ELALITTE

Zeste de savoir

Introduction au scan de ports

12 août 2019

Table des matières

Vous avez peut-être entendu parler de scan de port. Mais peu de gens savent exactement de quoi il en retourne. Nous allons essayer dans ce tutoriel de comprendre les notions techniques associées aux scans de ports (ports, protocoles TCP, UDP, etc.) pour pouvoir ensuite comprendre comment se déroule un scan de port et savoir s'en servir efficacement.

1. Introduction au scan de ports

1.1. Pré-requis

Vous devez avoir des connaissances de base en réseau pour suivre ce tutoriel, notamment sur le protocole IP ou les protocoles TCP et UDP.

Je vous conseille d'être sous linux pour suivre ce tutoriel, même s'il est possible de mettre en œuvre la plupart des commandes utilisées sous windows. Les exemples donnés seront cependant effectués sous linux debian etch, **en tant que root**. Si vous avez peur de faire des bêtises avec root, ne les faites pas, mais lancer ces commandes avec un utilisateur quelconque ne vous servira pas à grand chose car seul root a les droits nécessaires pour la plupart des commandes utilisées.

1.2. Le scan de ports c'est quoi ?

1.2.1. Heu... déjà, un port c'est quoi ?

Un port est **l'adresse d'une application** sur une machine.

Par exemple, si j'installe et que je mets en marche un serveur web sur ma machine, le port 80 sera *ouvert* et permettra à des personnes de se connecter sur mon port 80 pour y voir mon site web. 80 sera l'adresse de mon application web. Quand vous taperez le nom d'un site à joindre dans votre navigateur, celui-ci va automatiquement interroger le port 80 du serveur demandé.

Par défaut, la plupart des machines, qu'elles fonctionnent en tant que serveur ou client, ont des ports ouverts.

1.2.2. Comment voir les ports ouverts sur ma machine ?

Il y a différentes façons, la plus simple étant de se mettre en ligne de commande. La commande est *netstat*. Mais elle doit être utilisée avec certaines options pour être lisible.

Par exemple :

```
1 # netstat -an
```

Ou mieux pour avoir des détails sur le PID, l'utilisateur et le processus :

```
1 # netstat -anpe
```

Ou pour seulement les ports TCP :

```
1 # netstat -antp
```

Contenu masqué n°1

La première partie nous donne les ports TCP ouverts, puis TCP sur IPv6, puis UDP, UDP sur IPv6. le reste ne nous intéressant pas.

Nous voyons donc que ma machine a un bon paquet de ports ouverts !

On peut d'ores et déjà remarquer la colonne *Etat* qui a des résultats différents pour nos différents ports. Mais que sont ces états ?

1.2.3. Que sont les états des connexions ?

Quand vous utilisez le protocole TCP, chaque connexion doit être établie avant de pouvoir envoyer la moindre information. C'est comme quand vous utilisez le téléphone, vous ne parlez qu'après avoir dit Allo et avoir entendu votre correspondant vous répondre et que vous êtes sûr que la connexion est établie ! En TCP, c'est pareil ! TCP gère donc les états de la connexion. Nous n'allons pas rentrer dans les détails et allons nous concentrer sur deux états principaux, l'état LISTEN et l'état ESTABLISHED.

En état LISTEN, l'application est en écoute et en attente de requêtes de la part de clients sur Internet. En état ESTABLISHED, l'application a établi la communication suite à une demande de requête et on considère donc la connexion comme *établie*.

Quand quelqu'un se connecte à un port (une application web sur un serveur par exemple) l'application va passer de l'état LISTEN à l'état ESTABLISHED.

Mais alors comment quelqu'un d'autre pourra se connecter en même temps ?

L'application va en fait se dupliquer ! Nous aurons une instance de l'application qui sera en état ESTABLISHED en cours de communication avec un client. Et une autre instance qui sera de nouveau en écoute en état LISTEN pour pouvoir recevoir de nouvelles connexions !

C'est ce que nous voyons ci-dessous (extrait du netstat précédent)

```
1 tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
2 tcp        0    304 10.8.98.235:22          86.64.78.254:43580      ESTABLISHE
```

La première ligne montre bien que le port 22 est en état LISTEN. La seconde ligne montre qu'une connexion est établie en ce moment sur le port 22 entre la machine 86.64.78.254 et le serveur 10.8.98.235.

Nous voyons par ailleurs que la machine 86.64.78.254 écoute elle aussi sur le port 43580.

Quel est ce port bizarre ?

En fait, c'est le port choisi aléatoirement par le navigateur du client pour recevoir les réponses du serveur. Le client envoie ses requêtes sur le port 80 du serveur, et celui-ci lui répond sur le port aléatoire qu'il a choisi en initialisant la connexion.

Nous savons maintenant ce qu'est un port et nous avons vu que nos machine ont des ports ouverts.

- Ceux qui sont en écoute car nous avons des services pour les autres (serveur web, serveur de messagerie, etc.)
- Et ceux qui sont établis car nous sommes en train de communiquer avec une autre machine.

Donc revenons-en à nos moutons :

1.2.4. Qu'est-ce qu'un scan de port ?

Un scan de port a pour objectif de m'indiquer quels sont les ports ouverts sur une machine.

Cette définition est cependant très vague, car qu'entendons-nous exactement par port *ouvert* ?

Un port *ouvert* est un port en état LISTEN s'il s'agit de TCP, ou simplement en écoute s'il s'agit d'UDP (UDP né gérant pas l'établissement de connexions).

Ainsi lors d'un scan de ports, nous ne verrons *que* les ports en état LISTEN. Ceux en état ESTABLISHED n'apparaîtront pas. Et cela est normal puisque ces ports servent à une communication spécifique entre deux machines, une troisième machine n'a donc pas le droit d'entrer dans la conversation et de voir ces ports !

Maintenant que nous avons vu globalement ce qu'était un scan de ports, nous allons nous intéresser à son fonctionnement.

1.3. Comment fonctionne un scan de ports ?

Pour comprendre comment fonctionne un scan de ports, nous allons d'abord devoir comprendre comment fonctionne une connexion, que ce soit en TCP ou UDP.

1.3.1. Fonctionnement d'une connexion en UDP

Imaginons que j'aie une application en écoute en UDP sur un port donné. Nous choisirons l'application DNS dans notre cas qui utilise le port UDP 53.

```
# netstat -anpe | grep tinydns
udp        0      0 88.191.51.73:53         0.0.0.0:*
```

Nous voyons bien le port 53 en écoute. Nous allons maintenant essayer d'interroger ce port en faisant une requête DNS vers notre service :

```
# dig @88.191.51.73 www.itinet.fr

; <<>> DiG 9.3.4 <<>> @88.191.51.73 www.itinet.fr
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61052
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 3

;; QUESTION SECTION:
;www.itinet.fr.                         IN        A

;; ANSWER SECTION:
www.itinet.fr.                86400        IN        A        88.191.51.73

;; AUTHORITY SECTION:
```

```
itinet.fr.                259200        IN        NS        sd-8131.dedibox.fr
itinet.fr.                259200        IN        NS        sd-6555.dedibox.fr

;; ADDITIONAL SECTION:
sd-8131.dedibox.fr.        259200        IN        A        88.191.51.73
sd-8131.dedibox.fr.        259200        IN        A        88.191.51.73
sd-6555.dedibox.fr.        259200        IN        A        88.191.45.68

;; Query time: 2 msec
;; SERVER: 88.191.51.73#53(88.191.51.73)
;; WHEN: Wed Dec 23 14:06:25 2009
;; MSG SIZE  rcvd: 147
```

Et regardons ce qui s'est passé au niveau réseau :

```
# tcpdump -n -i eth0 udp and port 53
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
14:36:45.845240 IP 87.88.75.75.55947 > 88.191.51.73.53:  60135+ A? www.itinet.
14:36:45.845680 IP 88.191.51.73.53 > 87.88.75.75.55947:  60135*-
     1/2/3 A 88.191.51.73 (147)
```

Nous voyons bien la requête vers notre serveur et la réponse qui lui est faite.

? Maintenant que se passe-t-il si nous interrogeons un port fermé ?

Nous pouvons par exemple interroger un serveur sur son port 53 alors qu'il n'y a pas de service DNS en écoute :

```
# dig @10.8.98.235 www.itinet.fr
```

Et nous voyons le résultat au niveau réseau :

```
# tcpdump -i lo udp or icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on lo, link-type EN10MB (Ethernet), capture size 96 bytes
15:31:54.234103 IP localhost.60481 > localhost.domain: 41667+ A? www.itinet.fr
15:31:54.234158 IP localhost > localhost: ICMP localhost udp port domain unrea
15:31:59.233857 IP localhost.60481 > localhost.domain: 41667+ A? www.itinet.fr
15:31:59.233914 IP localhost > localhost: ICMP localhost udp port domain unrea
```

Nous voyons ici que le client émet la requête, et que le serveur répond. Mais il répond avec un message ICMP *port unreachable*.

Nous avons donc deux cas :

— Si le port est ouvert, l'application répond en UDP
— Si le port est fermé, l'application répond avec un message d'erreur ICMP port unreachable.

Ainsi nous pourrons scanner facilement des ports UDP et savoir ceux qui sont ouverts ou fermés. Il nous suffira d'envoyer des requêtes UDP. Selon la réponse en UDP ou ICMP, nous saurons si le port testé était ouvert ou fermé.

1.3.2. Fonctionnement d'une connexion en TCP

Le déroulement d'une connexion en TCP est un peu plus complexe, mais nous allons voir que cela reste simple dans le cas d'un scan de ports.

Regardons l'établissement d'une connexion TCP, avec une requête web vers un serveur web par exemple :

```
1 # wget 88.191.51.73
2 --2009-12-23 16:02:17--  http://88.191.51.73/
3 Connexion vers 88.191.51.73:80...connecté.
```

Et le résultat au niveau réseau :

```
1 # tcpdump -n -i eth0 port http
2 16:08:36.925010 IP 87.88.75.75.57353 > 88.191.51.73.80: S 3555076962:355507696
3 16:08:36.925040 IP 88.191.51.73.80 > 87.88.75.75.57353: S 2008108463:200810846
4 16:08:36.953605 IP 87.88.75.75.57353 > 88.191.51.73.80: . ack 1 win 92 <nop,nop
```

Nous voyons que la connexion a été établie en trois temps :

— Envoi d'un segment avec le flag SYN (S dans le premier paquet)
— Réponse d'un segment avec les flags SYN et ACK de positionnés (S et ack dans le second paquet)
— Réponse enfin d'un segment avec le flag ack (ack positionné, mais sans le S !).

C'est le fonctionnement normal de l'établissement d'une connexion TCP.

? Mais que se passe-t-il maintenant si le port est fermé ?

Essayons de nous connecter sur un port fermé :

```
1 # wget 88.191.51.73:81
2 --2009-12-23 16:12:54--  http://88.191.51.73:81/
3 Connexion vers 88.191.51.73:81...échec: Connexion refusée.
```

La réponse est claire, c'est fermé ! Regardons ce qu'il s'est passé au niveau réseau :

```
# tcpdump -n -i eth0 port 81
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
16:19:14.709972 IP 87.88.75.75.52063 > 88.191.51.73.81: S 678805342:678805342(0
16:19:14.710192 IP 88.191.51.73.81 > 87.88.75.75.52063: R 0:0(0) ack 678805343
```

Notre machine a bien envoyé un segment avec le flag SYN de positionné. Par contre la machine en face a répondu d'un segment avec le flag RESET de positionné. On s'est fait jetés !

Donc là encore nous avons rencontré deux cas :

- Si le port est ouvert, réponse avec un segment SYN + ACK
- Si le port est fermé, réponse avec un segment RST.
  Là encore il va nous être facile de balayer une plage de ports et de savoir lesquels sont ouverts quand nous recevrons une réponse SYN + ACK.

1.3.3. Et donc notre scan de ports

Nous avons vu comment répondaient des ports fermés et des ports ouverts, en UDP et TCP. Mais nous les avons interrogés un par un et cela est un peu fastidieux... Il serait plus sympa d'avoir un outil qui scanne toute une plage de ports pour nous.

C'est là qu'intervient l'outil magique **nmap**.

? Qu'est-ce que nmap ?

nmap est un outil qui permet de scanner de façon automatique toute une plage de ports, toute une plage d'adresses, et permet d'obtenir une foultitude de résultats. Dans notre étude nous allons nous limiter au scan de ports, mais vous aurez tout loisir de tester les innombrables fonctionnalités de nmap par vous-même.

Si je veux savoir par exemple quels sont les ports ouverts sur la machine 10.8.97.1, je peux faire la commande suivante :

```
# nmap -sS 10.8.97.1

Starting Nmap 4.62 ( http://nmap.org ) at 2009-12-23 16:32 CET
Interesting ports on (10.8.97.1):
Not shown: 1703 closed ports
PORT    STATE SERVICE
13/tcp  open  daytime
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
```

```
53/tcp  open  domain
80/tcp  open  http
113/tcp open  auth
139/tcp open  netbios-ssn
143/tcp open  imap
199/tcp open  smux
445/tcp open  microsoft-ds
993/tcp open  imaps
MAC Address: 00:00:24:C6:2B:D1 (Connect AS)

Nmap done: 1 IP address (1 host up) scanned in 18.965 seconds
```

J'ai utilisé l'option -sS qui précise que je veux faire un scan avec des segments SYN, je ne scanne donc que les ports TCP. Le résultat est probant, j'ai trouvé une foultitude de ports ouverts ! Attention cependant, nmap n'a pas scanné *tous* les ports possibles sur la machine distante, mais seulement les plus utilisés (environ 1750 ports).

Je peux aussi scanner les ports UDP :

```
# nmap -sU 10.8.97.1

Starting Nmap 4.62 ( http://nmap.org ) at 2009-12-23 16:37 CET
Interesting ports on labo.itinet.fr (10.8.97.1):
Not shown: 1481 closed ports
PORT     STATE         SERVICE
53/udp   open|filtered domain
137/udp  open|filtered netbios-ns
138/udp  open|filtered netbios-dgm
161/udp  open|filtered snmp
500/udp  open|filtered isakmp
514/udp  open|filtered syslog
4500/udp open|filtered sae-urn
MAC Address: 00:00:24:C6:2B:D1 (Connect AS)

Nmap done: 1 IP address (1 host up) scanned in 77.106 seconds
```

Là encore j'ai l'embarras du choix sur les ports ouverts !

Vous savez maintenant faire un scan de ports, mais qu'en faire ?

1.4. Pourquoi faire avec un scan de ports ?

Il y a globalement deux raisons pour lesquelles on voudrait faire un scan de ports :

— Pour tester nos propres systèmes
— Pour tester les sytèmes des autres...

1.4.1. Tester ses propres systèmes

Dans le premier cas, rien de plus normal. Nous souhaitons voir comment notre machine est vue depuis l'extérieur, quels sont les services qui sont accessibles, et quels sont ceux qui ne le sont pas. Cela permet aussi de vérifier que la sécurité que l'on a mise en place est correcte, notamment dans le cas d'un firewall.

1.4.2. Tester les systèmes des autres

Dans le second cas, les motivations ne sont pas toujours très honnêtes... Le scan d'une machine distante est souvent un préalable à une attaque en règle plus ciblée. Cela permet à l'attaquant de voir les services qui tournent à distance, et le cas échéant de trouver une faille sur l'un de ces services afin d'aller plus loin dans l'attaque. Donc pour faire simple, ce n'est pas bien.

Ne vous aventurez pas à faire un scan sur une machine que vous ne possédez pas. La personne responsable de la machine en face pourrait engager des poursuites judiciaires à votre encontre...

De la même façon, ne croyez pas tous les (boni)menteurs qui vous diront que vous êtes anonymes et que vous ne risquez rien. Si vous n'êtes pas un spécialiste, il est relativement facile de remonter jusqu'à vous.

1.4.3. Fonctionnalités avancées

Un scan de port peut apporter beaucoup plus d'informations que simplement savoir si un port est ouvert ou fermé. Vous pourrez notamment trouver :

— Les versions des services qui tournent
— Les types de systèmes d'exploitation et leurs versions
— La présence d'un firewall
— Les ports ouverts sur le firewall
— La politique de filtrage en fonction d'adresses IP spécifiques
— etc.

Bref, il y a une foultitude d'informations à trouver et à connaître. Pour les plus courageux, je vous invite à la lecture du site consacré à nmap qui décrit en détail son fonctionnement et ses nombreuses options et possibilités.

J'espère que ce tuto vous a plu et qu'il vous permettra de mieux comprendre la notion de ports et les scans qui leur sont associés.

Un élément important à prendre en compte est que pour faire de la sécurité, vous devez comprendre dans le détail ce que vous faites. Si vous ne faites que survoler la connaissance, vous ne ferez jamais qu'utiliser des outils que d'autres auront fait sans vraiment comprendre ce

que vous faites. Alors maintenant, potassez et faites vous plaisir ! (en scannant **vos propres machines**, ou **celles de vos amis**, bien entendu)

Contenu masqué

Contenu masqué n°1

```
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Adresse locale          Adresse distante        Etat
tcp        0      0 0.0.0.0:1984            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:8833          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:7009            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:11211         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:654           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:113             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:39185           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:25              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:8833          127.0.0.1:40104         ESTABLISHED
tcp        0      0 127.0.0.1:47475         127.0.0.1:1984          TIME_WAIT

tcp        0      0 127.0.0.1:22            127.0.0.1:59487         ESTABLISHED
tcp        0      0 127.0.0.1:59487         127.0.0.1:22            ESTABLISHED
tcp        0      0 127.0.0.1:40104         127.0.0.1:8833          ESTABLISHED
tcp        0      0 127.0.0.1:47476         127.0.0.1:1984          TIME_WAIT

tcp        0    304 10.8.98.235:22          86.64.78.254:43580      ESTABLISHED
tcp6       0      0 :::7009                 :::*                    LISTEN
tcp6       0      0 :::139                  :::*                    LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::445                  :::*                    LISTEN
tcp6       0      0 10.8.98.235:80          217.167.139.210:57864   TIME_WAIT

udp        0      0 10.8.98.235:137         0.0.0.0:*
udp        0      0 0.0.0.0:137             0.0.0.0:*
udp        0      0 10.8.98.235:138         0.0.0.0:*
udp        0      0 0.0.0.0:138             0.0.0.0:*
udp        0      0 0.0.0.0:926             0.0.0.0:*
udp        0      0 127.0.0.1:161           0.0.0.0:*
udp        0      0 0.0.0.0:68              0.0.0.0:*
udp        0      0 0.0.0.0:35693           0.0.0.0:*
udp        0      0 0.0.0.0:111             0.0.0.0:*
udp        0      0 10.8.98.235:123         0.0.0.0:*
```

Contenu masqué

```
udp        0      0 127.0.0.1:123           0.0.0.0:*
udp        0      0 0.0.0.0:123             0.0.0.0:*
udp6       0      0 fe80::208:2ff:fe3f::123 :::*
udp6       0      0 ::1:123                 :::*
udp6       0      0 :::123                  :::*
Sockets du domaine UNIX actives(serveurs et établies)
Proto RefCnt Flags       Type       State         I-Node   PID/Program name
unix  2      [ ACC ]     STREAM     LISTENING     5996     2185/hald
unix  2      [ ACC ]     STREAM     LISTENING     8547689  17734/gnome-session
unix  2      [ ACC ]     STREAM     LISTENING     5723     2143/master
unix  2      [ ACC ]     STREAM     LISTENING     5730     2143/master
unix  2      [ ACC ]     STREAM     LISTENING     5734     2143/master
unix  2      [ ACC ]     STREAM     LISTENING     5738     2143/master
```

Retourner au texte.