Password Cracking with John the Ripper

Khoi Nguyen Pham
Swinburne University of Technology
School of Science, Computing and Engineering Technologies
Hawthorn, Victoria 3122, Australia
Email: nguyenpham1441887@gmail.com

I. Introduction

The goal of this project is to demonstrate password cracking techniques using John the Ripper, a powerful open-source tool used for cracking password hashes. Understanding password security is critical in cybersecurity. This project will cover cracking simple password hashes in the MD5 crypt format using dictionary-based attacks and rule-based transformations, hence demonstrating the vulnerabilities of common passwords through various cracking techniques, emphasizing the importance of strong password practices.

II. Setup

A. Installing John the Ripper

```
sudo apt update
sudo apt install john
john --version
```

B. Preparing the Password Hashes

I created a file passwords.txt to store MD5-based password hashes for the following passwords:

- abc (Hash: `$1$jmmGI4Bk$b2jfTpo3tVyfhpjpFPYY7.`)
- 123 (Hash: `$1$r508PTUb$DEMxiJ0zbak6aA9TTQVmo0`)
- password1234 (Hash: `$1$06a4suW5$gjplux546hPFnWCczjOuH0`)
- P@ssw0rd123 (Hash: `$1$ZLATucMy$naFn96apH/YuulWImgJrz/`)

We can use a tool like openssl to create a hash of a simple password. For example, to create an MD5 hash of the password "123":

```
openssl passwd -1 123
```

There are different hashing algorithms, like SHA1, SHA256, etc., but for simplicity, I will use MD5 in this example.

III. Cracking Simple Passwords

A. Dictionary Attack

John the Ripper performs a dictionary attack by using a wordlist to test potential passwords and comparing the resulting hashes to those in the given file. In this case, the wordlist used is the popular rockyou.txt, located at /usr/share/wordlists/rockyou.txt.

```
john --wordlist=/usr/share/wordlists/rockyou.txt --format=md5crypt passwords.txt
```

Explanation of the command:

- –wordlist=/usr/share/wordlists/rockyou.txt: Specifies the wordlist for the attack.
- –format=md5crypt: Specifies the hash format being used (MD5-based crypt format here).
- passwords.txt: The file that contains the password hashes to be cracked.

John the Ripper will go through each word in the wordlist, generate its hash, and compare it with the hashes in the provided file to find any matches.

B. Check Progress and Results

After running the attack, any cracked passwords can be checked with the following command:

```
john --show passwords.txt
```

This command will display any passwords that were successfully cracked from the hashes in passwords.txt. In this case, John the Ripper successfully cracked the simple passwords.

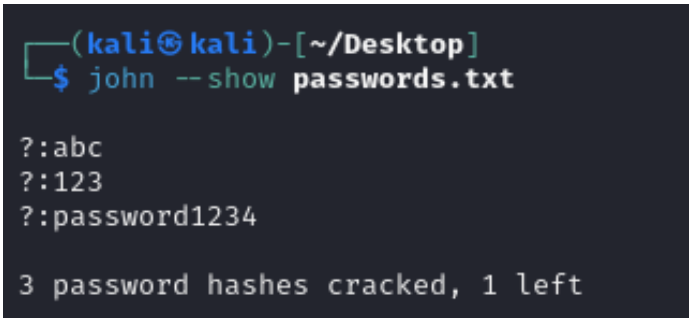

Fig. 1. Output of show command

IV. Cracking More Complex Passwords

In this phase of the project, I tested John the Ripper's ability to crack a more complex password: P@ssw0rd123. This password incorporates a mix of uppercase and lowercase letters, special characters, and numbers, making it stronger than simpler passwords.

A. Applying Rules

John the Ripper also allows us to apply rules to modify words in the wordlist, increasing the chances of cracking passwords that are simple variations or common patterns. For example, it can attempt changes like capitalizing the first letter or appending a number to the end of a word.

```
john --wordlist=/usr/share/wordlists/rockyou.txt --rules --format=md5crypt passwords.txt
```

Explanation of the command:

- –rules: Applies predefined transformations to each word in the wordlist (e.g., changing letter case, adding numbers, etc.).
- This increases the number of variations tested and can improve the chances of cracking more complex passwords.

B. Brute Force Attack

A brute-force attack attempts every possible combination of characters to crack a password. This is computationally expensive, especially for longer passwords, but it can work when no other methods succeed.

```
john --incremental --format=md5crypt passwords.txt
```

Explanation of the command:

- –incremental: Tells John the Ripper to perform a brute-force attack by trying all combinations of characters.
- This command can take a long time to run, especially for complex passwords.

C. Hybrid Attack

A hybrid attack combines both dictionary-based and brute-force methods. John the Ripper will first try words from the wordlist and then apply brute-force transformations (e.g., adding numbers at the end).

```
john --wordlist=/usr/share/wordlists/rockyou.txt
            --rules --incremental
        --format=md5crypt passwords.txt
```

This combines the dictionary attack with brute-force, allowing John the Ripper to try both common words and variations (e.g., adding numbers, changing letter case).

D. Check Progress and Results

After running these advanced attacks, the cracked passwords can be checked with the following command:

```
john --show passwords.txt
```

In this case, John the Ripper successfully cracked the P@ssw0rd123.

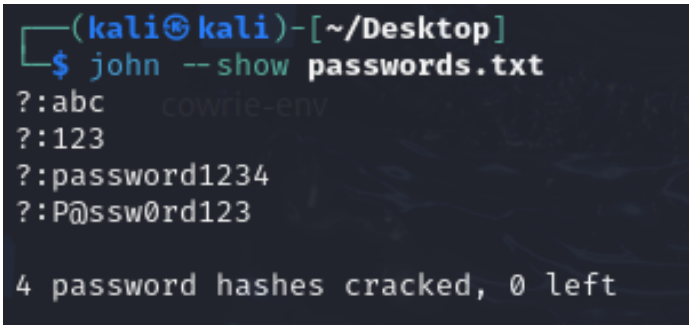

Fig. 2. Output of show command

V. Comparison: Simple vs. Complex Passwords

Cracking passwords with John the Ripper showcases the varying difficulty levels between simple and complex passwords. This section compares the approaches and results for cracking the simple passwords (abc, 123, and password1234) versus the complex password (P@ssw0rd123).

A. Performance of Cracking Techniques

The time and effort required to crack a password depend heavily on its complexity, the hash type, and the attack method.

- Simple Passwords:
  - Cracked using a basic dictionary attack with the wordlist `rockyou.txt`.
  - Time to crack was minimal since these passwords were commonly found in the wordlist without requiring transformations.
  - Demonstrates the vulnerability of using weak, common passwords.
- **Complex Password (‘P@ssw0rd123‘):**
  - Required rule-based attacks to account for variations in capitalization, symbols, and numbers.
  - Took significantly longer due to the need to test numerous permutations and modifications.
  - Brute-force methods could also crack the password, but they were computationally expensive and time-intensive.
  - Highlights the effectiveness of rule-based transformations in tackling passwords that mimic strong patterns.

B. Effectiveness of Attack Methods

TABLE I
Comparison of Cracking Techniques for Simple and Complex Passwords

Metric	Simple Passwords	Complex Password (P@ssw0rd123)
Dictionary Attack	Highly effective	Ineffective without rules
Rule-Based Attack	Not necessary	Highly effective
Brute-Force Attack	Redundant	Effective but time-consuming
Hybrid Attack	Not required	Effective
Time to Crack	Fast	Significantly longer

C. Key Observations

- Simple passwords are often found in pre-compiled wordlists, making dictionary attacks the most efficient method.
- Complex passwords require advanced methods, such as rule-based or hybrid attacks, to account for common password patterns and modifications.
- Brute-force attacks are a last resort and only practical for shorter passwords or when no other method succeeds.
- Rule-based attacks strike a balance between efficiency and effectiveness, making them ideal for cracking moderately complex passwords like P@ssw0rd123.

D. Challenges Encountered

A major challenge in this phase of the project was the failure to crack the password ”p@ssword123@”. Despite employing various attack strategies, including dictionary, rule-based, and hybrid attacks, John the Ripper could not successfully break this password. Possible reasons for this failure include:

- Password Complexity: The combination of special characters and numbers made it more resistant to basic dictionary and rule-based attacks.

- Lack of Appropriate Wordlist or Rules: The chosen wordlist and rules may not have covered common variations or patterns associated with this particular password.

This challenge underscores the necessity for more advanced methods or customized wordlists to crack passwords that include special characters and greater complexity. It also points out the limitations of traditional cracking techniques and emphasizes the value of using more sophisticated or customized rules to handle increasingly complex password formats.

E. Lessons Learned

This phase of the project reinforced the critical need for strong, unpredictable passwords. Key takeaways include:

- Common patterns, such as predictable substitutions (e.g., ”o” replaced with ”0”), make even complex-looking passwords vulnerable to rule-based attacks.
- Simple passwords are easily cracked using dictionary attacks, underscoring the need for more robust password choices.
- Truly secure passwords require a combination of randomness, length, and complexity that cannot be easily guessed or matched by dictionary or hybrid attacks.
- Advanced techniques like brute-force are effective but require significant time and computational resources for longer passwords.

This knowledge highlights the importance of educating users about secure password practices and implementing tools like password managers to improve security.

VI. Conclusion

This phase of the project successfully showcased how John the Ripper can crack simple to moderately complex password hashes using a variety of attack methods. The results demonstrated the effectiveness of dictionary and rule-based attacks in identifying weak passwords, while also revealing the limitations of brute-force and hybrid techniques when confronted with more complex, random passwords.

The findings emphasize the ongoing challenge of balancing security with usability in password creation. They also underscore the importance of promoting strong password habits, such as avoiding predictable patterns and adopting truly random, lengthy combinations.

Future work will focus on exploring additional hash formats, experimenting with optimized rule sets, and integrating advanced techniques to crack even more sophisticated passwords. These steps will further enhance my understanding of password vulnerabilities and help develop better defenses against potential attacks.