TP
« initiation à la sécurité informatique »

(Ph. Truillet)

Février 2018 – v. 1.5

1. introduction

Durant ces travaux pratiques, nous allons travailler avec le système d'exploitation Debian sous Raspberry Pi et nous aborderons quelques éléments et outils de la sécurité : intégrité de données, attaque par force brute, cryptage et stéganographie.

2. empreintes MD5 / SHA-1

En cryptographie, MD5 (**M**essage **D**igest algorithm 5 défini par Ronald Rivest en 1991 au MIT) et SHA-1 (**S**ecure **H**ash **A**lgorithm 1 défini par la NSA en 1993) sont des fonctions de hachage fréquemment utilisées pour vérifier l'intégrité des fichiers.
Devenu un standard internet (**`http://www.ietf.org/rfc/rfc1321.txt`**), MD5 est employé dans beaucoup d'applications de sécurité et permet rapidement de s'assurer de l'intégrité de fichiers (utile lors de téléchargement). Ainsi en comparant l'empreinte md5 du fichier que vous venez de récupérer à l'empreinte que le site de téléchargement vous donne, vous pouvez vous assurer que vous possédez le même fichier.

Préambule

`md5sum` est un outil de calcul d'empreinte déjà présent sur les systèmes linux.

Installez ensuite 7zip sur votre machine en tapant **`sudo apt-get install p7zip`**

Exercices

1. Téléchargez les fichiers images **`canal.jpg`** et **`pigeonnier.jpg`** sur le site **`http://www.irit.fr/~Philippe.Truillet/ens/cyber.php`**
   (commande `wget https://www.irit.fr/~Philippe.Truillet/ens/ens/m2ifcissd/fichiers/canal.jpg` par exemple)
2. Comparez les fichiers reçus avec les empreintes MD5 données sur le même site (fichier **`md5sum.txt`**)
3. Qu'en concluez-vous ?
4. Que feriez-vous ?

3. GPG

GnuPG (**G**NU **P**rivacy **G**uard) est l'implémentation GNU (**G**NU's **N**ot **U**nix) du standard OpenPGP (format de cryptographie à clés publique/privée) normalisé par la RFC 4880) et disponible dans les distributions Linux. GPG permet de signer et de chiffrer tout document numérique.
Nous allons commencer par générer la paire de clés publiques/privées en utilisant la commande **`gpg –gen-key`**

Choisir successivement les valeurs suivantes : 1 (RSA et RSA) / 2048 (taille de la clé) / 0 (pas d'expiration de la clé) / o (Oui) / (votre nom) / (votre adresse) / (éventuel commentaire) / (votre phrase secrète) / (encore votre phrase secrète) / (faire des choses pour générer de l'entropie) puis attendre …

Vérifiez ensuite que les clés ont bien été créées avec la commande **`gpg –list-keys`**
La commande **`gpg –fingerprint`** va nous permettre de trouver l'empreinte (*fingerprint*) de la clé publique

Il va falloir maintenant stocker sur un serveur notre clé publique avec de pouvoir recevoir par exemple de méls chiffrés.
Créons d'abord un certificat de révocation (au cas où) avec la commande **`gpg –gen-revoke (votre adresse mél) > revoc_(votre adresse mél).txt`** Répondez aux questions posées.
Ensuite, envoyez votre clé publique sur le serveur (par exemple **`pgp.mit.edu`**) en utilisant :
`gpg --keyserver pgp.mit.edu --send-keys (clé)`

Nous allons maintenant chiffrer notre premier fichier. Ouvrez un document (par exemple en utilisant nano mondoc.txt), écrivez du texte puis sauver.
Pour chiffrer, rien de plus simple, utilisez la commande : **`gpg -er (votre adresse mél) mondoc.txt`** (**-e** signifie que vous allez chiffrer – *encrypt* – et **-r** que vous allez utiliser la clé publique spécifiée après)
Normalement, vous devriez avoir un fichier de type **`mondoc.txt.gpg`** qui est apparu ☺

Pour déchiffrer un texte, il faut avoir la clé secrète ce qui tombe bien puisque vous la possédez ! Pour déchiffrer, rien de plus simple, il suffit de taper **`gpg mondoc.txt.gpg`**

Chiffrez un texte et envoyez-le à quelqu'un. Que faire pour que cela fonctionne et que la personne puisse le déchiffrer ?

4. stéganographie

La stéganographie est « *l'art de la dissimulation* ». Cela consiste à cacher un message/fichier dans un objet anodin. Il y a de nombreux exemples d'usage de cette technique depuis l'antiquité (tatouage sur le crane, encre invisible, micropoint, …)

L'informatique permet d'affiner cette technique en ajoutant soit des fichiers d'apparence anodine un autre fichier (utilisation par exemple de la technique LSB – Least Significant Bit)

Exercice

Commençons par installer **`steghide`** en tapant : **`sudo apt-get install steghide`**

Téléchargez les fichiers images **`canal_midi.jpg`** sur le site **`http://www.irit.fr/~Philippe.Truillet/ens/cyber.php`**

Vous avez appris par ailleurs que la passphrase utilisée est « *canal* ».
Vérifiez que la photo contient bien du contenu caché en utilisant **`steghide info canal_midi.jpg`**

Extrayez ensuite le contenu en tapant : **`steghide extract -sf canal_midi.jpg`**

5. codage RSA (Rivest, Shamir, Adleman, 1977)

RSA (du nom de ses inventeurs et un codage de cryptographie dit asymétrique (à clé publique et clé privée). RSA permet d'assurer la confidentialité (seul le propriétaire de la clé privée peut déchiffrer le message reçu) et la non-altération et non-répudiation. Seul le propriétaire de la clé privée peut signer un message. Une signature déchiffrée avec la clé publique permet de prouver l'authenticité du message. L'algorithme étant public, sa difficulté réside dans la difficulté à factoriser un grand nombre.
Une clé RSA de 768 bits a pu être craquée en 2010, l'ANSSI préconise l'emploi de clés RSA de 2048 bits.

Fonctionnement

Bob possède un message confidentiel qu'il souhaite transmettre à Alice. Alice construit deux clés :

- une clé de chiffrement publique qu'elle transmet à Bob.
- une clé de déchiffrement privée qu'elle conserve soigneusement.

Bob utilise la clé publique pour chiffrer le message, et le transmets à Alice. Alice utilise la clé privée pour déchiffrer le message reçu.

Génération des clés

Soient deux grands nombres premiers « *aléatoirement* » choisis : *p* et *q*. Notons ***n = p.q*** et ***phi(n) = (p-1).(q-1)***
Soient ***d*** un grand entier « aléatoirement » choisi **premier avec phi(n)** et ***e*** l'inverse de **d modulo phi(n). (**soit **e.d = 1 mod phi(n))**

La clé publique de chiffrement est le couple **(n,e)**, la clé privée de déchiffrement le couple **(n,d).**

Exercice

René utilise RSA et publie sa clé publique (**n=187**, **e=3**)

1. Encoder le message (ASCII) suivant : **BOB** avec la clé publique de René

   Rappel : chiffrement du caractère **M → $C=M^e \bmod n$**

2. En utilisant le fait que l'indicatrice d'Euler **phi(n)=(p-1).(q-1)** est égal à 160 dans notre cas, retrouvez les facteurs premiers **p** et **q** (permettant le calcul de la clé secrète de René)

6. attaque de mots de passe par force brute

Nous allons maintenant essayer de cracker un mot de passe par force brute (essai de toutes les combinaisons possibles). Il existe de nombreux logiciels permettant de tester toutes les combinaisons, alliant usage de la puissance du processeur comme celle de la carte graphique (GPU).
Téléchargez le fichier **`perdu.zip`** à l'adresse **`http://www.irit.fr/~Philippe.Truillet/ens/cyber.php`**.

Par « chance », vous avez appris par ailleurs que le mot de passe est au maximum composé de 5 éléments et seulement composé de caractères en minuscules … Trouvez-le ! ☺ … Enfin, vérifiez le contenu !

Calculez le nombre d'essais à effectuer maximum pour cracker le mot de passe. Estimer ce nombre d'essais si on utilise des majuscules. Qu'en concluez-vous ?

Pour essayer de *décrypter* le mot de passe, installons le logiciel fcrackzip (outil de décryptage) : **`sudo apt-get install zip`**

Décodons ensuite le fichier en utilisant : **`fcrackzip -u -c a -p aaaaa fichier.zip`**

-u permet d'essayer le mot de passe trouvé avec unzip
-c a indique que vous allez utiliser tous les caractères en minuscules pour essayer de décrypter le mot de passe
-p aaaaa indique que vous voulez décoder jusqu'à 5 caractères

Nota : Si vous désirez protéger votre fichier zip, utilisez la commande : **`zip -e fichier.zip fichiers_à_zipper`**
Vous devrez certainement installer zip en utilisant : **`sudo apt-get install zip`**

7. exercice général

En sachant que pour encoder un fichier, il faut utiliser la commande **`steghide embed -cf fichier.jpg -ef fichier_à_cacher`**,

- Créez un fichier zip contenant n'importe quoi protégé par un mot de passe de 5 lettres.
- Créez un deuxième fichier zip protégé par un autre mot de passe (à vous de décider lequel avec la taille de votre choix) contenant votre clé publique.
- Cachez votre fichier zip dans une image jpeg en utilisant le premier mot de passe
- Partagez deux images jpeg dont l'une contenant le message caché et le premier fichier zip

- Récupérez le travail d'un de vos collègues et essayez de retrouver la clé publique partagée !
- Codez un message secret avec la clé publique trouvée et la transmettre à votre collègue

- Récupérez le message codé et le décoder …

8. Aller plus loin

- distribution Kali Linux 2018.1, « *The most advanced penetration testing distribution* » (https://www.kali.org/downloads)

RSA

Partie 1 - Encoder le message encodé en ASCII **msg = BOB** avec la clé publique de Bob

Chiffrement → $C=M^e \bmod n$

B se code 66 en ASCII
O se code 79 en ASCII

Bob utilise RSA et publie sa clé publique (**n=187**, **e=3**)

B= 66^3 mod 187 = 77
O = 66^3 mod 187 = 107

Message = **77 107 77**

Partie 2 - En utilisant le fait que l'indicatrice d'Euler **phi(n)** est égal à 160, retrouver les facteurs premiers **p** et **q** de **n= p.q**

Déchiffrement : il s'agit de calculer la fonction réciproque $M = C^d \bmod n$ tel que $e.d= 1 \bmod ([p-1)(q-1)]$

phi(n) = (p-1).(q-1) = pq-p-q+1 = n –(p+q)+1
Soit (**p+q**) = n – phi(n) + 1 = 187-160+1 = **28**

On a donc :
p.q = 187
p+q = 28

→ p.(28-p) = 187
→ p^2-28p +187 = 0
delta = $(-28)^2$ - 4.1.187 = 784 - 748 = 36

soit p = (-b - sqrt(delta))/(2.a) = 11
q = 17