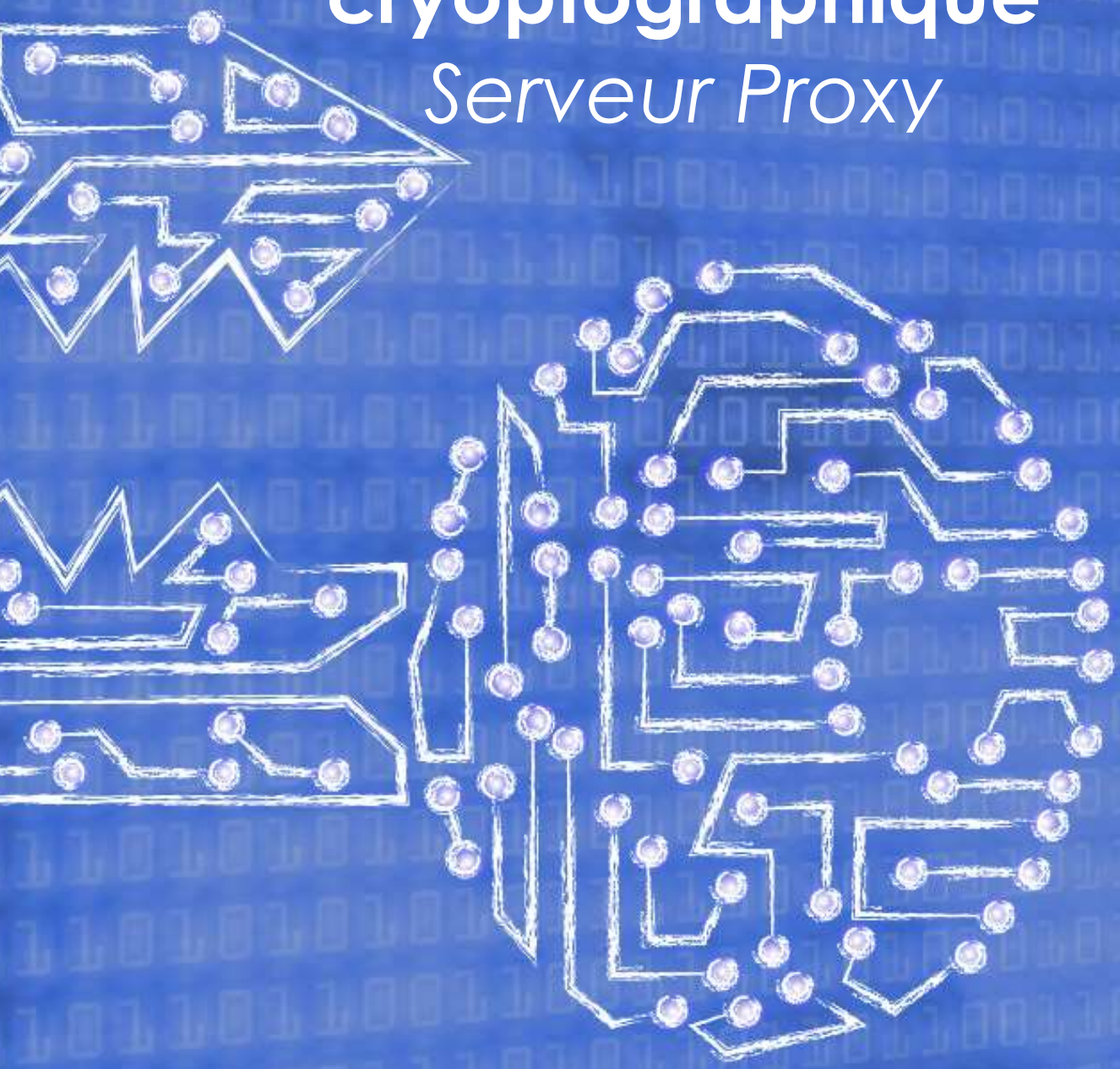
Compte rendu du projet cryoptographique

Serveur Proxy



AVRIL 15

BAH Rayan

PERUCCA Clément

MEUNIER Thomas

ISEN
ALL IS DIGITAL
MÉDITERRANÉE
yncréa

1 INTRODUCTION

1.1 Réalisation du projet

Dans le cadre de notre cours de cryptographie avec monsieur GIRARD Pierre, nous avions pour projet d'analyser le fonctionnement d'un proxy HTTP avec interception TLS afin d'implémenter ce dernier.

SOMMAIRE

2 SERVEUR PROXY

2.1 Définition

Un serveur proxy permet pour un ensemble de protocoles de relayer au nom de certains clients internes à un réseau les communications avec l'extérieur. Du côté du client, l'application communique avec le serveur proxy au lieu du serveur réel et du côté du serveur, l'application communique avec le serveur proxy au lieu du client.
Pour le serveur, il n'y a aucune différence entre communiquer avec le serveur proxy ou le client et inversement.

2.2 Objectif du serveur proxy

L'intérêt majeur d'un serveur proxy est de donner de la connectivité Internet à l'ensemble des machines locales tout en les maintenant invisible de l'extérieur, mais il existe de nombreux autres intérêts tels que :

- **Contourner la censure web**
- **Accélérer l'affichage des pages web**
- **Bloquer l'accès à certains sites internet**

Les serveurs proxy filtrant ainsi les requêtes HTTP dans les différents objectifs, ils doivent implémenter de l'interception TLS car actuellement, la majorité des requêtes HTTP sont sécurisées avec du TLS.

3 INTERCEPTION TLS : INTERETS, ENJEUX ET INCONVENIENTS

3.1 Définition

L'interception TLS correspond tout simplement à l'implémentation d'un proxy TLS entre le serveur et le client.
De ce fait, on distingue deux types d'interception TLS :

- **L'interception TLS sortante est la pratique la plus courante ; elle va permettre d'intercepter toutes les connexions dont les sources sont souvent les utilisateurs d'une entreprise vers des sites distants. Pour cela, les utilisateurs passeront, de façon transparente', par un proxy TLS qui interceptera les requêtes, puis ce dernier va initialiser lui-même les connexions vers les diverses destinations.**

1. Chris veut établir une connexion avec « www.isen.fr » et envoie une requête *Client Hello* à « www.isen.fr »

2. Le proxy TLS est positionné dans l'architecture réseau de telle sorte qu'il soit sur le chemin allant vers « www.isen.fr ». Il intercepte la requête TLS et met le client en attente puis il établit une connexion avec « www.isen.fr », soit en reprenant exactement les paramètres que le client a indiqué dans son *Client Hello,* soit en modifiant certaines valeurs pour accentuer les paramètres de sécurité par exemple.

3. Le serveur « www.isen.fr » établit une connexion sans problème particulier. Pour lui, le proxy TLS est un client comme un autre. Le serveur renvoie les paramètres qu'il a choisi ainsi que son certificat et sa chaîne de certificat.

4. Après que le proxy TLS ait reçu le certificat du serveur et sa chaîne, il procède aux vérifications d'usage et décide si la connexion peut être établie. Une fois la session TLS établie, elle reste en attente jusqu'à ce que le vrai client envoie des données. Par la suite, le proxy TLS va reprendre les données du certificat du serveur pour en recréer un autre temporaire et qui sera signé par son AC interne à l'entreprise « proxytls.local » et qui est connu par les navigateurs de l'entreprise. Puis ce nouveau certificat sera utilisé par le proxy TLS pour usurper l'identité de « www.isen.fr » auprès du client.

- **L'interception TLS entrante pour laquelle le serveur accessible par les clients se situe en interne du système géré alors que les clients proviennent de n'importe où.**

1. **Bob veut établir une connexion sécurisée avec « www.isen.com ». Il envoie une requête *Client Hello* à «** www.isen.fr **»**

2. Le proxy TLS intercepte la requête du client. Il récupère les différents paramètres que le client a initié et établi une connexion TLS avec le serveur interne.

3. Le proxy répond au client avec un *Server Hello* et envoie le certificat du serveur. La connexion TLS s'établit normalement. Puisque le certificat est valide (il correspond au domaine requêté et est signé par une AC publique), aucune erreur n'est détectée par les clients.

3.2 Intérêts et inconvénients

Le principal avantage du proxy TLS est qu'il ne nécessite pas de déchiffrement mais simplement deux connexions TLS distinctes du client au proxy et du proxy au serveur ce qui permet au proxy d'avoir le contenu en clair.
Néanmoins, cette pratique a une limite. Pour l'interception sortante, elle est concernée par une rupture de la confiance. En effet, habituellement le client vérifie l'identité du serveur de son côté. Or, dans cette architecture basée sur l'interception, le client établit une session TLS avec le proxy TLS à son insu, donc le certificat renvoyé par le proxy TLS ne correspond pas, aussi bien sur son nom que sur l'autorité de certification, à ce qu'a demandé le client.
Quant à l'interception entrante, même si elle permet d'éviter la création de certificats à la volée, elle nécessite un doublon des clés publiques et privées sur le réseau car le proxy TLS porte les clés du serveur interne pour que le client puisse les obtenir.

3.3 Enjeux juridiques

L'interception TLS ne respecte pas la sécurité du TLS définie dans le RFC (Request For Comments) et n'est donc pas une technique normalisée. Chaque équipement, chaque logiciel et chaque compagnie utilise sa propre implémentation sans oublier que l'interception TLS n'est rien de plus qu'une MiTM Attack si elle est utilisée sans le consentement des utilisateurs ; un accord entre les différentes parties est donc de mise. Chaque pays ayant des juridictions différentes, nous nous concentrerons sur la juridiction française.

La CNIL (Commission Nationale de l'Informatique et des Libertés) recommande de protéger les canaux de communications à l'aide du protocole HTTPS car il réduit considérablement les risques liés à l'interception de communication que ce soit en termes d'écoute ou de modification mais soulève tout de même un problème de sécurité qui est celui du déchiffrement des flux pour analyser leur contenu avec des outils de sécurité ad-hoc avant de les re-chiffrer. En effet, ce déchiffrement est légitime pour l'employeur car il assure la sécurité de son système d'information donc l'interception TLS devient légitime dans le cadre de la sécurité sous certaines conditions (gestion stricte des droits d'accès aux administrateurs, minimisation des traces conservées, protection des données d'alertes extraite de l'analyse, information précise sur les raisons du déchiffrement…).

De plus, concernant le STAD (Systèmes de Traitements Automatisés Données), selon les articles 323-1 à 323-7 du code pénal, il est formellement interdit « *d'accéder ou de se maintenir, frauduleusement, dans tout ou partie d'un STAD […] et d'entraver ou de fausser son fonctionnement* ». La CNIL insiste donc en rappelant que le recours du

déchiffrement / re-chiffrement devrait donc nécessiter une base légale justifiant que des mesures techniques déployées par des tiers, garantissant ainsi la confidentialité des échanges.

L'ANSSI (Agence Nationale de la Sécurité des Systèmes d'Information) rejoint cet avis : « *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.* »

Finalement, la mise en place d'outils tels que l'interception TLS doit obéir à des dispositions légales relatives à la cryptologie dont la violation peut être sanctionnée de différentes manières : administratif, civil ou pénal selon les articles 34, 35, 36 et 37 du LCEN (Loi de la Confiance de l'Economie Numérique). Par exemple, les peines encourues pour non-déclaration ou communication d'informations sont de 15.000 euros d'amende et un an d'emprisonnement.

4 IMPLEMENTATION

4.1 Moyens utilisés

Nous avons d'abord essayé d'implémenter le proxy en java cependant nous n'avons pas trouver beaucoup de ressource et n'étions pas trés à l'aise sur le langage. Nous nous sommes donc orientés vers du python car c'est le langage que nous maitrisons le mieux et ou on trouvait le plus de ressources.

Httplib : définit les classes qui implémentent le côté client des protocoles HTTP et HTTPS.

Urlparse : décomposer les chaînes d'URL en composants pour combiner les composants en une chaîne d'URL, et convertir une "URL relative" en une URL absolue à partir d'une "URL de base".

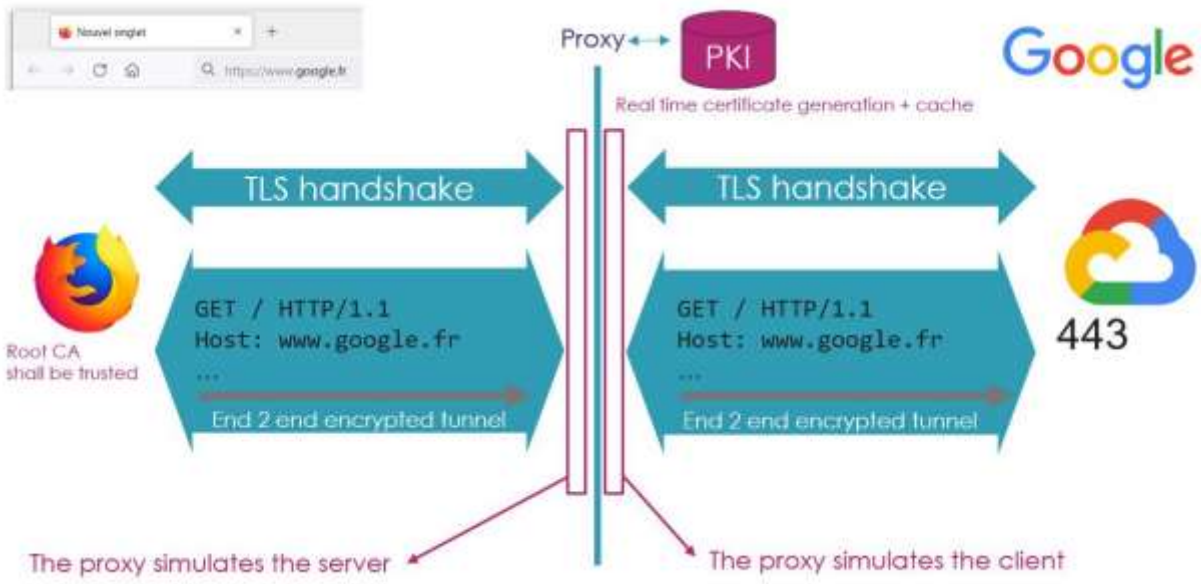
HTTPServer,BaseHTTPRequestHandler : définit deux classes pour l'implémentation de serveurs HTTP (serveurs Web).

Socket : fournit un accès à l'interface socket BSD.

Ssl : permet d'accéder aux fonctions de cryptage et d'authentification par les pairs de TLS pour les sockets réseau, tant du côté client que du côté serveur. Ce module utilise la bibliothèque OpenSSL.

Subprocess : permet de créer de nouveaux processus, de se connecter à leurs tuyaux d'entrée/sortie/erreur et d'obtenir leurs codes de retour.

4.2 Objectifs et démarche



En effet, pour réussir l'implémentation de notre serveur, nous avons suivi plusieurs étapes :

- **Etape 1 : Implémentation d'un proxy HTTP. En effet, si son implémentation est bonne, il devrait remplir entièrement son rôle mais uniquement pour les requêtes HTTP.**
  Pour cette première étape nous nous sommes inspirés de différents codes trouvé sur différents sites. Il en a résulté le code proxyHTTP.py qui gère les requêtes http.

  Nous avons créé une classe « server » avec des méthodes afin de gérer les différentes requêtes :

```
class Server:

    def __init__(self, config):
        signal.signal(signal.SIGINT, self.shutdown)
        self.serverSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)      # Cree une socket TCP
        self.serverSocket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.serverSocket.bind((config['HOST_NAME'], config['BIND_PORT'])) # Associe la socket au serveur proxy
        self.serverSocket.listen(10)
        self.__clients = {}
```

  Les méthodes sont les suivantes : ListenForClient() qui permet de get les clients durant l'exécution du thread, shutdown() afin d'arrêter le proxy et enfin proxy_thread() qui est le thread qui s'exécute .C'est le cœur de notre proxy http

- **Etape 2 : Création d'une Autorité de Certification Racine (ACR) avec une paire de clés et un certificat afin de pouvoir gérer nos certificats numériques (aussi bien pour leurs délivrances que leurs révocations).** ***Pour gérer nos certificats et la génération de CA et de clés nous avons utilisé open SSl. Pour se faire nous avons effectué les commandes suivantes sous linux :***

  $: openssl genrsa -out ca.key 2048
  $: openssl req -new -x509 -days 3650 -key ca.key -out ca.crt -subj "/CN=proxyISENCA"

  Ces deux commandes nous **permettent** ***de générer un certificat x509 pour notre CA et une paire de clé rsa de 2048 bits. On utilise également les commandes :***

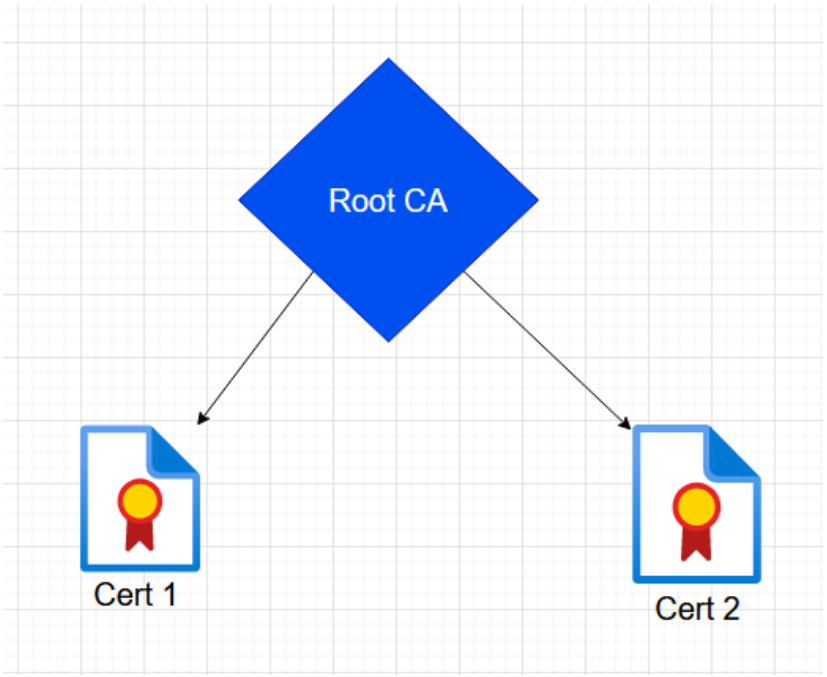
  $: openssl genrsa -out cert.key 2048

  $: openssl req -new -key cert.key -subj /CN = isen openssl x509 -req -days 3650 -CA ca.cert -CAkey ca.key

  Ces deux commandes nous permettent de générer une paire de clé et un certificat pour le domaine ISEN.

  Le choix du RSA nous semblez le plus adapté, en effet, 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.
  Pour ce qui est du choix de la taille de clé, 2048 est la taille recommandée pour le RSA.

Architecture de CA :



Notre autorité de certification possède une paire de clé privé/publique ainsi qu'un certificat auto-signé. La clé privée de la CA est stockée dans un dossier sur le PC ce qui n'est pas idéal. C'est cette autorité qui signera tous les certificats pour les différents sites

- **Etape 3 : On insère le certificat dans le navigateur Firefox. Nous pouvons désormais générer un certificat et une paire de clés issu de l'ACR pour le site de notre choix.**

- **Etape 4 : Extension du proxy aux requêtes HTTPS.**

Pour cette étape nous avons essayé d'étendre notre proxy http à l'HTTPS en utilisant le certificat qu'on génère à l'étape 3. Cependant nos certificats générer en ligne de commande présentait des erreurs lors du test du proxy https. Nous avons donc cherché un projet proxy https fonctionnelle pour pouvoir se l'approprier.

La génération de certificat se fait avec OpenSSL qu'on exécute sur le code python grâce aux Subprocess.

```
p1 = Popen(["openssl", "req", "-new", "-key", self.certkey, "-subj", "/CN={}".format(hostname)], stdout=PIPE)
p2 = Popen(["openssl", "x509", "-req", "-days", "3650", "-CA", self.cacert, "-CAkey", self.cakey, "-set_serial", epoch, "-out", certpath], stdin=p1.stdout, stderr=PIPE)
```

Ici la première ligne crée un CSR avec la clé privé cert.key qui prend le nom de l'hostname.

La deuxième commande permet de signer la CSR avec la clé publique de la CA. On choisit aussi la durée du certificat, ici 10 ans. On place la CSR signé dans le dossier cert.

```
self.connection = ssl.wrap_socket(self.connection, keyfile=self.certkey, certfile=certpath, server_side=True)
```

Ensuite cette ligne de code permet d'associer le certificat crée signé par la CA à la socket.

- **Etape 5 : On filtre le contenu et on teste les configurations TLS du proxy. Nous disposons désormais d'un proxy http avec interception TLS.**

4.3 Difficultés rencontrées

Implémentation HTTPS :

La complexité du protocole TLS et la gestion des requêtes HTTPS a été fastidieuse. Nous ne sommes pas arrivés à implémenter notre génération de certificat dans notre proxy. Malgré cela nous avons beaucoup appris sur tout cela.

4.4 Améliorations possibles

Une première amélioration possible serait d'ajouter une black List avec le lien des sites qu'on ne voudrait pas autoriser. Ainsi lors de la connexion, lorsque le proxy récupère le lien du site il va vérifier si le site fait partie de la liste et interdire la connexion si le site en fait partie.

Un autre problème est la manipulation de la clé privée de la root CA. Il faudrait donc sécuriser le stockage de cette clé. On pourrait aussi décider de changer la structure de chaine de confiance des certificats en ajoutant une sous CA. Elle serait signée par la Root CA et ce serai la sous CA qui signe les certificats. La Root CA permettrait donc seulement de prouver l'authenticité de la sous CA.

Voici également une liste de recommandations que nous pouvons faire concernant la gestion de certificats :

- **Utiliser une AC opérée sur le territoire national**
- **Utiliser une AC qui propose des certificats compatibles avec les exigences formulées par le RGS.**
- **Utiliser une AC reconnue comme étant de confiance par une large majorité des navigateurs web du marché.**
- **Choisir un prestataire qui est en mesure d'apporter des éléments prouvant son sérieux : accès sécurisé au service client, engagements du support technique, certification 35, délivrance d'EV Certificate (comme WebTrust For CA www.//webtrust.com)**

5 CONCLUSION

L'interception TLS semble ainsi utile et avantageuse dans de nombreux points cités précédemment. Néanmoins, cela présente des risques tel que la fuite de filtrage de contenu. Si un potentiel attaquant récupère les données concernant les données qui sont acceptés comme "sans risques" aux yeux du proxy on peut imaginer que cela permettra à cet attaquant de monter une attaque qui ne sera pas détecté par le proxy. Ainsi, si ces fuites ne sont pas gérées, l'utilité d'un proxy est compromise. Cela nous amène donc à nous poser les questions suivantes : Le proxy https est-il la meilleure solution actuellement pour les entreprises ? Que faut-il mettre en place pour empêcher les différentes formes d'attaques sur un proxy ?

6 RESSOURCES

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

https://www.ssi.gouv.fr/uploads/IMG/pdf/NP_TLS_NoteTech.pdf

https://www.ssi.gouv.fr/entreprise/reglementation/confiance-numerique/le-referentiel-general-de-securite-rgs/

https://www.varonis.com/fr/blog/serveur-proxy