Rapport du projet MyAdblock

Adam FACI
Quentin DECHAUX

25 Avril 2017

Cours : Réseaux et Systèmes Avancés : Partie Réseaux
Professeur : Isabelle CHRISMENT

Table des matières

1 Introduction

Projet effectué dans le cadre du cours de RSA visant à comprendre la notion de serveur proxy HTTP en créant le nôtre et en l'utilisant en tant que **AdBlock** ie. en tant que bloqueur de publicités.

Le serveur devra à chaque requête arrivant sur notre machine vérifier que l'hôte de celle-ci ne fait pas partie d'une liste prédéfinie de serveurs de publicités.

À priori la liste proviendra du site `https://easylist.to/`.

2 Analyse des échanges TCP et HTTP entre un client et un serveur

2.1 Analyse des échanges sans proxy

Pour cette analyse, nous nous sommes juste connectés au site `www.telecomnancy.net`. Afin de repérer la conversation sur wireshark, nous avons d'abord filtré avec **http** afin de repérer plus facilement dans la colonne info un *GET / HTTP/1.1*. Une fois trouvé, nous avons isolé les échanges via un filtre plus précis.

Les échanges TCP et HTTP entre le client (notre machine) et le serveur web (le site `www.telecomnancy.eu`) ont ainsi été triés via le filtre : **tcp && ip.addr == 193.54.21.201** où **193.54.21.201** correspond à l'adresse ip du site au moment de la communication et la condition **ip.addr == 193.54.21.201** exige que l'adresse soit présente dans la trace (en source ou destinataire). Aussi, la condition **tcp** impose qu'un en-tête TCP soit présent, ce qui est le cas pour les traces TCP et HTTP.

Le résultat de ces échanges sont en pièce jointe dans le fichier « *capture_file_question_1.pcapng* ».

Cela commence tout d'abord avec un *TCP three-way handshake* avec le serveur qui reçoit le segment SYN, nous qui recevons le segment SYN+ACK et enfin le serveur qui reçoit le segment ACK.

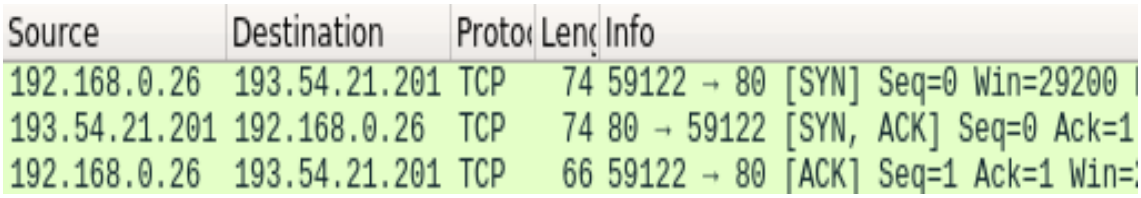

FIGURE 1 – Le three-way handshake

Cela correspond à l'établissement de la connexion entre le client et le serveur.

Ensuite, un paquet HTTP est envoyé au serveur avec pour en-tête HTTP *GET / HTTP/1.1*.

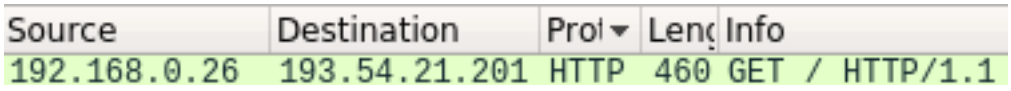

FIGURE 2 – Requête HTTP au serveur

Il correspond à la demande du contenu de la page d'accueil du site web par le client. Il contient les données TCP avec l'en-tête HTTP au dessus.

Ensuite le client reçoit deux paquets : un paquet TCP et un paquet HTTP.

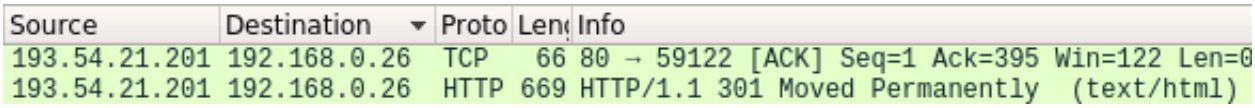

FIGURE 3 – La réponse du serveur

Le paquet TCP correspond à l'acquitement de la requête HTTP précédente. Le paquet HTTP correspond quant à lui à la réponse à la requête HTTP du client. Elle spécifie une redirection avec la mention *Moved Permanently* accompagné d'un contenu html.

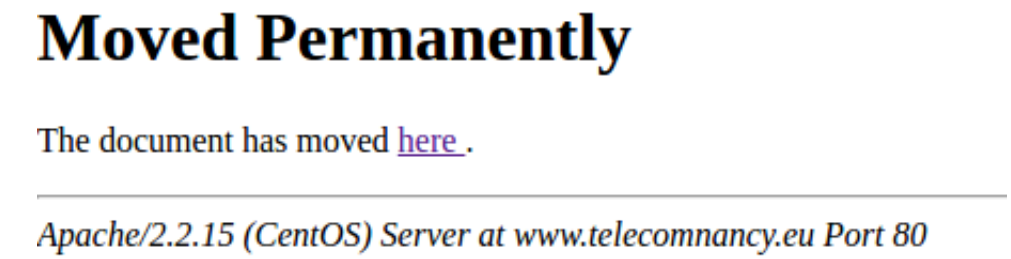

FIGURE 4 – Le contenu html de la réponse

Cela est dû au fait que le site de l'école est en désormais hébergé à l'adresse `www.telecomnancy.univ-lorraine.fr`. En réalité, lors de la navigation sur le site web, rien n'est affiché et la redirection est automatique.

Puis un acquittement de la réponse est envoyé par le client au serveur via un paquet TCP.

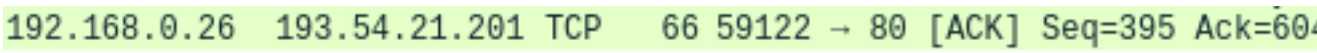

FIGURE 5 – L'acquitement de la réponse

À noter que selon les essais, et selon que nous allions sur différentes page du site ou nous, certains échanges supplémentaires avec d'autres serveurs comprenant des fichiers supplémentaires nécessaires comme des feuilles de style en cascade (.css) , des établissement de connexions (*TCP three-way handshake*), ou des requêtes jQuery (pour un affichage rapide et dynamique des éléments de la page web).

2.2 Mise en place du proxy

Nous devons ensuite effectuer la même analyse avec l'utilisation d'un serveur proxy.

Pour cela, nous avons cherché un proxy gratuit, et sommes tombé sur un proxy qui nous faisait passer par l'Allemagne afin que le site demadé nous localise là-bas. Voulant travailler sur le plus simple possible, nous avons encore cherché et avons finalement opté pour le proxy proposé sur le site `www.simpleproxy.net`.

Diverses options étaient disponibles comme la possiblité de crypter l'URL demandée ou de refuser les scritps (ajax par exemple), mais encore une fois nous avons tous décoché. Les résultats obtenus sont donnés dans la section suivante.

2.3 Analyse des échanges avec proxy

Lors de cette analyse, nous sommes donc allés sur le site du proxy, puis nous avons une fois de plus demandé à atteindre le site `www.telecomnancy.eu` avant de couper la connexion.

Nous avons cherché la conversation de la même façon que précédemment et sommes très vite tombé sur la requête HTTP ayant pour info *GET \browse.php ?u=http%3A%2F%2Ftelecomnancy.univ-lorraine.fr&b=4*. Afin d'isoler la conversation sur wireshark, nous avons alors appliqué le filtre **tcp.port == 51276** car le port **51276** correspondait à tous les paquets de la conversation. De plus ce filtre implique également de n'avoir que des paquets comprenant un en-tête TCP. Aussi, un port restant exclusif à une seule communication, cela nous assure également que les traces y venant et en sortant dont bien avec le même interlocuteur.

Au début de la conversation, on constate de la même façon un *three way handshake* correspondant à l'établissement de la connexion au site via le proxy.

Source	Destination	Proto	Leng	Info
192.168.1.22	104.207.154.152	TCP	74	51276 → 80 [SYN] Seq=0 Win=29200
104.207.154.152	192.168.1.22	TCP	74	80 → 51276 [SYN, ACK] Seq=0 Ack=1
192.168.1.22	104.207.154.152	TCP	66	51276 → 80 [ACK] Seq=1 Ack=1 Win=

FIGURE 6 – Le three way handshake

Puis il y a une première requête HTTP du site qui renvoie un paquet malformé. Puis un second essai avec cette fois une série de dizaines de paquets échangés avec des données arrivant et un acquittement en réponse.

Par la suite, contrairement à la situation sans proxy, il y a un

FIGURE 7 – Première requête HTTP

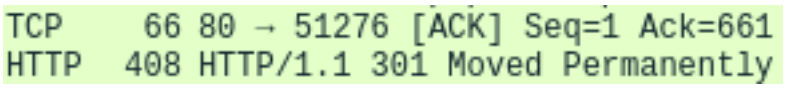

FIGURE 8 – Réponse du serveur

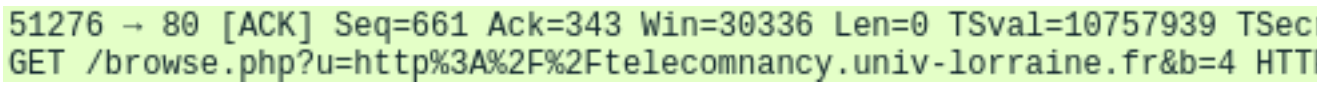

FIGURE 9 – Deuxième requête HTTP

échange d'une série de paquets TCP entre le client et le serveur, correspondant aux données de la page à charger. Cela doit être dû à la mauvaise connexion à cause du passage par le proxy.

Enfin lorsque nous quittons le site, contrairement à la première fois, le proxy prend le temps d'effectuer une fermeture de connexion via l'établissement d'un échange de paquets TCP ayant les flags *FIN* activés.

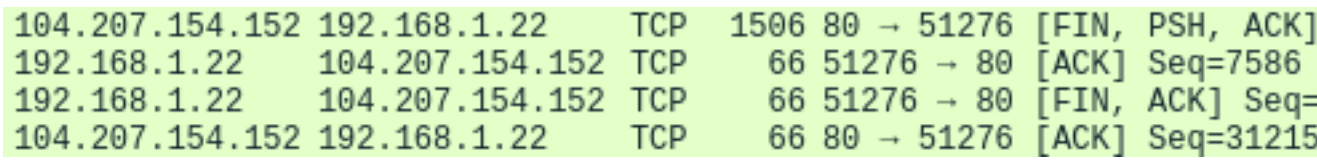

FIGURE 10 – Fin de connexion

3 Notre proxy

3.1 Spécification de l'algorithme

Nous allons détailler ici une première approche de l'algorithme de notre serveur proxy en pseudo-code.

```
Données : RequêteClient, RéponseServeur, SocketClient,
          SocketServeur, AdresseServeur
Résultats : Laisse passer la RéponseServeur au client ou lui renvoie
            une erreur
tant que Serveur Proxy Lancé faire
    //ie tant que le serveur proxy est en ligne.
    Attente de l'arrivée d'une requête ;
    //ie d'une requête par un client.
    si Requête arrivée alors
        //Si une requête est réceptionnée au niveau du socket
        RequêteClient = RécupérerRequête(SocketClient) ;
        AdresseServeur = RequêteToAdresse(RequêteClient) ;
        //Récupère l'adresse demandée dans la requête
        si not AdresseOK(AdresseServeur) alors
            //On teste si l'adresse est interdite ou pas.
            AfficherDansLaConsole(« L'adresse %s est bloquée par le
             proxy. »,AdresseServeur) ;
        finsi
        alors
            SocketServeur = CréerSocket(AdresseServeur) ;
            EnvoyerRequête(RequêteClient,SocketServeur) ;
            RéponseServeurTemp = 0 ;
            //Variable locale qui sockera les paquets reçus au fur et à
             mesure.
            tant que not RéponseServeur faire
                //Tant que la Réponse du serveur n'est pas définie.
                RéponseServeurTemp +=
                 RécupérerRéponse(SocketServeur) ;
                //On récupère le paquet envoyé par le serveur.
                TailleDonnéesRestantes =
                 RéponseServeurTemp.TailleDonnéesRestantes ;
                //On récupère dans les flags la taille des données
                 restantes.
                si TailleDonnéesRestantes == 0 alors
                    //ie le dernier paquet a été reçu.
                    RéponseServeur = RéponseServeurTemp ;
                    Fermer(SocketServeur) ;
                finsi
            fintq
            RetourneRéponse(SocketClient) ;
        finsi
    finsi
fintq
```


Algorithme 1 : Algorithme du proxy