Travaux pratiques de Sécurité à réaliser sous Linux

TP SSL (Secure Sockets Layer)

1) OpenSSL

OpenSSL est une boîte à outils de chiffrement comportant deux bibliothèques (libcrypto et libssl). La premiere fournit des algorithmes de chiffrement, la seconde implémente le protocole Transport Layer Security (TLS) et son prédécesseur Secure Sockets Layer (SSL)). Ces fonctionnalités sont accessibles en ligne de commande mais peuvent aussi être intégrée dans une application via une API. Développée en C, OpenSSL est disponible sur les principaux systèmes d'exploitation Slon Wikipedia, en 2014, deux tiers des sites web l'utilisaient. Voici quelques exemples d'utilisation :

Commandes	**Signification**
openssl ciphers -v	Liste des modes de chiffrements disponibles
openssl dgst -md5 File	digest md5 du fichier File
openssl genrsa	Génération d'une clé privée RSA 512
openssl genrsa -out maclee_priv.pem 1024	Idem mais 1024 bits au lieu de 512 et mise en fichier
openssl rsa -in maclee.pem -pubout -out maclee_pub.pem	Génération de la clé publique correspondant à la clé privée
echo -n XXXX | openssl base64	Codage base 64 de XXXX

Q1. Jetez un œil sur le man et sur le site http://wiki.openssl.org/index.php/Command_Line_Utilities puis essayez quelques commandes pour vous familiariser avec openssl.

Dans tous les exercices qui suivent, vous prendrez soin d'analyser le contenu de tous les fichiers générés (clés, chiffrés, déchiffrés, etc)

Q2. Il est possible de chiffrer avec ou sans le paramètre -salt. Identifier l'intérêt de ce paramètre.

Q3. Chiffrement Symétrique
Chiffrez et déchiffrez un petit fichier texte avec l'algorithme RC4. Vérifiez que vous retrouvez bien le fichier d'origine après la phase de déchiffrement. Refaites le test avec DES3.

Q4. chiffrement Asymétrique
Il existe plusieurs services utilisant des algorithmes de chiffrement. Contrairement au chiffrement symétrique qui chiffre et déchiffre avec la même clé, le chiffrement asymétrique (dit aussi à clé privé/publique) chiffre avec une des clés et

déchiffre avec la seconde. Nous allons tester le cas d'usage suivant : Bob souhaite envoyer un texte qui doit rester secret à Alice. Pour cela, il récupère la **clé publique** d'Alice avec laquelle il chiffre le texte secret. Il envoie le texte chiffré à Alice qui utilise sa **clé privée** pour le déchiffrer.

Pour tester ce scénario on vous demande de réaliser les étapes suivantes :

- ✔ Éditer et sauvegarder un fichier texte de test contenant la phrase suivante : « Ceci est un message secret »
- ✔ Générer une clé RSA privée et la clé publique correspondante qui seront chacune stockée dans un fichier.
- ✔ Chiffrer le contenu du fichier en clair avec la clé publique et stocker le résultat dans un fichier
- ✔ Déchiffrer le fichier secret avec la clé privé et assurer vous que vous récupérer bien le fichier avec le bon contenu en clair.

La commande suivante permet de chiffrer ou déchiffrer un fichier si on dispose de la clé (publique ou privée)

openssl **AL** -**SS** -in **FI** -inkey **KY** -out **FS**

- **AL** : Technique de chiffrement, on utilisera **rsautl**, une option d'openssl qui implémente des fonctions liées à RSA
- **FI** : Fichier de donnée à chiffrer ou déchiffrer
- **KY** : Fichier contenant la clée (publique ou privée)
- **FS** : Fichier résultat contenant les données chiffrées ou déchiffrée.
- **SS** : Sens chiffrement = **encrypt,** déchiffrement= **decrypt**
- Si le fichier de clé ne contient que la clé publique, rajouter l'option **-pubin** après KY

Q5. Signature numérique

Pour simuler l'expédition de fichiers, vous allez créer deux répertoires représentant les comptes de l'expéditeur et du récepteur et copier manuellement les fichiers à transmettre. L'expéditeur réalise les opérations suivantes :

- génération d'une paire de clé RSA
- création d'un fichier clair à signer
- calcul de la signature
- expédition du fichier clair et de la signature

Sachant que la creation et la vérification de la signature est réalisé comme suit :

```
openssl dgst -binary -out sig.sig -sign xxx.pem clair.txt
openssl dgst -signature sig.sig -verify yyy.pem clair.txt
```

Aprés avoir identifié **xxx.pem** et **yyy.pem** réalisez pour l'expéditeur et le récepteur les différentes opérations de génération et de vérification de la signature.