

Techniques et outils de piratage

Reconnaissance : Scanning

Comprendre les attaques pour mieux se défendre

Mohamed Mejri
Mohamed.Mejri@ift.ulaval.ca

September 28, 2016

Plan

- 1 Machines actives
- 2 Services actifs
 - Techniques
 - Outils
- 3 Détection de systèmes d'exploitation
 - Analyse active
 - Analyse passive

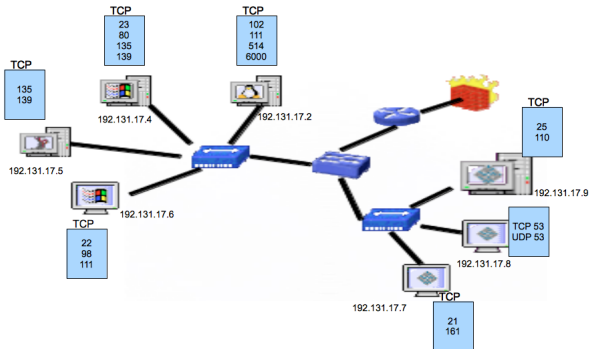
Introduction

Objectifs :

- ➡ Déterminer les machines actives
- ➡ Déterminer les services actifs
- ➡ Déterminer les systèmes d'exploitation

Introduction

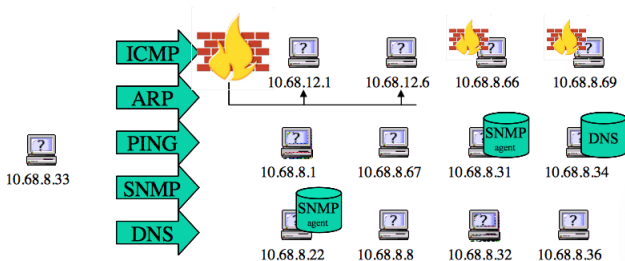
SCANNING



- @IP des machines actives?
- Ports ouverts sur les machines actives?
- Systèmes d'exploitation sur les machines actives?

Machines actives

Découvrir les ordinateurs



source : Centre de recherche sur les communications Canada

Machines actives

Balayage via *Message Echo Request* (ping): Différents outils sont disponibles

- ➔ Fping
- ➔ Nmap
- ➔ SuperScan
- ➔ Icmpenum
- ➔ Etc.

Machines actives

Ping est la moyen le plus basic pour déterminer si une machine est active

- ➡ Il envoie un message ICMP ECHO (type 8) à la machine en question
- ➡ Si la machine est active, elle répond par ICMP ECHO_REPLY (type 0)
- ➡ Sinon, pas de réponse

L'absence de réponse n'implique pas forcément que la machine est inactive (possibilité de blocage de messages par un pare-feu)

Machines actives

D'autres types de messages ICMP (voir RFC 792) peuvent permettre la détection des machines actives et même d'autres informations comme le masque réseau par exemple.

- ➡ type 3 : destination inaccessible
- ➡ type 12 : problème de paramètres
- ➡ type 13 : Demander l'heure dans une machine (*timestamp*). C'est aussi l'heure locale de la zone ciblée.
- ➡ type 14 : réponse au *timestamp*
- ▶ type 17 (*Address Mask Request*). Retourne le masque réseau (utile pour connaître l'@ de diffusion, les sous-réseaux, l'@ de passerelle, etc.).
- ➡ Etc.

Machines actives

fping : *ping* en parallèle) (Linux et Windows)

- ▶ -a : montrer seulement les machines actives
- ▶ -f : lire la liste des machines ciblées à partir d'un fichier

```
[root]$ fping -a -f in.txt
192.168.1.3 is alive
192.168.1.215 is alive
```
- ▶ -g : construire la liste ciblée à partir d'un masque IP ou des adresses début/fin

```
fping -g 192.168.1.0/24
fping -g 192.168.1.0 192.168.1.255
```

icmpquery

- ▶ l'option -m pour avoir le masque d'un réseau

```
[root]$ icmpquery -m 192.161.1.1
192.168.1.1 : 0xFFFFFEE0
```
- ▶ l'option -t pour obtenir l'heure (*Greenwich Mean Time*)

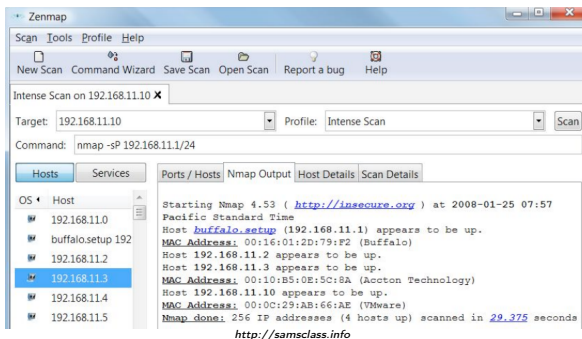
```
[root]$ icmpquery -t 192.161.1.1
192.168.1.1 : 11:36:19
```

Machines actives

`nmap -sP`

```
# nmap -sP 192.168.7.0/24

Starting nmap V. 2.12 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Host (192.168.7.11) appears to be up.
Host (192.168.7.12) appears to be up.
Host (192.168.7.76) appears to be up.
Nmap run completed -- 256 IP addresses (3 hosts up) scanned in 1 second
http://www.linuxsecurity.com/content/view/117695/49/
```



Machines actives

`nmap -sP -PA80`

- Permet aussi de faire des TCP *ping scan* via le port 80 : On envoie un message TCP ACK, les machines actives répondent par TCP RST
- Les paquets destinés au port 80 sont tolérés par le pare-feu pour accéder au DMZ par exemple
- D'autres ports (e.g. de courriels SMTP (25), POP (10), IMAP(143). AUTH(113) peuvent remplacer le port 80.
- Le port 113 est utilisé par des protocoles d'authentification. Quand vous envoyez une demande d'accès à un serveur (courriel, ftp, etc.), ce dernier peut lancer un protocole d'authentification qui utilise généralement le port 113.

Machines actives

nmap -sP -PA80

```
# nmap -sP -PT80 192.168.7.0/24
TCP probe port is 80

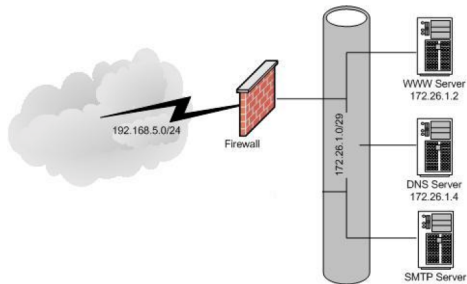
Starting nmap V. 2.12 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Host (192.168.7.11) appears to be up.
Host (192.168.7.12) appears to be up.
Host (192.168.7.76) appears to be up.
Nmap run completed -- 256 IP addresses (3 hosts up) scanned in 1 second

source : http://www.linuxsecurity.com/content/view/117695/49/
```

Machines actives

`nmap -sP -PA80`

Exemple



source : M. Wolfgang: Host Discovery with nmap

Remarque : La commande "`nmap -sP`" se comporte comme "`nmap -sP -PA80`"

Our command:

`Nmap -sP 172.26.1.1`tcpdump Output:

```
09:26:49.324016 192.168.5.20 > 172.26.1.1: ICMP: echo request
09:26:49.324083 192.168.5.20.40435 > 172.26.1.1.http: . ack 1942297083 win 3072
```

source : M. Wolfgang: Host Discovery with nmap

Machines actives

Scénario 1

Our Command:`nmap -sP 172.26.1.0/29`*Firewall Ruleset:*

pass from any to any

tcpdump Output:

```

08:59:58.840249 192.168.5.20 > 172.26.1.0: ICMP: echo request
08:59:58.840667 192.168.5.20.60923 > 172.26.1.0.http: . ack 2990889584 win 3072
08:59:58.840726 192.168.5.20 > 172.26.1.1: ICMP: echo request
08:59:58.840764 192.168.5.20.60923 > 172.26.1.1.http: . ack 1015938099 win 3072
08:59:58.840801 192.168.5.20 > 172.26.1.2: ICMP: echo request
08:59:58.840838 192.168.5.20.60923 > 172.26.1.2.http: . ack 1228729075 win 3072
08:59:58.840876 192.168.5.20 > 172.26.1.3: ICMP: echo request
08:59:58.840914 192.168.5.20.60923 > 172.26.1.3.http: . ack 1769982015 win 3072
08:59:58.840952 192.168.5.20 > 172.26.1.4: ICMP: echo request
08:59:58.840989 192.168.5.20.60923 > 172.26.1.4.http: . ack 1859940754 win 3072
08:59:58.841027 192.168.5.20 > 172.26.1.5: ICMP: echo request
08:59:58.841064 192.168.5.20.60923 > 172.26.1.5.http: . ack 1596045207 win 3072
08:59:58.841103 192.168.5.20 > 172.26.1.6: ICMP: echo request
08:59:58.841140 192.168.5.20.60923 > 172.26.1.6.http: . ack 550856434 win 3072
08:59:58.841178 192.168.5.20 > 172.26.1.7: ICMP: echo request
08:59:58.841215 192.168.5.20.60923 > 172.26.1.7.http: . ack 2476756145 win 3072
08:59:58.841886 172.26.1.2 > 192.168.5.20: ICMP: echo reply
08:59:58.842149 172.26.1.4 > 192.168.5.20: ICMP: echo reply
08:59:58.842377 172.26.1.2.http > 192.168.5.20.60923: R
1228729075:1228729075(0) win 0 (DF)
08:59:58.842699 192.168.5.5 > 192.168.5.20: ICMP: echo reply
08:59:58.842905 172.26.1.4.http > 192.168.5.20.60923: R
1859940754:1859940754(0) win 0 (DF)
08:59:58.843263 172.26.1.6 > 192.168.5.20: ICMP: echo reply
08:59:58.843487 172.26.1.6.http > 192.168.5.20.60923: R 550856434:550856434(0)
win 0 (DF)

```

source : M. Wolfgang: Host Discovery with nmap

Machines actives

Scénario 2 : Règles de pare-feu pour le trafic entrant

Our Command:

```
nmap -sP 172.26.1.0/29
```

Firewall Ruleset:

```
pass from any to any proto tcp port 80
pass from any to any proto tcp port 53
pass from any to any proto tcp port 25
drop all
```

tcpdump Output:

```
09:12:21.505016 192.168.5.20 > 172.26.1.0: ICMP: echo request
09:12:21.505125 192.168.5.20.60212 > 172.26.1.0.http: . ack 3755150488 win 3072
09:12:21.505166 192.168.5.20 > 172.26.1.1: ICMP: echo request
09:12:21.505204 192.168.5.20.60212 > 172.26.1.1.http: . ack 4073218537 win 3072
09:12:21.505242 192.168.5.20 > 172.26.1.2: ICMP: echo request
09:12:21.505280 192.168.5.20.60212 > 172.26.1.2.http: . ack 3464075465 win 3072
09:12:21.505318 192.168.5.20 > 172.26.1.3: ICMP: echo request
09:12:21.505355 192.168.5.20.60212 > 172.26.1.3.http: . ack 962650084 win 3072
09:12:21.505393 192.168.5.20 > 172.26.1.4: ICMP: echo request
09:12:21.505430 192.168.5.20.60212 > 172.26.1.4.http: . ack 337683576 win 3072
09:12:21.505468 192.168.5.20 > 172.26.1.5: ICMP: echo request
09:12:21.505505 192.168.5.20.60212 > 172.26.1.5.http: . ack 1839298263 win 3072
09:12:21.505544 192.168.5.20 > 172.26.1.6: ICMP: echo request
09:12:21.505581 192.168.5.20.60212 > 172.26.1.6.http: . ack 1701634905 win 3072
09:12:21.505619 192.168.5.20 > 172.26.1.7: ICMP: echo request
09:12:21.505656 192.168.5.20.60212 > 172.26.1.7.http: . ack 4287112447 win 3072
09:12:21.506577 172.26.1.2.http > 192.168.5.20.60212: R 3464075465:3464075465(0) win 0 (DF)
09:12:21.506830 172.26.1.6.http > 192.168.5.20.60212: R 1701634905:1701634905(0) win 0 (DF)
09:12:21.507104 172.26.1.4.http > 192.168.5.20.60212: R 337683576:337683576(0) win 0 (DF)
```

source : M. Wolfgang: Host Discovery with nmap

Machines actives

Scénario 3 : pare-feu avec des règles plus strictes

Our Command:`nmap -sP 172.26.1.0/29`*Firewall Ruleset:*

```

pass from any to 172.26.1.2 proto tcp port 80
pass from any to 172.26.1.4 proto tcp port 53
pass from any to 172.26.1.6 proto tcp port 25
drop all

```

tcpdump Output:

```

08:05:07.733225 192.168.5.20 > 172.26.1.0: ICMP: echo request
08:05:07.733334 192.168.5.20.44273 > 172.26.1.0.http: . ack 1621467562 win 2048
08:05:07.733375 192.168.5.20 > 172.26.1.1: ICMP: echo request
08:05:07.733412 192.168.5.20.44273 > 172.26.1.1.http: . ack 1213996683 win 2048
08:05:07.733450 192.168.5.20 > 172.26.1.2: ICMP: echo request
08:05:07.733487 192.168.5.20.44273 > 172.26.1.2.http: . ack 3921129299 win 2048
08:05:07.733525 192.168.5.20 > 172.26.1.3: ICMP: echo request
08:05:07.733561 192.168.5.20.44273 > 172.26.1.3.http: . ack 193217699 win 2048
08:05:07.733598 192.168.5.20 > 172.26.1.4: ICMP: echo request
08:05:07.733635 192.168.5.20.44273 > 172.26.1.4.http: . ack 3130918107 win 2048
08:05:07.733672 192.168.5.20 > 172.26.1.5: ICMP: echo request
08:05:07.733709 192.168.5.20.44273 > 172.26.1.5.http: . ack 638273071 win 2048
08:05:07.733746 192.168.5.20 > 172.26.1.6: ICMP: echo request
08:05:07.733783 192.168.5.20.44273 > 172.26.1.6.http: . ack 4151673259 win 2048
08:05:07.733820 192.168.5.20 > 172.26.1.7: ICMP: echo request
08:05:07.733856 192.168.5.20.44273 > 172.26.1.7.http: . ack 228721671 win 2048
08:05:07.734611 172.26.1.2.http > 192.168.5.20.44273: R 3921129299:3921129299(0) win 0 (DF)

```

source : M. Wolfgang: Host Discovery with nmap

Machines actives

Scénario 4 : pare-feux avec état

Our Command:

```
nmap -sP 172.26.1.0/29
```

Firewall Ruleset:

```
pass from any to 172.26.1.2 proto tcp port 80 keep state
pass from any to 172.26.1.4 proto tcp port 53 keep state
pass from any to 172.26.1.6 proto tcp port 25 keep state
drop all
```

tcpdump Output:

```
08:46:23.548456 192.168.5.20.44390 > 172.26.1.2.http: . ack 3476163011 win 2048
08:46:23.548468 192.168.5.20 > 172.26.1.3: ICMP: echo request
08:46:23.548501 192.168.5.20.44390 > 172.26.1.3.http: . ack 1149703540 win 2048
08:46:23.548559 192.168.5.20 > 172.26.1.4: ICMP: echo request
08:46:23.548596 192.168.5.20.44390 > 172.26.1.4.http: . ack 1314586500 win 2048
08:46:23.548635 192.168.5.20 > 172.26.1.5: ICMP: echo request
08:46:23.548673 192.168.5.20.44390 > 172.26.1.5.http: . ack 2068473993 win 2048
08:46:23.548712 192.168.5.20 > 172.26.1.6: ICMP: echo request
08:46:23.548749 192.168.5.20.44390 > 172.26.1.6.http: . ack 2732407633 win 2048
08:46:23.548789 192.168.5.20 > 172.26.1.7: ICMP: echo request
08:46:23.548825 192.168.5.20.44390 > 172.26.1.7.http: . ack 2518875875 win 2048
```

Results:

No hosts found.

source : M. Wolfgang: Host Discovery with nmap

Machines actives

Autres options pour nmap

```
nmap -sP -PS 172.26.1.2
```

tcpdump Output:

```
10:48:13.656653 192.168.5.20.50992 > 172.26.1.2.http: S 3312451587:3312451587(0) win 2048
```

source : M. Wolfgang: Host Discovery with nmap

```
nmap -sP -PS25 172.26.1.2
```

tcpdump Output:

```
10:49:50.436438 192.168.5.20.63376 > 172.26.1.2.smtp: S 948961283:948961283(0) win 4096
```

source : M. Wolfgang: Host Discovery with nmap

Machines actives

Scénario 5 : pare-feux avec état et trafic provenant d'un seul port 53

```
pass from any to 172.26.1.2 proto tcp port 80 keep state
pass from any to 172.26.1.4 proto tcp port 53 keep state
pass from any to 172.26.1.6 proto tcp port 25 keep state
pass from any port 53 to any keep state
drop all
```

Our Command:

```
nmap -sP 172.26.1.0/29 -g 53
```

tcpdump Output:

```
10:52:02.083065 192.168.5.20 > 172.26.1.0: ICMP: echo request
10:52:02.083260 192.168.5.20.domain > 172.26.1.0.http: . ack 2177885259 win 3072
10:52:02.083301 192.168.5.20 > 172.26.1.1: ICMP: echo request
10:52:02.083346 192.168.5.20.domain > 172.26.1.1.http: . ack 2684323392 win 3072
10:52:02.083384 192.168.5.20 > 172.26.1.2: ICMP: echo request
10:52:02.083421 192.168.5.20.domain > 172.26.1.2.http: . ack 1438652920 win 3072
10:52:02.083459 192.168.5.20 > 172.26.1.3: ICMP: echo request
10:52:02.083496 192.168.5.20.domain > 172.26.1.3.http: . ack 771338950 win 3072
10:52:02.083534 192.168.5.20 > 172.26.1.4: ICMP: echo request
10:52:02.083570 192.168.5.20.domain > 172.26.1.4.http: . ack 3541039396 win 3072
10:52:02.083608 192.168.5.20 > 172.26.1.5: ICMP: echo request
10:52:02.083645 192.168.5.20.domain > 172.26.1.5.http: . ack 2586779353 win 3072
10:52:02.083683 192.168.5.20 > 172.26.1.6: ICMP: echo request
10:52:02.083719 192.168.5.20.domain > 172.26.1.6.http: . ack 45434507 win 3072
10:52:02.083757 192.168.5.20 > 172.26.1.7: ICMP: echo request
10:52:02.083794 192.168.5.20.domain > 172.26.1.7.http: . ack 1886752887 win 3072
10:52:02.084616 172.26.1.2.http > 192.168.5.20.domain: R 1438652920:1438652920(0) win 0 (DF)
10:52:02.084845 172.26.1.4.http > 192.168.5.20.domain: R 3541039396:3541039396(0) win 0 (DF)
10:52:02.085219 172.26.1.6.http > 192.168.5.20.domain: R 45434507:45434507(0) win 0 (DF)
```

source : M. Wolfgang: Host Discovery with nmap

Machines actives

Temps et masque avec nmap :

Our Command:

```
nmap -sP -PP 172.26.1.4
```

tcpdump Output:

```
13:32:05.780376 192.168.5.20 > 172.26.1.4: icmp: time stamp query id 47345 seq 0 (DF)  
13:32:05.781066 172.26.1.4 > 192.168.5.20: icmp: time stamp reply id 47345 seq 0 : org 0x0  
recv 0x3c339bd xmit 0x3c339bd
```

source : M. Wolfgang: Host Discovery with nmap

Our Command:

```
nmap -sP -PM 172.26.1.4
```

tcpdump Output:

```
13:37:11.452204 192.168.5.20 > 172.26.1.4: icmp: address mask request (DF)
```

source : M. Wolfgang: Host Discovery with nmap

Machines actives

Scénario 6 : Cibler le protocole 25 avec un SYN

Our command:

```
nmap -sP -PS25 172.26.1.0/29
```

tcpdump Output:

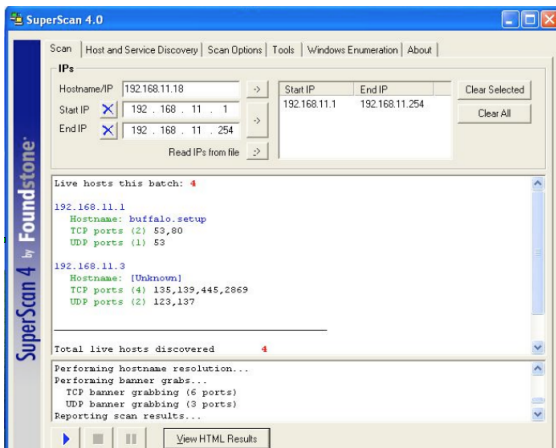
```
11:05:06.000617 192.168.5.20.38849 > 172.26.1.0.smtp: S 3544186883:3544186883(0) win 2048
11:05:06.000720 192.168.5.20.38849 > 172.26.1.1.smtp: S 4056940587:4056940587(0) win 2048
11:05:06.000759 192.168.5.20.38849 > 172.26.1.2.smtp: S 3272605779:3272605779(0) win 2048
11:05:06.000797 192.168.5.20.38849 > 172.26.1.3.smtp: S 4168614011:4168614011(0) win 2048
11:05:06.000835 192.168.5.20.38849 > 172.26.1.4.smtp: S 586154147:586154147(0) win 2048
11:05:06.000872 192.168.5.20.38849 > 172.26.1.5.smtp: S 3571974347:3571974347(0) win 2048
11:05:06.000910 192.168.5.20.38849 > 172.26.1.6.smtp: S 667418867:667418867(0) win 2048
11:05:06.000948 192.168.5.20.38849 > 172.26.1.7.smtp: S 902824219:902824219(0) win 2048
11:05:06.001909 172.26.1.6.smtp > 192.168.5.20.38849: S 203047548:203047548(0) ack
667418868 win 5840 <mss 1460> (DF)
```

source : M. Wolfgang: Host Discovery with nmap

Machines actives

SuperScan (windows)

- L'un des outils les plus rapide (parallèle)
- Il offre d'autres fonctionnalités telles que "port scanning", énumération, whois, etc.



Machines actives

icmpenum (Unix)

- ➔ Permet de détecter des machines actives via ICMP ECHO, ICMP TIMESTAMP REQUEST et ICMP INFO REQUEST
- ➔ L'exemple suivant énumère toute la classe C du réseau 192.168.1.0 en utilisant ICMP TIMESTAMP REQUEST

```
[root]$ icmpenum -i 2 -c 192.168.1.0  
192.168.1.3 is up  
192.168.1.25 is up  
192.168.1.215 is up
```

Machines actives

contre mesures

- ➡ Détection via des IDSs : Snort, Scanlog, Courtney, Ippl, Protolog
- ➡ Prévention via des ACLs (Firewalls) : bloquer tous les messages ICMP "inutiles". Exemples de règles :

```
access-list 101 deny icmp any any 13 ! timestamp request
access-list 101 deny icmp any any 17 ! address mask request
```


Services actifs

- ➔ Découvrir les ports TCP et UDP ouverts sur une cible.
- ➔ Les ports ouverts nous informent sur les services offerts sur la cible
- ➔ Ayant les services et le système d'exploitation installés sur la cible, nous regardons par la suite si la combinaison en question est connue par des vulnérabilités et comment les exploiter.
- ➔ **Le port scanning est illégal** : c'est comme frapper sur les portes et les fenêtres d'une maison pour la simple raison de voir s'il y a des personnes à l'intérieur.

Services actifs : techniques

Le scan "TCP connect"

- ➔ C'est une demande de connexion normale
- ➔ Ce type de scan est facile à détecter
- ➔ Si le port est fermé, on reçoit un RST/ACK, rien ou un ICMP (selon le système)

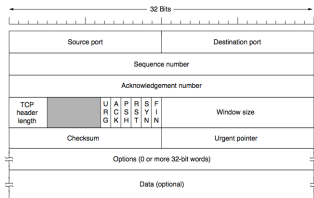


Fig. 6-29. The TCP header.

source : Livre A. Tanenbaum

Services actifs : techniques

Le scan "TCP syn"

- ➡ À la place d'envoyer un ACK, on envoie un RST
- ➡ On peut ne rien envoyer à la troisième étape, mais cela risque de faire un DoS
- ➡ Il serait non enregistré dans les fichiers log (un mode furtif). Mais avec un IDS en place, il peut soulever des soupçons (e.g. beaucoup de RST)

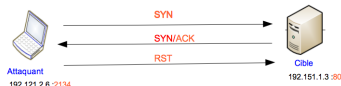
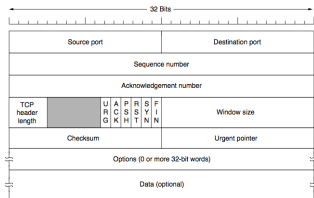


Fig. 6-29. The TCP header.

source : Livre A. Tanenbaum

Services actifs : techniques

Autres types

Quand les hackers ont réalisés que leurs SYN techniques sont souvent détectées, ils étaient obligés d'inventer d'autres *Stimulus/Réponses* non conventionnels. Les trois modes suivants passent généralement inaperçus (modes *Stealth*) pour un pare-feu qui est souvent configuré pour bloquer les SYN quand on veut empêcher une connexion à un port (Pas de SYN \Rightarrow pas de connexion \Rightarrow accès bloqué)

- ➡ TCP FIN : l'attaquant envoie un paquet FIN. La réponse attendue, selon RFC 793 (www.ietf.org/rfc/rfc0793.txt), est RST pour tous les ports fermés ou rien si le port est ouvert. Cette technique fonctionne généralement sous une cible à base de Unix
- ➡ TCP Xmas Tree (arbre de Noël) : l'attaquant envoie un paquet (FIN, URG et PUSH). La réponse attendue, selon RFC 793, est RST pour tous les ports fermés ou rien si le port est ouvert
- ➡ TCP Null : l'attaquant envoie un paquet où tous les drapeaux sont en bas. La réponse attendue, selon RFC 793, est RST pour tous les ports fermés ou rien si le port est ouvert

Remarque : certaines implantations de la pile TCP/IP ne respectent pas rigoureusement la recommandation ce qui peut fausser les conclusions du scan.

Services actifs : techniques

Autres types

- TCP ACK : l'attaquant envoie un paquet ACK. La réponse attendue, selon RFC 793, est RST indépendamment si le port est fermé ou il est ouvert. Mais, il pourrait donner des informations sur l'ACL et le type de pare-feu, surtout lorsqu'il est combiné avec un SYN.
 - Pas de réponse ou un ICMP de type 3 (destination port unreachable) \Rightarrow port filtré.
 - Un SYN retourne un SYN/ACK ou un RST et un ACK retourne un RST \Rightarrow port non filtré
 - Un SYN retourne un SYN/ACK, mais un ACK ne retourne pas de réponse \Rightarrow port filtré + pare-feu avec états
 - Un SYN ne retourne ni un SYN/ACK ni un RST, mais suivi par un ACK retourne un RST \Rightarrow port filtré
 - Si ni SYN ni ACK ne génèrent de réponse \Rightarrow port bloqué
- **Remarque :** Firewall est un outil plus spécialisé pour déterminer les règles de filtrage d'un pare-feu. Il détermine la distance n (via des TTL) au pare-feu, puis il envoie des datagramme ayant un $TTL=n+1$. S'il reçoit de réponses le port n'est pas filtré.

Services actifs : techniques

Autres types

- ➔ TCP Windows : l'attaquant envoie des ACKs à plusieurs ports et analyse le contenu du champ Windows dans les réponses. Des valeurs "anormales" peuvent impliquer que le port est ouvert. Par exemple, les réponses suivantes peuvent indiquer que le port 22 est ouvert.

```
host 132.164.1.2 port 20:  F:RST win:  0
host 132.164.1.2 port 21:  F:RST win:  0
host 132.164.1.2 port 22:  F:RST win: 512
host 132.164.1.2 port 23:  F:RST win:  0
```

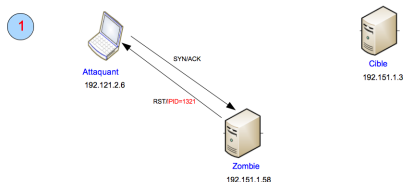
Remarque : Win indique la taille de données qu'on peut envoyer avant de recevoir un accusé de réception. Ce type de scan fonctionne seulement avec quelques systèmes comme AIX et FreeBSD.

Services actifs : techniques

Autres types

➡ TCP Idle (Zombie):

- Trouver une machine non active (idle) qui va jouer le rôle de zombie. Idéalement cette machine est de confiance pour les IDS et les pare-feux de la cible
- Déterminer le IPID (un champ dans l'entête IP pour gérer la fragmentation) du Zombie et vérifier qu'il est prédictible (e.g. $\text{next-IPID} = \text{IPID} + 1$)

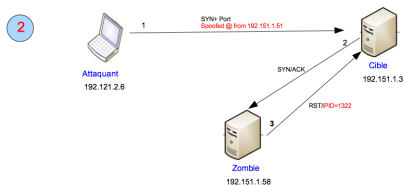


Services actifs : techniques

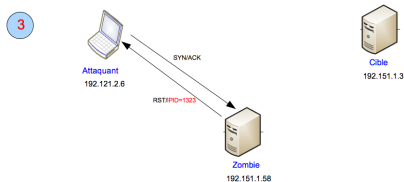
Autres types

➡ TCP Idle (Zombie):

- Envoyer un paquet (spoofed) à la cible



- Tester le IPID du zombie



Services actifs : techniques

Autres types

➡ FTP Bounce (rebond FTP):

- Le protocole FTP supporte les connexions par proxy : un utilisateur se connecte à un serveur FTP puis il lui demande d'envoyer des fichiers à autre serveur FTP.
- La machine qui fait la connexion est indépendante de la machine vers laquelle se fait le transfert de données.
- Cette fonctionnalité peut être détournée pour scanner une machine : on demande à un serveur FTP d'envoyer des fichiers à la cible et on analyse les réponses
- Elle a l'avantage de contourner le pare-feu de la passerelle externe
- La plupart de serveurs FTP ont cessés d'offrir cette fonctionnalité, mais certains serveurs l'offrent encore.

Services actifs : techniques

Autres types

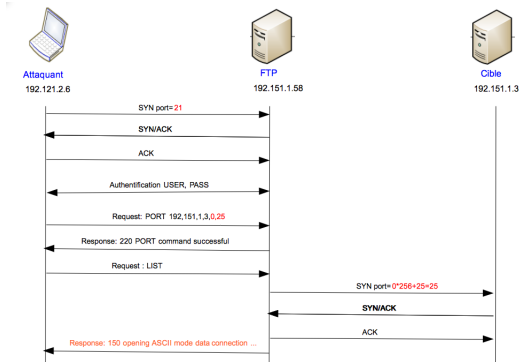
- FTP Bounce (rebond FTP): Deux commandes pour accomplir le travail
 - PORT a1,a2,a3,a4,p1,p2 : indique au serveur FTP que les prochaines commandes seront exécutées sur le port $p1*256+p2$ de la machine ayant comme adresse IP a1.a2.a3.a4
 - LIST : permet de lister le contenu du répertoire courant de la cible (elle peut être remplacée par d'autres commandes). C'est elle qui va déclencher l'ouverture de la connexion avec la cible.

Voir les figures suivantes

Services actifs : techniques

Autres types

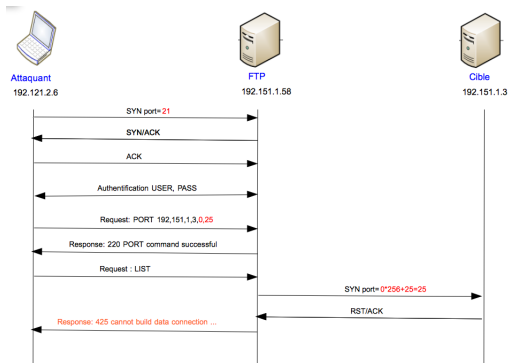
➡ FTP Bounce (rebond FTP): port ouvert



Services actifs : techniques

Autres types

➡ FTP Bounce (rebond FTP): port fermé



Services actifs : techniques

Autres types furtifs

- ➔ `namap decoys (-D)` : demander à `nmap` d'envoyer la même requête avec plusieurs adresses sources.
 - Une seule qui contient la bonne adresse et les autres sont des adresses spoofées.
 - Dans le fichier log de la cible, on trouve plusieurs machines qui ont demandé la même requête, ce qui complique l'investigation
- ➔ `namap -f` : fragmentation. Certains pare-feu ne rassemblent pas les fragments pour analyser le paquet
- ➔ `namap -P0` : demander à `nmap` de ne pas envoyer de message ICMP (ping)
- ➔ `namap -T0` : attendre au moins 5 minutes entre deux paquets.

Services actifs : techniques

Autres types

➡ UDP scan :

Reponse du stimulus	État assigné par nmap
Réponse reçu de la cible (rare)	open
Pas de réponses (après plusieurs transmissions)	open filtered
ICMP port non joignable (type 3, code 3)	closed
Autres ICMP non joignable (type 3, code 1, 2, 9, 10 et 13)	filtered

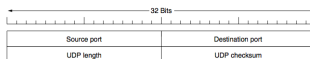


Fig. 6-23. The UDP header.

source : Livre A. Tanenbaum

Remarques :

- C'est la seule technique connue pour déterminer les ports UDP ouverts
- Plusieurs services importants (DHCP, DNS, SNMP, TFTP, etc) fonctionnent sur UDP

Services actifs : techniques

Autres types

• UDP scan : Très lent pour les raisons suivantes

- Les ports ouverts ne retournent aucune réponse. Combien doit-on attendre pour tirer la conclusion? (durée du *time-out*)
- S'il n'y a pas de réponses, le même stimulus est envoyé plusieurs fois pour distinguer entre absence de réponses et paquets perdus (UDP n'est pas fiable)
- Les ports fermés envoient ICMP port unreachable ICMP de type 3 et code 3), mais certains systèmes comme Linux2.4.20 limitent le nombre de ICMP, type 3, code 3 à 1 par seconde
- Le scan de 65 535 ports udp d'une machine qui limite les ICMP à 1/seconde prend 18h

Remarques : Comment accélérer

- Augmenter le nombre de machines à scanner en parallèle (`--min --hostgroup`)
- Très peu de ports UDP sont utilisés. On peut scanner juste les plus importants avec l'option `-F`
- Utiliser l'option `--host --timeout` pour ne pas trop attendre les réponses

Services actifs : outils

Strobe

- ➔ Rapide et fiable, mais il ne balaye que les ports de type TCP

```
[root] strobe 10.172.1.21
srobe 1.06 (c) 1999 Julian Assange (proff@suburbia.net).

10.172.1.21 echo          7/tcp Echo [95,JBP]
10.172.1.21 discard      9/tcp Discard [94,JBP]
10.172.1.21 sunrpc       111/tcp rpcbind SUN RPC
10.172.1.21 daytime      13/tcp Daytime [93,JBP]
10.172.1.21 chargen      19/tcp tftpd source
10.172.1.21 ftp          21/tcp File Transfer [Control] [96,JBP]
10.172.1.21 exec         512/tcp remote process execution;
10.172.1.21 login        513/tcp remote login a la telnet;
10.172.1.21 cmd          514/tcp shell like exec, but automatic
10.172.1.21 ssh          22/tcp Secure Shell
10.172.1.21 telnet       23/tcp Telnet [112,JBP]
10.172.1.21 smtp         25/tcp Simple Mail Transfer [102,JBP]
10.172.1.21 nfs          2049/tcp networked fi le system
10.172.1.21 lockd        4045/tcp
10.172.1.21 unknown     32772/tcp unassigned
10.172.1.21 unknown     32773/tcp unassigned
10.172.1.21 unknown     32778/tcp unassigned
10.172.1.21 unknown     32799/tcp unassigned
10.172.1.21 unknown     32804/tcp unassigned
```

source : <http://it.toolbox.com/wiki/index.php/Strobe>

udp_scan peut être utilisé pour compléter le travail de **strobe**

```
[root]$ udp_scan 192.168.1.1 1-1024
42 UNKNOWN
53 UNKNOWN
135 UNKNOWN
```


Services actifs : outils

netcat

- ➡ Très bon outil (couteau suisse) : Il peut faire différents types de scan (TCP, UDP, etc.)
- ➡ Quelques options utiles :
 - -v : affiche les ports ouverts (par défaut il ne les affiche pas)
 - -z : terminer le scan sans s'arrêter (par défaut il s'arrête à chaque fois qu'il trouve un port ouvert pour permettre d'envoyer des données à ce port)
 - -u : pour un scan UDP
 - -w n : n est le nombre maximum de secondes que netcat peut passer par port.

Services actifs : outils

netcat

➔ TCP scan :

```
[root]$ nc -v -z -w2 192.168.1.1 1-1024  
[192.168.1.1] 135 (?) open  
[192.168.1.1] 80 (http) open  
[192.168.1.1] 25 (smtp) open
```

➔ UDP scan :

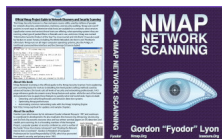
```
[root]$ nc -u -v -z -w2 192.168.1.1 1-1024  
[192.168.1.1] 123 (ntp) open  
[192.168.1.1] 53 (domain) open  
[192.168.1.1] 42 (name) open
```

- ➔ Par défaut, il parcourt les ports dans l'ordre décroissant. L'option (-r) permet de le faire dans l'ordre croissant.

Services actifs : outils

Network Mapper (nmap)

- ➡ Un excellent outil: c'est une référence (à connaître!)
- ➡ Permet une large variété de scan (TCP, UDP, etc.)
- ➡ License GPL
- ➡ Disponible sur la plupart des systèmes d'exploitation
- ➡ Livre de référence :



source : <http://nmap.org/book/>

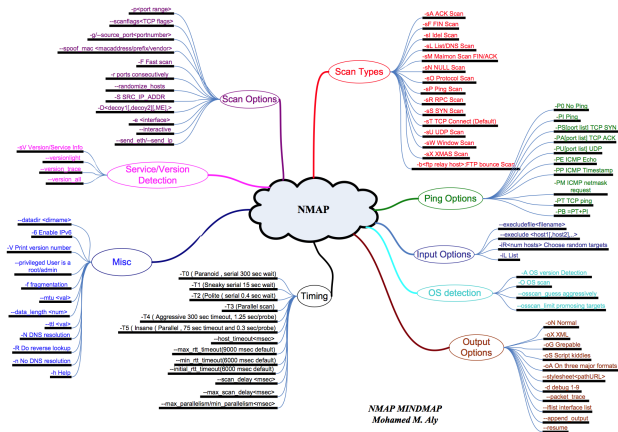
Services actifs : outils

Network Mapper (nmap)

➡ Quelques options :

type de scan	commande	l'utilisateur a besoin de privilège	Identifie les ports tcp	identifie les ports udp
TCP syn	-sS	oui	oui	non
TCP connect	-sT	non	oui	non
TCP FIN	-sF	oui	oui	non
TCP Xmas	-sX	oui	oui	non
TCP Null	-sN	oui	oui	non
TCP ACK	-sA	oui	oui	non
TCP Windows	-sW	oui	oui	non
TCP RPC	-sR	non	non	non
Idle	<zombie host [:probport]>	oui	oui	non
Rebond FTP	-b<ftp relay host>	non	oui	non
UDP scan	-sU	oui	non	oui

Network Mapper (nmap)



source : <http://nmap.org/docs/nmap-mindmap.pdf>

Services actifs : outils

Network Mapper (nmap)

➡ Exemple :

```
[root]$ nmap 192.168.1.1
Starting nmap V. 4.68 by fyodor@insecure.org (www.insecure.org/nmap)
Interesting ports on (192.168.1.1):
(The 1504 ports scanned but not shown below are in state:closed)
  Port      State    Portocol   Service
  --      -
  21        open     tcp        ftp
  23        open     tcp        telnet
  25        open     tcp        smtp
  79        open     tcp        finger
  80        open     tcp        http
  98        open     tcp        linuxconf
  110       open     tcp        pop-3
  111       open     tcp        sunrpc
  113       open     tcp        auth
  135       open     tcp        lloc-srv
  139       open     tcp        netbios-ssn
  443       open     tcp        https
```

Services actifs : outils

Network Mapper (nmap)

Remarques :

- Par défaut nmap ne fait pas le scan de tous les ports : juste les ports les plus connus (les ports dans le fichier *nmap-services* contenant 2200 *tcp* et *udp* ports)
- Pour spécifier tous les ports, on utilise l'option "`--allports`" ou `-p 0-65535`
- On peut spécifier un port par un nom ou par son numéro : `nmap -sS http, 443`
- Un scan UDP prend beaucoup plus de temps qu'un scan TCP, et ce, même pour les ports connus
- Un scan UDP retourne souvent "open|filtered" puisque les services UDP ne répondent pas toujours suite à la réception UDP.
- L'option `-sUV` permet d'avoir une réponse plus précise sur les port UDP : nmap essaye dans ce cas d'envoyer, selon le service, de requêtes UDP qui produisent des réponses

Services actifs : outils

Network Mapper (nmap)

➡ Fonctionnalités avancées (script) :

- nmap a un NSE (Nmap Scripting Engine) qui lui donne plusieurs fonctionnalités avancées (scan de vulnérabilités, découverte avancée de réseaux, détection de portes dérobées et même la réalisation de certains exploits).
- NSE répartit les scripts en catégories : auth, broadcast, brute, default, discovery, dos, exploit, external, fuzzer, intrusive, malware, safe, version, vuln.
- Chaque catégorie contient plusieurs scripts
- On peut invoquer un script ou toute les scripts d'une catégorie
- Pour plus de détails : <http://nmap.org/nsedoc/>
- Avoir des détails sur les ports (surtout les ports inhabituels) via leurs bannières

```
nmap --script banner 192.168.1.110
```

- Tous les scriptes de la catégorie vuln à la recherche de vulnérabilités connues

```
nmap --script vuln 192.168.1.110
```

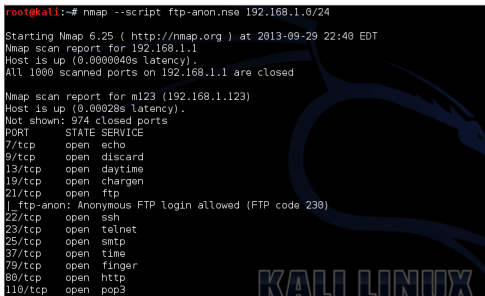

Services actifs : outils

Network Mapper (nmap)

➡ Fonctionnalités avancées (script) :

- trouver les serveurs permettant une connexion ftp anonyme

`nmap --script ftp-anon.nse`



```
root@kali:~# nmap --script ftp-anon.nse 192.168.1.0/24

Starting Nmap 6.25 ( http://nmap.org ) at 2013-09-29 22:40 EDT
Nmap scan report for 192.168.1.1
Host is up (0.0000040s latency).
All 1000 scanned ports on 192.168.1.1 are closed

Nmap scan report for m123 (192.168.1.123)
Host is up (0.00028s latency).
Not shown: 974 closed ports
PORT      STATE SERVICE
7/tcp     open  echo
9/tcp     open  discard
13/tcp    open  daytime
19/tcp    open  chargen
21/tcp    open  ftp
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
37/tcp    open  time
79/tcp    open  finger
80/tcp    open  http
110/tcp   open  pop3
```

- ▶ trouver les serveurs permettant les rebonds ftp

`nmap --script ftp-bounce.nse`

Services actifs : outils

Network Mapper (nmap)

•➔ Fonctionnalités avancées (script)

- trouver les serveurs web permettant de traverser des répertoires pour aller dans /etc/passwd

```
nmap --script http-passwd.nse
```

- voir si un serveur DNS permet la résolution récursive

```
nmap --script dns-recursion.nse
```

- voir si un serveur DNS permet un "zone transfert"

```
nmap --script dns-transfer.nse
```

- voir si on peut se connecter à un serveur POP avec un nom d'utilisateur et mot de passe choisis d'un dictionnaire

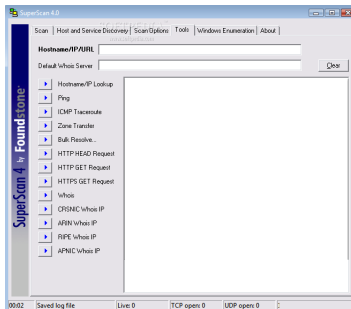
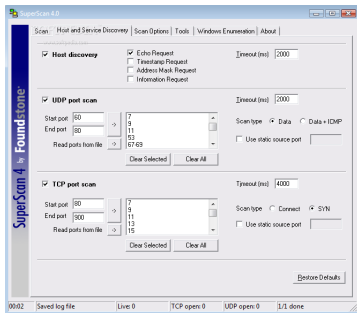
```
nmap --script pop3-brute.nse
```

- voir si on peut se connecter à un serveur SNMP avec un nom d'utilisateur et mot de passe choisis d'un dictionnaire

```
nmap --script snmp-brute.nse
```

Services actifs : outils

SuperScan (application windows) : une application qui offre de nombreuses fonctionnalités



Services actifs : outils

ScanLine

- ➡ Un autre excellent outil: très recommandé
- ➡ Permet une large variété de scan (TCP, UDP, etc.)
- ➡ Très rapide
- ➡ Via une seule commande, il peut faire des scans TCP et UDP

```
sl -t 80,81,88,8000,8080 -u 31337 10.0.2.2-20 -O out.txt
```

Services actifs : outils

ScanLine : Syntax

```
sl [-?bhijnprsTUvz] [-cdgmq <n>] [-flLo0 <file>] [-tu <n>[,<n>-<n>]] IP[,IP-IP]
-? - Shows this help text
-b - Get port banners
-c - Timeout for TCP and UDP attempts (ms). Default is 4000
-d - Delay between scans (ms). Default is 0
-f - Read IPs from file. Use "stdin" for stdin
-g - Bind to given local port
-h - Hide results for systems with no open ports
-i - For ping using ICMP Timestamp Requests in addition to Echo Requests
-j - Don't output "---..." separator between IPs
-l - Read TCP ports from file
-L - Read UDP ports from file
-m - Bind to given local interface IP
-n - No port scanning - only ping (unless you use -p)
-o - Output file (overwrite)
-O - Output file (append)
-p - Do not ping hosts before scanning
-q - Timeout for pings (ms). Default is 2000
-r - Resolve IP addresses to hostnames
-s - Output in comma separated format (csv)
-t - TCP port(s) to scan (a comma separated list of ports/ranges)
-T - Use internal list of TCP ports
-u - UDP port(s) to scan (a comma separated list of ports/ranges)
-U - Use internal list of UDP ports
-v - Verbose mode
-z - Randomize IP and port scan order
```

Ports actifs

contre mesures

- Détection via des IDSs : SNORT, Scanlog, Etc. Exemple de log (alerte) d'un scan de ports
- **fail2ban** : analyse le fichier log à la recherche de comportements suspects (plusieurs tentatives de mots de passe erronés, etc.). Il peut ensuite demander au pare-feu de bloquer une adresse.
- **port-knocking** (utilisé souvent avec ssh comme un "code" supplémentaire) : Pour pouvoir accéder à un service, il faut préalablement envoyer des demandes de connexion sur certains ports dans un ordre précis. Un outil analyse le log de pare-feu, établit la liste des actions de chaque machine externe et si une séquence est présente (exemple SYN vers les ports 100, 1000 et 3000), une règle est ajoutée pour ouvrir un port donné (exemple ssh)
- Prévention : la prévention de scans des ports est difficile. On peut juste réduire la surface ciblée en désactivant tous les services inutiles. Pour Unix, il suffit de mettre en commentaire ces services dans le fichier `/etc/inetd.conf`. Pour Windows : voir Panneau de configuration | Services

Détection de systèmes d'exploitation

Analyse de bannières

- ➔ Protocoles de réseau
- ➔ Analyse de bannières
- ➔ Prise d'empreintes

Détection de systèmes d'exploitation

Protocoles de réseau

•→ SNMP :

- La variable Object OID dans MIB
- Peut correspondre à un système d'exploitation, à équipement (commutateur, routeur, etc.), etc.

•→ DNS : champ de type HINFO

- NetBIOS : NULL Session permet de récupérer des informations sur les systèmes sans fournir des mots de passe

Détection de systèmes d'exploitation

Analyse de bannières

- ➡ Plusieurs systèmes annoncent ce qu'ils sont quand ils répondent à des requêtes (FTP, TELNET, SMTP, HTTP, POP, etc.).
- ➡ Il suffit de ramasser ces informations
- ➡ Mais, attention, elles peuvent être spoofées.
- ➡ Cette technique sera présentée plus en détail dans le chapitre suivant.

```
HTTP/1.1 200 OK
Date: Mon, 23 November 2009 21:17:14 EST
Server: Apache/2.0.46 (Unix) (Red Hat/Linux)

Content-Type: text/html CHARSET=ISO-8859-1
```

Détection de systèmes d'exploitation

Prise d'empreintes : Analyse active des empreintes de la pile TCP/IP : les détails d'implantation de la pile TCP/IP **varient** d'un système d'exploitation à un autre

➡ Sonde FIN :

- Pour terminer normalement une connexion : on envoie un segment FIN/ACK et on reçoit un ACK.
- Un segment qui contient seulement le drapeau FIN mis à 1 est anormal
- RFC 793 stipule qu'il ne faut pas répondre aux segments FIN
- Certains systèmes (Windows NT/ 200X/Vista, etc.) sont "hors la loi" et répondent par FIN/ACK

Détection de systèmes d'exploitation

Prise d'empreintes

• Drapeau SYN + Bugs :

- Les systèmes répondent différemment aux segments SYN mal construits (e.g. SYN/FIN, SYN/RST, etc.)
- Linux, par exemple, répond avec un SYN.

• Analyse de numéros des séquences initiaux (ISN) :

- Chaque segment est identifié par numéro de séquence
- Le numéro du premier segment de la connexion est généré aléatoirement"; le reste c'est de l'incrémentement
- L'idée est de ramasser les n premiers numéros de séquences de n connexions (ou tentatives de connexions) et de les analyser
- Plusieurs versions d'Unix génèrent ces nombres d'une manière croissante
- Linux 2.0 : c'est complètement aléatoire
- Windows : c'est des incrémentations avec des petits pas

Détection de systèmes d'exploitation

Prise d'empreintes

•➔ Bit de fragmentation (entête IP) :

- Certains systèmes mettent à un le bit DF (Dont Fragment) pour augmenter la performance
- Ce bit peut être monitoré pour révéler le système d'exploitation

•➔ Taille initiale de la fenêtre TCP :

- Cette taille est unique pour certains systèmes
- Donc, il suffit de la monitorer

•➔ Acquitement de messages :

- Pour acquitter un message certains systèmes retournent leurs numéros de séquences dans les segments ACK
- D'autres retournent leurs numéros de séquences + 1
- Donc, il suffit de les monitorer

•➔ Autres : Contenu de certains champs dans les protocoles IP, TCP, ICMP, etc., peuvent varier d'un système à un autre.

Détection de systèmes d'exploitation

Prise d'empreintes : nmap avec l'option (-O) utilise la plupart de ces techniques pour deviner un système d'exploitation

➡ Exemple :

```
[root]$ nmap -O 192.168.1.1
Starting nmap V. 4.68 by fyodor@insecure.org (www.insecure.org/nmap)
Interesting ports on (192.168.1.1):
(The 1504 ports scanned but not shown below are in state:closed)
  Port      State    Portocol   Service
  21        open     tcp        ftp
  23        open     tcp        telnet
  25        open     tcp        smtp
  111       open     tcp        sunrpc
TCP Sequence Prediction :   Class=random positive increments
                           Difficulty=26590 (worthy challeng)
Remote operating system guess: Solaris 2.5, 2.51
```

Détection de systèmes d'exploitation

Prise d'empreintes

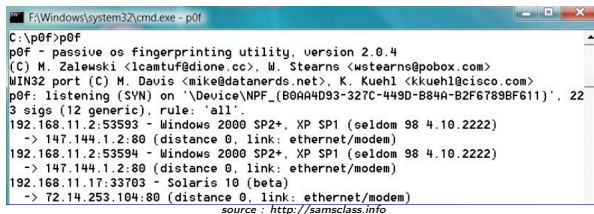
- ➡ Les signatures des systèmes d'exploitation auxquelles `nmap` se réfère se trouvent dans un fichier appelé *`nmap-os-fingerprint`*
- ➡ Des centaines de signatures se trouvent déjà dans ce fichier et la liste est mise à jour par chaque nouvelle version
- ➡ `queso` est un autre outil, plus vieux que `nmap`.
- ➡ `queso` n'est pas un scanneur de ports, il ne fait que la détection des OS
- ➡ `queso` travail sur un seul port (par défaut c'est 80) dont la valeur est 25 dans l'exemple suivant

```
[root]$ queso 192.168.1.3:25  
192.168.1.3  * Windows 95/98/NT
```

Détection de systèmes d'exploitation

La détection de systèmes d'exploitation peut se faire d'une manière passive

- ➡ Capturer le trafic avec Snort par exemple
- ➡ Analyser le contenu de certain champs (TTL, Windows, DF, etc.)
- ➡ *Siphon* est le premier outil qui a fait ce travail (il est désuet)
- ➡ *p0f* est un outil plus récent qui fonctionne sous Windows
- ➡ Il peut deviner tous les systèmes qu'il voit sur le LAN



```
F:\Windows\system32\cmd.exe - p0f
C:\p0f>p0f
p0f - passive os fingerprinting utility, version 2.0.4
(C) M. Zalewski <lcamtuf@disone.cc>, W. Stearns <wstearns@pobox.com>
WIN32 port (C) M. Davis <mike@datanerds.net>, K. Kuehl <kkuehl@cisco.com>
p0f: listening (SVN) on '\\Device\NPF_{B0AA4D93-327C-449D-B84A-B2F6789BF611}', 22
3 sigs (12 generic), rule: 'all'.
192.168.11.2:53593 - Windows 2000 SP2+, XP SP1 (seldom 98 4.10.2222)
-> 147.144.1.2:80 (distance 0, link: ethernet/modem)
192.168.11.2:53594 - Windows 2000 SP2+, XP SP1 (seldom 98 4.10.2222)
-> 147.144.1.2:80 (distance 0, link: ethernet/modem)
192.168.11.17:33703 - Solaris 10 (beta)
-> 72.14.253.104:80 (distance 0, link: ethernet/modem)
source : http://samsclass.info
```

Cheops : Un autre outil automatique (tout-en-un)

- ➔ De nombreux autres outils sont disponibles. *Cheops* (se prononce kee-ops) est parmi les plus connus
- ➔ C'est un outil graphique simple permettant une analyse rapide des résultats
- ➔ Le même package intègre ping, traceroute, scannage de ports et des systèmes d'exploitation via *queso*

