Projet : Cryptographie sous Linux

Objectifs :

- Comprendre le mécanisme de chiffrement et déchiffrement symétrique sous Linux
- Apprendre à utiliser la boîte 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).

Remarques :

- Travail individuel.
- Vous devez fournir un rapport unique sur votre travail, intitulé avec vos noms et prénoms.
- Le rapport doit contenir des captures d'écran de toutes les parties avec *

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 : 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 -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. (*) Vérifier 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 III : Chiffrement Asymétrique

Dans cet exercice, l'objectif est de chiffrer un document avec une clé publique et de le déchiffrer grâce à la clé privée.

1. Créer l'arborescence suivante dans le bureau de la machine Linux :

```
--- LabCrypto _______ Alice
            |_______ Bob
```

2. Créer un fichier dans le dossier de Alice avec le nom : AliceDocument (commande gedit)
3. Mettre dans le fichier avec le texte Suivant : Bonjour Bob, je suis Alice.
4. (*) Créer une paire de clés Privée/Publique pour Alice dans le dossier Alice (algorithme RSA), la clé privée doit s'appeler "AlicePrivateKey" et la clé publique "AlicePublicKey".
5. (*) Créer une paire de clés Privée/Publique pour Bob dans le dossier Bob (algorithme RSA) la clé privée doit s'appeler "BobPrivateKey" et la clé publique "BobPublicKey".

Indice : utiliser les commandes openssl genrsa -out KeyPair et openssl rsa -in KeyPair -pubout –out PublicKey

6. Alice veut envoyer le document AliceDocument à Bob d'une manière à garantir la confidentialité de son contenu, que doit-elle faire ?
7. Alice doit utiliser quelle clé pour chiffrer le fichier à envoyer ?
8. (*) En utilisant la bonne clé, chiffrer le document AliceDocument, le fichier résultant doit s'appeler "AliceDocumentEncrypted".

Indice : utiliser la commande openssl rsautl (remplacée par pkeyutl).

9. Copier le fichier chiffré dans le dossier de Bob.
10. Quelle clé doit être utiliser par Bob pour déchiffrer le document AliceDocumentEncrypted ?
11. (*) En utilisant la bonne clé, déchiffrer le fichier chiffré, le fichier résultant doit s'appeler "AliceDocumentDecrypted".
12. (*) Vérifier le contenu du fichier résultant, et comparer le avec le fichier AliceDocument.

13. Maintenant, créer un fichier de grande taille nommé "LargeFile" en utilisant la commande : openssl rand -out LargeFile -base64 $((2**30 * 3/4))
14. (*) Chiffrer le fichier LargeFile en utilisant la clé publique de Alice.

15. (*) C'est quoi le message affiché ? Pourquoi ?

Partie IV : Les signatures numériques

L'objectif de cette partie est de vous montrer comment Alice peut créer une signature électronique afin de : s'authentifier auprès de Bob, assurer la non-répudiation pour elle-même et garantir l'intégrité des données signées.

1. Créer un fichier AuthData dans le dossier de Alice, Avec le contenu : Je suis Alice.
2. (*) Vous allez maintenant procéder à l'application de la fonction de hachage SHA256 sur le document AuthData pour trouver son hachage HashAuthData. Pour ce faire, utiliser la commande openssl dgst avec les bons paramètres.
3. Pour que Alice signe le document HashAuthData, quelle clé doit-être utiliser ?
4. (*) Créer la signature du document en utilisant la bonne clé, le document résultant doit se nommer : AliceSignature.

Indice : utiliser la commande openssl rsautl et les options -sign, -in, -inkey, -out

5. Copier les deux fichiers : AuthData et AliceSignature dans le dossier de Bob.
6. Bob doit vérifier la signature d'Alice sur le document, quelle clé doit-il utiliser ?
7. (*) En utilisant la bonne clé, vérifier la signature d'Alice (montrer toutes les étapes).

Indice : la commande diff permet de comparer deux fichiers.

Partie V : Certificats X509

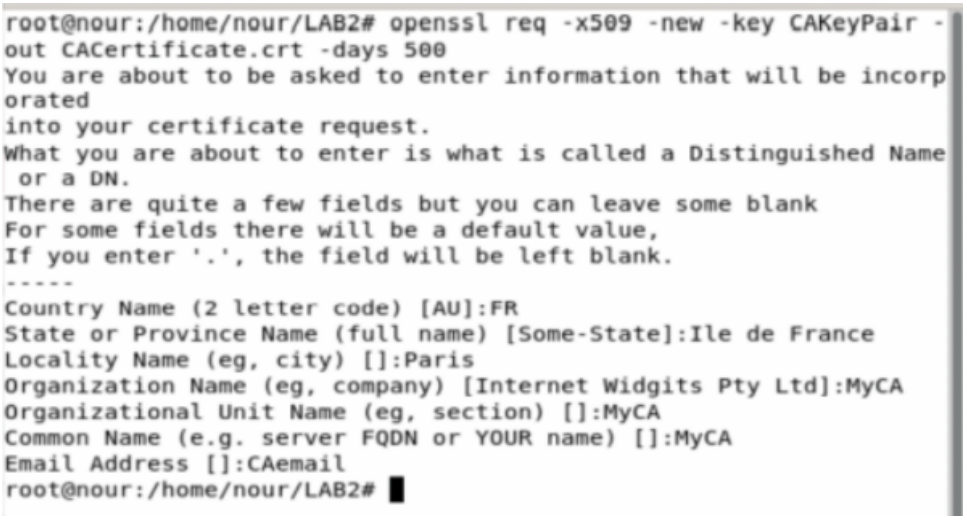
1. Retourner dans le dossier parent de Bob et Alice.
2. (*) Récupérer le certificat du serveur *www.lcl.fr* et stockez le résultat dans le fichier CertificateLCL en utilisant la commande : OpenSSL : openssl s_client -connect www.lcl.fr:443 > CertificateLCL
3. (*) Afficher le nom de l'autorité de certification qui a signé CertificateLCL en utilisant la commande OpenSSL : openssl x509 -noout -in CertificateLCL -issuer
4. Utiliser la commande openssl x509 -help pour plus d'informations.
5. (*) Afficher la date de validité du CertificateLCL.
6. (*) Afficher la signature du certificatLCL.
7. (*) Afficher le numéro de série du CertificateLCL.

8. (*) Afficher la clé publique du certificatLCL.
9. L'objectif de cette étape est de générer un certificat auto-signé pour une CA et un certificat pour un serveur en utilisant la clé privée de la CA.
10. Générer une paire de clés (2048 bits) pour un serveur protégé par un mot de passe de votre choix (par exemple :password) en utilisant la commande OpenSSL : openssl genrsa et les paramètres -out et -passout pass:password. La paire de clé doit se nommer : ServerKeyPair
11. Extraire la clé publique du serveur ServerPublicKey de ServerKeyPair, et renommer ServerKeyPair en ServerPrivateKey.
12. Générer la demande de certificat pour le serveur ServerRequest.csr comme indiqué dans la Figure suivante. Utiliser les mêmes informations que celles indiquées dans la Figure, et laisser vide en appuyant sur la touche Entrée pour Un mot de passe d'authentification, Un nom de société facultatif. nom de l'entreprise.

```
root@nour:/home/nour/LAB2# openssl req -new -key ServerPrivateKey
-out ServerRequest.csr
You are about to be asked to enter information that will be incorp
orated
into your certificate request.
What you are about to enter is what is called a Distinguished Name
 or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:Ile de France
Locality Name (eg, city) []:Paris
Organization Name (eg, company) [Internet Widgits Pty Ltd]:EPITA
Organizational Unit Name (eg, section) []:First year
Common Name (e.g. server FQDN or YOUR name) []:myserver.fr
Email Address []:youremail

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

13. (*) Générer une paire de clés (4096 bits) CAKeyPair pour une CA protégée par un mot de passe de votre choix. par exemple : mot de passe).
14. (*) Générez un certificat auto-signé CACertificate.crt pour la CA comme indiqué dans la
Figure suivante. Utiliser les mêmes informations que celles indiquées dans la figure.



15.(*)Maintenant, générer le certificat du serveur ServerCertificate.crt en utilisant le certificat et les clés de la CA. Utiliser la commande : openssl x509 et les paramètres ;-req -in -CA -CAkey -CAcreateserial -out -days (pour la durée de validité) et -sha256.

16. (*)Vous pouvez vérifier le certificat du serveur en entrant la commande OpenSSL : openssl verify-CAfile CACertificate.crt ServerCertificate.crt. Que remarquez-vous ?