LM Hash Cracking – Rainbow Tables vs GPU Brute Force

Lately, Eric Gruber and I have been speaking about the cracking box that we built at NetSPI. Every time we present, the same question always comes up.

"What about Rainbow Tables?"

Our standard response has been that we don't need them anymore. I honestly haven't needed (or heavily used) them for a while now, as our cracking box has been able to crack the majority of the hashes that we throw at it. This got me thinking about what the actual trade offs are for using our GPUs to crack LM hashes versus using the more traditional method of Rainbow Tables.

Windows Hashes

The LAN Manager (or LM) hashing algorithm is the legacy way of storing password hashes in Windows. The replacement (NTLM) has been around for quite a while, but we still see the LM hashing algorithm being used on both local and domain password hashes.

The LM hash format breaks passwords into two parts. Each part can be up to seven characters long. If the password is seven characters or less, the second part is just a blank LM hash. All of the alphabetical characters are converted to upper case, as the LM hash standard is case insensitive. Case sensitivity is stored in the NTLM hashes.

In the example below, the hash for the password "QBMzftvX" is broken into two parts (QBMZFTV and X). You will also see that all of the cleartext characters of these LM hashes are upper-cased.

c88062822433f468	bcbb464a6f1414b9
Characters 1 to 7 Cleartext:QBMZFTV	Characters 7 to 14 Cleartext: X

For the purpose of this blog, I'll only be covering the trade offs of using Rainbow Tables for LM hashes. There are NTLM Rainbow Tables, and they can be used with great success, but I won't be covering the GPU/Rainbow Tables comparison for NTLM in this post.

Rainbow Tables

Traditionally, LM hashes have been attacked with Rainbow Tables. It's easy to create large tables of these password/hash combinations for every possible LM hash, as you only have to create them for one to seven-character combinations. Once you've looked up the hash halves in the tables, you toggle cases on the letters to brute force the password for the case-sensitive NTLM hash. This method works well, but disk reads can be slow and sometimes your computer is busy doing other things, so adding in LM table lookups may slow the rest of your system down. Some great advances have been made by multi-

threading the table lookups. This ended up being one of the pain points in writing this blog, as I didn't have the correct multi-threaded Rainbow Table format to use with rcrack_mt. Additionally, the table lookups were helped by the fact that I was using an SSD to house the Rainbow Tables. I included stats for rcrack_mt table look ups for comparison in the table at the end.

There are two major tradeoffs with using Rainbow Tables. The primary one being disk space. The Rainbow Tables themselves can take up a fair amount of space. I know that disk space is relatively cheap now, but five years ago this was a much bigger deal. The second tradeoff is the time it takes to initially generate the tables. If you are not getting the tables off of the internet (also time consuming), you might be taking days (possibly months) to generate the tables.

GPU Cracking

I really can't give enough credit to the people working on the Hashcat projects. They do a great job with the software and we use it every day. We use oclHashcat to do GPU brute force attacks on the one to seven character LM hashes. Once cracked, the halves are reassembled and a toggle case attack is run with hashcat (CPU version). Using GPUs to do the brute forcing allows us to save some disk space that would typically be used by the LM tables, and it allows us to offload the cracking to our centralized system. Since we've scripted it all, I just pull the LM and NTLM hashes into our script and grab a cup of coffee. But does it save us any time?

Hybrid GPU Rainbow Tables

There are programs (RainbowCrack) that allow for a hybrid attack that uses GPU acceleration to do the Rainbow table lookups. I've heard that these programs work well, but RainbowCrack only supports the GPU acceleration through Windows, so that means we won't be using it on our GPU cracking rig.

The Breakdown

For this test, I generated a set of 100 LM/NTLM hashes from randomly generated passwords of various lengths (a mix of 6-14 character lengths). Cracking with Rainbow Tables was done from my Windows laptop (2.70GHz Intel i7, 16 GB RAM, SSD). GPU cracking was done on our GPU cracking box (5 GPUs).

Method	Cracked	Time
Rainbow Tables (OphCrack*)	99/100	24 Minutes 5 Seconds
oclHashcat/CPU Hashcat	100/100	18 Minutes 56 Seconds
Rcracki (multithreaded**)	100/100	5 Minutes 40 Seconds

*OphCrack 3.6.0 run with the XP Special Tables

**Rcracki_mt running with 24 threads

So after all of this effort, I can't totally justify saying that using oclHashcat/Hashcat is faster for cracking LM hashes, but given our setup, it's still pretty fast. That being said, if you don't have your own GPU cracking rig, you will definitely be better off using Rainbow tables, especially if you multi-thread it on a solid state drive.