task 4

1. Password Storage: Hashing vs. Encryption

Securely storing passwords requires understanding the fundamental difference between these two cryptographic processes.

- **Hashing (One-Way):** This process converts input data of any size into a fixed-length string of characters, often called a digital fingerprint or message digest. Hashing is **irreversible**; once data is hashed, the original plaintext cannot be retrieved from the resulting hash value. It is used for verifying data integrity and securely storing passwords without revealing the actual password to the system.
- **Encryption (Two-Way):** Encryption transforms readable plaintext into unreadable ciphertext using mathematical algorithms and cryptographic keys. Unlike hashing, it is **reversible**; the original data can be recovered by someone possessing the correct decryption key. This makes encryption ideal for protecting the confidentiality of data during transmission (e.g., emails) or storage (e.g., sensitive customer records).

---

2. Identifying and Generating Hash Types

Different algorithms have distinct characteristics that determine their security level and identify them during an audit.

- **MD5 (Message Digest 5):** Developed in 1991, it produces a 128-bit (32-character) hash. Although fast and simple, it is now considered **insecure** due to its vulnerability to collisions—where two different inputs produce the same hash—making it easy for modern hardware to crack.
- **SHA-1 (Secure Hash Algorithm 1):** Designed by the NSA, it produces a 160-bit (40-character) hash value. While more complex than MD5, it has also been found to have significant vulnerabilities over time and is no longer recommended for secure password storage.
- **bcrypt:** A modern, secure hashing algorithm that includes **key stretching** to make it intentionally slow to compute. This makes it highly resistant to high-speed brute-force attacks.
- **Generating Hashes:** On Linux, you can generate these hashes via the command line for testing:
  - **MD5:** `echo -n "password" | md5sum`
  - **SHA-1:** `echo -n "password" | sha1sum`
  - **Secure Linux Hashes:** Tools like `mkpasswd` or `openssl passwd` can generate modern hashes (like SHA-512) typically found in the `/etc/shadow` file.

3. Password Cracking Methodology

Attackers use several methods to bypass password protections, focusing on computational power or human psychology.

- **Dictionary Attack:** This method uses a precompiled list of likely words, phrases, or previously leaked passwords (dictionaries). It is faster and more efficient than brute force because it capitalizes on the common human behavior of using simple, memorable words.
- **Brute Force Attack:** This is an exhaustive trial-and-error approach that systematically attempts every possible combination of characters (e.g., a-z, 0-9, special symbols) until the correct password is found. While guaranteed to find the password eventually, it is computationally intensive and very slow for long or complex passwords.
- **Hybrid Attack:** Combines both methods by taking words from a dictionary and adding common variations, such as numbers or special symbols (e.g., "password123!").

---

4. Why Weak Passwords Fail

Security analysis reveals that weak passwords are the most common entry point for unauthorized access.

- **Predictability:** Passwords based on common words, simple sequences (e.g., "123456"), or personal information (birthdays, names) are the first targets in dictionary attacks.
- **Short Length:** Every additional character in a password exponentially increases the combinations an attacker must try. Short passwords (typically less than 8 characters) can be cracked in milliseconds by modern tools.
- **Credential Stuffing:** Once a weak password is leaked in one data breach, attackers use automated tools to try that same password on multiple other platforms, leading to cross-platform account intrusions.

---

5. Multi-Factor Authentication (MFA) and Recommendations

Strong authentication relies on layered defenses to protect accounts even when passwords are compromised.

- **Importance of MFA:** MFA requires two or more different categories of identification: something you **know** (password), something you **have** (authenticator app, hardware token), or something you **are** (fingerprint, facial recognition). It can block up to 99.9% of automated account attacks because stolen passwords alone are insufficient to gain access.

- **Recommendations for Strong Authentication:**
  - **Enforce Complex Policies:** Mandate a minimum length of 12+ characters and the use of diverse character types.
  - **Use Salting:** Add random data (a salt) to each password before hashing to ensure identical passwords produce unique hashes, neutralizing precomputed "rainbow table" attacks.
  - **Implement Modern Algorithms:** Use "slow" hashes like **Argon2id** or **bcrypt** to defend against brute-force attempts.
  - **Incident Response:** Develop plans to reset passwords and notify users immediately upon detecting potential leaks or consecutive failed login attempts.