Réseau et sécurité
M1 informatique

TP OpenSSL

Remarque : plusieurs exercices de ce TP sont à faire en binôme.

Définition : OpenSSL est une librairie *open-source* d'utilitaires cryptographiques.

1. Utilisation de l'outil openssl

Exercice 1.1 : Installez *openssl*.

Syntaxe :

openssl commande *[*options*] [*arguments*]*

Les commandes que nous utiliserons dans ce TP incluent : *passwd*, *enc*, *genrsa*, *x509*, *dgst*, *req*, *verify*.

2. Codage en base64

Définition : Base64 est un codage utilisant 65 caractères imprimables (les 26 lettres majuscules, les 26 lettres minuscules, les 10 chiffres, le caractère +, le caractère / et le caractère spécial =). Base64 permet d'échanger des données en réduisant les problèmes d'encodage des caractères spéciaux.

Syntaxe : Pour encoder en base64, on utilise :

openssl enc -base64 -in fichier

Pour décoder de la base64, on utilise :

openssl enc -base64 -d -in fichier

Exercice 2.1 : Encodez un mot de passe avec base64 et donnez le à votre binôme. Votre binôme devra retrouver le mot de passe.

Exercice 2.2 : Est-ce que base64 est un moyen sûr de protéger un mot de passe ?

3. Fichiers de mots de passe

Définition : Le fichier */etc/passwd* contient des informations sur les utilisateurs. Il est, par défaut, lisible par tous les utilisateurs.

man 5 passwd

Définition : Le fichier */etc/shadow* contient des informations sur les mots de passe des utilisateurs. Il est, par défaut, lisible uniquement par l'administrateur.

man 5 shadow

Exercice 3.1 : Créez un utilisateur *pass1* avec un mot de passe quelconque en utilisant la commande *adduser*.

```
man 8 adduser
```

Exercice 3.2 : Identifiez le champ contenant le mot de passe de l'utilisateur.

Définition : Le mot de passe peut être chiffré avec DES ou avec MD5. Si le champ du mot de passe de l'utilisateur commence avec *1*, le mot de passe est chiffré avec MD5, sinon il est chiffré avec DES. Le sel se trouve entre le deuxième *$* et le troisième *$*. Le mot de passe chiffré se trouve après le troisième *$*.

Syntaxe :

```
openssl passwd [options]
```

où les options possibles sont :
-*crypt* : pour l'algorithme standard de chiffrement (DES)
-*1* : pour un chiffrement basé sur MD5
-*salt* sel : pour ajouter le sel

Exercice 3.3 : Obtenez le champ contenant le mot de passe en utilisant *openssl*.

Exercice 3.4 : Changez le mot de passe de l'utilisateur *pass1* et ne tapez que quatre lettres minuscules. Faites un programme qui prend en entrée le champ contenant le mot de passe et qui retrouve le mot de passe de l'utilisateur.

Exercice 3.5 : Si vous ne disposez pas du sel mais uniquement du mot de passe chiffré, comment faudrait-il adapter votre programme ?

4. Chiffrement

Syntaxe : Pour le chiffrement, on utilise la commande *enc*. Pour le déchiffrement, on utilise la commande *enc* et l'option *-d*. Pour utiliser DES, on utilise l'option *-des*. Pour utiliser le triple DES, on utilise l'option *-des3*.

Exercice 4.1 : Chiffrez un fichier quelconque et déchiffrez le avec le bon mot de passe.

Exercice 4.2 : Chiffrez un fichier quelconque et déchiffrez le avec un mauvais mot de passe.

Remarque : *openssl* détecte que le mot de passe est mauvais. Nous allons chercher à savoir comment.

Exercice 4.3 : Comparez les tailles des fichiers clairs et chiffrés. Expliquez la différence.

Exercice 4.4 : Prenez deux fichiers très différents *f1* et *f2*. Chiffrez puis déchiffrez ces deux fichiers avec le bon mot de passe, et en utilisant l'option *-nopad*. Vous obtenez des fichiers que vous nommerez *f1b* et *f2b*.

Exercice 4.5 : Avec l'outil *xxd* (par exemple), étudiez la différence entre *f1* et *f1b*, ainsi qu'entre *f2* et *f2b*. Expliquez les différences.

Exercice 4.6 : Prenez à présent un fichier f3. Chiffrez-le puis déchiffrez-le en utilisant un mauvais mot de passe, et en utilisant l'option *-nopad*. Vous obtenez un fichier que vous nommerez *f3b*. Examinez le

fichier *f3b* (seulement). Sans comparer *f3* et *f3b*, comment pouvez-vous déterminer que *f3b* a été déchiffré de manière incorrecte ?

5. RSA

Génération des clés

Syntaxe : La génération de clefs se fait en utilisant :

openssl genrsa -out clef-privée taille

Exercice 5.1 : Créez une clef privée RSA *mySmall.pem* de 50 bits.

Syntaxe : Pour obtenir des informations sur une clef, on utilise :

openssl rsa -in clef *-text -noout*

Exercice 5.2 : Affichez des informations sur la clef privée que vous avez générée.

Exercice 5.3 : Faut-il conserver la clef privée en clair ? Est-ce que votre clef privée est sauvegardée en clair ?

Exercice 5.4 : Cassez la clef RSA *otherSmall.pem* suivante. Il est conseillé de faire un programme.

```
-----BEGIN RSA PRIVATE KEY-----
MDcCAQACBwLXEvQUSR0CAwEAAQIHAf9IXGeOgQIEAcx/UQIEAZQyDQIEAK3/cQIE
AKWD/QIDPAMq
-----END RSA PRIVATE KEY-----
```

Exercice 5.5 : Cassez votre clef RSA. Pourquoi avez-vous pu la casser ?

Syntaxe : Pour ne pas sauvegarder la clef privée en clair, on peut utiliser un algorithme de chiffrement symétrique :

openssl genrsa -des3 -out clef-privée taille

Exercice 5.6 : Créez une clef privée *private1.pem* de 1024 bits, sans chiffrement. Créez une clef privée *private2.pem* de 1024 bits, avec chiffrement.

Exercice 5.7 : Récupérez les informations sur chacune de ces deux clefs.

Syntaxe : Pour récupérer la clef publique associée à une clef, on utilise :

openssl rsa -in clef-privée *-pubout -out* clef-publique

Exercice 5.8 : Créez la clef publique associée à chaque clef privée.

Exercice 5.9 : Faut-il conserver la clef publique en clair ? Est-ce que votre clef publique est sauvegardée en clair ?

Syntaxe : Pour afficher des informations sur une clef publique, il faut utiliser l'option *-pubin* pour

indiquer que le fichier d'entrée comprend seulement des informations publiques.

Exercice 5.10 : Récupérez des informations sur les deux clefs publiques.

Chiffrement et déchiffrement

Syntaxe : Pour chiffrer avec une clef publique RSA, on utilise :

openssl rsautl -encrypt -in fichier-clair *-inkey* clef-publique *-pubin -out* fichier-chiffré

Pour déchiffrer, on utilise l'option *-decrypt*.

Exercice 5.11 : Chiffrez et déchiffrez un petit fichier.

Exercice 5.12 : Chiffrez un gros fichier. Que se passe-t'il ? Pourquoi ?

Exercice 5.13 : Demandez à votre binôme sa clef publique. Chiffrez un court message avec sa clef publique. Transmettez-le lui, et demandez-lui de le déchiffrer.

Syntaxe : Pour chiffrer avec DES, on utilise :

openssl enc -des -in fichier-clair *-out* fichier-chiffré

Pour déchiffrer avec DES, on utilise :

openssl enc -des -d -in fichier-chiffré *-out* fichier-clair

Exercice 5.14 : Demandez à votre binôme sa clef publique. Chiffrez un long message avec un algorithme de chiffrement symétrique (comme DES), en utilisant un mot de passe. Chiffrez le mot de passe avec la clef publique de votre binôme. Transmettez à votre binôme le mot de passe chiffré et le message chiffré. Demandez-lui de déchiffrer le message.

Signatures

Syntaxe : Pour hacher un fichier, on utilise :

openssl dgst -md5 -out haché fichier

Syntaxe : Pour signer un haché, on utilise :

openssl rsautl -sign -in haché *-inkey* clef *-out* signature

Exercice 5.15 : Faut-il utiliser la clef publique ou la clef privée pour une signature ?

Syntaxe : Pour vérifier une signature, on utilise :

openssl rsautl -verify -in signature *-pubin -inkey* clef-publique *-out* haché2

Exercice 5.16 : Signez un message.

Exercice 5.17 : Vérifiez sa signature. Vous pourrez utiliser l'outil *diff*.

Exercice 5.18 : Créez trois messages. Signez ces trois messages. Modifiez-en un ou deux légèrement. Transmettez à votre binôme les messages (éventuellement modifiés) ainsi que les signatures

correspondantes. Demandez-lui de déterminer quels sont les messages qui ont été modifiés.

6. Certificats

Création de certificats

Syntaxe : Pour demander un certificat, il faut créer une requête de la manière suivante :

openssl req -new -key clef *-out* requête

Remarque : la clef doit être suffisamment grande pour pouvoir signer le certificat. Une clef de 1024 bits convient généralement.

Exercice 6.1 : Créer une requête de certificat appelée *user-request.pem* concernant une entité appelée *user*. Cette requête est en attente de signature par une autorité de certification.

Syntaxe : Pour visualiser une requête de certificat ou un certificat, on utilise :

openssl x509 -in certificat *-text -noout*

Exercice 6.2 : Visualiser le contenu de votre requête de certificat.

Exercice 6.3 : Créer une paire de clefs pour l'autorité de certification. Créer une requête de certificat nommée *ca-request.pem* concernant l'entité de certification appelée *CA*.

Syntaxe : Pour autosigner une requête de certificat, on utilise :

openssl x509 -req -in requête *-signkey* clef-privee *-out* certificat

Exercice 6.4 : Autosigner la requête de certificat de *CA* dans un certificat nommé *ca-certificat.pem*.

Exercice 6.5 : Visualiser le certificat *ca-certificat.pem*.

Syntaxe : Pour signer une requête de certificat, on utilise :

openssl x509 -days duree *-CAserial* serial *-CA* certificat *-CAkey* clef *-in* requête *-req -out* certificat

Le fichier serial contient un nombre enregistré en hexadécimal (c'est-à-dire un nombre pair de chiffres hexadécimaux).

Exercice 6.6 : Signer la requête de certificat *user-request.pem* pour produire un certificat *user-certificat.pem*.

Syntaxe : Pour vérifier un certificat, on utilise :

openssl verify -CAfile ca-certificat certificat

Exercice 6.7 : Vérifier le certificat *user-certificat.pem*.

Exercice 6.8 : Modifier le contenu du certificat *user-certificat.pem* et tentez de le vérifier.

Chaîne de certification

Exercice 6.9 : Créez deux autorités de certifications CA1 et CA2. CA1 est l'autorité de certification racine, et CA2 est certifiée par CA1.

Exercice 6.10 : Créez une requête de certificat et signez la par CA2.

Exercice 6.11 : Vérifiez la chaîne complète de certification du certificat que vous venez de créer.