# **Password Security Analysis Report**

## **1. Introduction**

Passwords remain one of the most widely used authentication mechanisms in modern systems. However, weak password practices and improper storage methods continue to be a leading cause of security breaches. This report analyzes how passwords are stored, common hashing algorithms, attack methods used against weak passwords, and the importance of Multi-Factor Authentication (MFA). It concludes with best-practice recommendations for strong authentication.

## **2. Password Storage Methods**

### **2.1 Hashing**

Hashing is a one-way cryptographic process that converts a password into a fixed-length hash value. The original password cannot be reconstructed from the hash. During authentication, the system hashes the user’s input and compares it to the stored hash.

Advantages:

* One-way (non-reversible)
* Protects passwords even if the database is leaked
* Industry standard for password storage

Best Practice: Use salted and slow hashing algorithms.

### **2.2 Encryption**

Encryption is a two-way process that allows data to be decrypted using a key. While encryption is useful for protecting data in transit or at rest, it is unsuitable for password storage.

Risks:

* If the encryption key is compromised, all passwords can be decrypted.
* Violates modern security standards for authentication systems.

Conclusion: Passwords should always be hashed, not encrypted.

## **3. Common Hashing Algorithms**

Algorithm	Security Status	Description
MD5	Insecure	Extremely fast, easily cracked
SHA-1	Insecure	Vulnerable to collision attacks
SHA-256	Moderate	Secure cryptographically but too fast for passwords
bcrypt	Secure	Slow, adaptive, salted
scrypt	Secure	Memory-hard, resistant to GPU attacks
Argon2	Very Secure	Modern and recommended standard

Fast hashing algorithms are unsuitable for password storage because attackers can test billions of guesses per second.

## **4. Password Hash Generation**

Password hashing converts plain-text passwords into unreadable strings.

Examples:

* MD5 and SHA hashes generate the same output for the same input.
* bcrypt produces different hashes for the same password due to salting.

This salting mechanism protects against rainbow table attacks and mass hash cracking.

## **5. Password Cracking Techniques**

### **5.1 Dictionary Attacks**

Dictionary attacks use predefined lists of common passwords obtained from previous data breaches. These attacks are fast and highly effective against weak passwords.

Effectiveness: Very high against reused or predictable passwords.

### **5.2 Brute Force Attacks**

Brute force attacks attempt every possible password combination. While guaranteed to succeed eventually, they become impractical as password length and complexity increase.

Password Length	Estimated Time
6 characters	Seconds
8 characters	Minutes–hours
12+ characters	Years or longer

Key Difference: Dictionary attacks rely on human behavior; brute force relies on computation.

## **6. Analysis of Weak Password Failures**

Weak passwords fail due to several predictable factors:

* Use of common words or patterns
* Reuse across multiple platforms
* Short length
* Inclusion in leaked credential databases

Attackers rarely guess passwords manually; instead, they automate attacks using previously compromised data.

## **7. Multi-Factor Authentication (MFA)**

Multi-Factor Authentication enhances security by requiring multiple verification factors:

1. Something the user knows (password)
2. Something the user has (mobile device, security token)
3. Something the user is (biometrics)

Security Impact:

* Prevents unauthorized access even if passwords are compromised
* Blocks the majority of automated credential attacks
* Essential for high-value and administrative accounts

## **8. Recommendations for Strong Authentication**

### **8.1 User Recommendations**

* Use password managers to generate and store credentials
* Create passwords with a minimum length of 14–16 characters
* Avoid password reuse
* Enable MFA on all supported services

### **8.2 Organizational Recommendations**

* Store passwords using bcrypt, scrypt, or Argon2
* Implement rate limiting and account lockout mechanisms
* Enforce MFA for sensitive systems
* Monitor login attempts for credential-stuffing attacks