

# Computer Security Project Report

**CSE 406**

**Tool: John The Ripper**

**Supervisor:**

Abdur Rashid Tushar  
Lecturer  
CSE, BUET

**Presented By:**

Rayan Islam - 1905106  
Fahad Ahmed Akash - 1905107

Level-4 Term-1

Department of CSE

Bangladesh University of Engineering & Technology

11 March, 2024

# Contents

1	Introduction	2
2	Tool Overview	2
3	Installation	3
4	Working Procedure	4
5	Feature Description	4
6	Feature Documentation	7
7	Failures	13
8	Optimization:	14

# 1 Introduction

John the Ripper is a password cracking tool. The tool is used by security professionals to test the strength of passwords. It is very powerful and versatile. It employs various techniques such as brute-force attacks, dictionary attacks, and hybrid attacks to uncover weak passwords in different environments. Originally developed for Unix systems, it has since been ported to numerous platforms, making it widely accessible for security testing purposes.

# 2 Tool Overview

John the ripper is an open source password cracking tool. It is widely used for analysis of the strength of passwords. Here is an overview of John the ripper's key features and functionalities:

1. **Multiple Hash Formats:** It supports numerous hash types, allowing it to crack passwords stored in various formats -

- Windows LM hashes
- MD5
- SHA-1 etc.

Moreover, it uses a modular architecture, enabling users to add new hash types or cracking methods easily through plugins.

2. **Multiple Attacks:** John the ripper allows the user to work in multiple modes, such as -

- (a) **Dictionary Attack**

- **Wordlist Mode:** This mode allows John to use a list of potential passwords (often derived from real words or common password choices) to attempt to crack the hash.
- **Brute Force and Incremental Mode:** It can perform brute-force attacks, trying all possible combinations of characters up to a certain length, and incremental attacks, which systematically cover all possible plaintexts.

- (b) **Rules-based Attack:** This allows users to apply various modifications to wordlist entries (like capitalization, character replacement, or affixing numbers) to crack passwords that involve common substitutions or simple variations on dictionary words.
- 3. **Parallel Processing:** It can take advantage of multiple CPUs and multi-core CPUs, along with certain GPUs, to increase cracking speed.
- 4. **Community-enhanced Versions:** There are community-enhanced versions like John the Ripper Jumbo, which include additional features and support for many more hash types and cracking methods not found in the standard version.
- 5. **Portability:** It runs on a variety of operating systems, including -
  - Unix
  - Windows
  - DOS
  - BeOS
  - OpenVMS

### 3 Installation

To install John the Ripper from its GitHub repository, follow these steps -

- **Clone the repository:**  
git clone <https://github.com/magnumripper/JohnTheRipper.git>
- **Navigate to the JohnTheRipper directory:**  
Open the terminal -
  - *cd JohnTheRipper/src*
  - *./configure*
  - *make*
  - *make install*
  - *cd ../run*

## 4 Working Procedure

John the Ripper operates by comparing hashed passwords against possible plaintext inputs to find matches. Here's a general outline of how it works -

- **Obtain Hashes:**

Initially, John the Ripper needs to obtain the hashed versions of passwords. These hashes are typically stored in the system's password database or security files, such as */etc/shadow* on Unix systems. The software supports different formats and can extract these hashes from various sources.

- **Select Mode:**

- Single Mode
- Wordlist Mode
- Brute-force Mode
- Rule-Based Mode

- **Cracking Process:** John the Ripper takes each input from the selected mode, applies the relevant hash function, and then compares the result with the target hash(es). If a match is found, it means the original plaintext password has been successfully identified.

## 5 Feature Description

John can be configured to operate in three different modes.

1. **Wordlist Mode:**

- **Description:** The wordlist method, also known as a dictionary attack, involves using a pre-defined list of words, phrases, or common passwords (referred to as a wordlist or dictionary) to attempt to crack passwords. The attacker systematically tries each word or phrase in the wordlist, hashing it and comparing it against the hashed passwords to see if there's a match.

- **Usage:** This method is effective when passwords are commonly used words, phrases, or variations of them. It's also useful when attackers have some knowledge about the target, such as the language or interests of the users.
- **Example:** An attacker might use a wordlist containing common passwords, words from literature or popular culture, or commonly used phrases in the target language.

## 2. Rule-Based Mode:

- **Description:** The rule-based method involves applying a set of rules or transformations to words or characters in the wordlist to generate variations of them and attempt to crack passwords. These rules can include modifications such as adding prefixes or suffixes, substituting characters with similar-looking ones, or applying common patterns.
- **Usage:** Rule-based attacks are useful for generating a larger set of password variations from a relatively small wordlist, increasing the chances of finding a match. They can also help account for common password patterns or behaviors, such as appending numbers or special characters.
- **Example:** An attacker might use rules to apply common modifications like adding numbers or symbols to the end of words, capitalizing letters, or substituting certain characters with similar-looking ones (e.g., replacing 'o' with '0' or 'i' with '1').

## 3. Brute-Force Mode:

- **Description:** The brute-force method involves systematically trying every possible combination of characters until the correct password is found. This approach is the most exhaustive and time-consuming, as it doesn't rely on any pre-defined wordlist or patterns. Brute-force attacks can be further categorized into different types based on the character sets and lengths being considered.
- **Usage:** Brute-force attacks are typically used as a last resort when other methods fail, or when the password complexity is unknown or believed to be very high. They can also be targeted towards specific character sets, such as lowercase letters, uppercase

letters, numbers, or special symbols, depending on the suspected password requirements.

- **Example:** An attacker might start with a brute-force attack using a specific character set (e.g., lowercase letters only) and then progressively expand to include other character sets and increase the password length until the correct password is found.

Each of these methods has its strengths and weaknesses, and attackers often use a combination of them to maximize their chances of success in cracking passwords. Wordlist Mode is Fast but there is less probability of finding the correct password. Rule-Base Mode is effective if and only if user can predict how the password was created. The Brute-force method is slow, but after quiet a few time it might crack the password.

## 6 Feature Documentation

### 1. Single Crack Mode:

- First you have to select a password protected file to crack the password. Let us assume we want to crack a zip file.
- Create a zip file and make it password protected.

```
> touch securityTools.txt
> cat > securityTools.txt
This is the testing of tools
^C
> cat securityTools.txt
This is the testing of tools

> zip -e -r securedTools.zip securityTools.txt
[Enter password:
[Verify password:
  adding: securityTools.txt (stored 0%)
```

- Create a hashfile of password  
./zip2john <zip\_file> <hash\_file>

```
./zip2john securedTools.zip > hack.hashes
```

- Crack the file in single mode  
./john --single <hash\_file>

```
./john hack.hashes
```

- Cracking Process



```

Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
0g 0:00:00:00 DONE 1/3 (2024-03-06 15:47) 0g/s 3015Kp/s 3015Kc/s 3015Kc/s Txtzip1900..Tsecuritytools1900
Proceeding with wordlist:./password.lst
Enabling duplicate candidate password suppressor
security123      (securedTools.zip/securityTools.txt)
1g 0:00:00:00 DONE 2/3 (2024-03-06 15:47) 9.091g/s 1581Kp/s 1581Kc/s 1581Kc/s liberia1..ballin9
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

```

- Show the password

```

> ./john --show hack.hashes
securedTools.zip/securityTools.txt:security123:securityTools.txt:securedTools.zip::securedTools.zip
1 password hash cracked, 0 left

```

```

> cat john.pot
$pkzip$1*2*2*0*19*d*9fff9164*0*3d*0*19*794f*f
$pkzip$1*2*2*0*29*1d*c16e2875*0*4b*0*29*7da1*
pkzip$:security123

```

- Unzip and access the file using the password you get  
`./john --single <hash_file>`

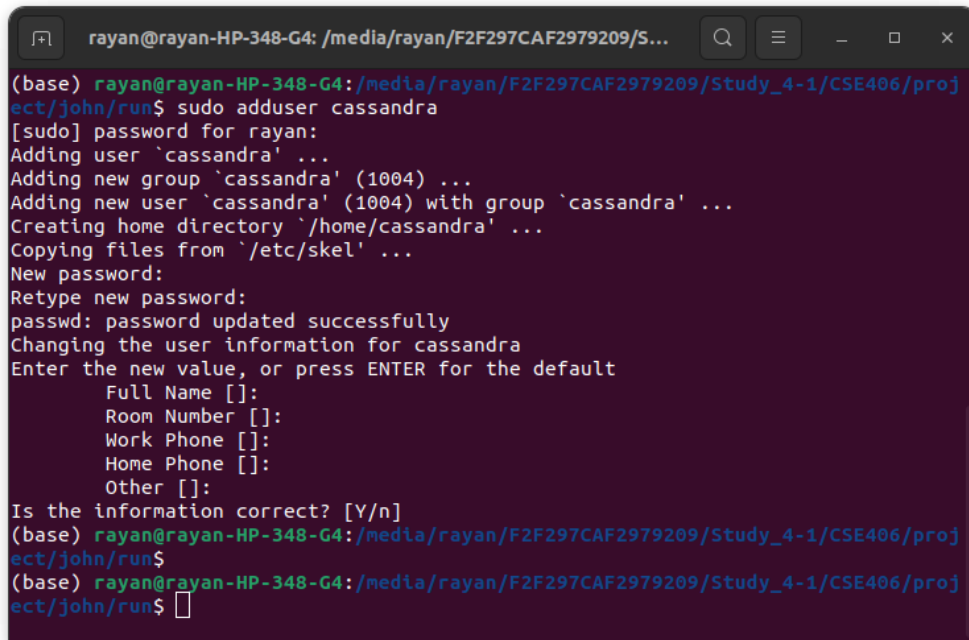
```

> ./john --single hacked.hashes
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for status
akash123      (akash1.zip/akash1.txt)
1g 0:00:00:00 DONE (2024-03-06 15:55) 100.0g/s 94400p/s 94400c/s 94400c/s Takash12..akashakash.zip/akash.txt123
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

```

## 2. Dictionary Attack(Wordlist method):

- Create a temporary user to crack his password  
*sudo adduser cassandra*

A terminal window with a dark purple background. The title bar shows the user 'rayan' on a machine named 'rayan-HP-348-G4' at the path '/media/rayan/F2F297CAF2979209/S...'. The terminal text shows the command 'sudo adduser cassandra' being executed. It prompts for a password, adds the user and group 'cassandra' (1004), creates a home directory at '/home/cassandra', and copies files from '/etc/skel'. It then prompts for a new password and its retype. After confirming the password, it prompts for user information: Full Name, Room Number, Work Phone, Home Phone, and Other. The user enters 'cassandra' for the full name. It then asks if the information is correct. The terminal ends with the prompt 'ect/john/run\$'.

```
(base) rayan@rayan-HP-348-G4:/media/rayan/F2F297CAF2979209/Study_4-1/CSE406/project/john/run$ sudo adduser cassandra
[sudo] password for rayan:
Adding user `cassandra' ...
Adding new group `cassandra' (1004) ...
Adding new user `cassandra' (1004) with group `cassandra' ...
Creating home directory `/home/cassandra' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for cassandra
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n]
(base) rayan@rayan-HP-348-G4:/media/rayan/F2F297CAF2979209/Study_4-1/CSE406/project/john/run$
(base) rayan@rayan-HP-348-G4:/media/rayan/F2F297CAF2979209/Study_4-1/CSE406/project/john/run$
```

- Shadow folder has the hashes of passwords. The only access is of root. We have to grep */etc/shadow* folder to *newpass*.

```
rayan@rayan-HP-348-G4: /media/ryan/F2F297CAF2979209/S...
(base) rayan@rayan-HP-348-G4:/media/ryan/F2F297CAF2979209/Study_4-1/CSE406/project/john/run$ sudo cat /etc/shadow |grep cassandra>newpass
(base) rayan@rayan-HP-348-G4:/media/ryan/F2F297CAF2979209/Study_4-1/CSE406/project/john/run$ cat newpass
cassandra:$y$j9T$mo4X09xLDHuyLVHZsIWY/0$QeKgbGq2T3NVQe94o2ixgj33afjVQ1bSFzMoG8xU
QC.:19788:0:99999:7:::
(base) rayan@rayan-HP-348-G4:/media/ryan/F2F297CAF2979209/Study_4-1/CSE406/project/john/run$
```

- Default wordlist attack `./john -wordlist=pospass newpass`

```
./john --wordlist=pospass newpass
```

### 3. Dictionary Attack(Manual wordlist method):

- Create a your own wordlist

```
(base) rayan@rayan-HP-348-G4:/media/ryan/F2F297CAF2979209/Study_4-1/CSE406/project/john/run$ cat pospass
akib
cassandra
Mrakib
Akash
batash
guitar
tommy
sugra
```

- Attack with manual wordlist `./john -wordlist=pospass newpass`

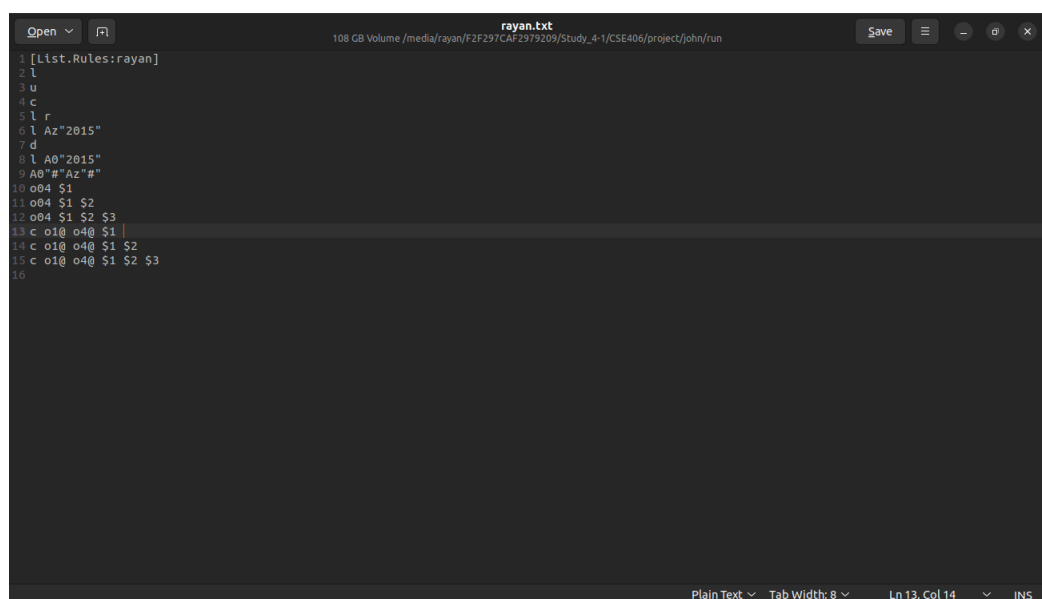
```
(base) rayan@rayan-HP-348-G4:/media/ryan/F2F297CAF2979209/Study_4-1/CSE406/project/john/run$ ./john --wordlist=pospass newpass
```

- Cracking Process

```
Warning! john.conf section [list.rules:rayan] is multiple declared.
Using default input encoding: UTF-8
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Cracked 1 password hash (is in ./john.pot), use "--show"
No password hashes left to crack (see FAQ)
(base) rayan@rayan-HP-348-G4: /media/ryan/F2F297CAF2979209/Study_4-1/CSE406/project/john/run$ sudo cat /etc/shadow|grep akash > newpass
(base) rayan@rayan-HP-348-G4: /media/ryan/F2F297CAF2979209/Study_4-1/CSE406/project/john/run$ ./john --wordlist=pospass newpass
Warning! john.conf section [list.rules:rayan] is multiple declared.
Using default input encoding: UTF-8
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Cost 1 (algorithm [0:unknown 1:descrypt 2:md5crypt 3:sunmd5 4:bcrypt 5:sha256crypt 6:sha512crypt 7:scrypt 10:yescrypt 11:gost-yescrypt]) is 10
for all loaded hashes
Cost 2 (algorithm specific iterations) is 1 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for status
Warning: Only 8 candidates buffered, minimum 96 needed for performance.
guitar
      (akash)
1g 0:00:00:02 DONE (2024-03-11 15:28) 0.3401g/s 2.721p/s 2.721c/s 2.721c/s akib..sugra
```

## 4. Dictionary Attack(Rule-Based method):

- Create some Rules



```
1 [List.Rules:rayan]
2 l
3 u
4 c
5 l r
6 l Az"2015"
7 d
8 l A0"2015"
9 A0"#Az"#
10 o04 $1
11 o04 $1 $2
12 o04 $1 $2 $3
13 c o1@ o4@ $1 |
14 c o1@ o4@ $1 $2
15 c o1@ o4@ $1 $2 $3
16
```

- Show the rules *`./john --wordlist=pospass --stdout --rules:rayan`*

```
(base) rayan@rayan-HP-348-G4: /media/ryan/F2F297CAF2979209/Study_4-1/CSE406/project/john/run$ ./john --wordlist=pospass --stdout --rules:rayan
Warning! john.conf section [list.rules:rayan] is multiple declared.
Using default input encoding: UTF-8
Enabling duplicate candidate password suppressor
akib
cassandra
hrakib
akash
batash
guitar
tommy
sugra
AKIB
CASSANDRA
HRAKIB
AKASH
```

```
rayan@rayan-HP-348-G4: /media/ryan/F2F297CAF2979209/Study_4-1/CSE406/project/john/run
4kash123
4atash123
4ultar123
4k@b1
4a@sandra1
4r@k1b1
4k@sh1
4a@ash1
4u@tar1
4k@b12
4a@sandra12
4r@k1b12
4k@sh12
4a@ash12
4u@tar12
4k@b123
4a@sandra123
4r@k1b123
4k@sh123
4a@ash123
4u@tar123
A@1b1
C@s@s@ndra1
M@ak@b1
A@as@1
B@ta@h1
G@1t@r1
A@1b12
C@s@s@ndra12
M@ak@b12
A@as@12
B@ta@h12
G@1t@r12
A@1b123
C@s@s@ndra123
M@ak@b123
A@as@123
B@ta@h123
```

- Rule-Based Attack `./john --wordlist=pospass --rules:rayan`

```
./john --wordlist=pospass --rule:rayan newpass
```

- Cracking Process

```
rayan@rayan-HP-348-G4: /media/ryan/F2F297CAF29...
G@it@r123
(base) rayan@rayan-HP-348-G4:/media/ryan/F2F297CAF2979209/Study_4-1/CSE406/project/john/run$ ./john --wordlist=pospass --rule:rayan newpass
Using default input encoding: UTF-8
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Cost 1 (algorithm [0:unknown 1:descript 2:md5crypt 3:sunmd5 4:bcrypt 5:sha256crypt 6:sha512crypt 7:scrypt 10:yescrypt 11:gost-yescrypt]) is 10 for all loaded hashes
Cost 2 (algorithm specific iterations) is 1 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for status
Enabling duplicate candidate password suppressor
C@ss@ndra123 (cassandra)
1g 0:00:00:01 DONE (2024-03-06 15:41) 0.6757g/s 68.92p/s 68.92c/s 68.92C/s
A@ib123..G@it@r123
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
(base) rayan@rayan-HP-348-G4:/media/ryan/F2F297CAF2979209/Study_4-1/CSE406/project/john/run$
```

## 7 Failures

Despite its effectiveness, John the Ripper may fail to crack passwords under certain circumstances, such as:

- **Strong Passwords:** Passwords that are sufficiently complex and long may remain uncrackable by John the Ripper, especially if they are not included in any dictionary or have random character sequences.
- **Salted Hashes:** If the passwords are stored using cryptographic hashing algorithms with salts, it significantly increases the difficulty of cracking them, and John the Ripper may not be successful without additional resources or time.

John the Ripper is a valuable tool in the arsenal of security professionals for identifying weak passwords and enhancing the overall security posture of systems and networks. However, it's essential to understand its capabilities, limitations, and proper usage to derive maximum benefit from its features.

## 8 Optimization:

John the ripper can be optimized for better performance -

1. Parallel computation (multiple CPU) 2. Optimised hash (format) 3. Optimised algorithm (adding rules)

- **Parallel computation:** It can distribute the workload across multiple processors, cores, or GPUs to increase speed.
- **Optimized Hash Functions:** If we can guess what hash function is used in password, we can select respective mode for efficient comparison and performance.
- **Efficient Algorithms:** For brute-force and incremental modes, it employs efficient algorithms to reduce unnecessary computations.