PROTECTING USER PASSWORD KEYS AT REST (ON THE DISK)

Submitted By:

I. **Abhishek C.**
II. **Gayatri M.**
III. **Laxmi G.**
IV. **Pallavi B.**
V. **Steffi M.**

Under the Guidance of:

Prof. Govind M.R

Abstract

This project report explores methods for protecting user password keys at rest (on disk). It outlines the threats to stored passwords and analyses various password storage techniques, including hashing, salting, and key derivation functions. The report emphasizes the importance of secure key management practices like Hardware Security Modules (HSMs) and key rotation. Additionally, it highlights the role of strong password policies and Multi-Factor Authentication (MFA) in fortifying overall security. The project concludes by recommending regular security audits, staying updated on evolving threats, and user education on password hygiene.

When it comes to safeguarding user password keys at rest, especially on the disk, it's crucial to employ robust security measures to prevent unauthorized access or theft of this sensitive information. One common method to protect these keys is through encryption.

Encryption involves converting the password keys into a secure format that can only be accessed with the correct decryption key. This way, even if someone gains access to the stored keys on the disk, they won't be able to make sense of the information without the decryption key.

Additionally, it's essential to follow best practices like using strong encryption algorithms, securely managing and storing encryption keys, implementing access controls, and regularly updating security protocols to stay ahead of potential threats.

By implementing these security measures, organizations can significantly reduce the risk of unauthorized access to user password keys stored at rest on the disk.

In addition to encryption, another important aspect of protecting user password keys at rest is implementing secure storage practices. This involves ensuring that the disk where the keys are stored is secure and inaccessible to unauthorized users.

One common practice is to store the keys in a secure vault or key management system that provides strong access controls and auditing capabilities. This way, only authorized personnel can access and manage the keys, reducing the risk of unauthorized access.

Regularly monitoring and auditing access to the stored keys is also crucial to detect any unusual activities or potential security breaches. By keeping a close eye on who accesses the keys and when, organizations can quickly identify and respond to any suspicious behaviour.

Overall, a combination of encryption, secure storage practices, access controls, and monitoring can help effectively protect user password keys at rest on the disk, ensuring the security and integrity of this sensitive information.

Table of Contents

1. Introduction

User passwords are the cornerstone of access control in many systems. However, storing them securely at rest (on disk) is crucial to prevent unauthorized access and data breaches. This report explores different methods for protecting user password keys at rest and discusses their strengths and weaknesses.

When it comes to keeping user password keys safe while they're at rest on the disk, it's super important to have solid security measures in place. Protecting these keys is crucial because they hold sensitive information that should only be accessible to authorized users.

One of the main ways to safeguard these keys is through encryption. Encryption basically scrambles the keys into a format that's unreadable without the right decryption key. This way, even if someone gets their hands on the stored keys, they won't be able to understand them without the proper decryption key.

Alongside encryption, it's also vital to ensure the physical security of the disk where the keys are stored. This means making sure that only authorized personnel can access the disk and that it's protected from unauthorized entry or theft.

By combining encryption with secure storage practices and access controls, organizations can create a strong defense against potential threats and unauthorized access to user password keys at rest on the disk.

In addition to encryption and secure storage practices, it's essential to implement access controls to further protect user password keys at rest on the disk. Access controls help regulate who can view, modify, or manage the stored keys, reducing the risk of unauthorized access.

Access controls can include mechanisms like user authentication, role-based access control (RBAC), and multi-factor authentication (MFA). User authentication ensures that only authorized users can access the keys, while RBAC allows organizations to define specific roles and permissions for different users based on their responsibilities.

Moreover, implementing MFA adds an extra layer of security by requiring users to provide multiple forms of verification before accessing the password keys. This can include something they know (like a password) and something they have (like a security token or biometric data).

By combining encryption, secure storage practices, and access controls, organizations can create a robust security framework to protect user password keys at rest on the disk effectively. This comprehensive approach helps mitigate the risk of unauthorized access and ensures the confidentiality and integrity of sensitive information.

2. Threats to Password Keys at Rest

- **Data Breaches:**

Hackers can gain access to the server where passwords are stored and steal them if not adequately protected.

- **Insider Threats:**

Malicious insiders with authorized access to the system could steal password keys.

- **Accidental Exposure:**

System administrators or software vulnerabilities could inadvertently expose password keys.

3. Password Storage methods

- **Plaintext Storage (Never Do This):**

Storing passwords in plain text is highly insecure and should never be done.

- **Hashing:**

Passwords are converted to a fixed-length string of characters using a one-way hashing function. This makes it impossible to recover the original password, but attackers could use rainbow tables to reverse some hashes.

- **Salting:**

A random string (salt) is added to the password before hashing. This prevents pre-computed attacks (rainbow tables) as the same password with different salts will generate different hashes.

- **PBKDF2 (Password-Based Key Derivation Function):**

A slow hashing function that makes brute-force attacks more difficult by taking longer to compute the hash.

- **Encryption:**

Passwords are encrypted with a strong encryption key before storing them. This requires secure storage of the encryption key itself.

4. Key Encryption

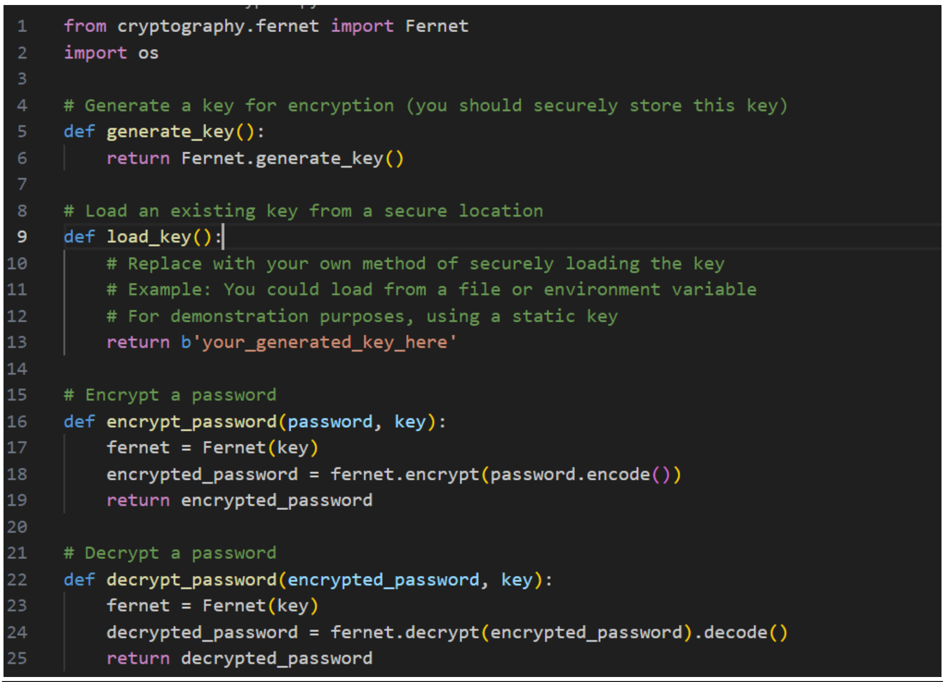
- **Master Key:**

A single master key encrypts all password keys. This creates a single point of failure if the master key is compromised.

- **Key Hierarchy:**

A hierarchy of keys is used, with a master key encrypting key encryption keys (KEKs) which in turn encrypt individual password keys. This offers better security but increases complexity.

5. Code for Encryption



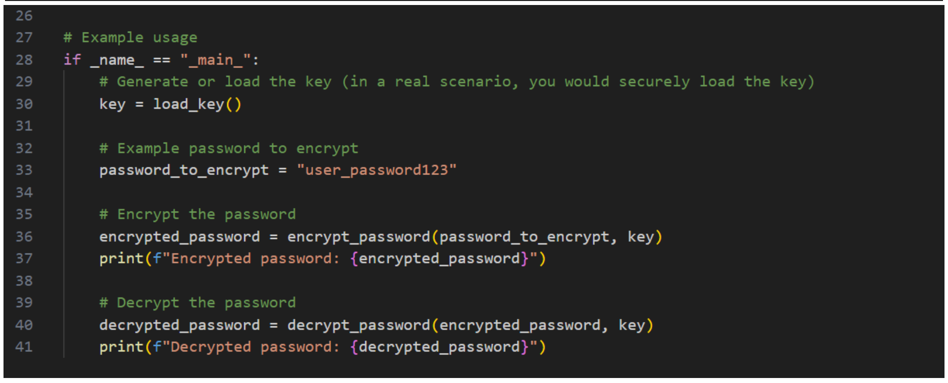
```python
from cryptography.fernet import Fernet
import os

# Generate a key for encryption (you should securely store this key)
def generate_key():
    return Fernet.generate_key()

# Load an existing key from a secure location
def load_key():
    # Replace with your own method of securely loading the key
    # Example: You could load from a file or environment variable
    # For demonstration purposes, using a static key
    return b'your_generated_key_here'

# Encrypt a password
def encrypt_password(password, key):
    fernet = Fernet(key)
    encrypted_password = fernet.encrypt(password.encode())
    return encrypted_password

# Decrypt a password
def decrypt_password(encrypted_password, key):
    fernet = Fernet(key)
    decrypted_password = fernet.decrypt(encrypted_password).decode()
    return decrypted_password
```



```python

# Example usage
if _name_ == "_main_":
    # Generate or load the key (in a real scenario, you would securely load the key)
    key = load_key()

    # Example password to encrypt
    password_to_encrypt = "user_password123"

    # Encrypt the password
    encrypted_password = encrypt_password(password_to_encrypt, key)
    print(f"Encrypted password: {encrypted_password}")

    # Decrypt the password
    decrypted_password = decrypt_password(encrypted_password, key)
    print(f"Decrypted password: {decrypted_password}")
```

6. Secure Password Packages

- **Strong Password Policies:**

Enforce strong password requirements (length, complexity).

- **Multi-Factor Authentication (MFA):**

Implement MFA to add an extra layer of security beyond passwords.

- **Regular Password Updates:**

Encourage users to update their passwords regularly.

7. Applications

➢ **Key Management System (KMS):**

- A centralized platform for securely storing and managing encryption keys.

- Enables implementing access controls and automating key rotation for enhanced security.

➢ **Encryption Software:**

- Utilizes robust encryption algorithms to secure stored password keys.

- Ensures that even if unauthorized access occurs, the keys remain encrypted and unreadable without the decryption key.

➢ **Secure Backup and Recovery Solutions:**

- Regularly backs up encrypted password keys and stores them securely.

- Facilitates data recovery in case of disk failure or unexpected incidents, ensuring data availability.

➢ **Tokenization Solutions:**

- Tokenization replaces sensitive data like password keys with unique tokens.

- Helps in reducing the exposure of actual keys, enhancing security during storage and transmission.

➢ **Secure Containers:**

- Utilizes secure containers to isolate and protect password keys.

- Ensures that even if the container is compromised, the keys remain secure within the isolated environment.

➢ **Hardware Security Modules (HSM):**

- Hardware devices dedicated to secure key storage and cryptographic operations.

8. Conclusion

In conclusion, by implementing a combination of Key Management Systems, Encryption Software, Secure Backup and Recovery Solutions, Tokenization, Secure Containers, and Hardware Security Modules, organizations can significantly enhance the security of user password keys stored on disks. These applications work together to create a robust security framework that protects sensitive information, minimizes the risk of unauthorized access, and ensures data integrity. By adopting a multi-layered approach to data security, organizations can maintain the confidentiality and integrity of password keys, safeguarding user information effectively.