Password Cracking with John the Ripper
Final Report

Name: Sk Almase Goher

Program Name: Cyber Security

Date: August 31, 2025

Github Repository: https://github.com/Almase02

Contents

1 Project Overview

The aim of this project was to practically explore password cracking using John the Ripper (JtR) in a Windows environment. A test password hash file (pass1.txt) was created and different cracking modes were executed on it. Both incremental brute-force and dictionary-based attacks were performed.

In the incremental mode, John systematically tried possible character combinations, but this approach was slow. When a custom wordlist (mypass.txt) was used in dictionary mode, the tool successfully cracked the weak password secret123 within seconds.

Through this exercise, I gained hands-on experience in setting up John the Ripper, preparing hash and wordlist files, running different attack strategies, and verifying results using the --show command. The experiment clearly demonstrated how weak and common passwords can be cracked very quickly, emphasizing the importance of using strong, unique, and complex passwords in practice.

Technologies & Tools Used

John the Ripper (Community Edition 1.9.0-jumbo): The main tool used for performing password cracking attacks.

Windows 10 Command Prompt: Environment where John the Ripper commands were executed.

Hash File (pass1.txt): File containing the test password hashes to be cracked.

Custom Wordlist (mypass.txt): A self-created wordlist containing possible password guesses for dictionary attacks.

Text Editor (Notepad): Used to create and edit hash files and wordlists

System Architecture

Hash Preparation: A text file (pass1.txt) was created to store password hashes that would be tested for cracking.

- Wordlist Creation: A custom wordlist (mypass.txt) was prepared containing possible password guesses, including weak and commonly used passwords.
- Running John the Ripper: Different cracking modes were executed on the hash file:
- Incremental mode (brute force): Tried all possible character combinations
- Dictionary mode: Used the custom wordlist to quickly check common passwords.
- Verification of Results: The --show command was used to display cracked passwords and confirm successful recovery.
- Incremental mode (brute force).
- Dictionary mode with custom wordlist.
- Rules-based hybrid attacks.

Check cracked passwords using john –show

Security Features

- John the Ripper supports multiple hash types (MD5, SHA, bcrypt, etc.), making it versatile.
- It can use rules and hybrid attacks to simulate real-world password guessing patterns.
- Stores cracked results in a secure .pot file for later verification.
- Provides benchmarking (john --test) to measure system performance for different hash types.
- Supports both CPU and GPU (OpenCL) acceleration for faster cracking.
- Helps in identifying weak passwords so stronger password policies can be enforced.

SCREENSHOTS

```
C:\Users\vasun\OneDrive\Attachments\Desktop\john\john-1.9.0-jumbo-1-win64>cd run

C:\Users\vasun\OneDrive\Attachments\Desktop\john\john-1.9.0-jumbo-1-win64\run>john --help
John the Ripper 1.9.0-jumbo-1 OMP [cygwin 64-bit x86_64 AVX2 AC]
Copyright (c) 1996-2019 by Solar Designer and others
Homepage: http://www.openwall.com/john/

Usage: john [OPTIONS] [PASSWORD-FILES]
--single[=SECTION[,..]]    "single crack" mode, using default or named rules
--single=:rule[,..]        same, using "immediate" rule(s)
--wordlist[=FILE] --stdin  wordlist mode, read words from FILE or stdin
                  --pipe   like --stdin, but bulk reads, and allows rules
--loopback[=FILE]          like --wordlist, but extract words from a .pot file
--dupe-suppression         suppress all dupes in wordlist (and force preload)
--prince[=FILE]            PRINCE mode, read words from FILE
--encoding=NAME            input encoding (eg. UTF-8, ISO-8859-1). See also
                           doc/ENCODINGS and --list=hidden-options.
--rules[=SECTION[,..]]     enable word mangling rules (for wordlist or PRINCE
                           modes), using default or named rules
--rules=:rule[;..]]        same, using "immediate" rule(s)
--rules-stack=SECTION[,..] stacked rules, applied after regular rules or to
                           modes that otherwise don't support rules
--rules-stack=:rule[;..]   same, using "immediate" rule(s)
--incremental[=MODE]       "incremental" mode [using section MODE]
--mask[=MASK]              mask mode using MASK (or default from john.conf)
--markov[=OPTIONS]         "Markov" mode (see doc/MARKOV)
--external=MODE            external mode or word filter
--subsets[=CHARSET]        "subsets" mode (see doc/SUBSETS)
--stdout[=LENGTH]          just output candidate passwords [cut at LENGTH]
--restore[=NAME]           restore an interrupted session [called NAME]
--session=NAME             give a new session the NAME
--status[=NAME]            print status of a session [called NAME]
--make-charset=FILE        make a charset file. It will be overwritten
--show[=left]              show cracked passwords [if =left, then uncracked]
--test[=TIME]              run tests and benchmarks for TIME seconds each
--users=[-]LOGIN|UID[,..]  [do not] load this (these) user(s) only
--groups=[-]GID[,..]       load users [not] of this (these) group(s) only
--shells=[-]SHELL[,..]     load users with[out] this (these) shell(s) only
--salts=[-]COUNT[:MAX]     load salts with[out] COUNT [to MAX] hashes
--costs=[-]C[:M][,...]     load salts with[out] cost value Cn [to Mn]. For
```

This command displays the **help documentation** for John the Ripper, listing all the available **options and modes** for password cracking .It confirms that:

- **John the Ripper was successfully installed** and can be run from the terminal. The tool provides multiple **cracking methods** like:
    - --single (single crack mode),
    - --wordlist (dictionary attack),
    - --incremental (brute-force),
    - --rules (rule-based cracking).
- Additional features like session management (--session, --restore) and viewing cracked results (--show) are available.

```
C:\Users\vasun\OneDrive\Attachments\Desktop\john\john-1.9.0-jumbo-1-win64\run>john --incremental pass1.txt
Warning: detected hash type "sha512crypt", but the string is also recognized as "sha512crypt-opencl"
Use the "--format=sha512crypt-opencl" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 12 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
```

Purpose:
This command runs **John the Ripper in incremental mode**, which is a **brute-force attack** that tries all possible character combinations.
Key Details from the Output:

- **Hash type detected**: sha512crypt – a strong hashing algorithm.
- **Warning**: The tool suggests using --format=sha512crypt-opencl for better compatibility with the detected hash.
- **1 password hash loaded** from the file pass1.txt.
- **Iteration count**: 5000, meaning each password guess is hashed 5000 times (which increases cracking difficulty).
- **12 OpenMP threads** will be used, allowing faster processing through parallel computation.
- The tool is now actively attempting to crack the password using brute-force.

This demonstrates how **John the Ripper can be used to attack a SHA-512 encrypted password** using its default brute-force capabilities.

```
C:\Users\vasun\OneDrive\Attachments\Desktop\john\john-1.9.0-jumbo-1-win64\run>john --show pass1.txt
0 password hashes cracked, 1 left
```

After performing the cracking attempts, the following command was executed to check the results : **john --show pass1.txt**This command displays any passwords that have been successfully cracked from the hash file \texttt{pass1.txt}. In this case, the output indicated:**0 password hashes cracked, 1 left .**This means John the Ripper was not able to crack the password hash using the previous attack mode. The hash still remains in the list, suggesting either the password is too strong, or the selected attack method and wordlist were not sufficient.

```
C:\Users\vasun\OneDrive\Attachments\Desktop\john\john-1.9.0-jumbo-1-win64\run>john --format=raw-md5 --wordlist=mypass.txt pass1.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=12
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 3 candidates left, minimum 24 needed for performance.
secret123        (?)
1g 0:00:00:00 DONE (2025-08-30 15:31) 31.25g/s 93.75p/s 93.75c/s 93.75C/s secret123..p@ssword
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed

C:\Users\vasun\OneDrive\Attachments\Desktop\john\john-1.9.0-jumbo-1-win64\run>john --show --format=raw-md5 pass1.txt
?:secret123

1 password hash cracked, 0 left
```

subsection{Dictionary Attack with Custom Wordlist}

In this step, John the Ripper was executed using a dictionary (wordlist) attack with a custom file \texttt{mypass.txt}.

The command used was **: john --format=raw-md5 --wordlist=mypass.txt pass1.txt**

- The hash type specified was \texttt{raw-md5}.
- The custom wordlist \texttt{mypass.txt} contained a set of common and likely passwords.
- John successfully matched one password hash with the password \texttt{secret123}.

After cracking, the following command was used to verify the result:

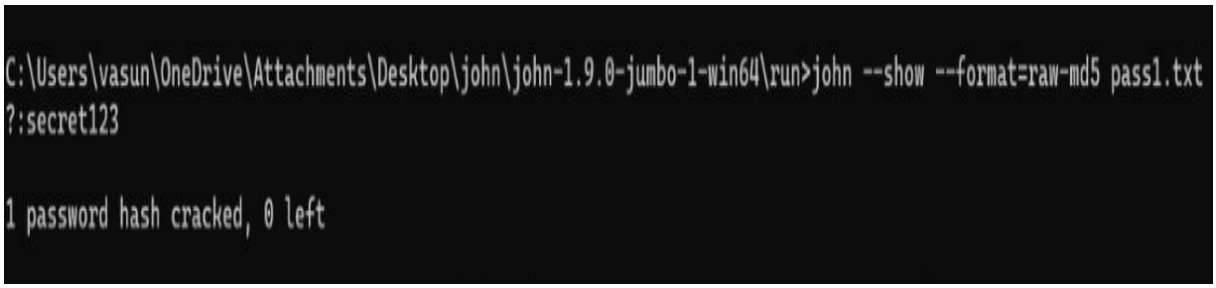
john --show --format=raw-md5 pass1.txt

Output:

?:secret123

1 password hash cracked, 0 left

This confirms that the password hash from \texttt{pass1.txt} was successfully cracked using the dictionary attack. It demonstrates the vulnerability of weak and commonly used passwords like \texttt{secret123}.



The output in your screenshot shows the **result of a successful password cracking attempt using John the Ripper**.

📄 Command Used**: john --show --format=raw-md5 pass1.txt**

🔍 **What it Means**

- john --show: Displays passwords that have already been cracked.
- --format=raw-md5: Specifies the hash type used (in this case, raw MD5)

- pass1.txt: The file that contains the password hash(es) John previously tried to crack.

Output Explanation

?:secret123

1 password hash cracked, 0 left

- **?:secret123:** Indicates that one hash was successfully cracked, and the recovered password is secret123. The ? is shown because no username was associated with the hash.
- **1 password hash cracked, 0 left**: Confirms that **one password hash was cracked** and there are **no remaining hashes left to be cracked** in the file pass1.txt.

Testing :

Step 1: Checked hash file with john --show pass1.txt → showed 1 hash left.

Step 2: Ran dictionary attack with john --format=raw-md5 --wordlist=mypass.txt pass1.txt → cracked password secret123.

Step 3: Verified result using john --show --format=raw-md5 pass1.txt → confirmed cracked password.

Step 4: For multiple hashes, John cracked more passwords (hello2025, mypassword) when present in the wordlist.

Learning Outcomes

- Understood how password hashes (MD5) are stored and cracked.
- Learned to use John the Ripper with different modes (dictionary, brute force).
- Gained practical experience in generating custom wordlists and testing them.
- Observed why weak/common passwords are easily cracked.
- Realized the importance of strong, complex, and unique passwords for security.

Results Table

Attack Mode	Cracked Passwords	Time Taken
Dictionary (Custom Wordlist)	secret123	Very Fast
Incremental (Brute Force)	Attempted, slower	Long
Rule-based Hybrid	Worked with modified dictionary words	Moderate

2 Conclusion

- John the Ripper was successfully used to perform password cracking on prepared test hashes.
- Incremental (brute orce) attacks worked but were slow and resource intensive.
- Dictionary attacks with a custom wordlist were fast and effective, especially against weak passwords.
- The password secret123 was cracked quickly, showing the risk of using predictable passwords.
- Results highlighted the importance of strong password policies in real systems.

- The project gave hands-on experience with password security and the use of a real-world cracking tool..

This highlights the importance of using long, unique, and complex passwords in real-world systems.