Wiki Enseignement

admin:tp3

TP3 (Scapy et Divers Trucs)

Scapy

Un peu de Scapy avec l'interpréteur de commande en langage Pyhthon. Lancez la commande “scapy” dans un shell root.

Voici un exmple pour construire un message de type ICMP/IP Echo-Request (ping). La fonction show() montre le contenu des headers & payloads pour ce message.

```
$ scapy
Welcome to Scapy (2.2.0)
>>> ping = IP(dst='192.168.0.2')/ICMP()
>>> ping.show()
###[ IP ]###
  version= 4
  ihl= None
  tos= 0x0
  len= None
  id= 1
  flags=
  frag= 0
  ttl= 64
  proto= icmp
  chksum= None
  src= 192.168.0.1
  dst= 192.168.0.2
  \options\
###[ ICMP ]###
     type= echo-request                          # valeur par défaut (ping)
     code= 0
     chksum= None
     id= 0x0
     seq= 0x0
>>>
```

Même chose avec un message TCP/IP à destination de 192.168.0.1 (port 80) avec le flag TCP SYN (demande d'ouverture de connexion). La fonction sr1() permet d'effectuer un envoi/réception…

```
>>> x=IP(dst="192.168.0.1")/TCP(flags="S",dport=80)    # on forge un paquet TCP/IP
>>> x.show()
>>> y = sr1(x)                                        # send/receive
>>> y.show()
```

Voici un exemple de script *Scapy* pour faire un traceroute.

traceroute.py

```
#!/usr/bin/env python

import sys
from scapy.all import *

def mytraceroute(target,maxttl):
  for x in range(maxttl):
    rsp = sr1(IP(dst=target, ttl=x)/ICMP(),verbose=0)
    if rsp.getlayer(ICMP).type==11 and rsp.getlayer(ICMP).code==0:
      print rsp.src
```

```
mytraceroute("10.0.0.2",8)
```

Un autre script *Scapy* pour faire un Syn Scan.

synscan.py

```
#!/usr/bin/env python

import sys
from scapy.all import *

def synscan(host):
   ports = range(1000)
   ip = IP(dst=host)
   tcp = TCP(dport=ports,flags="S")
   ans,unans = sr(ip/tcp)
   for sent,rcvd in ans:
      if rcvd.haslayer(TCP):
         # if rcvd.getlayer(TCP).flags & 2:
         if rcvd.sprintf("%TCP.flags%") == 'SA':
            print sent.dport

synscan("192.168.0.1")
```

Pour aller un peu plus loin : http://www.secdev.org/projects/scapy/doc/usage.html [http://www.secdev.org/projects/scapy/doc/usage.html]

Une autre version avec un XMAS scan…

xmasscan.py

```
#!/bin/python

from scapy.all import *

ports = range(1,1000)
ipdst = "10.0.0.2"

for i in ports:
        pkt=IP(dst=ipdst)/TCP(dport=i, flags="PUF")
        a = sr1(pkt, verbose=False, timeout=0.1)
        if a is None:
                print '{} is open'.format(i)
```

Pour finir avec la connexion TCP/IP, c'est ici : https://samsclass.info/124/proj11/proj18-scapy-tcp.html [https://samsclass.info/124/proj11/proj18-scapy-tcp.html]

Telnet

Ouverture d'une connexion TCP/IP avec Telnet vers un serveur web sur immortal.

```
$ telnet @immortal 80
  GET / HTTP/1.0  # enter
                  # enter again

  # la réponse HTTP du serveur web...
  HTTP/1.1 200 OK
  ...
  ...
```

Pour sortir d'un Telnet, tapez “Ctrl-]”.

HTTP Connect

Considérons une machine A qui souhaite ouvrir une session telnet (port 23) sur une machine C en utilisant une connexion “rebond” sur un serveur web B (configuré pour autoriser le mode *proxy*).

```
A$ telnet @B 80
  CONNECT @C:23 HTTP/1.0  # enter
                 # enter again
  C$login: xxxxxx
  C$password:  xxxxxxx
  C$ ...
```

Attaque Man-in-the-Middle

Dans un LAN, *syl* souhaite intercepter les échanges entre *nile* et *immortal*.

```
syl$ arpspoof -i eth0 -t @immortal @nile  &> /dev/null &
syl$ arpspoof -i eth0 -t @nile @immortal &> /dev/null &
syl$ echo 1 > /proc/sys/net/ipv4/ip_forward                # on active le routage
syl$ tcpdump -i eth0 not arp                               # on ignore l'ARP
syl$ pkill -9 arpspoof                                     # on fait le ménage à la fin !!!
```

Un peu de nmap

Le principe est relativement simple. Si l'on envoit en paquet TCP/IP avec le flag SYN (demande d'ouverture de connexion) sur un port donné, alors si un service écoute sur ce port, il nous répond SYN/ACK pour accépter la connexion. On en déduit que le port est ouvert. En revanche, si aucun serveur n'écoute sur ce port, le paquet est ignoré par le noyau. Autre cas possible. Si l'on envoi un paquet invalide, c'est-à-dire un paquet en dehors de toute connexion et sans flag SYN (ex. NULL, XMAS, …), alors ce paquet est ignoré si un service est présent. En revanche, s'il n'y a pas de service à l'acoute sur le port cible, alors le noyau répond RST (reset). Ce qui nous permet de déduire que le port est effectivement fermé. A l'inverse, les autres ports sont supposés ouverts (ou filtrés).

En résumé, voici les réponses dans les différents cas de figure :

	Envoi paquet avec SYN	**Envoi paquet invalide sans SYN**
Service à l'écoute	SYN/ACK	ignoré
Pas de service(*)	ignoré	RST

(*) Réponse du noyau.

Quelques exemples de *scan* simples :

```
#ping scan
$ nmap -sP 192.168.0.1-10

#syn scan
$ nmap -sS -p 21,22,80 192.168.0.1

#xmas scan
$ nmap -sX -p 4000-10000 192.168.0.1

#traceroute
$ nmap --traceroute 192.168.0.1
$ traceroute -T 192.168.0.1      # option -T pour faire du TCP plutôt que de l'UDP
```

Nota Bene : Si le nmap est trop lent dans les VMs, vous pouvez le rendre plus agressif avec les options ”-n -T5”.

Pour aller plus loin : https://nmap.org/book/man-port-scanning-techniques.html [https://nmap.org/book/man-port-scanning-techniques.html]

admin/tp3.txt · Last modified: 2016/09/22 21:44 by orel