Group - 3

Write the encryption/decryption component and submit its description as a short memo via Canvas.

Overview

This memo describes the encryption and decryption components created for simulating a ransomware scenario. The encryption component is responsible for encrypting files and directories with a strong encryption algorithm (AES-256), while the decryption component allows the victim to decrypt the files after the ransom is paid and the decryption key is securely received.

Encryption Component

The encryption component is implemented using the **AES-256** encryption algorithm in **CBC mode**. It encrypts files and directories recursively by following these steps:

1. **Generate an AES Key**: A 256-bit (32-byte) AES key is generated randomly using the `get_random_bytes()` function from the `Crypto.Random` module.
2. **Initialization Vector (IV)**: A random 16-byte IV is generated for each file and is used in conjunction with the AES key for encryption. The IV is saved as the first 16 bytes of each encrypted file.
3. **Padding**: Since AES encryption requires input data to be a multiple of 16 bytes, padding is added to the data using the **PKCS7** padding scheme.
4. **Encrypt Data**: The data is encrypted using the AES algorithm in **CBC mode**. The encrypted data and the IV are saved in the `.enc` file.
5. **File Handling**: Each file is processed one by one, and the `.enc` extension is added to the encrypted file's name (only once).

Decryption Component

The decryption component uses the same AES key (which should be securely transmitted to the victim after the ransom is paid) to reverse the encryption process. The decryption process works as follows:

1. **Extract IV**: The first 16 bytes of each `.enc` file are read as the **Initialization Vector (IV)**.
2. **Decrypt Data**: The AES key and IV are used to decrypt the file contents.
3. **Remove Padding**: After decryption, the padding added during encryption is removed by checking the last byte (which indicates the padding length).
4. **Restore Original Files**: The decrypted data is saved to a new file with the `.enc` extension removed.

How It Works:

1. **Encryption**:
    - Files are recursively encrypted using AES-256 CBC mode.
    - A random IV is used for each file, ensuring the encryption is unique.
    - Padding is added to ensure that the data is a multiple of 16 bytes, and the encrypted data is saved with the `.enc` extension.
2. **Decryption**:
    - The victim, after receiving the decryption key, runs the decryption tool on their files.
    - The `.enc` extension is removed after the file is decrypted, and padding is stripped to restore the original content.

Security Considerations:

- **Key Management**: The decryption key must be securely transmitted to the victim after the ransom is paid. Ensure that the transmission method is secure (e.g., HTTPS, encrypted communication).
- **File Integrity**: Proper checks should be in place to verify the integrity of the files before decryption to avoid potential corruption during the encryption/decryption process.

Conclusion:

This encryption and decryption system simulates ransomware encryption and recovery. It uses AES-256 CBC mode to securely encrypt and decrypt files, ensuring both security and confidentiality. Proper handling of the IV, padding, and file structure is essential for ensuring the correctness of the encryption/decryption operations.