

# TD Sécurité Réseau: Scanner de ports

Elie Bursztein  
elie@bursztein.net

2007

## Table des matières

<b>1 Scanner de port</b>	<b>1</b>
1.1 But	1
1.2 Historique	2
1.3 Principe de fonctionnement	2
<b>2 Instructions</b>	<b>3</b>
<b>3 Vanilla scan</b>	<b>3</b>
<b>4 Half open scan</b>	<b>3</b>
<b>5 Bring it on</b>	<b>3</b>

## Introduction

Ce TD a pour objectif de vous faire mieux comprendre la programmation réseau orientée sécurité. Il est l'occasion de découvrir les bibliothèques open source pcap (capture de paquet) et libnet (envoi de paquet). Il est aussi l'occasion de voir en détail comment un scanner de port fonctionne.

## 1 Scanner de port

Cette section vous rappelle brièvement ce qu'est un scanner de port.

### 1.1 But

L'objectif d'un scanner de ports est d'indiquer pour un ensemble de machines donné quels sont les services qui répondent. Il existe une correspondance standard entre le numéro de port et le service qui écoute. On retrouve une liste de cette association dans `/etc/services` par exemple. Cependant, cette association

n'est pas forcément respectée, en effet rien n'empêche de faire tourner un serveur apache sur le port 25 (port standard du smtp) mais cela aurait tendance à perturber les utilisateurs et rendre le comportement par défaut des navigateurs inopérants. En effet, ceux-ci implicitement assume que le serveur web est sur le port standard 80.

## 1.2 Historique

Les premiers scanner de ports utilisaient simplement la fonction **connect()** de la librairie C standard. C'est en 1997 que Fyodor via *nmap* et un papier dans phrack nommé "*The Art of Port Scanning*", popularise le *Half-open scan* connu aussi sous le nom de *stealth scan*.

Depuis Nmap a subi de nombreuses évolutions mais le half open scan reste toujours la méthode de référence en raison de sa rapidité et du fait qu'elle n'utilise pas de configuration exotique de flags.

## 1.3 Principe de fonctionnement

**Scan Vanilla.** Le scanner de port va utiliser les deux techniques de balayage les plus connues.

Le TCP connect() scanning aka Vanilla scan : c'est la méthode de base. Elle se fait via l'appel à la fonction système **connect()** qui est généralement utilisé pour ouvrir une connexion.

Si le port est ouvert alors la fonction connect() retourne un descripteur valide, autrement l'appel échoue. Le grand avantage de cette technique c'est qu'elle n'a besoin d'aucun privilège spécial pour être exécuté à l'opposé du half open scan. Ainsi n'importe quel utilisateur pourra utiliser votre scanner. Pour des raisons de rapidité, il est recommandé d'utiliser plusieurs sockets en parallèle avec un système I/O non-bloquant qui a un faible time-out. Le gros désavantage d'une telle méthode est que ce genre de scan est facilement détectable. et filtrable.

**Half-open scan.** Cette technique est connue sous le nom de half-open scan car le principe de cette technique est de ne pas terminer le *handshake* TCP. Un packet avec le flag *SYN* est envoyé sur le port à tester. Le service teste répond par un packet avec les flags *SYN+ACK* si le port est ouvert ou avec un packet avec le flag *RST*. Si le paquet retourné est un *SYN+ACK* alors le comportement normal à l'instar du vanilla scan serait de finir la poignée de main en envoyant un paquet contenant un *ACK*. Cependant, ici le scanner ferme la connexion en envoyant un paquet contenant un *RST*. Ainsi la connexion est close avant même d'avoir été validée, d'où le nom d'*half open scan*. Comme ce comportement n'est pas standard, réaliser un scanner qui utilise cette technique demande de forger ses propres packets et d'écouter directement le réseau. C'est pourquoi, cette méthode n'est utilisable qu'avec des privilèges spéciaux. (généralement root)

## 2 Instructions

Veillez traiter les parties dans l'ordre. Ce n'est de toute façon pas possible autrement. Le TD est à faire par groupe de 4 maximum. Le langage demandé est le C. Le but ultime étant d'avoir un scanner de port le plus rapide possible. Tout les méthodes de parallélisation sont autorisées : mode non bloquant, fork, threads. Le scanner doit tourner sous Linux.

## 3 Vanilla scan

La première partie vous demande d'implémenter la méthode de scanning de type vanilla. Veuillez respecter les contraintes suivantes :

- Le scanner accepte via l'option -p une liste de ports à scanner. Par exemple 1-100 ou encore 1-10, 50-70.
- Le scanner via l'option -h vous permet de spécifier la liste des machines à scanner. Par exemple 192.168.0.\* ou encore 192.168.1-5.1-255
- Utiliser la lib stantard getopt pour parser les options
- Utiliser la lib socket

## 4 Half open scan

Cette deuxième partie vous propose d'ajouter à votre scanner la technique half-open scan, en respectant les contraintes suivantes :

- Utiliser la *libpcap* pour récupérer les packets.
- Utiliser la *libnet* pour forger les packets.
- Vérifier que l'utilisateur peut effectivement utiliser la technique.
- Ajouter une option pour sélectionner le type de scan désiré : vanilla ou half open.
- Ajouter une option pour spécifier l'interface réseau à utiliser.

## 5 Bring it on

Si vous avez le temps et souhaitez maximiser votre note vous pouvez ajouter des options en plus à votre scanner. Par exemple :

- Afficher le service suppose en intégrant le contenu du fichier */etc/services*.
- Rendre l'ordre des ports tester aléatoire.
- Ecrire un manpage .
- Prévoir une sortie XML.

Cette liste est non exhaustive.