John the Ripper: Password Security Analysis

Elif Nur Aslıhan Celepoğlu

CONTENTS

23:11

1.DISCLAIMER

This project was prepared for educational and informational purposes only. In the project, programs, codes and other materials and knowledge were used for understanding cybersecurity fundamentals and hacking defense.

2.OBJECTIVE

We are conducting an investigation using the John the Ripper tool to analyze the effectiveness of password policies and protection measures. Through this process, we gather information and data, allowing us to draw conclusions. Additionally, we determine which methods provide better security. We compile this information into a user-friendly interface to facilitate the user's tasks.

3.TOOLBOX

-Kali Linux
-For cracking using John The Ripper
-Cracking method is Dictionary Mode
-For interface of the application using Bash Script
-For hash generation online cracking decoder

- https://10015.io/tools/md5-encrypt-decrypt
- https://10015.io/tools/sha1-encrypt-decrypt
- https://10015.io/tools/sha256-encrypt-decrypt
- https://10015.io/tools/sha512-encrypt-decrypt
- https://md5decrypt.net/en/Md4/

In the pursuit of password analysis, our methodology involves leveraging the capabilities of Kali Linux, a robust penetration testing platform. Utilizing the renowned John the Ripper tool, we employ the Worldlist Method for password cracking. The interface of our application is designed with efficiency in mind, utilizing a Bash Script for seamless usability.

4.THEORETICAL BACKGROUND

In this part, there are basic definitions and terminology for code breaking.

What is the John The Ripper ?

John the Ripper is an open-source password cracking software used for security testing and password analysis. Its primary goal is to identify and strengthen weaknesses in passwords through testing. In addition to focusing on brute-force attacks, it also supports dictionary and hybrid attacks.
Here are some attack modes of John the Ripper:

- **Brute-force Attacks (Single Crack Mode)**

Tries all password combinations, time-consuming. Users protect by using complex passwords.

- **Dictionary Attacks (Wordlist Mode)**

Uses predefined dictionaries for password cracking, effective for common or weak passwords.

- **Hybrid Attacks**

Combines brute-force and dictionary attacks, optimizing with a word list before trying remaining combinations.

What is the Bash Script ?

A Bash script is a text-based script or program written in the Bash shell scripting language, commonly used for automating tasks or executing a sequence of commands. It is frequently employed in Linux and other Unix-like operating systems, often featuring reusable and functional code blocks.

What is Hash Function ?

Hash functions are mathematical operations converting input data into fixed-size byte strings, generating unique digests.
Key properties include;

Determinism (consistent output for the same input)
Quick Computation (to return the hash value quickly.)
Pre-image Resistance (The original input should be computationally infeasible to derive from the hash value.)
Small Changes in Input Change the Output: (Small input changes should result in significantly different outputs.)
Collision Resistance (Finding two different inputs that yield the same output should be challenging.)
Puzzle Friendly (Finding an input that produces a given output should be difficult.)

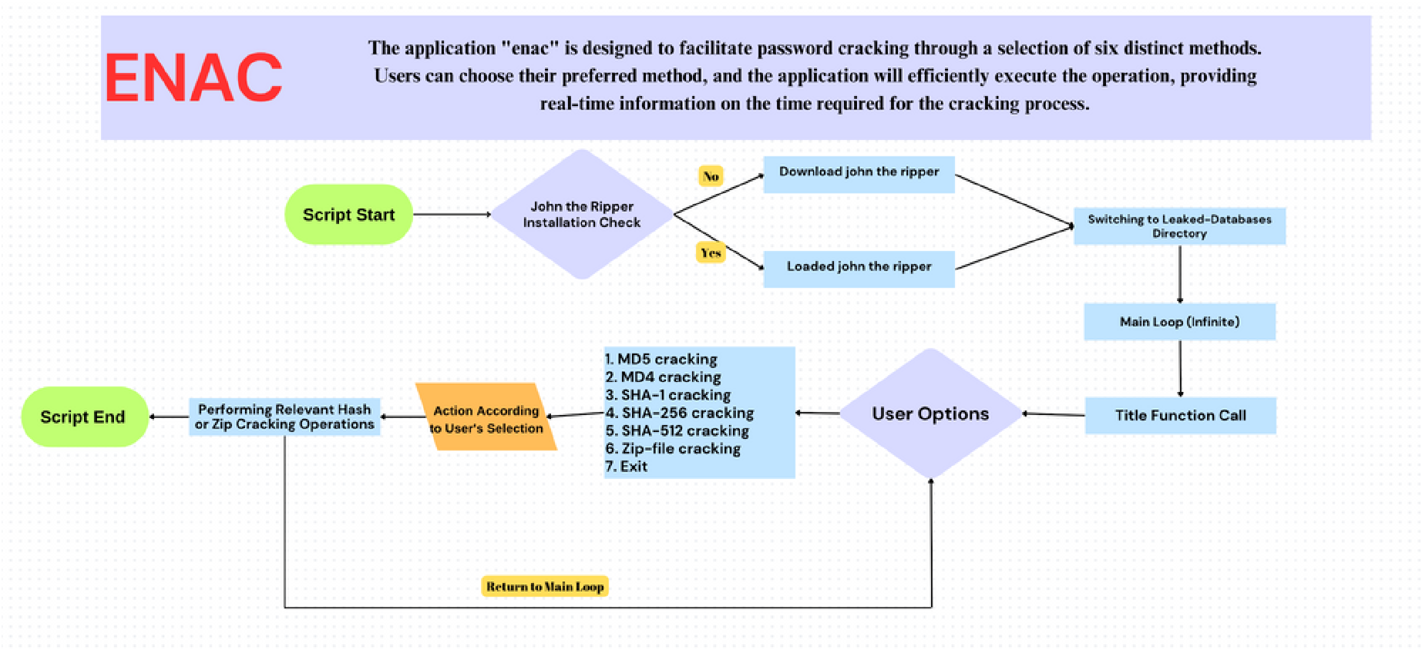

The application "enac" is designed to facilitate password cracking through a selection of six distinct methods. Users can choose their preferred method, and the application will efficiently execute the operation, providing real-time information on the time required for the cracking process.

Figure-1

Integration and Execution:

- The integration of these methods into a single platform streamlines the password cracking process.
- A Bash script is employed to execute the selected cracking method, providing a user-friendly interface.

Time Measurement:

- The application includes a feature to measure the time required for each cracking operation.
- This functionality assists users in assessing the efficiency of each method and making informed decisions.

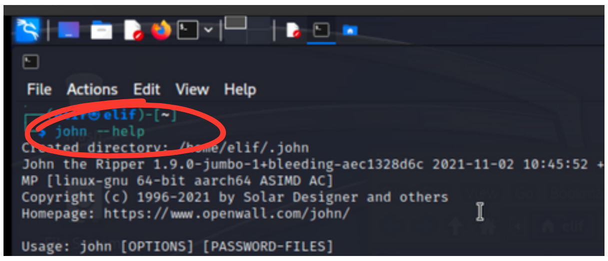
5.PROJECT STEPS

5.1 WHAT WAS DISCUSSED IN THE FIRST REVIEW ?

1. Vmmare -Fusion-13.5.0 download

2. Kali linux -arm64

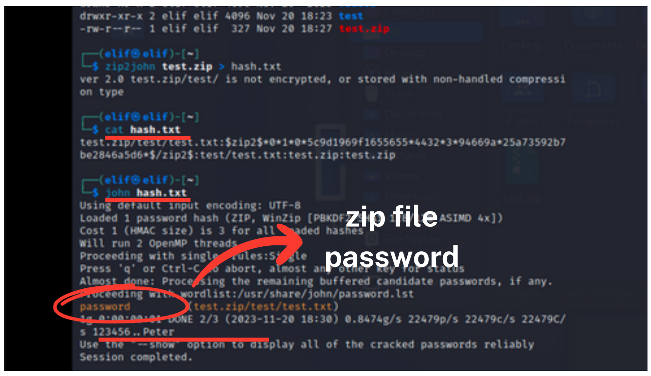
3. We are creating encrypted zip file to crack John the ripper zip file.



```
┌──(elif㉿elif)-[~]
└─$ john --help
Created directory: /home/elif/.john
John the Ripper 1.9.0-jumbo-1+bleeding-aec1328d6c 2021-11-02 10:45:52 +
MP [linux-gnu 64-bit aarch64 ASIMD AC]
Copyright (c) 1996-2021 by Solar Designer and others
Homepage: https://www.openwall.com/john/

Usage: john [OPTIONS] [PASSWORD-FILES]
```

Figure-2

4. Breaking the zip file password



```
drwxr-xr-x 2 elif elif 4096 Nov 20 18:23 test
-rw-r--r-- 1 elif elif  327 Nov 20 18:27 test.zip

┌──(elif㉿elif)-[~]
└─$ zip2john test.zip > hash.txt
ver 2.0 test.zip/test/ is not encrypted, or stored with non-handled compressi
on type

┌──(elif㉿elif)-[~]
└─$ cat hash.txt
test.zip/test/test.txt:$zip2$*0*1*0*5c9d1969f1655655*4432*3*94669a*25a73592b7
be2846a5d6*$/zip2$:test/test.txt:test.zip:test.zip

┌──(elif㉿elif)-[~]
└─$ john hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (ZIP, WinZip [PBKDF2-SHA1 128/128 ASIMD 4x])
Cost 1 (HMAC size) is 3 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
password         (test.zip/test/test.txt)
1g 0:00:00:01 DONE 2/3 (2023-11-20 18:30) 0.8474g/s 22479p/s 22479c/s 22479C/
s 123456..Peter
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Figure-3

1.step: zip2john test.zip >hash.txt
2.step: cat hash.txt
3.step : john hast.txt

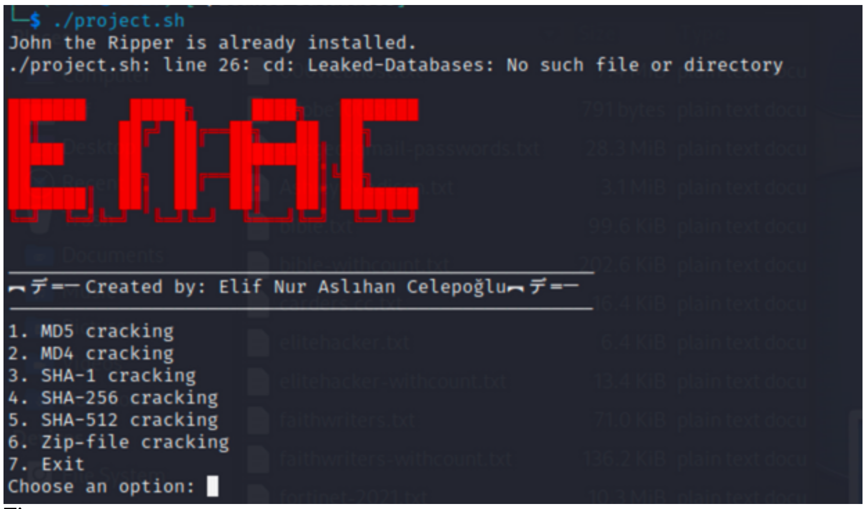
5.2 WHAT SHOULD I TALK ABOUT IN THE SECOND REVIEW?

In the second presentation, I will tell you about successfully merging the methods I used. I integrated all these methods on the same platform, and I used a bash script to achieve this. The name of my application is "enac," and there are a total of six different cracking methods. By selecting any of these methods, we can quickly perform the operation. Additionally, it shows how long it takes to crack.

Six different cracking methods:

- md5 cracking
- md4 cracking
- sha-1 cracking
- sha-256 cracking
- sha -512 cracking
- zip-file cracking

(interface of the application)

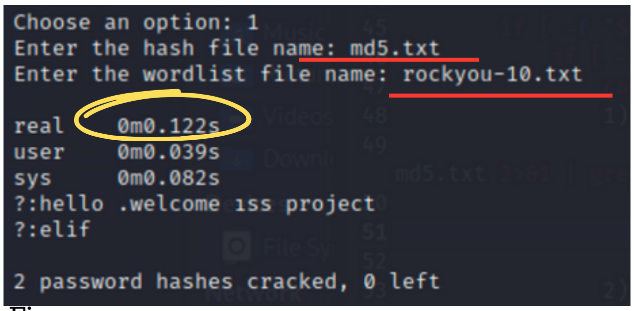


```
└─$ ./project.sh
John the Ripper is already installed.
./project.sh: line 26: cd: Leaked-Databases: No such file or directory
```

```
¬デ=─ Created by: Elif Nur Aslıhan Celepoğlu¬デ=─

1. MD5 cracking
2. MD4 cracking
3. SHA-1 cracking
4. SHA-256 cracking
5. SHA-512 cracking
6. Zip-file cracking
7. Exit
Choose an option:
```

Figure-4

MD5 CRACKING OPTION 1



```
Choose an option: 1
Enter the hash file name: md5.txt
Enter the wordlist file name: rockyou-10.txt

real    0m0.122s
user    0m0.039s
sys     0m0.082s
?:hello .welcome ıss project
?:elif

2 password hashes cracked, 0 left
```

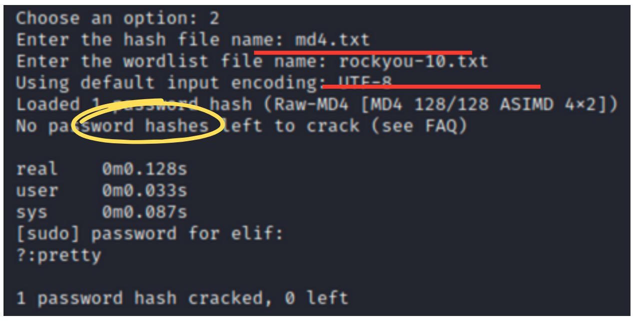
Figure-5

0m0.122s

1.step: time john --format=raw-md5 --wordlist=rockyou-10.txt md5.tx
2.step: sudo john --show --format=raw-md5 "$hash_file"

- **real:** Elapsed real (wall clock) time used by the process, in seconds.
- **user:** Total number of CPU-seconds that the process
- **sys:** Total number of CPU-seconds used by the system on behalf of the process (in kernel mode), in seconds

MD4 CRACKING OPTION 2



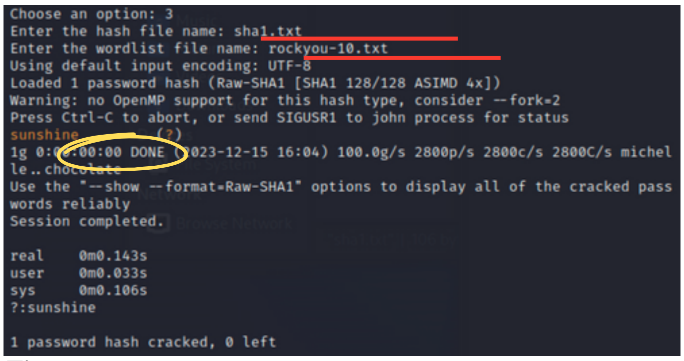
```
Choose an option: 2
Enter the hash file name: md4.txt
Enter the wordlist file name: rockyou-10.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD4 [MD4 128/128 ASIMD 4×2])
No password hashes left to crack (see FAQ)

real    0m0.128s
user    0m0.033s
sys     0m0.087s
[sudo] password for elif:
?:pretty

1 password hash cracked, 0 left
```

Figure-6

0m0.128s

1.step: time john --format=raw-md4 --wordlist="$wordlist" "$hash_file"
2.step: sudo john --show --format=raw-md4 "$hash_file"

SHA-1 CRACKING OPTION 3



```
Choose an option: 3
Enter the hash file name: sha1.txt
Enter the wordlist file name: rockyou-10.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA1 [SHA1 128/128 ASIMD 4x])
Warning: no OpenMP support for this hash type, consider --fork=2
Press Ctrl-C to abort, or send SIGUSR1 to john process for status
sunshine         (?)
1g 0:00:00:00 DONE (2023-12-15 16:04) 100.0g/s 2800p/s 2800c/s 2800C/s michel
le..chocolate
Use the "--show --format=Raw-SHA1" options to display all of the cracked pass
words reliably
Session completed.

real    0m0.143s
user    0m0.033s
sys     0m0.106s
?:sunshine

1 password hash cracked, 0 left
```

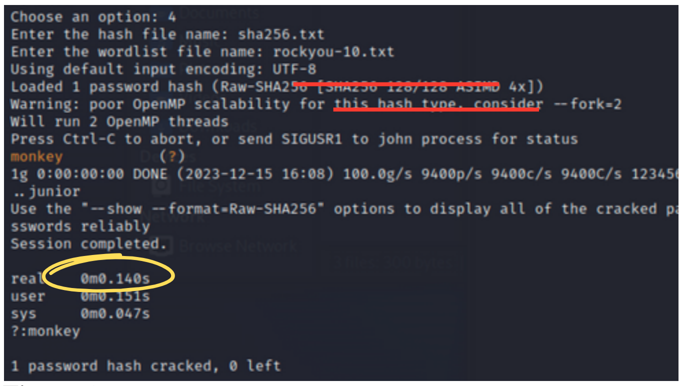
Figure-7

0m0.143s

1.step: time john --format=raw-sha1 --wordlist="$wordlist" "$hash_file"

2.step: sudo john --show --format=raw-sha1 "$hash_file"

SHA-256 CRACKING OPTION 4



```
Choose an option: 4
Enter the hash file name: sha256.txt
Enter the wordlist file name: rockyou-10.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA256 [SHA256 128/128 ASIMD 4x])
Warning: poor OpenMP scalability for this hash type, consider --fork=2
Will run 2 OpenMP threads
Press Ctrl-C to abort, or send SIGUSR1 to john process for status
monkey           (?)
1g 0:00:00:00 DONE (2023-12-15 16:08) 100.0g/s 9400p/s 9400c/s 9400C/s 12345
..junior
Use the "--show --format=Raw-SHA256" options to display all of the cracked pa
sswords reliably
Session completed.

real    0m0.140s
user    0m0.151s
sys     0m0.047s
?:monkey

1 password hash cracked, 0 left
```

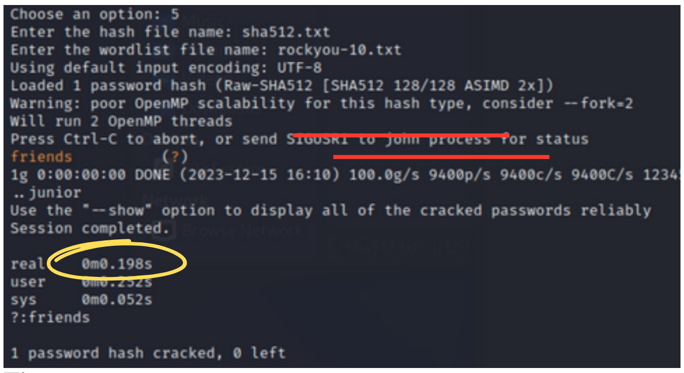
Figure-8

0m0.140s

1.step: time john --format=raw-sha256 --wordlist="$wordlist" "$hash_file"

2.step: sudo john --show --format=raw-sha256 "$hash_file"

SHA-512 CRACKING OPTION 5



```
Choose an option: 5
Enter the hash file name: sha512.txt
Enter the wordlist file name: rockyou-10.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA512 [SHA512 128/128 ASIMD 2x])
Warning: poor OpenMP scalability for this hash type, consider --fork=2
Will run 2 OpenMP threads
Press Ctrl-C to abort, or send SIGUSR1 to john process for status
friends          (?)
1g 0:00:00:00 DONE (2023-12-15 16:10) 100.0g/s 9400p/s 9400c/s 9400C/s 1234
..junior
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

real    0m0.198s
user    0m0.252s
sys     0m0.052s
?:friends

1 password hash cracked, 0 left
```

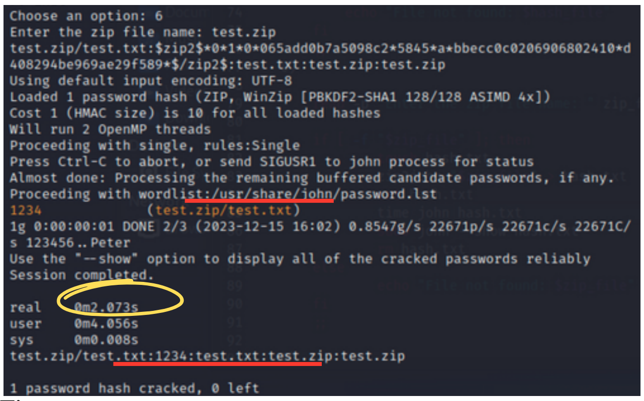
Figure-9

0m0.198s

1.step: time john --format=raw-sha512 --wordlist="$wordlist" "$hash_file"

2.step: sudo john --show --format=raw-sha512 "$hash_file"

ZIP FILE CRACKING OPTION 6



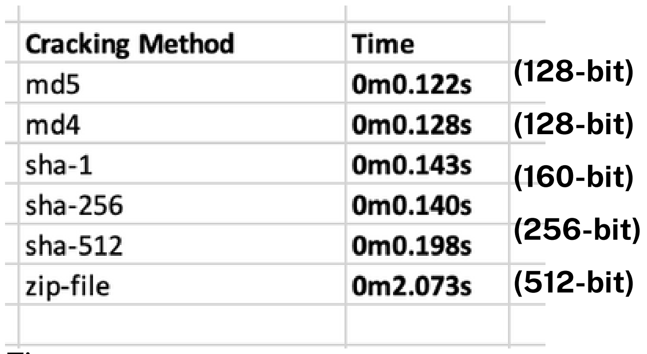
```
Choose an option: 6
Enter the zip file name: test.zip
test.zip/test.txt:$zip2$*0*1*0*065add0b7a5098c2*5845*a*bbecc0c0206906802410*d
408294be969ae29f589*$/zip2$:test.txt:test.zip:test.zip
Using default input encoding: UTF-8
Loaded 1 password hash (ZIP, WinZip [PBKDF2-SHA1 128/128 ASIMD 4x])
Cost 1 (HMAC size) is 10 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press Ctrl-C to abort, or send SIGUSR1 to john process for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
1234             (test.zip/test.txt)
1g 0:00:00:01 DONE 2/3 (2023-12-15 16:02) 0.8547g/s 22671p/s 22671c/s 22671C/
s 123456..Peter
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

real    0m2.073s
user    0m4.056s
sys     0m0.008s
test.zip/test.txt:1234:test.txt:test.zip:test.zip

1 password hash cracked, 0 left
```

Figure-10

0m2.073s

1.step: touch hash.txt
2.step: zip2john "$zip_file" > hash.txt
3.step: cat hash.txt
4.step: time John hash.txt
5.step: sudo john --show hash.txt
6.step: rm hash.txt

6.DATA AND ANALYSIS



Cracking Method	Time	
md5	0m0.122s	(128-bit)
md4	0m0.128s	(128-bit)
sha-1	0m0.143s	(160-bit)
sha-256	0m0.140s	(256-bit)
sha-512	0m0.198s	(512-bit)
zip-file	0m2.073s	

Figure-11

zip-file > sha-512 > sha-1 > sha-256 > md4 >md5

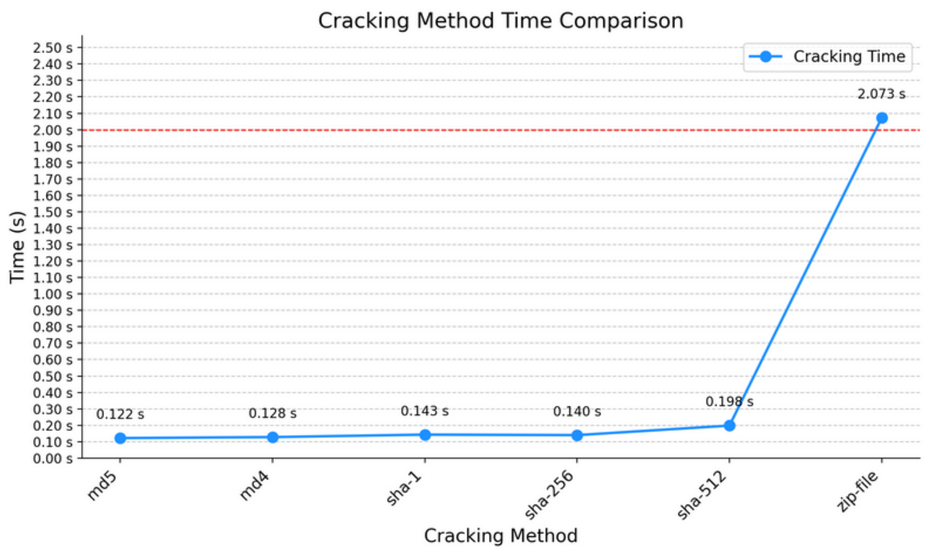


Figure-12

7. RESULT & CONCLUSION

Result:

The analysis of password cracking methods reveals insights into computational efficiency and security levels:

- **md5, md4, sha-1, sha-256, sha-512:**

  The varying processing times reflect the differing levels of security among these hash functions.

  Faster processing times (e.g., md5 and sha-1) indicate vulnerabilities, while slower times (e.g., sha-256 and sha-512) suggest stronger resistance against brute-force attacks.

- **zip-file:**

Cracking a password-protected zip file proved to be a more time-consuming task compared to individual hash functions.

The extended processing time highlights the effectiveness of encryption methods employed in compressed files, making them more resilient to password cracking attempts.

Conclusion:

1. **Algorithm Selection Matters:**
   - The choice of hash algorithm significantly influences the security of password storage.
   - Stronger algorithms (sha-256 and sha-512) contribute to enhanced resistance against unauthorized access.
2. **Trade-off Between Speed and Security:**
   - Faster algorithms may sacrifice security, as seen with md5 and sha-1, which are now considered insecure for cryptographic purposes.
   - Slower algorithms, such as sha-256 and sha-512, offer heightened security but may require more computational resources.
3. **Password-Protected Archives:**
   - Password protection in compressed files adds an additional layer of security, increasing the time and effort required for successful cracking.
4. **Recommendations:**
   - Emphasize the use of modern, secure hash functions like sha-256 or sha-512 for password storage.
   - Encourage users to adopt complex passwords to enhance overall system security.

8.REFERENCE

- john --help
- https://www.openwall.com/john/
- https://github.com/openwall/john $\longrightarrow$ Wordlist file download link