John The Ripper

Crack the hash

Mina Atef 22-05542 Mina Adel 22-05319 IT infrastructure course

🔐 Password Security Testing with John the Ripper

Introduction

In this project, we explored **password vulnerabilities** using **John the Ripper** on **Kali Linux** (running on VMware). Our goal was to understand how different **hashing algorithms** and **salting techniques** impact password security in real-world scenarios.

🎯 Objectives

- **Evaluate Password Strength** using various hash algorithms.
- **Learn Ethical Hacking Tools** for password security testing.
- **Apply Research to Practice** by using real tools and real-world password datasets.

🛠️ Tools & Setup

Operating System:

- **Kali Linux (in VMware)**

Password Cracking Tool:

- **John the Ripper**

Wordlist:

- **rockyou.txt** – A file containing millions of common passwords used for testing.
  Note: The `rockyou.txt` file used was downloaded directly via Firefox in an uncompressed format, so no decompression was needed.

Installation:

```
sudo apt install john
```

🔍 What We Did

1. **Downloaded and Prepared `rockyou.txt`**
   - Downloaded directly from the internet via Firefox as an uncompressed text file containing millions of passwords.
2. **Generated Hashes**
   - Used online tools to generate hashes for test passwords (e.g., MD5, SHA-1, bcrypt).
3. **Ran Cracking Tests with John the Ripper**
   Example:

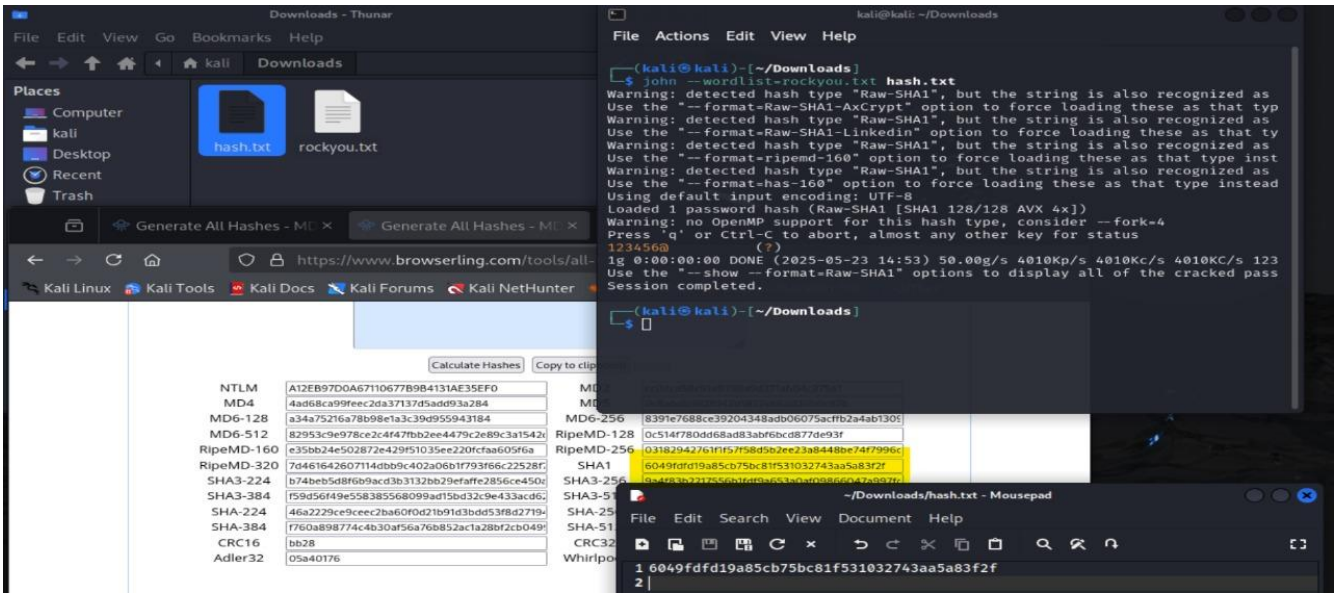
   ```
   john --wordlist=rockyou.txt hash.txt
   ```

4. **Tested bcrypt Hashes**
   - bcrypt includes automatic salting.
   - John was able to crack some weak passwords (e.g., ones found in rockyou.txt).
   - Strong passwords remained uncracked.

🔐 Hashing Algorithms Tested

Algorithm	Security Level	Notes
MD5	❌ **Weak**	**Easy to crack.**
SHA-1	⚠️ **Moderate**	**Slightly better, but still not safe for sensitive data.**
bcrypt	☑️ **Strong**	**Uses automatic salting; very resistant to cracking unless the password is weak and common.**

This is a sample



🧂 Salting Note

We did **not manually create or test salted hashes**, but we **used bcrypt**, which **automatically includes a salt**.

What We Learned:

- Even salted hashes (via bcrypt) can be cracked if the password is weak.
- **Salting helps**, but **strong, unique passwords** are still essential.

🧪 How Salting Works

Salting means adding a **random string (salt)** to a password **before hashing** it. This makes each password hash unique, even if the same password is used.

Example:

- Password: `Password!`
- Salt: `L3tme!n`
- Combined: `L3tme!nPassword!` → then hashed

Why It Matters:

- Without salt: same passwords = same hashes
- With salt: same passwords = different hashes (much safer)

Key Points:

- Every user should have a **unique, random salt**
- Salts are stored with hashes in the database
- Salts protect against **precomputed attacks** *(like rainbow tables)*
- Weak passwords can **still be cracked**, even if salted

Conclusion

In this project, we learned that using small or simple passwords makes it easy for attackers to crack hashes, even if salting is used. To protect passwords effectively, it's important to create strong, complex passwords combined with salting. Salting adds extra security by making each hash unique, which helps prevent attackers from easily cracking the password hashes.