Rapport du projet de RSA : MyAdBlock

Rémi Morel
Pierre Maeckereel

April 2017

Table des matières

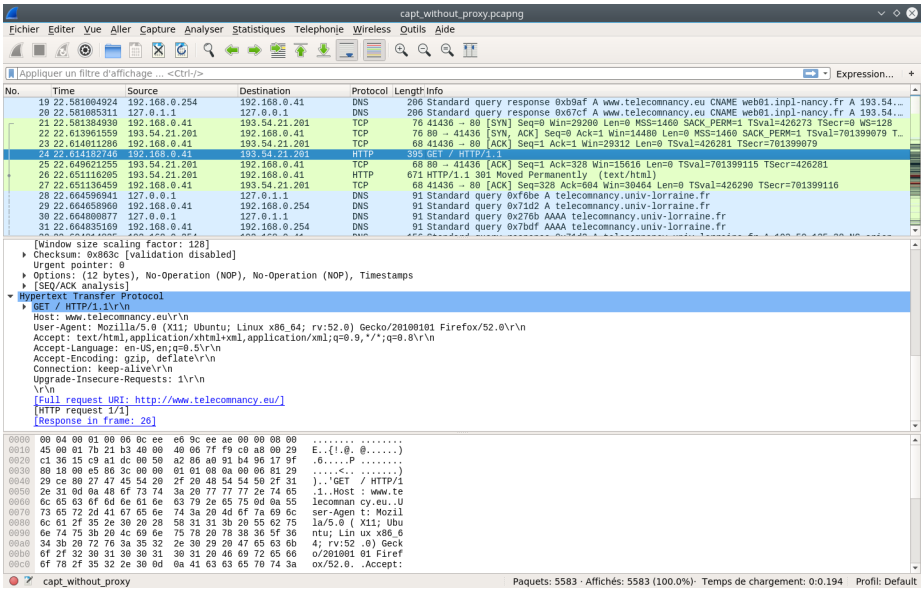
1 Présentation du projet

Le but de ce projet est de réaliser un proxy http qui bloque les publicités, un peu comme un adBlock d'où son nom. Le travail a été organisé en plusieurs étapes qui seront présentées ci-dessous.

2 Le projet étape par étape

2.1 Etape 1 : Analye de traces avec wireshark

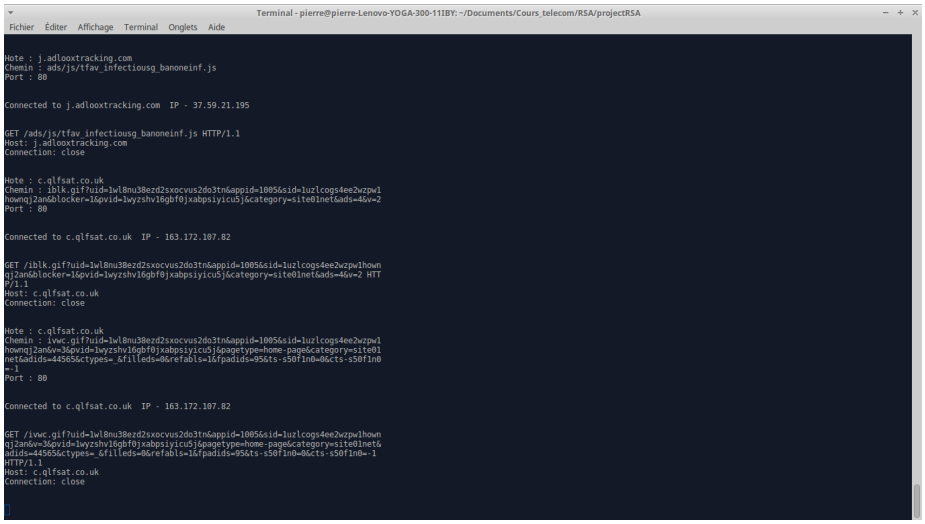
Quand on charge une page web, le client envoie au serveur une trame TCP de synchronisation.
Ensuite le serveur renvoie une trame d'acquittement au client et le client répond par un acquittement, toujours en TCP.
Le client envoie ensuite une trame HTTP avec l'URL que l'on souhaite atteindre.
Le serveur renvoie un acquittement TCP et une trame HTTP qui indique que l'adresse demandée redirige vers une autre.
Enfin le client renvoie un acquittement TCP.



2.2 Etape 2 : Configuration du navigateur web pour utiliser le proxy

Comme expliqué dans le fichier *readme.md* joint au projet, pour que le proxy fonctionne avec le navigateur web, il faut préalablement configurer celui-ci. Sur firefox par exemple, aller dans **Préférences -> Avancé -> onglet Réseau -> Paramètres** Là rentrer la configuration manuelle du proxy HTTP : 127.0.0.1

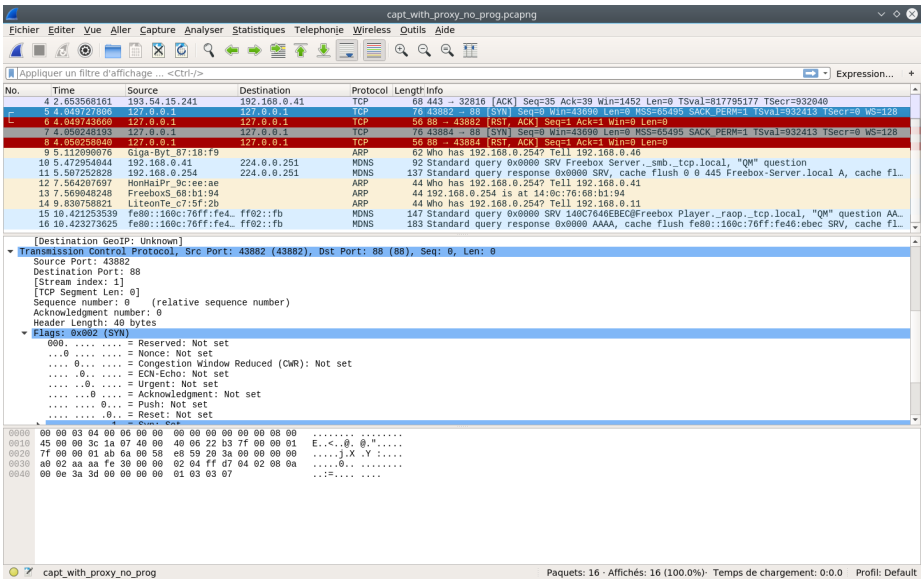
pour l'adresse , et comme numéro de port celui rentré en argument du projet quand il est lancé en ligne de commande. Une fois les changements appliqués, le proxy est activé et chaque requête HTTP (chargement d'une page web) passera par le proxy. On peut d'ailleurs observer dans la console dans laquelle on a lancé le proxy un historique des log, c'est-à-dire tous les sites visités depuis que le proxy a été activé, avec les éventuelles redirections.



2.3 Etape 3 : Analyse de traces avec wireshark

Le client envoie une trame TCP de synchronisation au serveur local.
Le serveur renvoie un acquittement TCP RST demandant au client de renvoyer une trame de synchronisation.
Le client renvoie sa trame, le serveur re-répond avec un reset.
Le client essaye d'envoyer une trame à un proxy mais celui-ci n'est pas activé.

Le reset est un message d'erreur renvoyé automatiquement.
Le proxy rajoute un intermédiaire entre le client et le serveur, il relaie les trames envoyées du client vers le proxy.



2.4 Etape 4 : algorithme général de My AdBlock

1. on ouvre une socket de connexion entre le client et le proxy
2. on attend une requête du client
3. quand il en reçoit une, il regarde le protocole utilisé
4. si c'est HTTP, on regarde le nom de l'hôte
5. on récupère les adresses IPV4 et IPV6 avec getaddr-info()
6. on ouvre une connexion entre le proxy et le serveur
7. on envoie la requête du client au serveur
8. le serveur renvoie des infos
9. on filtre selon la easylist le PATH
10. on renvoie au client les données ce qui a passé le filtre

2.5 Etape 5 : Implémentation du proxy en IPV4 puis IPV6

Nous avons donc implémenté notre application en suivant l'algorithme dans l'étape 4. Afin de réaliser une application durable, nous avons mis en place la gestion des adresses IPv4 et IPv6, ce dernier protocole étant destiné à se développer de plus en plus dans un futur proche.

Ainsi, nous voulions récupérer à la fois l'adresse IPv4 et IPv6 d'un hôte à partif de son nom. Nous avons donc utilisé la fonction getaddrinfo, et ainsi récupéré toutes les adresses relatives à un nom d'hôte, qu'elles soient en IPv4 ou 6. Nous tentons ensuite une connexion sur chaque adresse IP jusqu'à réussir cette connexion avec le serveur de destination.

2.6 Etape 6 : Gestion du multi-clients

Pour cette étape plusieurs solutions s'offraient à nous. Nous avons choisi de la réaliser en utilisant des fork, où chaque processus fork est démarré à la suite de la demande de connexion d'un nouveau client au AdBlock.

2.7 Etape 7 : Gestion des requêtes HTTPS

Cette étape n'a pas été réalisée malgré une réflexion sur sa conception, et des tentatives de réalisation.

La première tentative fut d'utiliser la librairie SSL afin de créer un canal sécurisé afin d'échanger les données entre client, proxy et serveur. La deuxième tentative fut de gérer de les requêtes HTTPS de la même manière que les requêtes HTTP.

3 Choix retenus pour la conception

Plusieurs choix se sont offerts à nous pour la conception de ce proxy bloqueur de pub.

— pour la gestion du multi-client, on avait le choix entre le select et le fork. Nous avons choisi d'utiliser le fork car c'est une méthode que nous maîtrisons mieux, de par nos cours de système du module RS notamment

4 Fonctionnalités de MyAdBlock

Notre projet est avant tout un proxy : il intercepte les requêtes HTTP envoyées d'un client (un navigateur web) à un serveur distant. Il renvoie ensuite la requête au serveur après l'avoir analysée. Sachant que le proxy my AdBlock bloque les publicités, s'il reconnaît une URL présente dans la blacklist *easylist.txt* il ne renvoie pas la requête vers le serveur. Le proxy gère les requêtes au format IPV4 et IPV6 : il sait faire la différence et adapte son comportement en fonction. Le proxy gère le multi-client, c'est-à-dire que plusieurs navigateurs peuvent utiliser ce proxy en même temps.

5 Difficultés rencontrées

Plusieurs difficultés on été rencontrées pendant la réalisation de ce projet :

— réalisation de la 7e étape : gestion du protocole HTTPS

— affichage d'images de taille importante : pendant les tests sur japscan, nous avons remarqué que les images de taille importante étaient bloquées par le proxy. Pour résoudre le problème nous avons augmenté la taille du buffer de stockage.

6 Tests réalisés

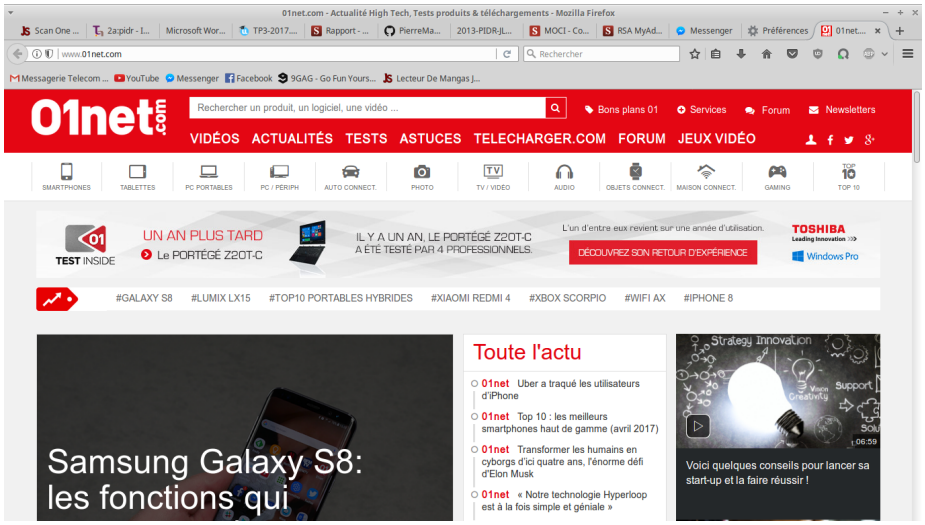
Pour vérifier que le projet répondait aux attentes nous avons effectué des tests, notamment sur les sites suivants :

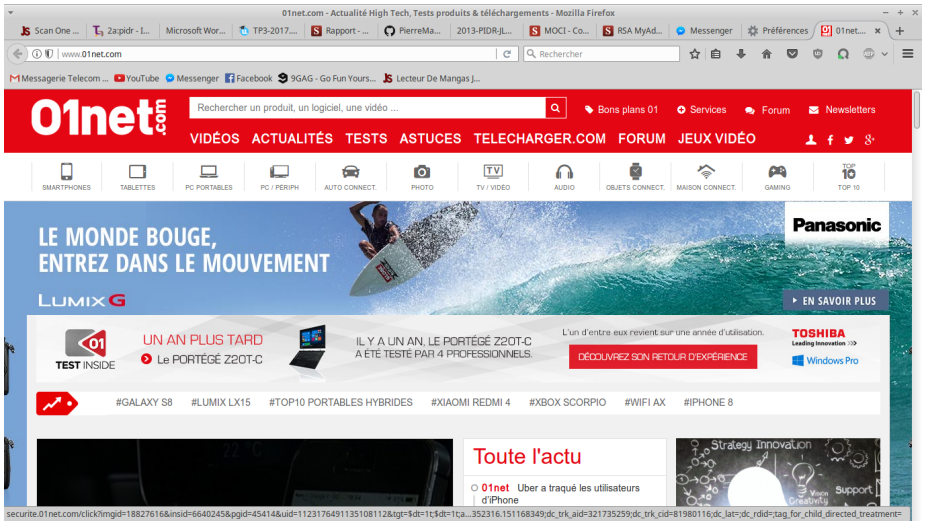
— www.01net.com : site de test recommandé dans le sujet

— http://www.japscan.com : site internet permettant de lire des chapitres de mangas. L'utilité de ce site est qu'il comporte beaucoup de publicité, mais aussi des contenus important comme les pages des chapitres à lire. Nous avons ainsi pu vérifier le chargement complet des informations nécessaires pour les utilisateurs.

— https://www.leboncoin.com/annonces/ : site de test recommandé dans le sujet pour le protocole HTTPS.

Ci-dessous un comparatif avec/sans le proxy.





7 Pistes d'amélioration

Pour améliorer notre proxy, on pourrait commencer par gérer les requêtes HTTPS. En effet une grande majorité des sites webs disponibles sur internet utilisent ce protocole, surtout pour ceux les plus utilisés. Cela rendrait le proxy beaucoup plus utile et polyvalent.

8 Conclusion

Ce projet fut intéressant à réaliser, car il nous a permis de mettre en pratique les notions vues en cours de RSA partie réseaux, comme les interactions client-serveur au travers d'un protocole spécifique (ici HTTPP). Il n'était pas spécialement difficile, surtout avec la multitude d'exemples présents sur le net.

9 Annexes

— Un fichier *Readme.md* qui présente le projet et comment l'utiliser
— Un fichier *Makefile* qui permet de compiler le projet rapidement