Team Hashcat: Write-up CMIYC 2015

Team Hashcat would like to congratulate every team that participated in CMIYC 2015. We were also intrigued to see how CynoSure has matured into a real "PRO" team. The "Street" class competitors were also very active and following them was exciting. It appears that the contest is getting more and more attention, thanks to KoreLogic's effort, as well as increasing global trends and awareness of password and password-related matters.

KoreLogic's message to us was that we needed multibyte support in Hashcat. We understood that and, as the tool's developers, we will look into how this can be accomplished whilst still maintaining performance. Implementing a rule-engine that is capable of supporting multibyte characters for slow hashes is not a problem and we foresee it being added in the future.

We hope to see everyone again for CMIYC 2016!

Preparation

To mitigate the challenging task of information sharing and collaboration we used our own home-grown tool; list-condense (which we have mentioned in previous contests), multiple IRC channels, a Teamspeak Server and Google Drive.

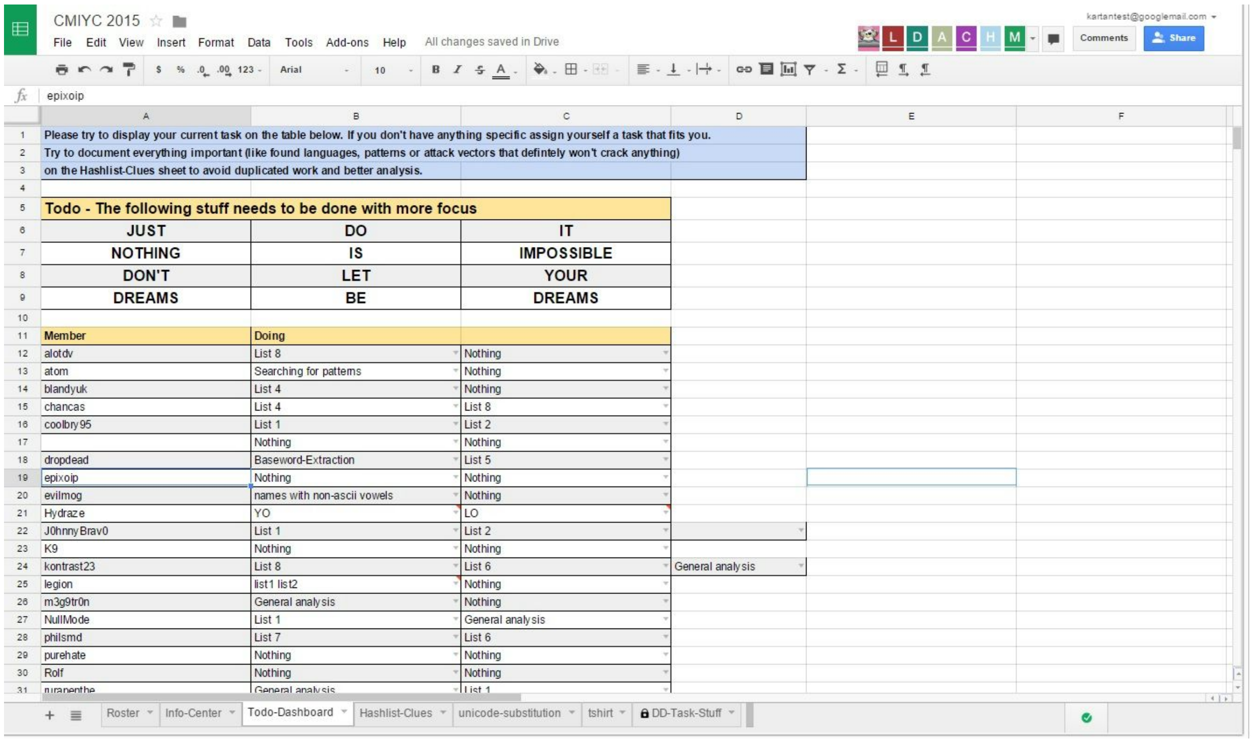
To enhance team collaboration we decided to try a different approach than we had in previous contests. The management of subgroups has shown to have a certain overhead and has proven to not be as efficient as we had thought. We opened up the SCRUM approach and let everyone decide on their own what to do and to just give outlines in which scope certain tasks should be. To achieve that, we opted for a google spreadsheet (as shown below) and assign a team manager to update the available tasks. Furthermore, members updated their findings and impressions on a second sheet to minimize redundancy.

We dropped the idea of our previous organizational groups and decided that every team member should work on all hash lists. The reason is that, typically, during CMIYC contests, the plaintext patterns were shared across all of the hash lists. This means they are not unique per hash list as you would find them in the Positive Hack Days (PHD) contest. Therefore it made sense not to create specific groups where specific team members are assigned to crack specific hash lists.

Team Members

alotdv	J0hnnyBrav0	Szul
atom	K9	tehnlulz
blandyuk	kontrast	The_Mechanic
chancas	legion	T0XIC
coolbry95	m3g9tr0n	undeath
dropdead	NullMode	unix-ninja
EvilMog	philsmd	Xanadrel
Hydraze	RuraPenthe	xmisery

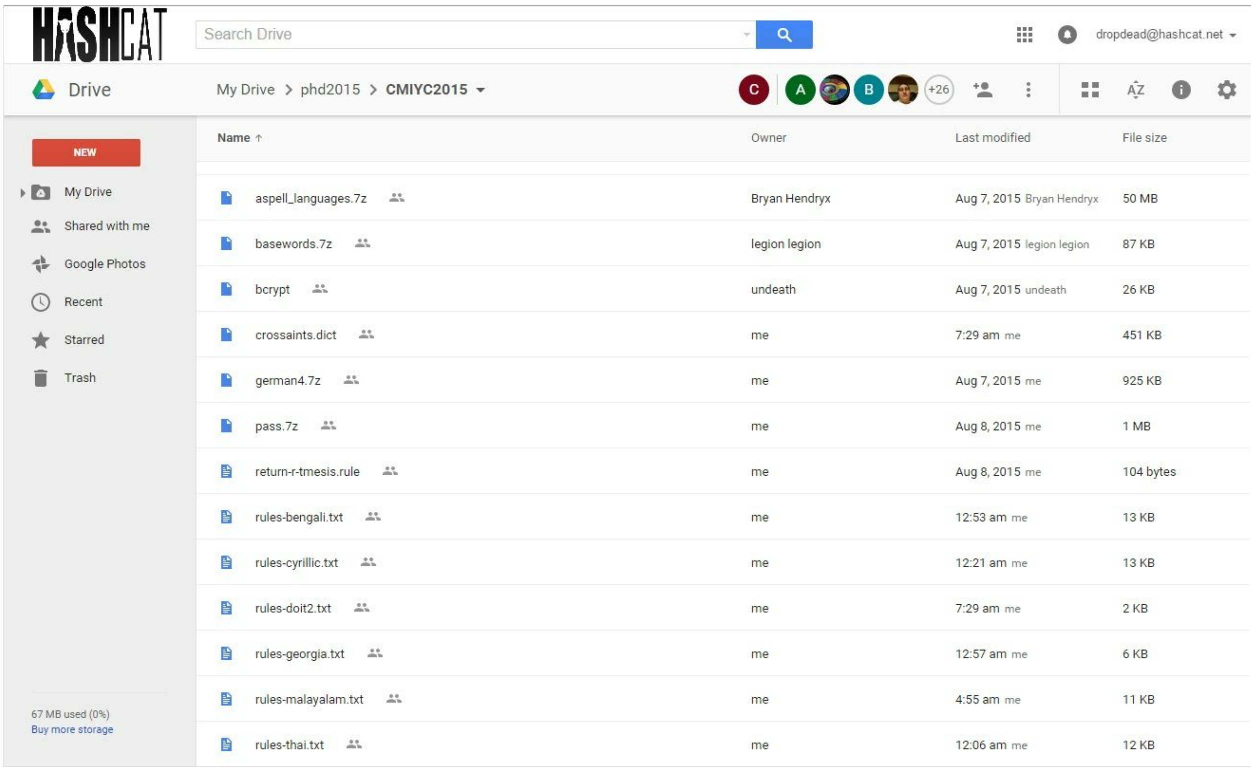
Spreadsheet for Information Sharing and Coordination



fx epixoip

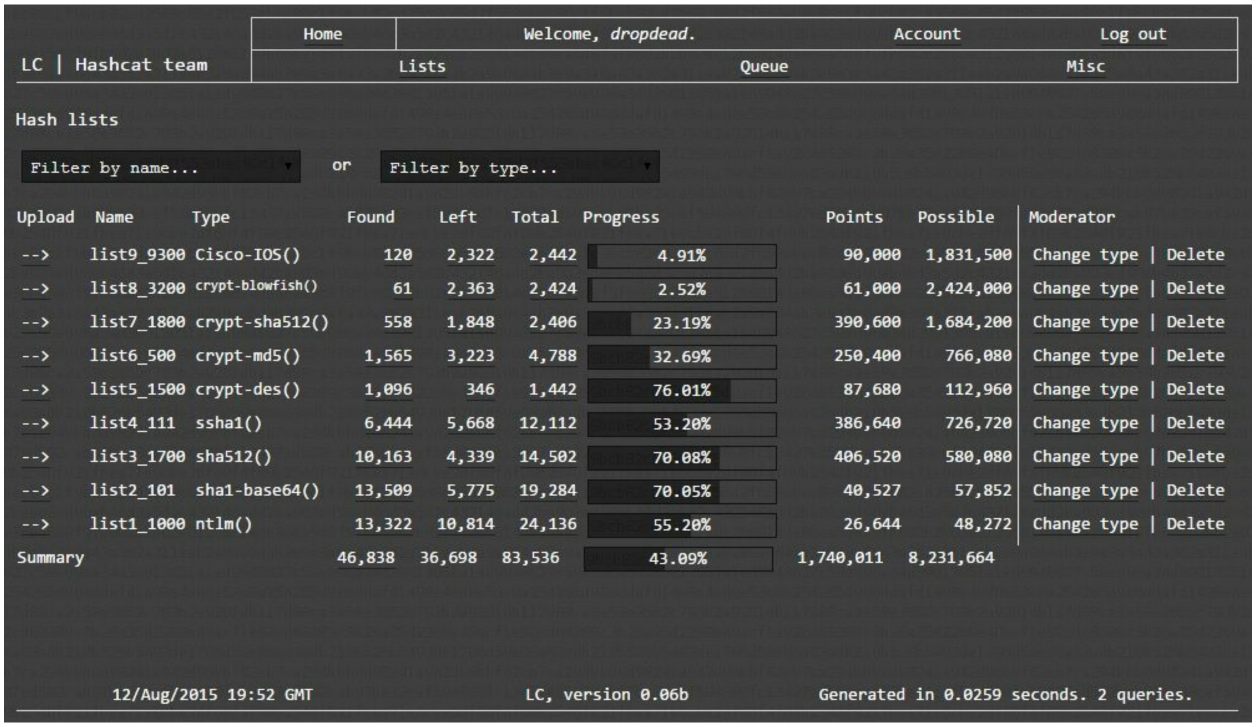
	A	B	C	D
1	Please try to display your current task on the table below. If you don't have anything specific assign yourself a task that fits you.			
2	Try to document everything important (like found languages, patterns or attack vectors that defintely won't crack anything)			
3	on the Hashlist-Clues sheet to avoid duplicated work and better analysis.			
4				
5	**Todo - The following stuff needs to be done with more focus**			
6	**JUST**	**DO**	**IT**	
7	**NOTHING**	**IS**	**IMPOSSIBLE**	
8	**DON'T**	**LET**	**YOUR**	
9	**DREAMS**	**BE**	**DREAMS**	
10				
11	**Member**	**Doing**		
12	alotdv	List 8	Nothing	
13	atom	Searching for patterns	Nothing	
14	blandyuk	List 4	Nothing	
15	chancas	List 4	List 8	
16	coolbry95	List 1	List 2	
17		Nothing	Nothing	
18	dropdead	Baseword-Extraction	List 5	
19	epixoip	Nothing	Nothing	
20	evilmog	names with non-ascii vowels	Nothing	
21	Hydraze	YO	LO	
22	J0hnnyBrav0	List 1	List 2	
23	K9	Nothing	Nothing	
24	kontrast23	List 8	List 6	General analysis
25	legion	list1 list2	Nothing	
26	m3g9tr0n	General analysis	Nothing	
27	NullMode	List 1	General analysis	
28	philsmd	List 7	List 6	
29	purehate	Nothing	Nothing	
30	Rolf	Nothing	Nothing	
31	rurapenthe	General analysis	List 1	

Google Drive - Our Temporary Information Storage



Name	Owner	Last modified	File size
aspell_languages.7z	Bryan Hendryx	Aug 7, 2015 Bryan Hendryx	50 MB
basewords.7z	legion legion	Aug 7, 2015 legion legion	87 KB
bcrypt	undeath	Aug 7, 2015 undeath	26 KB
crossaints.dict	me	7:29 am me	451 KB
german4.7z	me	Aug 7, 2015 me	925 KB
pass.7z	me	Aug 8, 2015 me	1 MB
return-r-tmesis.rule	me	Aug 8, 2015 me	104 bytes
rules-bengali.txt	me	12:53 am me	13 KB
rules-cyrillic.txt	me	12:21 am me	13 KB
rules-doit2.txt	me	7:29 am me	2 KB
rules-georgia.txt	me	12:57 am me	6 KB
rules-malayalam.txt	me	4:55 am me	11 KB
rules-thai.txt	me	12:06 am me	12 KB

List Condense



Home	Welcome, *dropdead.*	Account	Log out

LC | Hashcat team

Lists Queue Misc

Hash lists

Filter by name... or Filter by type...

Upload	Name	Type	Found	Left	Total	Progress	Points	Possible	Moderator
-->	list9_9300	Cisco-IOS()	120	2,322	2,442	4.91%	90,000	1,831,500	Change type | Delete
-->	list8_3200	crypt-blowfish()	61	2,363	2,424	2.52%	61,000	2,424,000	Change type | Delete
-->	list7_1800	crypt-sha512()	558	1,848	2,406	23.19%	390,600	1,684,200	Change type | Delete
-->	list6_500	crypt-md5()	1,565	3,223	4,788	32.69%	250,400	766,080	Change type | Delete
-->	list5_1500	crypt-des()	1,096	346	1,442	76.01%	87,680	112,960	Change type | Delete
-->	list4_111	ssha1()	6,444	5,668	12,112	53.20%	386,640	726,720	Change type | Delete
-->	list3_1700	sha512()	10,163	4,339	14,502	70.08%	406,520	580,080	Change type | Delete
-->	list2_101	sha1-base64()	13,509	5,775	19,284	70.05%	40,527	57,852	Change type | Delete
-->	list1_1000	ntlm()	13,322	10,814	24,136	55.20%	26,644	48,272	Change type | Delete
Summary			46,838	36,698	83,536	43.09%	1,740,011	8,231,664	

12/Aug/2015 19:52 GMT LC, version 0.06b Generated in 0.0259 seconds. 2 queries.

Contest Antics

This time the hash list was not overloaded with tons of different algorithms, which was refreshing because it brought back the original spirit of a hash cracking contest: Pattern detection and finding out how to use tools to produce matching plains. We were able to concentrate on the cracking part itself and not spend significant periods of time finding out which hashing algorithms were used, converting them into a format that hashcat could use, or adding them to our management-system. Many thanks for that!

The Hash Cracking Itself

The biggest problem facing the competitors was how to work around the many variations of unicode characters found during the competition. One of the techniques used that proved to be successful was to insert multi-byte characters into an ASCII plaintext string via hashcat using the tmesis.pl tool from hashcat-utils. Once a charset and a base dictionary could be identified we generated the rules with tmesis.pl run them with hashcat.

Tmesis.pl reads a word line by line from a file and splits the word into bytes. Tmesis.pl creates rules that effectively insert this word into a plaintext password candidate using hashcat’s rule engine (like the “a” rule in John The Ripper but does not append it). Tmesis.pl does not handle multibyte unicode characters as single characters but handles them as individual bytes. This technique is typically used to insert whole words into password candidates, but worked to insert multibyte characters into passwords well.

Another efficient cracking method was using the increase/decrease rules supported by hashcat and oclHashcat. UTF-8 follows some logic in it’s character ordering, similar to the ASCII character set. This functionality is useful because it allows you to generate words that do not currently exist in a wordlist. For example, a wordlist may contain the word “password45” but not “password46”. By using this functionality within the rule engine, it is possible to generate the missing candidate because the character ‘6’ follows the character ‘5’ in the ASCII table. After doing this, it was possible to identify that KoreLogic had inserted randomly selected UTF-8 characters into random places in the given passwords. That together means we can assume the byte-wise increment or decrement of any random position of a to-be-generated plain, even if it’s a UTF-8 character, which is likely to crack something successfully. For slow hashes, this technique generates too many candidates for it to be a viable attack vector. However, this technique was effective at cracking the fast hashes. It was helpful in discovering which UTF-8 characters were used to generate the passwords and also helped in the identification of other patterns that existed within the hashes.

Patterns Found

Korean Hangul Alphabet (bieup, chieut, digeut, giyok, hieut, ieung, …)

To make use of this pattern we used combinator.bin and combinator3.bin from hashcat-utils. This is like a Brute-Force attack since it combines all variations of the alphabet with all of them again and again. Of course this technique only worked for the fast hashes. We create 3 wordlists, were each wordlist was of the size of the alphabet ^ N to have a good base for the -a 1 combinator attack-mode in oclHashcat itself. After that we used those wordlists for the attack, for example length 3 plus length 1 to crack 4-length words of that alphabet. For the slow hashes we simply used plain dictionary attack up to length 3.

Japanese ぱす (means “pass” and then replaced substrings “pass”)

Base words: rockyou. There is no way to replace whole substrings with whole substrings in the rule-engine. But since we tried to crack slow hashes only with this pattern, it was enough to simply *sed* our wordlists.

German words (umlauts only)

These were kinda hard because there are so many different german dictionaries floating around the net and it was not easy to find the one which KoreLogic was using. Also, some of them were encoded using ISO-8859-1, so we had to make sure to re-encode them to UTF-8 before trying to attack with it. At least we thought so because it was clear that KoreLogic tried to tell (we could see this because of the already cracked plains) that this contest has an internalization focus and to be true internationalization you need at least UTF-8.

Arabic numbering (written form)

A very easy pattern to detect for the human eye. Once we saw some appear we translated them and the pattern almost immediately presented itself. A quick combination of dictionaries containing all the possible variations brought quite a few hits.

Insert of multibyte char into english plain

This is how rules-trash.rule was made: A constant monitoring of all the found plains were necessary to see those patterns. Luckily a bunch of them got cracked by chance which we picked up after already realizing the focus on multibyte chars (without the hints from the contest t-shirt, which we got to see 12 hours before the end of the contest). We quickly realized that there must have been a lot more charsets in use and in a lot more combinations than we thought initially. After a first test with the already identified charset Malayalam inserted via tmesis into rockyou got quite a few hits on the fast algos. After a little refinement rules-trash.rule came about that covered all the identified characters from Malayalam, Arabic, Thai, Greek and Currency Symbols. The selection of the dictionary to apply the ruleset on was quite a bit harder. We identified WikiTop50k as a good start but it turned out not to hold all the base words. At the end of the contest we were still in the middle of finding the right base words.

Thoughts and Feedback

The team was very impressed with the competition for a number of reasons:

- A manageable number of hash lists: This made the competition easier to manage as a team and allowed us time to concentrate and focus on the lists we had rather than spend time trying to organise the team around a larger number of hash lists
- Standard algorithms: The use of fairly common algorithms meant that the competition time could be spent cracking passwords instead of needing the hashcat developer to work on developing new algorithms
- Using largely unicode based passwords was both interesting and useful to the team as password crackers. It was a welcome change to standard ASCII passwords and allowed the team to hone their skills, develop tools and get familiar with cracking UTF-8
- The team members scores were relatively close to each other, and all of the members did well to contribute
- “DO IT, JUST DO IT” became the theme of the competition for the team and the default answer to questions in general

Here’s Team Hashcat signing off for this year’s CMIYC.
Thanks to all who participated and to KoreLogic!
Until next year...