Université du Québec à Chicoutimi

RAPPORT DE PROJET

Automne 2017

Sécurité des réseaux et du Web (8TRD157)

Local : P1-5050

Département d'informatique et de Mathématiques (DIM)

Chargé de cours ***Valère Plantevin***
P3-5030
418-545-5011 (Poste 2355)
Valère_plantevin@uqac.ca

Auteurs ***Pierre-Marc Laforest***
Michael Lizotte-Gagnon
Marc-Antoine Tremblay

Introduction

De nos jours, il est presque impossible de penser passer notre quotidien sans aucune connection Internet. En effet, en l'espace de quelques années, Internet est devenu aussi indispensable que l'électricité ou encore l'eau potable. C'est un fait, internet fait partie de nos vie et nous en sommes dépendents. Mais quand est-il, comment fonctionne-t-il réellement ?

Une des beautés de l'utilisation d'Internet et des réseaux d'ordinateurs en général réside dans toute la complexité cachée. La plupart des utilisateurs l'ignore mais même s'il peut être aisé de naviguer sur internet, de consulter des pages et d'obtenir du contenu en ligne, il en est tout autrement de comprendre les détails internes de ces échanges d'informations. Les accès à Internet, autrement dit l'échange d'information entre différents ordinateurs, implique une multitude de protocoles réseaux, et un nombre incroyable d'échanges de messages et d'accusés réception à travers ces différents protocoles.

Le travail présenté dans ce document consiste en un « sniffeur de paquets ». Un « sniffeur de paquets » est, en fait, un logiciel qui identifie les paquets entrants dans un ordinateur. Les paquets sont réceptionnés sur la carte réseau et sont inspectés par ce logiciel, qui leur attribue une identité. Ce travail consiste en trois parties, une introduction, un développement qui explique le projet en détails ainsi qu'une conclusion.

Corps du travail

Présentation générale

Tel que mentionné dans l'introduction, le travail consiste en un sniffeur de paquets, i.e un logiciel qui inspecte les paquets entrants/sortants d'un ordinateur vers le réseau. Plutôt que de partir de zéro, les développeurs ont plutôt opté d'utiliser un code trouvé sur internet comme point de départ[1].

L'annexe A présente le fonctionnement général du programme. En fait le programme est un peu comme un gros « switch case » contenant les différents types de protocoles. Il peut aussi y avoir des « switch cases » à plusieurs niveaux. Si par exemple un en-tête de type ip est trouvée, un autre switch case sera nécessaire pour les protocoles une couche plus haut dans la hiérarchie (tcp, udp, etc.). Si un paquet est identifié comme utilisant un protocole ne particulier, celui-ci est enregistré dans le fichier. Finalement, si le paquet n'utilise aucun protocole connu, aucune action n'est effectuée et le programme recommence a lire des paquets avec le raw socket.

Version 1.0

La première version utilisée par les développeurs était une simple implémentation de sniffeur de paquets avec l'API winsock. Le code en question utilise les « raw sockets » pour décoder quatre types de paquets différents : ip, tcp, udp et icmp. Même si les fonctionnalités de ce programme sont assez limitées, il s'avère qu'il constitue tout de même un bon point de départ pour créer un logiciel un peu plus évolué. L'exécution normale du programme se déroule comme suit : une fois le programme lancé, le sniffeur

[1] http://www.binarytides.com/packet-sniffer-code-in-c-using-winsock/

compte le nombre de paquets de chaque type qui arrive sur la carte réseau. Il enregistre chacun de ces paquets dans un fichier prévu à cet effet.

Version 1.1

La deuxième version du sniffeur de paquets de l'équipe permettait la détection de paquets Ethernet. Le code a été tiré d'un deuxième tutoriel[2]. La détection des paquets Ethernet nécessitait une modification importante : l'installation d'un nouveau pilote de carte réseau. Sans celui-ci, il était impossible d'identifier les paquets ethernet. En effet, le logiciel a succès wireshark utilise aussi le pilote pour détecter ce genre de paquets.

Amélioration 1 : ajout d'un filtre pour paquets

Dans tous sniffeurs digne de respect, il y a possibilité d'ajouter des filtres pour ne laisser passer que certains types de paquets. L'équipe a réussi à ajouter ce genre de filtres à l'aide des fonctions intégrées du pilote winpcap. Lors de l'exécution normale du programme, après avoir choisi une interface sur laquelle écouter, l'utilisateur est interrogé pour savoir s'il désire ajouter un filtre sur les paquets reçus. Par exemple, pour ne récupérer que les paquets tcp, l'utilisateur n'a qu'à entrer tcp lorsque demandé.

Amélioration 2 : Reconnaissance des paquets ARP

Arp est un protocole de première importance qui permet aux appareils sur un réseau de découvrir l'adresse physique d'autres appareils sur le réseau en connaissant leur adresse ip. Puisque le projet utilise maintenant winpcap et reconnaît les paquets ethernet, il était maintenant possible de détecter les paquets ARP. Pour y arriver, il suffit d'examiner le champ type de l'en-tête ethernet pour y trouver la valeur 0x0806. En effet, les paquets arp ne sont pas considérés comme des paquets IP, ils ont plutôt un type distinct. Une fois cela fait, sachant le format de l'en-tête arp, il est possible d'enregistrer les informations contenu dans l'en-tête dans le fichier de sortie.

Amélioration 3 : Reconnaissance des paquets ipv6

Ipv6 est un nouveau protocole qui vise à pallier le nombre d'adresses limitées d'ipv4. Les concepteurs d'ipv4 ne pouvaient pas se douter qu'ils allaient un jour manquer d'adresses disponibles, mais malheureusement cela s'est produit. C'est donc pourquoi les adresses ipv6 ont fait leur apparition. Un peu comme le protocole arp, la détection d'ipv6 nécessite d'inspecter le champ type de l'en-tête Ethernet. pour trouver la valeur 0x86dd. Par la suite l'en-tête ipv6 est décortiqué pour ensuite être enregistré dans le fichier de sortie.

Cependant, dans le travail la tâche de reconnaissance de ce type de paquets n'a pas pu être complétée. Un bug est présent dans la séparation de l'en-tête en ses différentes composantes. L'équipe de développeurs a une bonne idée sur la nature de ce bug : la structure C qui contient l'en-tête. En effet, les compilateurs C font une opération de padding pour aligner les différents membres d'une structure. Cela a comme conséquence d'accélérer les opérations de lecture/écriture en mémoire. Cependant, dans le cas qui nous concerne cela a aussi comme effet indésirable d'introduire des zéros dans certains champs de l'en-tête

[2] http://www.binarytides.com/code-packet-sniffer-c-winpcap/

ipv6. Puisqu'un de ces champs ne contient que 20 bits, l'équipe croit que le compilateur annexe 4 zéros à la fin de ce champ ce qui produit un décalage dans les valeurs des champs suivants.

Amélioration 4 : Reconnaissance des paquets http

Http est un protocole de niveau applicatif nécessaire pour l'échange de message avec les serveurs web. En fait, la totalité des pages consultées sur internet proviennent d'une communication entre un serveur et un ordinateur via le protocole http, ou son pendant sécurisé, https.

Certains membres de l'équipe croient avoir intégré ce protocole au travail mais il n'en est rien. L'exemple de la figure suivante le démontre très bien. La fonction validate_filter_substr a comme but de valider les mots clés de l'expression pour ajouter un filtre entré par l'usager. Ceci est fait dans le but de s'assurer que l'expression entrée est conforme au format exigé par le programme. Un tableau de mot clés est passé à la fonction puis chaque membre de l'expression est passé tour à tour à cette fonction accompagnée d'un tableau de mots clés acceptés par le programme. Le développeur en question à plutôt choisi de coder en dur le protocole http dans cette fonction qui était au départ, globale.

La reconnaissance des paquets http est donc un échec, et doit être recommencée.

```
// compare the filter expression against an array passed as an argument.
int validate_filter_substr(char *filter_word, char *filter_exp_to_comp[], int nbr_str)
{
    int word_nbr = 0;

    while (word_nbr < nbr_str) {

        if (strncmp(filter_word, "http", 4) == 0) // <-----  Why the F*** did you do that ????
        {
            return 1;
        }
        else if ((strncmp(filter_word, filter_exp_to_comp[word_nbr], strlen(filter_exp_to_comp[word_nbr]))) == 0)
        {
            return 0;
        }
        word_nbr++;
    }
    return -1;
}
```

Amélioration 5 : changement de l'interface utilisateur

La détection de nouveaux paquets a obligé l'équipe de développeur à revoir leur stratégie concernant l'affichage. L'interface précédent utilisait un retour à la ligne (/r) pour écraser la ligne en cours et ainsi incrémenter les compteurs de paquets en temps réel. L'ajout de nouveaux paquets à augmenter le nombre de caractères à afficher si bien que l'affichage dépassait maintenant une ligne.

Pour remédier à la situation, l'équipe de développeurs a utilisé l'api console de Windows. L'api console a permis à l'équipe de développer une fonction qui repositionne le curseur un nombre x de lignes plus haut. Le résultat net est présenté à la figure 2, ou nous pouvons voir les paquets reçus s'incrémenter sur plusieurs lignes en temps réel.

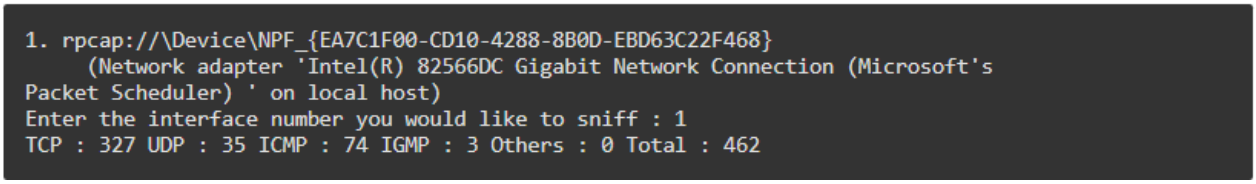

Figure 1. Interface du sniffeur de paquet

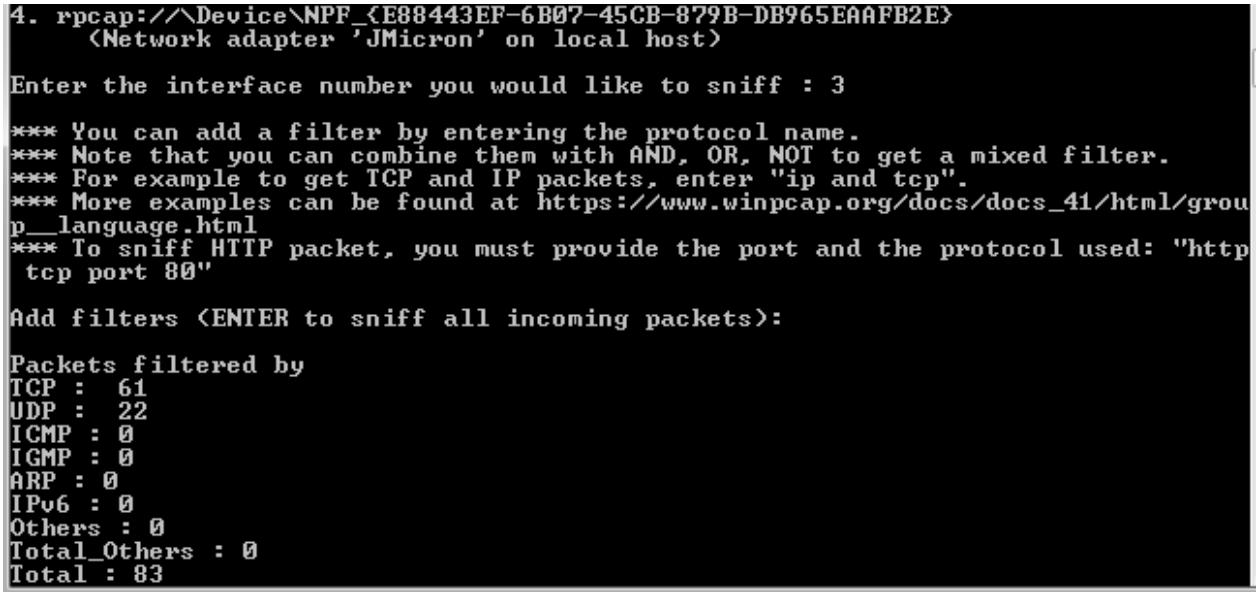

Figure 2. Interface améliorée du sniffeur de paquets

Conclusion

En conclusion, ce travail a permis de faire évoluer un simple « sniffeur de paquets » vers un logiciel un peu plus sophistiqué. Bien que le projet n'ait pas permis de détecter tous les types de protocoles, l'équipe est satisfaite du travail rendu puisqu'elle a réussi à travailler dans l'environnement de quelqu'un d'autre sans avoir à réécrire tous le code. Elle a pu tirer profit du travail d'un autre individu et a su bâtir un logiciel un peu meilleur que le précédent.

Annexe A – Fonctionnement général du programme