Evaluation of the security of password-protected encrypted RAR3 and RAR5 archives

Ehlimana Krupalija[1*], Saša Mrdović[1], Emir Cogo[1], Irfan Prazina[1], Šeila Bećirović[1]
[1]Department of Computer Science and Informatics, Faculty of Electrical Engineering, University of Sarajevo, Sarajevo, Bosnia and Herzegovina
[*]ekrupalija1@etf.unsa.ba

Abstract

Roshal Archive (RAR) format is one of the most widely used data archive formats, enabling users to reduce the size of data and protect it with the desired password before the data is transferred to its intended recipients over the network. This work focuses on the security of encrypted RAR archives and various different approaches for their decryption. Two different datasets composed of randomly generated and real-world user passwords were used for deploying brute force and dictionary attacks on password-protected RAR archives. Two available and widely used tools, John the Ripper and Hashcat, were used for cracking passwords of encrypted RAR3 and RAR5 archives. Experimental results indicate that both brute force and dictionary attacks were unsuccessful for RAR archives protected with randomly generated passwords, even of very small length. Real-world user passwords were successfully cracked only partially by brute force attacks, whereas dictionary attacks were very successful. The success rate for RAR5 archives was only slightly lower than for RAR3 archives and processing times were similar, indicating that this new version of the RAR format does not significantly improve data security. Instead, the security of RAR archives can be increased by using longer passwords more similar to randomly generated data, which are not present in commonly used dictionaries, as indicated by the experimental results.

Keywords: Password cracking, RAR format, Brute force attacks, Dictionary attacks, Software security.

1 Introduction

Data is often compressed into one of the many available archive formats (such as RAR and ZIP) in order to reduce its size so that it can be transferred via the network as quickly as possible. This trend was amplified in the recent years by the large availability of various cloud services offering remote storage and sharing [1], as well as the increasing internet bandwidth and speed. It became possible to quickly share large amounts of data with many users all over the world. This includes big data transportation for Internet of Things (IoT) purposes [2], storage of large amounts of geographical experiment data [3] and enhanced compression of encrypted images [4]. However, transferring data over the worldwide public network is not always safe, and malicious users can steal or copy the data, sometimes without the user even being aware of it (e.g. man-in-the-middle attack described in [5], [6] and [7]).
In order to increase the security of compressed data, the archives are often protected with passwords after the data compression process is complete. The user-defined password is used for encrypting the compressed data. This is meant to protect the data from malicious users by making the processing time of brute force attacks for data decryption unreasonably long and therefore ineffective (e.g. 152 days for passwords containing 7 characters [8]). The user can choose one of the available types of encryption formats, algorithms used for data compression and hashing functions of varying strength and processing speed.
This work explores the security of the Roshal Archive (RAR) format (version 3 and 5). The RAR format is analysed in detail, including the contents of password-protected archives and methods used for data encryption. The purpose of this work is to test whether the RAR5 format version offers a higher level of security than the RAR3 format version. Evaluation of different attacks was conducted to determine which attacks are more successful depending on password length, strength and similarity to real-world passwords chosen by users. In order to evaluate the security of data encryption of RAR archives and compare the security between RAR3 and RAR5 versions, two available password-cracking tools were used: John the Ripper [9], for deploying brute force attacks and Hashcat [10], for deploying dictionary attacks. Two different datasets consisting of randomly generated and user-defined passwords were used for encrypting RAR archive contents. The experimental results indicate that RAR5 version does not significantly improve the security of data, because the success rates for both RAR3 and RAR5 format types were more than 95%. RAR5 archives were cracked more quickly (77.5 s on average) by using the brute force attack. Matches for the dictionary attack were also found more quickly for RAR5 archives.
This work is structured as follows. In Section 2, the RAR format is explained in detail, as well as the differences between RAR3 and RAR5 types. Previous work on password cracking is also systematically explained. Section 3 gives a summary of datasets and dictionaries used for cracking encrypted RAR archives by using John the Ripper and Hashcat tools. The methodology for the entire password cracking process is explained. In Section 4, a detailed evaluation of the deployed attacks for different RAR format versions and the analysis of the experimental results are given. Section 5 offers a summarization of the presented concepts, including instructions for future work and possible improvements.

2 Background and related work

2.1 RAR format

Roshal Archive (RAR) format is one of the most frequently used formats for creating compressed data archives. The native software tool for creating RAR archives is WinRAR [11]. It is available on all popular operating systems (e.g. Microsoft Windows, Linux). The initial RAR version was released in 1995. It has gone through many changes which were necessary due to security issues of earlier standards. The initially used data encryption algorithm with 40-bit encryption keys [8] became insecure after the technological advancement led to the increased speed of brute force attacks. In 2002, the AES-128 encryption algorithm with 128-bit encryption keys was incorporated for data encryption [6]. It was replaced by AES-256 with 256-bit encryption keys in 2013 [7] with the introduction of the RAR5 format. The increase of encryption key size resulted in drastic increase of processing time for brute force attacks.

RAR archives can contain folders and files of different types. In order to be able to reconstruct the contents when unpacking the archive, all information is stored in various headers. Every RAR3 archive is composed of the marker block, archive header, file header, file contents and terminator block, as explained in detail in [12] and [13]. Without the usage of data encryption, all metainformation and archive contents can be directly retrieved. When data encryption is used for creating RAR3 archives, archive content cannot be directly read without decrypting the data first. The structure of an example encrypted RAR3 archive created by compressing a single file named *file.txt* with four bytes of content is shown on Figure 1. The usage of data encryption results in additional 8 bytes of data which contain the salt used for encryption.

The RAR5 format introduces many differences to the previously explained concepts [14]. RAR5 is not used as the default format by WinRAR. Instead, the user needs to manually specify that they want to use this format type. The reason for this is because the usage of the RAR5 format, while increasing the security of data, results in significant increase in archive size. The contents of the RAR5 archive are very different when compared to the corresponding RAR3 archives, as visible on Figure 2. All headers except for the marker block are variable in size and cannot be accurately located. The main reasons for the larger archive size are due to an increase in marker block size, cyclic redundancy check (CRC) checksum size, and salt size. New flags were added to the terminator block and a new header was added – the archive encryption header. This header contains information about the version of the encryption algorithm, additional checksums and password checks. All of this increases the security of the encrypted data and decreases the speed of attacks targeting checksums, salt values and encrypted archive contents.

2.2 Data encryption methods

The entire data encryption process for RAR archives is explained in [8], [15] and [16]. The randomly generated salt value is first appended to the password provided by the user, as well as 3 additional bytes for describing the current iteration number of the encryption process. This value is used as the input for Password-Based Key Derivation Function 1 (PBKDF1). It uses Hash-Based Message Authentication Code (HMAC/SHA256) [17] pseudorandom function in 262,144 iterations for generating the 256-bit output hash of the provided password. The hash is used as the secret key for the Advanced Encryption Standard (AES-256) block cipher, which is then used for encrypting the data contained in the RAR archive.

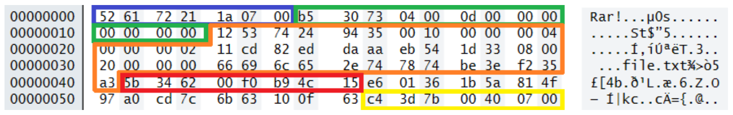
Encrypting the contents of the RAR archive through the usage of brute-force resistant methods increases the security of data. However, some approaches such as [6], [7] and [18] have shown that it is possible to retrieve the original contents by only using the metadata, which is not encrypted by default. For this reason, WinRAR also offers RAR archive metadata encryption. If the metadata is encrypted, neither the names and sizes of files, their extensions nor CRC checksums can be directly read from the archive, effectively blocking all attacks described in [5].

The usage of 8-byte and 16-byte salt size for RAR3 and RAR5 archives respectfully makes the number of combinations for rainbow table attacks too large for their successful deployment [19] [20]. However, RAR archives can be attacked an unlimited number of times (unlike e.g. website login forms which can have time limits and locks). This enables brute force, dictionary and hybrid attacks [21]. For this reason, additional capabilities enabled by WinRAR can be used to further enhance RAR archive security. For example, different passwords can be used for separate files and archives can be split into multiple volumes. The maximum password length for encryption is 127 characters [15], but there are no security requirements forcing the minimum password length and different character set usage. This means that the final security of the RAR archive depends on the end user.

2.3 Related work

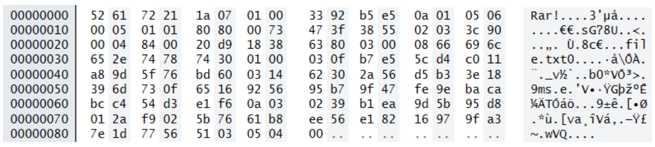
The main security issues which cannot be solved by the usage of bigger encryption keys and more secure encryption standards have been analysed in [5], [6] and [18]. If metadata is not encrypted, file contents can be guessed by using the checksum and file extensions. Even if parts of the archive are missing, they can be automatically repaired by using methods such as [22]. Making changes in compression method headers or names of encrypted files can enable the man-in-the-middle attack described in [5], [6] and [7] for successfully retrieving file contents.

Many different methods have been applied for evaluating the security of encrypted RAR archives. GPU parallelization techniques based on Computer Unified Device Architecture (CUDA) were used for manual AES key decryption in [23] and [24]. OpenCL GPU parallelization was applied in [25] and collaborative pipeline computing by using both CPU and GPU was proposed in [16]. Usage of distributed computing was proposed in [20] for achieving additional speedup. Brute force attacks were applied in [8] in order to prove that the time cost of password exhaustive search is



```
00000000 | 52 61 72 21 1a 07 00 b5 30 73 04 00 0d 00 00 00   Rar!...µ0s......
00000010 | 00 00 00 00 12 53 74 24 94 35 00 10 00 00 00 04   .....St$"5......
00000020 | 00 00 00 02 11 cd 82 ed da aa eb 54 1d 33 08 00   .....Í,íÚªëT.3..
00000030 | 20 00 00 00 66 69 6c 65 2e 74 78 74 be 3e f2 35    ...file.txt¾>ò5
00000040 | a3 5b 34 62 00 f0 b9 4c 15 e6 01 36 1b 5a 81 4f   £[4b.ð¹L.æ.6.Z.O
00000050 | 97 a0 cd 7c 6b 63 10 0f 63 c4 3d 7b 00 40 07 00   – Í|kc..cÄ={.@..
```

Figure 1 Structure of an example encrypted RAR3 archive: marker block (blue), archive header (green), file header (orange), salt (red), file content (no colour), terminator block (yellow)



```
00000000 | 52 61 72 21 1a 07 01 00 33 92 b5 e5 0a 01 05 06   Rar!....3’µå....
00000010 | 00 05 01 01 80 80 00 73 47 3f 38 55 02 03 3c 90   .....€€.sG?8U..<.
00000020 | 00 04 84 00 20 d9 18 38 63 80 03 00 08 66 69 6c   ..„. Ù.8c€...fil
00000030 | 65 2e 74 78 74 30 01 00 03 0f b7 e5 5c d4 c0 11   e.txt0.....·å\ÔÀ.
00000040 | a8 9d 5f 76 bd 60 03 14 62 30 2a 56 d5 b3 3e 18   ¨._v½`..b0*VÕ³>.
00000050 | 39 6d 73 0f 65 16 92 56 95 b7 9f 47 fe 9e ba ca   9ms.e.’V•·ŸGþžºÊ
00000060 | bc c4 54 d3 e1 f6 0a 03 02 39 b1 ea 9d 5b 95 d8   ¼ÄTÓáö...9±ê.[•Ø
00000070 | 01 2a f9 02 5b 76 61 b8 ee 56 e1 82 16 97 9f a3   .*ù.[va¸îVá‚.—Ÿ£
00000080 | 7e 1d 77 56 51 03 05 04 00 .. .. .. .. .. .. ..   ~.wVQ....
```

Figure 2 Structure of an example encrypted RAR5 archive

infeasibly long. The same method was applied in [8], [19] and [26] to prove brute-force attacks as ineffective for passwords containing more than 6 characters.
Password cracking of encrypted RAR archives is supported by many commercial and open-source tools, such as John the Ripper, Hashcat and Wrathion [26]. Cloud computing can also be used for deploying more resources in order to increase the processing speed [27]. John the Ripper and the proposed TDT model based on machine learning methods were used for cracking passwords of the RockYou dataset in [28]. Hashcat and the proposed distributed computing model have been applied for password cracking of encrypted RAR3 and RAR5 archives in [29]. John the Ripper and Hashcat were used for password cracking of encrypted RAR3 and RAR5 archives in [30] by using resource scheduling algorithms and GPU utilization. A comparison between John the Ripper and Hashcat, their available password-cracking modes and strengths and weaknesses for encrypted RAR archive password cracking was done in [31]. Rule-based attacks for reducing the processing time of John the Ripper and Hashcat attacks were used in [32].

3 Methods

Two types of data were used for encrypting RAR archives:

1. Data extracted from the RockYou dataset available at [33], which contains 14,341,564 unique real-world passwords of 32,603,388 user accounts. Only the first 210 passwords were used for evaluation due to the time-consuming process of password cracking which, in some cases, did not terminate after reaching the time limit of 2 hours of attempting different combinations of input characters. Due to equipment limitations and the inability to use GPU parallelization techniques, a total processing time of over 400 hours was needed to process these passwords. However, due to the purpose of comparing RAR3 and RAR5 security by using the same passwords to protect the archives, 210 randomly selected rows were sufficient for a meaningful evaluation. The distribution of password length in the evaluation subset of the RockYou dataset is shown on Figure 3. Most passwords are 6 and 7 characters long (47.62% and 20% respectfully). Only 3 passwords of length 10 (1.43%) and 19 passwords of length 9 (9.05%) are present in the subset.

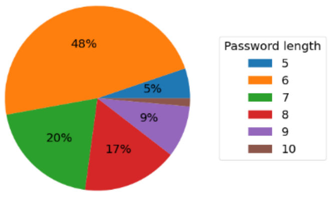


Figure 3 RockYou dataset password length distribution

2. Data consisting of randomly generated passwords, which do not contain any similarity to real-world data. Password strength is based on the size of the character set used for generating the password (inspired by basic elements of user passwords shown in [28]), as shown in Table 1. The lowest security is offered by the character set containing only lowercase English letters (total $26^{noOfCharacters}$ permutations necessary for exhaustive password search) whereas the highest security is offered by the character set containing lowercase and uppercase English letters, digits, special and language-specific characters (total $95^{noOfCharacters}$ permutations necessary for exhaustive password search).

Table 1 Different password strengths based on character sets used for generating the desired password

Character set type	**Character set size**	**Password strength**
a-z	26	1
a-z, A-Z	52	2
a-z, A-Z, 0-9	62	3
a-z, A-Z, 0-9, special characters[1]	90	4
a-z, A-Z, 0-9, special characters, language-specific characters[2]	95	5

[1] The following 28 special characters were used: !, “, #, $, %, &, /, (,), =, ‘, ?, +, *, „ ;, ., :, -, _, <, >, @, {, }, [,] and \

[2] The following 5 language-specific characters were used: č, ć, đ, š and ž

Depending on the type of data (randomly generated passwords or user passwords), different dictionaries were used for deploying dictionary attacks in Hashcat. A compilation of dictionaries from which the dictionaries were selected are available at [34]. Five dictionaries containing a total of 8,017,883 passwords were used for the RockYou dataset. Twenty dictionaries containing a total of 9,022,794 passwords were used for the randomly generated password dataset. This increase in password space size was necessary due to the lack of similarity of randomly generated passwords with user passwords.

The password-cracking process consists of three iteratively repeated steps: creating a RAR archive and protecting it with a password, attempting to crack the password by using the available tools and exporting information about the success of cracking and execution time. The user needs to specify which type of dataset they want to use (randomly generated password or RockYou dataset), their desired cracking tool (John the Ripper or Hashcat), type of RAR archive (RAR3 or RAR5 archive) and the desired password length and strength. If the randomly generated password dataset is used, then passwords of different lengths and strengths are iteratively randomly generated, otherwise the first 210 rows of the RockYou dataset are used. After the desired password is obtained, the password is used for generating a new RAR archive. For this purpose, the console version of WinRAR is used. The RAR archive contains a 4-byte *.txt* file. It is encrypted using the provided password. Hashcat and John the Ripper tools cannot directly use encrypted RAR archives as input. Instead, the hash of the encrypted archive must first be extracted into a *.txt* file. For this purpose, the *rar2john* tool is used.

The final step of the password-cracking process is the execution of the desired tool on the extracted hash. The desired tool is executed through a shell command as a separate process without the use of multithreading. If the Hashcat tool is used, various dictionaries are used for attempting the dictionary attack on the password hash. If multiple attacks are successful, only the lowest execution time (the fastest match with existing passwords from the dictionaries) is recorded. In case of John the Ripper tool, the cracking is attempted only once by using the brute force attack. Negative overall execution time (-1) is used to describe the situation when no attacks are successful.

4 Results

Evaluation was performed on a single machine with the following CPU specifications: Processor Intel(R) Core (TM) i5-7200U CPU @ 2.50GHz, 2712 Mhz, 2 Core(s), 4 Logical Processor(s) and 4 GB of RAM memory.

4.1 Randomly generated password dataset

The processing time for password cracking was limited to a maximum of 2 hours per password for the randomly generated password dataset, due to equipment limitations. Results achieved by using the brute force attack in John the Ripper are shown on Figure 4. The average processing time is lower for RAR5 archives (724.4 s < 801.9 s) mainly due to the very low processing time for passwords containing 1 and 2 characters. The highest processing time was achieved for the RAR5 version (5,043.3 s). The average processing time was very high for passwords containing 4 characters (4,293.6 s for RAR3 and 5,043 s for RAR5).

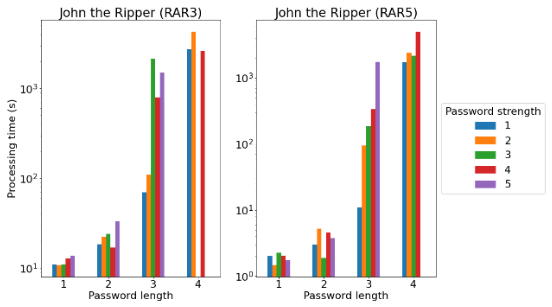


Figure 4 Speed of password cracking by using John the Ripper

Results achieved by using the dictionary attack utilized in Hashcat did not result in exponential processing time increase, as shown on Figure 5. High processing times were achieved for passwords of all lengths and strengths, due to the nature of the dictionary attack. Regardless of the RAR format type, only a small number of passwords of length 3 and 4 were successfully cracked. The average processing times for RAR5 archives were considerably smaller (310.8 s as opposed to 2,169.6 s for RAR3 archives), although the success rate of cracking for RAR5 was smaller only by a single password.

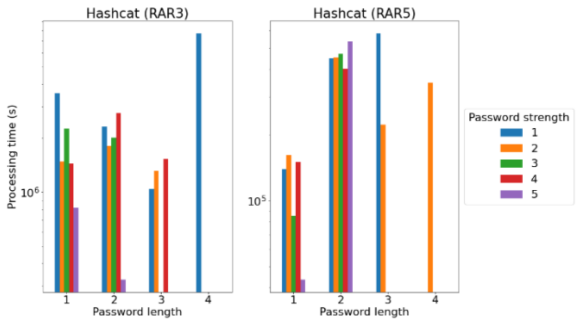


Figure 5 Speed of password cracking by using Hashcat

4.2 RockYou dataset

The processing time for password cracking was limited to a maximum of 300 s for the RockYou dataset, in order to maximize the number of processed passwords. This was done because brute force attacks do not terminate in feasible time for passwords longer than five characters, and most dictionary attacks terminate before reaching this time limit. The results achieved by using John the Ripper and Hashcat are shown in Table 2. It is visible that the success rate of password cracking by using the dictionary attack utilized in Hashcat is much higher than the success rate by using the brute force attack utilized in John the Ripper (58.58% on average). Comparing the successfully cracked passwords by password length indicates that the success rate for longer passwords is significantly higher for the dictionary attack. This was expected due to the exponential increase in the number of combinations for every additional character, reducing the success rate of the brute force attack significantly.

Table 2 Summary of successfully cracked passwords by password length using John the Ripper and Hashcat

Password length	**John (RAR3)**	**John (RAR5)**	**Hashcat (RAR3)**	**Hashcat (RAR5)**
5	6	7	11	10
6	38	45	98	98
7	15	15	39	40
8	10	14	34	34
9	4	5	18	17
10	0	0	3	3
Success rate	34.76%	40.95%	96.67%	96.19%

The success rate of password cracking for RAR3 and RAR5 archives by using the dictionary attack differed only by one additional RAR3 password. However, the difference when using the brute approach was significant (13 additional RAR5 passwords were successfully cracked). Figure 6 shows the password-cracking speed for passwords which were successfully cracked by John the Ripper and Hashcat. The largest number of passwords were successfully cracked in less than 120 s of processing time for both tools, regardless of RAR archive type. The number of successfully cracked passwords with processing time longer than 120 s was very low for the brute force attack. John the Ripper successfully recovered only 12 RAR3 passwords and 11 RAR5 passwords in this execution time period, whereas this is not true for the dictionary attack (Hashcat successfully recovered 90 RAR3 passwords and 77 RAR5 passwords in this execution time period).

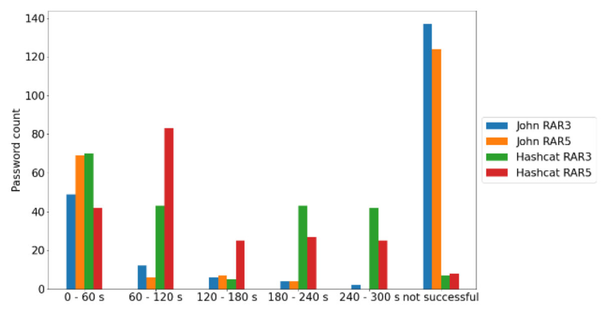


Figure 6 Distribution of cracked passwords by processing time using John the Ripper and Hashcat tools

Figure 7 and 8 show the ratio of cracked passwords by processing time and password length using John the Ripper and Hashcat for RAR3 and RAR5 types. The highest number of passwords cracked by John the Ripper was cracked in less than 60 s and most of the passwords were 6 or 7 characters long. The most significant difference when comparing results achieved by John the Ripper for different archive types is that for RAR5 archives, a lower number of 6-character passwords and a higher number of 8-character passwords were cracked in the next 60 s (between 60 and 120 s) of processing time. These passwords were successfully cracked in the first 60 s instead, bringing the percentage of 32.88% for RAR3 up to 40.70% for RAR5. The ratio of cracked passwords was significantly different for the dictionary attack utilized in the Hashcat tool. Passwords of length 6-8 were continually successfully cracked between 0 and 300 s of processing time as opposed to John the Ripper cracking, which was mostly unsuccessful after 120 s of processing time. The most significant difference when comparing the ratio between RAR3 and RAR5 archives is the speed of cracking. Hashcat successfully cracked 19.21% RAR3 passwords of length 6 in the first 60 s of processing time and 6.90% RAR3 passwords of length 6 after 120 s of processing time (a total of 26.11%). For RAR5 archives, 9.41% passwords of length 6 were successfully cracked in the first 60 s of processing time and 23.27% passwords of length 6 after 120 s of processing time (a total of 32.68%).

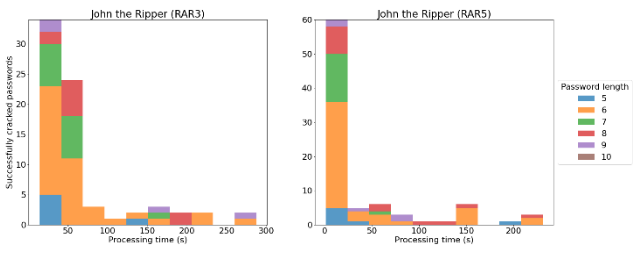


Figure 7 Ratio of cracked passwords by cracking time and password length successfully cracked by John the Ripper

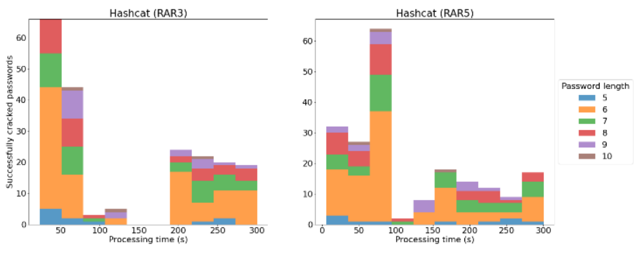


Figure 8 Ratio of cracked passwords by cracking time and password length successfully cracked by Hashcat

The dictionary attack utilized in the Hashcat tool resulted in different password-cracking speed for RAR3 and RAR5 types. An analysis of the success of each dictionary for password cracking is shown on Figure 9. 88.61% of all cracked passwords were successfully matched in the first three dictionaries for RAR5 archives as opposed to RAR3 archives, where only 49.26% of all cracked passwords were successfully matched in the first three dictionaries.

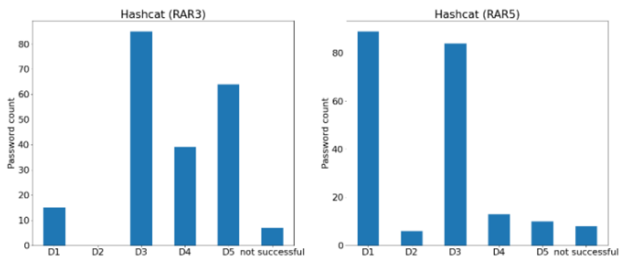


Figure 9 Distribution of cracked passwords by dictionaries in which a match was found using Hashcat

4.3 Discussion

The results achieved on the randomly generated password dataset indicate that RAR5 archives did not offer a higher level of protection. The average processing times were lower for all RAR5 archives, and both brute force and dictionary attacks were able to crack passwords of RAR5 archives more quickly. The dictionary attack was far more successful for the RockYou dataset and the success rate for RAR5 type was smaller only by 0.48%. This password-cracking efficiency is higher than the results achieved in [29] (~4% drop in efficiency), even though clusters of computers were not used for evaluation. A percentage of RAR5 passwords higher by 6.57% was cracked in 120 s of processing time. Best password-cracking time of 5.5 hours for character length of 9 characters achieved by the distributed computing approach [20] was far outperformed by using Hashcat, with a time limit of 300 s. The dictionary attack (with average success rate of 96.43%) was also more successful than the TDT model based on machine learning methods proposed in [28] (with hit rate of up to 41% for the customized RockYou dataset). The results also show that dictionary entries are processed faster for RAR5 archives, which results in a higher percentage of matches in the first three dictionaries. All of this indicates that the usage of RAR5 archive type, which significantly increases archive size as the main drawback, does not significantly increase the processing time but instead results in a higher success rate for the brute force attack and a similar attack success rate for the dictionary attack.

5 Conclusion

The availability of many tools which utilize brute force, dictionary and hybrid attacks has endangered the security of password-protected RAR archives. The purpose of the RAR5 format was meant to increase security of encrypted data. In this work, the security of encrypted RAR3 and RAR5 archives was evaluated by using John the Ripper and Hashcat tools. Two types of data, including randomly generated and real-world user passwords of different length and strength were used in this process. Brute force attacks were ineffective even for randomly generated passwords of small sizes due to very high processing time, whereas they were partially successful on the RockYou dataset. Dictionary attacks did not have the same drawbacks regarding processing time, resulting in very high success rates for the RockYou dataset. However, the success on randomly generated passwords was very limited. This indicates that randomly generated passwords offer a higher level of security for RAR archives. The RAR5 format, which is meant to improve security of data and make the processing time of brute force attacks astronomically high, did not yield the expected results. The success rate for real-time user passwords was comparable for both format types, and the processing times were similar (the speed of cracking for the RAR5 version was even considerably higher in the dictionary attack on randomly generated passwords). Matches of passwords were found faster (in the first two dictionaries) for RAR5 as opposed to RAR3. This indicates that the usage of the RAR5 format did not considerably improve the security of encrypted data. However, the success rate of the deployed attacks mainly depends on the password used to protect the archive. Therefore, WinRAR software should prohibit usage of short passwords which are easy to crack and every password should contain characters from different character sets. This way, password strength can be significantly increased. RAR5 archives should also use metadata encryption methods by default. This would make the password-cracking process significantly longer, as it would be harder to obtain the password hash and compare it to hashes generated by the cracking tools.
It is important to note that GPU parallelization techniques were not used for speeding up the cracking process. Due to this drawback, only a small portion of the RockYou dataset could be used for evaluation. Utilizing the GPU functionalities could lead to a better evaluation of a larger portion of the RockYou dataset. The processing times can also be lowered by directly attacking the archive contents instead of using the aid of software tools. The tool start-up and initialization process are included in the total password cracking time, which also reduces the available cracking time. Splitting archives into multiple volumes and metadata encryption techniques should also be explored in order to verify whether the RAR5 format version offers a higher level of security when additional protection methods are applied.

6 Literature

[1] C. S. Kumar and K. V. Kumar, “An enhanced data integrity technique for cloud storage with integrated archive using PDP,” *International Journal of Engineering & Technology,* no. 7, pp. 905-907, 2018.

[2] Q. Jiancheng, L. Yiqin and Z. Yu, “Parallel algorithm for wireless data compression and encryption,” *Journal of Sensors,* vol. 2017, no. 4209397, 2017.

[3] Y. Zhang, Y.-G. Wang, Y.-P. Bai, Y.-Z. Li, Z.-Y. Lv and H.-W. Ding, “A new rockburst experiment data compression storage algorithm based on big data technology,” *Intelligent Automation and Soft Computing,* vol. 25, no. 3, pp. 561-572, 2019.

[4] N. S. Noor, D. A. Hammood, A. Al-Naji and J. Chahl, “A fast text-to-image encryption-decryption algorithm for secure network communication,” *Computers,* vol. 11, no. 39, 2022.

[5] T. Kohno, “Attacking and repairing the WinZip encryption scheme,” in *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS 2004)*, Washington, DC, 2004.

[6] G. S.-W. Yeo and R. C.-W. Phan, “On the security of the WinRAR encryption feature,” *International Journal of Information Security,* vol. 5, no. 2, pp. 115-123, 2006.

[7] K. Arthur-Durett, “The weakness of WinRAR encrypted archives to compression side-channel attacks,” Open Access Theses, West Lafayette, 2014.

[8] J. Chen, J. Zhou, K. Pan, S. Lin, C. Zhao and X. Li, “The security of key derivation functions in WinRAR,” *Journal of Computers,* vol. 8, no. 9, pp. 2262-2268, 2013.

[9] “John the Ripper password cracker,” Openwall, [Online]. Available: https://www.openwall.com/john/. [Accessed 25 August 2022].

[10] “hashcat - advanced password recovery,” [Online]. Available: https://hashcat.net/hashcat/. [Accessed 25 August 2022].

[11] “WinRAR 6.11 - Compress, Encrypt, Package and Backup with only one utility,” RARLAB, [Online]. Available: https://www.win-rar.com/start.html?&L=0. [Accessed 25 August 2022].

[12] “RAR,” Forensics Wiki, [Online]. Available: https://forensicswiki.xyz/page/RAR. [Accessed 25 August 2022].

[13] J. Schiller, “The RAR Format,” [Online]. Available: https://codedread.github.io/bitjs/docs/unrar.html. [Accessed 25 August 2022].

[14] “RAR 5.0 archive format,” RARLAB, [Online]. Available: https://www.rarlab.com/technote.htm. [Accessed 25 August 2022].

[15] “WinRAR Encryption Frequently asked questions (FAQ),” RARLAB, [Online]. Available: https://www.win-rar.com/encryption-faq.html?&L=0. [Accessed 25 August 2022].

[16] Q. Ji and H. Yin, “Speedup and password recovery for encrypted WinRAR3 without encrypting filename on GPUs,” in *Journal of Physics: Conference Series - 2020 6th Annual International Conference on Computer Science and Applications*, Guangzhou, 2020.

[17] J. Katz and Y. Lindell, “Introduction to Modern Cryptography,” CRC Press, Boca Raton, 2021.

[18] T. Kohno, “Analysis of the WinZip encryption method,” *IACR ePrint Archive,* no. 078, pp. 1-19, 2004.

[19] J. A. Chester, “Analysis of password cracking methods & applications,” *Honors Research Projects,* no. 7, 2015.

[20] R. Hranický, M. Holkovič and P. Matoušek, “On efficiency of distributed password recovery,” *The Journal of Digital Forensics, Security and Law,* vol. 11, no. 2, pp. 79-96, 2016.

[21] T. Kakarla, A. Mairaj and A. Y. Javaid, “A real-world password cracking demonstration using open source tools for instructional use,” in *2018 IEEE International Conference on Electro/Information Technology (EIT)*, Rochester, 2018.

[22] Y. Wei, N. Zheng and M. Xu, “An automatic carving method for RAR file based on content and structure,” in *2010 Second International Conference on Information Technology and Computer Science*, Kiev, 2010.

[23] G. Hu, J. Ma and B. Huang, “Password recovery for RAR files using CUDA,” in *2009 Eight IEEE International Conference on Dependable, Autonomic and Secure Computing*, Chengdu, 2009.

[24] Y. Zhang and G.-d. Sheng, “RAR password decryption by utilizing GPU,” in *The 2010 International Conference of Apperceiving Computing and Intelligence Analysis Proceeding*, Chengdu, 2010.

[25] X. An, H. Zhao, L. Ding, Z. Fan and H. Wang, “Optimized password recovery for encrypted RAR on GPUs,” in *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, New York, 2015.

[26] R. Hranický, P. Matoušek, O. Ryšavý and V. Veselý, “Experimental evaluation of password recovery in encrypted documents,” in *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016)*, Rome, 2016.

[27] G. Korlam, “Password cracking in the cloud,” UC Santa Barbara, Santa Barbara, 2012.

[28] H.-C. Chou, H.-C. Lee, H.-J. Yu, F.-P. Lai, K.-H. Huang and C.-W. Hsueh, “Password cracking based on learned patterns from disclosed passwords,” *International Journal of Innovative Computing, Information and Control,* vol. 9, no. 2, pp. 821-839, 2013.

[29] R. Hranický, L. Zobal, V. Večeřa and P. Matoušek, “Distributed password cracking in a hybrid environment,” in *Proceedings of Security and Protection of Information (SPI) 2017*, Brno, 2017.

[30] P. Kubelka, “Password recovery job scheduling for online deep file analysis,” Czech Technical University in Prague, Prague, 2020.

[31] D. Pahuja and P. Sidana, “Implementing and comparing different password cracking tools,” *International Research Journal of Engineering and Technology (IRJET),* vol. 8, no. 5, pp. 2089-2095, 2021.

[32] E. Liu, A. Nakanishi, M. Golla, D. Cash and B. Ur, “Reasoning analytically about password-cracking software,” in *2019 IEEE Symposium on Security and Privacy*, San Francisco, 2019.

[33] W. J. Burns, “Common Password List (rockyou.txt),” Kaggle, 13 January 2019. [Online]. Available: https://www.kaggle.com/datasets/wjburns/common-password-list-rockyoutxt. [Accessed 23 August 2022].

[34] D. Miessler, J. Haddix and g0tmi1k, “SecLists - The Pentester's Companion,” GitHub, 19 December 2017. [Online]. Available: https://github.com/danielmiessler/SecLists. [Accessed 23 August 2022].