Master RT
RT802

TP1

Commandes de bases Openssl

Le protocole SSL (Secure Socket Layer, en français couche de socket sécurisée) a été développé par la société Netscape Communications Corporation pour permettre aux applications client/serveur de communiquer de façon sécurisée. En 1994, Netsape diffuse le protocole SSL 2.0 et en 1996 le SSL 3.0.

TLS (Transport Layer Security) est une évolution de SSL réalisée par l'IETF[1]. En 2001, l'IETF rachète le protocole et le renomme en TLS (considéré comme un SSL 3.1).
SSL est un protocole qui s'intercale entre TCP/IP et les applications qui s'appuient sur TCP. Il est donc transparent pour celles-ci. Une session SSL se déroule en deux temps. Une première phase de poignée de mains (*handshake*) va permettre au client et au serveur de s'identifier (certificats X509), et de se mettre d'accord sur le système de chiffrement pour ensuite partager une clé qu'ils utiliseront plus tard. La seconde phase correspond à la phase de communication durant laquelle les données échangées sont compressées, chiffrées et signées.
`openSSL` est une boîte à outils cryptographiques implémentant les protocoles SSL et TLS offrant : • une bibliothèque de programmation en C permettant de réaliser des applications client/serveur sécurisées s'appuyant sur SSL/TLS.
• une commande en ligne (**`openssl`**) permettant
- la création de clés RSA, DSA (signature)
- la création de certificats X509
- le calcul d'empreintes (MD5, SHA, RIPEMD160, . . .)
- le chiffrement et déchiffrement (DES, IDEA, RC2, RC4, Blowfish, . . .)
- la réalisation de tests de clients et serveurs SSL/TLS
- la signature et le chiffrement de courriers (S/MIME)

Exercice 1 : Découverte d'openSSL

1. Quels sont les commandes/outils contenu(e)s dans openssl ?
2. Déterminer la version de votre openssl avec la commande `openssl version -a`
3. Qu'obtient-on en tapant la commande `openssl ciphers -v` ?
4. Comment lister les commandes qui utilisent uniquement l'algorithme de chiffrement 3DES ?

Exercice 2 : Chiffrement / Déchiffrement

La commande `openssl` **`enc`** permet de chiffrer et déchiffrer des messages. Plus d'informations sur cette commande peuvent être trouvées en tapant `openssl` **`enc`** `-h`.
1.Créer un fichier texte et y écrire quelques mots. Chiffrer ce fichier avec l'algorithme DES-CBC.
2.Ouvrir le fichier chiffré. Qu'observez-vous au niveau des premiers caractères ? Qu'est-ce qu'un sel en cryptographie et à quoi cela sert-il ?
3.Déchiffrer le fichier chiffré et vérifier qu'on retombe bien sur le fichier de départ.
4.Est-il possible de chiffrer avec un AES-128 et déchiffrer avec un AES-192 ?
5.Reprendre la question 1 mais cette fois ci en indiquant l'option *-p*. Détailler les informations obtenues.
6.Recommencer la question 5. Les informations sont-elles identiques ?
7.Refaire ce test deux nouvelles fois en ajoutant l'option *-nosalt*. Comparer et expliquer les résultats obtenus avec ceux de la 6.

Exercice 3 : Premiers pas sur RSA *Dans cet exercice, vous allez travailler avec les commandes* `openssl genrsa`, `openssl rsa` *et* `openssl rsautl`.

1. Générer une paire de clés RSA 1024 bits que vous nommerez **signature.pem**.
2. Visualiser le contenu de votre paire de clé avec la commande`openssl rsa -in <fichier> -text -noout`

- Que signifie les infos *«modulus», «exposants»* et *«prime »*.
- Pourquoi l'exposant publique vaut *65537* ?

3. Extraire ensuite la clé publique que vous nommerez **signature_publique.pem** et visualiser son contenu.

- Créer un fichier qui a une taille inférieure à 100 octets.
- Chiffrer ce fichier avec votre clé publique **signature_publique.pem**. Déchiffrer ce fichier avec votre clé privée **signature.pem**.
- Chiffrer un fichier avec votre clé privée **signature.pem**. Déchiffrer ce fichier avec votre clé publique **signature_publique.pem**.
- Expliquer à quels problèmes de sécurité répondent ces deux chiffrements différents.

Florian Legendre

Pour les exercices qui suivent, vous allez devoir envoyer des messages à votre voisin (et en recevoir). Choisissez donc une autre personne avec qui échanger.

Exercice 5 : Chiffrement avec la clé publique

Dans cet exercice, l'intérêt est de chiffrer un document avec une clé publique que seul son créateur pourra déchiffrer.

1. Préparatifs...

5. Générer une paire de clés RSA 2048 bits que vous nommerez **signature<votrenom>.pem**.
6. Extraire ensuite la clé publique que vous nommerez **signature_publique<votrenom>.pem**.
7. Envoyer à votre voisin votre clé publique.

2. Quand vous recevez une clé publique...

4. Chiffrer un fichier de votre choix avec cette clé.
5. Renvoyer le fichier chiffré à votre voisin

3. *Quand vous recevez un fichier chiffré avec votre clé...*

- Déchiffrer le fichier.
- Créer un fichier d'une taille de plusieurs mo. Puis, chiffrer ce message avec une clé publique RSA. Vous devez remarquer un problème. Comment l'expliquez-vous ?
- Qu'est-ce qu'un attaquant pourra et ne pourra pas faire tout au long des opérations effectuées

Exercice 6 : Envoi sécurisé de clé avec RSA

1. Générer une paire de clés RSA 512 bits, puis envoyer la clé publique à votre voisin.

- Générer une clé aléatoire de 256 bits
- Chiffrer cette clé avec la clé publique RSA reçue (de votre voisin).
- Envoyez-lui votre clé chiffrée.

2. Quand vous recevez la clé chiffrée de votre voisin, déchiffrez-là, puis chiffrez un fichier de votre choix avec l'algorithme AES-256-CBC et cette clé. Envoyez ce fichier chiffré à votre voisin.
3. Qu'est-ce qu'un attaquant pourra et ne pourra pas faire tout au long des opérations effectuées.