HASHCAT - CRACKING EHARMONY HASHES EASILY

I have recently been using *hashcat* to crack through the linkedin hashdump (see here). So I figured that if I wanted to show some easy steps of how to use *hashcat*, lets start by using what I have found for linkedin against the eharmony hashdump. This will be a way to see how "valuable" the linkedin dictionary is and how to use *hashcat*. So to get started:

- linkedin dictionary
- eharmony hashdump
- hashcat

Lets start by taking a look at the initial file. We see that there are just over 1.5 million hashes and we know they are MD5. So lets start with the simplest check, we can feed *hashcat* our dictionary and it will check for each entry as a straight comparison *(-a 0)*. This is very quick but misses a lot obviously:

```
./hashcat-cli32.bin -m 0 -a 0 --remove ../eharmony.hash ./linked.dic
.
.
Input.Mode: Dict (./linked.dic)
Index.....: 2/2 (segment), 828550 (words), 7909187 (bytes)
Recovered.: 1325/1513805 hashes, 0/1 salts
Speed/sec.: - plains, - words
Progress..: 828550/828550 (100.00%)
Running...: --:--:--:--
Estimated.: --:--:--:--
```

So you can see that straight off the bat we got 1325 passwords. But we also saw something interesting. If you have been following along you would have seen that all the passwords are uppercase. This fits in great with our next test. *Hashcat* has a mode which switches the case of each character in each dictionary entry *(-a 2)*, lets see what it finds:

```
./hashcat-cli32.bin -m 0 -a 2 --remove ../eharmony.hash ./linked.dic
.
.
Input.Mode: Dict (./linked.dic)
Index.....: 2/2 (segment), 828364 (words), 7909187 (bytes)
Recovered.: 69970/1512480 hashes, 0/1 salts
Speed/sec.: 23.76M plains, 41.38k words
Progress..: 828364/828364 (100.00%)
Running...: 00:00:00:20
Estimated.: --:--:--:--
```

Alright, doing that got us another 69970 passwords. Again, if you are following along you will have seen that the uppercase entries persist, so lets convert our dictionary to uppercase now:

```
cat ./linked.dic | tr [a-z] [A-Z] | sort | uniq > ./linked-upper.dic
```

Now lets
carry on. *Hashcat* also comes with a set of rules each of which is basically a set of word mangling rules. These rules are split by origin, focus, etc. Lots of people also release rulesets that have worked for them (Kore security for example). So lets run our test using some of the rules I have personally found to be useful:

```
./hashcat-cli32.bin -m 0 -r ./rules/best-422RvtEplJ.rule -r ./rules/best64.rule -r ./
rules/best-mpGFdsZoei.rule -r ./rules/passwordspro.rule --remove ../eharmony.hash ./
linked-upper.dic
.
```

```
.
Input.Mode: Dict (./linked-upper.dic)
Index.....: 2/2 (segment), 706443 (words), 6743575 (bytes)
Recovered.: 336059/1442507 hashes, 0/1 salts
Speed/sec.: 15.01M plains, 4.81k words
Progress..: 706443/706443 (100.00%)
Running...: 00:00:02:27
Estimated.: --:--:--:--
```

That was useful! As an example of custom rulesets, I will run the test using the rules I built up while going through the linkedin hashes (see here), lets see how useful that is:

```
./hashcat-cli32.bin -m 0 -r ./linkedin2012.rule --remove ../eharmony.hash ./linked-
upper.dic
.
.
Input.Mode: Dict (./linked-upper.dic)
Index.....: 2/2 (segment), 706443 (words), 6743575 (bytes)
Recovered.: 367980/1106448 hashes, 0/1 salts
Speed/sec.: 16.69M plains, 1.03k words
Progress..: 706443/706443 (100.00%)
Running...: 00:00:11:24
Estimated.: --:--:--:--
```

It seems that the ruleset I built up is proving useful in this instance. Up until now we have been trying to use "*smart*" methods of guessing the passwords, but *hashcat* also allows us to fall back to a brute-force *(-a 3)* method. Now since it seems that *EHarmony* made all the passwords uppercase (and I just have to say - *What the Hell!*) and there appear to be almost no special characters in the passwords so far (I shudder to think of the implications of that), we can narrow down our character set and thus shorten the time taken:

```
./hashcat-cli32.bin -m 0 -a 3 -1 ?u?d --remove ../eharmony.hash ?1?1?1?1?1?1?1
.
.
Input.Mode: Mask (?1?1?1?1?1?1?1)
Index.....: 0/1 (segment), 78364164096 (words), 0 (bytes)
Recovered.: 118051/738468 hashes, 0/1 salts
Speed/sec.: 10.57M plains, 10.57M words
Progress..: 78364164096/78364164096 (100.00%)
Running...: 00:02:03:30
Estimated.: --:--:--:--
```

Once again it is like shooting fish in a barrel. So we have seen that the linkedin dictionary and rules are a valuable and useful resource, we have seen that with a few simple commands we managed to crack 893388 hashes, 59% of the full list. So grab the hashes, the dictionary and hashcat and give it a try.