P.Lafourcade, O.Allabwani

Session 5

Instructions : Vous devez envoyer un compte rendu de ce TP à `osama.allabwani@limos.fr` avant 2026. Le fichier doit suivre le format suivant : `LASTNAME-Firstname.pdf`. *Une image vaut mille mots*, vous pouvez bien sûr utiliser des captures d'écran tant que la taille du PDF est inférieure à 5 Mo. Attention :

- **Le sujet du mail doit être "[SSI - TP] Compte Rendu"**
- **Le mail doit être envoyé en utilisant votre adresse e-mail UCA ! (prénom.nom@etu.uca.fr)**

Exercise 1 (L'attaque au président)
Bienvenue dans l'entreprise **Crypto**, en tant que secrétaire. Le président est actuellement en déplacement. Dans cet exercice, vous jouerez en alternance le rôle de la secrétaire et du président, en ajoutant et retirant les clés. Le rôle que vous devrez incarner est présenté entre paranthèse au début de la question. Téléchargez l'archive à l'adresse suivante:

`https://sancy.iut.uca.fr/~lafourcade/SECU-3A/gpg-president.tar`

Attention ! L'outil que l'on va utiliser, `gpg`, est capable d'enregistrer plusieurs clés en même temps. Lorsque un rôle est spécifié, *seule la clé secrète associée à ce rôle* devra être présent. La clé secrète du président se trouve dans le fichier `keys/president.prv`, tandis que sa clé publique se trouve dans le fichier `keys/president.pub`. La clé secrète de la secrétaire se trouve dans le fichier `keys/secretary.prv` tandis que la clé publique est dans `keys/secretary.pub`. La clé secrète du président est protégée par le mot de passe `president`, et la clé secrète de la secrétaire par `secretary`.

1. Ajouter la clé secrète de la secrétaire.
2. (Secrétaire) Vous recevez un message chiffré de le part du président, situé dans le dossier `q1/message1.txt.gpg`. Découvrer le contenu du message.
3. Ajouter la clé publique de la présidente.
4. (Secrétaire) Quel drôle de message ! Assurez-vous qu'il ne s'agisse pas d'une erreur, en chiffrant *et* signant le message `q2/message2.txt`.
5. Retirer la clé secrète de la secrétaire et ajouter la clé secrète du président.
6. (Président) Rien ne va pas plus, contacter immédiatement la secrétaire pour lui indiquer qu'il s'agit d'une erreur, et même d'une arnaque ! Pour cela, signez le message `q3/message3.txt`. Puisque le message ne contient pas d'information sensible, il sera inutile de le chiffrer.
7. Retirer de nouveau la clé secrète du président et ajouter la clé secrète de la secrétaire.
8. (Secrétaire) Vérifier le message que vous venez de signer.
9. Par une source anonyme, nous sommes parvenu à trouver une clé publique située dans `keys/.attacker.pub`. Par chance, le message demandé le versement de façon scrupuleuse a été signée par l'attaquant ! Avant toute chose, il convient de s'assurer qu'il s'agisse de la bonne clé. Commencer par ajouter la clé publique dans `gpg`.
10. Vérifier ensuite la signature qui se trouve dans `q4/signature.txt.sig`.
11. Quel heureux hasard, la clé publique contient beaucoup d'information ! Donnez l'adresse email, et son visage.

Pour vous aider, nous vous fournissons les commandes `gpg` essentielles:

- `gpg --list-keys`: Liste l'ensemble des clés publiques enregistrées.
- `gpg --list-secret-keys`: Liste l'ensemble des clés secrètes enregistrées.
- `gpg --import <key>`: Permet d'importer une clé (publique ou secrète) dans gpg.
- `gpg --encrypt -r alice message.txt`: Chiffre le contenu du fichier `message.txt` en utilisant la clé publique de chiffrement d'Alice. Produit le fichier `message.txt.gpg`.
- `gpg -r bob --sign --encrypt message.txt`: Chiffre et signe le contenu du message `message.txt`, en utilisant la clé de signature locale et la clé de chiffrement de bob.
- `gpg --sign message.txt`: Signe le contenu du message `message.txt`, en utilisant la clé de signature locale. Produit le fichier `message.txt.gpg`.
- `gpg --detach-sign message.txt`: Comme avant, mais la signature est déportée dans un autre fichier. Produit le fichier `message.txt.sig`.
- `gpg --verify message.txt.gpg`: Vérifie si la signature auprès de toutes les clés publiques enregistrées dans `gpg`. Si aucune clé ne correpond, la signature est considérée comme invalide. La commande ne fait que vérifier le message, pas l'afficher.
- `gpg --verify message.txt.sig message.txt`: Vérifie si la signature auprès de toutes les clés publiques enregistrées dans `gpg`. Si aucune clé ne correpond, la signature est considérée comme invalide. La commande ne fait que vérifier si la signature est valide, pas afficher le message.
- `gpg -o output.txt --decrypt message.txt.gpg`: Déchiffre et/ou vérifie la signature d'un message `message.txt.gpg` et produit le claire dans le fichier `output.txt`.
- `gpg --list-options show-photos --fingerprint keyID`: Affiche la photo associée à la clé publique, s'il y en a. Le paramètre `keyID` correspond à l'identifiant de la clé et doit être récupéré via l'option `--list-keys`.

Exercise 2 (HTTPS)
Dans cet exercice, vous allez configurer un site web Python 3, acceptant les connexions via le protocole HTTPS.

1. Téléchargez le dépôt à l'adresse suivante :

   `https://gitlab.limos.fr/gamarcad-teaching/ssi-tp-certificates`

   et se déplacer dans le projet.

2. Créez les certificats avec `make certificates` et lancez le serveur web Python 3 avec `make server`.
3. Ouvrez votre navigateur à l'adresse `https://localhost:4443`. Qu'obtenez-vous ?
4. Ajoutez les certificats `root.pem` et `dsi.pem` (générés à la racine du projet `ssl-tp-certificates`) dans votre navigateur préféré en tant que nouvelle autorité.
5. Rafraîchir la page de votre navigateur à `https://localhost:4443`.
6. Modifiez le fichier `server_req.config` afin d'obtenir un certificat de site web dont le nom commun (champ `CN`) suffixé par `-NOM-Prenom`. Prenez une capture d'écran du certificat dans votre navigateur. Remarque : Vous devez répondre à nouveau à la question 2 pour que les modifications soient prises en compte.

Exercise 3 (OpenSSL)
Génération de clés

1. Générez une paire de clés RSA de longueur 4096 avec la commande suivante :

   `openssl genrsa -aes256 -out rsakey.pem 4096`

2. Extrayez la clé publique du fichier `rsakey.pem` avec la commande suivante :

   ```
   openssl rsa -in rsakey.pem -pubout -out public.pem
   ```

3. Les informations relatives au modulo et aux nombres premiers utilisés sont cryptées avec AES-256 dans `rsakey.pem`. Utilisez la commande suivante pour trouver N, p, q tels que $N = p \cdot q$.

   ```
   openssl rsa -in rsakey.pem -noout -text
   ```

4. Expliquez comment vous pouvez générer des valeurs de p et q.

Chiffrement symétrique

1. Créez un fichier texte `message.txt` contenant un message de votre choix. Utilisez la commande suivante pour chiffrer ce message à l'aide de `openssl`:

   ```
   openssl enc -aes-256-cbc -in message.txt -out cipher.txt
   ```

   Envoyez ce message chiffré à un autre groupe avec le mot de passe. Bien sûr, en pratique, on n'envoie pas le mot de passe. Proposez deux méthodes sécurisées.

2. Déchiffrez le chiffré reçu `cipher.txt` d'un autre groupe avec la commande suivante :

   ```
   openssl enc -aes-256-cbc -in cipher.txt -d -out m.txt
   ```

Chiffrement asymétrique

1. Utilisez la commande suivante pour chiffrer le même message `message.txt` à l'aide de RSA :

   ```
   openssl rsautl -encrypt -in message.txt -inkey rsakey.pem -out cipher.txt
   ```

   Quelles sont vos observations par rapport au chiffré obtenu avec un chiffrement symétrique ?

2. Déchiffrez le chiffré `cipher.txt` avec la commande suivante:

   ```
   openssl rsautl -decrypt -in cipher.txt -inkey rsakey.pem -out m.txt
   ```

Comparaison

1. Téléchargez un fichier de quelques mégaoctets et chiffrez-le à l'aide des algorithmes AES-256-CBC et RSA. Que pouvez-vous dire concernant le temps de chiffrement ?

Signature

1. Pour signer un message, nous utilisons un système de chiffrement à clé public (ici RSA). Utilisez la commande suivante pour signer le message `msg.txt` avec l'algorithme RSA :

   ```
   openssl rsautl -in msg.txt -inkey rsakey.pem -sign -out signed.txt
   ```

2. Vérifiez la validité de la signature avec la commande suivante :

   ```
   openssl rsautl -verify -in signed.txt -inkey rsakey.pem -out check.txt
   ```

Créer un certificat racine

1. La première étape consiste à créer une clé pour une autorité racine. Pour ce faire, nous utilisons la commande suivante : `openssl genrsa -out rootca.key`

   Quelle est la taille de cette clé RSA ? Que conseillez-vous ?

2. Il est également possible de choisir la valeur aléatoire utilisée pour générer la clé. Pour créer une valeur aléatoire de 8192 octets, utilisez la commande suivante : `sudo openssl rand -out noise 8192`

3. Utilisez ce nombre aléatoire pour créer une nouvelle clé RSA de taille 4096 chiffré avec AES-256 à l'aide des commandes suivantes :

   ```
   export RANDFILE=noise
   ```

   ```
   openssl genrsa -out rootca3.key -aes256 4096
   ```

   Conseil : supprimez le fichier aléatoire `noise` pour vous assurer qu'un adversaire ne puisse pas l'utiliser.

4. Nous allons maintenant créer un certificat racine autosigné `rootca.crt` associé à la clé `rootca.key`. Pour ce faire, nous utilisons la commande suivante :

   `openssl req -new -x509 -days 3650 -key rootca.key -out rootca.crt`

   Cette commande demande des informations sur l'identité de l'entité qui l'exécute. Veuillez saisir les informations d'un site web fictif de votre choix.

   Ce certificat racine a une durée de vie de 3650 jours et sera au format x509.

5. Vous pouvez consulter les informations du certificat créé à l'aide de la commande suivante :

   `openssl x509 -text -in rootca.crt`

   Expliquez l'utilité d'un tel certificat.

Exercise 4 (Tor)
Une adresse IP fournit de nombreuses informations sur la localisation de l'utilisateur. Elle permet aux sites web d'adapter la langue ou de rediriger l'utilisateur vers un serveur plus proche, mais elle peut aussi servir à suivre l'activité des internautes. Pour protéger notre anonymat, nous pouvons utiliser le navigateur Tor.

1. Installez le navigateur Tor sur votre ordinateur. Vous pouvez le télécharger à l'adresse suivante: `https://www.torproject.org/`

2. Accédez à un site web de votre choix en utilisant Tor et un autre navigateur (Firefox ou Chromium, par exemple). Que constatez-vous ? Expliquez pourquoi.

3. Dans Tor, rendez-vous sur `https://en.wikipedia.org/wiki/Main_Page` et cliquez sur le cadenas vert. Vous pouvez observer le circuit utilisé par votre navigateur pour vous connecter au site web. Comparez ce circuit à celui utilisé par un autre groupe. Qu'observez-vous ?

4. Dans Firefox (ou Chromium), accédez à : `https://iplookup.flagfox.net/` Accédez au même site web avec le navigateur Tor. Que remarquez-vous ?

Nous souhaitons maintenant accéder au Dark Web pour lire un article intéressant intitulé "How to Exit the Matrix?".

6. Dans Firefox, accédez à `https://thehiddenwiki.org/` Ce site Web fait référence à des liens dans *.onion*.

7. Toujours dans Firefox, cliquez sur le lien correspondant à "The Hidden Wiki". Expliquez pourquoi vous ne pouvez pas accéder à ce site web depuis Firefox.

8. Dans le navigateur Tor, copiez le lien *.onion* de "The Hidden Wiki". Vous constatez alors que vous pouvez accéder au wiki : vous êtes sur le dark web (incroyable !).

9. Dans "The Hidden Wiki", Vous pouvez cliquer sur le numéro 1 "The Matrix" dans la section "editor's picks".

Exercise 5 (Qualys)
Qualys, Inc. est un fournisseur de services de sécurité cloud, de conformité et de services connexes pour les petites et moyennes entreprises. Elle propose un service d'évaluation de la sécurité des sites web (`https://www.ssllabs.com/ssltest/`) qui attribue une note au site (de F à A+).

1. Comparez les niveaux de sécurité de `https://www.fnac.com/` et `https://limos.fr`.

2. Quelle configuration (navigateur Web / système d'exploitation) ne permet pas une connexion sécurisée à `https://limos.fr`?

3. Testez la sécurité de trois banques en ligne différentes. Essayez d'en trouver une avec la note $A+$ et une autre avec la note B ou C.