

Password House: A Lightweight Command-Line Password Management System

Ms. Sure Mamatha¹, Riya Chaurasiya², Nipuna Shyamala³, Pedduri Chandu⁴,
Nagavarshith Yadav⁵

¹Assistant Professor, Dept. of Computer Science and Engineering, HITAM, Telangana, India

²³⁴⁵Student, Dept. of Computer Science and Engineering, HITAM, Telangana, India

Abstract - Password managers are increasingly used to store and manage multiple credentials, but they must also address risks related to credential exposure, weak passwords, and poor password management practices. Password House is a lightweight command-line password management system developed for Kali Linux using Python and Bash. The system uses a master password for authentication and Fernet-based encryption to protect stored credentials. It provides password generation, password storage and retrieval, password strength checking, and password expiry handling. A default 90-day expiry period is included to encourage regular credential maintenance. Recent research has identified security risks in password managers, including memory exposure and injection-based leakage, while other studies have examined weak and reused password practices. Password House combines secure local credential storage with features that encourage stronger password practices in a simple CLI environment. Functional testing of the implemented modules verified authentication, encryption and decryption, password generation, credential retrieval, strength feedback, and expiry handling.

Keywords - password manager, password security, Kali Linux, Fernet encryption, SHA-256, password generation, password strength, command-line interface

1. INTRODUCTION

The increasing use of online services has resulted in users managing a large number of accounts and passwords. Remembering multiple passwords can lead to password reuse, weak credential selection, or insecure storage practices. Password managers provide a practical solution by allowing multiple credentials to be stored and managed through a single master password. However, password managers themselves must protect sensitive credentials against exposure and unauthorized access. Therefore, an effective password management system should combine secure credential storage with features that encourage stronger password practices.

Password House is a lightweight command-line password management system developed using Python and Bash for a Kali Linux environment. The system provides user

authentication, secure credential storage, password generation, password retrieval, password strength checking, and password expiry handling. Fernet-based encryption is used to protect stored credentials, while the additional password-management features support stronger credential practices.

1.1 Background and Password Management

Password management involves creating, storing, retrieving, and maintaining credentials for different online services. As the number of accounts increases, maintaining unique passwords becomes difficult for users. Password managers address this problem by storing multiple credentials within a protected system, reducing the need to remember each password individually. Since these systems handle sensitive information, secure storage and controlled access are important requirements. Password House uses Fernet symmetric encryption for stored password data and a master password for access to the password vault. It also provides password generation, strength checking, and expiry handling to support credential maintenance.

1.2 Need for a Secure Password Management System

Manual password management can result in weak password selection, reuse, or insecure storage. Although password managers reduce these difficulties, the security of the password manager itself is important because unauthorized access could expose multiple credentials. Password House addresses this need through master-password authentication and encrypted local storage, while its password generator and strength checker help users create stronger credentials. The 90-day expiry mechanism further encourages periodic credential review. The system is designed as a lightweight command-line application for Kali Linux, providing essential password-management functions without requiring complex external infrastructure.

2. LITERATURE REVIEW

Password management has become an important area of cybersecurity research because users increasingly manage multiple credentials for different online services. Recent

studies have examined password manager security, credential leakage, password generation, password strength assessment, and user behavior. These studies show that secure storage must be combined with appropriate protection against leakage and with features that encourage stronger password practices.

2.1 Password Manager Security

Password managers store sensitive credentials and therefore require protection against unauthorized access and information leakage. Research has examined the exposure of password data through memory analysis and other attack techniques [1], [6]. These findings indicate that protecting credentials only within the storage file may not be sufficient, highlighting the importance of secure encryption and careful credential handling [7], [12].

2.2 Credential Leakage and Password Manager Vulnerabilities

Password managers can be affected by vulnerabilities in their application design and interaction mechanisms. Studies have demonstrated that memory exposure and injection-based attacks can potentially reveal sensitive password information [1], [2], [12]. These findings emphasize the need for controlled authentication, secure credential handling, and minimized exposure of stored information.

2.3 Secure Password Generation

Password generation is important because manually selected passwords may be predictable or reused. Research has examined the security and reliability of password-generation algorithms and their intended security properties [3]. Studies have also investigated whether password managers encourage users to adopt randomly generated passwords [9]. These findings support the inclusion of a password generator in Password House.

2.4 Password Strength and Credential Assessment

Password strength assessment helps users identify credentials with weak characteristics. Recent research has investigated password strength detection using measurable characteristics and machine-learning-based approaches [4]. Password manager checkup tools have also been studied for their ability to assess credential security [8]. Password House provides a simpler strength-checking mechanism that gives immediate feedback during password creation.

2.5 Password Hygiene and User Behavior

The effectiveness of password managers also depends on how users manage their credentials. Research has examined password reuse, password-manager adoption, and whether password managers encourage stronger password practices [5], [9]–[11]. These studies indicate that secure storage alone does not necessarily ensure good password hygiene. Password House therefore combines encrypted storage with password generation, strength checking, and expiry handling.

2.6 Research Gap

Existing research has addressed password manager security, credential leakage, password generation, password strength assessment, and user behavior. However, these areas are often investigated separately, creating a need for a lightweight system that integrates multiple password-management functions. Password House addresses this gap by combining master-password authentication, encrypted local storage, password generation, strength checking, and expiry handling within a single command-line application for Kali Linux.

3. METHODOLOGY

The methodology followed for developing Password House focuses on secure credential management, password protection, and command-line interaction. The system is implemented using Python with Bash-based installation and setup support in a Kali Linux environment. The development process includes requirement analysis, system design, authentication, encryption and secure storage, password management, CLI implementation, and testing.

3.1 Requirement Analysis

The system requirements were identified based on the need to securely manage multiple credentials through a lightweight local application. The analysis focused on both the functional and security requirements of the proposed password manager. The main functional requirements include user registration, master-password authentication, credential storage and retrieval, password generation, password strength checking, password updating and deletion, and password expiry handling. The system should allow users to perform these operations through a simple command-line interface without requiring complex external services or infrastructure. The password generator is intended to assist users in creating stronger credentials, while the strength checker provides immediate feedback about password characteristics. The expiry mechanism is included to encourage users to periodically review and update stored credentials.

Security requirements were defined to ensure that sensitive password information is not maintained as directly readable plaintext within the local vault. Access to password-management operations must be restricted to authenticated users through the master password. Stored credentials must be protected using encryption, while the corresponding decryption process should only be available after successful authentication. The system also requires local storage suitable for a Kali Linux environment and Bash-based installation support for setting up the application. These requirements formed the basis for the subsequent system design, authentication, encryption, password-management, and testing stages.

Table 2: Software and Technology Requirements

Component	Technology / Tool	Purpose
Operating System	Kali Linux	Development and execution environment
Programming Language	Python	Core application implementation
Installation	Bash	Installation and setup
Encryption	Fernet	Credential encryption and decryption
Hashing / Key Processing	SHA-256	Password-related processing and key derivation
Interface	Command Line	User interaction
Credential Storage	Local Vault	Storage of encrypted credentials

3.2 System Design

Password House follows a modular design consisting of authentication, encryption, password-management, and local-storage components. Users interact with the system through the command-line interface, and authentication is performed before access to stored credentials is provided. After successful authentication, users can add, retrieve, update, or delete credentials and use the password-generation and strength-checking functions. Sensitive credential information is encrypted using Fernet before being stored locally, while authorized retrieval uses the corresponding decryption process.



Fig. 1: Overall Password House System Workflow

The system uses local storage to maintain the required user and credential information. Sensitive password data is protected using Fernet encryption before being stored. During retrieval, the encrypted data is decrypted only after the required authentication process has been completed. This design keeps the system lightweight while maintaining a clear separation between authentication, credential protection, and password-management operations.

3.3 User Authentication and Key Generation

User authentication is performed using a master password that controls access to the password vault. During registration, the user provides the required account information and master password, while subsequent login attempts verify the entered credentials before granting access. The master-password-based information is also used to derive the key required by the Fernet encryption mechanism. SHA-256 is included in the implementation for password-related processing and key derivation, providing the information required to obtain the key format used by Fernet. The same derived key is required for authorized decryption of stored credentials.

3.4 Password Encryption and Secure Storage

Password House uses Fernet symmetric encryption to protect credentials before local storage. When a credential is added, the password information is encrypted using the derived key and the resulting encrypted value is stored in the password vault. Fernet also provides authenticated encryption, helping protect the stored data against

unauthorized modification. During retrieval, the encrypted value is decrypted using the appropriate key after successful authentication, allowing the original credential to be recovered without maintaining it as readable plaintext in storage.

3.5 Password Management

The system provides command-line operations for adding, retrieving, updating, and removing stored credentials. A password generator creates credentials using supported character types, reducing dependence on predictable manually selected passwords. The strength checker evaluates password characteristics such as length and character composition and provides immediate feedback. Password House also uses a default 90-day expiry period to encourage users to review and update stored credentials periodically.

3.6 Bash Integration and Command-Line Implementation

Python handles the primary application functionality, while Bash supports installation and setup in the Kali Linux environment. The command-line interface provides access to authentication, password generation, credential management, and other system functions through terminal commands and prompts. This approach keeps the application lightweight while providing direct interaction with the password-management functions.

3.7 Testing and Validation

Testing was performed to verify the functional and security-related behavior of the implemented modules. Authentication was tested using valid and incorrect master-password inputs, while encryption and decryption were checked during credential storage and retrieval. Password generation, credential management, strength checking, and expiry handling were also tested using appropriate inputs. The testing process additionally verified that stored credentials were protected through encryption rather than maintained as directly readable plaintext. The results were used to confirm that the major Password House components operate together as intended.

4. RESULTS

The developed Password House system was evaluated to verify the functionality of its major modules. Testing focused on authentication, password generation, encryption and decryption, credential storage and retrieval, password strength checking, password management, and password expiry. Valid and invalid inputs were used to verify whether each implemented function produced the expected behavior. The results

confirmed that the major components operate together through the command-line interface and that stored credentials are protected using the implemented encryption mechanism.

4.1 Functional Testing

Functional testing was performed on the main operations of Password House. User registration and login were tested using valid and incorrect authentication information. Password generation and strength checking were evaluated using different password inputs, while credential addition, retrieval, updating, and deletion were tested through the command-line interface. The password expiry mechanism was also checked using the configured 90-day period. The tested functions produced the expected results for the implemented use cases.

Table 1: Functional Test Cases and Results

Test Case	Input/Action	Expected Result	Result
User Registration	Valid user details	Account created	Pass
Login	Correct master password	Access granted	Pass
Invalid Login	Incorrect master password	Access denied	Pass
Password Generation	Generate password	Password generated	Pass
Credential Storage	Add credential	Encrypted credential stored	Pass
Credential Retrieval	Valid authentication	Original credential retrieved	Pass
Password Strength	Weak/strong inputs	Strength feedback displayed	Pass
Credential Update/Delete	Modify or remove entry	Credential updated/deleted	Pass
Password Expiry	90-day expiry condition	Credential flagged for review	Pass

4.2 Security and Feature Results

The authentication mechanism restricts access to password-management operations until successful verification. Credentials are encrypted using Fernet before local storage and can be recovered through authorized

decryption. The password generator provides credentials using supported character types, while the strength checker provides immediate feedback based on the implemented password characteristics. The expiry mechanism identifies credentials requiring periodic review, supporting continued password maintenance.



Fig. 2. Password House Command-Line Interface

4.3 Overall Evaluation

The overall evaluation indicates that Password House integrates authentication, encrypted credential storage, password generation, strength checking, credential management, and expiry handling within a single command-line application. The implemented functions operate together as intended and provide the core features required for local password management in the Kali Linux environment.

5. DISCUSSION

The results demonstrate that Password House successfully integrates the major functions required for local password management, including authentication, encrypted credential storage, password generation, strength checking, credential management, and expiry handling. The functional tests verified the expected behavior of these modules under the implemented use cases. The results also show how the individual components work together rather than operating as isolated features. Authentication controls access to the vault, encryption protects stored credentials, and password-management features support the creation and maintenance of credentials. This integrated approach is consistent with the need identified in existing research for password-management systems to address both credential protection and password-management behavior.

5.1 Security and Usability Discussion

Security and usability are important considerations in password management because a system must protect sensitive credentials while remaining practical for users.

Password House combines master-password-based authentication with Fernet encryption to provide controlled access to locally stored credentials. The encryption mechanism ensures that credentials are not maintained directly as readable password values within the vault, while authorized retrieval allows the original credentials to be recovered after successful authentication. This provides a clear separation between credential storage and credential access.

The system also incorporates features intended to support better password practices. The password generator reduces dependence on manually selected credentials, while the password strength checker provides immediate feedback during password creation. The 90-day expiry mechanism provides an additional reminder for users to review and update stored credentials. These features complement the encryption mechanism by addressing password creation and maintenance rather than focusing only on storage security.

From a usability perspective, the command-line interface keeps the system lightweight and provides direct access to the implemented functions. The use of Python for the main application and Bash for installation and setup makes the system suitable for the intended Kali Linux environment. However, the CLI-based approach may require users to be familiar with terminal interaction. In addition, the overall security of the system depends on users maintaining a strong master password and protecting the local environment in which the encrypted vault is stored.

5.2 Comparison with Existing Research

Existing research has examined several individual aspects of password management, including password manager security, credential leakage, password generation, password strength assessment, and user behavior [1]–[12]. Studies investigating memory exposure and injection-based attacks have demonstrated that password managers can face risks beyond the security of their stored files [1], [2], [12]. Research on password generation has examined the security properties of generation algorithms and the use of randomly generated credentials [3], [9]. Other studies have investigated password strength assessment and password backup mechanisms for identifying weak credentials [4], [8].

Research has also considered the behavioral aspects of password management. Studies have examined password reuse, password-manager adoption, user acceptance, and whether password managers encourage stronger password practices [5], [9]–[11]. These findings indicate that providing secure storage alone does not address every aspect of password management. Password House incorporates several of these areas within a single system by combining encrypted local storage with password

generation, strength checking, credential management, and expiry handling.

The proposed system therefore represents an integrated implementation of commonly studied password-management requirements within a lightweight command-line environment. Rather than focusing on a single security mechanism, Password House combines credential protection with features intended to support password creation and maintenance. This integration forms the main practical contribution of the project.

5.3 Limitations

Despite providing the core functionality required for local password management, Password House has several limitations. The current implementation is primarily designed for a single local environment and does not provide secure cross-device synchronization or cloud-based backup. Consequently, users are responsible for protecting the local system and maintaining access to the encrypted vault.

The system also does not currently implement multi-factor authentication. Access is primarily controlled through the master password, which means that the security of the authentication process depends significantly on the strength and confidentiality of that password. In addition, the current command-line interface does not provide the convenience of a graphical interface for users who are less familiar with terminal-based applications.

Another limitation is that the current implementation does not include advanced password-breach monitoring or comprehensive credential auditing. Larger-scale security evaluation could also be performed in future work to examine the system against a broader range of attack scenarios. These limitations provide clear directions for extending Password House while retaining its lightweight local architecture.

6. CONCLUSION

Password House presents a lightweight command-line password management system designed for a Kali Linux environment. The system was developed using Python with Bash-based installation and setup support and integrates authentication, encrypted credential storage, password generation, password strength checking, credential management, and password expiry within a single application.

The implemented authentication mechanism controls access to the password vault, while Fernet symmetric encryption protects credentials before they are stored locally. The password generator provides an alternative

to manually creating predictable credentials, and the strength checker provides immediate feedback regarding password characteristics. The credential-management functions allow users to add, retrieve, update, and remove stored entries, while the 90-day expiry mechanism encourages periodic credential review.

Functional testing was performed on the major modules of the system, including registration, authentication, password generation, credential storage and retrieval, strength checking, credential management, and expiry handling. The test results showed that the implemented functions produced the expected behavior for the defined use cases. The results therefore demonstrate that the proposed system can combine secure local credential storage with features that encourage improved password-management practices.

Overall, Password House provides a simple and practical approach to local password management without requiring complex external infrastructure. Although the current implementation has limitations such as the absence of multi-factor authentication, graphical interaction, cross-device synchronization, and advanced credential auditing, it establishes a foundation that can be extended with additional security and usability features.

6.1 Future Scope

Future development of Password House can focus on improving both security and usability. Multi-factor authentication could be incorporated as an additional layer of protection beyond the master password. A graphical user interface could also be developed to make the system more accessible to users who are less comfortable with command-line interaction. Secure backup and cross-device synchronization could be introduced while maintaining protection for the encrypted vault.

Additional security features such as password-breach checking, advanced credential auditing, and stronger password-analysis mechanisms could further improve the system. Future versions could also undergo larger-scale security testing to evaluate resistance against different attack scenarios and identify potential vulnerabilities. These extensions would allow Password House to evolve from a lightweight local CLI application into a more comprehensive password-management platform.

REFERENCES

- 1) E. Chatzoglou, V. Kampourakis, Z. Tsiatsikas, G. Karopoulos, and G. Kambourakis, "Keep Your Memory Dump Shut: Unveiling Data Leaks in Password Managers," in *ICT Systems Security and Privacy Protection, SEC 2024*, Springer, 2024, pp. 61–75.
- [2] A. Fábrega, A. Namavari, R. Agarwal, B.

- Nassi, and T. Ristenpart, "Exploiting Leakage in Password Managers via Injection Attacks," in Proc. 33rd USENIX Security Symposium, 2024, pp. 4337–4354.
- 2) M. Grilo, J. F. Ferreira, and J. B. Almeida, "Towards Formal Verification of Password Generation Algorithms Used in Password Managers," arXiv:2106.03626, 2021.
 - 3) J. Mo, H. Kuang, and X. Li, "Password Strength Detection via Machine Learning: Analysis, Modeling, and Evaluation," arXiv:2505.16439, 2025.
 - 4) D. W. Ng, J. Ho, C. Hercules, C. Bravo-Lillo, and S. Schechter, "Do Password Managers Improve Password Hygiene?," 2022.
 - 5) P. Sabev and M. Petrov, "Android Password Managers and Vault Applications: Data Storage Security Issues Identification," Journal of Information Security and Applications, vol. 67, Art. no. 103152, 2022.
 - 6) A. Nash and K.-K. R. Choo, "Password Managers and Vault Application Security and Forensics: Research Challenges and Future Opportunities," 2024.
 - 7) A. Hutchinson, C. W. Munyendo, A. J. Aviv, and P. Mayer, "An Analysis of Password Managers' Password Checkup Tools," in Extended Abstracts of the CHI Conference on Human Factors in Computing Systems, 2024.
 - 8) S. Zibaei, D. R. Malapaya, B. Mercier, A. Salehi-Abari, and J. Thorpe, "Do Password Managers Nudge Secure (Random) Passwords?," in Proc. 18th Symposium on Usable Privacy and Security (SOUPS), 2022, pp. 581–597.
 - 9) M. Nehme, M. L. Li, and M. Warkentin, "Adaptive and Maladaptive Factors behind Password Manager Use: A Hope-Extended Protection Motivation Perspective," Computers & Security, vol. 144, Art. no. 103941, 2024.
 - 10) H. Jamil, T. Zia, and T. Nayeem, "User Acceptance of Password Manager Software: Evidence from Australian Microbusinesses," 2021. S. Chatzoglou, V. Kampourakis, Z. Tsiatsikas, G. Karopoulos, and G. Kambourakis, "Unmasking the Hidden Credential Leaks in Password Managers and VPN Clients," Computers & Security, vol. 150, Art. no.