COMP3028 Coursework 3 Group 12 (Task 3)

**Tool Used**

For this simulation, we used **John the Ripper (JtR)**, a widely-used password cracking tool capable of handling a variety of hash formats. The password hashes were generated using **PBKDF2-HMAC-SHA256** with 100,000 iterations, simulating real-world secure hashing schemes.

**Converting Hashes to JtR Format**

Since JtR requires hashes in a specific format to identify PBKDF2-HMAC-SHA256, we first processed the output of our Python hashing function from Task 1, which returns strings in the form salt$hash. We used a custom script to convert this into the format recognized by JtR (typically $pbkdf2-sha256$...), placing the converted hashes into a text file before launching the cracking process.

**Cracking Results**

The following passwords were tested using JtR with the known hashing format. The time taken to crack each password is noted:

|  |  |  |
| --- | --- | --- |
| **Password** | **Cracking Time (seconds)** | **Result** |
| apple | 36.05 | ✅ Cracked |
| king | 1232.24 | ✅ Cracked |
| abc123 | 14.26 | ✅ Cracked |
| qwerty123 | 1627.97 | ✅ Cracked |
| comp9303sec | > 24 hours | ❌ Not cracked |

**Analysis of Results**

The cracking results demonstrate that simple and commonly used passwords are highly vulnerable, even when hashed with a strong algorithm like PBKDF2-HMAC-SHA256:

* **Short/simple passwords** like "apple" and "abc123" were cracked in under a minute.
* **Slightly more complex passwords** like "king" and "qwerty123" took significantly longer, but were still eventually cracked.
* **Longer, less predictable passwords** such as "comp9303sec" remained secure even after 24 hours of attack, demonstrating the effectiveness of strong passwords.

**Recommendations for Improving Password Policies**

To strengthen password security and reduce the risk of successful cracking attempts:

1. **Enforce strong password requirements**:
   * Minimum length of 12 characters
   * Combination of uppercase, lowercase, numbers, and symbols
   * Avoid dictionary words or predictable patterns
2. **Use multi-factor authentication (MFA)**:
   * Supplement passwords with additional authentication steps
   * Protects accounts even if passwords are compromised
3. **Implement account lockout mechanisms**:
   * Temporarily disable login after several failed attempts
   * Reduces the effectiveness of brute-force attacks
4. **Encourage password managers**:
   * Allow users to generate and store strong, unique passwords securely
5. **Monitor for password reuse and leaks**:
   * Alert users when their credentials have appeared in data breaches

**Conclusion**

This simulation underscores the importance of using strong, unique passwords. Tools like John the Ripper can easily break weak passwords—even with advanced hashing techniques. Effective password policies are essential in preventing unauthorized access and protecting user data.

**Appendix: Cracking Outputs**

apple:

A screen shot of a computer

AI-generated content may be incorrect.

king:

A screen shot of a computer

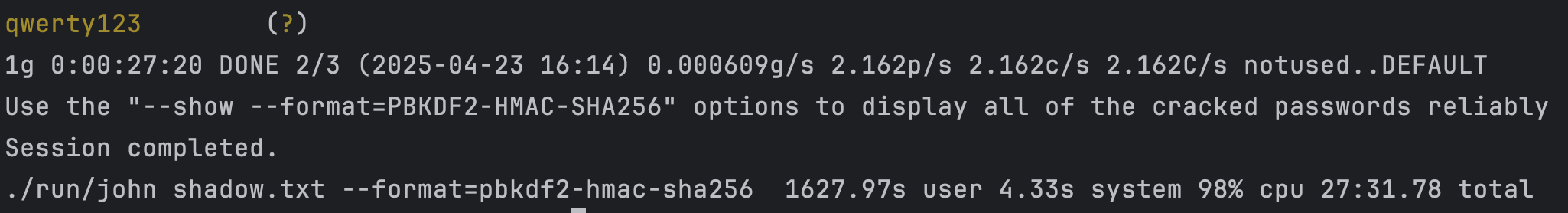
AI-generated content may be incorrect.

abc123:

A screenshot of a computer

AI-generated content may be incorrect.

qwerty123:



comp9303sec:

*crack failed*