Le scannage de ports vu par l'administrateur

Konstantin Klyagin

Bien que le scannage en tant que tel ne soit pas nuisible, il peut être utilisé à des fins malicieuses. Grâce à ces informations, l'intrus pourrait avoir des accès non autorisés au système. Quelles méthodes pouvons-nous appliquer pour rendre plus difficile le scannage des ports ? Comment détecter ces actions quand l'intrus se sert de techniques de plus en plus sophistiquées ?

Une surprise vous attend sur *Hakin9 Live*. Après avoir lancé le CD, tapez la commande :

```
$ nmap localhost
```

Ce que vous venez de lancer s'appelle un scannage des ports. C'est une des techniques (non invasives) de collecte d'informations sur toutes victimes potentielles, ce sont des informations qui ont leurs importances. En effet, elles permettent de collecter les données des services lancés sur un ordinateur distant. Si un service écoute sur le port TCP, il s'agit certainement d'un programme et peut donc être vulnérable à une attaque. La connaissance des ports ouverts de la victime est très utile pour les attaques futures.

Prenons un exemple, le port 22 (SSH) est ouvert il sera alors facile de déterminer la version du démon qui écoute ce port. Admettons que sur le port fonctionne le serveur *OpenSSH* en version 3.7.1p1. N'importe quelle personne se tenant au courant sur la sécurité informatique va pouvoir obtenir un accès à cette machine et cela sans aucun mot de passe !

Méthodes de scannage

Pour mieux comprendre le principe de scannage, imaginons que nous voulons entrer dans une maisons à 65535 portes. Nous appuyons donc sur chaque poignée pour voir si certaines ne sont pas ouvertes. Mais le fait qu'une porte soit ouverte ne signifie pas que nous pouvons entrer dans la maison. De plus, chaque porte peut être surveillée par un gardien (nous parlons dans notre cas d'un service démarré).Vous ne serez donc pas en mesure d'y accéder tant que vous ne l'aurez pas rendu

Cet article explique ...

- la manière de scanner les ports à l'aide de méthodes difficiles à détecter,
- la manière de détecter les tentatives de scannage effectuées par le biais de ces méthodes.

Ce qu'il faut savoir ...

- utiliser les systèmes de la famille Unix,
- avoir des notions de base sur le protocole TCP/IP (nous recommandons l'article *Attaques DDoS – bases et pratique* de *Hakin9* 4/2004 et la documentation sur le CD).

inopérant. Pourtant, il peut arriver que le gardien ne soit pas présent ou qu'il soit dans l'incapacité de surveiller correctement sa porte. Notre tâche d'information sur une future intrusion est alors accomplie si nous savons que la maison est ouverte. Analysons d'abord les méthodes qui vont nous permettre d'appuyer sur les poignées.

TCP connect() : un simple toc-toc

La méthode la plus simple de vérifier si quelqu'un écoute le port TCP consiste à tenter de s'y connecter. Pour effectuer le scannage complet à l'aide de cette méthode, il faut se connecter successivement à tous les ports, de 1 à 65535. Du point de vue technique, cette méthode nécessite d'appeler la fonction *connect()*. Mais nous n'allons pas nous occuper de la programmation d'un tel logiciel, nous allons utiliser pour ce faire les outils les plus simples disponibles dans chaque système d'exploitation. Le plus simple est d'utiliser le programme *telnet* :

```
$ telnet boite.inutile.com 25
```

Après l'exécution de cette commande, nous pouvons savoir si quelqu'un écoute le port 25 (SMTP ordinaire) – si le message *connection failed* n'apparaît pas, cela signifie qu'un programme attend la connexion sur ce port.

La situation se complique quand le port scanné est filtré. Cela signifie qu'entre la machine de celui qui effectue le scannage et la machine de la victime, il y a un pare-feu qui refuse les paquets de l'adresse IP source effectuant le scannage de ports vers ce port destination et cette adresse IP de destination. Les paquets peuvent être aussi filtrés du point de vue des ports cibles ou autres paramètres réseaux – les pare-feux sont de leur nature assez souples. Si les ports de la victime sont filtrés, le scannage peut s'avérer inutile.

Le scannage de la plage définie (par exemple, de 1 à 6667) des ports est appelé *strobe scan*, par contre, le scannage d'un port sur plusieurs hôtes est appelé – *sweep scan*. Ces types de modifications du scannage *TCP connect()* simulent les tentatives de connexion, elles sont alors plus difficiles à identifier. Un administrateur va s'interroger à savoir si c'était un scannage ou bien une tentative aléatoire de connexion.

La méthode *TCP connect()*, bien qu'elle soit simple, a un défaut grave. Chaque service enregistrera une tentative de connexion. Il n'enregistrera pas le numéro de la carte d'identité de l'intrus, mais sans doute son adresse IP sera enregistrée dans le journal du programme et il devient alors facile de détecter l'intrus. Pour ne pas nous faire capturer, nous pouvons utiliser une autre méthode exploitant les paquets TCP SYN.

TCP SYN : sur la pointe des pieds

Un autre nom du scannage TCP SYN est *scannage mi-ouvert*. Cela signifie que lors du scannage, la connexion TCP pleine n'est pas établie. Pour bien comprendre ce mécanisme, il faut connaître le principe de fonctionnement du protocole TCP/IP. La description que nous donnons ci-dessous est assez générale, les personnes intéressées trouveront plus d'informations sur ce sujet dans l'article *Attaques DDoS – base et pratique* du *Hakin9* 4/2004 et dans la documentation disponible sur le CD.

Dans la phase initiale de l'établissement d'une connexion TCP, le client est obligé d'envoyer le paquet SYN. Après sa réception, le serveur devrait répondre par le paquet SYN+ACK ou RST. SYN+ACK signifie la réponse positive (*j'accepte que la connexion soit établie*), par contre RST – négative. L'établissement de la connexion se termine par l'accusé de réception, c'est-à-dire par l'envoi du paquet ACK par le client qui initialise la connexion.

Il est alors évident que pour vérifier si un port donné est ouvert, nous ne devons pas établir une connexion en tant que telle, – il suffit donc d'envoyer le paquet SYN et d'attendre la réponse du serveur. Si nous obtenons la réponse SYN+ACK nous pouvons en déduire que le port est ouvert et il ne sert à rien de continuer à établir la connexion. Grâce au fait que la connexion n'aie pas complètement été établie, nous pouvons duper certains outils simples de détection de scannage de ports.

Essayons à présent d'appliquer ce mécanisme manuellement. Un simple *telnet* que nous avons utilisé auparavant ne va plus suffir. Pour créer les paquets TCP nécessaires, nous allons utiliser le programme *SendIP*. Cet outil doit être lancé en mode administrateur :

```
# sendip 192.168.1.2 -p ipv4 \
  -is 192.168.1.1 -id 192.168.1.2 \
  -p tcp -td 25 -tfs 1
```

SendIP est un petit programme très puissant qui permet d'envoyer les paquets voulus. À l'aide des options qui suivent la commande il est alors possible de déterminer le type de paquet et les paramètres de ses entêtes. Dans la commande ci-dessus, nous avons déterminé que le paquet sera envoyé via le protocole IPV4 (`-p`), nous avons défini l'adresse IP source (`-is`) et cible (`-id`). Ensuite, nous sommes entrés dans les détails en définissant le type de paquet (`-p`), le port de destination (`-td`) et pour finir nous avons positionné le bit SYN (`-tfs`). Pour plus d'informations sur les possibilités de *SendIP,* référez vous à la page *man* du programme.

SendIP permet d'envoyer uniquement des paquets, nous allons nous servir d'un autre outil pour enregistrer la réponse du serveur. Dans notre cas nous utilisons un sniffeur standard *tcpdump* :

```
# tcpdump \
  'dst 192.168.1.1 && src 192.168.1.2'
```

Cette commande lance une écoute et va donc afficher tous les paquets à destination de notre ordinateur provenant de l'adresse 192.168.1.2. Si le trafic entre ces deux machines n'est pas trop important, il ne va pas