Project Title: Hash-Based Password Cracking Using John the Ripper
Tool Used: John the Ripper (JtR)
Environment: Kali Linux Terminal
Duration: 3–4 hours
Date: 8 May 2025
Level: Beginner – Intermediate

1. Objective

This lab project aimed to develop a deep understanding of hash-based password cracking using **John the Ripper**, a popular and powerful open-source tool. The objective was to experiment with multiple password cracking strategies including **dictionary attacks**, **custom wordlists**, **incremental brute-force**, and **realistic shadow file extraction**, mimicking real-world attack scenarios.

2. Tools and Techniques Used

- **John the Ripper**
- `echo`, `md5sum`, `nano`, `unshadow`, `gzip`
- Wordlists: `john.lst`, `rockyou.txt`, custom wordlists
- `hashid` for hash identification
- Kali Linux terminal

3.Step-by-Step Procedure

Step 1: Hash Generation

- Created a test MD5 hash using:

```
echo -n "mypassword" | md5sum > hashes.txt
```

- This simulated a scenario where a password is stored as a hash and the plaintext is unknown.

Step 2: Identify Hash Type

- Used `hashid` to confirm the type of hash:

  ```
  hashid hashes.txt
  ```

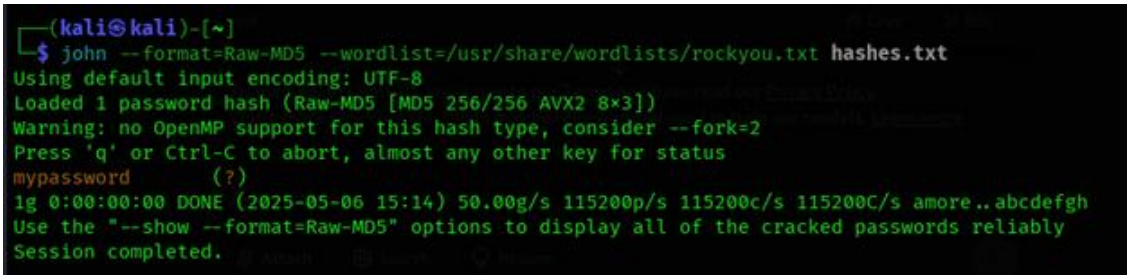
- Ensured John the Ripper uses the correct cracking mode (e.g., `--format=raw-md5`).

Step 3: Wordlist Cracking (Dictionary Attack)

- First attempted using the default `john.lst` wordlist, which was too small and ineffective.
- Then extracted and used the **larger rockyou.txt wordlist**:

  ```
  gzip -d /usr/share/wordlists/rockyou.txt.gz
  john --wordlist=/usr/share/wordlists/rockyou.txt hashes.txt
  ```

- Successfully cracked the hash with a matching entry from `rockyou.txt`.



Step 4: Custom Wordlist Creation

- Created a custom wordlist using:

  ```
  nano custom.txt
  ```

- Added commonly guessed passwords and appended the rockyou list for more thorough coverage:

  ```
  cat rockyou.txt >> custom.txt
  john --wordlist=custom.txt hashes.txt
  ```

Step 5: Cracking Linux Shadow File

- Simulated real-world cracking of `/etc/shadow` by first combining `passwd` and `shadow` files using:

```
unshadow passwd shadow > combined.txt
john combined.txt
```

Step 6: Incremental Brute Force

- Performed a brute-force attack using:

```
john --incremental hashes.txt
```

- This mode attempts **all possible combinations**, ideal when no useful wordlist or hints are available.
- Required significantly more time but demonstrated John's full capability.

4. Observations & Analysis

Technique Used	Result	Strength
Default wordlist (john.lst)	Failed	Too limited
Rockyou.txt	Success	Highly effective for weak/common passwords
Custom wordlist	Success	Good for niche or targeted attacks
`/etc/shadow` cracking	Success (in controlled lab)	Simulates real-world breach

Incremental brute-force	Time-consuming but complete	Effective when no dictionary works

---

5. Skills Gained

- Proficiency in using **John the Ripper** for various cracking modes.
- Ability to identify and match **hash types** with cracking formats.
- Learned **dictionary attack optimization** using custom and large-scale wordlists.
- Gained exposure to **brute force strategies** when no dictionary exists.
- Simulated real-world Linux-based password cracking using `unshadow`.

6. Challenges Faced

- Initially used an incorrect wordlist size (`john.lst`) which failed to crack basic hashes.
- Had to research appropriate formats and syntax (`--format=raw-md5`) for specific hash types.
- Incremental cracking took significant time — required process patience and CPU resource awareness.

7. Key Takeaways

- **Weak passwords are easily recoverable** if common wordlists like `rockyou.txt` are used.
- Identifying the correct **hashing algorithm** is critical to successful cracking.
- **Incremental brute-force** is effective but computationally expensive and best used as a last resort.
- Wordlists are a **critical component** of any successful password cracking strategy.
- The practice highlighted the importance of **password complexity and salting** in real-world systems.