Vérification de chaînes de certificats

Projet de cryptographie

2024-2025

1 Introduction

Ce projet consiste à coder un logiciel d'analyse d'une chaîne de certificats X.509 pour valider (ou non) sa sécurité. Ce code pourrait constituer un module d'une pile TLS. Les algorithmes RSA et ECDSA doivent être supportés pour les signatures des certificats. Les certificats à analyser sont stockés unitairement dans des fichiers. Les formats DER et DER encodé en base 64 (PEM) doivent être supportés. Les spécifications du programme sont données progressivement ci-dessous. Il est fortement conseillé de travailler en mode incrémental et de bien s'assurer qu'une étape fonctionne avant de commencer à coder la suivante. A cette fin on effectuera des tests avec des certificats *valides* et *invalides*.

Les chaînes de trois certificats des sites `https://www.tbs-certificates.co.uk` et `https://www.lemonde.fr` doivent être validées à la fin du projet.

2 Prérequis

Vous devez disposer de certificats de test dans les formats et avec les signatures qui doivent être supportés. Le plus simple est de sauvegarder des certificats de sites web utilisant TLS depuis un navigateur. Des exemples de certificats invalides sont disponibles sur `badssl.com`. Il est également possible de créer des certificats avec `openssl`.

Le projet est à implémenter en Java (dans la suite des indications sont données sur les API à utiliser dans ce langage). Il est possible d'utiliser un autre langage mais les API utilisées doivent être de niveau équivalent. Vous devez comprendre chaque ligne de code de votre projet et insérer des commentaires détaillés.

3 Réalisation du projet

3.1 Validation d'un certificat d'autorité racine

Le but est de coder un programme qui s'utilisera de la façon suivante :

`validate-cert -format DER|PEM` *myRCAcertfile*

Seuls les certificats d'autorité racine sont supportés à ce stade. En sortie le programme fournira des informations extraites du certificat et un status sur leur sécurité avec un message d'erreur détaillé le cas échéant.

1. Coder la prise en compte des arguments de la ligne de commande, la lecture du fichier au format DER ou PEM et la création d'un objet de classe `java.security.cert.X509Certificate`.
2. Ajouter l'extraction de la clef publique du certificat auto-signé et la vérification de la signature avec `java.security.cert.X509Certificate.verify(PublicKey key)`.
3. Afficher le sujet et l'émetteur du certificat.
4. Vérifier l'extension `KeyUsage`.
5. Vérifier la période de validité.
6. Extraire l'algorithme de signature du certificat ainsi que la signature. Vérifier celle-ci avec l'API cryptographique (`Signature.verify(byte[ ])`). Notez que vous devrez traiter les signatures RSA et ECDSA.

3.2 Validation d'une chaîne de certificats

Le but est d'étendre le programme qui s'utilisera de la façon suivante :

`validate-cert-chain -format DER|PEM` *myRCAcertfile myICAcertfile . . .myLeafcertfile*

On suppose ici que les fichiers sont passés en argument dans l'ordre et sont tous du même format.

1. Coder la validation récursive de la chaîne de certificats en vérifiant toutes les signatures comme ci-dessus et la correspondance des sujets et des émetteurs.
2. Effectuer la vérification de signature RSA en utilisant une API de calcul sur les grands nombres au lieu d'une API de cryptographie (par exemple `java.math.BigInteger`).
3. Effectuer la vérification de signature ECDSA en utilisant une librairie de calcul sur courbe elliptique en plus de celle sur les grands nombres (par exemple `org.bouncycastle.math.ec.ECPoint`) au lieu d'une API de cryptographie.
4. Ajuster la vérification de l'extension `KeyUsage` suivant le niveau de certificat.
5. Effectuer la vérification de l'extension `BasicConstraints`.

3.3 Vérification du status de révocation

1. Ajouter la vérification du status de révocation en téléchargeant la CRL pour chaque certificat (attention à vérifier la validité de la CRL dont sa signature).
2. Ajouter la vérification du status de révocation en utilisant le protocole OCSP s'il est disponible pour un certificat donné.
3. Ajouter un mécanisme de cache pour ne pas télécharger une CRL s'il elle n'a pas été mise à jour.

Bien afficher les différentes vérifications effectuées ou la nature des erreurs.

3.4 Question bonus

Si *Le Monde* veut intenter un procès à son fournisseur de certificat, quelles sont les lois qui s'appliquent et quel tribunal est compétent ?

4 Compte-rendu du projet

Votre compte-rendu du projet comprendra les éléments suivants :
— Une introduction sur l'importance et l'intérêt des différentes vérifications à faire sur une chaîne de certificats ;
— Une explication de votre choix d'outils, de langage et de librairies ;
— Indiquez quelles étapes ont été effectivement implémentées, quels tests ont été effectués et leurs résultats (sorties de vos programmes) ;
— Vos autres choix techniques et la structure de votre programme ;
— Retour sur les difficultés rencontrées ;
— Les améliorations possibles ;
— Une liste des ressources utilisées pour ce projet (site web, code source, IA . . .) ;

4.1 Rendu du projet

Le projet doit être rendu sous forme d'une archive disponible sur un site de partage de fichiers de votre choix que vous me communiquerez par e-mail. L'archive comprendra les fichiers commentés de votre projet ainsi qu'un compte-rendu au format PDF.