# Proxy sortant avec prise en charge de HTTPS et sans rupture du chiffrement

**Laurent VERHEIRSTRAETEN**
Université de Rennes 1 – DSI Pôle Applications et Services Numériques
Campus de Beaulieu
263 Avenue du Général Leclerc – 35042 Rennes Cedex

## Résumé

*L'objectif du projet est d'empêcher le téléchargement d'outils de piratage sur des serveurs via une session sortante, tout en permettant les mises à jour légitimes des composants, système d'exploitation et logiciels applicatifs.*

*L'architecture retenue est un proxy (mandataire) sortant (Squid). Le mode transparent est indispensable car les clients web légitimes sont trop nombreux et variés pour les configurer chacun.*

*Les principes de mise en œuvre sont :*

- *un équipement actif du réseau redirige toutes les sessions TCP sortantes sur les ports 80 et 443 vers un serveur proxy chargé de filtrer les connexions autorisées ;*
- *le proxy Squid rejette ce qui ne figure pas en liste blanche.*

*L'accent est mis sur la technologie « Peek and Splice » qui exploite l'élément Server Name Indication (SNI) échangé avant que le chiffrement ne soit effectif dans les nouvelles évolutions de TLS.*

*Ce SNI est donc disponible pour filtrer les sessions sortantes avec une liste blanche, sans déchiffrement du flux, celui-ci étant simplement transporté via un tunnel TCP. Avec cette solution, il n'est plus nécessaire de générer des certificats à la volée pour que le proxy fonctionne en mode vraiment transparent.*

*Sur cette base, le projet consiste alors essentiellement à élaborer une liste blanche des Fully Qualified Domain Names (FQDN) couvrant tous les besoins de mises à jour après une période d'observation des logs.*

*La vérification des logs peut permettre le cas échéant, de repérer une éventuelle tentative de téléchargement qui pourrait sembler suspecte.*

## Mots-clefs

*SQUID, Proxy sortant, mandataire, HTTP, HTTPS, Peek and Splice, Server Name Indication*

# 1 Un proxy sortant ?

Au sein de l'Université de Rennes 1, la DSI a en charge les infrastructures et systèmes pour un public composé de 29 000 étudiants et près de 3700 personnels. Les serveurs mettant en œuvre les différents services proposés par la DSI sont au nombre d'environ 500.

Historiquement, un serveur proxy HTTP (mandataire) sortant était mis en place devant les serveurs.

La solution logicielle pour ce proxy sortant était Squid. Il avait pour rôle de gérer les accès sortants HTTP des serveurs physiques ou virtuels vers internet.

Ces accès sont en majorité des mises à jour des systèmes d'exploitation (Windows ou Linux).

Dans cette configuration, seuls les accès en HTTP sont pris en charge par le serveur proxy.

Les accès sont filtrés par un système de listes blanches autorisant certaines URLs ou certains domaines pour une ou un groupe de machines.

Les accès en HTTPS ne sont pas pris en charge par le proxy et les sorties vers internet ne sont pas filtrées.

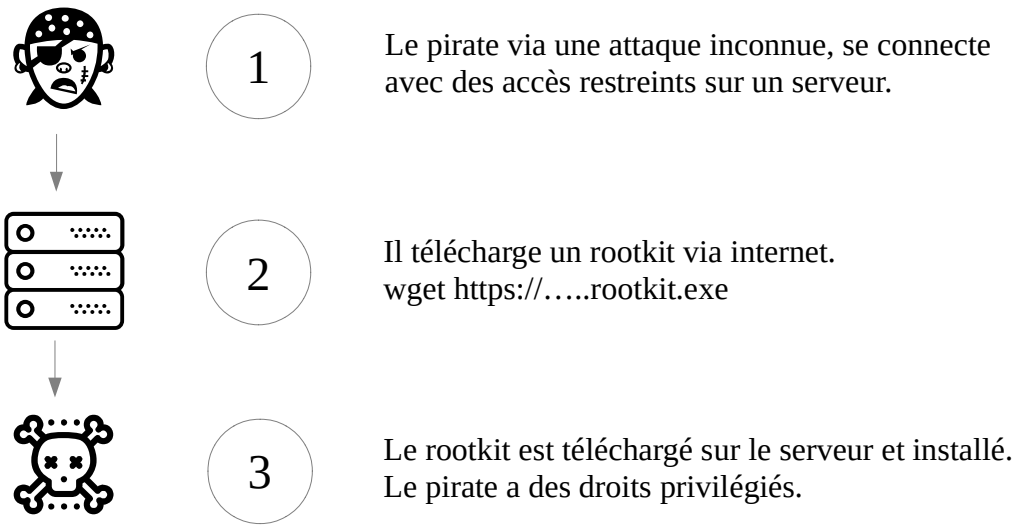
## 1.1 Pourquoi ?

La mise en place d'un proxy sortant ne vise pas seulement à pouvoir autoriser ou enregistrer les demandes de sorties des serveurs.

L'un des objectifs principaux est la protection du système d'information en évitant le téléchargement d'outils de piratage via un serveur. En limitant les possibilités de sorties des serveurs vers seulement une liste de domaines autorisés, on élimine les téléchargements par une personne mal intentionnée depuis des sites pouvant accueillir des programmes ou logiciels aussi nombreux que dangereux.

Nous souhaitons éviter les phénomènes de rebond entre les serveurs . Le pirate ou les bots d'attaque abandonnent s'ils ne parviennent pas à télécharger les *rootkits*.

Dans le scénario décris ci-dessous, un pirate via une première attaque sur un serveur est parvenu à se connecter sur un de nos serveurs. Par une commande simple comme *wget* ou *curl*, il télécharge un *rootkit* (ensemble d'outils permettant de dissimuler et/ou d'obtenir un accès privilégié sur une machine) depuis un serveur sur internet. Une fois le téléchargement effectué, il installe le *rootkit* sur le serveur et obtient des privilèges sur celui-ci.



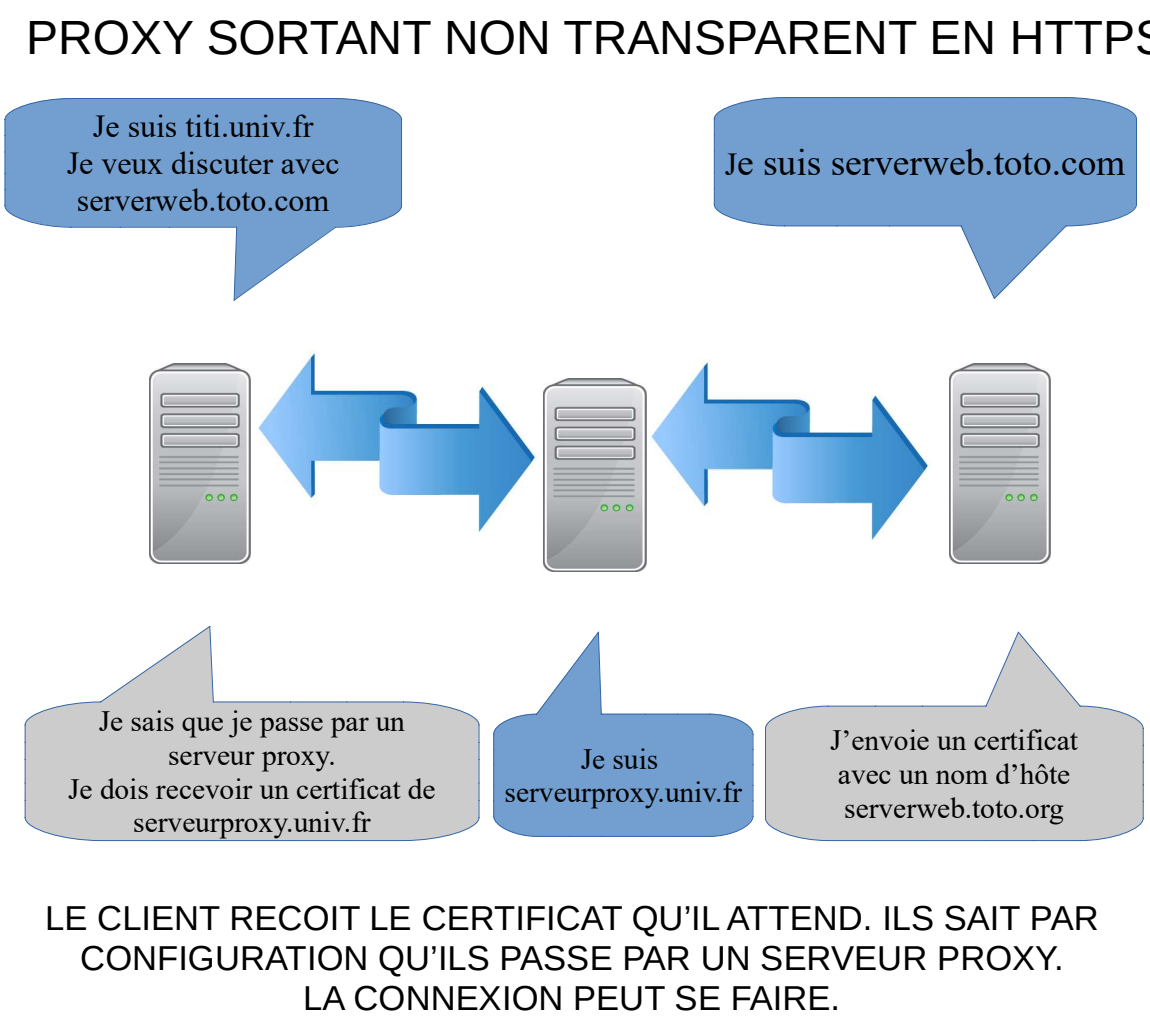
C'est justement ce cas de figure que nous souhaitons éviter. Même si il n'est pas possible d'éliminer la première attaque sur le serveur, le proxy mis en place bloquera le téléchargement du *rootkit* depuis internet.

Le protocole HTTP est en train de disparaître. La prise en charge du protocole HTTPS est primordiale dans la mise en place d'un serveur proxy. Le protocole HTTP devient même un problème de sécurité s'il reste utilisé en majorité.

## 1.2 Quel type de proxy ?

Le choix du type de proxy sortant se révèle important. Deux choix sont possibles.

La première possibilité est un proxy dit « non transparent ». Dans cette configuration, chaque machine derrière le proxy doit être configurée afin de l'utiliser.

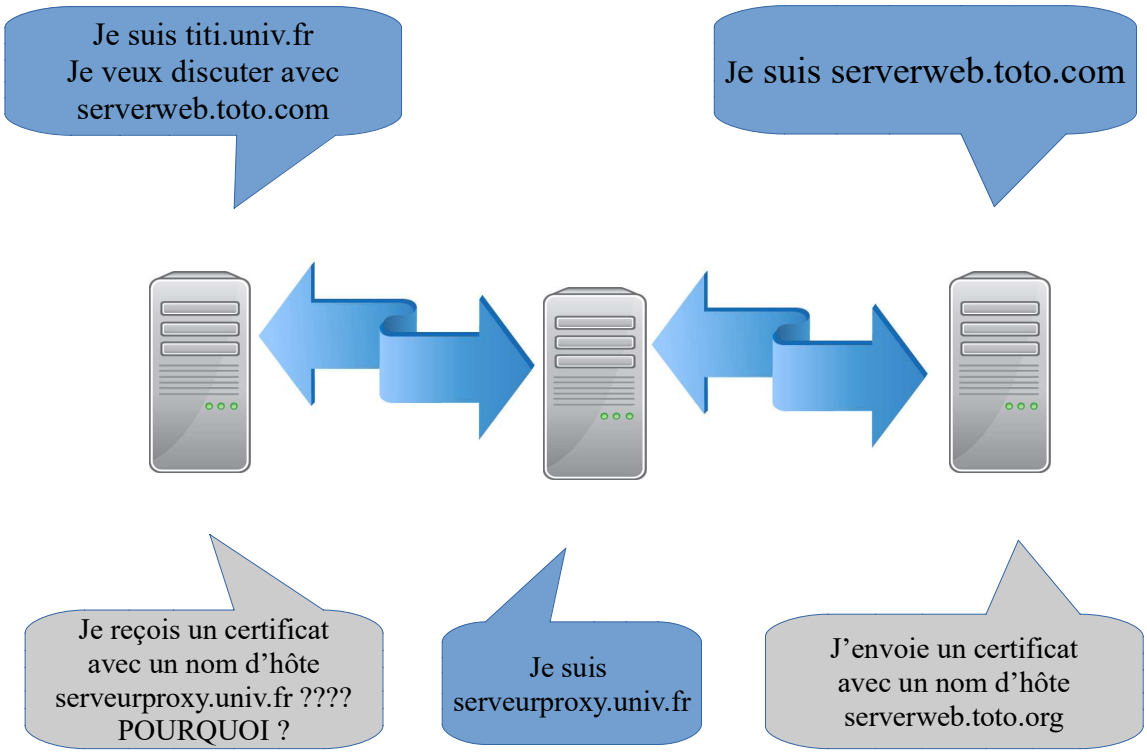



Cette solution implique de configurer chaque programme ou application avec les paramètres du proxy. Dans un environnement comprenant de nombreux serveurs et d'applications, cette solution n'est pas viable.

La deuxième possibilité était un proxy dit « transparent ». Cette architecture ne nécessite pas de configurer les serveurs et applications clientes du proxy. Elle nécessite de mettre en œuvre une redirection des flux sortants vers le serveur proxy.

Dans le cas d'une connexion en HTTPS, une difficulté apparaît : le client doit authentifier le serveur avec lequel il souhaite converser, mais il ne voit que le serveur proxy.

## PROXY SORTANT TRANSPARENT EN HTTPS
## Sans « Peek and Splice »




LE CLIENT RECOIT UN CERTIFICAT QUI NE CORRESPOND PAS A CE QU'IL ATTEND.
IL NE SAIT PAS QU'IL PASSE PAR UN SERVEUR PROXY.
LA CONNEXION NE PEUT PAS SE FAIRE.

Il faut donc que le serveur proxy possède un moyen de faire l'intermédiaire pour l'authentification. Squid possède à partir de sa version 3.5 une méthode nommée « Peek and Splice » qui permet de faire cela.

# 2 Les principes fondamentaux

L'architecture correspondant aux choix techniques est simple à mettre en place. Cependant, il est nécessaire d'impliquer des compétences réseaux dans un projet comme celui-ci.

Le principe de notre infrastructure est simple. Le flux HTTP utilisant le port 80 désirant sortir est automatiquement redirigé vers le serveur proxy sur un port d'écoute particulier de celui-ci. Ce fonctionnement est établi via une règle réseau placé sur un élément actif.

Il en va de même pour le flux HTTPS utilisant le port 443. Celui-ci est automatiquement redirigé vers un autre port d'écoute du serveur proxy via une règle sur le même élément actif.

Les règles de redirection réseau constituent le seul point de sortie des flux utilisant les ports 80 et 443. Les flux utilisant d'autres ports sont bloqués par d'autres règles réseau.

Squid peut exécuter plusieurs instances d'écoute simultanément. Nous avons choisi de travailler avec une instance dédiée aux flux HTTP et une autre aux flux HTTPS.

Ce choix permet d'éviter les confusions dans l'établissement des règles dans Squid. En effet, l'implémentation de la prise en charge du protocole HTTPS dans Squid a été établie sur des commandes différentes de celles utilisées du protocole HTTP.

Nous faisons correspondre à chaque instance un fichier de log distinct. L'instance gérant HTTP a son propre fichier de log et l'instance gérant HTTPS possède le sien.

L'analyse des logs est grandement simplifié notamment lors de la mise en place.

Les autorisations de connexion sont basées sur le principe de l'établissement d'une liste blanche :

- par défaut toutes les connexions sont refusées sauf pour les éléments figurants dans cette liste ;
- pour le protocole HTTP, la liste blanche est basée sur des urls ou des noms de domaines ;
- pour le protocole HTTPS, la liste blanche est basée sur le nom d'hôte du serveur.

Il est important de partir sur la version 4 de Squid. La version 3.5 qui intègre déjà la fonctionnalité « Peek and Spice » rencontre des problèmes.

## 2.1 La prise en charge du protocole HTTPS

Même si la prise en charge du protocole HTTPS était l'objectif principal, nous ne souhaitions pas effectuer de déchiffrement du flux ni mettre en place un système de certificats auto-signés générés à la volée.

La majorité des implémentations de proxy transparent prenant en charge le HTTPS utilise des attaques de type « Man in the middle ». En résumé simplifié, dans cette configuration, le serveur proxy génère des certificats à la volée et remplace le nom d'hôte attendu dans ceux-ci afin qu'ils soient acceptés lors de l'établissement de la connexion.

Il faut savoir que la génération de certificats à la volée est gourmande en ressources. Si on place un nombre important de machines derrière le serveur proxy, il faudra que celui-ci possède une puissance de calcul et de mémoire suffisante pour gérer convenablement toutes les connexions.

Squid propose une alternative plus propre que cette solution. La fonctionnalité nommée « Peek and Splice » permet de ne pas utilisé de certificats de substitution et de ne pas déchiffrer le flux HTTPS.

Elle utilise le SNI (Server Name Indication) qui est une évolution du protocole TLS (Transport Layer Security).

Lors de l'établissement d'une connexion TLS, préambule d'une connexion HTTPS, l'élément SNI est récupéré. Il correspond au nom d'hôte ou généralement au FQDN (Fully Qualified Domain Name). C'est

pour Squid l'étape du « Peek ». Le SNI est utilisé comme élément de référence pour la vérification de la liste blanche fournie à Squid.

Squid vérifie que le nom d'hôte est présent dans la liste blanche. Si celui-ci n'est pas présent dans celle-ci, la connexion n'est pas autorisée. Si celui-ci est présent, la connexion est autorisée.

Dans ce cas, un tunnel TCP est établi entre le client et le serveur. Le flux est simplement transporté. Aucune modification n'est effectuée sur le flux, aucun déchiffrement n'est effectué.

Pour Squid, cela correspond à l'étape du « Splice ». Une fois la connexion terminée, le tunnel TCP est fermé.

Par cette méthode, l'intégrité du flux est préservée.

## 2.2 Établissement de la liste blanche

La création de la ou des listes blanches demande une courte période durant laquelle on accepte toutes les connexions pour les machines placées derrière le serveur Squid.

Il est recommandé de commencer avec un nombre restreint de machines. La lecture et l'analyse des fichiers de log n'en sera que simplifiée.

En récupérant dans le fichier de log de l'instance chargée du flux HTTP, la valeur du champ nommé « squid_requestURL », on pourra établir la liste blanche des URLs ou des domaines désirés.

Pour le protocole HTTPS, il faudra récupérer le contenu du champ nommé « squid_sni_received ».

L'extraction de ces valeurs via l'utilisation d'une ligne de bash, utilisant habilement une commande awk et une commande grep, pourra grandement vous faciliter la tâche. Le format de log de Squid pouvant être fixé à un format comme JSON, la séparation de chaque champs permet d'en automatiser l'analyse aisément.

La nature de l'élément recueilli dans le cas du protocole HTTPS est un nom de machine.

La liste pourra contenir le nom de machine ou le domaine auquel elle appartient mais ne pourra contenir une URL comme pour le protocole HTTP.

Au bout d'un certain temps d'analyse des logs et du recueil de ces valeurs, il sera possible de créer la liste blanche et de n'autoriser que les éléments de cette liste à accéder à Internet.

Il est possible d'établir des listes distinctes pour chaque instance de Squid.

On peut également faire une liste commune aux deux instances de Squid comme nous en avons fait le choix. Ceci permet de mettre à jour les autorisations à une ressource simultanément pour les deux protocoles.

# 3 L'exploitation de la solution

Une fois la liste blanche initiale mise en place, il faudra s'assurer régulièrement en regardant les logs, que des accès à des ressources ne sont pas refusées par un oubli ou un manque.

Les commandes ou scripts utilisés pour l'élaboration de la liste initiale seront toujours d'actualité pour scruter les logs d'accès et repérer ces cas.

## 3.1 Le cas du code 409

Il peut arriver que certains accès à des ressources ne fonctionnent pas toujours comme attendu bien que la liste blanche soit correcte. La connexion à la ressource s'effectue de manière aléatoire. Dans les logs de Squid, on voit apparaître des codes de retour 409.

Ce type de problèmes survient avec des machines qui changent dynamiquement d'adresse IP. Lors de l'établissement de la connexion, une adresse IP est présentée à Squid qui la mémorise, puis lors de l'acheminement du flux l'adresse IP change. Squid ne fonctionne qu'en utilisant les adresses IP. Si celles-ci changent durant le transfert du flux, celui-ci met fin à la connexion.

Malheureusement, il n'y a pas actuellement de solution à ce problème. Dans ce cas, il faut espérer que par de nouvelles tentatives, Squid puisse effectuer le transfert du flux sans tomber sur un changement d'adresse IP. Ce fonctionnement en l'état est acceptable car les mises à jours finissent par se faire dans un délai raisonnable.

## 3.2 Séparation des fichiers de configuration et de la liste blanche

Il est préférable de ne pas construire et d'éditer la liste blanche dans le fichier de configuration comme il courant de le faire avec Squid. Il vaut mieux faire une inclusion de la liste sous forme d'un fichier texte contenant les noms de domaines ligne par ligne et de la faire correspondre à une règle dans le fichier de configuration.

Ainsi le fichier de configuration qui contient les paramètres indispensables au bon fonctionnement de Squid, sont préservés d'une éventuelle erreur ou corruption involontaire. Il suffit de modifier le fichier correspondant à la ou les listes blanches (On peut inclure plusieurs listes blanches pouvant correspondre à plusieurs règles différentes) sans prendre le risque de casser la configuration.

C'est un moyen possible aussi de faire la délégation de la liste. L'administrateur peut garder la gestion de la configuration et déléguer à d'autres personnes qui ne sont pas forcément informaticiens, la maintenance de cette liste. Cette solution a été retenue dans notre projet.

# 4 Bilan et perspectives

La mise en place d'un serveur proxy sortant transparent gérant le protocole HTTPS sans déchiffrement est relativement simple.  La partie matérielle se fait via une redirection du port 80 et 443 via un élément actif du réseau.

L'élaboration de la liste blanche se fait via le passage progressif des clients derrière le serveur proxy.

Ceci se fait d'abord par une courte période d'ouverture quasi-totale du client.

Après récupération des SNI dans les fichiers de logs, la liste blanche est établie et fixe l'ouverture du client vers INTERNET.

Il est possible d'établir les listes blanches pour une ou plusieurs machines. La granularité peut se faire sur la population des machines ou le niveau d'ouverture vers INTERNET souhaité.

L'historisation des fichiers de configuration et de la liste blanche avec des outils comme « Git » et « GitLab » est très recommandé pour l'exploitation de la solution

La gestion de plusieurs centaines de machines sur un seul serveur proxy ne pose pas de difficulté.

Il est possible d'effectuer la mise en place d'une infrastructure doublée via une solution comme « HAProxy ». Ainsi en cas de défaillance, les flux normalement destinés au premier serveur proxy sont automatiquement redirigés vers un deuxième serveur proxy qui se chargera du traitement des requêtes.