Password Security Analysis Report

1. How Passwords Are Stored: Hashing vs Encryption

Passwords should **never be stored in plain text**. Instead, systems use **hashing** or sometimes **encryption**, but both are very different in purpose.

Hashing is a one-way process. When a user creates a password, it is converted into a fixed-length string called a *hash*. This hash cannot be reversed back to the original password. During login, the entered password is hashed again and compared with the stored hash. If both hashes match, access is granted.

Encryption, on the other hand, is a two-way process. Encrypted data can be decrypted back to the original form using a secret key. Encryption is useful for protecting data like files or messages, but it is **not ideal for passwords** because if the key is leaked, all passwords can be recovered.

---

2. Different Hash Types (MD5, SHA-1, bcrypt)

MD5

MD5 produces a 128-bit hash and was widely used in older systems. However, it is **very fast and broken**, meaning attackers can crack it easily using modern hardware.

Example:

Password: password123

MD5 Hash: 482c811da5d5b4bc6d497ffa98491e38

SHA-1

SHA-1 produces a 160-bit hash. It is stronger than MD5 but still considered **insecure** today due to collision attacks.

Example:

Password: password123

SHA-1 Hash: cbfdac6008f9cab4083784cbd1874f76618d2a97

bcrypt

bcrypt is a **modern and secure hashing algorithm**. It uses **salting** and **multiple rounds**, making it slow and resistant to brute-force attacks.

Example:

$2b$12$KIX0RZ6Yh9r0dZQx4p9MxuZ2ZQnYqQzF7zG2x1zZc7Zk1Z9bZcQ2K

---

3. Generating Password Hashes (Linux Practice)

On Linux, password hashes can be generated using built-in tools.

Generate MD5 Hash

```
echo -n password123 | md5sum
```

Generate SHA-1 Hash

```
echo -n password123 | sha1sum
```

Generate bcrypt Hash (using htpasswd)

```
htpasswd -nbB user password123
```

This command generates a bcrypt hash safely.

---

4. Cracking Weak Hashes Using Wordlists

Attackers use **wordlists** containing common passwords to crack weak hashes.

Example using Hashcat

```
hashcat -m 0 hashes.txt wordlist.txt
```

- -m 0 → MD5
- hashes.txt → file containing hashes
- wordlist.txt → common passwords list

If the password is simple like 123456 or password, it can be cracked within seconds.

---

5. Brute Force vs Dictionary Attacks

Brute Force Attack

- Tries **every possible combination**
- Very slow for long passwords
- Example: aaaa → aaab → aaac → ...

Dictionary Attack

- Uses **predefined wordlists**
- Much faster for weak passwords

- Example: tries admin, password, welcome

---

6. Why Weak Passwords Fail

Weak passwords fail due to:

- Short length
- Common words (password, admin)
- Predictable patterns (name123)
- No symbols or complexity

Example:

Password: abc123

Result: Cracked in seconds

Attackers rely on **human behavior**, not random guessing.

---

7. Multi-Factor Authentication (MFA) and Its Importance

MFA adds an **extra layer of security** beyond passwords.

Common MFA factors:

- Something you know → Password
- Something you have → OTP, mobile phone
- Something you are → Fingerprint, face ID

Even if a password is compromised, MFA **prevents unauthorized access**.

---

8. Recommendations for Strong Authentication

To improve password security, the following practices should be followed:

- Use long passwords (minimum 12–16 characters)
- Combine uppercase, lowercase, numbers, and symbols
- Avoid common words and personal information
- Use bcrypt or Argon2 for password hashing
- Implement MFA for all critical systems

- Limit login attempts to prevent brute-force attacks
- Encourage users to use password managers