جامعة الشعب

قسم : هندسة الامن السيبراني

الاسم : اباذر محمد علي مزهر

الاسم : زينب عدنان حنون

شعبة / F

Title: Secure Password Hashing using SHA-256 Algorithm

Abstract

This project focuses on the secure hashing of passwords using the SHA-256 algorithm. The primary objective is to demonstrate the implementation of a secure hash function in Python to ensure the confidentiality and integrity of password data. The outcomes include a Python script capable of generating SHA-256 hashes and a comprehensive understanding of the hashing process and its importance in data security.

1. Introduction

The selected topic addresses the need for secure password storage in applications. Passwords are sensitive data and must be protected against unauthorized access and attacks such as brute force or rainbow table attacks. Hashing algorithms, like SHA-256, convert passwords into fixed-size strings that are difficult to reverse, thus ensuring their security. This project aims to implement SHA-256 hashing in Python and explain its components and logic.

2. Implementation

2.1 Pseudocode Explanation

The Python script for hashing a password using the SHA-256 algorithm includes several key components:

1. **Input Handling**: Accept the password string from the user.
2. **Hashing Process**: Apply the SHA-256 hash function to the input string.
3. **Output**: Display the hashed password.

Pseudocode:

```
```

START

Input: User enters a password string

Process:

1. Import hashlib library

2. Encode the password string to a byte format

3. Apply SHA-256 hash function to the encoded password

4. Convert the hash object to a hexadecimal string

Output: Display the hashed password

END

```

**Explanation of Algorithms and Logic**:

- **Encoding**: Converts the input string to a byte format required by the hash function.

- **Hashing**: Utilizes the SHA-256 algorithm from the hashlib library to generate the hash.

- **Hexadecimal Conversion**: Converts the hash object to a readable hexadecimal format.

### 2.2 Flowchart

Below is the flowchart of the password hashing process:

1. Start

2. Input password string

3. Encode password to bytes

4. Apply SHA-256 hash function

5. Convert hash to hexadecimal
```

6. Output hashed password

7. End

```plaintext
   +-------------+
   |   Start     |
   +------+------+
          |
          v
   +------+------+
   | Input       |
   | Password    |
   +------+------+
          |
          v
   +------+------+
   | Encode      |
   | to Bytes    |
   +------+------+
          |
          v
   +------+------+
   | Apply SHA-256|
   | Hash Function|
```

```
   +------+------+
          |
          v
   +------+------+
   | Convert to   |
   | Hexadecimal  |
   +------+------+
          |
          v
   +------+------+
   | Output       |
   | Hashed       |
   | Password     |
   +------+------+
          |
          v
   +-------------+
   |    End      |
   +-------------+
```

3. Conclusion

- **Achievements and Outcomes**: Successfully implemented a Python script to hash passwords using the SHA-256 algorithm. The project highlighted the importance of secure password storage and the effectiveness of hashing algorithms.

- **Reflection**: The project development process involved understanding the principles of cryptographic hashing, learning to use the hashlib library, and implementing secure coding practices.

- **Future Improvements**: Future enhancements could include the implementation of salting techniques to further enhance security and exploring other hashing algorithms like bcrypt for better resistance to brute-force attacks.

References

- [Python hashlib documentation](https://docs.python.org/3/library/hashlib.html)

- [National Institute of Standards and Technology (NIST) SHA-256 Algorithm Description](https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf)

- [OWASP Cryptographic Storage Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html)

```python
import hashlib

def hash_password(password: str) -> str:
    # Encode the password to bytes
    encoded_password = password.encode('utf-8')

    # Apply SHA-256 hash function
    sha256_hash = hashlib.sha256(encoded_password)

    # Convert the hash object to a hexadecimal string
    hashed_password = sha256_hash.hexdigest()
```

```
    return hashed_password


# Sample usage

password = input("Enter the password: ")

hashed_password = hash_password(password)

print("Hashed Password:", hashed_password)
```

This code snippet accepts a password input from the user, applies the SHA-256 hashing algorithm, and prints the hashed password.