

Scan de ports avec Scapy

Quelques rappels (Linux)

- Un **port** est l'**adresse** d'une application (ou service) **TCP ou UDP** sur un hôte au niveau de la **couche 4 (Transport)**,
- selon un **service** est **actif** ou en **écoute**, le port applicatif correspondant (associé) est **ouvert**,
- les numéros de port sont **codés sur 16 bits**, ce qui permet **65 536 ports distincts** par hôte,
- les ports sont **classés** en 3 catégories en fonction de leur numéro:
 - les numéros de port de **0 à 1 023** correspondent aux ports **bien-connus** (well-known ports) et sont utilisés pour les services réseaux les plus courants,
 - les numéros de ports de **1 024 à 49 151** correspondent aux **ports enregistrés** (registered ports), assignés par l'IANA,
 - les numéros de ports de **49 152 à 65 535** correspondent aux **ports dynamiques**, utilisables pour tout type de requêtes TCP ou UDP autres que celle citées précédemment.

Pour en savoir plus voir la page de Wikipdia : [https://fr.wikipedia.org/wiki/Port_\(logiciel\)](https://fr.wikipedia.org/wiki/Port_(logiciel))

Pour **visualiser** les ports ouverts d'un hôte et donc connaître les **services actifs** (en écoute) :

```
root@debian:~# netstat -tanpu
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Adresse locale     Adresse distante      Etat        PID/Program name
tcp      0      0 127.0.0.1:3306  0.0.0.0:*
tcp      0      0 0.0.0.0:22    0.0.0.0:*
tcp      0      0 127.0.0.1:25  0.0.0.0:*
tcp      0      464 192.168.1.159:22 192.168.1.196:50264 ESTABLISHED 1119/sshd: root@pts
tcp6     0      0 :::80          :::*
tcp6     0      0 :::22          :::*
tcp6     0      0 ::1:25          :::*
udp      0      0 0.0.0.0:68    0.0.0.0:*
root@debian:~#
```

- la colonne **Etat** indique :
 - **LISTEN** pour une application qui est en écoute et en attente de requêtes de la part de clients,
 - **ESTABLISHED** quand une application a établi la communication suite à une demande de requête. * Quand un client a pu se connecter à un port, l'état de l'application passe de **LISTEN** à **ESTABLISHED**. * Quand plusieurs clients se connectent à la même application : * il y a pour chaque client une instance de l'application avec l'état **ESTABLISHED**, * et une une autre instance en écoute avec l'état **LISTEN** pour recevoir de nouvelles connexions.

Exemple du service SSH (port 22) :

```
tcp  0      0 0.0.0.0:22      0.0.0.0:*
tcp  0      464 192.168.1.159:22 192.168.1.196:50264 ESTABLISHED 1119/sshd:
root@pts
```

- la première ligne montre que le port 22 est à l'état **LISTEN**,
- La seconde ligne montre une connexion établie (**ESTABLISHED**)
 - sur le **port 22 du serveur** 192.168.1.159
 - avec le client 192.168.1.196 sur le **port 50264**.

Le **port 50264 est choisi aléatoirement** par le client lors de l'initialisation de la connexion, pour communiquer avec le serveur.

- Un **port TCP ouvert** est un port en état **LISTEN**,
- Un **port UDP** est simplement en **écoute** car UDP **ne gère pas** l'établissement de connexions.

```
udp  0      0 0.0.0.0:68      0.0.0.0:*
root@pts
```

- Lors d'un scan de ports, on ne **visualise que les ports en état LISTEN**.
- Les ports en l'état **ESTABLISHED** ne peuvent pas être visualisés puisque cela correspond à une communication spécifique entre deux hôtes et un troisième hôte ne peut entrer dans la communication qui a été établie.

===== Scan de port TCP===== Source Wikipedia :

https://fr.wikipedia.org/wiki/Transmission_Control_Protocol#.C3.89tablissement_d.27une_connexion Schéma

d'établissement d'une connexion TCP :

- Le client **envoie** un segment avec le flag **SYN** au serveur, - Le serveur lui **répond** par un segment avec le flag **SYN/ACK**, - Le client **confirme** par un segment avec le flag **ACK**. Durant cet échange initial, les numéros de séquence des deux parties sont synchronisés : - Le client utilise son numéro de séquence initial dans le champ "Numéro de séquence" du segment SYN (x par exemple), - Le serveur utilise son numéro de séquence initial dans le champ "Numéro de séquence" du segment SYN/ACK (y par exemple) et ajoute le numéro de séquence du client plus un (x+1) dans le champ "Numéro d'acquittement" du segment, - Le client confirme en envoyant un ACK avec un numéro de séquence augmenté de un (x+1) et un numéro d'acquittement correspondant au numéro de séquence du serveur plus un (y+1). Si le serveur **rejette la connexion**, il répond avec un segment dont le flag est **RESET**. ===== Attributs disponible et valeur par défaut pour le protocole TCP ===== <code python> >> ls(TCP) sport : ShortEnumField = (20) dport : ShortEnumField = (80) seq : IntField = (0) ack : IntField = (0) dataofs : BitField = (None) reserved : BitField = (0) flags : FlagsField = (2) window : ShortField = (8192) checksum : XShortField = (None) urgptr : ShortField = (0) options : TCPOptionsField = ({}) >> </code> ===== Envoi d'un paquet TCP sur le service Web d'un serveur (port 80) avec le flag SYN ===== * valeurs du segment TCP à préciser : ** **port de destination 80** pour le service Web ** **port source client 55 555** (choisi au dessus de 49 151) ** **flag** positionné à **SYN** (valeur S) * création du segment encapsulé dans un paquet IP destiné à l'hôte 192.168.1.1 (Box Internet) : <code python> >> paquet = IP(dst='192.168.1.1') / TCP(sport=55555, dport=80, flags='S') >> </code> * envoi du paquet <code python> >> rep, non_rep = sr(paquet) Begin emission: ..Finished to send 1 packets. .. Received 5 packets, got 1 answers, remaining 0 packets >> rep.show() 0000 IP / TCP 192.168.1.159:55555 > 192.168.1.1:htp S ==> IP / TCP 192.168.1.1:htp > 192.168.1.159:55555 SA / Padding >> </code> * Le résultat est toujours composé des deux couples paquet émis / paquet reçu. * Le paquet envoyé sur le port 80 (Scapy affiche **http** à la place du numéro de port 80) avec le flag positionné à **SYN** (Scapy affiche un **S** pour l'attribut flags ce qui correspond à la valeur numérique 2) <code python> >> rep[0][0][TCP].show() ###[TCP]###[sport= 55555 dport= http seq= 0 ack= 0 dataofs= None reserved= 0 flags= S window= 8192 checksum= None urgptr= 0 options= {} >> </code> * le paquet reçu correspond a les flags positionnés à **SYN** et **ACK** (Scapy affiche **SA** ce qui correspond à la valeur numérique 18) => le port 80 est ouvert. <code python> rep[0][1][TCP].show() ###[TCP]###[sport= http dport= 55555 seq= 3631598158 ack= 1 dataofs= 6 reserved= 0 flags= SA window= 14600 checksum= 0x5f04 urgptr= 0 options= [('MSS', 1460)] ###[Padding]###[load= 'I5' >> </code> ===== Envoi d'un paquet TCP sur le port 22 d'un serveur alors que le service SSH n'est pas actif ===== * valeurs du segment TCP à préciser : ** **port de destination 22** pour le service SSH ** **port source client 55 555** (choisi au dessus de 49 151) * **flag** positionné à **SYN** (valeur S) * création du segment encapsulé dans un paquet IP destiné à l'hôte 192.168.1.100 (Imprimante réseau) : <code python> >> paquet = IP(dst='192.168.1.100') / TCP(sport=55555, dport=22, flags='S') >> rep, non_rep = sr(paquet) Begin emission: ..Finished to send 1 packets. .. Received 5 packets, got 1 answers, remaining 0 packets >> rep.show() 0000 IP / TCP 192.168.1.159:55555 > 192.168.1.100:ssh S ==> IP / TCP 192.168.1.100:ssh > 192.168.1.159:55555 RA / Padding >> rep[0][1][TCP].show() ###[TCP]###[sport= ssh dport= 55555 seq= 0 ack= 1 dataofs= 5 reserved= 0 flags= RA window= 0 checksum= 0x5262 urgptr= 0 options= {} ###[Padding]###[load= '\x00\x00\x00\x00\x00\x00' >> </code> * Le résultat est toujours composé des deux couples paquet émis / paquet reçu. * Le paquet envoyé sur le port 22 (Scapy affiche **ssh** à la place du numéro de port 80) avec le flag positionné à **SYN** (Scapy affiche un **S** pour l'attribut flags ce qui correspond à la valeur numérique 2) * le paquet reçu correspond a les flags positionnés à **RESET et ACK** (Scapy affiche **RA** ce qui correspond à la valeur numérique 20) => le port 22 n'est pas ouvert. <code python> >> rep[0][1][TCP].flags 20 >> </code> ===== Préciser plusieurs ports ===== Il est possible d'utiliser une liste de valeurs pour les attributs des protocoles sous la forme : * dport=[80,443] pour indiquer une **liste de ports** : ici uniquement les ports 80 et 446 ; * dport=(80,443) pour indiquer une **plage de valeurs** : ici les ports allant de 82 à 443. ===== Scan de port UDP=====

Rappel : Un port UDP est simplement en écoute car UDP ne gère pas l'établissement de connexions.

==== Exemple de scan UDP pour déterminer si le service DNS est actif sur un hôte (en écoute) === Les paramètres à utiliser * le nom/adresse IP du serveur DNS dans le **paquet IP**, * Le **port UDP** a utilisé et qui est **53** dans le **segment UDP**. * il faut **encapsuler** les **informations DNS** sans préciser la requête de résolution de nom car il s'agit d'un simple scan. Deux réponses sont possibles : * Si le port est **ouvert** et donc le service DNS est **actif**, l'application répond en **UDP**, * Si le port est **fermé**, l'application répond avec un message d'**erreur ICMP port-unreachable**. <code python> >> paquet = IP(dst='192.168.1.1') / UDP(dport=53) / DNS() >> rep, nonrep = sr(paquet) Begin emission: ..Finished to send 1 packets. .. Received 5 packets, got 1 answers, remaining 0 packets >> rep.show() 0000 IP / UDP / DNS Qry ==> IP / UDP / DNS Ans / Padding >> paquet = IP(dst='192.168.1.100') / UDP(dport=53) / DNS() >> rep, non_rep = sr(paquet) Begin emission: ..Finished to send 1 packets. .. Received 4 packets, got 1 answers, remaining 0 packets >> rep.show() 0000 IP / UDP / DNS Qry ==> IP / ICMP 192.168.1.100 > 192.168.1.159 dest-unreach port-unreachable / IPerror / UDPerror >> </code> ===== Retour à Python : la bibliothèque Scapy ... =====

■ Python : la bibliothèque Scapy pour manipuler les paquets réseau

From:

/ - **Les cours du BTS SIO**

Permanent link:

</doku.php/dev/python/scapy/scapyscanport-1>

Last update: **2017/11/04 22:42**

