Shaik Abdul Muneer

ELEVATE LABS
TASK 4
Password Security & Authentication Analysis

1. How passwords are stored:- Hashing and Encryption

What is Hashing

Hashing is a one-way mathematical function that converts a password into a fixed-length string called a hash.

Password: myPassword123

Hash: $2b$10$N9qo8uLOickgx2ZMRZo4i.eQXj6c...

How Login Works

1. User enters password
2. System hashes the entered password
3. Hash is compared with stored hash
4. If both match → login allowed

Encryption (Less Secure for Passwords)

What is Encryption?

Encryption is a two-way process:

1. Data is encrypted using a key
2. Data can be decrypted back using the same key

Best Practices for Password Storage

1.Use bcrypt / Argon2 / scryp
2. Always use salting
3. Never store plain-text passwords
4. Apply rate limiting & account lockout
5. Use multi-factor authentication (MFA)

2. Different Hash Types

1.MD5

1. MD5 is a cryptographic hash function that generates a 128-bit hash value from an input of any length.

2. It was widely used in the past for data integrity and password storage.

3. MD5 is considered cryptographically broken due to its vulnerability to collision attacks, where two different inputs can produce the same hash.

4. MD5 is very fast, attackers can perform brute-force and rainbow table attacks easily.

5. MD5 is not suitable for password storage in modern systems and is only seen today in legacy systems or for non-security purposes like checksums.

2. SHA -1

1. SHA-1 is a cryptographic hash function developed by the NSA that produces a 160-bit hash value.

2. It was designed to be more secure than MD5 and was widely adopted in digital signatures, SSL certificates, and password hashing.

3. Although stronger than MD5, SHA-1 is still too fast and vulnerable to modern attack techniques.

4. Deprecated for password storage

3. bcrypt

1. bcrypt is a password-hashing function specifically designed for secure password storage.

2. Unlike MD5 and SHA-1, bcrypt is intentionally slow, which makes brute-force attacks computationally expensive.

3. It automatically applies a unique salt to each password, protecting against rainbow table attacks.

4. bcrypt also supports a cost factor (work factor), which allows the hashing process to be adjusted as hardware becomes more powerful.

5. Due to the above features, bcrypt is widely used in modern authentication systems and is considered highly secure and industry-recommended for storing passwords.

3.Generate password hashes

Generate MD5 Hash

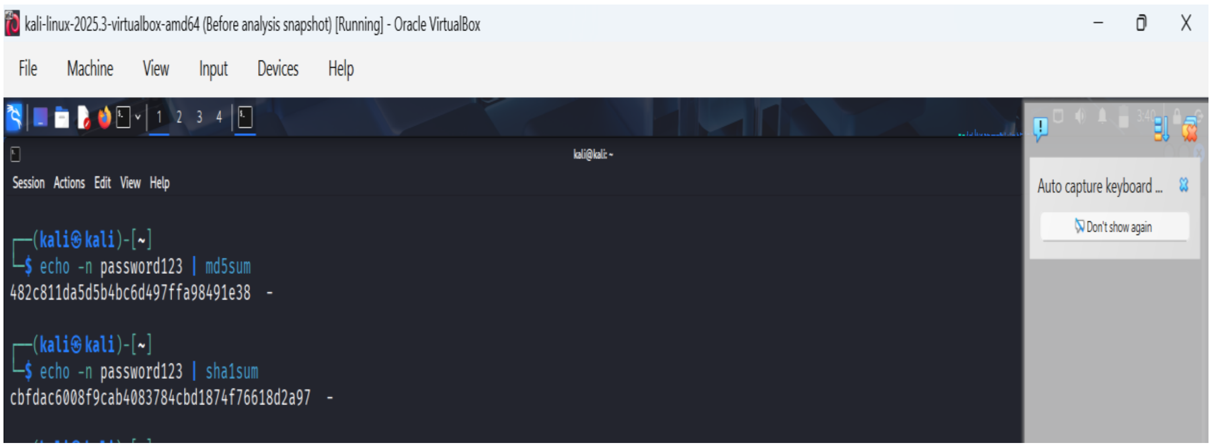
1. Give the command as echo -n password123 | md5sum

2. The output is as follows
482c811da5d5b4bc6d497ffa98491e38

Generate SHA-1 Hash

1. Give command as echo -n password123 | sha1sum

2.The output will be as cbfdac6008f9cab4083784cbd1874f76618d2a97


kali-linux-2025.3-virtualbox-amd64 (Before analysis snapshot) [Running] - Oracle VirtualBox
File Machine View Input Devices Help
kali@kali: ~
Session Actions Edit View Help
Auto capture keyboard ...
Don't show again

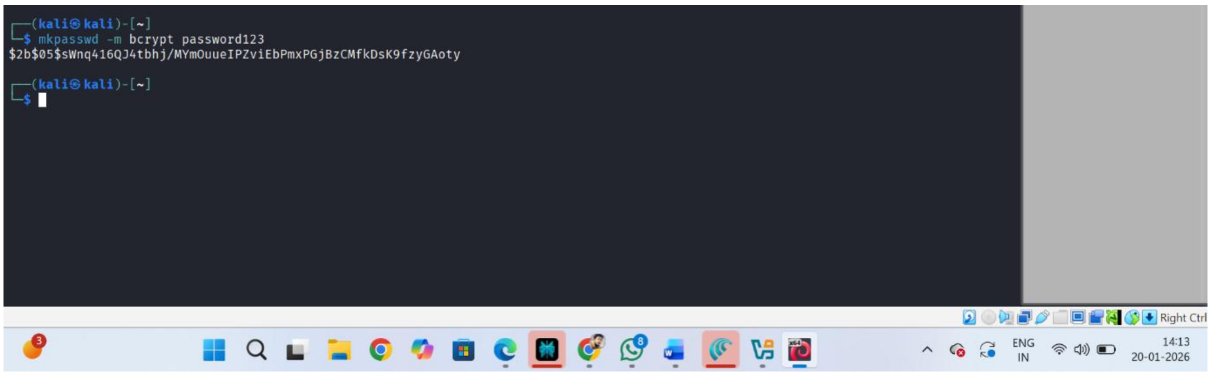
```
┌──(kali㉿kali)-[~]
└─$ echo -n password123 | md5sum
482c811da5d5b4bc6d497ffa98491e38  -

┌──(kali㉿kali)-[~]
└─$ echo -n password123 | sha1sum
cbfdac6008f9cab4083784cbd1874f76618d2a97  -
```

Generate bcrypt Hash

1. Give command as mkpasswd -m bcrypt password123

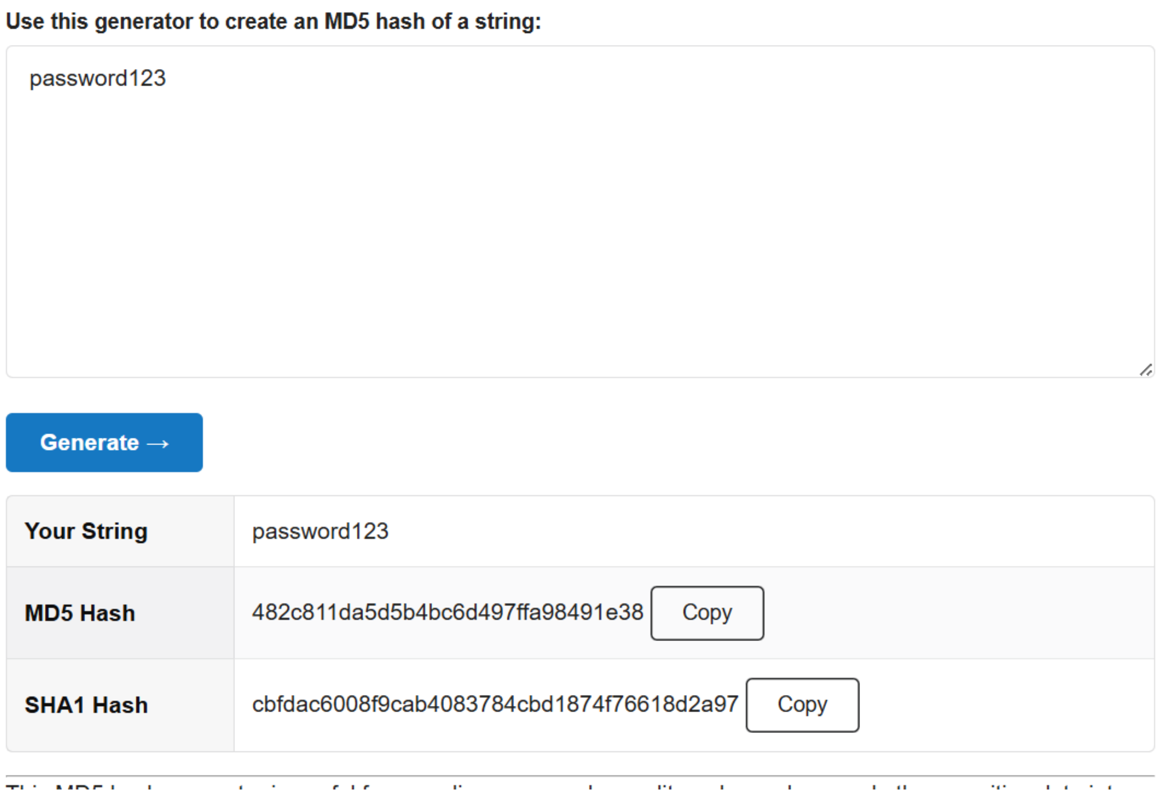
2.The output is as follows
$2y$10$QFZkX1z0C8XJw1J6RZ7rDuGkG7kR0dJ4N9PzKJvYF7hJvP0p4QZ9a



```
┌──(kali㉿kali)-[~]
└─$ mkpasswd -m bcrypt password123
$2b$05$sWnq416QJ4tbhj/MYmOuueIPZviEbPmxPGjBzCMfkDsK9fzyGAoty

┌──(kali㉿kali)-[~]
└─$ 
```

USING Hash Generators



Use this generator to create an MD5 hash of a string:

password123

Your String	password123
MD5 Hash	482c811da5d5b4bc6d497ffa98491e38 Copy
SHA1 Hash	cbfdac6008f9cab4083784cbd1874f76618d2a97 Copy

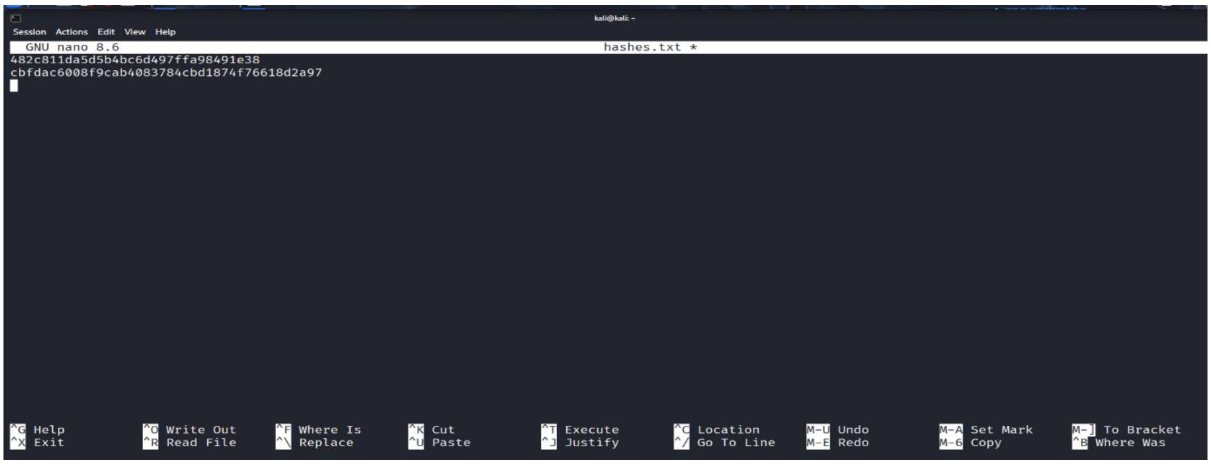
4. Attempt cracking weak hashes using wordlists.

John the Ripper (JtR) is a password-cracking tool that attempts to recover plaintext passwords from hashed values.

One of the most effective methods for cracking weak passwords is a dictionary (wordlist) attack, where John tests each word from a wordlist against the hash.

Steps:-

1. Prepare the Hash Value



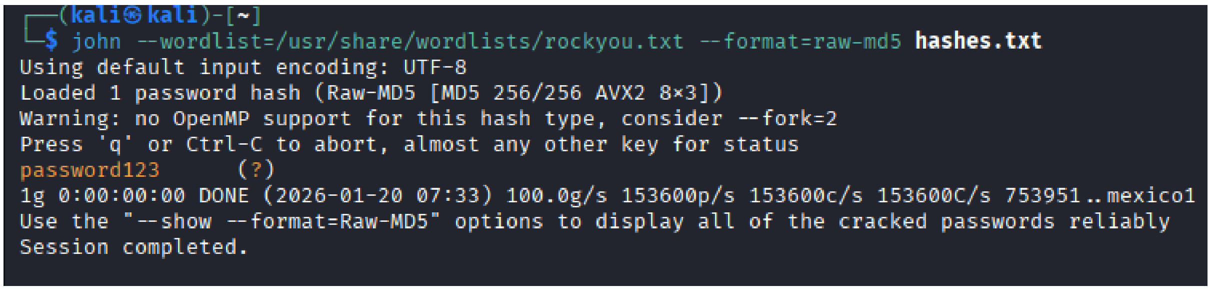
```
  GNU nano 8.6                                   hashes.txt *
482c811da5d5b4bc6d497ffa98491e38
cbfdac6008f9cab4083784cbd1874f76618d2a97


^G Help       ^O Write Out  ^F Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark  M-] To Bracket
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy      ^B Where Was
```

2. Choose a Wordlist

3. Crack MD5 Hash Using Wordlist

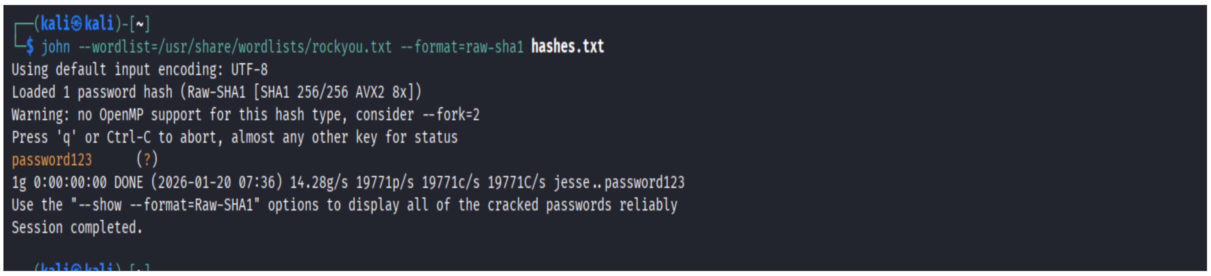
Give the command as john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 hashes.txt



```
┌──(kali㉿kali)-[~]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 hashes.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8×3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
password123      (?)
1g 0:00:00:00 DONE (2026-01-20 07:33) 100.0g/s 153600p/s 153600c/s 153600C/s 753951..mexico1
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

4. Crack SHA-1 Hash Using Wordlist

Give the command as john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-sha1 hashes.txt

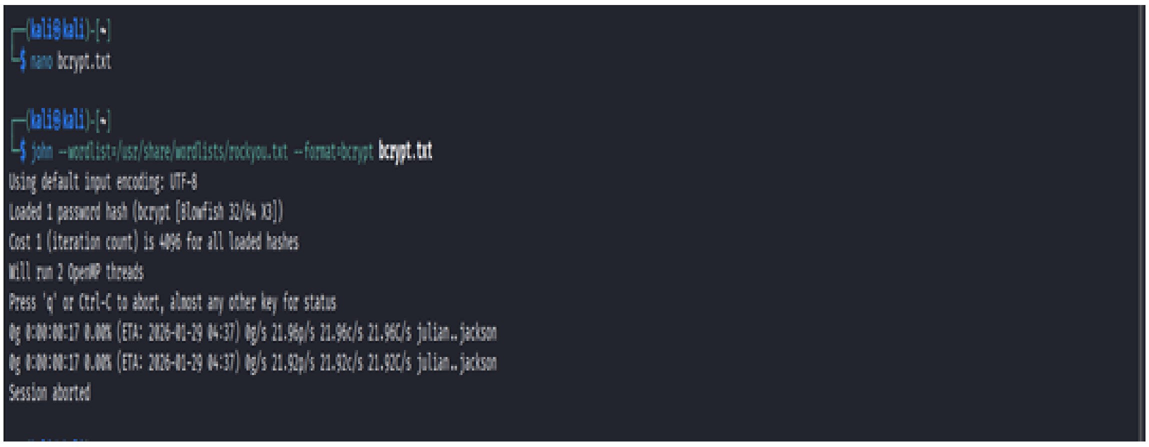


```
┌──(kali㉿kali)-[~]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-sha1 hashes.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA1 [SHA1 256/256 AVX2 8x])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
password123      (?)
1g 0:00:00:00 DONE (2026-01-20 07:36) 14.28g/s 19771p/s 19771c/s 19771C/s jesse..password123
Use the "--show --format=Raw-SHA1" options to display all of the cracked passwords reliably
Session completed.
```

5. Crack brcypt Hash using wordlist
nano bcrypt.txt

Give command as
$2b$12$QFZkX1z0C8XJw1J6RZ7rDuGkG7kR0dJ4N9PzKJvYF7hJvP0p4QZ9a

save and exit



```
┌──(kali㉿kali)-[~]
└─$ nano bcrypt.txt

┌──(kali㉿kali)-[~]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=bcrypt bcrypt.txt
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
Cost 1 (iteration count) is 4096 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:17 0.00% (ETA: 2026-01-29 04:37) 0g/s 21.96p/s 21.96c/s 21.96C/s julian..jackson
0g 0:00:00:17 0.00% (ETA: 2026-01-29 04:37) 0g/s 21.92p/s 21.92c/s 21.92C/s julian..jackson
Session aborted
```

5.Understand brute force vs dictionary attacks

1. Brute Force

A brute force attack is a method in which all possible combinations of characters are tried systematically until the correct password is found. The attacker does not rely on any prior knowledge of the password. Instead, the attack attempts every possible combination based on a defined character set (letters, numbers, symbols) and password length.

A brute force attack is guaranteed to succeed if there are no restrictions and enough time and resources are available. However, as password length and complexity increase, the number of possible combinations grows exponentially, making brute force attacks computationally impractical for strong passwords.

Modern security systems further reduce the feasibility of brute force attacks by using rate limiting, account lockout mechanisms, and slow hashing algorithms like bcrypt.

Dictionary Attack

A dictionary attack is a more targeted approach that relies on a predefined list of commonly used passwords, known as a dictionary or wordlist. Instead of trying all possible combinations, the attacker tests only words that are likely to be chosen by users, such as simple passwords, leaked passwords, or common phrases.

Dictionary attacks are much faster and more efficient than brute force attacks because they exploit predictable human behavior. Many users choose weak passwords that are short, simple, or reused, making them vulnerable to this type of attack. However, dictionary attacks are not guaranteed to succeed, as they will fail if the actual password does not appear in the wordlist or its variations.

6. .Analyze why weak passwords fail.

Weak passwords fail because they do not provide enough entropy, unpredictability, or resistance against modern attack techniques. Attackers exploit both human behavior and computational efficiency, making simple or common passwords easy to break.

1. Password Entropy Theory

Password entropy represents the degree of randomness in a password. Weak passwords have low entropy because they:

Use a small character set

Have short length

Follow predictable structures

2.Search Space Reduction

In theory, password security depends on the size of the **search space**.

Short passwords reduce the total combinations exponentially

Predictable patterns shrink the effective search space even further

Attackers exploit this by prioritizing the most likely combinations, making weak passwords vulnerable long before the full search space is exhausted.

3.Dictionary-Based Probability Models

From a theoretical standpoint, attackers rely on **probability-based guessing** rather than random guessing.

Common words and phrases have higher probability

Frequently used passwords appear early in attack sequences

Weak passwords align closely with high-probability models, causing early compromise.

4.Human Cognitive Constraints

Humans favor memorability over randomness. This creates:

Repetition,Reuse,Predictable transformations

Attack strategies are theoretically optimized around these predictable behaviors, making weak passwords inherently vulnerable.

7. Study MFA and its importance.

Multi-Factor Authentication (MFA) is a security mechanism that requires a user to verify their identity using two or more independent factors before gaining access to a system. Unlike single-factor authentication, which relies only on a password, MFA combines multiple forms of verification to increase security.

Authentication Factors (Theory)

MFA is based on three fundamental categories of authentication factors:

1. **Something you know**
   This includes knowledge-based information such as passwords, PINs, or security questions.
2. **Something you have**
   This refers to a physical or digital possession, such as a smartphone, hardware token, smart card, or OTP generator.
3. **Something you are**
   This includes biometric characteristics like fingerprints, facial recognition, iris scans, or voice patterns.

Importance of MFA (Theory)

1. Protection Against Password Compromise

Passwords can be stolen through phishing, keylogging, database breaches, or brute-force attacks. MFA ensures that even if a password is compromised, attackers **cannot authenticate without the additional factor**.

2. Defense Against Automated Attacks

Automated attacks such as credential stuffing and dictionary attacks rely solely on passwords. MFA blocks these attacks by introducing a second verification step that cannot be easily automated.

3. Reduction of Single Point of Failure

Single-factor authentication creates a single point of failure. MFA distributes trust across multiple factors, reducing the impact of a single compromised element.

8. Write recommendations for strong authentication.

1. Use Multi-Factor Authentication (MFA)

Combine something you know (password), something you have (security token, mobile authenticator app), and something you are (biometrics like fingerprint or facial recognition).

2. Enforce Strong Password Policies

Minimum length: 12–16 characters.

Include a mix of upper and lower case letters, numbers, and special characters.

Avoid common or reused passwords.

3. Educate Users

Train users on phishing attacks, social engineering, and the importance of MFA.

Promote good password hygiene.

4. Protect Authentication Channels

Always use encrypted communication (e.g., HTTPS, TLS) for transmitting credentials.

Prevent credential interception using secure protocols.

5. Use Hardware Security Keys

Deploy FIDO2 or U2F security keys for high-security accounts.

These keys provide strong phishing-resistant authentication.

9. SUMMARY:-

Passwords are not stored in plain text; they are usually hashed (one-way, can't reverse) or sometimes encrypted (reversible with a key). Common hash types include MD5 and SHA-1 (fast but weak) and bcrypt (slow and secure). Weak passwords can be cracked using wordlists (dictionary attacks) or by trying all possible combinations (brute force). Strong passwords fail less because they are long, unique, and complex. To enhance security, Multi-Factor Authentication (MFA) adds extra layers like codes, tokens, or biometrics. Strong authentication practices include secure password storage, using MFA, monitoring logins, and educating users—together, these measurpyes make accounts much harder to compromise.