# Réseau et Sécurité
# TP 3 - Authentification

Le TP nécessite une machine virtuel.

## 1 Votre banque favorite

Il existe un certain nombre de vulnérabilités pour SSL/TLS qui nécessitent une maintenance constante et des mises à jour régulières 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'accés aux comptes de votre banque favorite (**sans vous identifier** ; la page de connexion suffit bien). Vérifiez que vous étes 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 (pouvant aussi être à l'étranger) et établissez un classement en le justifiant.

## 2 Sécurité SSH

Clonez le dépôt `https://github.com/jtesta/ssh-audit`

1. Dans votre machine virtuel, scannez votre serveur SSH local 127.0.0.1 avec ssh-audit. Décrivez les résultats.
2. Quel est version du serveur SSH tourne sur la machine ?
3. Modifiez le fichier `/etc/ssh/sshd_config` de votre machine virtuel de façon à corriger les faiblesses détectées par ssh-audit. Les mots clefs à utilisés sont : `Ciphers`, `HostKeyAlgorithms`, `MACs` and `KexAlgorithms`. Faites en sorte de ne supporter que les méthodes sans vulnérabilitées.

Il faut utiliser `systemctl retart sshd` pour relancer le service à chaque modification.

## 3 CVE

**Dans une machine virtuel**, télécharger cette archive `https://perso.isima.fr/~chaolivi/reseausecu/xray.tar.gz`, puis lancer `docker-compose up` dans le répertoire `gehealth` pour obtenir le site `http://localhost:21346` (ouvrez `firefox`).

Trouver une CVE pour pénétrer sur ce site avec 3 identifiants différents.

## Exercise 4 Malléabilité

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 palier 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 considerons 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 controlez 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 original 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.

## 5 EFAIL

Un peu de préparation sera nécessaire avant de commencer.

1. **Extraction et Installation** : Téléchargez l'archive suivante :

   `wget https://perso.isima.fr/~chaolivi/reseausecu/tp3.tar.xz`

   Vous allez l'extraire avec `tar -xvf tp3.tar.xz`, puis aller dans le dossier `ressec25-tp3` et lancer les commandes suivantes :

   ```
   apt update
   pip install python3-requests python3-cryptodome
   ```

   À ce stade, vérifiez que votre prompt sur le terminal commance bien par `(ressec25-tp3)`.
2. **Lancement des serveurs (quand cela sera nécessaire)** : Dans un *autre terminal*, lancer les serveurs avec la commande suivante, en prenant bien soin de laisser les serveurs tourner :

   ```
   docker load -i ressec25-tp3.tar.xz
   docker run --rm -p 5001:5001 -p 5002:5002 -p 5003:5003 websec23-tp2
   ```

Aller dans le dossier `efail`. Votre objectif sera de modifier le mail chiffré (`mail.txt`), et de l'envoyer sur le serveur SMTP (`http://localhost:5003`) pour en révéler son contenu. Durant cet exercice, **seul** le fichier `efail_exercice.py` devra être modifié, aucune autre modification n'est nécessaire. L'objectif est de compléter le fichier `efail_exercice.py` pour monter l'attaque EFAIL vue en cours.

1. Installer les dépendances nécessaires avec `pip3 install requests pycryptodome` si ce n'est pas déjà la cas.
2. Désactiver les proxy `http_proxy` et `https_proxy`.
3. Consulter vos mails en allant sur la page `http://127.0.0.1:5003`, utilisez un navigateur classique.
4. C'est à cette étape que votre travail commence. Le mail que dont vous souhaitez voir le contenu se trouve dans `mail.txt`. Dans cet exercice, vous n'allez modifier seulement le fichier `efail_exercice.py`, qui contient la fonction

`def efail( iv, ciphertext, known_plaintext , begin_replace, end_replace )` où `iv` est le vecteur d'initialisation, `ciphertext` est le chiffré, `known_plaintext` est le texte clair connu, tandis que `begin_replace` et `end_replace` sont respectivement le message à remplacer au début et à la fin du message.

La fonction doit retourner un couple `(iv', ct')` où `iv'` sera le nouveau vecteur d'initialisation et `ct'` le nouveau chiffré. La fonction est partiellement codée, il ne vous faudra que remplacer les parties entre chevrons (`< .. >`).

Pour vous éviter de prendre du temps, nous vous fournissons `efail.py`, qui se chargera de lire le fichier `mail.txt`, de parser le chiffré, et d'appeler la fonction `efail` définie plus haut avec les bons paramètres. Pour exécuter le fichier `efail.py`, entrer la commande `python3 efail.py --mail mail.txt` Si votre fonction retourne le nécessaire, `efail.py` se chargera alors de générer le nouveau mail et de l'envoie au serveur SMTP. Le message sera alors visible dans les logs du serveur. Le voyez-vous ?