

# **Project Title:** Secure Local Password Manager Using Python

**Name :** S.Krishna Kumar

**Platform / Context:** Cybersecurity / Secure Software Development Project (Kali Linux)

---

## **1. Project Overview**

This project is a **local password manager** developed using **Python 3 on Kali Linux**. It allows users to securely store, retrieve, delete, and search credentials while ensuring that all sensitive information is **encrypted before being saved to disk**.

The application uses **strong symmetric encryption (AES)** and protects all data using a **master password**.

No data is stored in plaintext, and no cloud services are used.

## **2. Project Objectives**

- Securely store passwords on a **Kali Linux system**
- Prevent plaintext credential storage
- Use strong encryption techniques
- Implement basic password management operations
- Protect access using a master password

## **3. Project Requirements**

The project satisfies the following requirements:

- Encrypted local storage of credentials
- Strong symmetric encryption (AES)
- Secure key handling
- Add, retrieve, delete, and search password entries
- Master password authentication
- Secure local storage format (encrypted JSON)

## **4. Tools and Technologies Used**

<b>Component</b>	<b>Description</b>
Operating System	<b>Kali Linux</b>
Programming Language	Python 3

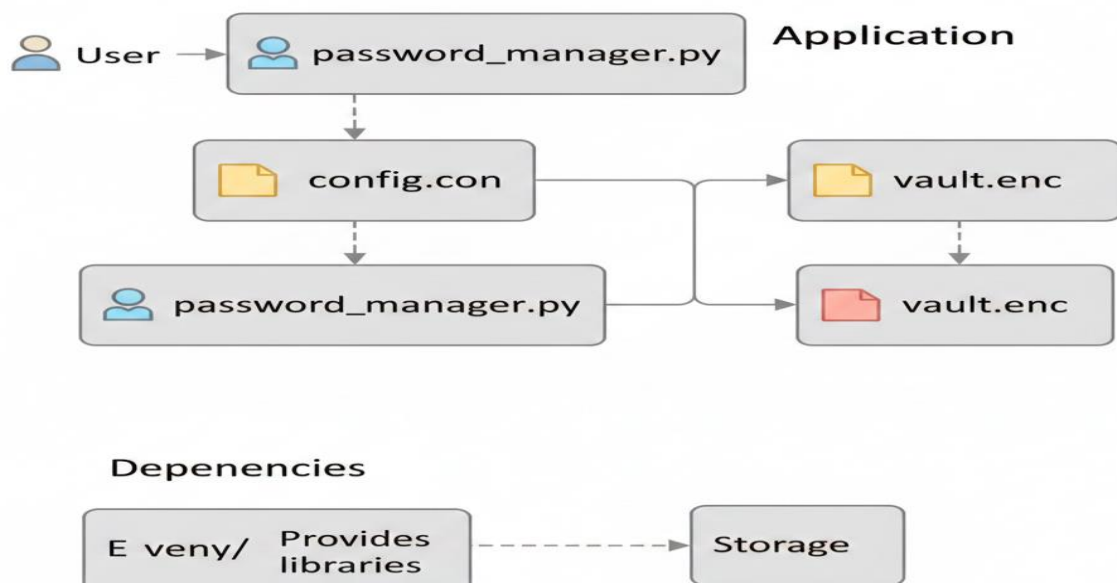
Component	Description
Encryption	AES (Fernet)
Key Derivation	PBKDF2
Library	cryptography
Storage	Encrypted file (vault.enc)
Environment	Python Virtual Environment (venv)

## 5. Project Directory Structure

password\_manager/

- password\_manager.py → Main application
- config.json → Encryption configuration
- vault.enc → Encrypted password vault
- venv/ → Python virtual environment

### Project Setup (Kali Linux)



## 6. System Design and Security

### 6.1 Master Password

- Created during first execution
- Required every time the application runs
- Never stored on disk

## 6.2 Encryption and Key Handling

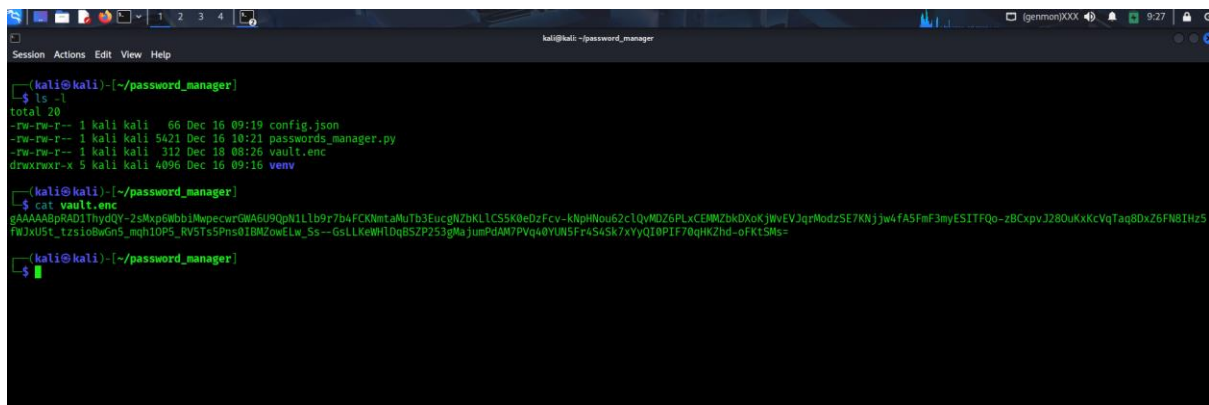
- AES encryption implemented using Fernet
- Encryption key derived from master password using PBKDF2
- Key exists only in memory during program execution

## 7. Secure Storage Format

- Credentials are stored as JSON in memory
- JSON data is encrypted before saving
- Disk contains only encrypted data

vault.enc → Encrypted and unreadable file

### Encrypted Vault (Kali Linux)



```
(kali@kali)~/password_manager
$ ls -l
total 20
-rw-rw-r-- 1 kali kali 66 Dec 16 09:19 config.json
-rw-rw-r-- 1 kali kali 5421 Dec 16 10:21 passwords_manager.py
-rw-rw-r-- 1 kali kali 312 Dec 18 08:26 vault.enc
drwxrwxr-x 5 kali kali 4096 Dec 16 09:16 venv

(kali@kali)~/password_manager
$ cat vault.enc
gAAAAABpRAD1ThydQY-2sMxp6WbbiMwpecwrGWA6U9QpN1L1b9r7b4FCKNmtaMuTb3EucgNZbKLLCS5K0eDzFcv-kNpHwou62clQvMDZ6PLxCENMZbKDXoKJWVEVJqrModzSE7KNjjw4fA5FmF3myESITFQo-z8CxpVJ280uKxKcVqTaq8DxZ6FN8IHZ5
FWJxU5t_tzsi0BwGn5_mqh1OP5_RVSTs5Pns0IBMZowELw_Ss--GsLLKeWfLDqBSZP253gMajumPdAM7PVq40YUN5Fp4S4Sk7xYyQ10PIF70qHKZhd-oFKt5Ms=

(kali@kali)~/password_manager
$
```

## 8. Features Implemented

Feature	Description
Add Password	Save credentials securely
Retrieve Password	View stored credentials
Delete Password	Remove credentials
Search Password	Search by keyword
Auto Lock	Vault encrypted on exit

## 9. Program Execution Flow

1. User runs the program in Kali Linux terminal
2. Master password is requested
3. Encryption key is generated
4. Vault is decrypted in memory
5. User performs operations
6. Vault is re-encrypted
7. Program exits securely

### Program Execution

```
Session  Actions  Edit  View  Help
(kali㉿kali)-[~]
$ cd password_manager

(kali㉿kali)-[~/password_manager]
$ ls
config.json  passwords_manager.py  vault.enc  venv

(kali㉿kali)-[~/password_manager]
$ python3 passwords_manager.py
Enter master password:

===== PASSWORD MANAGER =====
1. Add new password
2. Retrieve password
3. Delete password
4. Search entries
5. Show all entries (table)
6. Exit

Choose an option: █
```

```
(kali㉿kali)-[~/password_manager]
$ python3 passwords_manager.py
Enter master password:

===== PASSWORD MANAGER =====
1. Add new password
2. Retrieve password
3. Delete password
4. Search entries
5. Show all entries (table)
6. Exit

Choose an option: 1
Website/App name: cybersecurity
Username: cyberwarriors
Password:
✓ Entry added successfully.
```

## 10. File Description

### password\_manager.py

- Handles user input
- Performs encryption and decryption
- Implements all password operations

### config.json

- Stores salt and encryption parameters
- Contains no passwords

### vault.enc

- Stores encrypted credentials
- Cannot be accessed without master password

```
===== PASSWORD MANAGER =====
1. Add new password
2. Retrieve password
3. Delete password
4. Search entries
5. Show all entries (table)
6. Exit

Choose an option: 5

Stored Passwords (Table View):

Website                Username                Password
-----
Facebook               kumar@12               *****
Instagram              Premkumar_s            *****
LinkedIn               Kumar                  *****
Snapchat               Helin                   *****
Twitter                Cluster                 *****

===== PASSWORD MANAGER =====
1. Add new password
2. Retrieve password
3. Delete password
4. Search entries
5. Show all entries (table)
6. Exit

Choose an option: 6
🔒 Vault locked. Goodbye.
```

## 11. Testing and Validation

- Tested on Kali Linux terminal
- Verified encrypted vault integrity
- Tested correct and incorrect master passwords
- Confirmed add, delete, and search operations

## Security Testing

```
🔒 Vault locked. Goodbye.

(kali@kali)-[~/password_manager]
$ python3 passwords_manager.py
Enter master password:
✖ Incorrect master password or corrupted vault.

(kali@kali)-[~/password_manager]
$ █
```

## 12. Advantages

- Strong encryption ensures data security
- Local storage provides full control
- No plaintext password storage
- Works fully on Kali Linux

## 13. GitHub Repository

The complete source code for this project is hosted on GitHub.

The repository includes:

- Full source code (password\_manager.py)
- Encrypted vault file example
- Configuration file
- Documentation and screenshots
- Instructions to run the project on **Kali Linux**

### Repository Link

[https://github.com/Premkumar-pro/Syntecxhub\\_TCPportscanning\\_using\\_python/blob/main/syntecxhub\\_Password\\_manager.py](https://github.com/Premkumar-pro/Syntecxhub_TCPportscanning_using_python/blob/main/syntecxhub_Password_manager.py)

## 14. Conclusion

This project successfully implements a **secure local password manager on Kali Linux** using Python. By applying strong encryption, secure key handling, and master password protection, the system ensures safe storage and management of sensitive credentials.