ASSIGNMENT 1: VERACRYPT
Digital Forensics

Elisa Pioldi
ID 12305812

October 25, 2023

1 Factual part

1.1 Task

The goal of this task was to recover the password of a **Veracrypt container** through a **bruteforce** attack. The container encryption was performed with the cryptographic algorithm *AES* with *RIPEMD-160* as the hash function.

This report includes the analysis of multiple containers, to provide specific benchmarks.

1.2 Containers analyzed

I downloaded from the website `is.gd/mkveracrypt` the following containers, with input `12305812`:

- Lower case - 4 digits
- Alpha - 4 digits
- Alphanumeric - 4 digits
- Special - 4 digits
- Numbers - 4 digits
- Numbers - 5 digits
- Numbers - 8 digits

The reason for this choice is explained in the section 2.1.1.

1.3 Containers hashes

To verify the containers' integrity I computed just after the download of the SHA-1 checksum [1] for each of them (see Table 1).

Character set	Length	SHA-1 checksum
Lower case	4	`8da0a7118fc2fedadb5714728435c4a1e03ec78c`
Special	4	`cfcd5ee35c6729e88e0ff1b12c0ebdf80af6aa12`
Alpha	4	`338d19ea6ab41ca770a92c0fd4d876cff746c3f3`
Alphanumeric	4	`1fd07b016f56e9e4e3a3b8c9efa6d37684955b7b`
Numbers	4	`265fcd87b60091d957be21cd6e39355c3918a191`
Numbers	5	`5f670ad147cc2513dadf5e400ced2a4e7db9777d`
Numbers	8	`21cdedf401ff1027885b6ee42c8b506bbc54e76e`

Table 1: SHA-1 checksum for each container.

1.4 Software and hardware specifications

For this analysis, I utilized the following software:

- **Veracrypt** [2] – version 1.24 (update 7); to perform container mounting and access.
- **Hashcat** [3] – version 6.2.6; the world's fastest password cracker, used to find containers' passwords through brute force. Hashcat provides a specific mode to attack the container's encryption [4]; moreover, I exploited its GPU optimization on the graphic card GTX 1060 (6 GB of VRAM), choosing a moderate workload profile for its execution.

1.5 Password cracking

After the preliminary analysis, I ran Hashcat on every downloaded container to collect statistics and passwords.

You can see some of the statistics (such as execution time and number of hashes/second) concerning simple character sets in Table 2. For better understanding of time behavior, see Section 2.3.1.

For every execution, the hashrate (number of hashes per second) was around 400 H/S average (for hardware benchmarks, see Section 1.4). I couldn't crack the container with the numeric password of length 8, since it would have required around 3 days, according to Hashcat's estimation.

After collecting all the passwords I could mount every Veracrypt container to check the content inside. All the recovered passwords are shown in Table 3.

Character set	Length	Time	Hashrate
Numbers	4	26 secs	397 H/s
Numbers	5	1 min, 9 secs	363 H/s
Lower case	4	12 mins, 37 secs	411 H/s

Table 2: Statistics of some cracked containers.

Character set	Length	Password
Lower case	4	`yxgy`
Special	4	`=!(;`
Alpha	4	`kDfl`
Alphanumeric	4	`EvXY`
Numbers	4	`2709`
Numbers	5	`83450`

Table 3: Recovered passwords for each container.

1.6 Container contents

Every container contains four different files: three pictures (Figure 2, 3a and 3b) and one text file.

Imagination
2011 © WALLPAPER CRÉÉ PAR REVERS - SPONGE BOB © TOUS DROITS RESERVÉS

Figure 2: First picture stored in the container (awesome.jpg).

(a) trippin.jpg (b) wasted.jpg

Figure 3: Last two pictures stored in the container.

Each text file, named `secret.txt`, contains a **42-character long string** (see Table 4), slightly different from container to container.

Character set	Length	Secret
Lower case	4	`fede1cea08a6465514160a54076874f2c967ffc53b`
Special	4	`fede1cea08a6465514160d54076874f2c967ffc53b`
Alpha	4	`fede1cea08a6465514160e54076874f2c967ffc53b`
Alphanumeric	4	`fede1cea08a6465514160f54076874f2c967ffc53b`
Numbers	4	`fede1cea08a6465514160c54076874f2c967ffc53b`
Numbers	5	`fede1cea08a6465514161c54076874f2c967ffc53b`

Table 4: Secrets inside each container.

2 Expert testimony

2.1 Background

2.1.1 Containers choice

Since cracking passwords is computationally demanding, I had to choose by selecting which containers to analyze. I wanted to have a significant pool in matters of time benchmarks, passwords and revealed contents.

Since numeric passwords are easy to crack, I looked for 3 different containers with passwords of 4, 5 and 8 digits, to have an overview of benchmarks when changing the password length. I already knew evaluating my hardware benchmark would have been infeasible, but I wanted to see how much computational time Hashcat would have estimated. Moreover, since the number of lower case and upper case letters are the same, I chose to analyze only one container with such passwords.

All of the reasons above led me to choose the ones specified in Section 1.2.

2.1.2 Hashrate

I chose to use the Hashcat workload profile 3 (out of 4). Still, nonetheless, the hashrate was very low (only hundreds of hashes per second) compared to the most classical hashing algorithms (in order of thousands of MH/s) [5]. I checked various benchmarks and all the Veracrypt ones showed this behaviour.

This is evidence of how well-resistant Veracrypt is against brute-force attacks.

2.2 Secrets analysis

The *secrets* reported in Table 4 seem to present some specific pattern.

The differences in the secrets only concern two characters in the middle of the string: we have the same 20 characters, 2 specific digits and the same other 20 characters. Since SHA-1 produces a message digest of 40 characters [1], I supposed the first and the last characters in the secret are a sort of checksum of the input ID and something else. Concerning the central two digits, I noticed a recurring pattern: the first digits indicate the length of the password (0 for a 4 digits long password, 1 for a 5 digits long password and so on) and the last digits refer to the character set (a for lower case, c for numbers and so on).

2.3 Strength analysis

At this point, we have all the data to perform a password strength analysis. In particular, I elaborated two formulas to find two specific quantities, always concerning a **brute-force attack**: the maximum amount of time needed to crack a password and the minimum password length to resist for a given period.

All the following computations are done considering a hashrate of 400 H/s for the previous tests with Hashcat.

2.3.1 Maximum cracking time

First, let's find a way to compute how much time is needed to brute-force any container.

Given S the character set (for instance, $|S| = 26$ if we choose the set of lowercase letters), l the password length and hps the number of hashes per second, time (maximum number of seconds, i.e. upper bound) is computed as follows:

$$time = \frac{|S|^l}{hps}$$

Through this formula, we can compute for instance the cracking time for a 4, 5 and 6-digit password (I'll consider these amounts of digits to show the exponential growth in comparison to the 4-digit passwords used previously for tests). Results are displayed in Table 5.

As we could expect, time grows exponentially with the character set's cardinality; moreover, note that this is just an upper bound: this explains why in the Hashcat tests I managed to recover passwords in less time (see Table 2).

Character set	Set cardinality	Time
4 digits		
Numbers	10	25 seconds
Special	17	3 minutes
Lower case / Upper case	26	19 minutes
Alpha	52	5 hours
Alphanumeric	62	10 hours
Full	79	27 hours
5 digits		
Numbers	10	4 minutes
Special	17	1 hour
Lower case / Upper case	26	8 hours
Alpha	52	11 days
Alphanumeric	62	26 days
Full	79	89 days
6 digits		
Numbers	10	41 minutes
Special	17	16 hours
Lower case / Upper case	26	8 days
Alpha	52	11 days
Alphanumeric	62	1.5 years
Full	79	19 years

Table 5: Maximum amount of time to crack passwords of different lengths with hashrate of 400 H/s.

2.3.2 Minimum password length

Let's assume now we have to find a **10-year-resistant password** (I will consider for the calculation that in a year there are approximately 3.156×10^7 seconds [6], therefore 3.156×10^8 seconds in 10 years).

Given S the character set and hps the number of hashes per second, the length (minimum number of digits in the password, i.e. lower bound) is computed as follows:

$$length = \log_{|S|} (3.156 \times 10^8 \times hps)$$

Using the formula above, we can easily compute how long our password has to be to resist a brute-force attack for 10 years, performed on hardware with the same specifications as mine (hashrate of 400 H/s). The results are shown in the Table 6.

Sources

[1] *SHA-1*. URL: `https://en.wikipedia.org/wiki/SHA-1` (visited on 10/2023).

[2] IDRIX. *Veracrypt*. URL: `https://www.veracrypt.fr/code/VeraCrypt/` (visited on 10/2023).

[3] *Hashcat*. URL: `https://hashcat.net/hashcat/` (visited on 10/2023).

Character set	**Set cardinality**	**Password length**
Numbers	10	11.1
Special	17	9.0
Lower case / Upper case	26	7.8
Alpha	52	6.4
Alphanumeric	62	6.1
Full	79	5.8

Table 6: Minimum average length for a 10-year resistant password with hashrate of 400 H/s.

[4] Hashcat. *Hash types*. URL: `https://hashcat.net/wiki/doku.php?id=example_hashes` (visited on 10/2023).

[5] *Hashcat benchmarks*. URL: `https://gist.github.com/Chick3nman/32e662a5bb63bc4f51b847bb422222fd` (visited on 10/2023).

[6] Google. *Unit Converter*. URL: `https://support.google.com/websearch/answer/3284611?hl=en-EN#unitconverter` (visited on 10/2023).