Techniques d'attaque - cryptographie

Tristan BILOT
EPITA

31 Mai 2021

Abstract

L'objectif de ce TP va être de se familiariser avec openSSL, afin de chiffrer/déchiffrer des messages. openSSL est une boîte à outils cryptographiques implémentant les protocoles SSL et TLS qui offre une bibliothèque de programmation en C permettant de réaliser des applications client/serveur sécurisées s'appuyant sur SSL/TLS.

openSSL peut être utilisé de la façon suivante:

```
openssl <commande> <option>
```

S'il n'est pas installé, l'installation peut-être faite sous Linux:

```
apt-get install openssl
```

1 Chiffrement symétrique

L'option -k permet de spécifier la clé de (dé)chiffrement, elle ba devenir obsolète donc il est préférable d'utiliser -pass. L'option -K permet de donner le pass en hexadécimal. L'option -kfile ou -passfile utilise la clé dans le fichier donné. Si le paramètre -k n'est pas utilisé, openssl demandera à l'utilisateur de renseigner une clé directement dans le terminal

2. Le déchiffrement est possible grâce à l'option -e (encode). Il faut également spécifier le type de chiffrement à utiliser, le fichier input à chiffrer et le nom du fichier output qui sera chiffré.

```
openssl enc -e -des-cbc -in cipher.txt -out decipher.txt -k key
```

Il est possible de consulter la liste de toutes les méthodes de chiffrement avec:

```
openssl enc -h
```

Après chiffrement du 1er fichier, en faisant cat cipher.txt, des symboles illisibles apparaissent, ce qui signifie que le fichier original a bien été chiffré.

3. En analysant ce fichier chiffré, nous voyons qu'au début se trouve le sel, un nombre aléatoire généré dans le durant le chiffrement afin d'éviter de retrouver facilement le message même si on possède la clé. Le sel est donc envoyé en clair dans le fichier chiffré et sert à ajouter de la robustesse au chiffrement.

4. Le déchiffrement est possible grâce à l'option -d (decode).

```
openssl enc -d -des-cbc -in cipher.txt -out decipher.txt -k key\\
```

5. en ajoutant l'option -base64, nous obtenons un message chiffré plus clair, encodé en base64. Le sel se trouvera toujours au début du message.

6. grâce à l'option -p, le détail du chiffrement/déchiffrement sera affiché, cette option peut donc être vue comme un mode "verbose" avec le détail des opérations.

En reprenant notre exemple précédent, nous obtenons 3 informations: le sel, la clé et le vecteur d'initialisation. openssl enc -e -p -base64 -des-cbc -in plain.txt -out cipher.txt
salt=383D9D2D2BF66FDA key=B64EF9867A838B13 iv =D70C9BA40CAF00C4
Dans AES ECB, afin de ne pas pouvoir déchiffrer facilement chaque bloc, on applique le chiffrement de chaque bloc de 128bit en prenant comme input un xor entre le bloc et le résultat chiffré du bloc précédent. Le 1er bloc à être chiffré, n'ayant pas de bloc précédent utilisera un vecteur d'initialisation aléatoire.

7. en chiffrant une nouvelle fois le même message, nous obtiendrons un message chiffré différent vu que le sel et le vecteur d'initialisation utilisés seront différents

8. L'option -nosalt permet de dire à l'algorithme de chiffrement de ne pas utiliser de sel, ce qui permet de retrouver le même message chiffré à partir du même message original.

9. openssl rand -out random-data.bin 1000000000

10. Temps d'exécution de différents algorithmes de chiffrement DES cbc: 18s 3DES: 31s RC2 ecb : 19s AES 128 ecb: 3s

11. on voit nettement qu'AES est le plus rapide, avec 3DES le plus lent et DES/RC2 équivalents.

12. De façon générale, CBC est plus lent que EBC. CBC est parallélisable alors qu'EBC non.

13. L'option -K va permettre de récupérer une clé à partir d'un fichier, en utilisant une clé 36D1456C26A3670D et un IV FB22881684E1864D, le chiffrement marchera donc car l'algorithme ajoutera automatiquement du padding.

2 Chiffrement asymétrique

1. Afin de générer une paire de clé publique/privée ((n/e) clé publique), ((n/d) clé privée), utilisable pour RSA ($C^d mod P$), nous pouvons utiliser les commandes:

```
openssl genrsa -out PrivateKeyTristan 2048
```

```
openssl rsa -in PrivateKeyTristan -pubout -out PublicKeyTristan
```

Le chiffrement asymétrique est assez lent, le chiffrement symétrique est plus rapide, donc en pratique on utilise le chiffrement hybride. Chiffrement asymétrique: on chiffre la clé par chiffrement asymétrique et

le message en chiffrement symétrique. Example: fichier est chiffré en AES et on envoie la clé symétrique AES au destinataire via une méthode asymétrique type RSA afin qu'il récupère la clé et déchiffre le fichier.

Chiffrement

```
openssl rsautl -encrypt -pubin -inkey public.key -in plaintext.txt -out encrypted.txt
```

Déchiffrement

```
déchiffrer: openssl rsautl -decrypt -inkey private.key -in encrypted.txt -out plaintext.txt
```