John the Ripper and Hashcat - Markov Mode Comparison

***UPDATE**: Below was my first article on this, my updated article is here*

Looking back at the year it seems I spent a lot of time dealing with passwords. While it was fun and enlightening, it brought home again how we start taking our tools for granted. We should always be looking at our tools to see if they can be used better or if they should be replaced. I will admit right up front, I am a great fan of both *hashcat* and *john the ripper*, and I think they are both great tools. They each excel in certain areas, but there is always a lot of debate over which is better. This intensifies when the comparison is around the implementation of the "*Markov*" mode attack. I will not repeat what you can find on wikipedia but in VERY simple terms it allows for better "brute forcing" after analyzing a list of already cracked passwords.

Both *john the ripper* and *hashcat* implement this, but in different ways. I wanted to see how they stacked up against one another. I have seen some articles on this, but nothing that I personally could look at and say "aha". So I thought I would try it. Let me explain the methodology I used:

- Done on linux
- Using *hashcat-0.41* with hashutils 0.9 and statsprocessor 0.8
- Using *john the ripper jumbo 1.7.9-7*
- The list of hashes I was testing against was all the LinkedIn unmasked hashes (2935345)
- I used the *shuf* utility to create 3 random training sets of 15% (440301)
- I used the *shuf* utility to create 3 random training sets of 30% (880602)
- My final training set was the full list of unmasked hashes I have cracked (2051219)
- I used no rules / masks / mangling / etc in these attacks, just a straight word cracking
- I am checking for password lengths from 1-12

I then needed make sure the tests were as fair as what I could make them, so I did some tests. For *john the ripper* you can generate a listing of how many words would be generated depending on your options chosen:

```
# ./genmkvpwd ./15-1.stats 0 12
lvl=203 (5304 KB for nbparts) 279 M possible passwords (279211970)
lvl=204 (5330 KB for nbparts) 308 M possible passwords (308990934)
lvl=205 (5356 KB for nbparts) 341 M possible passwords (341968690)
lvl=206 (5382 KB for nbparts) 378 M possible passwords (378479996)
lvl=207 (5408 KB for nbparts) 418 M possible passwords (418920809)
lvl=208 (5434 KB for nbparts) 463 M possible passwords (463686444)
lvl=209 (5460 KB for nbparts) 513 M possible passwords (513245298)
lvl=210 (5486 KB for nbparts) 568 M possible passwords (568133765)
lvl=211 (5512 KB for nbparts) 628 M possible passwords (628904972)
lvl=212 (5538 KB for nbparts) 696 M possible passwords (696203842)
lvl=213 (5564 KB for nbparts) 770 M possible passwords (770693752)
lvl=214 (5590 KB for nbparts) 853 M possible passwords (853137221)
lvl=215 (5616 KB for nbparts) 944 M possible passwords (944394405)
lvl=216 (5642 KB for nbparts) 1 G possible passwords (1045396223)
lvl=217 (5668 KB for nbparts) 1 G possible passwords (1157150786)
lvl=218 (5694 KB for nbparts) 1 G possible passwords (1280839452)
lvl=219 (5720 KB for nbparts) 1 G possible passwords (1417711153)
lvl=220 (5746 KB for nbparts) 1 G possible passwords (1569173589)
lvl=221 (5772 KB for nbparts) 1 G possible passwords (1736786871)
lvl=222 (5798 KB for nbparts) 1 G possible passwords (1922282761)
lvl=223 (5824 KB for nbparts) 2 G possible passwords (2127558397)
lvl=224 (5850 KB for nbparts) 2 G possible passwords (2354700749)
lvl=225 (5876 KB for nbparts) 2 G possible passwords (2 606 095 008)
lvl=226 (5902 KB for nbparts) 2 G possible passwords (2884331225)
```

```
lvl=227 (5928 KB for nbparts) 3 G possible passwords (3192234349)
lvl=228 (5954 KB for nbparts) 3 G possible passwords (3533019214)
lvl=229 (5980 KB for nbparts) 3 G possible passwords (3910225126)
```

For *hashcat* I had to be a bit more creative:

```
# ./sp32.bin --pw-max 12 -t 1 ../hashcat-utils-0.9/15-1.stats | wc -l == 12
# ./sp32.bin --pw-max 12 -t 2 ../hashcat-utils-0.9/15-1.stats | wc -l == 8190
# ./sp32.bin --pw-max 12 -t 3 ../hashcat-utils-0.9/15-1.stats | wc -l == 797160
# ./sp32.bin --pw-max 12 -t 4 ../hashcat-utils-0.9/15-1.stats | wc -l == 22369620
# ./sp32.bin --pw-max 12 -t 5 ../hashcat-utils-0.9/15-1.stats | wc -l == 305 175 780
(305M)
# ./sp32.bin --pw-max 12 -t 6 ../hashcat-utils-0.9/15-1.stats | wc -l == 2 612 138 802
(2.6G)
```

So as you can see, *hashcat* thresholds 5 and 6 corrolate most closely to the *john the ripper* 204 and 225 levels respectively. One last thing, lets walk through how we use the markov mode in each tool. Lets start with john the ripper (as it is easier):

```
rm -rf ./john.pot ./john.rec ./john.log
./calc_stat -p ../../markov-train-15-1.txt stats
./john -markov:225:0:0:12 ../../unmasked.lst
```

So we remove previous work first, train based upon the control set and then run the tool. For *hashcat*, because it cannot natively accept the stdin, you have to get creative again:

```
#First create a pipe
mkfifo temp.pipe

#In one screen / session
./hashcat-utils-0.9/hcstatgen.bin ./hashcat-utils-0.9/15-1.stat < ../markov-
train-15-1.txt
./statsprocessor-0.8/sp32.bin --pw-max 12 --threshold 6 ./hashcat-utils-0.9/15-1.stat >
./temp.pipe

#In a second screen / session
./hashcat-cli32.bin -m 100 -a 0 ../../unmasked.lst ./temp.pipe
```

So what happens when we run the tests? Honestly not what I expected..

		15-1	15-2	15-3	30-1	30-2	30-3	full
hashcat 0.41 = --pw-max 12 -t 5	**found**	194	194	203	198	201	202	179
hashcat 0.41 = --pw-max 12 -t 6	**found**	745	730	716	742	750	750	581
1.7.9-7 = ./john -markov:204:0:0:12	**found**	2282	2294	2277	2262	2264	2268	2082
1.7.9-7 = ./john -markov:225:0:0:12	**found**	17727	17761	17596	17624	17549	17552	19172

Each tool attempted the expected number of passwords, and each did try passwords from 1-12 characters, but regardless - *john the ripper* consistently cracked more hashes in these tests. But lets take a look at the cracked passwords in more detail:

Training Set	**Cracked**	**Uniq**
JTR-1.7.9-7 -> lvl 204 -> all 15% tests cases combined	6853	2455
JTR-1.7.9-7 -> lvl 204 -> all 30% tests cases combined	6794	2370
JTR-1.7.9-7 -> lvl 204 -> all tests cases combined	*15729*	*2808 - (17% of total is unique)*
JTR-1.7.9-7 -> lvl 225 -> all 15% tests cases combined	53084	18737
JTR-1.7.9-7 -> lvl 225 -> all 30% tests cases combined	52725	18263
JTR-1.7.9-7 -> lvl 225 -> all tests cases combined	*124981*	*22633 - (18% of total is unique)*
Hashcat 0.41 -> t5 > all 15% test cases combined	591	249
Hashcat 0.41 -> t5 > all 30% test cases combined	601	215
Hashcat 0.41 -> t5 > all test cases combined	*1371*	*326 - (23% of total is unique)*
Hashcat 0.41 -> t6 > all 15% test cases combined	2191	925
Hashcat 0.41 -> t6 > all 30% test cases combined	2242	834
Hashcat 0.41 -> t6 > all test cases combined	*5014*	*1077 - (21% of total is unique)*

So even with the different training sets, the majority of passwords attempted were identical. What if we look at the overlap between the two tools? Well when you look for this overlap:

```
# grep -x ./jtr-all.pass -f ./hc-all.pass | wc -l
180
```

Now that is interesting. There is actually very little overlap between the two. Seeing this I ran dictionary analysis on the two results. First lets look at the *john the ripper* analysis:

```
Top 10 passwords
04-308 = 1 (0.0%)
05%link = 1 (0.0%)
08@@11 = 1 (0.0%)
0belu7 = 1 (0.0%)
0bess0 = 1 (0.0%)
0nerider = 1 (0.0%)
0roel0 = 1 (0.0%)
0trav0 = 1 (0.0%)
0tuan0 = 1 (0.0%)
1000*er = 1 (0.0%)
```

```
Top 10 base words
link = 81 (0.36%)
luke = 6 (0.03%)
linkme = 5 (0.02%)
jared = 5 (0.02%)
bang = 5 (0.02%)
anne = 4 (0.02%)
bear = 4 (0.02%)
lipw = 4 (0.02%)
kate = 4 (0.02%)
chan = 4 (0.02%)

Password length (length ordered)
6 = 6338 (28.0%)
7 = 6448 (28.49%)
8 = 7973 (35.23%)
9 = 1780 (7.86%)
10 = 92 (0.41%)
11 = 1 (0.0%)
12 = 1 (0.0%)

Password length (count ordered)
8 = 7973 (35.23%)
7 = 6448 (28.49%)
6 = 6338 (28.0%)
9 = 1780 (7.86%)
10 = 92 (0.41%)
12 = 1 (0.0%)
11 = 1 (0.0%)

        |
        |
        |
      |||
      |||
      |||
      |||
      |||
      |||
      |||
      |||
      |||
      ||||
      ||||
      ||||
||||||||||||||
00000000001111
01234567890123

One to six characters = 6338 (28.0%)
One to eight characters = 20759 (91.72%)
More than eight characters = 1874 (8.28%)
```

Now lets look the the *hashcat* analysis:

```
Top 10 passwords
abanishe1 = 1 (0.09%)
abbamoses = 1 (0.09%)
abbashanane = 1 (0.09%)
abbieseth = 1 (0.09%)
abbyandelyse = 1 (0.09%)
abbyanderer = 1 (0.09%)
abeersam1 = 1 (0.09%)
abenialata = 1 (0.09%)
abernas111 = 1 (0.09%)
abetastic = 1 (0.09%)

Top 10 base words
cara = 2 (0.19%)
setlolela = 1 (0.09%)
abbashanane = 1 (0.09%)
abbieseth = 1 (0.09%)
abbyandelyse = 1 (0.09%)
abbyanderer = 1 (0.09%)
abeersam = 1 (0.09%)
abenialata = 1 (0.09%)
abernas = 1 (0.09%)
abetastic = 1 (0.09%)

Password length (length ordered)
7 = 1 (0.09%)
8 = 47 (4.36%)
9 = 235 (21.82%)
10 = 348 (32.31%)
11 = 285 (26.46%)
12 = 161 (14.95%)

Password length (count ordered)
10 = 348 (32.31%)
11 = 285 (26.46%)
9 = 235 (21.82%)
12 = 161 (14.95%)
8 = 47 (4.36%)
7 = 1 (0.09%)

          |
          |
          ||
          ||
          ||
         |||
         |||
         |||
         ||||
         ||||
         ||||
         ||||
         ||||
        |||||
        |||||
||||||||||||||
00000000001111
01234567890123

One to six characters = 0 (0.0%)
One to eight characters = 48 (4.46%)
More than eight characters = 1029 (95.54%)
```

It seems that most of the passwords which *hashcat* found were more-then-8 characters, while *john the ripper* found most passwords in the 1-to-8 character range. Although to keep it in perspective, *john the ripper* did actually find more passwords then *hashcat* in the more-than-8 character range. What was interesting was that only *john the ripper* found "*link*" as a common base word. And as one last test, what happens if I take these attempts and try them against the uncracked hashes I have for LinkedIn? Not the full list of unmasked hashes but the ones I am working through but have not cracked yet. Well when I used the *john the ripper* cracked passwords from my test, none of them were ones I had not already cracked. When I tried the *hashcat* cracked passwords, I found 3 passwords I had not cracked yet.

Final Thoughts
Where does all this leave us at the end of the day? Well specifically speaking about the "Markov" mode implementation..

- Well *john the ripper* does seem to be the one to choose when you are starting out on cracking your hash list. It seems to find more passwords with the same training, no doubt there.
- We also see that the amount of training seems to be a case of dimishing returns, 15% more training does not mean 15% more passwords, and this is true for both tools.
- *John the ripper* found a common base word (in two variations) that *hashcat* missed
- The implementation of the markov method seems quite different in each tool since at the end of the day, there was very little overlap between the attempts generated
- The *hashcat* implementation did generate 3 passwords that had not been found through previous cracking, while *john the ripper* did not

So *john the ripper* does seem to be the tool to start with, but you should not abandon *hashcat* as it does seem to generate passwords *john the ripper* does not. Not exactly a smoking gun, but at least we now have a better idea about why and when to use each tool. And that is to be expected, this was not about choosing one over the other, but to better understand each. Try this yourself, play around, learn and have fun.