Tyler Jackson
Advance Network and System Security
3/19/2015
HW 3

Password Cracking Report

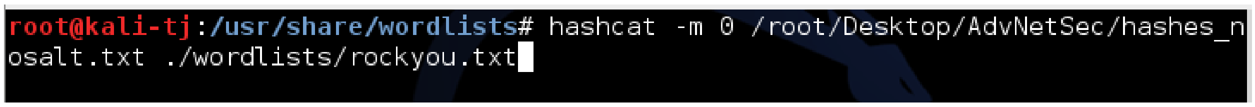
This report describes the techniques I used to crack an MD5 hashed password. We were given approximately 385 MD5 hashes with a mix of salted hashes and non salted hashes. The salts differ between the hashes that are salted, and we have no information regarding the length of each password.

The first step was to do a little bit of research about possible cracking techniques. Since a hash will always result in the same size, there is no way to discern how long the passwords are by working backwards. This leaves us with a couple options. Dictionary attacks, and brute force attacks. Dictionary attacks will match a list of previously calculated hashes against the hash list you provide. If your hashes match then we know the password they have on file is the password for our hash as well. This is under the assumption that a good hashing algorithm would never map 2 passwords to the same hash value. The dictionaries also work for salted passwords assuming they contain lists of salted hashes.

To start off I looked into some of the password cracking tools that come built into Kali, and one of the easier to use tools I came across was hashcat. Hashcat comes with built in tools to crack many different types of encryptions.

Step1: I grouped the salted passwords together and the unsalted passwords together into separate files because they will be handled differently in hashcat. This was easier to do once I loaded the file into SequelPro, because then I could query the passwords more easily.

Step2: Download a dictionary file to be used for dictionary attacks. Kali comes with a word list to use for cracking passwords so I started with that. The first command to run was

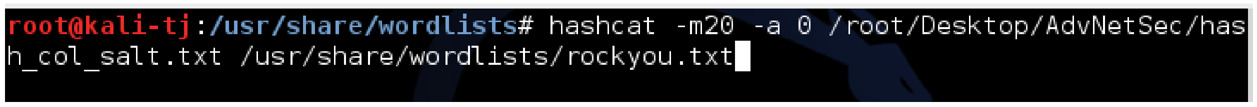


```
root@kali-tj:/usr/share/wordlists# hashcat -m 0 /root/Desktop/AdvNetSec/hashes_n
osalt.txt ./wordlists/rockyou.txt
```

"hashcat" - tells Kali to run the hashcat program
"-m 0" -flag in hashcat that specifies the input file has normal MD5 hashes
"hashes_nosalt.txt" -this if the file of hashes you want to crack - you can see here it is my unsalted hashes
"./wordlists/rockyou.txt" - this is the wordlist that comes with Kali. I believe all you have to do is decompress it

This dictionary attack found a decent amount of the passwords.

Step 3: Repeat that process for the salted hashes



```
root@kali-tj:/usr/share/wordlists# hashcat -m20 -a 0 /root/Desktop/AdvNetSec/hash_col_salt.txt /usr/share/wordlists/rockyou.txt
```

The differences in this command is the "-m20" and -a 0.

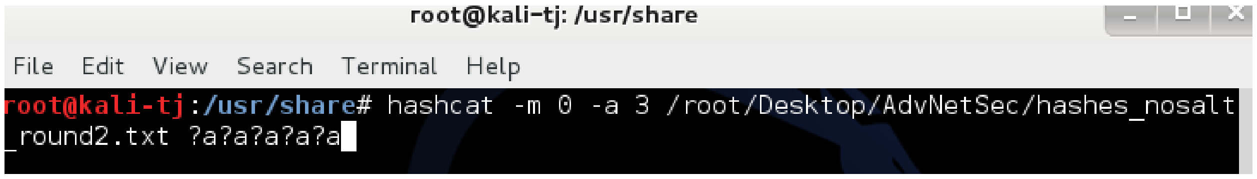
-m20 -describes the type of md5 hash attack to perform. Instead of -m 0 which is an unsalted hash attack -m20 specifies that the input file has passwords where the salt was prepended to the front of the password.

-a 0 - this just says to use a dictionary attack

Step 4: This next step is to remove the cracked hashes from our list and repeat with a bigger dictionary. The next dictionary I downloaded was 19 GB when expanded. You can find dictionary files online relatively easily. This found a lot more passwords. I ran this on my native device (Mac OS) as I hadn't allocated enough memory for my virtual machine.

My Mac is also faster than my vm, so it made sense now that I am getting into more complicated attempts to switch over to my more powerful machine. I did run some simple brute force attacks in Kali which I will show below, but the more complex the brute force attack became, the more important it was that my machine was fast. In order to do this I had to install hashcat on my mac. Again this is not difficult to figure out, with google.

Step 5: Now that I had used up my reasonable options it was time to try the more computationally expensive approach: brute force. Brute force essentially tries every possible combination of characters in a password, hashes the password, and then compares the hash to your value for a match. This can become extremely expensive very quickly.



```
root@kali-tj:/usr/share# hashcat -m 0 -a 3 /root/Desktop/AdvNetSec/hashes_nosalt_round2.txt ?a?a?a?a?a
```

This says to crack an md5 hash using a file of unsalted hashes, using brute force ("-a 3").
The ?a?a?a?a?a specifies what type of brute force to attempt.
hashcat has 4 different character sets, l u d and s. The "a" set represents a combination of them all. So having 5 "?a" in this command means try every combination of characters for a match of password sizes from 1-5.

?l = abcdefghijklmnopqrstuvwxyz

?u = ABCDEFGHIJKLMNOPQRSTUVWXYZ
?d = 0123456789
?s = !"#$&'()*+,-./:;<=>?@[\]^_`{|}~
?a = ?l?u?d?s

While running this command you can hit enter to see the current progress.

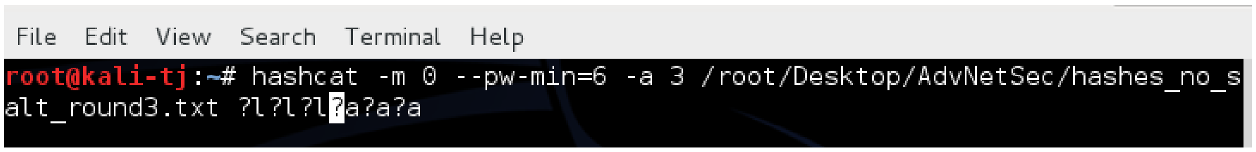
Step 6: At this point the only thing to do is to get creative with your brute force attacks, or get a more powerful machine. On my Mac ?a?a?a?a?a?a was a reasonable amount of time, but 7 characters was too long. I tried some combinations with

?l?l?l?a?a?a

in order to cut down on the computation time a little, as well as

?l?l?l?l?l?l?l

Seven character passwords took a little longer, but choosing only 1 character set made it a little easier. Another helpful hashcat command is



```
root@kali-tj:~# hashcat -m 0 --pw-min=6 -a 3 /root/Desktop/AdvNetSec/hashes_no_s
alt_round3.txt ?l?l?l?a?a?a
```

--pw-min=6 means it won't try passwords less than 6 characters long. This was nice because as you have already tried the smaller hashes, all that trying them does at this point is take up time.

Hardest Parts:

The hardest parts for me were regarding hashcat syntax. Most of the time was spent finding the tool I wanted to use, and then learning how to get it to do what I want. Now that I know how to use it, it is very easy in my opinion, but to troll the internet for tutorials is always somewhat time consuming. The only other difficulty was not having a strong enough computer to try harder attacks. In theory, an attacker could set up 10,000 bots, and make the brute force attack each one tries very specific so that the computation time is small for each computer and then crack them all. I am not sure if there is really any trick to solving really long uncommon passwords other than something like that.

Its a little easy how easy MD5 is to crack since it is still used in some systems today, and this was when the salts were varying for each password which I would imagine a lot of the time they are not.