PRACTICAL NO. - 01

**AIM -** Program to implement password salting and hashing to create secure password

**THEORY -**

In modern applications, securely storing user passwords is critical to prevent unauthorized access and data breaches. Plain-text password storage is highly insecure and should never be used. Instead, secure password storage mechanisms involve **salting** and **hashing**.

### **1. Password Hashing**

Hashing is the process of converting data (in this case, a password) into a fixed-length string using a cryptographic hash function. Hash functions like SHA-256, SHA-512, or bcrypt are commonly used. These functions are **one-way**, meaning the original password cannot be retrieved from the hash. Even a small change in the input will produce a significantly different output hash.

### **2. Password Salting**

A **salt** is a random string added to the password before hashing. It ensures that even if two users have the same password, their hashes will be different because the salts are different. This **defeats precomputed hash attacks** such as rainbow tables.

Salting involves:

* Generating a unique random salt for each user.
* Concatenating the salt with the user’s password.
* Hashing the combined string.
* Storing both the hash and the salt in the database.

### **3. Secure Hashing Algorithms**

Common secure algorithms include:

* **SHA-256**: A cryptographic hash function that outputs a 256-bit hash.
* **bcrypt**: A slower, adaptive hash function designed specifically for passwords.
* **PBKDF2, Argon2**: Key derivation functions that support salting and iteration counts to increase computation time, enhancing resistance against brute-force attacks.

### **4. Implementation Overview**

To implement secure password storage using salting and hashing:

1. Take the user's password input.
2. Generate a random salt.
3. Combine the password and salt.
4. Hash the result using a secure algorithm.
5. Store both the salt and the hash securely.

**CODE -**

**import bcrypt**

**pwd=input("Enter your password: ")**

**falsePwd=str(input("Enter password: "))**

**bytePwd=pwd.encode('utf-8')**

**print(bytePwd)**

**byteFalsePwd=falsePwd.encode('utf-8')**

**print(byteFalsePwd)**

**#Generate Salt**

**mySalt=bcrypt.gensalt()**

**#Hash password**

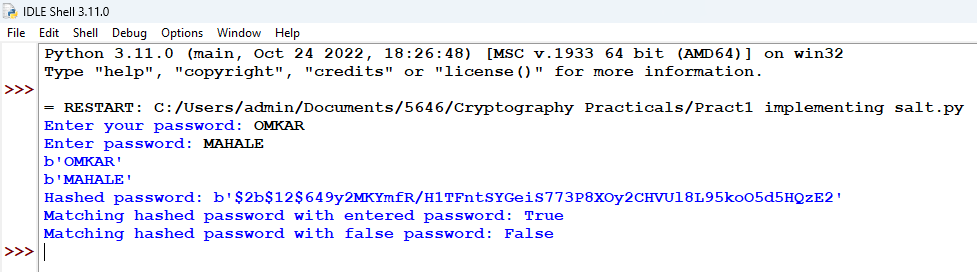
**hash=bcrypt.hashpw(bytePwd,mySalt)**

**print('Hashed password:',hash)**

**print('Matching hashed password with entered password:',bcrypt.checkpw(bytePwd,hash))**

**print('Matching hashed password with false password:',bcrypt.checkpw(byteFalsePwd,hash))**

**OUTPUT -**

****