Bugeaud Secure Certificate

UE – Sécurité des usages TIC
Enseignant – P-F Bonnefoi - E Conchon
Réalisation – Morgane Vollmer
Clément Hoffmann
Baptiste Beltzer

Projet Git : https://github.com/maxilix/Projet_TIC

Contextualisation

Pour ce projet, nous avons choisi de nous présenter en tant que prestataire de service fournissant, auprès d'entités comme *CertifPlus*, la possibilité de créer des attestations au format image, des certificats de réussite. Pour utiliser notre service, les clients doivent être inscrits dans notre base de données, et avoir partagé un secret avec nous. Notre programme est un serveur traitant des requêtes de génération ou de vérification d'un certificat.

Fonctionnalités

1) OTP et serveur

Notre programme lance un serveur qui va attendre des requêtes clients. Un processus fils est créé pour chacune des requêtes. Tout d'abord, le client devra envoyer son nom, et le serveur vérifiera qu'il est bien dans notre base de données et récupérera le secret partagé. Le client et le serveur génère ensuite séparément un OTP à partir du secret partagé. Si les OTP correspondent, le client est reconnu comme légitime (étant le seul à partager ce secret avec nous). Cette méthode permet de ne pas envoyer le secret sur le réseau (la fonction `CreerOTP(secret)` est dupliquée dans les fichiers `server.py` et `client.py`).

Le client a ensuite le choix entre deux services (détaillés par la suite) : la génération ou la vérification d'un certificat. S'il souhaite générer un certificat, il doit envoyer les données nécessaires (nom, prénom, adresse-mail et intitulé du certificat). S'il souhaite en vérifier un, il doit l'envoyer encodé en base 64.

2) Générer un certificat

Après avoir sélectionné l'option de génération de certificat, le client (*CertifPlus*) envoie au serveur les informations personnelles de l'utilisateur devant recevoir son certificat. Par la suite, la fonction `CreerAttestation(client,informations)` se charge de construire l'image PNG du certificat qui sera envoyée à l'utilisateur. Cette fonctionnalité se décompose en quatre étapes :

- on vérifie si le certificat existe déjà à l'aide du nom de dossier `../clients/name_firstName_entitle/`, si il existe déjà, en réenvoie le certificat à l'utilisateur, sinon on le crée.
- on crée alors dans ce dossier différents fichiers utilisateur :
  - un fichier contenant l'adresse mail
  - un fichier personnal_data contenant les autres informations personnelles
  - un fichier query contenant les informations personnelles, un timestamp, le tout formater en requête

    - un fichier timestamp_sign correspondant à la requête signée et certifiée par FreeTSA
- une fois l'ensemble des informations formatées, l'image finale est assemblée :
    - le fond réalisé par spirographie est chargé en mémoire
    - on y ajoute le texte lisible
    - on cache des informations personnelles par stéganographie
    - on ajoute le QrCode (où les données ont été préalablement signées par la clé privée de *CertifPlus*)
- on envoie le mail à l'utilisateur correspondant :
    - on chiffre et on signe le mail, le tout au format S/MIME
    - on envoie le mail à l'utilisateur

3) Vérifier un certificat

Après avoir sélectionné l'option de vérification de certificat, *Certifplus* envoie au serveur le certificat à vérifier, encodé en base64. La fonction `ExtrairePreuve(client, certificateFileName)` est divisée en plusieurs vérifications :

- on contrôle la taille du certificat reçu (si celle-ci est différente de celle des certificats habituellement généré, on ne peut pas extraire le bloc de l'image).
- on récupère ensuite le bloc caché par stéganographie.
- avec le nom inscrit dans celui-ci, on vérifie si le répertoire associé existe, pour s'assurer qu'un certificat a bien été délivré à cette personne.
- on recrée le fichier personnal_data à partir des informations du bloc, puis on vérifie que ce fichier est bien similaire à celui aillant fait la requête d'horodatage auprès de FreeTSA.
- on vérifie la signature du QrCode et la compare à l'estampille d'horodatage du bloc.
- finalement, on recrée un certificat à partir des informations cachées dans l'image et du QrCode, et on compare les deux certificats pixel à pixel.

Si toutes ces étapes sont validées, nous considérons que le certificat est authentique. En cas d'erreur, un message expliquant la nature du problème est envoyé, et le programme s'arrête à cette vérification.

Exemple de fonctionnement

Les noms, secrets et informations envoyées au serveur sont contenues dans un programme client de test (client.py). Ce fichier se trouve à la racine du répertoire car nous estimons qu'il ne fait pas partie des sources. Le programme client est lancé par les lignes :

```
# user : pour générer un nouveau certificat
user = ["Hoffmann", "Clement", "clement.hoffmann@etu.unilim.fr", "Plongée"]
start_client("CertifPlus", "LameSecret", user)
```

Scénario 1)

Le client utilise un mauvais nom de client ou un mauvais secret partagé :

```
cirthy@cirthy-MS-7978:~/Documents/Unilim/S8/Projet_TIC$ python3 client.py
Name not in server database
cirthy@cirthy-MS-7978:~/Documents/Unilim/S8/Projet_TIC$ python3 client.py
Name found in database - Authentification...
wrong passphrase
```

Dans ces cas, il n'y a aucun affichage coté serveur.

Scénario 2)

Le client *CertifPlus* se connecte au serveur et demande la génération d'un certificat au nom de Clément Hoffmann pour un diplôme de Plongée.

```
cirthy@cirthy-MS-7978:~/Documents/Unilim/S8/Projet_TIC/sources$ python3 server.py

Server started, enter not empty line to exit

Connected client : CertifPlus
generating certificate ...
mkdir ../clients/CertifPlus/Hoffmann_Clement_Plongée
Using configuration from /usr/lib/ssl/openssl.cnf
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1325    0  1285  100    40   1032     32  0:00:01  0:00:01 --:--:--  1063
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 28886  100 28750  100   136   156k    759 --:--:-- --:--:-- --:--:--  157k
generated certificate
sending mail
sent mail
```

```
cirthy@cirthy-MS-7978:~/Documents/Unilim/S8/Projet_TIC$ python3 client.py
Name found in database - Authentification...
Authentification succeeded
[1] generate certificate
[2] verify certificate
Choice : 1
Certificate has been generated and sent to clement.hoffmann@etu.unilim.fr
```

Scénario 3)

Le client *CertifPlus* se connecte au serveur et demande la génération d'un certificat au nom de Clément Hoffmann pour un diplôme de Plongée. Cependant, ce certificat a déjà été généré, le serveur se contente de réenvoyer le mail à clement.hoffmann@etu.unilim.fr.

```
cirthy@cirthy-MS-7978:~/Documents/Unilim/S8/Projet_TIC/sources$ python3 server.py

Serveur started, enter not empty line to exit

Connected client : CertifPlus
certificate has been already generated
sending mail
sent mail
```

```
cirthy@cirthy-MS-7978:~/Documents/Unilim/S8/Projet_TIC$ python3 client.py
Name found in database - Authentification...
Authentification succeeded
[1] generate certificate
[2] verify certificate
Choice : 1
Certificate has been generated and sent to clement.hoffmann@etu.unilim.fr
```

Coté client, les informations visibles sont les mêmes que dans le scénario 2.

Scénario 4)

Le client *CertifPlus* se connecte au serveur et demande la vérification d'un certificat en renseignant le chemin d'accès à l'image PNG qu'il veut faire vérifier.

```
cirthy@cirthy-MS-7978:~/Documents/Unilim/S8/Projet_TIC/sources$ python3 server.py
Server started, enter not empty line to exit

Connected client : CertifPlus

Number of lines to receive : 68723
        --> Waiting base64 image certificate
        --> Received base64 image certificate
        --> Create base64 certificate file : OK
        --> Number of lines received : 68723
        --> Reception completed

Create PNG certificate file : OK

Check image size
        --> Right size : x=1753 and y=1240

Extract block data from steganography : OK

Check if repository exists
        --> Found repository

Extract timestamp data from block : OK

Block verification :
        --> Create personnal_data file
        --> Create query file
Using configuration from /usr/lib/ssl/openssl.cnf
        --> Verify TSA sing
Using configuration from /usr/lib/ssl/openssl.cnf
        --> Block verification completed

QrCode verification :
        --> QrCode verification completed

Recreate image certificate :
        --> Create QrCode sign
        --> Create assembled stego cetificate
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 28886  100 28750  100   136   157k    764 --:--:-- --:--:-- --:--:--  158k
        --> Recreate image certificate completed

Check differences between sent and recreated certificate :
        --> Check differences completed

Verification SUCCEDED : Certificate is valid
```

```
cirthy@cirthy-MS-7978:~/Documents/Unilim/S8/Projet_TIC$ python3 client.py
Name found in database - Authentification...
Authentification succeeded
[1] generate certificate
[2] verify certificate
Choice : 2
Path to certificate's filename : (without '.png') ./clients/CertifPlus/Hoffmann_Cl
ement_Plongée/certificate


Ready to send 68723 lines.
        --> Sending in progress : OK
        --> Reception completed
Verification SUCCEDED : Certificate is valid
```

Remarque)

Pour plus de tests, un script bash `restart.sh` permet de supprimer le dossier `./clients/CertifPlus` et de nettoyer la base de données client. Un nouveau client (*CertifPlus* par exemple) peut ensuite être enregistré via le programmes `./sources/add_client.py`. Enfin, le programme `./client.py` permettant la connexion au serveur contient les informations liées au client ainsi que les données nécessaires à la génération d'un nouveau certificat.

Analyse de risques

Notre analyse de risque est détaillée en plusieurs scénarios possibles touchant les actifs primaires essentiels que nous avons identifiés durant notre travail. Nous les mettrons également en relation avec les actifs secondaires correspondant.

Tout d'abord, nous nous intéressons aux données clients (actif primaire). Les actifs secondaires associés sont les ordinateurs, le réseau de mise en commun des informations au sein de l'entreprise et les moyens d'accès aux données. Nous pouvons penser que ces informations sont vulnérables, car accessibles depuis un ordinateur de l'entreprise par n'importe qui. Un individu ne faisant pas partie de l'entreprise pourrait mettre la main sur le dossier de nos clients, contenant pour chacun d'entre eux le secret que nous partageons avec celui-ci, ainsi que leurs clés de signature. Cet individu serait alors en capacité de délivrer des certificats comme bon lui semble, certificats qui seraient reconnus comme authentique par notre logiciel. Il pourrait également les diffuser ou les vendre :-o ! Les préconisations contre cette menace sont le chiffrement du fichier client et la protection des équipements à partir desquels ce fichier est accessible.

Nous nous intéressons maintenant aux données délivrées par les personnes souhaitant un certificat. Ces données (nom, prénom, adresse email et intitulé du certificat) nous sont transmises par le réseau (actif secondaire associé), et stockées dans un fichier pour créer l'attestation. Ces données sont donc menacées, comme les précédentes, par le vol sur le système de stockage de l'entreprise, mais également par l'écoute du réseau. Ces menaces proviennent des vulnérabilités liées au stockage ainsi qu'au transfert des informations. La récupération de ces données personnelles peut être évitée grâce aux mêmes préconisations que précédemment, mais également grâce à la sécurisation du réseau local de l'entreprise.

Conclusion

Notre service clientèle reste à votre disposition pour toute question … et demande de devis.

CDLT,
néanmoins bien à vous.