

Port 0 OS FingerPrint.

Table of contents

1. Introduction.
2. Qu'est-ce que le port 0?
3. Port 0 et la détection de systèmes d'exploitation.
4. iptables et le port 0.
5. Conclusion.

1. Introduction.

Depuis quelques temps, de plus en plus d'outils de détection de système d'exploitation à distance (actifs) sont développés, nmap, queso, ring et xprobe sont surement les plus connus et les plus utilisés. Pour détecter un OS (système d'exploitation) à distance, ils utilisent chacun leurs propres techniques. Queso et nmap envoient une série de paquets malformés et déterminent l'OS en fonction de leur base d'empreintes et des réponses reçues, ring se base sur les temps de retransmission d'un paquet tcp (retransmission timeout lengths) et xprobe utilise lui le protocole ICMP. De nouvelles méthodes de détection d'OS sont trouvés régulièrement et notamment celle du "PORT 0 OS FingerPrint" [1] qui est expliqué dans cet article.

2. Qu'est-ce que le port 0?

Chaque application se voit attribuer une adresse unique sur la machine, codée sur 16 bits non signé: un port, ce qui fait un total de 65536 ports utilisables (de 0 à 65535). Tout comme les adresses IP, les ports ont été regroupés et classés. Ainsi les ports de 0 à 1023 sont les ports reconnus ou réservés (Well Known Ports), ils sont attribués par l'IANA (Internet Assigned Numbers Authority). Les ports 1024 à 49151 sont appelés ports enregistrés (Registered Ports). Les ports 49152 à 65535 sont les ports dynamiques ou privés. Parmi tous ces ports, un seul est vraiment particulier, il s'agit du port 0. Ce port est utilisé par les programmeurs. En effet lorsque l'on assigne le port (source ou destination) à 0, le noyau se doit de modifier le port par un port de type enregistrés (Registered Ports). Un petit exemple avec ce programme qui normalement bind un shell sur le port 0 (sin_port = 0).

```
[...]  
int main(void)  
{  
    int fd,dup;  
    struct sockaddr_in yeah;  
  
    fd=socket(AF_INET,SOCK_STREAM,IPPROTO_IP);  
    yeah.sin_family=AF_INET;  
    yeah.sin_addr.s_addr=INADDR_ANY;  
    yeah.sin_port=0;  
  
    bind(fd,(struct sockaddr *)&sin,sizeof(yeah));  
    listen(fd,1);  
    dup=accept(fd,0,0);  
    dup2(dup,0);  
    dup2(dup,1);  
    dup2(dup,2);  
    execl("/bin/sh","sh",0);  
}
```

Après compilation et exécution, on remarque grâce à l'utilitaire netstat que le shell n'est pas bindé sur

le port 0 mais sur un port compris en 1024 à 49151. Le noyau a donc bien changer le port, le port 0 est utile aux programmeurs mais aussi aux outils de détection d'OS.

3. Port 0 et la détection de systèmes d'exploitation.

Les différentes RFCs ne spécifient pas de comportement pour un système d'exploitation lors de la réception d'un paquet sur le port 0. Ainsi on retrouve différent comportement suivant le système d'exploitation lorsqu'il reçoit un paquet sur le port 0. Il ne reste plus qu'à déterminer ces différents comportements et les inscrire dans une base d'empreintes. L'équipe Network Penetration s'est penché sur le sujet et a réalisé les tests sur 7 OS. Ces tests consistaient à envoyé 7 différents paquets (tcp, udp) avec un port de destination de 0 et/ou un port source de 0:

```
P1: paquet tcp (flag=S) (src-port: 0 - dst-port: 0)
P2: paquet tcp (flag=S) (src-port: x - dst-port: 0)
P3: paquet tcp (flag=S) (src-port: 0 - dst-port: port-ouvert)
P4: paquet tcp (src-port: 0 - dst-port: port-fermé)
P5: paquet udp (src-port: 0 - dst-port: 0)
P6: paquet udp (src-port: 53 - dst-port: 0)
P7: paquet udp (src-port: 0 - dst-port: port-fermé)
```

Pour le paquet P2, x signifie un port différent de 0. Le port source choisit (53 - dns) pour le paquet P6 est pour passer outre certains firewalls.

On peut mettre en oeuvre ces tests avec l'outil hping [2]. Sur une machine tournant sous linux (slackware) avec un noyau de la série 2.4.x ça donne:

```
# hping -S localhost -s 0 -p 0 -c 1 // P1
HPING localhost (lo 127.0.0.1): S set, 40 headers + 0 data bytes
len=40 ip=127.0.0.1 ttl=255 id=0 sport=0 flags=RA seq=0 win=0 rtt=0.2 ms

--- localhost hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss

# hping -S localhost -p 0 -c 2 // P2
HPING localhost (lo 127.0.0.1): S set, 40 headers + 0 data bytes
len=40 ip=127.0.0.1 ttl=255 DF id=0 sport=0 flags=RA seq=0 win=0 rtt=0.2 ms
len=40 ip=127.0.0.1 ttl=255 DF id=0 sport=0 flags=RA seq=1 win=0 rtt=0.4 ms

--- localhost hping statistic ---
2 packets tramitted, 2 packets received, 0% packet loss

# hping -S localhost -p 80 -s 0 -c 1 // P3
HPING localhost (lo 127.0.0.1): S set, 40 headers + 0 data bytes
len=44 ip=127.0.0.1 ttl=255 DF id=0 sport=80 flags=SA seq=0 win=32767 rtt=0.3 ms

--- localhost hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.3/0.3/0.3 ms

# hping -S localhost -p 1337 -s 0 -c 1 // P4
HPING localhost (lo 127.0.0.1): udp mode set, 40 headers + 0 data bytes
len=40 ip=127.0.0.1 ttl=255 DF id=0 sport=1337 flags=RA seq=0 win=0 rtt=0.2 ms

--- localhost hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

# hping -2 localhost -p 0 -s 0 -c 1 // P5
HPING localhost (lo 127.0.0.1): udp mode set, 28 headers + 0 data bytes
```

```

ICMP Port Unreachable from ip=127.0.0.1 name=localhost

--- localhost hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

# hping -2 localhost -p 0 -s 53 -c 1 // P6
HPING localhost (lo 127.0.0.1): udp mode set, 28 headers + 0 data bytes
ICMP Port Unreachable from ip=127.0.0.1 name=localhost

--- localhost hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

# hping -2 localhost -p 1337 -s 0 -c 1 // P7
HPING localhost (lo 127.0.0.1): udp mode set, 28 headers + 0 data bytes
ICMP Port Unreachable from ip=127.0.0.1 name=localhost

--- localhost hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

```

On peut remarquer ainsi qu'une machine tournant sous linux répond par des paquets avec les flags RST-ACK aux tests P1, P2 et P4, SYN-ACK au test P3 et par des paquets ICMP Port_Unreachable aux tests P5, P6 et P7. Je n'ai pas pu réaliser ces tests sur d'autres machines. D'après les tests réalisés par NetworkPenetration [3], une machine tournant sous linux ou windows(tm) répond de la même manière comme on vient le voir. Par contre, OpenBSB ne réagit pas de la même manière, en effet il ne répond pas aux paquets P3, P5 et P6. On ne peut donc pas différencier une machine linux/windows mais on peut identifier une machine tournant sous OpenBSD, ce que fait une partie du programme gobbler [4].

4. Iptables et le port 0.

Ainsi avec ces quelques informations, on peut faire passer une machine sous linux comme tournant sous OpenBSD avec iptables et les règles suivantes:

```

# P3
iptables -A INPUT -p tcp --sport 0 -j DROP
# P5
iptables -A INPUT -p udp --sport 0 --dport 0 -j DROP
# P6
iptables -A INPUT -p udp --sport 53 --dport 0 -j DROP

```

5. Conclusion.

La détection d'OS par le port 0 est une méthode peu fiable face aux méthodes de détection 'classiques' utilisées par nmap ou queso. Elle peut cependant être utilisée en complément d'autres méthodes de détection d'OS.

6. Références.

- [1] - "Port 0 OS Fingerprinting" <http://www.securiteam.com/securityreviews/5XP0Q2AAKS.html>
- [2] - hping2 <http://www.hping.org>
- [3] - "NetwordPenetration" <http://www.networkpenetration.com>
- [4] - "The Gobbler" <http://gobbler.sourceforge.net/>