

UNIVERSITÉ DE LIMOGES

FACULTÉ DES SCIENCES ET TECHNIQUES

Rapport du projet "Sécurité des usages TIC et cryptographie" : Authentification, Web sécurisé et Stéganographie

Auteurs :

BRIKA LYDIA
DIB KAMAL
OUAFI SOUFIANE

Superviseurs :

M. PIERRE-FRANÇOIS BONNEFOI
M. EMMANUEL CONCHON

2020/2021



Université
de Limoges

Table des matières

| | | |
|----------|--|----------|
| 1 | Introduction : | 2 |
| 2 | Conception du projet : | 2 |
| 2.1 | Création de notre autorité de certification (CA) : | 2 |
| 2.2 | Fonction de création d'une attestation : | 3 |
| 2.3 | Fonction de vérification d'une attestation : | 4 |
| 2.4 | Création du serveur web : | 4 |
| 2.5 | Communication entre serveurs : | 4 |
| 3 | Exemple d'exécution : | 5 |
| 3.1 | Création d'une attestation : | 5 |
| 3.2 | Vérification d'une attestation : | 6 |
| 3.3 | Récupération d'une attestation : | 6 |
| 4 | Analyse de risques : | 7 |
| 5 | Conclusion : | 7 |

1 Introduction :

Dans le cadre du projet de l'UE Sécurité des usages TIC et cryptographie, il nous a été demandé de mettre en place un procédé permettant la création et la vérification d'attestation, en utilisant les notions abordés en cours et TP : autorité de certification, sténographie, outils cryptographiques de la bibliothèque Openssl, etc. Pour la réalisation de ce projet, nous avons formé un groupe de travail de trois personnes en répartissant d'abord les tâches entre nous afin que chacun de nous puisse réaliser une mission individuelle.

Cependant, vu le contexte actuel et le fait que l'on soit tous débutants en programmation en Python, cela s'est avéré difficile. Nous avons ainsi changé de méthode en nous retrouvant régulièrement sur Discord ou chez l'un d'entre nous et en travaillant ensemble sur les mêmes parties jusqu'à les faire fonctionner.

2 Conception du projet :

La construction projet a été faite en plusieurs parties qu'on a développées au fur et à mesure qu'on avançait. Voici les principales étapes par lesquelles nous sommes passés pour arriver au résultat final :

2.1 Création de notre autorité de certification (CA) :

Pour la création de notre autorité de certification, qu'on appellera CA, nous avons utilisé les commandes et informations du TP5, ce qui nous a permis de créer une CA en utilisant les courbes elliptiques.

1. **Création de la CA** : La première étape a été de générer une clé privée (ecc.ca.key.pem), à partir de laquelle on créera la CA (ecc.ca.cert.pem).
2. **Création des certificats** : Ensuite, on a généré les certificats pour nos deux serveurs : le serveur frontal (ecc.serveur-frontal.pem) et le serveur applicatif (ecc.serveur-applicatif.pem).
3. **Création de la clé publique** : Enfin, nous avons généré une clé publique pour le serveur applicatif (qu'on utilisera dans la fonction de vérification de signature plus bas), et nous avons réunis la clé privée et le certificat du serveur applicatif dans un fichier appelé bundle-serveur-frontal.pem (comme il nous a été conseillé de faire dans le sujet du projet).

Voici les commandes utilisées pour la création de notre CA, on précise que le mot de passe choisi est "TIC2021".

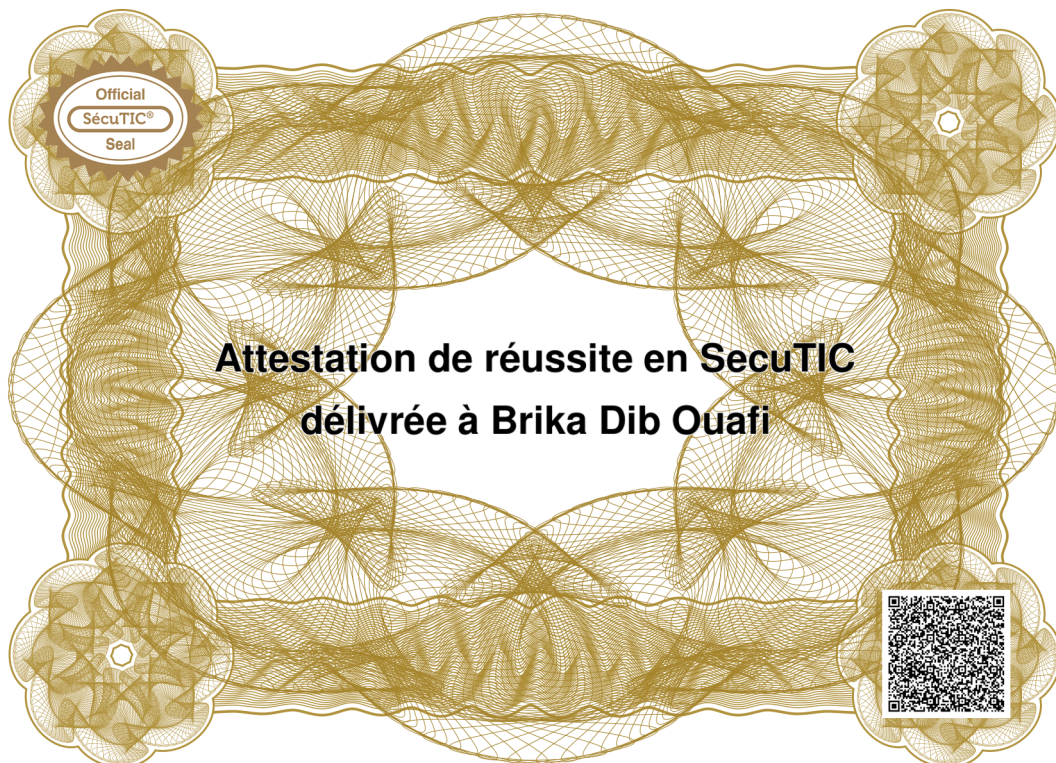
```
brika1@fst-o-i-211-02:~/Bureau/Projet SecuTIC/CAS$ openssl ecparam -out ecc.ca.key.pem -name prime256v1 -genkey
brika1@fst-o-i-211-02:~/Bureau/Projet SecuTIC/CAS$ openssl req -config <(printf "[req]\ndistinguished_name=dn\n[dn]\n[ext]\nbasicConstraints=CA:TRUE")
-new -nodes -subj "/C=FR/L=Limoges/O=CRYPTIS/OU=SecuTIC/CN=localhost" -x509 -extensions ext -sha256 -key ecc.ca.key.pem -text -out ecc.ca.cert.pem -
passin pass:TIC2021
brika1@fst-o-i-211-02:~/Bureau/Projet SecuTIC/CAS$ openssl req -config <(printf "[req]\ndistinguished_name=dn\n[dn]\n[ext]\nbasicConstraints=CA:FALSE"
) -new -subj "/C=FR/L=Limoges/O=CRYPTIS/OU=SecuTIC/CN=localhost" -reqexts ext -sha256 -key ecc.ca.key.pem -text -out ecc.csr.pem -passin pass:TIC2021
brika1@fst-o-i-211-02:~/Bureau/Projet SecuTIC/CAS$ openssl x509 -req -days 3650 -CA ecc.ca.cert.pem -CAkey ecc.ca.key.pem -CAcreateserial -extfile <(p
rintf "basicConstraints=critical,CA:FALSE\nkeyUsage=critical,digitalSignature") -in ecc.csr.pem -text -out ecc.serveur_applicatif.pem -passin pass:TI
C2021
Signature ok
subject=C = FR, L = Limoges, O = CRYPTIS, OU = SecuTIC, CN = localhost
Getting CA Private Key
brika1@fst-o-i-211-02:~/Bureau/Projet SecuTIC/CAS$ openssl x509 -req -days 3650 -CA ecc.ca.cert.pem -CAkey ecc.ca.key.pem -CAcreateserial -extfile <(p
rintf "basicConstraints=critical,CA:FALSE") -in ecc.csr.pem -text -out ecc.serveur_frontal.pem -passin pass:TIC2021
Signature ok
subject=C = FR, L = Limoges, O = CRYPTIS, OU = SecuTIC, CN = localhost
Getting CA Private Key
brika1@fst-o-i-211-02:~/Bureau/Projet SecuTIC/CAS$ openssl x509 -pubkey -noout -in ecc.serveur_applicatif.pem -out ecc.pubkey.serveur_applicatif.pem
brika1@fst-o-i-211-02:~/Bureau/Projet SecuTIC/CAS$ cat ecc.ca.key.pem ecc.serveur_frontal.pem > bundle_serveur_frontal.pem
```

2.2 Fonction de création d'une attestation :

La création d'une attestation a été assez long à cause du travail à faire pour générer l'attestation finale. Ainsi notre fonction de création d'une attestation est composée de plusieurs parties :

1. **Enregistrer les informations** : nous avons récupéré l'identité et l'intitulé du certificat, que nous avons mis dans un fichier texte (CarteCrypto.txt).
2. **Signature des informations** : nous avons signé le fichier (CarteCrypto.txt) en utilisant la clé privée de notre CA avec le mot de passe qu'on a saisi au lancement du serveur. Une fois la signature effectuée, on l'enregistre dans un fichier sig.txt, qu'on convertit ensuite en base64 dans un fichier signature.txt (cette conversion a été demandée dans le sujet du projet).
3. **Création du code QR** : nous avons récupéré le fichier signature.txt, et créé un fichier qrcode.png (en rb), dans lequel on va stocker cette signature en tant que code QR grâce à la bibliothèque qrcode.
4. **Ajout des informations au fond d'attestation** : pour ajouter les informations et le code QR au fond d'attestation que nous a fourni monsieur Bonnefoi, nous avons utilisé l'outil Google Chart, qui nous retourne une image texte.png avec le texte "Attestation en (intitulé de la formation) délivrée à (nom et prénom)". Une fois qu'on a texte.png et qrcode.png, on a pu les superposer à notre fond d'attestation après avoir modifié leurs dimensions, grâce aux outils graphiques du sujet du projet.
5. **Demande de timestamp** : La prochaine étape a été l'envoi d'une demande de timestamp au site web "freetsa.org", qui rajoute à notre attestation un horodatage de confiance, qu'on récupère par l'outil Curl.
6. **Stéganographie** : La dernière étape a été de cacher les informations : (nom prénom || intitulé de la certification || timestamp) dans notre attestation finale par stéganographie, où "||" signifie concaténation, et pour cela on a utilisé les fonctions mises à notre disposition dans le sujet du projet.

Toutes ces étapes ont été réunies dans une même fonction "créattestation(identité,intitulécertif)" dans le fichier "creationattestation.py", ainsi faire appel à cette fonction donnera en sortie une image "steganoattestation", qui est une attestation qui comporte les informations, le code QR et les informations dissimulées par stéganographie. Voici un exemple d'une attestation généré par notre programme :



2.3 Fonction de vérification d'une attestation :

Une fois qu'on a réussi à créer des attestations, nous nous sommes lancés dans la vérification de ces attestations, et cela s'est fait en plusieurs étapes :

1. **Récupération des informations** : La première étape a été la récupération des informations dissimulées par téganographie, qu'on a ensuite divisées en deux parties, les informations et le timestamp.
2. **Vérification du timestamp** : Après récupération du timestamp, nous l'avons mis dans un fichier qu'on vérifiera en utilisant l'option `verify` de `openssl`, en l'envoyant à au site "freetsa.org" et en utilisant sa CA.
3. **Vérification de la signature et du code QR** : Une fois nos informations récupérés, nous les avons mises dans un fichier, ensuite on récupère les informations cachés dans le code QR, c'est à dire la signature. A partir de ces deux fichier, on a pu utiliser l'option `verify` de `openssl` et en utilisant la clé publique de notre CA afin de vérifier que les informations correspondent bien à la signature du document.

Toutes ces étapes ont été réunies dans une même fonction `"verifattes(fichier)"` dans le fichier `"verifattestation.py"`, qui prendra en entrée une image (une attestation) et en sortira les informations à vérifier et si les deux étapes de vérifications sont remplies, retournera un message confirmant que l'attestation donnée en entrée est vraie.

2.4 Création du serveur web :

Une fois nos fonctions de création et de vérification finalisées, nous nous sommes tournés vers le serveur web, et bien que le code nous ait été donné en majeure partie, faire fonctionner le serveur applicatif en passant par un serveur frontal a été un peu plus difficile, mais nous avons quand même réussi. Ainsi, nous avons importé nos fonctions de création et de vérification en tant que bibliothèques pour nous faciliter le travail.

2.5 Communication entre serveurs :

Afin de faciliter la communication entre client-serveur, nous avons décidé de créer un programme `client.py`, qui demande à l'utilisateur ce qu'il souhaite faire et envoie à sa place les requêtes 'Curl' correspondantes au serveur frontal en TCP, sur port 9000, qui relaie ensuite les informations au serveur applicatif. Voici ce que reçoit le client lorsqu'il lance ce programme :

```
brika1@fst-o-i-211-02:~/Bureau/Projet Securite TIC$ python3 client.py
Bienvenue au service CertiPlus

Si vous souhaitez générer une attestation tapez '1'

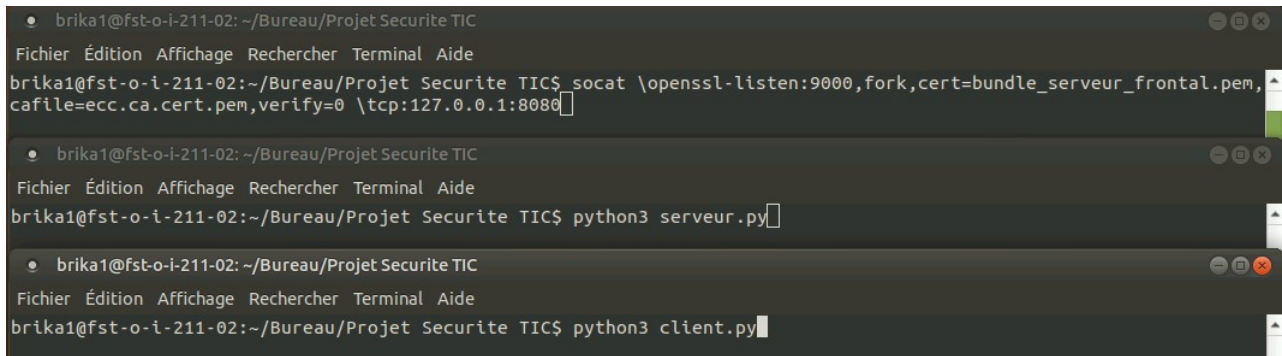
si vous souhaitez vérifier une attestation tapez '2'

si vous souhaitez recuperer une attestation tapez '3'

vous pouvez quitter a tout moment en tapant ctrl+c
```


3 Exemple d'exécution :

Pour lancer notre programme, il faut ouvrir trois terminaux : un pour le serveur applicatif (serveur.py), un pour le serveur frontal et un pour le client (client.py)



```
brika1@fst-o-i-211-02: ~/Bureau/Projet Securite TIC
Fichier Édition Affichage Rechercher Terminal Aide
brika1@fst-o-i-211-02:~/Bureau/Projet Securite TIC$ socat \openssl-listen:9000,fork,cert=bundle_serveur_frontal.pem,cafile=ecc.ca.cert.pem,verify=0 \tcp:127.0.0.1:8080

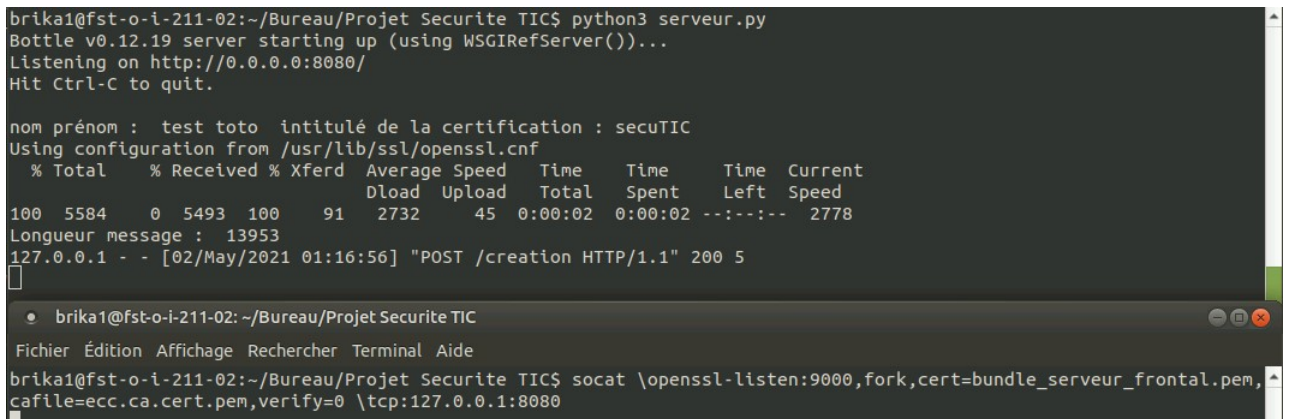
brika1@fst-o-i-211-02: ~/Bureau/Projet Securite TIC
Fichier Édition Affichage Rechercher Terminal Aide
brika1@fst-o-i-211-02:~/Bureau/Projet Securite TIC$ python3 serveur.py

brika1@fst-o-i-211-02: ~/Bureau/Projet Securite TIC
Fichier Édition Affichage Rechercher Terminal Aide
brika1@fst-o-i-211-02:~/Bureau/Projet Securite TIC$ python3 client.py
```

3.1 Création d'une attestation :

Voici ce qui se passe aux niveau des serveurs et du client lorsque ce dernier souhaite créer une attestation :

— Serveurs :

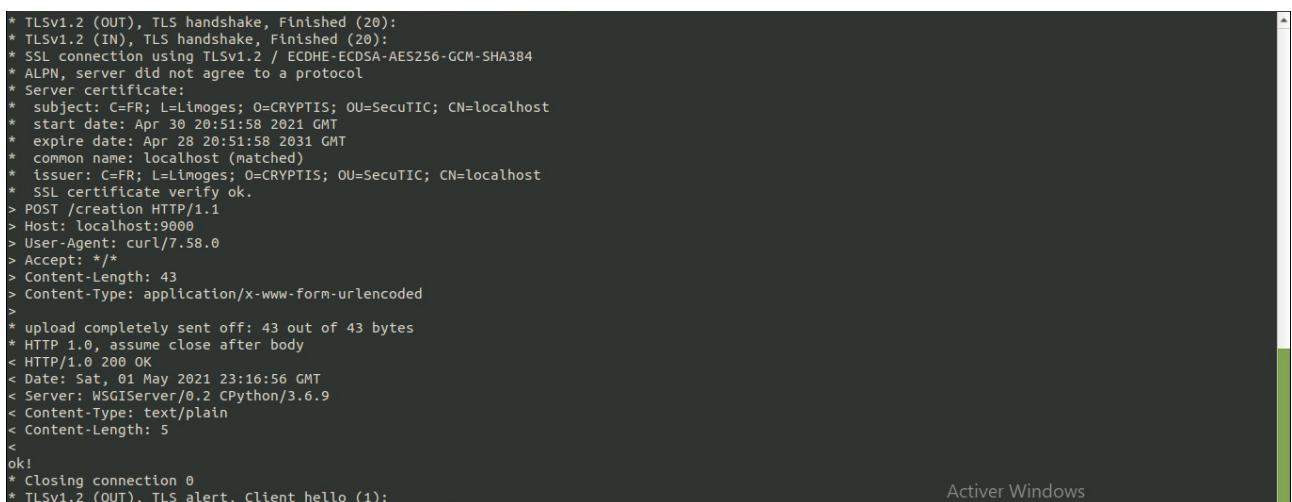


```
brika1@fst-o-i-211-02:~/Bureau/Projet Securite TIC$ python3 serveur.py
Bottle v0.12.19 server starting up (using WSGIRefServer())...
Listening on http://0.0.0.0:8080/
Hit Ctrl-C to quit.

nom prénom : test toto intitulé de la certification : secuTIC
Using configuration from /usr/lib/ssl/openssl.cnf
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 5584 0 5493 100 91 2732 45 0:00:02 0:00:02 --:--:-- 2778
Longueur message : 13953
127.0.0.1 - - [02/May/2021 01:16:56] "POST /creation HTTP/1.1" 200 5

brika1@fst-o-i-211-02: ~/Bureau/Projet Securite TIC
Fichier Édition Affichage Rechercher Terminal Aide
brika1@fst-o-i-211-02:~/Bureau/Projet Securite TIC$ socat \openssl-listen:9000,fork,cert=bundle_serveur_frontal.pem,cafile=ecc.ca.cert.pem,verify=0 \tcp:127.0.0.1:8080
```

— Client :



```
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-ECDSA-AES256-GCM-SHA384
* ALPN, server did not agree to a protocol
* Server certificate:
* subject: C=FR; L=Limoges; O=CRYPTIS; OU=SecuTIC; CN=localhost
* start date: Apr 30 20:51:58 2021 GMT
* expire date: Apr 28 20:51:58 2031 GMT
* common name: localhost (matched)
* issuer: C=FR; L=Limoges; O=CRYPTIS; OU=SecuTIC; CN=localhost
* SSL certificate verify ok.
> POST /creation HTTP/1.1
> Host: localhost:9000
> User-Agent: curl/7.58.0
> Accept: */*
> Content-Length: 43
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 43 out of 43 bytes
* HTTP 1.0, assume close after body
< HTTP/1.0 200 OK
< Date: Sat, 01 May 2021 23:16:56 GMT
< Server: WSGIServer/0.2 CPython/3.6.9
< Content-Type: text/plain
< Content-Length: 5
<
ok!
* Closing connection 0
* TLSv1.2 (OUT), TLS alert, Client hello (1):
```

3.2 Vérification d'une attestation :

Voici ce qui se passe au niveau des serveurs et du client lorsque ce dernier souhaite vérifier une attestation :

— Serveurs :

```
brika1@fst-o-i-211-02:~/Bureau/Projet Securite TIC$ python3 serveur.py
Bottle v0.12.19 server starting up (using WSGIRefServer())...
Listening on http://0.0.0.0:8080/
Hit Ctrl-C to quit.

Using configuration from /usr/lib/ssl/openssl.cnf
Verification: OK
timestamp ok
Verified OK
sig ok
127.0.0.1 - - [02/May/2021 01:20:02] "POST /verification HTTP/1.1" 200 23

brika1@fst-o-i-211-02:~/Bureau/Projet Securite TIC
Fichier Édition Affichage Rechercher Terminal Aide
brika1@fst-o-i-211-02:~/Bureau/Projet Securite TIC$ socat \openssl-listen:9000,fork,cert=bundle_serveur_frontal.pem,
cafile=ecc.ca.cert.pem,verify=0 \tcp:127.0.0.1:8080
```

— Client :

```
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-ECDSA-AES256-GCM-SHA384
* ALPN, server did not agree to a protocol
* Server certificate:
* subject: C=FR; L=Limoges; O=CRYPTIS; OU=SecuTIC; CN=localhost
* start date: Apr 30 20:51:58 2021 GMT
* expire date: Apr 28 20:51:58 2031 GMT
* common name: localhost (matched)
* issuer: C=FR; L=Limoges; O=CRYPTIS; OU=SecuTIC; CN=localhost
* SSL certificate verify ok.
> POST /verification HTTP/1.1
> Host: localhost:9000
> User-Agent: curl/7.58.0
> Accept: */*
> Content-Length: 3331266
> Content-Type: multipart/form-data; boundary=-----21c7b78fc0d7b0ea
> Expect: 100-continue
>
* Done waiting for 100-continue
* HTTP 1.0, assume close after body
< HTTP/1.0 200 OK
< Date: Sat, 01 May 2021 23:20:02 GMT
< Server: WSGIServer/0.2 CPython/3.6.9
< Content-Type: text/plain
< Content-Length: 23
<
Attestation certifiée
* Closing connection 0
* TLSv1.2 (OUT), TLS alert, Client hello (1):
```

3.3 Récupération d'une attestation :

Voici ce qui se passe au niveau des serveurs et du client lorsque ce dernier souhaite récupérer l'attestation créée :

— Serveurs :

```
brika1@fst-o-i-211-02:~/Bureau/Projet Securite TIC$ python3 serveur.py
Bottle v0.12.19 server starting up (using WSGIRefServer())...
Listening on http://0.0.0.0:8080/
Hit Ctrl-C to quit.

127.0.0.1 - - [02/May/2021 01:22:10] "GET /fond HTTP/1.1" 200 3331065

brika1@fst-o-i-211-02:~/Bureau/Projet Securite TIC
Fichier Édition Affichage Rechercher Terminal Aide
brika1@fst-o-i-211-02:~/Bureau/Projet Securite TIC$ socat \openssl-listen:9000,fork,cert=bundle_serveur_frontal.pem,
cafile=ecc.ca.cert.pem,verify=0 \tcp:127.0.0.1:8080
```

— Client :

```
* Server certificate:
* subject: C=FR; L=Limoges; O=CRYPTIS; OU=SecuTIC; CN=localhost
* start date: Apr 30 20:51:58 2021 GMT
* expire date: Apr 28 20:51:58 2031 GMT
* common name: localhost (matched)
* issuer: C=FR; L=Limoges; O=CRYPTIS; OU=SecuTIC; CN=localhost
* SSL certificate verify ok.
} [5 bytes data]
> GET /fond HTTP/1.1
> Host: localhost:9000
> User-Agent: curl/7.58.0
> Accept: */*
>
[5 bytes data]
* HTTP 1.0, assume close after body
< HTTP/1.0 200 OK
[5 bytes data]
< Date: Sat, 01 May 2021 23:22:10 GMT
< Server: WSGIServer/0.2 CPython/3.6.9
[5 bytes data]
< Content-Type: image/png
< Content-Length: 3331065
<
[8140 bytes data]
100 3252k 100 3252k 0 0 158M 0 --:--:-- --:--:-- --:--:-- 167M
* Closing connection 0
} [5 bytes data]
* TLSv1.2 (OUT), TLS alert, Client hello (1):
[2 bytes data]
```

Activer Windows

4 Analyse de risques :

Il nous a été demandé dédier une analyse de risques, ainsi au cours de notre travail nous avons identifié les biens primaires et secondaires à protéger, les menaces sur ces biens, et les relations entre les deux.

Le bien principal à protéger est bien sûr les données qui nous sont communiqués pour la création d'une attestation, c'est à dire nom, prénom et l'intitulé de la certification, ces informations sont un secret partagé entre le client et nous en tant qu'organisme délivrant une attestation. Ainsi, ces informations qui nous sont transmises par réseau et qui sont stockés dans un fichier encourent plusieurs risques, comme l'écoute sur le réseau, ou par le vol des informations de notre entreprise qui stocke ces informations. En réalisant les dangers qu'encourt l'entreprise en elle-même, nous l'avons identifié en tant que bien secondaire à protéger, car elle encourt elle-même plusieurs risque en particulier un risque sur le dossier où sont enregistrés les informations du client ou bien sur le programme délivrant les attestations à cause des vulnérabilités de stockage des informations, ainsi que de leurs transport lors de l'échange entre client et entreprise.

Pour terminer cette analyse, nos recommandations pour régler ces problèmes de sécurité sont le cryptage des données du client lors de l'enregistrement afin qu'elle soient indéchiffrable à toute personne en dehors du secret, ensuite il est important de sécuriser les moyens d'échange entre client et entreprise afin d'éviter tout espionnage sur le réseau et possible vol d'informations de valeur, enfin l'entreprise a pour devoir de sécuriser les appareils de stockage des données et des programmes de création d'attestation, dans le cas d'un vol de ses biens matériels.

5 Conclusion :

En conclusion, on aimerait exprimer notre fierté du travail que nous avons accompli, car bien qu'on réalise qu'il soit imparfait, il est le résultat de dur travail, en particulier car aucun de nous trois n'avait utilisé Python ou Linux avant cette année.

Nous sommes néanmoins déçus de ne pas avoir eu plus de temps pour ajouter des améliorations comme la création d'une base de données avec les attestations créées et la suppression des fichiers intermédiaires (tel que combinaison.png et texte.png) à la fin du programme de création.

Pour finir nous souhaitons vous remercier de cette opportunité d'apprentissage qui nous a appris travail de groupe, coordination et acharnement pour atteindre un but commun.