improve on simple dictionary attacks?

JTR and other tools improve upon simple dictionary attacks by providing a medium which modifies the dictionary files you provide it with, john the ripper can do many different iterations of the same word. Similarly, HashCat can be used and customized to provide type matching, specific patterns, and rulesets which aid in lessening the keyspace of the dictionary attack.

4. Does using a GPU help? How so?

Yes, if using a gpu, the gpu can utilize many cores rather than the limitations of your processor which typically has 4 to 8 on modern systems with "reasonably specifications". The throughput of the CPU proves to be several times slower than any GPU attack.

5. How would you improve these techniques further?

The way to improve these techniques further would be to increase wordlist sizes, specify ranges/sizes of words to limit the keyspace, provide rules based on testing of the cracking, and use masking files. I would also used a faster gpu than what I was currently using as this will decrease computational times by a large percentage in most cases.