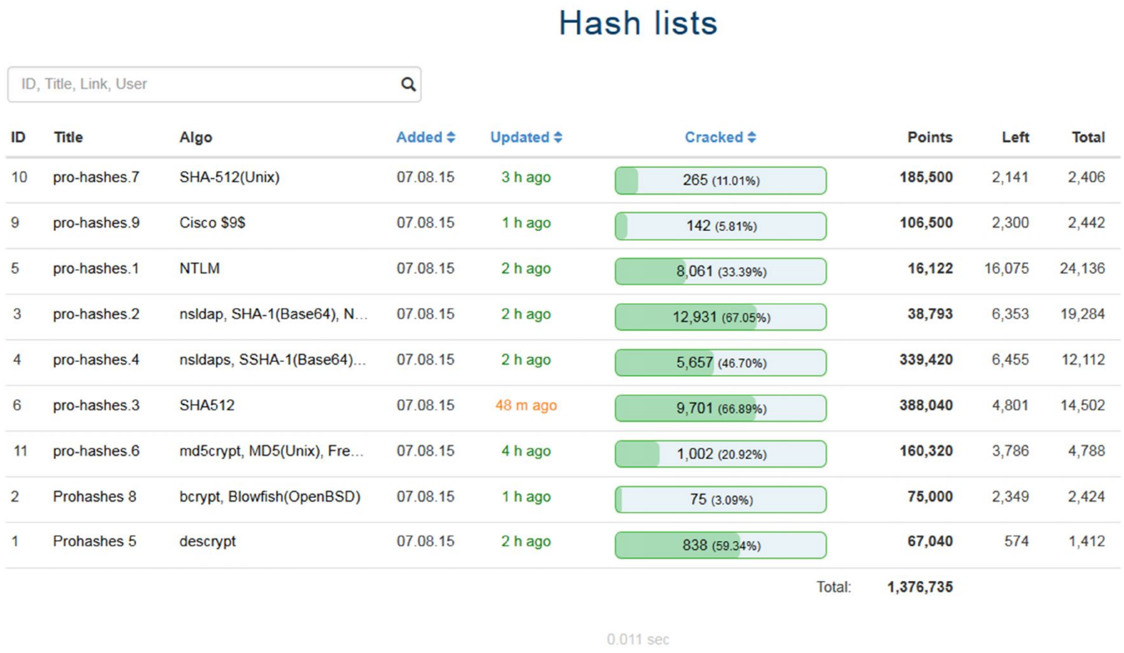
CynoSure Prime

Crack me if you can

2015 write-up

12/08/2015



Hash lists

ID	Title	Algo	Added ⬍	Updated ⬍	Cracked ⬍	Points	Left	Total
10	pro-hashes.7	SHA-512(Unix)	07.08.15	3 h ago	265 (11.01%)	**185,500**	2,141	2,406
9	pro-hashes.9	Cisco 9	07.08.15	1 h ago	142 (5.81%)	**106,500**	2,300	2,442
5	pro-hashes.1	NTLM	07.08.15	2 h ago	8,061 (33.39%)	**16,122**	16,075	24,136
3	pro-hashes.2	nsldap, SHA-1(Base64), N...	07.08.15	2 h ago	12,931 (67.05%)	**38,793**	6,353	19,284
4	pro-hashes.4	nsldaps, SSHA-1(Base64)...	07.08.15	2 h ago	5,657 (46.70%)	**339,420**	6,455	12,112
6	pro-hashes.3	SHA512	07.08.15	48 m ago	9,701 (66.89%)	**388,040**	4,801	14,502
11	pro-hashes.6	md5crypt, MD5(Unix), Fre...	07.08.15	4 h ago	1,002 (20.92%)	**160,320**	3,786	4,788
2	Prohashes 8	bcrypt, Blowfish(OpenBSD)	07.08.15	1 h ago	75 (3.09%)	**75,000**	2,349	2,424
1	Prohashes 5	descrypt	07.08.15	2 h ago	838 (59.34%)	**67,040**	574	1,412
					Total:	**1,376,735**		

0.011 sec

A screenshot from our Hash Management System “TeamLogic” dashboard

AMD	dook	splitter	winxp5421
blazer	gearjunkie	tony	wonder
casha	hops	usasoft	
cvsi	noob	waffle	

14 participating members, 12 active from all over the world

Before the contest

Prior to the contest several MPI clusters were setup rocking the latest Bleeding-Jumbo, props to the JtR community for their marvellous work on John-the-ripper. Our hash management system 'TeamLogic' was also updated for improved dynamic hash format support as we expected really exotic algorithms to be added to the mix. We used a TeamSpeak3 server for primary communications and a forum reserved for miscellaneous uses.

During the contest

After seeing the algorithms we decided to reserve all CPU resources for the slow crypt functions including SHA-512(unix), scrypt & bcrypt. All other algos including NTLM, SHA512, md5(crypt), descrypt and nsldap/s were predominately run on GPU. While the hashes with known hash types were being loaded into our hash management system, we were able to leverage the simultaneous hash algorithm parallel processing capability of MDXfind to rapidly identify ambiguous hash types. A very early analysis of our initial cracks revealed that we were dealing with foreign passwords, through the use of MDXfind as a wordlist parser, we were able to quickly filter through our lists selecting only passwords with foreign characters. Dictionary attacks against the slower algorithms were immediately started early on and we had hits across all algorithms roughly 3 hours into the contest.

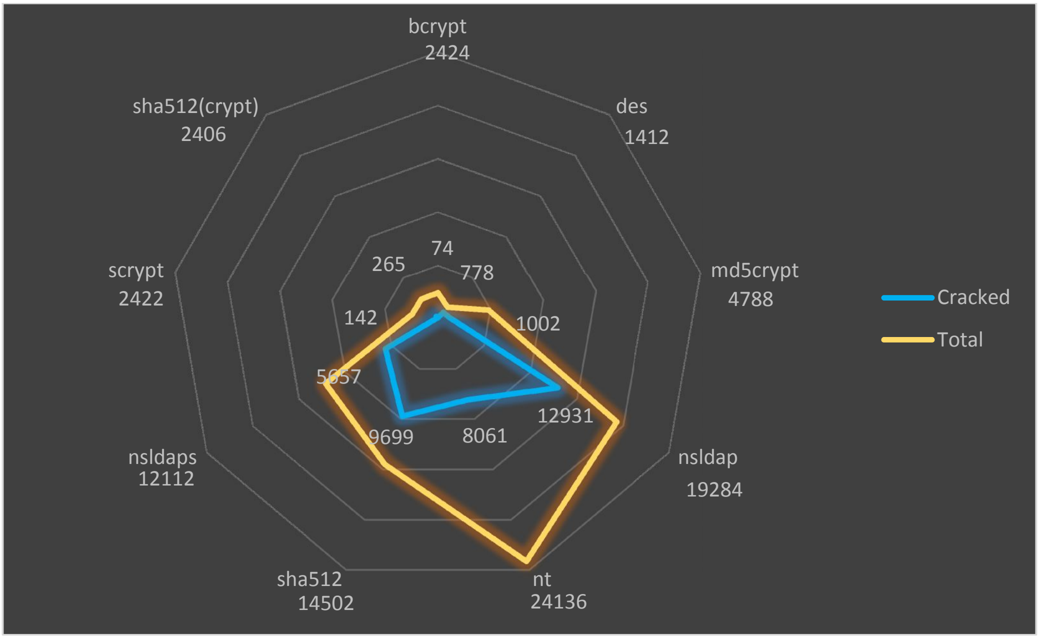
Upon noticing foreign passwords, our developer immediately enforced strict HEX$[] encoding on all algorithms to ensure optimal compatibility for plaintexts. Thankfully, our system was already UTF-8 compliant so we didn't have issues at all submitting/exporting bad plains, we didn't hear anything from Korelogic, so assumed we were doing it right.

The basic approach was simple, identify patterns in the quicker algos and slowly work our way up in algorithm complexity, applying the same rules and patterns. Two algorithms namely NTLM and MD5(crypt)/des(crypt) weren't of particular interest to us, as they were either too low in points or not worth their crack value, these were not a high priority for us. However, we couldn't help ourselves from the slow bcrypt hashes especially since they were 1000 points each.

Initial attacks using ?b?b masks were carried out to test the waters, once enough information was gathered we switched to a much more efficient method of emulating specific mask attacks by using pre-generated lists in combination attacks. Similarly we were able to modify our rule processor 'Rulify3' to accommodate for multi-byte character replacement. Custom rule processors weren't really necessary, by understanding the basic principles of UTF-8 encoding and how a rule processor functions, we were able to successfully emulate UTF-8 characters, and incorporate them into our attacks by crafting special rules which could be used with standard rule processors.

Several hours into the competition our team *automagically* divided into two groups, one group attacked whatever they wanted with whatever they wanted. The second group focussed on maximizing and optimizing pattern attacks and crafting specialized attacks, but more importantly ensuring there was as little resource idling as possible. Resources were continuously shared between the two groups. Without the tremendous efforts of group one, group two wouldn't have been able continuously obtain unique data for analysis to refine attacks. This was only made possible by combining the different skillsets of each member and working in unity.

For some visuals please refer to the figure below which depicts the hash/crack distribution spread for our team. Orange denotes total hashes for each algorithm while blue shows indicates the number of cracks carried out by our team.



Patterns

UTF-8 (ASCII look-a-like charset) / &%%#

These were detected and 3 dictionaries where produced, singles, doubles, triples. Using a combination attack mode, this allowed us to 'bruteforce' the full keyspace for 'length 4' and 'length 5' of this mixed ascii + UTF-8 keyspace giving us very successful results across all algorithms. This one was slightly trickly as it appeared Korelogic had also mixed in standard asci characters into the keyspace. We did not have time to thoroughly explore the other look-a-like alpha characters and only dealt with the symbols and numbers, however did note their presence.

Korean alphabet Hangul / `kieukssangdigeutchieut`

Similar to the above, we produced singles, doubles and triples of the Korean Hangul alphabet and used them in combination attacks across all algorithms for combinations of 2,3 & 4.

German, Finnish, French, Danish and other European words / `attaché`

Rulify3 was modified to incorporate the '~' rule (~XYZ, swaps X for YZ) supporting 1 to 2 byte character swaps, our wordlists were run through this tool and then dictionaries generated were uniqued and run through as dict attacks. Existing dictionaries where filtered either with Unified List manager (ULM) or MDXfind to select for extended ascii or UTF-8 containing character containing words.

Other Tibetan, Chinese, Japanese, Cyrillic, Arabic charsets

UTF-8 charsets were extracted from ourfounds, these charsets were then expanded for better coverage. These UTF-8 encoded characters were then reversed as their byte form followed by expansion using ULMs prefix everywhere function, transforming them into ‘multi-byte insertion rules’ which were compatible with existing non-UTF-8 aware rule processors. We used this technique to target and insert UTF-8 characters into various positions across our wordlists for hybrid rule attacks. We found these rules worked well when paired with small standard clean dictionaries for target languages. We also yielded success when stacking the ‘c’ rule.

Overwrite rules were generated in a similar manner, instead we would insert dummy characters into the wordlist to make up for bytes then overwrite them. However due to the pairing of UTF-8 characters with specific groups of words it was more optimal to use the aforementioned Rulify3 rule processor and the rule sets with specific wordlists.

Other patterns / Rage¬Meze

Word UTF-8 Word, Word UTF-8 UTF-8 Word pattern was also detected, however due to time constraints we were not able to fully scan this small pattern across every algorithm. It however appeared that there were only a limited number of word combinations.

Highlights

Some memorable moments would be; receiving a picture from one of the members who was participating in the contest while attending a wedding. It really gave us a good laugh to see his dedication. Another instance one member gave us full reign of their GPU farm and we joked that they would probably have a surprise power bill after we were finished, considering the number of gpus and their lack of idle time…. only to our surprise it was business as usual for them.

Several hours in someone asked… “so when are the challenges coming out?” “It’s okay... give them twelve hours and we might have something”, someone else replied.

We didn’t worry too much about the tweeted hints, until later in the contest when we asked our rep to pay Korelogic another visit to obtain some intel (thanks dook!!). After promptly receiving the MD5 hashes and just as quickly cracking them, we had a good laugh since it wasn’t like we didn’t know all the passwords were foreign at that point.

After the contest

Once the contest ended one of the members pointed out that not all crackers were capable of cracking the UTF-8 encoded NTLM’s properly, while some other programs did. We suspect this is due to byte-order reversal and zero adding in alternating steps as a ‘cheat’ UTF-16LE conversion in the MD4 digest, for optimization purposes. This would have resulted in incorrect UTF-8 to UTF-16LE conversion leading to hashes being missed, JtR did not appear to suffer this issue.

Thoughts

We may not have had the biggest team and probably didn’t possess largest amount of compute power. Considering the fact that we are a relatively new team, we were able to work cohesively and use our resources both efficiently and effectively resulting in placing 2nd for KoreLogic’s “Crack me if you can 2015”. We have thoroughly impressed ourselves; having beaten the former champions John-users and InsidePro, we have demonstrated that we are worthy opponents. Congratulations to Team Hashcat for taking the 1st place win. Congratulations to “Shining Ponies”, “Toil” and “ICantBelieveItsNotButter”, you guys also did an impressive job. If I recall correctly, at one stage one of the street teams were actually scoring higher than one of the pro teams.

12/08/2015

Thank you to KoreLogic for hosting ‘Crack me if you can” for the 6th time, we understand how difficult it is to come up with something challenging. I guess UTF-8 password were simply inevitable, we really appreciate your prompt replies preparedness and visuals on the statistic pages.

#FOLOW_**US #**JOIN_**US #**LOVE_**US #**HATE_**US #**CONTACT_**US @CynoPrime**

Twitter: @CynoPrime
Blog: cynosureprime.blogspot.com
Email: cynosureprime@gmail.com

Resource Package