The **Polytechnician Onion Routing Circuitry**

Michel Blancard
Nathan Skrzypczak

Projet de modal d'informatique - juin 2013

I - Introduction

Notre objectif initial était de concevoir un système client/serveur qui permette d'anonymiser et de crypter des connexions TCP. L'utilisateur final se connecte sur un serveur local (le client PORC), qui agit comme un proxy pour acheminer les flux TCP d'une manière anonymisée et cryptée vers la cible, en passant pas un réseau de relais (relais PORC).

Manquant de temps pour présenter un programme stable répondant à tous ces critères, nous avons abandonné l'objectif d'anonymisation. Notre but est maintenant de permettre le cryptage d'une connexion entre deux points distincs (un serveur côté client et un serveur côté destination), la liaison entre les deux étant considérée comme peu fiable.

Pour cela, le serveur côté client implémente le protocole SOCKSv4, qui a l'avantage d'être très facile de mise en oeuvre et compatible avec beaucoup de logiciels, tels que Mozilla Firefox. La liaison est cryptée par le protocole TLS, et le serveur côté destination se contente de retransmettre les connexions qui lui arrivent.

II - Manuel d'utilisation

La compilation du code nécessite la bibliothèque gnutls, version >= 3.1.9. Des instructions sont disponibles à l'adresse *http://www.gnutls.org/manual/gnutls.html#Downloading-and-installing*
Attention : les dernières versions (>=3.2) ne sont pas encore compatibles avec la distribution linux Ubuntu.

Puis, exécuter la commande '*make*' dans le repertoire porc.

Lancer le serveur :
./serveur.out

Lancer le serveur côté client :
./client.out

Il ne reste plus qu'à paramétrer le logiciel dont on veut sécuriser les connexions, par exemple

Mozilla Firefox, pour utiliser le proxy SOCKSv4 à l'adresse 127.0.0.1:5555 par défaut.

Il est possible de changer cette configuration en éditant le fichier config.h et en recompilant.

A des fins de tests, le client sclient.out permet de faire une requette HTTP vers un serveur dont l'adresse est indiquée dans le fichier config.h .

Il est possible que le programme ne marche pas, si les ports configurés par défaut sont déjà utilisés. Dans ce cas, éditer le fichier config.h et recompiler.

III - Spécification détaillées

L'implémentation du protocole SOCKSv4 ne tiens pas compte du nom d'utilisateur donné. L'utilisation du programme est transparente pour le serveur de destination.

IV - Fonctionnement du programme

Le proxy SOCKS crée un thread par connexion établie avec un client. Ce thread engage une connexion sécurisée avec le relai, lui indique l'adresse de destination puis attend le flux entrant ou sortant avec la fonction select().

Le relai engage les connexions sécurisées avec le proxy SOCKS, garde en mémoire les descripteurs de flux, et attend les flux avec select().

V - Bibliothèques utilisées

Nous utilisons la librairie gnutls pour la gestion des connexions TLS, openssl pour la gestion des certificats utilisés pour garantir l'authenticité des serveurs, et nous nous sommes inspirés d'un example de proxy SOCKSv5 écrit en C++ disponible à l'adresse *http://average-coder.blogspot.fr/2011/09/simple-socks5-server-in-c.html*