Travaux pratiques
SSL/TLS

Anaïs Gantet, Benoît Camredon

28 novembre 2019

Objectifs

— Observer divers types de messages échangés pour diverses versions de SSL/TLS
— Se familiariser avec `openssl` (parties client et serveur)
— Etudier un cas d'interception SSL/TLS et en comprendre les enjeux

1 Analyse de traces PCAP SSL/TLS

Des traces PCAP ont été placées dans la VM dans `~/cours/tls/tp/pcap`. Avec l'outil de votre choix (`Wireshark`, `scapy`, etc.), analyser les échanges contenus dans ces 6 pcaps et, pour chacun :

1. Donner la version de SSL/TLS utilisée
2. Dans l'une des diverses capture, récupérer, si présent, le certificat du serveur et en donner ses caractéristiques :
   — Qui est l'issuer ?
   — Qui est le subject ?
   — Quelle est sa date de validité ?
   — Par qui est-il signé ? Selon quel algorithme de signature ?

2 `OpenSSL` (`s_client` et `s_server`)

`OpenSSL` est un toolkit cryptographique implémentant, entre autres, les protocoles SSL/TLS. `openssl` est un programme en ligne de commande qui peut être utilisé pour :
— Créer et gérer des paires de clés publique/privée et leur paramètres
— Créer des certificats `X509`, des CSR ou des CRL
— Effectuer des tests avec un client SSL/TLS ou un serveur SSL/TLS
— Chiffrer et déchiffrer
— etc.

Les versions récentes d'`openssl` ne gérant pas forcément les anciennes versions d'SSL/TLS, une ancienne version d'`openssl` a été installée dans `~/cours/tls/bin/testssl.sh/bin/openssl.Linux.x86_64`. Dans la suite du TP, utiliser ce binaire plutôt que l'`openssl` installé dans la distribution actuelle.

2.1 Test de connexion avec `s_client`

`s_client` est une commande d'`openssl` implémentant un client SSL/TLS capable de se connecter à un serveur (host : port) donné. Exemple basique d'utilisation de `s_client` : `openssl s_client -connect www.google.com:443`

1. Utiliser `s_client` pour trouver se connecter sur le site de l'ENSEEIHT.
2. Quelle version d'SSL/TLS sont supportées par ce site ? (A déterminer avec `s_client`)
3. Utiliser `s_client` pour récupérer le certificat du site web de votre choix.

2.2 Déployement de son propre serveur `OpenSSL`

`s_server` est une commande d'`openssl` qui implémente un serveur SSL/TLS qui se met en écoute sur un port donné utilisant SSL/TLS. Les paramètres du serveur sont entièrement configurables :

— Clés du serveur
— Certificat du serveur
— Suites cryptographiques supportées
— Versions d'SSL/TLS supportées
— etc.

1. Commencer par générer une paire de clé et un certificat X509 avec `openssl req`
2. Démarrer un `s_server` avec la clé privée et le certificat créés précédemment
3. Initier une connexion avec `s_client` sur le serveur créé.
   — Quelle est la version d'SSL/TLS utilisé par défaut pour cet échange ?
   — Quelle est la suite cryptographique négociée ?
4. Les versions acceptées par le serveur sont configurables avec les options `-ssl2, -ssl3, -tls1, -tls1_1, -tls1_2, -no_ssl2, -no_ssl3, -no_tls1, etc.`. De même, les suites cryptographiques sont configurables avec l'option `-cipher`. Redémarrer votre `s_server` en restreignant les versions et suites supportées. Retenter des connexions avec `s_client`.

3 Interception `SSL/TLS`

Parfois, il peut être utile de faire de l'interception SSL/TLS. `mitmproxy` est un proxy mettant en oeuvre une interception SSL/TLS. Une version de l'outil se trouve dans `~/bin/mitmproxy/mitmproxy`.

1. Lancer `mitmproxy` sur le port de votre choix.

2. Une fois lancé `mitmproxy` se met à intercepter tout le trafic SSL/TLS qui passe. Avec `mitmproxy` activé, se connecter aux sites suivants depuis votre navigateur :
   — www.google.fr
   — www.leboncoin.fr
3. Quel est le problème observé ? Comment y remédier ?
4. Faire en sorte que l'interception SSL/TLS soit acceptée par votre navigateur.

4 Utiliser python pour faire une connexion SSL/TLS

1. Coder votre propre client en python capable de récupérer la page de `www.google.com`.