Hashrunner Writeup 2015

A massive congratulations to the top tier teams Hashcat, John-Users and of course InsidePro, you guys set the bar really high and kept us thoroughly entertained, while we sat comfortably in 4th place watching you battle it out epically.

If you are not already aware, we are a newly assembled 16 man team comprised of

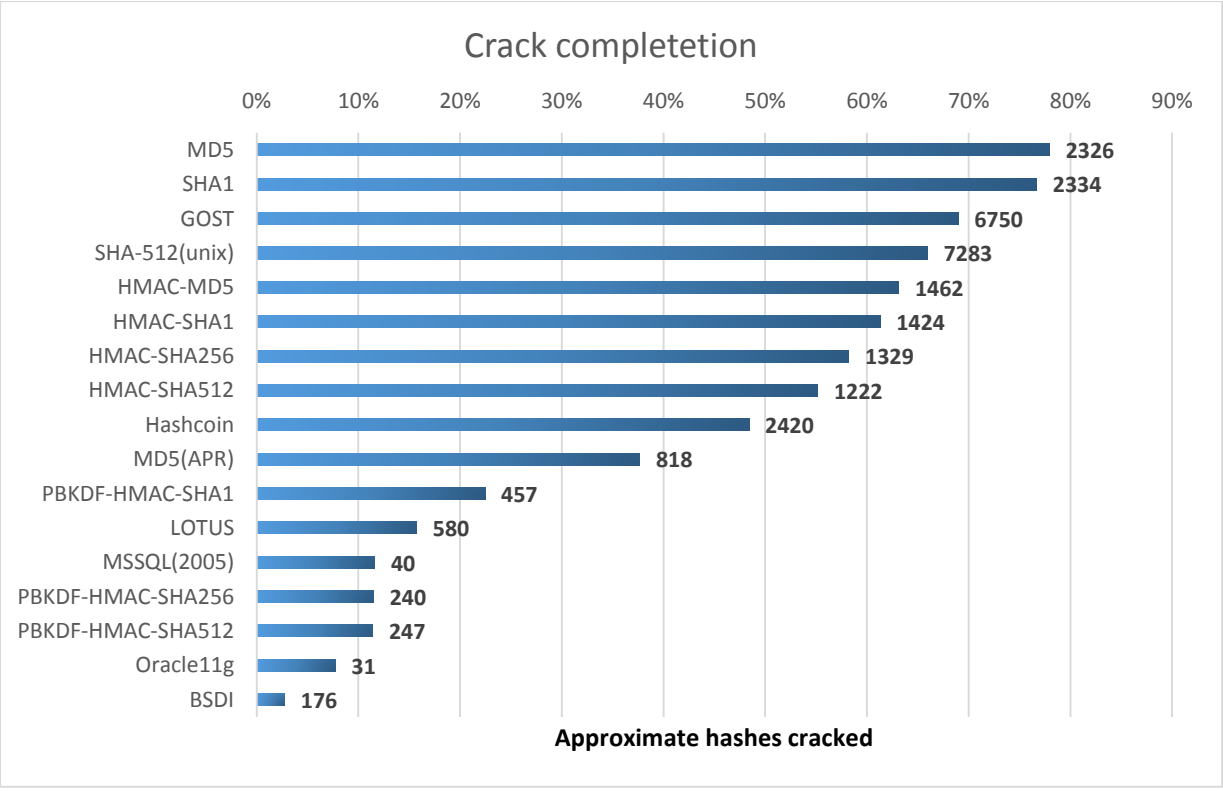
Amd	gearjunkie	jugganuts420	usasoft
blazer	Hartwell	Milzo	Waffle
casha	Hash-IT	splitter	winxp5421
cvsi	hops	Tony	Wonder

Only a few members from our team had competed in something like this before, so it was mostly a learning experience and a bonding exercise for everyone.

We expected the contest to be closer to the conference date so it was a little unexpected to have it so early, though does make sense especially as conference tickets are given out as prizes. It seemed slightly rushed from the announcement post on Hashcat forum to the actual contest day and didn't leave us much time to prepare.

After hearing that the PHC hash algorithms may be used in the contest, crackers for Lyra2, Argon, Parallel & Battcrypt & Pomelo were coded but were not used (see resources). We used TeamSpeak as our primary communications channel, backed by a forum and used TeamLogic as our hash management system.

For some visuals here is a rough breakdown of the hash crack progress, LM omitted as they were fully bruteforced.



Challenge 1: Pomelo

As we had prepared a pomelo cpu cracker, we figured all we had to do was reformat the hashes and load them into our cracker. Wrong, we tried to match the sample candidates generated from the source provided with our candidate generator and noticed something was odd. It appeared the hex string was not printed correctly, which wasn't a problem as we could tweak our cracker to produce the same variable hex string lengths. Further inspection of the source showed that instead of the password string being hashed, it was actually the pointer to the string, with the actual string length used. It appeared that brute forcing using the address space was not feasible as the submission still would not have succeeded, since we would still need a 'valid' password to match those generated.

Challenge 1: Pufferfish

A brief test was conducted with our candidate generator and it appeared pufferfish was a slow algo, considering the amount of points that was award we did not pursue this algorithm further.

Challenge 2: Scrypt

We extracted all the words from the lyrics and tried it against every single hash using hashcat CPU… 0 results. The entire team spent the whole last day on scrypt without getting results after applying advanced mutations, 1337, toggle, using the sentences, best64, translating the text into 50 languages and using aforementioned mutations in varying combinations. We suspected a bug in hashcat, so supplied a sample candidate which hashcat successfully cracked. I think the dolphins will haunt us forever, no thanks for the fish!!!

Challenge 7: Invuln sha512(unix)

Bug#1 was spotted in the algorithm in which the salt was derived from the password, a reference php implementation was produced to abuse this bug, which was then translated in to python and finally the code was ported to c and implemented to a 'salt cracker' (see resources). By using known plaintexts which generated the salt collision against its hash pair, we were able to reduce the work factor considerably, instead of performing 5000 rounds of sha512(unix) on each plaintext.

Bug#2 was identified shortly after, this allowed the full keyspace len1-12 to be quickly bruteforced by reducing the steps necessary in the salt generator to filter password collisions. A reference php implementation was produced (see resources) and again translated to python. This bug exploited various aspects of the salt derivation function. We were able to filter out password candidates and gain enormous speedups for specific password lengths, through pre-computation and skipping unnecessary steps in the salt generator. By performing a full keyspace bruteforce including unicode, we were able to recover over 65% of plaintexts for this challenge.

Challenge 8: GOST

This is where quick easy points could be easily obtained. As there was no publicly available cracker, a custom solution had to be implemented. Waffle was able with some modifications add support for hashes digested with (GOST R 34.11-2012, RFC-6986). Adding support was not difficult; furthermore as MDXfind already supported variable length hash formats, we were able to crack both -32 and -64 GOST hashes simultaneously. We did encounter submission issues, as there were a mix of UTF-8/16 passwords and hashrunner didn't support $HEX[] format, this was resolved through a conversion script.

Challenge 9:

An early python implementation was coded for this challenge; this later evolved into a multi-threaded python solution and ultimately ended up being coded in c. There is no known way of solving this challenge quickly, except for iterating through each step and applying the correct hash then searching through all possible candidates. The completion of each step resulted in the 'password' to grow in size and therefore increasing the difficulty factor. Each step required at most 1,081,516,448 hash calculations making this challenge rather complicated. The final password was about 35k long.

By pre-calculating partial hashes, we were able to reduce the total hashing time substantially and generating a solved step, took roughly 30 seconds. With only a few hours left in the contest, a cluster of 88 cores were assigned to this task. Unfortunately due to a race condition in the code, we were unable to complete the run and only solved about 2500 hashcoins.

Patterns and other breakthroughs

Tony was able to catch on early with the rules u, c, d, $2$0$1$1, $2$0$1$2, $2$0$1$3, $2$0$1$4, $2$0$1$5 and the rules worked well when they were combined as well. We also got breaks with the Italian and chinese wordlists.

Game plan

To keep everything as laid back as possible, we took the free flow approach and did not assign individuals specific tasks. Instead, anyone could accept tasks distributed by the system or do what they were comfortable with, be it, cracking with gpu/cpu, coding, analysing, scripting or cracking jokes over the teamspeak channel and having a laugh. In the last 24 hours we did however advise everyone to focus on scrypt, sha512(unix) and GOST hashes, as cracking the other algorithms were simply unfeasible considering the amount of points rewarded for them. Our main goal was to ensure everyone had fun and remained motivated. We also tried to introduce different techniques to the new guys so they could understand the various approaches to competitive hash cracking.

Closing thoughts

A thank you to the organisers of hashrunner for giving us the opportunity to refine our skills. Thanks for putting up with our annoying emails and thanks for resolving our problems promptly. We noticed that a fair bit of coding was involved at our end during the contest, compared to previous years where it was more cracking/analysis based. It certainly mixed things up and sharpened our minds.

Despite being a team put together in a very short period of time with most of us not having any prior competing experience, I think our team faired pretty well, especially since we competed against renowned hash cracking groups. Overall it was an awe-inspiring contest, most importantly our team shared both a great time and lots of fun and had incredible team spirit. Overlooking the lack of sleep this contest caused, we hope to compete again soon.

WANT TO⁉ #JOIN_US, #HATE_US, **#LOVE_US**, feel free to **#CONTACT_US @CynoPrime**

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

Resource Package