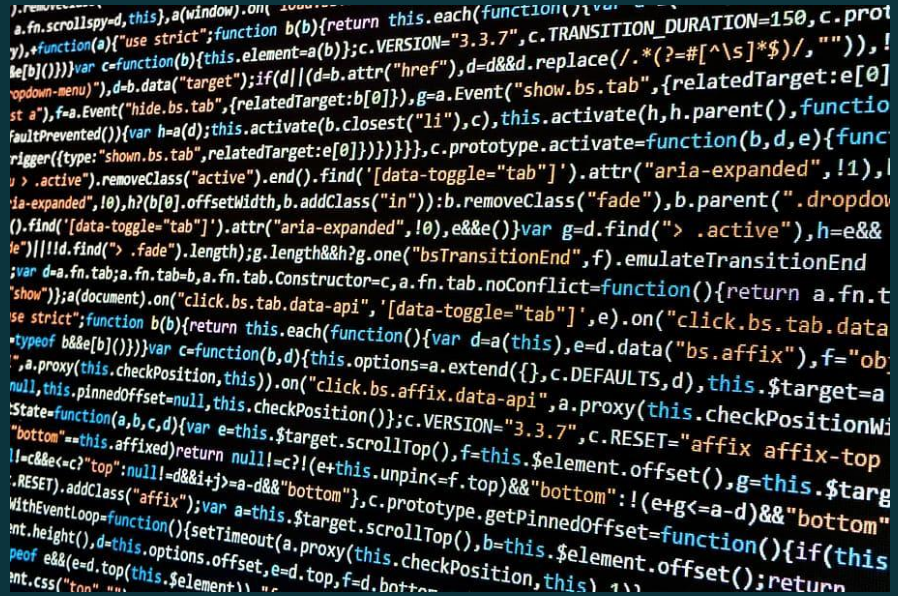
Part 2 code snippets

Password Vault with Master Key Security to securely store and retrieve passwords.

By: Zeina.ElSharkawy

Part B

Screenshots of the programs



Part B.1: Master Key Creation and Hashing

1) User Interface for Master Key Creation

- Code a prompt for the user to create a master key on first use.

2) Implement Hashing Function

- Hash the master key using a secure algorithm, optionally adding salt.
- Use the SHA-256 algorithm for hashing

3 Secure Storage of Hashed Master Key

- Store the hashed key securely, separate from the main password database.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

```
PS D:\Zeina's\project> python Masterkey_creationandHashing.py
Welcome! Please create your master key.
Create your master key (minimum 12 characters): zZ_3_zZ%
Password must be at least 12 characters and contain uppercase, lowercase, numbers and special characters
Create your master key (minimum 12 characters): zZ_3_zZ%lvl4
Master key created and stored successfully!
PS D:\Zeina's\project> python Masterkey_creationandHashing.py
Master key already exists! Please use your existing master key.
PS D:\Zeina's\project>
```

Part B.1: Master Key Creation and Hashing

1) User Interface for Master Key Creation

- Code a prompt for the user to create a master key on first use.

2) Implement Hashing Function

- Hash the master key using a secure algorithm, optionally adding salt.
- Use the SHA-256 algorithm for hashing

3 Secure Storage of Hashed Master Key

- Store the hashed key securely, separate from the main password database.

```python
def hash_master_key(master_key):
    #Function returns the salt and hash separated by a '$'
    salt = secrets.token_hex(16)
    key_with_salt = (master_key + salt).encode('utf-8')
    hashed_key = hashlib.sha256(key_with_salt).hexdigest()
    return f"{salt}${hashed_key}"

def store_master_key(hashed_key):
    with open("master_key.hash", "w") as file:
        file.write(hashed_key)
```

Part B.2: Storing Passwords

1) Take User Input

- Implement python code to receive input - domain,username, and password.

2) Password Encryption

- Encrypt passwords using a unique key for each entry.

3) Password Hashing and Storage

- Hash the encrypted password and store all details securely in a password file.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

```
PS D:\Zeina's\project> python B2_program.py
Enter domain (e.g. https://www.example.com): https://www.google.com
Enter username: Zeina
Enter password: Zz_3_zZ@3
Non-compliant:
Password must contain at least 12 characters.
Enter password: Zz_3_zZ@1234
Password stored successfully!
PS D:\Zeina's\project>
```

Part B.2: Storing Passwords

1) Take User Input

➔ Implement python code to receive input - domain,username, and password.

2) Password Encryption

➔ Encrypt passwords using a unique key for each entry.

3) Password Hashing and Storage

➔ Hash the encrypted password and store all details securely in a password file.

```python
def encrypt_password(password):
    # Function encrypts password using Fernet symmetric encryption
    try:
        key = Fernet.generate_key()  # Secure random key
        f = Fernet(key)
        encrypted_password = f.encrypt(password.encode())
        return key.decode(), encrypted_password.decode()
    except Exception as e:
        print("Encryption error:", e)
        raise

def hash_password(password):
    # Function creates SHA-256 hash of password
    try:
        return hashlib.sha256(password.encode()).hexdigest()
    except Exception as e:
        print("Hashing error:", e)
        raise
```

```python
def store_password(domain, username, key, encrypted_pass, pass_hash):
    # Function stores password entry in password file
    try:
        entry = f"{domain}:{username}:{key}:{encrypted_pass}:{pass_hash}\n"
        with open("passwords.txt", "a") as f:
            f.write(entry)
        print("Password stored successfully!")
    except Exception as e:
        print("Error storing password:", e)
        raise
```

Part B.2: Storing Passwords

1) Take User Input

➔ Implement python code to receive input - domain,username, and password.

2) Password Encryption

➔ Encrypt passwords using a unique key for each entry.

3) Password Hashing and Storage

➔ Hash the encrypted password and store all details securely in a password file.

passwords.txt

https://www.google.com:Zeina:_qS9khnax-MmwVyEUyAt78IbTRzBI9GCfB3kOcYicIg=:gAAAAABnwr8y5z84wUjO8p7K-XEaTTtYxyeR1WT6xXeaShUTSuiZL0_fr4tXd25ys9dk7SwFrSUQzpBtIT-SLZ
https://www.udacity.com:Zeina:1c3icPEWVr0j2w1R8M4-YJmIS1LklP68f6upuFFpMg0=:gAAAAABnwtfVoqUYZZ18DBtSQXT5l-OzercixcyDj5w7MlZoFMRrhv4GDkt207vFnokNKAq_JKn0-BddH2hIr

Part B.3: Retrieving Passwords

1) Implement a function

- Prompt the user for the master key upon request to retrieve a password.
- Hash the input and compare it with the stored master key hash for authentication.

2) Upon successful authentication

- Search the password file for the requested domain.
- Decrypt the password using the stored encryption key.

3) Password Hashing and Storage

- Display the decrypted password to the user

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

```
PS D:\Zeina's\project> python B3_program.py
Enter master key: Zz_3_zZ@1234
Invalid master key. 2 attempts remaining.
Enter master key: zZ_3_zZ%1vl4
Enter domain to search: https://www.google.com

Found credentials:
Domain:   https://www.google.com
Username: Zeina
Password: Zz_3_zZ@1234
PS D:\Zeina's\project>
```

Part B.4: Shell scripting

1)

- a shell script that prompts the user for a domain name and then executes a Python script to retrieve a password for that domain.

2)

- a shell script that authenticates the master key by calling the Python script written in the first part of the project, that performs the actual authentication process.

Retrieving an existing password command:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Mohamed@DESKTOP-3C0KC3S MINGW64 /d/Zeina's/project
$ ./B4_program.sh
Welcome to the Password Manager
Please choose an option:
1. Add a new password
2. Retrieve an existing password
3. Exit
Enter your choice [1-3]: 2
Enter the domain name for which to retrieve the password: https://www.google.com
Retrieving password for https://www.google.com...
Enter master key: zZ_3_zZ@1234
Invalid master key. 2 attempts remaining.
Enter master key: Zz_3_zZ@1234
Invalid master key. 1 attempts remaining.
Enter master key: zZ_3_zZ%lvl4
Enter domain to search: https://www.google.com

Found credentials:
Domain:   https://www.google.com
Username: Zeina
Password: Zz_3_zZ@1234
Password retrieval completed.
```

Part B.4: Shell scripting

1)

- a shell script that prompts the user for a domain name and then executes a Python script to retrieve a password for that domain.

2)

- a shell script that authenticates the master key by calling the Python script written in the first part of the project, that performs the actual authentication process.

Adding a new password command:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Mohamed@DESKTOP-3C0KC3S MINGW64 /d/Zeina's/project
$ ./B4_program.sh
Welcome to the Password Manager
Please choose an option:
1. Add a new password
2. Retrieve an existing password
3. Exit
Enter your choice [1-3]: 1
Starting the process to add a new password...
Enter domain (e.g. https://www.example.com): https://www.udacity.com
Enter username: Zeina
Enter password: haha
Non-compliant:
Password must contain at least 12 characters.
Password must contain at least 1 uppercase character.
Password must contain at least 1 digit.
Password must contain at least 1 special character.
Enter password: Zaserfdc3!
Non-compliant:
Password must contain at least 12 characters.
Enter password: Zaswerdfxcv!!3
Password stored successfully!
New password entry added successfully.
```

Part B.4: Shell scripting

1)

- a shell script that prompts the user for a domain name and then executes a Python script to retrieve a password for that domain.

2)

- a shell script that authenticates the master key by calling the Python script written in the first part of the project, that performs the actual authentication process.

Exiting command:

```
Mohamed@DESKTOP-3C0KC3S MINGW64 /d/Zeina's/project
$ ./B4_program.sh
Welcome to the Password Manager
Please choose an option:
1. Add a new password
2. Retrieve an existing password
3. Exit
Enter your choice [1-3]: 3
Exiting.
```