Task 4: Password Security & Authentication Analysis

1. Introduction

Password security is a critical component of authentication systems. Weak passwords and improper storage mechanisms are among the most common causes of security breaches. This task focuses on understanding how passwords are stored, how attackers attempt to crack weak passwords, and how strong authentication mechanisms can mitigate such risks.

---

2. Password Storage: Hashing vs Encryption

Hashing

- Hashing is a **one-way cryptographic function**.
- The original password **cannot be retrieved** from the hash.
- Commonly used for password storage.

Examples:

- MD5
- SHA-1
- SHA-256
- bcrypt

Encryption

- Encryption is a **two-way process**.
- Data can be decrypted using a secret key.
- Not recommended for password storage.

Conclusion:
Passwords should always be **hashed**, not encrypted.

---

3. Common Password Hash Types

Hash Type	Description	Security Level
MD5	128-bit hash, very fast	Weak (Broken)
SHA-1	160-bit hash	Weak (Deprecated)

Hash Type	Description	Security Level
SHA-256	Part of SHA-2 family	Strong (without salt = risky)
bcrypt	Adaptive, salted hash	Very Strong

---

4. Password Hash Generation (Example)

Using Linux (Kali)

```
echo -n "password123" | md5sum
echo -n "password123" | sha1sum
echo -n "password123" | sha256sum
```

Using bcrypt (Python example)

```
import bcrypt
password = b"password123"
hashed = bcrypt.hashpw(password, bcrypt.gensalt())
print(hashed)
```

---

5. Hash Identification

Before cracking, identifying the hash type is important.

Methods:

- Hash length
- Character format
- Tools:
    - Hashcat
    - John the Ripper
    - Online Hash Identifiers

Example:

- 32 characters → Likely MD5
- 40 characters → Likely SHA-1
- Starts with $2y$ → bcrypt

---

6. Password Cracking Techniques

Dictionary Attack

- Uses predefined wordlists (e.g., rockyou.txt)
- Effective against weak passwords

Brute Force Attack

- Tries all possible character combinations
- Very slow for long, complex passwords

---

7. Cracking Weak Hashes (Educational Lab)

Using Hashcat (MD5 example)

hashcat -m 0 hash.txt rockyou.txt

Using John the Ripper

john --wordlist=rockyou.txt hashes.txt

Observation:
Simple passwords like 123456, password, and admin are cracked within seconds.

---

8. Why Weak Passwords Fail

Weak passwords fail due to:

- Short length
- Common words
- Lack of complexity
- Reuse across platforms
- Absence of salting

Example:
password123 is vulnerable to dictionary attacks.

---

9. Multi-Factor Authentication (MFA)

What is MFA?

MFA requires **two or more** of the following:

- Something you know (password)

- Something you have (OTP, phone)
- Something you are (biometrics)

Importance of MFA

- Prevents unauthorized access even if password is compromised
- Reduces risk of credential-stuffing attacks

---

10. Recommendations for Strong Authentication

1. Use **bcrypt or Argon2** for password hashing
2. Enforce minimum password length (12+ characters)
3. Apply **salting and key stretching**
4. Enable **Multi-Factor Authentication**
5. Prevent password reuse
6. Implement account lockout policies
7. Conduct regular password audits

---

11. Final Outcome

Through this task, the following were achieved:

- Understanding of password storage mechanisms
- Identification of common hash algorithms
- Practical exposure to password cracking techniques
- Awareness of password attack methods
- Knowledge of defensive strategies and MFA importance

---

12. Conclusion

Password security plays a vital role in protecting systems from unauthorized access. Weak passwords and outdated hashing algorithms significantly increase the risk of compromise. Implementing strong hashing methods, enforcing password policies, and enabling MFA are essential best practices for secure authentication systems.

NAME: NITIN G N

Submission: Task 4

Date & Day: 20th January, Tuesday

Internship: Elevate Labs

THANKYOU