Une image contenant texte, clipart

Description générée automatiquement

Interception TLS



Projet de cryptographie

2021-2022

Florian PEDEPRAT, Océane BRAS, Laetitia COLLODET

Sommaire

1 – L’interception TLS expliquée

2 – Implémentation

3 – Autorité de certification

4 – Autres choix techniques

5 – Difficultés rencontrées

6 – Améliorations envisageables

7 – Ressources utilisées

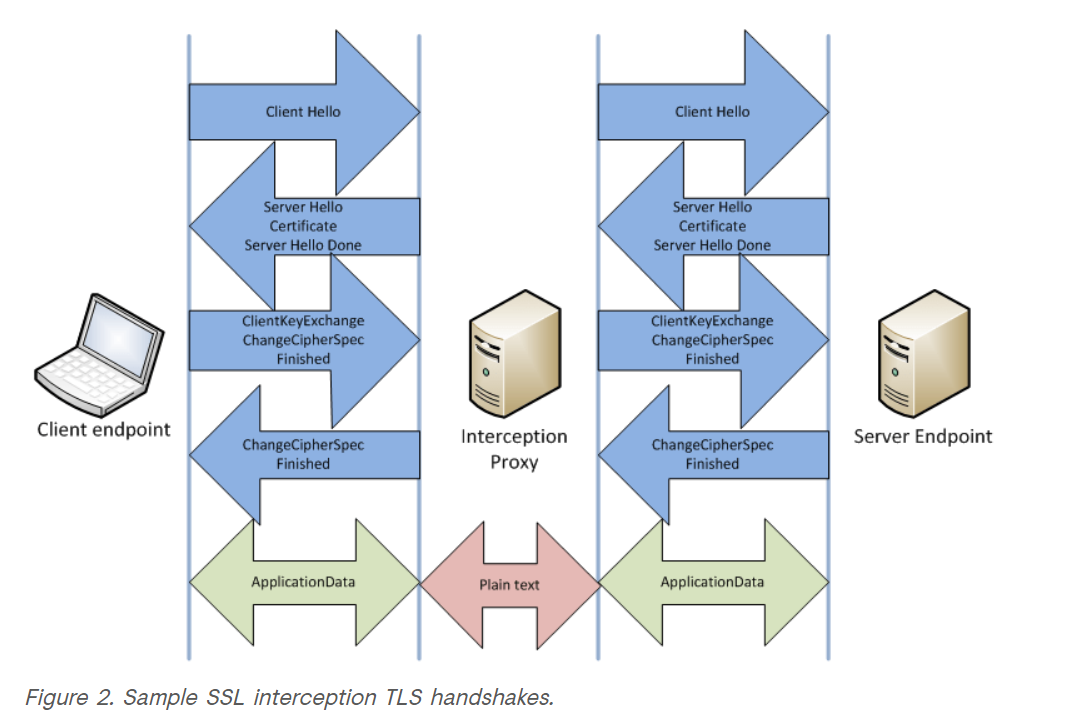
8 – Conclusion

1 – Interception TLS expliquées

1. Principes

Sur Internet, la majorité des communications client-serveur utilisent le protocole HTTP (Hypertext Transfer Protocol), depuis plusieurs années maintenant, ce protocole a été enrichi avec un protocole de chiffrement, le TLS (Transport Layer Security). Le TLS a apporté de la confidentialité, de l’authenticité et de l’intégrité aux échanges entre clients et serveur. En effet, le HTTP classique permettait à n’importe qui de pouvoir lire les communications sur un réseau, puisque le contenu des requêtes apparaissait en clair.

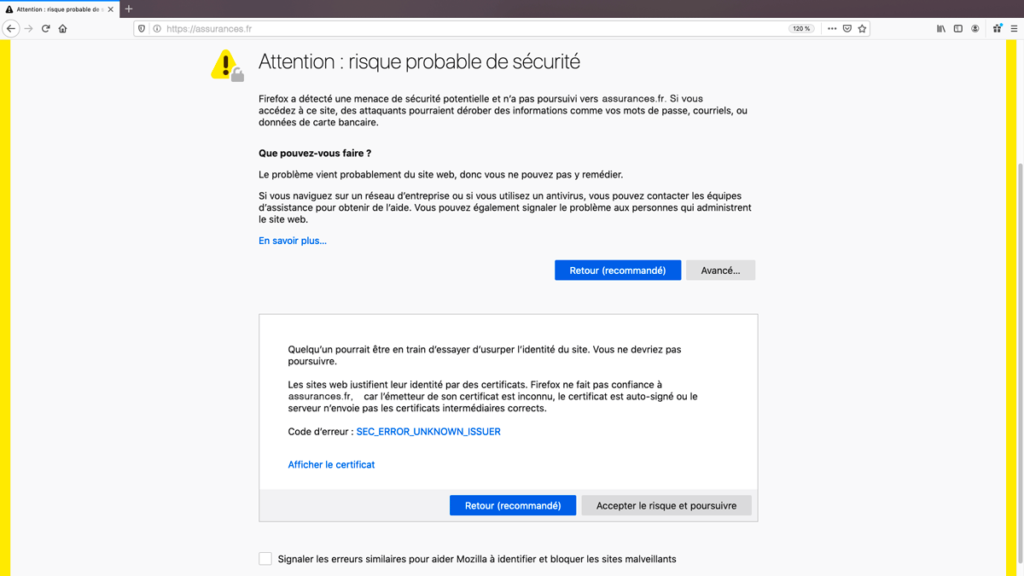
L’élément clef du protocole HTTPS est le certificat. Ce certificat a pour but de prouver l’authenticité et l’intégrité de la clef public du serveur, pour que le client soit certain de s’adresser à la bonne personne. Cette clef sera notamment utilisée lors du handshake pour élaborer une clef symétrique secrète commune, qui permettra de chiffrer les échanges.

Le principe de l’interception TLS repose sur le fait de pouvoir se positionner entre un client et un serveur, d’intercepter les requetes qu’ils pourraient s’échange et de les déchiffrer. C’est ce qu’on appelle un proxy. Pour se faire le proxy doit se faire passer pour le serveur aux yeux du client. Ainsi la demande de connexion d’un client, un handshake va être réalisé entre le client et le proxy, puis entre le proxy et le serveur que le client souhaite joindre.

Pour se faire passer pour le serveur, le proxy doit avoir accès à la clef privée du serveur, pour pouvoir déchiffrer les échanges lors du handshake, or ceci, n’est pas possible. De plus, lorsque le client effectue une requête vers un nom de domaine, il s’attend à recevoir un certificat présentant le même nom de domaine. Autrement, la communication est bloquée. Le proxy doit donc lors du handshake utiliser un certificat ayant le même nom de domaine. Pour ce faire le proxy doit embarquer avec lui une autorité de certification racine qui va générer des certificats à la volée selon le nom de domaine demandé par le client.

Le certificat de l’autorité racine devra être présente dans le magasin de certificat du navigateur si on souhaite que le client valide les certificats générés à la volée.

En ce qui concerne la version du protocole TLS, il est vivement recommandé d’utiliser au minimum la version 1.2, car les versions précédentes sont vulnérables à diverses attaques.



Exemple d’un site ayant fourni un certificat auto signé

Si l’intercepteur TLS utilise un certificat auto signé, le navigateur estime que ce certificat n’est potentiellement pas légitime

L’interception TLS est souvent utilisé en entreprise pour protéger les systèmes d’informations qui sont vulnérables a de divers attaques (injection de malware,..)

1. Intérêts

L’arrivée du protocole TLS a été une révolution en termes de sécurité, permettant à tous de communiquer sur internet de façon sécurisée. Malheureusement, cette révolution a permis aux menaces (tel que les botnets, malware, attaques web) de passer inaperçu.

Les principaux intérêts sont :

* Filtrer les requetes pour interdire la visiter de certains sites web
* Détecter l’injection de malware, en permettant au proxy de voir les binaires ou les documents téléchargés. Ces documents vont être analysés. Par exemple en comparant avec la signature de malware connus ou en ouvrant ces fichiers dans des sandbox.
* Détecter si des fichiers propriétés de l’entreprise sont exfiltrés

1. Inconvénients

Imaginons qu’une personne malveillante ait pu installer et configurer un proxy intercepteur TLS sur votre pc, si on ne vérifie pas la configuration de notre navigateur, il est compliqué de se rende compte que le trafic est intercepté.

Les données sensibles peuvent etre récupérées (voir section Test et Résultats), ce qui viole la vie privée de utilisateurs.

L’intercepteur TLS est un outil à manipuler avec précaution, en effet, un proxy mal implémenté, ne fera pas forcément la différence entre un bon site internet ou non. Il pourra alors laisser l’utilisateur se connecter sur un serveur mal intentionné ouvrant la porte à toutes sortes d’attaques

Cet outil pourrait être un inconvénient pour les employés qui n’utiliseront plus leur pc d’entreprise pour visiter leur site bancaire, réseau sociaux etc pendant leur pause par mesures de sécurités, car ils préfèrent être en sécurité sur leur réseau sans proxy.

1. Enjeux juridiques

Finalement, le principe de l’interception TLS n’est autre qu’une *man in the middle* *attack* s’il est appliqué sans le consentement des utilisateurs, les personnes ou organisations ayant recours à ce type d’installation sur leur réseau se doivent donc de justifier d’un accord signé entre les différentes parties

* La loi en France
  + CNIL

L’organisme recommande l’utilisation du protocole https pour protéger les flux de données entre un client et le serveur demandée afin d’éviter l’interception de donnée dans le but d’écouter ou modifier les requêtes.

Cependant, cette sécurité peut aussi constituer une vulnérabilité pour la sécurité des systèmes d’informations d’une entreprise. En effet, les données étant chiffrés, il devient impossible pour un administrateur de surveiller les flux circulant sur le réseau ou voire meme de détecter de potentiel malware.

Ces deux besoins de sécurité sont contradictoires mais nécessaire pour les systèmes d’informations. Il en résulte donc que l’utilisation de l’interception TLS est considérée comme légitime si elle intervient dans un intérêt de sécurité pour l’organisme tout en étant transparent avec ses salariés.

* + ANSSI

Extrait d’un rapport « Recommandations de sécurité concernant l’analyse des flux https » :

***« Pour pouvoir mettre en œuvre le déchiffrement de flux TLS de façon maîtrisée, il est nécessaire de disposer, entre autres, d’un niveau de connaissance suffisant dans les deux domaines spécifiques et évolutifs que sont les IGC*([Infrastructure de Gestion de Clés)*] et la cryptographie. Quel que soit le contexte, la mise en place de mécanismes de déchiffrement HTTPS présente des risques dans la mesure où cette opération entraîne la rupture d’un canal sécurisé et expose des données en clair au niveau de l’équipement en charge de l’opération. Lorsqu’un tel déchiffrement est nécessaire, sa mise en œuvre doit s’accompagner de beaucoup de précautions et se faire uniquement après validation de la direction des systèmes d’information voire d’une autorité de niveau supérieur. »***

Cet extrait évoque la fragilité que peut entrainer le déchiffrement de flux https, une bonne maitrise est requise. En effet, l’infrastructure qui effectue une telle opération doit être protégé. Personne ne doit être capable d’accéder à ces flux en clair. Ainsi, les paires de clefs utilisées pour signer les certificats de la chaine de certification doivent être stockées de façons sécurisé, dans un HSM par exemple. Les algorithmes utilisés doivent présenter un niveau de sécurité acceptable.

2 – Implémentation

1. Langage

Pour ce projet, nous avons hésité entre 3 langages de programmation : Python, Java, Rust. Nous avons choisi le Rust.

En effet, nos compétences sont plus développées dans ce langage. De plus, le Rust est un langage de programmation offrant de très bonnes performances tout en garantissant un très bon niveau de sécurité, notamment grâce à sa façon de gérer les erreurs, la mémoire et les types.

Concernant les types, le rust présente un typage fort, c’est-à-dire que les types des variables sont définis à la compilation du programme et ne pourront pas être changés lors de l’exécution. De même la vérification des types est effectuée à la compilation, c’est ce qu’on appelle le typage statique.

C’est aussi un langage multiplateforme, qui n’aura donc pas de mal à être exécuté sur d’autre plateforme que ce soit windows, Linux, MacOS. Il sera donc possible d’exécuter le proxy sur différent système d’exploitation.

1. Librairies utilisées

Voici la liste des librairies que nous avons utilisées et pourquoi :

* Openssl : pour la génération de certificat, de paires de clef privé/publique
* Hudsocker : ouvrir une socket entre client et proxy et proxy et serveur
* Tracing : pour la visualisation de log
* Tokio pour l’asynchronisme des processus
* Hyper pour la gestion de requetes http, body et header

Bien évidemment, nous utilisons la dernière version de chaque librairie.

1. Etapes implémentées

Nous avons généré la paire de clef et le certificat de l’autorité racine de certification dans un terminal à l’aide de commande OpenSSL. (*Voir section 3- Autorité de certification).*

Une fois que ceci est fait, nous les avons récupérés dans le code source en binaire.

Une image contenant texte

Description générée automatiquement

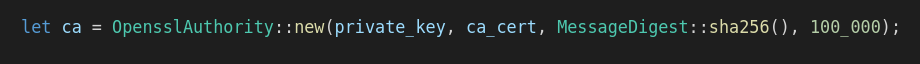
Par la suite, nous allons devoir manipuler ces deux fichiers avec la librairie **OpensSSL**, cette librairie nécessite de convertir ces deux fichiers en 2 types bien particulier : **PKey et X509**.

Une image contenant texte

Description générée automatiquement

La prochaine étape est la construction du proxy, qui ouvre une socket sur notre localhost au port 2000. On lui injecte une autorité de certification dans with\_ca afin qu’il puisse gérer la génération de certificat à la volée. Une image contenant texte

Description générée automatiquement

Voici l’objet ca utilisé dans with\_ca.

Chaque fois qu’une requête arrive sur le proxy, il fait appel à une structure *LogHandler* qui est la suivante.

Une image contenant texte

Description générée automatiquement

Cette structure comporte 2 fonctions asynchrones :

* *Handle\_request* qui dès que le proxy va recevoir une requête va en décortiquer chaque partie (contexte, body, header) et va transmettre la requête au bon destinataire.
* *Handle\_Response* qui dès qu’elle va recevoir une réponse, va la transmettre au client.

Une fois que notre proxy est terminé, il faut le configurer dans notre navigateur en insérant le certificat de la root CA dans le magasin de certificat

Une image contenant texte

Description générée automatiquement

Configuration du proxy dans le navigateur.

1. Tests et résultats

Nous avons tout d’abord mis en place un proxy en http. De la meme façon, nous avons ouvert une socket sur le port 2000 qui récupérer les requêtes faite par le client sur le navigateur et les transmet au serveur que le client veut joindre. Lorsque le serveur lui répond il fait suivre la réponse. Nous avons testé cette configuration sur des site en http tel que <http://www.ecoledulouvre.fr/> et nous avons pu afficher la page complète.

Puis nous sommes passé sur une implémentation des requêtes en HTTPS.

Une image contenant texte

Description générée automatiquementOn se rend sur google.com avec notre proxy en place. On peut voir que la connexion est sécurisée. Le certificat est signé par notre root CA ISEN.

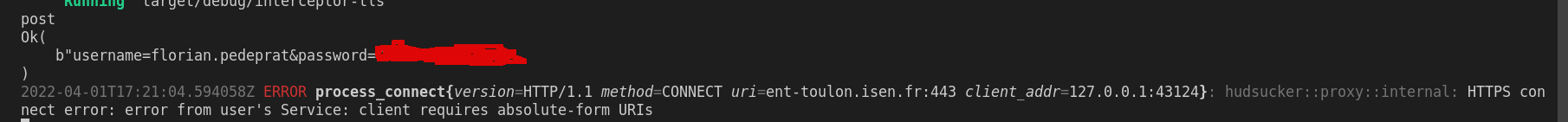
Si on demande au navigateur d’afficher le faux certificat de google.com, on a ceci :

Une image contenant table

Description générée automatiquement

Une image contenant texte

Description générée automatiquementNous avons testé notre proxy sur un autre site internet, afin de vérifier que la génération de certificats à la volée était bien fonctionnelle. Suite à la demande de connexion du client, nous avons les logs suivants qui sont les échanges entre le client et le proxy, on peut voir une première requête effectué par le client, puis la réponse renvoyée par le proxy. Le header contient les cookies en clair, le body ne s’affiche pas dans ces logs car c’est un objet.

Nous avons testé sur le site de l’isen de nous connecter et de récupérer au moment de la requête post le login et le mot de passe. Nous avons formaté l’affichage des logs afin de pouvoir lire le contenu du body. Voici ce que nous avons obtenu :

Une image contenant texte

Description générée automatiquementNous avons également intercepté les requêtes sur wireshark :

Une image contenant texte

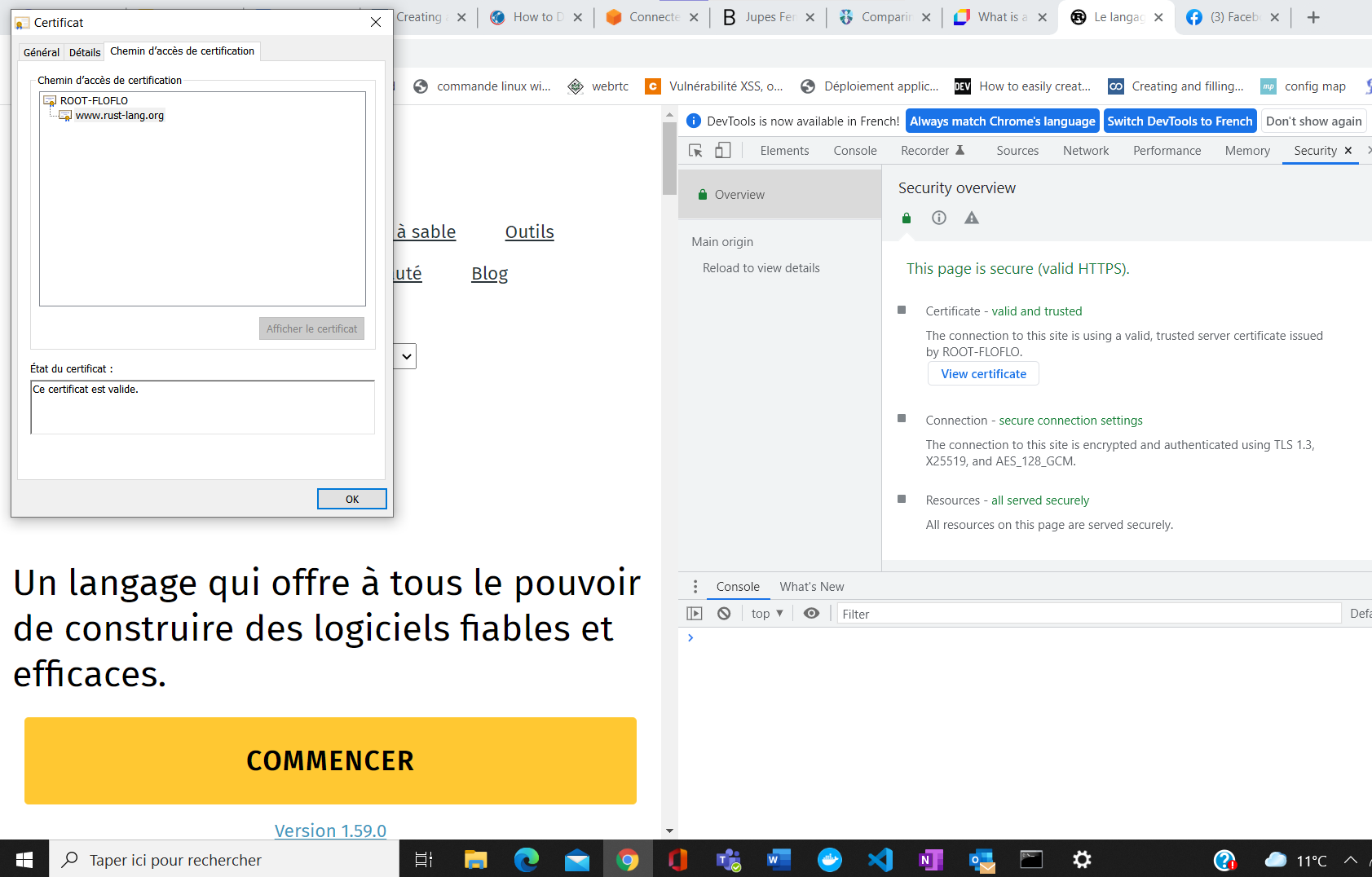
Description générée automatiquement

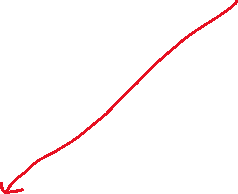
On peut voir la version TLS utilisées, et les données sont bien chiffrées entre le client et le proxy.

Nous avons également lancer le proxy sur un pc et nous avons configuré sur le navigateur d’un autre pc le proxy pour lui dire de se rendre sur notre proxy.

Une image contenant texte

Description générée automatiquementComme on peut le voir ici, nous avons configuré le proxy à l’adresse ip sur lequel tourne le proxy : 192.168.43.151 :2000

Après quoi, nous allons sur le navigateur ou l’on peut voir que la communication est sécurisée et le certificat est considéré comme valide



3 – Autorité de certification

1. Structure

La structure de notre autorité de certification est la suivante :



ROOT CA

Niveau 1



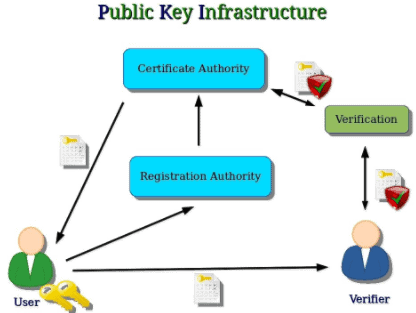


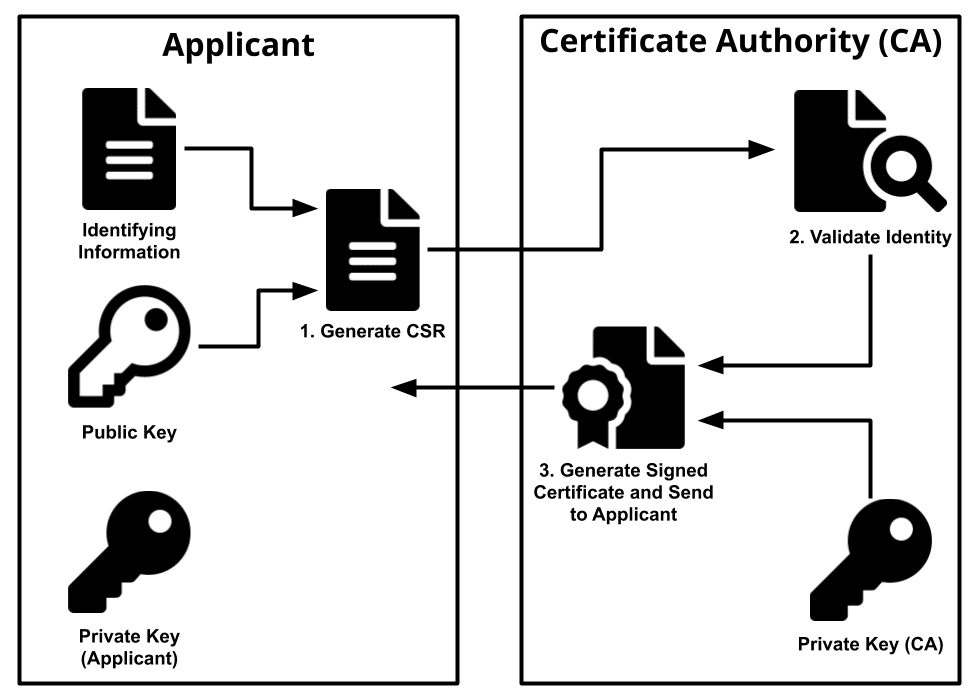


Nous avons une autorité de certification racine qui possède une paire de clef privé/publique.

Cette root CA possède un certificat auto signé, c’est ce certificat que nous allons incorporer dans le magasin de certificat du navigateur.

N’ayant pas de HSM, nous pouvons protéger la clef privée de la root CA dans un fichier protégé par un mot de passe.

La root CA va générer des certificats pour les noms de domaine visités par le client à la demande du proxy qu’elle signera avec sa clef privée. Le navigateur pourra, dans son magasin, trouver la clef publique de la root CA et vérifier la signature du certificat que le proxy a envoyé.



1. Implémentation

Tout d’abord nous avons crée le certificat de la root CA à l’aide de commandes openssl.

Une image contenant texte

Description générée automatiquementCette commande permet de générer la clef privé RSA de 2048 bit.

Cette commande permet de générer un certificat auto-signé avec la clef privée générée au préalable

Une image contenant texte

Description générée automatiquement

Nous utilisons l’algorithme RSA avec une paire de clef privé/publique de taille 2048 bit.

Pourquoi ce choix ?

RSA est un algorithme de chiffrement qui se base sur la factorisation de grand nombres premiers en utilisant les congruences sur les entiers et le petit théorème de Fermat.

Les fonctions qu’utilisent cet algorithme sont dites à sens unique puisque on chiffre avec la clef publique et on déchiffre avec la clef privée. L’inverse n’est pas possible.

Aujourd’hui la sécurité de l’algorithme RSA repose sur le fait que pour casser le RSA et donc découvrir la clé privée, il faut factoriser le nombre n. La factorisation ce nombre est aujourd’hui considérée comme un problème difficile à résoudre en effet Les nombres aléatoires premier choisis sont très grand, et résoudre une telle opération mathématique nécessité de très grosses ressources qui ont un coût vraiment très élevé.

Par ailleurs en 25 ans malgré les tentatives pour casser le RSA aucune n’y ait parvenu.

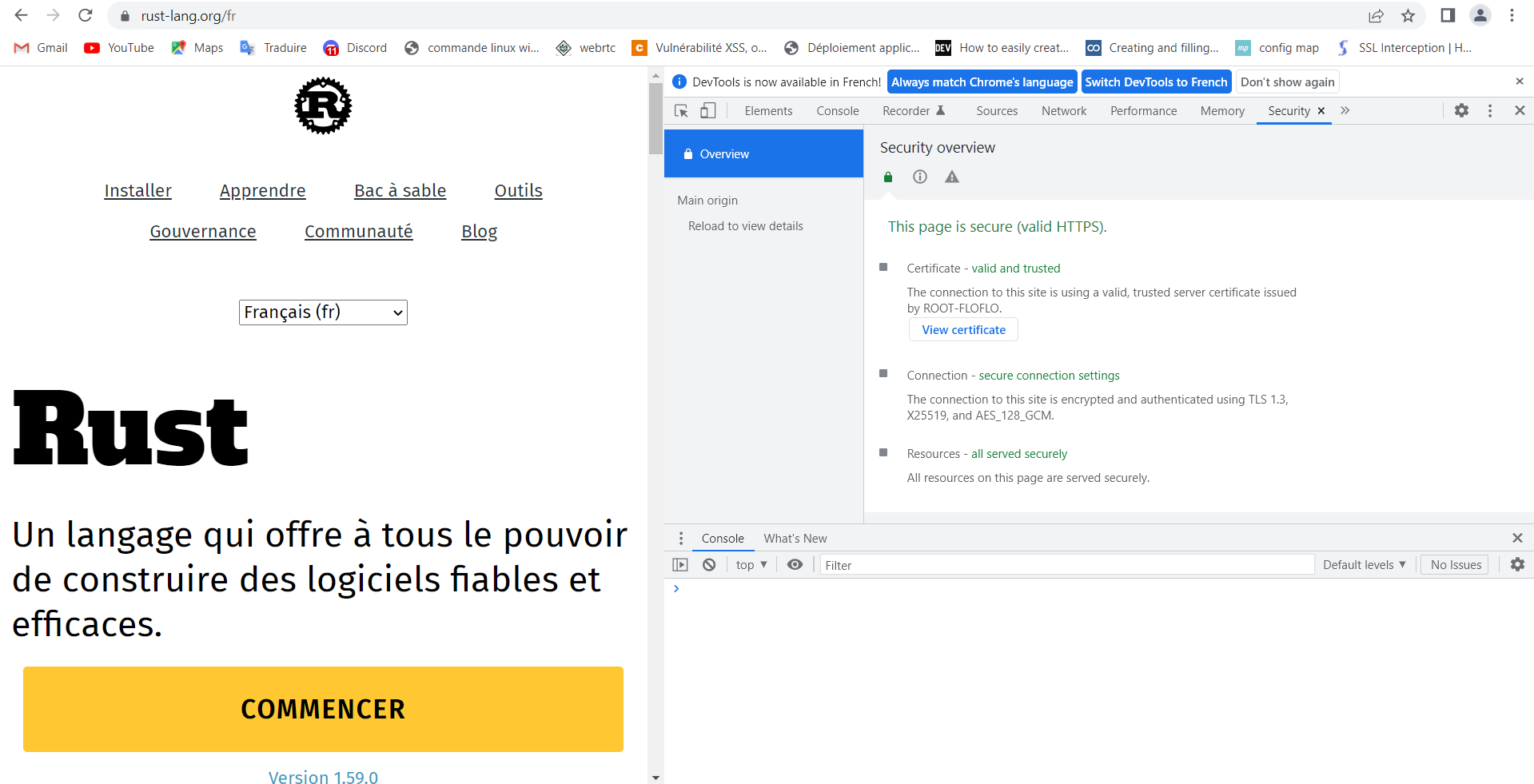
La taille de clé recommandée pour le RSA est 2048 bits.

La puissance de calcul des machines ne cesse continuellement d’augmenter, ce qui va permettre à l’algorithme de générer des clefs plus grosses apportant une sécurité plus élevée, rendant la tâche des attaquants plus compliqué pour casser les clefs.

Les certificats utilisés pour les connexions sont signés par notre autorité de certification racine.

Ils sont générés à la volée dans notre code source. Les paires de clefs sont générées avec le protocole de chiffrement RSA et ont une taille de 2048 bits. Les certificats quant à eux sont en version V3 avec une signature en SHA256.

4 – Autres choix d’implémentations

Lorsqu’on ouvre l’inspecteur du navigateur, dans l’onglet sécurité, nous pouvons constater la configuration du TLS utilisée pour la communication.

La version utilisée sur ce site est 1.3. Quant à l’algorithme de chiffrement, est utilisé l’AES 128 bit GCM avec la courbe elliptique X25519.

La librairie OpenSSL nous permet de mettre en cache les certificats générés à la volée. Ceci va permettre d’accélérer la navigation pour l’utilisateur.

5 – Difficultés rencontrées

Nous avons rencontré des difficultés à établir une connexion entre le serveur et le proxy car nous n’avons pas fait attention à la structure des requêtes, notamment à la requête CONNECT.

Lorsqu’un client effectue une demande de connexion un site internet, le navigateur regarde s’il doit se connecter au proxy en premier lieu. Lorsque c’est le cas, le navigateur va effectuer une requête CONNECT au proxy de la façon suivante :

Cette requête, une fois reçue par le proxy, indique que le client souhaite se connecter sur l’URL précisée. Le proxy récupère le contenu de la requête CONNECT, et effectue une requête au nom du client, vers le serveur distant, au travers d’un tunnel TCP, qu’il aura ouvert au préalable. Bien évidemment, le proxy n’interprète et ne modifie pas les requêtes effectuées par le client.

6 – Améliorations envisageables

On a vu précédemment que les entreprises utilisent des proxy intercepteurs TLS pour protéger les systèmes d’informations. On pourrait envisager de définir une white list. Cette white list contiendrait les sites que les employés seraient autorisés à visiter. Dans notre code source il suffirait d’intercepter l’url que le client souhaite visiter et comparer celle-ci avec notre white list.

On pourrait aussi lors de l’interception d’une requête ou d’une réponse modifier le contenu.

Une amélioration majeure à envisager serait aussi d’améliorer la chaine de certification en rajoutant une sous CA. Cette sous CA signerai les certificats générés à la volée à la place de la root CA.

Pourquoi ce serait mieux niveau sécurité ? Pour la génération des certificats, nous avons besoin de la clef privée de la root CA, la sécurité du TLS repose sur la confidentialité de cette clef privée. C’est pour cette raison que cette clef ne devrait pas se retrouver exposée de la sorte. La convention est d’avoir une root CA qui signe une sous CA, qui elle, signe les certificats. La root CA permet donc de prouver l’authenticité de la sous CA.

7 – Conclusion

Comme on a pu le voir, l’intercepteur TLS est un outil très puissant puisqu’il met à nu les communications qui pourrait survenir entre un client et un serveur distant. Cependant, il ne respecte pas les principes du protocole TLS qui sont : la confidentialité, l’authenticité, l’intégrité. L’utilisation de cet outil devrait donc faire l’objet d’une justification très solide auprès des autorités nationales de sécurité.

Non seulement, le TLS a permis d’augmenter le niveau de sécurité des échanges sur internet, mais il a aussi été une occasion pour les cybercriminels de passer inaperçus. Les organisations vulnérables à leurs attaques ont su démontrer l’intérêt d’un tel outil pour assurer la protection de leurs systèmes d’information. L’usage de l’intercepteur TLS peut alors être justifié dans ce type de cas mais à quelques conditions : les employés doivent etre clairement informés, les données analysées doivent être explicitement décrites, l’outil se doit d’intervenir uniquement dans l’intérêt de l’entreprise et non à l’encontre d’une des personnes physiques de l’entreprise.

8 – Ressources utilisées

Ressources

Images :

<https://www.digitalberry.fr/connexion-non-securisee-messages-alerte-certificats/>

<https://www.01net.com/actualites/pourquoi-les-antivirus-et-les-proxy-ssl-rendent-le-web-moins-sur-1113890.html>

<https://kidan.co/fr/blog/quest-ce-que-linfrastructure-a-cle-publique-icp-et-comment-securise-t-elle-votre-organisation/>

<https://d1smxttentwwqu.cloudfront.net/wp-content/uploads/2019/07/ca-diagram-b.png>

<https://www.researchgate.net/profile/Sufyan-Al-Janabi/publication/287994115/figure/fig3/AS:310233172856834@1450976760846/X509-V3-certificate-12.png>

<https://www.ionos.fr/digitalguide/sites-internet/developpement-web/rust-langage-de-programmation/#:~:text=des%20applications%20Web.-,Particularit%C3%A9s%20de%20Rust,%C2%AB%20panic!%20%C2%BB%20se%20d%C3%A9clenche>.

Site web :

<https://ldapwiki.com/wiki/SSL-TLS%20Interception>

<https://wiki.openrightsgroup.org/wiki/TLS_interception>

<https://greenlock.ghost.io/tls-1-3/>

<https://interstices.info/nombres-premiers-et-cryptologie-lalgorithme-rsa/>

<https://docs.rs/>

<https://www.ssi.gouv.fr/guide/recommandations-de-securite-concernant-lanalyse-des-flux-https/>

<https://www.cnil.fr/fr/analyse-de-flux-https-bonnes-pratiques-et-questions>

<https://www.legifrance.gouv.fr/loda/id/JORFTEXT000000801164/>

<https://www.helpnetsecurity.com/2017/03/08/https-interception-dilemma/>

<https://www.codeheroes.fr/2021/01/18/premiers-pas-avec-le-langage-rust/>

<https://www.secureworks.com/research/transitive-trust>

<https://stackoverflow.com/questions/11697943/when-should-one-use-connect-and-get-http-methods-at-http-proxy-server>