Nom module – Cybersécurité

TP : Chiffrement symétrique sous Linux

Objectifs :

- Comprendre le mécanisme de chiffrement et déchiffrement symétrique sous Linux
- Apprendre à utiliser la boite à outil cryptographique « openssl »

Prérequis :

- Un système d'exploitation Linux (Ubuntu, CentOS, etc.).
- La commande "openssl" doit être installée sur le système (elle est généralement préinstallée sur la plupart des distributions Linux).

Partie I : Chiffrement symétrique avec OpenSSL sous Linux

Étape 1 : Vérification de la disponibilité d'OpenSSL

Vérifiez que la commande "openssl" est installée sur le système en exécutant la commande suivante

openssl version

- Si la commande est installée, elle affichera la version d'OpenSSL installée sur votre système.

Étape 2 : Génération de la clé de chiffrement

Générez une clé de chiffrement symétrique
en utilisant OpenSSL :

openssl rand -base64 32 > ma_cle.txt

- Cette commande génère une clé de chiffrement de 32 octets (256 bits) et la stocke dans un fichier nommé "ma_cle.txt".
  - **"openssl"** : C'est la commande qui permet d'effectuer diverses opérations liées à la sécurité à l'aide de la bibliothèque OpenSSL.
  - **"rand"** : C'est l'option de la commande OpenSSL qui permet de générer des données aléatoires.
  - **"-base64"** : C'est une option supplémentaire qui spécifie que la sortie générée doit être encodée en Base64.
  - **"32"** : C'est l'argument qui spécifie le nombre d'octets de données aléatoires à générer. Dans ce cas, nous générons une clé de 32 octets (256 bits).

Étape 3 : Chiffrement d'un fichier

Chiffrez un fichier en utilisant la clé générée à l'étape précédente **:**
openssl enc -aes-256-cbc -salt -in mon_fichier.txt -out mon_fichier_encrypted.txt -pass file:ma_cle.txt

- Cette commande utilise AES-256 en mode CBC (Cipher Block Chaining) pour chiffrer le fichier "mon_fichier.txt" en utilisant la clé stockée dans le fichier "ma_cle.txt". Le fichier chiffré est enregistré sous le nom "mon_fichier_encrypted.txt".
    - **"enc"** : C'est l'option de la commande OpenSSL qui spécifie que nous souhaitons effectuer des **opérations de chiffrement**.
    - "-aes-256-cbc" : C'est l'algorithme de chiffrement que nous souhaitons utiliser, dans ce cas, AES avec une taille de clé de 256 bits en mode CBC.
    - **"-salt"** : C'est l'option qui ajoute un sel (valeur aléatoire) au processus de chiffrement pour renforcer la sécurité.
    - **"-in mon_fichier.txt"** : C'est l'option qui spécifie le fichier source contenant les données à chiffrer, dans ce cas, "mon_fichier.txt".
    - **"-out mon_fichier_encrypted.txt"** : C'est l'option qui spécifie le fichier de destination dans lequel le texte chiffré sera enregistré, dans ce cas, "mon_fichier_encrypted.txt".
    - **"-pass file:ma_cle.txt7"** : C'est l'option qui spécifie la clé de chiffrement utilisée pour le processus de chiffrement. Ici, la clé est fournie dans le fichier "ma_cle.txt7".

Étape 4 : Déchiffrement du fichier

Déchiffrez le fichier chiffré en utilisant la même clé de chiffrement :

openssl enc -d -aes-256-cbc -in mon_fichier_encrypted.txt -out mon_fichier_decrypted.txt -pass file:ma_cle.txt

- Cette commande déchiffre le fichier chiffré "mon_fichier_encrypted.txt" en utilisant la clé stockée dans "ma_cle.txt" et enregistre le résultat dans "mon_fichier_decrypted.txt".

    - "-d" : C'est l'option qui indique à OpenSSL que nous souhaitons effectuer une opération de déchiffrement (déchiffrer le fichier).

Étape 5 : Vérification

Vérifiez que le contenu du fichier déchiffré ("mon_fichier_decrypted.txt") correspond au contenu original du fichier avant le chiffrement ("mon_fichier.txt").

Partie II : Utilisation de différentes longueurs de clé

1. Générer des clés de chiffrement de différentes longueurs, 128 bits, 192 bits et 256 bits.
2. Chiffrer un fichier en utilisant chaque clé générée.
3. Comparer les tailles des fichiers chiffrés.
4. Expliquer pourquoi les tailles des fichiers chiffrés diffèrent en fonction de la longueur de clé utilisée.
5. Donner des avantages et des inconvénients d'utiliser des clés de différentes longueurs en termes de sécurité et de taille des fichiers chiffrés.
6. Expliquer la relation entre la longueur de clé et la sécurité du chiffrement symétrique.

Partie III : Chiffrement avec mot de passe

Objectif : générer une clé dérivée à partir d'un mot de passe donné en utilisant une fonction de dérivation de clé sécurisée.
Dans cet exemple, nous utiliserons **PBKDF2** (Password-Based Key Derivation Function 2) avec un sel aléatoire pour renforcer la sécurité.

Rappel :

Un sel aléatoire, également appelé "salt" en anglais, est une valeur aléatoire générée de manière aléatoire et utilisée comme paramètre supplémentaire dans des algorithmes de chiffrement et de hachage. Son objectif principal est d'ajouter de l'entropie et de la sécurité au processus de chiffrement ou de hachage.
Dans le contexte du chiffrement symétrique, le sel est utilisé pour mélanger les données claires avant le chiffrement, rendant ainsi les attaques par dictionnaire ou par force brute plus difficiles. Chaque fois que les mêmes données claires sont chiffrées, le texte chiffré sera différent en raison de la présence du sel aléatoire.

1. Générer un sel aléatoire de 16 octets (128 bits) et le sauvegarde dans le fichier "salt.txt".
2. Inviter l'utilisateur à entrer un mot de passe en tapant la commande
   read -s -p "Entrez le mot de passe : " password
   (l'option -s masque l'entrée). Le mot de passe saisi sera stocké dans la variable "password".

3. Taper la commande suivante pour effectue le chiffrement en utilisant l'algorithme AES-256-CBC, le sel généré précédemment et le mot de passe saisi par l'utilisateur.
   openssl enc -aes-256-cbc -salt -kdf pbkdf2 -in mon_fichier.txt -out mon_fichier_encrypted_with_password.txt -pass "pass:$password" -p -S $(cat salt.txt)

   - **"-pass"** : C'est l'option qui indique à OpenSSL que nous allons fournir une clé via la ligne de commande plutôt que de laisser OpenSSL la demander directement à l'utilisateur.

   - **""pass:$password""** : C'est la manière de spécifier la clé à OpenSSL. Dans cette expression, "pass:" indique à OpenSSL que la clé sera fournie sous la forme d'une chaîne de caractères (password-based). Le mot de passe saisi par l'utilisateur est représenté par "$password".

   - **"-p"** : C'est l'option qui demande à OpenSSL d'afficher le mot de passe utilisé pour le chiffrement ou le déchiffrement à des fins de débogage. Cela signifie que lors de l'exécution de la commande, OpenSSL affiche le mot de passe saisi par l'utilisateur, ce qui peut être utile pour vérifier que le mot de passe est correct.

   - **"-S $(cat salt.txt)"** : C'est l'option qui spécifie le sel (salt) utilisé pour le processus de chiffrement ou de déchiffrement. Le sel est extrait du fichier "salt.txt" à l'aide de la commande "cat". Le sel est ensuite passé à OpenSSL pour être utilisé dans le processus de dérivation de clé PBKDF2 (Password-Based Key Derivation Function 2).

4. Déchiffrer le fichier chiffré avec le mot de passe utilisé précédemment.
5. Vérifiez que le contenu du fichier déchiffré.
6. Donner des avantages et des inconvénients du chiffrement avec mot de passe par rapport à l'utilisation de clés de chiffrement symétrique.

Partie IV : Sécurité et stockage des clés de chiffrement

Objectif : Comprendre l'importance de la sécurité et du stockage adéquat des clés de chiffrement symétrique pour protéger les données chiffrées.

1. Générer une clé de chiffrement symétrique à l'aide de OpenSSL et d'enregistrer la clé dans un fichier protégé.
2. Chiffrer un fichier en utilisant la clé stockée dans le fichier.
3. Donner les meilleures pratiques pour la gestion des clés de chiffrement