P. Lafourcade, M. More, M. Puys

Securité Web
2026/2027

TP1 : Rappels Cryptographie

Rappel : `proxycl.iut.uca.fr` (192.168.128.139), port 8080

Exercise 1 (http vs https (2 points))

1. (2 points) Faire l'analyse sur les dumps `.pcapng` accessibles ici :

   `https://sancy.iut.uca.fr/~lafourcade/SECU-3A/data/`

   Qu'observez-vous ?

   Il est possible de filtrer les trames avec les adresses IP pour avoir moins de bruit `ip.addr == 192.168.128.30` et aussi par protocole.

Exercise 2 (Brute-force locale (3 points))

Récupérez sur ce site[1] la base de données utilisateurs (à extraire avec `bunzip2 phpbb.txt.bz2`). Vous y trouvez des bases de données de mots de passe qui ont fuité ces dernières années.

1. (1 point) `$1$AAAA$H9wXcd/WaaomJUgWKFspy.` est un mot de passe hashé et salé, retrouvez l'agorithme de hachage utilisé, identifiez le sel (`rtfm openssl`[2]). Avec `openssl` et la commande `passwd`, vérifiez que le mot de passe est `!!1331xxx`

2. (2 points) Quels sont les mots de passe associés à `$1$BABA$DOzBWHNx08SgVSX/YuYvC/` et à `$1$CACA$XLWo4OqFFCYICqYrZ0y5i/` (à partir de `phpbb`).

   que vos calculs sont corrects.

Exercise 3 (L'attaque au président (6 points))

Bienvenue dans l'entreprise **Crypto**, en tant que secrétaire. Le président est actuellement en déplacement. Dans cet exercise, 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 qu'on va utiliser, `gpg`, est capable d'enregistrer plusieurs clés en même temps. Lorsqu'un rôle est spécifié, *seule la clé secrète associée à ce rôle* devra être présente dans `gpg`. 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 sa 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 dans `gpg`.

[1] À l'adresse `https://sancy.iut.uca.fr/~lafourcade/phpbb.txt.bz2`

[2] `openssl passwd -1 -salt BBBB 'Toto'`

2. (Secrétaire) Vous recevez un message chiffré de la part du président, situé dans le dossier `q1/message1.txt.gpg`. Découvrez le contenu du message.

3. Ajouter la clé publique du président.

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) Vérifiez le message que le président vient de signer.

9. Par une source anonyme, nous sommes parvenus à trouver une clé publique située dans `keys/.attacker.pub`. Par chance, le message demandant le versement de façon frauduleuse a été signé par l'attaquant ! Avant toute chose, il convient de s'assurer qu'il s'agisse de la bonne clé. Commencez par ajouter cette clé publique dans `gpg`.

10. Vérifiez 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 de l'attaquant, 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 4 (Votre banque favorite (4 points))
Il existe un certain nombre de vulnérabilités pour SSL/TLS qui nécessitent une maintenance constante et des mises à jour réguliéres des serveurs. Ce n'est malheureusement pas toujours le cas. La société Qualys propose sur le site *SSL Labs* le projet *SSL Test*, qui permet de tester un serveur HTTPS contre les vulnérabilités connues du protocole.

1. Allez sur la page d'accès aux comptes de votre banque favorite (**sans vous identifier** ; la page de connexion suffit bien). Vérifiez que vous êtes bien connectée en HTTPS.
2. Allez sur le site `https://www.ssllabs.com/ssltest/`, rentrez le nom de domaine précédent, puis validez. Cela prend un peu de temps, puis vous allez obtenir une note et un compte-rendu détaillé.
3. Analysez les rapports de 4 sites de banques et établissez un classement en le justifiant.

Exercise 5 (Digicode (2 points))
Pour rentrer chez lui Bob doit taper un code à 4 chiffres sur un digicode qui possède 10 chiffres et deux lumières : une rouge et une verte. A chaque fois qu'il tape un chiffre faux, la lumière rouge s'allume et il recommence au début, alors que s'il tape le bon chiffre, la lumière verte s'allume et il passe au chiffre suivant. Le problème est que Alice s'amuse régulièrement à changer le code.

- Quel est le nombre de codes que peut générer Alice ?
- Combien d'essais que devra faire Bob s'il s'y prend bien pour ouvrir la porte ?

Pour vous entraîner : `https://sancy.iut.uca.fr/~lafourcade/naive-digicode/`

Exercise 6 (CVE (1 point))
Nous allons utiliser `vdn`. La documentation est disponible :
`https://codefirst.iut.uca.fr/gitlab/IUT_INF63/vdn/vdn-doc`
Choisir le réseau `secure` et démarrer la machine `debian-1`
Dans VDN, téléchargez cette archive (avec `wget`) :
`https://sancy.iut.uca.fr/~lafourcade/SECU-3A/xray.tar.gz`.
Décompressez-la (`tar zxf`), puis lancez `docker-compose up`[3] dans le répertoire `gehealth` pour obtenir le site `http://localhost:21346` (ouvrez `firefox` dans `vdn-ssh -X`).
Trouvez une CVE (*Common Vulnerabilities and Exposures*) pour pénétrer sur ce site avec 3 identifiants différents.

[3]Le - est nécessaire parce-que la machine virtuelle est vieille...

Bonus

Exercise 7 (Malléabilité (5 points))
Les standards de chiffrement symétrique comme AES chiffrent avec une clé symétrique des messages de longueur 128 bits. Afin de chiffrer des messages plus grands, de nouveaux modes de chiffrement ont vu le jour. La façon naturelle de chiffrer un grand message consiste à le scinder en blocs de 128 bits et de les chiffrer les uns après les autres. Cette technique s'appelle ECB, mais n'assure pas une grande sécurité, car deux blocs identiques sont chiffrés par le même message.

Afin de pallier ce problème, le mode de chiffrement CBC a été proposé par le NIST. Il fonctionne comme suit, où IV est un vecteur initial, K est la clé de chiffrement symétrique, $x \oplus y$ est le XOR entre x et y, et $E_K(m)$ est le chiffré du message m avec la clé K :

$$C_0 = IV \text{ et } C_i = E_K(P_i \oplus C_{i-1}).$$

Un attaquant peut attaquer un schéma de chiffrement dans différents buts. L'objectif le plus évident est de connaître les messages échangés entre deux personnes. Une autre attaque, par ailleurs, vise la perturbation par l'attaquant d'un message échangé (pour changer, par exemple, le sens d'une phrase). Ce type de perturbation s'appelle *attaque par malléabilité*.

Pour comprendre la puissance d'une attaque exploitant la malléabilité d'un schéma de chiffrement, il suffit d'observer que, entre les messages M_1 et M_2 ci-dessous, il n'y a pas beaucoup de différence :

$$M_1 = \texttt{Je confirme l'achat de deux pizzas.}$$
$$M_2 = \texttt{Je confirme l'achat de deux cents pizzas.}$$

Ici nous considérons le mode de chiffrement CBC. Supposez que vous connaissez la valeur de P_1 pour un chiffré donné $IV||C_1||C_2||C_3||C_4$....

Vous contrôlez aussi la valeur initiale IV'. Essayez de construire un second message chiffré avec la même clef : $IV'||C'_1||C'_2||C'_3||C'_4$... tel que $C'_i = C_{i-2}$ pour i supérieur ou égal à 3. Votre objectif est de remplacer le texte chiffré originel par un autre texte chiffré, pour lequel les valeurs de certains blocs de texte clair (P'_1 et P'_3) sont choisis par l'attaquant.

Saurez-vous trouver les valeurs de IV', C'_1, C'_2 et C'_3 telles que P'_1 et P'_3 sont fixés ? Prenez par exemple $P'_1 = 0111\ 1111\ 0111$ et $P'_3 = 1000\ 1001\ 1011$ si cela peut vous aider.