

CRACK ME IF YOU CAN

2023 DEBRIEF



Team Hashcat
18 August 2023

Table of Contents

<u>About the Contest</u>	<u>3</u>
<u>About the Team</u>	<u>3</u>
<u>Organization & Planning</u>	<u>3</u>
<u>Software Stack</u>	<u>4</u>
<u>Hardware Stack</u>	<u>5</u>
<u>Competition Narrative</u>	<u>7</u>
<u>Street Hashes</u>	<u>10</u>
<u>In Conclusion</u>	<u>11</u>

About the Contest

Crack Me If You Can (CMIYC) is an annual password cracking competition created and hosted by KoreLogic Security. It is most frequently held during the annual DEF CON hacker conference in Las Vegas, NV (with two historical exceptions during DerbyCon in Louisville, KY). This competition involves cracking cryptographic password hashes and password-protected files such as documents, archives, and disk images from various artificial sources, with plaintext values typically following a common pattern or theme. Point values for each successful crack typically scale with the difficulty of the password hashing function or underlying KDF. The contest is partitioned into two classes: the cut-throat Pro class, and the more casual Street class.

About the Team

Founded in 2010, Team Hashcat is a static, hand-selected fraternity of professional password crackers who have proven themselves worthy of representing the Hashcat name. Organized and led by Hashcat founder *atom* and managed by team member *dropdead*, Team Hashcat has taken First Place in fifteen password cracking competitions over the past thirteen years, including ten First Place CMIYC victories. Team Hashcat represents the best of the best the password cracking community has to offer.

The following team members actively participated in this year's CMIYC competition:

atom	baybedoll	blandyuk	Chick3nman	dropdead
epixoip	EvilMog *	kontrast23	kryczek	m3g9tr0n
N GHT5	philsmd	rurapenthe	The_Mechanic	T0XIC
TychoTithonus	unix-ninja	Xanadrel	xmiserj	_NSAKEY

* *EvilMog experienced a medical emergency that limited his participation.*

Organization & Planning

The 14th CMIYC competition was held during DEF CON 31 in Las Vegas, NV from 11 August 2023 11:00 PDT until 13 August 2023 11:00 PDT. Team Hashcat naturally competed in the Pro class for this competition. As Team Hashcat is heavily geographically dispersed - with some team members located on site in Las Vegas and others located around the globe - we rely upon Discord for both real-time and asynchronous communications, Google Sheets for formal planning, and our home-grown collaborative cracking platform *List Condense* (LC), maintained by our very own Xanadrel, for operations.

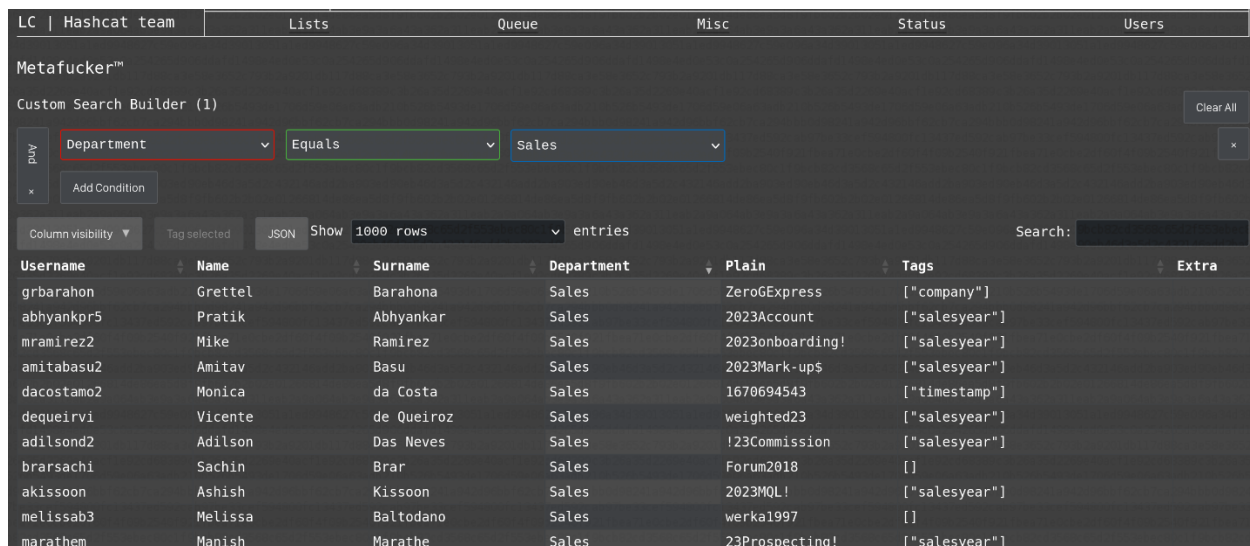
Software Stack

Team Hashcat utilized the following software during the competition:

Name	Version	Link
List Condense (LC)	0.1.0a	Internal collaboration platform, not for distribution
hashcat-lc	6.2.6+LC	Internal fork of Hashcat that integrates with the LC API
hashcat-utils	1.9	https://github.com/hashcat/hashcat-utils
John the Ripper	Bleeding Jumbo	https://github.com/openwall/john
PCFG Cracker	4.5	https://github.com/lakiw/pcfg_cracker
princeprocessor	0.22	https://github.com/hashcat/princeprocessor
maskprocessor	0.73	https://github.com/hashcat/maskprocessor
PACK2	0.1.0	https://github.com/hops/pack2
r1ng	1.74	https://github.com/Cynosureprime/r1ng

When the test hashes were released, we made several improvements to our extensive automation. Xanadrel added additional automation to handle all communications with KoreLogic, including decrypting and parsing received emails, as well as adding a new Discord bot to monitor uploads and submissions.

After the contest had started, Xanadrel added a new section to LC lovingly dubbed *Metafucker* that displays the user metadata in a queryable table format alongside their corresponding cracked plaintexts for easier analysis. Kudos to Xanadrel for taking time away from the fun of cracking for the benefit of the team.



The screenshot shows the 'Metafucker' interface with a search filter set to 'Department = Sales'. The table below displays the resulting user metadata and cracked plaintexts.

Username	Name	Surname	Department	Plain	Tags	Extra
grbarahon	Grettel	Barahona	Sales	ZeroGExpress	["company"]	
abhyankpr5	Pratik	Abhyankar	Sales	2023Account	["salesyear"]	
mr Ramirez2	Mike	Ramirez	Sales	2023onboarding!	["salesyear"]	
amitbasu2	Amitav	Basu	Sales	2023Mark-up\$	["salesyear"]	
dacostamo2	Monica	da Costa	Sales	1670694543	["timestamp"]	
dequeirvi	Vicente	de Queiroz	Sales	weighted23	["salesyear"]	
adilsond2	Adilson	Das Neves	Sales	!23Commission	["salesyear"]	
brarsachi	Sachin	Brar	Sales	Forum2018	[]	
akisssoon	Ashish	Kissoon	Sales	2023MQL!	["salesyear"]	
melissab3	Melissa	Baltodano	Sales	werka1997	[]	
marathem	Manish	Marathe	Sales	23Prospecting!	["salesyear"]	

Photo enlarged to show texture.

Hardware Stack

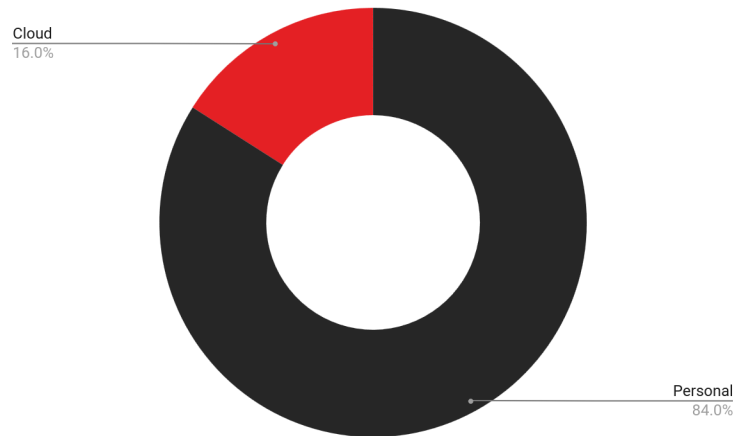
Most members of Team Hashcat utilize personal hardware that is readily accessible to them. Most commonly, these are personal desktop computers with mid- to high-end CPUs and consumer-grade graphics processors. Occasionally, cloud assets and special-purpose hardware such as FPGAs are deployed for specific challenges. Team Hashcat utilized a total of 47 FPGA boards and 78 GPUs for this competition:

Count	Model
47	ZTEX 1.15y FPGA boards (4 x Xilinx Spartan-6 LX150 FPGAs per board)
19	NVIDIA GTX 1080 Ti
18	NVIDIA RTX 4090
10	NVIDIA GTX 1080
8	NVIDIA Tesla T4
5	NVIDIA GTX 2080 Ti
3	NVIDIA GTX 1070
3	NVIDIA GTX 980 Ti
2	NVIDIA Titan Xp
2	AMD Radeon RX 6900 XT
2	NVIDIA RTX 3090
2	NVIDIA RTX 3080 Ti
2	NVIDIA RTX 3060 Ti
1	AMD Radeon RX 7900 XTX
1	NVIDIA RTX 3080

While this may seem like a lot of hardware, this essentially translates to a 3.9:1 GPU-to-member ratio (a 39% reduction vs. CMIYC 2021), a 2.35:1 FPGA-to-member ratio, and a 1.6:1 GPU-to-FPGA ratio.

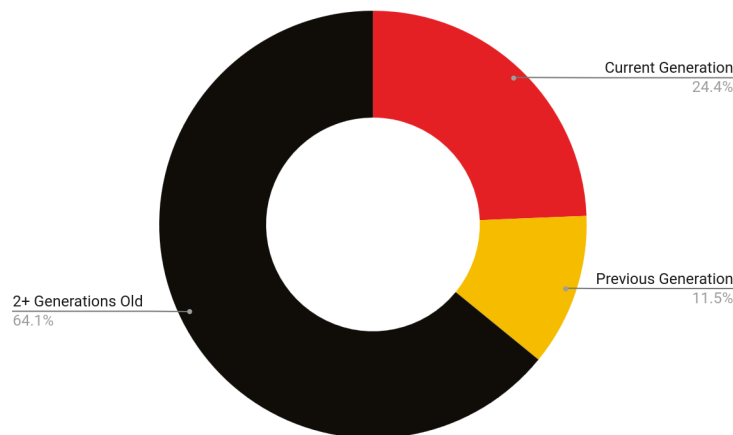
Nearly 85% of the hardware utilized by Team Hashcat during the competition was personally owned. All cloud assets, combined with all ZTEX FPGA boards, were solely dedicated to cracking bcrypt hashes (because *fuck* bcrypt), while the remaining graphics processors were utilized to crack all other hash algorithms. The total cloud expenditure for the contest was \$815.00 USD, or an average of \$40.75 per team member.

We recognize the use of cloud resources is potentially contentious, and that there may be some disagreement as to whether the use of such during the contest is particularly sporting. This has been a source of some internal debate as well, where some team members feel we should make the most efficient use of the resources we already have, and others feel cloud resources are more economical and more environmentally friendly, further adding that they are spending less on cloud resources than they would on a new GPU. While we have yet to reach a consensus, one undeniable fact is that the cloud is open and available to all, and affordable computing power is easily accessible and well within reach of even the *entry-level* adversaries that we all ultimately seek to simulate.



Distribution of cloud assets vs. personally-owned compute devices.

Less than 25% of the graphics processors utilized by Team Hashcat were current generation models. In fact, more than 64% of the graphics processors were two or more generations old, while the three-generation old GTX 1080 Ti was the most represented GPU amongst team members. This serves to demonstrate that the latest and great hardware is not a prerequisite for success, nor is old hardware a barrier to competitiveness.



Distribution of graphics processors by age.

Even with the hardware dedicated to bcrypt, it was challenging keeping our hardware anywhere near 100% utilized, as the particular nature of this competition leads to most time being spent on pattern analysis and attack preparation. A mere 30% or less of our non-dedicated hardware was in use at any given time, with our most successful non-ZTEX team members using only two or three GPUs apiece.

Competition Narrative

At the start of the contest, each team member independently downloaded and decrypted the GPG-encrypted hash file provided by KoreLogic. Upon opening the decrypted file, we realized “fuck, it’s YAML, just like the test hashes.” The file appeared to emulate an authentication database with users from multiple companies and locations, perhaps as some sort of merger or acquisition scenario. This was a cool twist, but it also presented new challenges versus the usual “go crack this shit” format.

A typical account had the following metadata:

- Username
- Given Name
- Surname
- Created Date (weird format and timezone discrepancies, which turned out to be a hint)
- City
- Phone
- Company
- Department

We raced to convert the data structure into a more usable format, and within the first five minutes we had isolated the following hash types and imported them into LC:

- Raw MD5 (mode 0)
- Raw SHA-1 (mode 100)
- Raw SHA-256 (mode 1400)
- LDAP SSHA-1 (mode 111)
- LDAP SSHA-512 (mode 1711)
- md5crypt (mode 500)
- sha1crypt (mode 15100)
- sha256crypt (mode 7400)
- bcrypt (mode 3200)

With bcrypt, of course, being the most pestiferous – and the most valuable.

We knew that the metadata would prove to be a key component of the contest, so we attempted to enrich the metadata as much as we could with what we had learned from the test hashes. One way in which we did this was to map each user to their corresponding language and charset based on their location and phone number. The phone number turned out to be inconclusive, so we used ChatGPT 3.5 to give us an estimate of the primary language for each user. This would be used later to feed Google Translate for translation into different charsets.

KoreLogic always has to throw a curveball our way, and it wasn’t long before we identified issues with how Hashcat handles md5crypt and SSHA-512. Many of the md5crypt hashes shared the same salt values, which is not something Hashcat supported, but of course we were able to quickly resolve this issue. Similarly, our parser for SSHA-512 expects the tag to be uppercase, while the hashes in the database had lowercase tags. A simple substitution was enough to work around this issue, although we found that we also had to substitute the lowercase tags back before submitting our cracks to KoreLogic.

At this point in the contest, the name of the game was to search for patterns in the fast hashes that could then be correlated back to identify slower hashes that could be cracked with that knowledge. The attacks run against Raw MD5 and Raw SHA-1 were, in most cases, not feasible to also run against the much slower and much more costly sha256crypt and bcrypt hashes without additional optimization. The work loop was essentially as follows:

- Blitz shit at the fast hashes, to see what sticks
- Identify plaintexts that look interesting
- Attempt to identify correlations or patterns
- Find groupings based on any known metadata
- Devise and tune a targeted attack against slow hashes
- Test it
- GOTO 10

One of the first patterns we identified was **epoch timestamps**; the all-numeric plaintexts were easily identifiable in juxtaposition with the other, more traditional plaintexts. For the faster hashes, it was trivial to brute force `16??d??d??d??d??d??d`. For the slower hashes, however, this was not feasible. Here we identified that most of the account creation dates in the metadata were provided in CST, but some were provided in CDT. It did not take long to determine that the ones with CDT timestamps could be converted to epoch time and correlated on a user-by-user basis to crack.

Example: Thu Sep 1 21:33:54 CDT 2022 -> 1662086034

Another pattern we identified early on was **foreign languages**. Not only bare dictionary words in languages like Polish, Dutch, and Russian, but we also rapidly identified that some of the plaintexts were kanji, which then led to the realization that those kanji are the actual names of some of the users. We also stumbled upon the Hindi river names midway through the contest by running a list of Hindi Wikipedia article titles. The keyspaces were mostly too large to hit on the slow and salted hash types, which was a common theme throughout this contest, but correlating the hashes by inferred language helped reduce the search space.

Example: Imai -> 今井

We also identified more metadata-related hits in the distinct pattern of "**CompanyName Suffix**", such as "QuantumLeap Games". To fully exploit this pattern, we devised a list of common company suffixes with the help of Reddit and other sources and ran simple combinator attacks (-a 1) against the fast hashes. After enumerating as many of the company names as possible, an association attack (-a 9) was used against the bcrypt hashes, cracking all of the hashes that followed this pattern in less than 30 seconds on a single GPU.

Examples: AeroDyn Aerospace, DataLeapDynamics, CyberGuard Security, TerraGlideFoods

We further identified additional metadata-related hits for users in the **Sales** departments. All of these plaintexts were corporate-style passwords that followed very predictable patterns and a limited vocabulary.

Examples: !23Executive, Quota23@, 2023Profit\$, Account2023!

We also hit upon a peculiar pattern that initially threw us for a loop. We want to precede this by reiterating that the sources for these competitions are *supposed to be* completely artificial, and that's usually what gives these competitions a less-than-genuine vibe. In the real world, we leverage the human element of the

password creation process by exploiting the universal patterns we are all prone to following. But in CMIYC, we have Hank. And the game, at its very essence, is to reverse-engineer Hank - what sources did Hank use, what patterns did Hank devise, etc. One of the drawbacks of the “throw everything at the wall” approach is that it makes it difficult to trace back the source of a crack when needed (who cracked it, what attack did they run, etc.) So color us fucking surprised when we found a large number of plaintexts prefixed with “#3&4%#!”, which appeared to be some sort of static salt or pepper, and we traced the source back to the `hashmobs-all-found` and `hashes.org` wordlists, and then to the Turk-Internet leak specifically. Holy shit, there’s **real breach data** in here! This wasn’t too terribly difficult to exhaust once we found some of the sources - but we couldn’t exactly run an entire 16 GiB wordlist through the salted hashes. A bit of analysis concluded that users with these real-world passwords were all from the Telecom departments, which enabled us to reduce the search space.

Examples: #3&4%#!alpo123789, #3&4%#!etcpasswd, #3&4%#!morpheus

One last pattern before we dive into the two patterns we spent the most time and effort on: the **math and physics equations and chemical formulas**. This was a theme that we got a few hits on, but when we tried to exploit this theme, we largely came up empty-handed. We believe this was primarily due to the use of unicode **subscript & superscript** characters. When attempting to create wordlists for this, all of the sources we found used images or HTML entities.

Examples: $A = \pi r^2$, $I = \sum mr^2$, $PV = nRT$, $C_{12}H_{22}O_{11}$, C_6H_5OH , CH_2O

Let’s talk **GHosting**. Most of the users from this company were different from the other users in the database as they didn’t have names or phone numbers, so we felt like we didn’t have much metadata to go off of. The pattern eluded us for quite a while and left us banging our heads against the wall, until we had that epiphany moment and realized the usernames for the users in the IT department were a sort of code. For example, take the username “pk4923”: the ‘p’ stands for ‘production environment’, the ‘k’ stands for Kubernetes, and ‘4923’ is the system ID. We had already identified and began exploiting this pattern when the last hint was released, but that did provide confirmation that we were taking the right approach. Unfortunately, it turned out finding what all those letters in the usernames represented was fairly difficult. Initially, we constructed a table attack to expand each of the characters in the usernames. However, we ended up writing a special tool called *ghost-cracker* to exploit what we knew and only target the slow hashes with the actual combinations of possible words and separators. While we were able to crack a fair number of these passwords, a decent amount remained uncracked.

Examples: PRODKube8784, dev|Oracle_1777, UAT%2dweb%5f2375,

Finally, the largest and most prevalent pattern – the one we identified early on in the competition, but didn’t fully master until the final day – the two, three, and four-word **passphrases**. First, it was overwhelmingly obvious that the phrases were not random – they were from *something*. So we initially tried the three books and three movies listed on the DEF CON “about the theme” page, and that yielded quite a few cracks, but we still felt there was more. We began to expand our search for more potential sources, which led to us creating a special tool called *epub2phrases* to generate two, three, and four-word phrases from EPUB archives using a sliding window. For movie scripts that are not distributed in EPUB format, we found a tool to convert PDF files to EPUBs. With this tool we were able to rapidly build a very large collection of phrases from dozens of potential sources. We had also identified that single quotes, not just punctuation characters, were used to split the source words leading to words like “m” instead of “I’m”, “ve” instead of “you’ve”, etc., and our tool

incorporated this as well. Over time, we realized that only certain numbers and special characters were appended to the phrases, which enabled us to reduce our search space. Unfortunately, we still felt like we hadn't found "the" source after generating millions of potential phrases, so we decided to turn to AI candidate generation. Using lakiw's Probabilistic Context Free Grammar (PCFG) cracker, we were able to train a model on all of our cracked phrases to help us generate new phrases without having to download and scrape more EPUBs. This approach worked surprisingly well, and we were able to crack dozens of bcrypt hashes with PCFG. For the majority of the contest we only had two to four team members working on these phrases, but once it became clear that the distribution of our cracks was skewing ~70% towards phrases, it was all hands on deck for the final sprint.

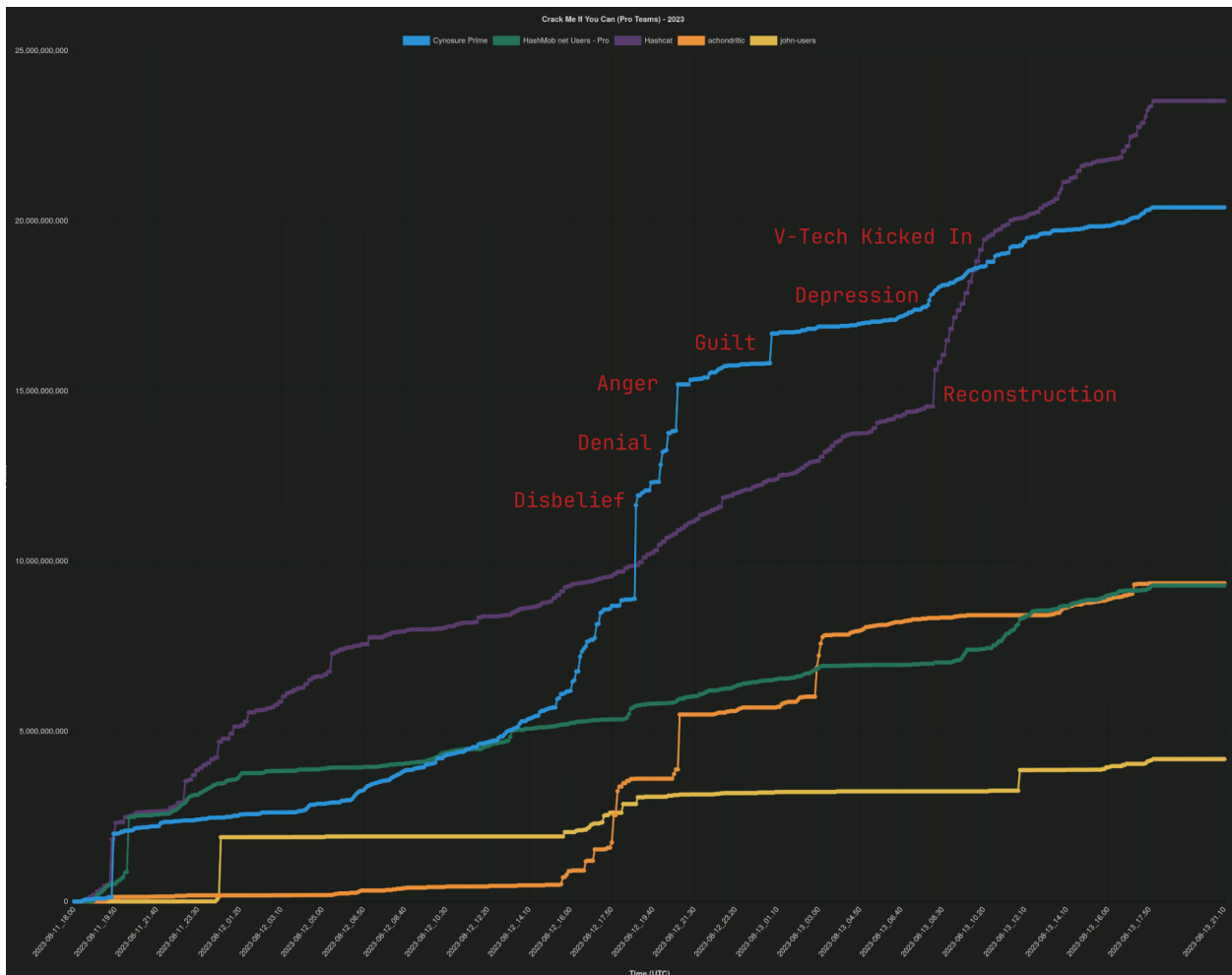
Examples: a little misunderstanding1, about my boobs1, all AIs want9%, They hold images1

Street Hashes

Similar to previous years, we began cracking the Street class hashes at some point on the second day of the contest in order to identify overlapping plaintexts and patterns. However, while there was some overlap with the phrases, we identified very little overlap in other areas. This had the benefit of confirming some of the patterns, but overall didn't help us directly.



In Conclusion



The seven stages of grief, CMIYC edition.

This year's contest attempted to emulate a real world scenario of a service provider managing authentication for multiple entities (Okta-esque), or a merger, or an acquisition, or whatever, complete with real world breach data. The delivery format was unique, the setting was engaging and quite challenging, and overall it was barely manageable, and at times on the verge of being ridiculously hard. Despite the usual frustration during specific times of the contest (especially when CsP managed to take and hold the lead for a third of the contest) the team pulled together and fought even harder to find and exploit more patterns.

The biggest takeaway for us was that we need to increase the focus on the tooling to handle these complex association attacks and allow for better analysis, which was probably the core idea of what KoreLogic wanted us to go for.



Overall – great job, KoreLogic! This was probably the smoothest and most well-balanced competition so far.

We would also like to thank all of the Pro teams – Cynosure Prime, achondritic, HashMob, and john-users – for competing against us again this year, and for giving us a great fight to the end! We greatly look forward to our next match-up.

