HashMob

Passwords are like a box of Chocolates, You never know what you're going to get.

Forrest Gump?

CrackMeIfYouCan 2022
Write-up

HashMob.net
`vavaldi@hashmob.net`

Chapter 1

Preface

Allow me to preface this write-up by beginning to thank the wonderful employees of KoreLogic for organizing the contest and helping out with the PasswordVillage at DEFCON 30, as well as the amazing members of HashMob.net who contributed to the performance of the teams during the contest itself. HashMob participated with two different teams in this contest. One team for the Pro division, and one team for the Street division. Although a few of the Pro team had participated in CMIYC 2021 and CTC 2022 before, a portion of them had not played in any contest prior to CrackmeIfYouCan, and the majority of the Street team had never played together at all. With this in mind both teams performed admirably during the contest.

The Pro team earned a #3 spot on the leaderboard, closely behind CynoSurePrime and Hashcat. The Street team landed on the #7 spot; gaining many experiences along the way. We look forward to participating in more contests in the future, hope to see you all there!

Pro Team

- Vavaldi
- penguinkeeper
- Shooter3k
- _cin
- cyclone
- gatete
- RealEnder
- WHYPHY
- Cochino
- w00dsman
- Flagg
- NocFlame
- clem9669
- SoSander

Street Team

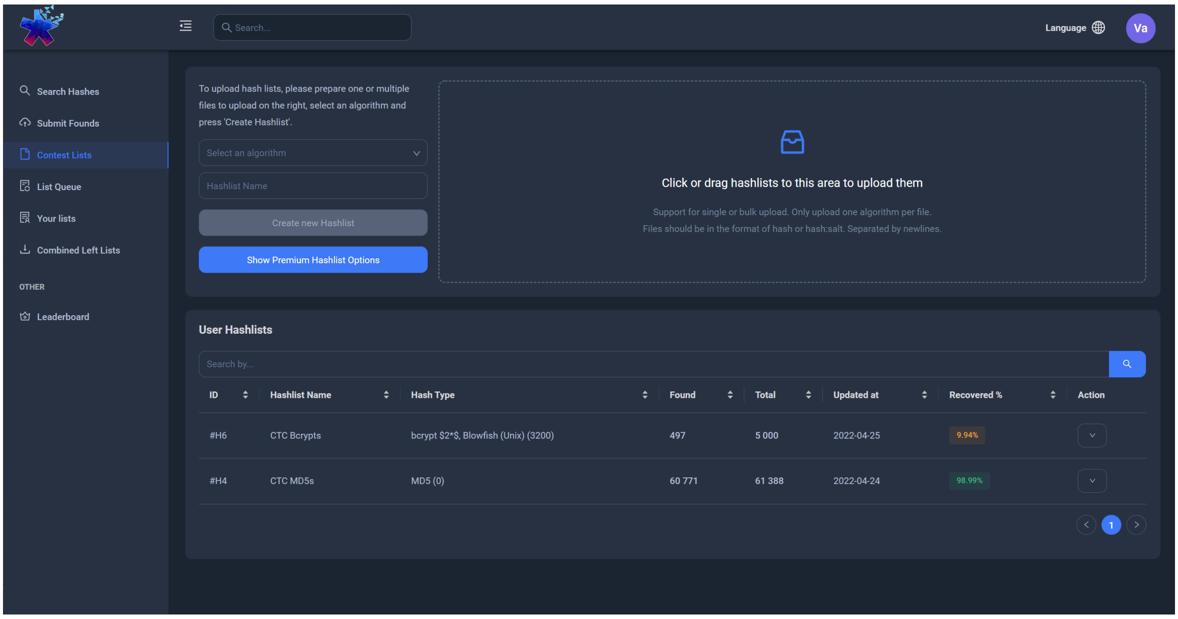
- mostwanted002
- cake
- DAK
- zorbanoodle2
- 0x4C00
- sensei

1.1 About HashMob

HashMob is a large, mostly discord based, community which focuses on Cryptography and Hash password recovery. Users have picked password recovery up as a hobby over the years due to their interest in security or because of their jobs, and spend a lot of time working with cracking hashes and research on the passwords of users. It was founded in 2021 almost half a year after Hashes.org closed its doors in January of 2021. Since then it has recovered over 442 648 954 passwords, discovered more than 101 469 287 new plaintexts, and amassed a following of more than 1,000 members.

1.2 Contest Environments

Each team was given access to a copy of the HashMob.net web-application with a custom back-end script that would use the available API of the contest to automatically submit new found solutions regularly. These environments were restricted in access so only authenticated team members from each respective team could participate. Registration was protected with a secret key which was made aware to each team via their discord channels. i.e. the teams were unable to see each others' hashlists and founds and their applications did not interfere with each other.



To upload hash lists, please prepare one or multiple files to upload on the right, select an algorithm and press 'Create Hashlist'.

Click or drag hashlists to this area to upload them

Support for single or bulk upload. Only upload one algorithm per file.
Files should be in the format of hash or hash:salt. Separated by newlines.

User Hashlists

ID	Hashlist Name	Hash Type	Found	Total	Updated at	Recovered %	Action
#H6	CTC Bcrypts	bcrypt $2*$, Blowfish (Unix) (3200)	497	5 000	2022-04-25	9.94%	
#H4	CTC MD5s	MD5 (0)	60 771	61 388	2022-04-24	98.99%	

Chapter 2

Pro Team write-up

2.1 The Preparation

In preparation of the contest the members of the Pro team went through the learning points from CTC 2022 to see what actions we could do to best prepare for the contest. Some of the scripts were updated to reflect the new environment to work in - which was a big upgrade compared to the CMIYC 2021 environment.

2.2 Software Used

Listed below are some of the software used by the Pro team, although a majority of them are public / open source tools, some are closed source, modified open source, or specifically developed for the contest. The Hashtopolis instance was modified to automatically submit any founds to the HashMob instance based on the hashtype of the hashlist it was discovered in. We modified the SendProgress API to perform this in a rather 'hacky' fashion and for future instances we would like to improve this (more on that later).

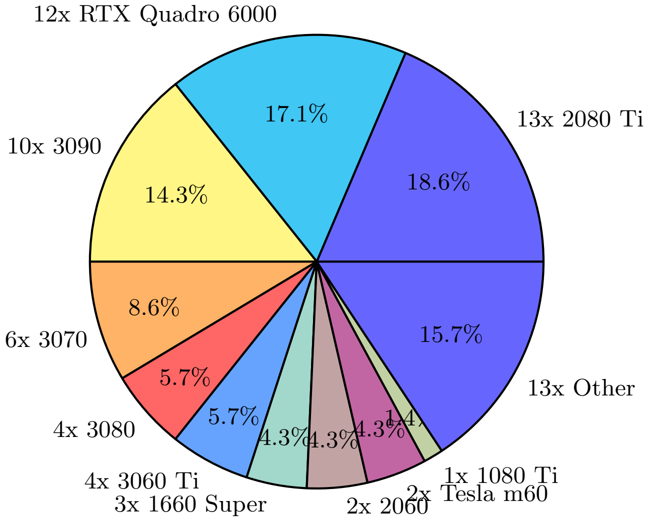
Overview of Used Software			
Name	**Open Source**	**Public**	**Purpose**
Hashcat*	yes	yes	Password recovery
MDXfind	no	yes	Password recovery
HashMob Search*	no	yes	Hash Lookup
Hashtopolis	yes**	yes	Collaboration
HashMob Mirror	no	no	Collaboration
Gramify*	yes	yes	Analytics
PACK	yes	yes	Analytics
PACK2	yes	yes	Analytics
rulesfinder	yes	yes	Analytics
Plain-Finder*	no	yes	Analytics
RuleProcessorY*	yes	yes	PW generation
pcfg_cracker	yes	yes	PW generation
PRINCE	yes	yes	PW generation
sync.py	no	no	Auto submission

* These tools can (also) be found on HashMob.net or their discord.

** Source code was modified and tweaked to suit our needs.

2.3 The Hardware

The Pro team has a large show up with a variation of hardware. Some participating with some impressive hardware like the UHD Graphics 620, or the more mediocre 12X RTX Quado 6000s that were rented for the duration of the contest. A grand total of 70 Graphics cards were used. More graphics cards were rented for the duration of the contest compared to last time.



2.4 Ready? Set...

Go! The first minutes of the contest were spent investigating the cards we were dealt. Each list was stored within a special container which was protected, encrypted or otherwise obscured from view. The easiest ones came first and the first minutes was spent breaking open the two ZIP files as the zip2john scripts made it easy to extract their hashes.

2.5 The Infamous Lists & Results

The lists presented to us for the 2022 CrackMeIfYouCan contest were made up of various algorithms. We'll briefly go over each hashlist one at a time, discussing how we approached it, what we found and the final conclusion. If you are interested in taking on the 2022 list yourself, I recommend stopping after this section. The lists we were offered are listed below.

HashMob finished in third place with a total of 1 569 730 points. Closely following CynoSure Prime with 1 580 986 points and trailed by john-users with 1 239 917 points.

Hashlist	**Plains Found**	**Point Value**	**Total Hashes**
yescrypt	4	100 000	4
sha384	5 940	46	6 023
sha512	5 270	43	5 382
mysqlna	4 602	17	5 043
sha224	4 003	14	9999
sha256	10 231	13	10 231
mssql05	9 973	9	10 000
vBulletin	7 637	6	7 805
nsldaps	22 006	5	10 000
sha1	17 424	5	17 444
half-md5	18 961	3	20 600
md5	12 989	1	13 323

2.6 The Containers

Over the course of the contest, 5 tar.gz files were released containing different files. Most contained one or multiple archives and one contained a key file to unlock a previously locked list. An overview of all containers, their type / encryption and the unlocking password is shown below.

Type	**Name**	**Password**
7z	wopr.7z	joshua
web url	web.conf.tgz	*N/A*
ZIP-BIG	riddle_wrapped_up_in_an.zip	Enigma22!
PDF	ManMadeSelf.pdf	2022Defcon
GPG	Authori...spiracy.hashes.gpg	LasV3gas
LoopAES	LoopAESLoopAESLoopAES	PasswordPasswordPassword
KeePass	DEFCON.kdbx	Summer22
LibreOffice	20DollarDumps.odt	homecoming2022
KeePass-Key	DEFCON-with-key.kdbx	*opened with .key file
Gocryptfs	gocryptfs	Gadget
zip-small	halfmd5.zip	password
rar	new_and_unbroken	password

2.6.1 The Contest

Our general approach to the lists consisted of identifying the container, extracting / cracking the hash. Opening the container, identifying the hashing algorithm, cracking hashes and finally identifying what the potential source might be on which the list is based. In some cases this was already partially possible by the name of the file such as the "Riddle wrapped up in an" file. The most jarring of the contest was not being able to recover specific plains for the archives. LasV3gas's GPG file password went unguessed for a significant amount of time despite a significant amount of effort and being unable to progress in any list because you're unable to hit a single (relatively simple) password was fairly disheartening to the team members. On the other side we had files like "fooo" and "LoopAESLoopAESLoopAES" where the archive type was completely unknown and we were uncertain where to start exactly.

Fooo turning out to be bogus random bytes did not help with this. LoopAESLoopAESLoopAES threw us for a loop thinking it might be a triple-AES encryption matching the CBC encryption cipher we attempted to decrypt it with a multitude of keys looking for invalid PKCS #7 to verify if decrypting it twice returned a valid result and finally printing out readable bytes. In the end we ended up cracking this archive open by writing the password by hand (of all things) and ended up opening it using the aespipe tool.

The PDF ManMadeSelf contained both a user and master password, when using hashcat it only discovered a single password with the user password being seemingly random data. This is presumably caused by hashcat matching on the master password and deriving the user-password from it. The solution to this was to use –keep-guessing for a short period to collect multiple "valid" passwords and looking for user-passwords that were in the printable character range and plausible. This approach left only one solution.

CMIYC 2022's 5th release from the Pro hashes contained a file called Pennysuncle.tgz. The content of this folder contains three files; gocryptfs.conf, gocryptfs.diriv and a long random looking string as filename. Looking inside the conf file we identified another reference to this being a gocryptfs file. Next we followed the quick start guide from the gocryptfs site to get us up and running. The instructions to create and mount a gocryptfs file system are easy and straight forward:

```
mkdir cipher_folder
mkdir plain_folder
gocryptfs -init cipher_folder
[...]
gocryptfs cipher_folder plain_folder
```

This successfully completes the requirements to create and mount an encrypted folder. This can be verified by creating a test file in the plain folder and seeing a subsequent encrypted version of it being created in the cipher folder; this file has a random string as filename. Based on this we can deduce that the file with a random name is the file that should be our target of decryption.

With a working setup we can now attempt to decrypt the container. After unmounting our test environment with the `fusermount -u plain_folder` command we investigated the gocryptfs help / manual where we found it allowed for the passing of a `-passwd` argument. Adding our password as argument allowed us to build a quick script to enumerate possible passwords (Additionally it also accepts piped strings when given a folder to mount). When this was done, we took a closer look at the filename to try and find suggestions for possible passwords. The first search result turned out to be a red-herring; however, it did not take long for us to find another source which contained the correct password for the container.

Chapter 3

Closing Notes

This concludes the write-up of the Pro of HashMob.net. Both Street and Pro team have had many learning opportunities and were shown room for improvement in different forms. We again wish to thank the contributions of all members of both teams, and the contest staff. This write-up was written not to only talk about how we achieved what we did, but also how we could have improved ourselves even more. We discussed the things we struggled with and hope you learn both from our successes and mistakes in your future endeavors. Since this contest was primarily focused on containers, and the hashes themselves were largely simple / easy we did not dive into them as much as we have in the previous write-up. Finally I'd like to invite everyone to check out the https://hashmob.net/ website and community, and join the discord community (linked on the website). It's an open community where you can actively research passwords, attacks and learn more about the general field of cryptography. Our community contains members of various backgrounds with a wide variety of skill sets and most relevant questions can be answered expertly.

Unfiltered Opinions

Below are some unfiltered opinions of the different members of the teams regarding the contest. Due to the nature of the contest a few users had criticisms regarding the contest. This small section at the end provides some room for them to 'vent' or provide feedback.

penguinkeeper: Overall, it was a decent competition. The archive hashes had plains that were reasonably reachable although advancing in skill for hash cracking is very much about increasing the chance of a crack, it's never guaranteed. This means that if the first place team (Team Hashcat) just by pure luck didn't find the pass for LoopAES, they would have become third and it would have been impossible to make up for that score loss through the other lists as they mostly all had very easy plains that were < 95% cracked in minutes. This is a mechanic I personally disagree with and find tiresome when almost the whole competition was waiting for plains to validate with the slow archive hashes and not finding and exploiting plaintext patterns as is tradition for hash cracking. This, along with the then-unknown uselessness of the `fooo` file made it particularly afflictive. Would do again, 6/10 IGN

SoSander: Nice competition

Shooter3k: Overall, I thought the competition was really well thought out and great. I really do enjoy and love this competition. As advertised, I thought the competition was going to be ”The premier annual password cracking contest sponsored by KoreLogic”. However, I felt misled and felt KoreLogic’s thoughtful challenges were not the type of password cracking challenges I was hoping for. Instead, I felt they had more to do with pentesting and far less to do with actual password cracking. Overall, I felt the competition was 95% pentesting and 5% password cracking. I really love this contest and I’m hoping to see either the competition description/name changed or more focus on password cracking in the future.

w00dsman: Was a great effort. Looking forward to next year

Vavaldi: This year’s rendition of the contest suffered from some issues that real life scenarios suffer from as well. By having very limited hashes holding back a lot of hashes we were stuck waiting on a single hash to be cracked before we could unlock the others. The result being us running attacks and getting no positive or negative feedback on what we were doing. We can be stuck running the same attack for 2-3 hours before trying a completely different angle because of slow hashes like gpg could mean we just haven’t hit the right password yet. The result being a lot of bored members trying to crack or work on hashes with 0 cracks in multiple hours. Realistic? Check. Ideal format for a contest? Not as much. Having been able to open all containers I can say that the contest was enjoyable but can guarantee that if we failed to open one container we would not have had the same outlook due to losing automatically because we could not crack 1 hash and having no way to compensate the 80k+ lost points.