1. Learn How Passwords Are Stored

(Hashing vs Encryption)

Explanation

- Hashing converts a password into a fixed-length value using a mathematical function.
- It is a one-way process, meaning the original password cannot be recovered.
- Encryption is a two-way process and can be reversed using a key, so it is not safe for storing passwords.

Operating systems and websites store password hashes, not real passwords.

Example (Hashing a Password):

Bash

Echo -n admin123 | md5sum

Output:

(0192023a7bbd73250516f069df18b500)

🔷 The password admin123 is now stored as a hash.

🔷 Even if the database is stolen, the real password is hidden.

2. Identify Different Hash Types

(MD5, SHA-1, bcrypt)

Explanation

Different systems use different hash algorithms.

Each hash type has unique length and format.

Hash Type	Length / Format
MD 5	32 hex characters
SHA-1	40 hex characters

Bcrypt Starts with $2a$ or $2b$

Example (Identifying Hash Type)

Bash

Hashid 5f4dcc3b5aa765d61d8327deb882cf99

Output:

MD5

🔷 This tells us the password uses MD5, which is weak.

3. Generate Password Hashes

Explanation

Security testers generate hashes to test password strength and compare security algorithms.

Example 1: Generate MD5 Hash

Bash:

Echo -n hello123 | md5sum

Example 2: Generate SHA-1 Hash

Bash:

Echo -n hello123 | sha1sum

Example 3: Generate bcrypt Hash (Strong)

Bash:

Htpasswd -nbBC 12 user hello123

Output : user:$2y$12$...

🔷 bcrypt uses salt + slow hashing, making it very secure.

4.Attempt Cracking Weak Hashes Using Wordlists

Explanation

Weak hashes can be cracked using dictionary attacks, which compare hashes of known passwords.

Tool used: John the Ripper

Example (Cracking an MD5 Hash)

Step 1: Create hash file

Bash:

Nano hash.txt

Add:

5f4dcc3b5aa765d61d8327deb882cf99

Step 2: Crack hash

Bash:

John –wordlist=/usr/share/wordlists/rockyou.txt hash.txt

Step 3: Show cracked password

Bash:

John –show hash.txt

Output:

Password

🔷 The hash is cracked because the password was common.

5.Brute Force vs Dictionary Attack

Explanation

🔷 Dictionary Attack

- Uses a list of known passwords
- Faster
- Limited to wordlist

🔷 Brute Force Attack

- Tries every possible combination
- Very slow
- Guaranteed result if no lockout

Example: Dictionary Attack

Bash:

John –wordlist=rockyou.txt hash.txt

Example: Brute Force Attack

Bash:

John –incremental hash.txt

🔷 Brute force is rarely practical against strong passwords.

6. Analyze Why Weak Passwords Fail

Explanation

Weak passwords:

1. Short
2. Common words
3. Predictable patterns
4. Reused across sites
5. Such passwords exist in wordlists and get cracked quickly.

Example

- Password: Admin123
- Reason it fails: Admin is common
- 123 is predictable

Cracking command:

Bash:

John –wordlist=rockyou.txt hash.txt

🔷 Cracked within seconds.

6. Study MFA and Its Importance

Explanation

Multi-Factor Authentication (MFA) requires more than one verification factor:

- Something you know (password)
- Something you have (OTP / phone)
- Something you are (fingerprint)
- Even if the password is cracked, MFA blocks login.

Example Scenario

- Attacker cracks password using Kali
- Login fails because OTP is required
- Account remains safe

MFA defeats password-only attacks.

8. Write Recommendations for Strong Authentication

Explanation

Strong authentication reduces hacking risk and protects user data.

Example Recommendations:

1. Use bcrypt or Argon2 for hashing
2. Password length ≥ 12 characters
3. Mix:
    - Uppercase
    - Lowercase
    - Numbers
    - Symbols
4. Enable MFA
5. Lock account after failed attempts
6. Avoid common passwords
7. Use password managers

Final Summary:

Topic	Tool
Hashing	Md5sum, sha1sum
Hash identification	Hashid
Secure hashing	Htpasswd (bcrypt)
Cracking	John the Ripper
Wordlist	Rockyou.txt
Defense	MFA

Password Security Analysis Report

STEP 1: Define the Objective

What to write in report (Objective section):

The objective of this experiment is to analyze password security mechanisms by studying password hashing techniques, identifying different hash types, generating password hashes using Linux commands, understanding password cracking methods, and evaluating the importance of strong passwords and multi-factor authentication.

STEP 2: Study How Passwords Are Stored in Linux

Action (Linux):

Bash:

Sudo cat /etc/shadow

What to observe:

Passwords are not stored in plain text

They are stored as hashes

Hash type is identified by prefixes like 6

What to write:

Linux stores passwords using hashing, not encryption

Hashing is one-way and secure/etc/shadow file

Is protected

STEP 3: Identify Different Hash Types

Action (Linux):

Use known prefixes or hashid (if available):

Bash:

Hashid <hash_value>

Hash Identification Table (write in report):

Prefix.	Algorithm
1	MD5
5	SHA-256
6	SHA-512
$2y$	Bcrypt

What to write:

Different algorithms provide different security levels

Older hashes like MD5 and SHA-1 are weak

STEP 4: Generate Password Hashes Using Linux Commands

Choose a sample password (example: hello123).

Commands:

Bash:

Echo -n “hello123” | md5sum

Echo -n “hello123” | sha1sum

Echo -n “hello123” | sha256sum

Echo -n “hello123” | sha512sum

Openssl passwd -bcrypt

What to write:

Same password generates different hashes

Bcrypt is slow and more secure

Echo -n avoids newline issues

STEP 5: Analyze Hash Strength

What to observe:

- MD5 & SHA-1 hashes are short and fast
- SHA-256 & SHA-512 are stronger
- Bcrypt includes salt and cost factor

What to write:

- Fast hashes are vulnerable to attacks
- Password hashing should be slow

STEP 6: Demonstrate Password Cracking Concept (Educational)

Only use self-generated hashes

Dictionary attack concept:

Bash

John hash.txt –wordlist=rockyou.txt

Brute force concept:

Bash

John –incremental hash.txt

What to write:

- Dictionary attacks are faster
- Brute force attacks take more time
- Weak passwords fail quickly

STEP 7: Analyze Why Weak Passwords Fail

Observations:

- Short passwords are cracked easily
- Common words exist in wordlists

- Patterns like 123, @, years are predictable

What to write:

- Weak passwords reduce security
- Reuse of passwords increases risk

STEP 8: Study Multi-Factor Authentication (MFA)

Concept:

- MFA uses more than one authentication factor
- Example: Password + OTP

Linux example (optional):

Bash

Google-authenticator

What to write:

- MFA prevents account compromise even if password is leaked
- Strongly recommended for sensitive systems

STEP 9: Recommendations for Strong Authentication

Write these points:

- Use bcrypt or SHA-512

- Password length ≥ 14 characters
- Avoid dictionary words
- Enable MFA
- Lock accounts after failed attempts
- Use password managers

STEP 10: Conclusion

Sample Conclusion:

This experiment demonstrates that weak passwords and outdated hashing algorithms are vulnerable to attacks. Linux systems use secure hashing mechanisms to protect user credentials. Strong passwords, modern hashing algorithms, and multi-factor authentication significantly improve system security.

STEP 11: Final Report Structure (Very Important)

Your Password Security Analysis Report should be in this order:

- Title
- Objective
- Tools Used (Linux, Terminal)
- Theory
- Procedure (Steps 1–9)
- Observations

- Result
- Conclusion