Intro to Hash Cracking

So you want to crack hashes? Let's first briefly cover what hashing is and why it's used.

Intro

Rather than storing sensitive information such as login credentials in plaintext, most databases use hashing to obscure sensitive database entries. For example, if you used an insecure password such as, *password*, and the server were using the hashing algorithm MD5, your hashed password would be, *5F4DCC3B5AA765D61D8327DEB882CF99*. Databases store passwords in hashed entries and hash cracking is the process of turning these hashed entries back into plaintext. Besides a few outdated hashing algorithms which are no longer in common use, the only way to crack hashes is to guess the plaintext. There's no exploits or hacks, you literally have to guess the password. This is where good hash cracking practises come into play with good wordlists, rules and fast GPU's all playing an important part in this process. The most important part of this process is you as the hash cracker must know when and where to apply your skills for this process to be effective and efficient.

Hash Cracking Programs

There are multiple programs that can be used for hash cracking such as John the Ripper or MDXfind, but we'll focus on Hashcat. If you haven't already, download the binary files here: https://hashcat.net and unzip the hashcat-5.x.x.7z file we just downloaded into directory called "hashing" (or whatever you want to call it).

You'll notice Hashcat includes binaries for both Windows (.exe) and Linux (.bin), but for this tutorial we'll be demonstrating the commands from a Linux terminal. Keep in mind most of the commands we'll be using for this tutorial are similar for both OS's, so once you learn how to run hashcat on Linux, switching over to Windows is fairly easy and visa-versa.

Wordlists

The wordlist we use is extremely important to the success of our attack. A poor wordlist will give poor results, and just because a wordlist is huge with 10's of gigabytes doesn't mean it's efficient. This is especially true with slow algorithms such as bcrypt where running an efficient wordlist without rules may be necessary due to time.

Since we'll need a wordlist for our attack we'll download one from weakpass.com which contains a short list of 100,000 most common passwords. Unzip this wordlist to your hashcat folder. I like to keep my wordlists in their own subfolder called "wordlists", but you can choose to set up your folder structure however you like.

https://weakpass.com/wordlist/49

The Attack

Now we need a sample set of hashes to try our attack on, so copy and paste the following MD5 hashes into a file called *hashes.txt* and place this file in your hashcat folder as well.
All of the password hashes listed below were derived from the wordlist we downloaded in the previous step.

```
5f4dcc3b5aa765d61d8327deb882cf99
0d107d09f5bbe40cade3de5c71e9e9b7
84d961568a65073a3bcf0eb216b2a576
f25a2fc72690b780b2a14e140ef6a9e0
8afa847f50a716e64932d995c8e7435a
55f9c405bd87ba23896f34011ffce8da
58e50f904aab05ac687efc1635421d78
72a97fb793d496318518aebc7e9298b2
c62d929e7b7e7b6165923a5dfc60cb56
e0e34c5ad05aac3eef6ab31eacbf7a5c
5e8667a439c68f5145dd2fcbecf02209
ce608d5892ee2872771f4b67144405e0
dbfd69abf4df53dee082a1165ded4f08
94d756215ab39626d9e84a5c1a63aef5
97db1846570837fce6ff62a408f1c26a
faf5ba124a0b97c9e59c1f9c8c7d2d24
0d2489f3616693c707709295fd091c84
eb01627e574fe1054d0e1be3dbd02335
cd08b61ac93896d976abde8694980166
9fceecbb0047b7f21a62ab4cb07d073d
1b4329e9b4051be1f368e300285b6903
084a0ff182baa188032cf8e9b4d632cc
cadac4d712c63f4cec146d91f18bfb85
e88d4453db667bfcf194521b5072fa63
5dcc146ae54e2f6ed3c0e1400b0bdc6b
a36d09f3df5a75f6814c74fa1b20a06a
bc11dd3115d29514c9362c77ffa7427c
```

Open a terminal in your hashcat directory and run the following command:
Note: you may need to edit this command to match the folder structure on your system if your wordlists are in a subfolder, etc.

```
hashcat.bin -m 0 -a 0 -O -w 3 hashes.txt 10_million_password_list_top_100000.txt
```

Let’s take a moment to explain the command and options we’re using.

hashcat.bin	hashcat binary	
-m 0	-m = MODE	0 = MD5
-w 3	Workload profile	3 = High
-a 0	-a = attack	0 = wordlist
-O	Optimised kernel - speeds up attack on supported modes	
hashes.txt	Text file containing hashes we're trying to crack	
10_million_pass...	Wordlist we're using for the attack	

If everything went well, we should have cracked 13 / 27 hashes! If you received an error, double check your command syntax and make sure you've installed the correct drivers for your GPU.

```
0d2489f3616693c707709295fd091c84:PLATINUM
f25a2fc72690b780b2a14e140ef6a9e0:iloveyou
e0e34c5ad05aac3eef6ab31eacbf7a5c:victoria
5e8667a439c68f5145dd2fcbecf02209:87654321
8afa847f50a716e64932d995c8e7435a:princess
72a97fb793d496318518aebc7e9298b2:cowboys
97db1846570837fce6ff62a408f1c26a:1q2w3e4r5t
c62d929e7b7e7b6165923a5dfc60cb56:q1w2e3r4
5f4dcc3b5aa765d61d8327deb882cf99:password
0d107d09f5bbe40cade3de5c71e9e9b7:letmein
84d961568a65073a3bcf0eb216b2a576:superman
55f9c405bd87ba23896f34011ffce8da:maverick
58e50f904aab05ac687efc1635421d78:eagles
```

Rules

Now let's expand our attack by using rules for our wordlist. This is done by adding the -r switch to our hashcat command and specifying the ruleset to use. Hashcat includes several rulesets so we'll start of by using a basic set of rules, *best64.rule*. Using rules will mangle each word contained in our wordlist. For example, the rule **l** (*lower case L*) will lower the case of each word, so *PassWord* becomes *password*.

Hashcat Rules: https://hashcat.net/wiki/doku.php?id=rule_based_attack

Run the following command from terminal.
*Note the added **-r rules/best64.rule** appended to the previous command.*

```
hashcat.bin -m 0 -a 0 -O -w 3 hashes.txt 10_million_password_list_top_100000.txt
-r rules/best64.rule
```

Applying best64.rule to our wordlist cracked 6 more hashes!

```
eb01627e574fe1054d0e1be3dbd02335:qweasdzxc13
faf5ba124a0b97c9e59c1f9c8c7d2d24:garfield123
94d756215ab39626d9e84a5c1a63aef5:liverpool77
ce608d5892ee2872771f4b67144405e0:qwerty123123
dbfd69abf4df53dee082a1165ded4f08:theqwerty123
cd08b61ac93896d976abde8694980166:CzPS5NYNDWCkSC23
```

Let’s try another ruleset called generated.rule and see how many more hashes we can crack.

```
hashcat.bin -m 0 -a 0 -O -w 3 hashes.txt 10_million_password_list_top_100000.txt
-r rules/generated.rule
```

Running our attack with generated.rule cracked 3 passwords.

```
1b4329e9b4051be1f368e300285b6903:uoyevo,li
9fceecbb0047b7f21a62ab4cb07d073d:Mcoolbugi2000Mcoolbugi2000
cadac4d712c63f4cec146d91f18bfb85:9coolbugi2000G
```

Rule Stacking

Let’s try something else, rule stacking. This is where we’ll run multiple sets of rules to further mangle our wordlist. This typically only works well with smaller rulesets as rule stacking adds considerable time to the attack with larger wordlist / rules. While this attack is not efficient, as we’ll soon see this can be an effective attack for finding more complex passwords. For this attack we’ll stack best64.rule. Make sure to add a second -r switch like our example below:

```
hashcat.bin -m 0 -a 0 -O -w 3 hashes.txt 10_million_password_list_top_100000.txt
-r rules/best64.rule -r rules/best64.rule
```

Two more passwords fell to our rule stacking attack.

```
a36d09f3df5a75f6814c74fa1b20a06a:seirrebnarc21
bc11dd3115d29514c9362c77ffa7427c:|hecoolbugi2000
```

The Last 20%

While our last 3 uncracked hashes do not equal 20% of the test set, the last remaining hashes that haven’t cracked are commonly referred to as *The Last 20%*. Some of these remaining hashes will crack with targeted wordlist and rulesets, but in the real world of hash cracking some hashes with long, complex plaintext will never crack even with mask / brute force attacks due to the time required to run through their entire keyspace. The good news for you is the 3 remaining hashes in this test set will crack easily if you find the right wordlist + ruleset, so explore some of your newly learned skills and have fun cracking these.

Conclusion

While we’ve only scratched the surface when it comes to hash cracking, I hope you’ve had fun cracking these hashes and learned a few things along the way.

~Cyclone

In case you couldn’t figure them out I’ve included the hash:plaintext list below.

```
5f4dcc3b5aa765d61d8327deb882cf99:password
0d107d09f5bbe40cade3de5c71e9e9b7:letmein
84d961568a65073a3bcf0eb216b2a576:superman
f25a2fc72690b780b2a14e140ef6a9e0:iloveyou
8afa847f50a716e64932d995c8e7435a:princess
55f9c405bd87ba23896f34011ffce8da:maverick
58e50f904aab05ac687efc1635421d78:eagles
72a97fb793d496318518aebc7e9298b2:cowboys
c62d929e7b7e7b6165923a5dfc60cb56:q1w2e3r4
e0e34c5ad05aac3eef6ab31eacbf7a5c:victoria
5e8667a439c68f5145dd2fcbecf02209:87654321
ce608d5892ee2872771f4b67144405e0:qwerty123123
dbfd69abf4df53dee082a1165ded4f08:theqwerty123
94d756215ab39626d9e84a5c1a63aef5:liverpool77
97db1846570837fce6ff62a408f1c26a:1q2w3e4r5t
faf5ba124a0b97c9e59c1f9c8c7d2d24:garfield123
0d2489f3616693c707709295fd091c84:PLATINUM
eb01627e574fe1054d0e1be3dbd02335:qweasdzxc13
cd08b61ac93896d976abde8694980166:CzPS5NYNDWCkSC23
9fceecbb0047b7f21a62ab4cb07d073d:Mcoolbugi2000Mcoolbugi2000
1b4329e9b4051be1f368e300285b6903:uoyevo,li
084a0ff182baa188032cf8e9b4d632cc:grarfield12333333
cadac4d712c63f4cec146d91f18bfb85:9coolbugi2000G
e88d4453db667bfcf194521b5072fa63:cranberris!!!
5dcc146ae54e2f6ed3c0e1400b0bdc6b:thegarfield1234
a36d09f3df5a75f6814c74fa1b20a06a:seirrebnarc21
bc11dd3115d29514c9362c77ffa7427c:|hecoolbugi2000
```