# Information Security IA-1: Implementation of a security tool

# Title: John The Ripper

## Group Members Details:


- 16010122283 – Hitanshi Patil
- 16010122284 – Harikrishnan Gopal
- 16010122288 – Aditya Raut
- 16010122294 – Adhiraj Patole


## Introduction:

John the Ripper (JtR) is an open-source password auditing and recovery tool designed to test the strength of passwords by decrypting hash values into plaintext. Initially developed for Unix-based systems, it has expanded to support multiple platforms, including Linux, Windows, and macOS. Cybersecurity professionals, ethical hackers, and penetration testers widely use JtR to identify weak passwords, assess security policies, and improve authentication mechanisms.

John the Ripper is highly versatile, supporting a wide range of password hashing algorithms, including MD5, SHA-1, SHA-256, SHA-512, crypt(), bcrypt, LM, NTLM, and many more. The tool automatically detects the hash format, streamlining the password recovery process. To crack passwords, it employs various techniques such as brute force attacks, dictionary attacks, rule-based modifications, hybrid attacks, and incremental mode. This flexibility allows it to effectively test different types of password security implementations.

A key strength of JtR is its speed and efficiency, utilizing optimized algorithms and hardware acceleration for faster password cracking. The tool supports multithreading and multiprocessing (fork implementation) to distribute workload across multiple CPU cores, significantly improving performance. Additionally, it can leverage GPU acceleration (via OpenCL and CUDA) for large-scale attacks, making it highly effective in password recovery operations. John the Ripper also supports graphical user interfaces (GUI), distributed cracking over networks, session management for pausing and resuming attacks, and integration with third-party plugins and wordlists for enhanced functionality.

## Features:

1. Automatic Hash Format Detection – Identifies and determines the encryption type used.
2. Support for Multiple Hash Types – Cracks a variety of hashes, including MD5, SHA-1, SHA-256, LM, NTLM, bcrypt, and more.
3. Single Crack Mode – Optimized for cracking passwords based on known username-password correlations.
4. Wordlist Mode – Uses a predefined dictionary and applies mutations to improve success rates.
5. Incremental Mode – Exhaustively generates and tests all possible character combinations.
6. Customizable Rule-Based Attacks – Allows defining complex password attack strategies with rule-based modifications.
7. Password Cracking for Compressed Files – Supports cracking ZIP, RAR, and other encrypted archive formats.
8. Graphical User Interface (GUI) Support – Provides an accessible alternative for users unfamiliar with CLI.
9. Multiprocessing with Fork Implementation – Distributes tasks across multiple CPU cores for enhanced performance.
10. GPU Acceleration – Can leverage OpenCL and CUDA for hardware-accelerated password cracking on GPUs.
11. Logging and Reporting – Saves cracked passwords and attack statistics for analysis.

## Methodology:

The process starts with Hash Collection, where password hashes are gathered from various sources: Linux systems, Windows systems, or from database dumps using commands like strings and grep.

Following this, Hash Type Identification is carried out using tools such as hash-identifier to recognize hash types like MD5, SHA-1, SHA-256, SHA-512, crypt(), or bcrypt.

Once the hash types are identified, Preparation is done to ensure compatibility with John the Ripper. UNIX hashes are formatted with unshadow, Windows hashes are extracted using samdump2, and specific formatting is applied to each hash type. After preparation, Cracking Techniques are utilized.

A Dictionary Attack is attempted first, using common password lists like rockyou.txt to test various combinations, while a Rule-Based Attack modifies the dictionary entries. Once passwords are successfully cracked, Password Recovery is conducted using (john --show hashfile.txt).

Finally, Analysis and Reporting take place, which involves reviewing the results, pinpointing weak passwords, and recommending security enhancements such as implementing strong password policies, enabling multi-factor authentication, and improving storage methods. This organized approach in Kali Linux enables security professionals to effectively evaluate and enhance password security.

## Implementation:

The Python script demonstrates a basic integration of password cracking methods similar to those used by John the Ripper.

In the Python script, the md5_hash(password) function generates an MD5 hash from a given string, while load_hash_file(filename) reads user-hash pairs and stores them in a dictionary. The load_wordlist(filename) function imports words from a file for use in password attacks.

Three primary cracking modes are implemented:

single_crack(user_hashes),

wordlist_crack(user_hashes, wordlist),

incremental_crack(user_hashes, wordlist), which uses generate_case_permutations(word).

Finally, the main() function provides a menu-based interface for selecting any of the three modes or exiting the program, demonstrating a streamlined approach to basic password-cracking techniques.

Single Crack Mode

Automatically generates possible password candidates from each username (e.g., reversed, uppercase, capitalized).

Compares the MD5 hash of each candidate with the stored hash.

Wordlist Mode

Reads a predefined wordlist (e.g., rockyou.txt).

For each word, tests lowercase, uppercase, and capitalized variations against the stored hashes.

Incremental Mode

Generates all possible case permutations of each word in the wordlist (e.g., “Pass”, “pAss”, “paSs”, etc.).

Tests each variant’s MD5 hash against the stored hashes, mimicking an expanded brute-force approach.

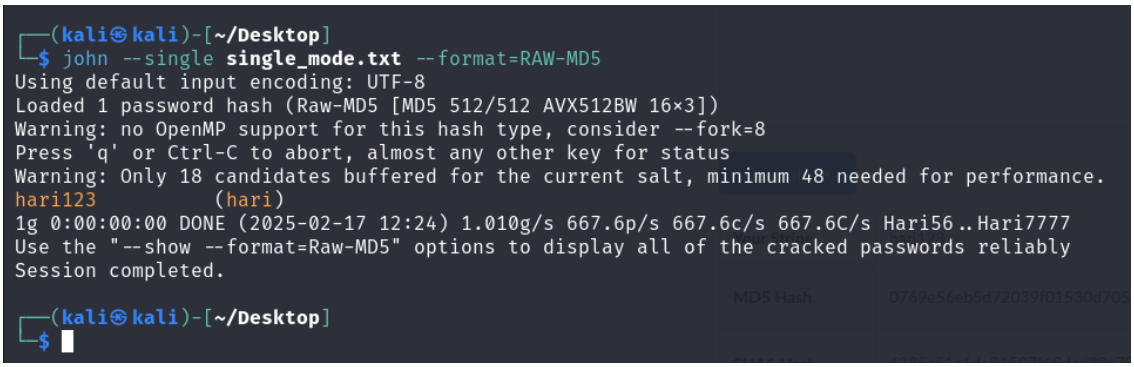
Users are prompted to select the attack mode, and the script attempts to recover passwords for each username in the hash file. This modular design (using separate functions for each mode) illustrates how password-cracking techniques can be programmatically automated and customized using Python.

## Results:

### Single Crack Mode

Overview: Leveraged JtR's single crack mode to exploit known correlations between user and password fields.

Outcome:




### Wordlist Mode

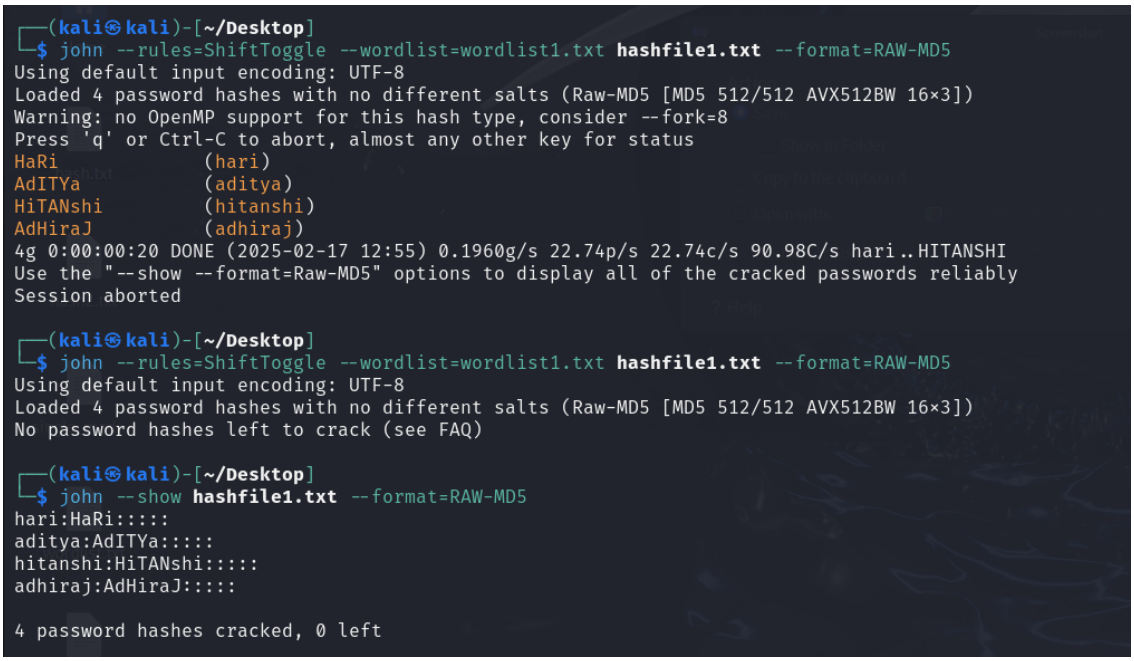
Overview: Tested a precompiled wordlist to attempt matches against stored password hashes.

Outcome:

## Incremental Mode

Overview: Performed an exhaustive search by systematically generating all possible combinations of characters.

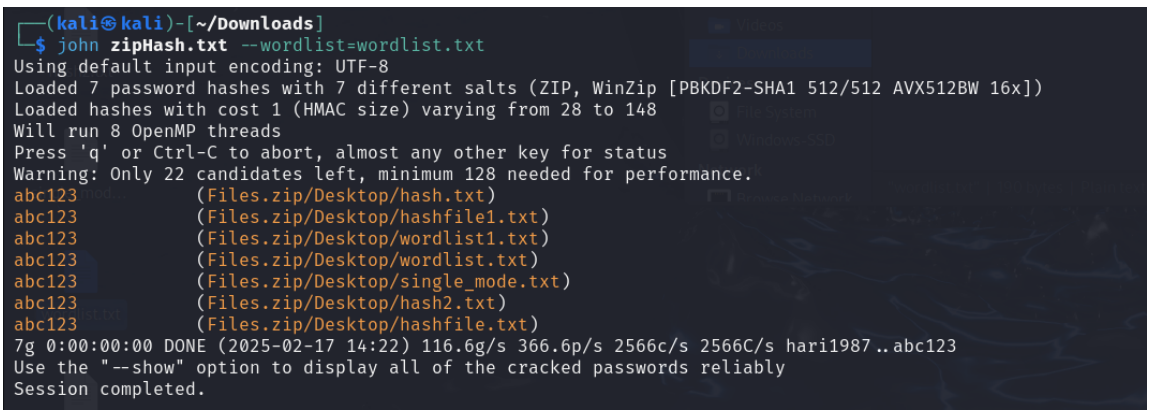
Outcome:




## Cracking a Password-Protected ZIP File

Overview: Applied John the Ripper's capabilities to extract and crack the ZIP file's password hash.

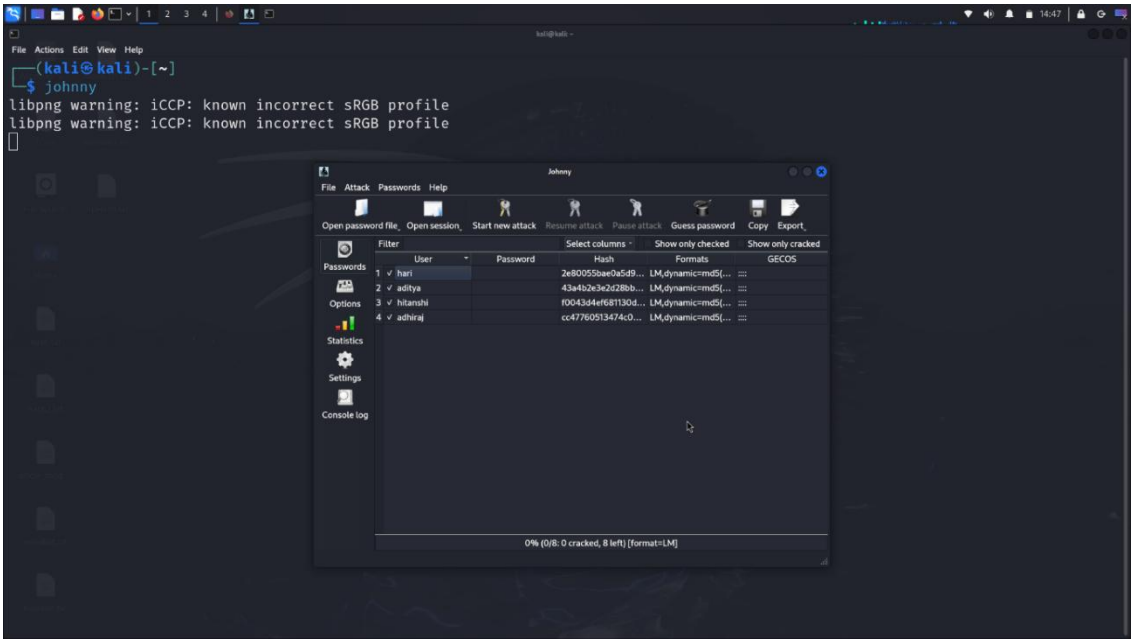
Outcome:



## GUI Demonstration

Overview: Explored John the Ripper's Graphical User Interface to facilitate password cracking without command-line operations.

Outcome:




## Multi-Processing Implementation

Overview: Enabled John the Ripper to run multiple processes in parallel using the fork option.

Outcome:

```
┌──(kali㉿kali)-[~/Desktop]
└─$ john --fork=4 --format=RAW-MD5 hashes.txt
Using default input encoding: UTF-8
Loaded 5 password hashes with no different salts (Raw-MD5 [MD5 512/512 AVX512BW 16×3])
Node numbers 1-4 of 4 (fork)
Proceeding with single, rules:Single
3: Warning: Only 33 candidates buffered for the current salt, minimum 48 needed for performance.
Press 'q' or Ctrl-C to abort, almost any other key for status
1: Warning: Only 31 candidates buffered for the current salt, minimum 48 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
Proceeding with incremental:ASCII
```

```
┌──(kali㉿kali)-[~]
└─$ ps aux |grep john
kali      138928 85.3  1.9 261560 152488 pts/1   RN+  15:42   0:20 john --fork=4 --format=RAW-MD5 hashes.txt
kali      138929 85.4  1.9 261560 149120 pts/1   RN+  15:42   0:20 john --fork=4 --format=RAW-MD5 hashes.txt
kali      138930 85.3  1.8 261560 140808 pts/1   RN+  15:42   0:20 john --fork=4 --format=RAW-MD5 hashes.txt
kali      138931 85.4  1.9 261560 149000 pts/1   RN+  15:42   0:20 john --fork=4 --format=RAW-MD5 hashes.txt
kali      139138  0.0  0.0   6528  2176 pts/0    S+   15:43   0:00 grep --color=auto john

┌──(kali㉿kali)-[~]
└─$
```

## Basic Python Implementation

Overview: Incorporated John the Ripper commands into a Python script to automate password cracking workflows.

Outcome:

```
PS C:\Users\DELL\Desktop\IS IA1> & C:/Users/DELL/AppData/Local/Programs/Python/Pytho
Enter the filename containing usernames and hashes: hash.txt
Enter the filename containing the wordlist: wordlist.txt

Select attack mode:
1. Single Crack Mode
2. Wordlist Mode
3. Incremental Mode
4. Exit
Enter choice (1/2/3/4): 1
Running Single Crack Mode...
[SUCCESS] Username: aditya, Password: aytida
[SUCCESS] Username: hitanshi, Password: hitanshi

Select attack mode:
1. Single Crack Mode
2. Wordlist Mode
3. Incremental Mode
4. Exit
Enter choice (1/2/3/4): 2
Running Wordlist Mode...
[SUCCESS] Username: hitanshi, Password: hitanshi

Select attack mode:
1. Single Crack Mode
2. Wordlist Mode
3. Incremental Mode
4. Exit
Enter choice (1/2/3/4): 3
Running Incremental Mode (All Case Permutations)...
[SUCCESS] Username: harikrishnan, Password: HAri
[SUCCESS] Username: hitanshi, Password: hitanshi

Select attack mode:
1. Single Crack Mode
2. Wordlist Mode
3. Incremental Mode
4. Exit
Enter choice (1/2/3/4): 4
Exiting...
PS C:\Users\DELL\Desktop\IS IA1>
```

## Conclusion:

John the Ripper is an effective and widely used password-cracking tool designed for security testing and password strength evaluation. We utilized its ability to detect hash formats, which made it easier to identify the type of encryption used.

We also applied several cracking modes, such as Single Crack Mode, Wordlist Mode, and Incremental Mode, to efficiently test passwords. The tool's ability to crack ZIP file passwords and use a Graphical User Interface (GUI) made it more user-friendly for those not familiar with command-line tools.

Additionally, the use of multiprocessing allowed us to speed up the process by distributing tasks across multiple CPU cores.

Our methodology included steps like hash collection, identification, preparation, and testing with various cracking techniques. This helped us uncover weak passwords that could be easily cracked with common dictionaries. Although John the Ripper worked well for weak passwords, it struggled with complex, strong passwords, which emphasizes the importance of using strong passwords, enabling multi-factor authentication, and employing better password storage practices.

In conclusion, John the Ripper is a powerful tool for ethical hacking and penetration testing. It highlights the importance of regularly assessing password security and adopting best practices to protect sensitive information from potential threats.