# Security Testing and Cyber-Security: Use Case Simulated Man in The Middle Attack and Prevention

## Problem:

## Solution:

Tools are used in Man-In-The\_Middle Attack:

### 1. MAC Address Changer

Media Access Control (MAC Address) is a permanent physical and unique address assigned to network interfaces by the device manufacturer. Whether the target has a wireless card or an Ethernet card, each of these network cards comes with a unique address to this card. Therefore, there are no two devices within the target’s network(or in the world) that would have the same Mac address. This address will always be the same for this specific device even if the target unplugs it from one computer and connect it to another.

While the Mac address is used within the network to identify devices and transfer data between devices, the IP address is used on the Internet to identify computers and communicate between devices on the Internet. Therefore, each piece of data or packet that is sent within the network contains a source Mac and a destination Mac. This packet would flow from the source Mac to the destination Mac because this is how it identifies devices within its network.

Mac address is often used by filters to prevent or allow devices to connect to networks and do specific tasks on the network. Therefore, changing the target’s MAC address to another device's MAC address will allow hackers to impersonate this device and do things they are not allowed to do. Also, they'll be able to bypass filters or connect to networks that only specific devices with specific Mac addresses can connect to, or hide their identity.

Technology:

A basic MAC Address Changer can be created by Python and Linux commands. Depending on the hacking tools and interface that the hacker is going to use, the required packages are different. Since I am using Kali Linux and Python, the required Python packages are ” subprocess”, ”optparse”, and “re”.

```

def mac\_changer(interface, new\_mac):  
 print("[+] Changing MAC Address for " + interface + " to " + new\_mac)  
 subprocess.call(["ifconfig", interface, "down"]) # disable the MAC address  
 subprocess.call(["ifconfig", interface, "hw", "ether", new\_mac])  
 subprocess.call(["ifconfig", interface, "up"]) # enable the mac Address

```

### 2. Network Scanner

Information gathering is one of the most important steps when it comes to hacking or penetration testing. Hackers can't gain access to a system if they don't have enough information about it. For our attack, the plan is to hack into a private network, and one or more of the devices connected to this network is our target. For us to hack into the target, first we need to discover all the connected clients to this network, get their MAC addresses, and their IP addresses. Then, try to gather more information or run some attacks to gain access to our target.

There are several programs (Net discover or Nmap) that do this job well. Moreover, Python also has a Nmap package that was rebuilt and can be used similar to the Nmap application. However, I decided to write my network scanner to enable automation and more source code control. My Network Scanner is very similar to Net Discover (a scanner that comes with Kali Linux) that shows us all the connected devices within the same network (plus their IP addresses and their Mac address).

As we receive the returned data (IP addresses and Mac addresses) from the scan, we can use its data to gather information on the next attack. Measuring the security level of each device gives us a better success rate.

Technology:

A basic Network Scanner can be created by Python and Linux commands. Depending on the hacking tools and interface that the hacker is going to use, the required packages are different. Since I am using Kali Linux and Python, the required Python packages are ”scapy”, and “argparse”.

```

def scan(ip):  
 arp\_request = scapy.ARP(pdst=ip)  
 # use custom Ether to capture MAC of the broadcast  
 broadcast = scapy.Ether(dst='ff:ff:ff:ff:ff:ff')  
 arp\_request\_broadcast = broadcast / arp\_request  
  
 # srp return 2 list answered and unanswered list  
 answered\_list = scapy.srp(arp\_request\_broadcast, timeout=1, verbose=False)[0]  
 client\_list = []  
 for answer in answered\_list:  
 client\_list.append({"ip": answer[1].psrc, "mac": answer[1].hwsrc})  
  
 return client\_list

```

### 3. ARP Spoofer (ARP Cache Poisoning)

ARP(\*) can be used to discover all the connected clients on the same network. An ARP Spoofing program allows us to redirect the flow of the packets. Instead of them flowing through the internet, they would flow through the hacker’s computer first. Any requests sent and any responses received by the target computer will have to flow through the hacker’s computer. This means any messages, websites, images, usernames, and passwords entered by the victim will have to flow through the hacker’s computer. This allows hackers to read this information, modify it, or drop it. Therefore, ARP Spoofing is a very serious and very powerful attack. The reason why the attack is possible is that ARP is not very secure.

To understand how this works we need to have a basic understanding of ARP functionality. In networking, ARP is used to help network clients to identify other connected clients on the same network. ARP gets other clients’ Mac addresses that connect on the same network using the Mac address table (or ARP table). Each computer has an ARP table that links IP addresses on the same network to their Mac addresses.

The value in the ARP table can be easily modified by exploiting the ARP protocol. Normally when a device connects to the network and wants to send requests, it will send them to the router. Then, the router will go and send that request to the Internet, wait for the response, and then forward the response to the device that requested it.

If the hacker or the victim or any other computer on the network wants to send a request, they will send that request directly to the router. To interfere with the connections, I can exploit the ARP protocol and send two ARP responses: “one to the gateway and one to the victim”. I’m gonna tell the gateway that I am at the IP of the victim. Then, the gateway (access point) will update its ARP table. Moreover, it will associate the IP of the victim with my Mac address (I have to change the hacking computer’s MAC address to the victim's MAC address). I'll do the same with the victim: I'll send it an ARP response. I’m going to tell the victim that I am at the router IP. Therefore it's going to update its ARP table and associate the IP of the router with my own Mac address. As a result of this, the victim is gonna think that I am the router and the router is gonna think that I am the victim.

Anytime the victim wants to send any requests the requests will have to flow through my computer. I'm gonna forward them to the router. Anytime the access point or the router wants to send responses they're gonna go to my machine because it thinks that I am the victim. Then I'm going to forward it to the victim. This puts me in the middle of the connection and gives me so much power once I become the man in the middle

The main reason why I can do all of this is that ARP is not secure. First of all, clients can accept responses even if they did not send the request.

I am gonna send the response to the access point and the response to the victim telling them that I am at a specific IP. They will accept that response anyway without asking who am I or asking where is this IP’s original destination. Furthermore, they're also not going to verify who I am.

When I say that I am at the victim IP, I am not at that IP. because this computer is at this IP, the access point will trust, and it will update its ARP table based on the information that I sent. The same goes for the victim. Thus, these are the two main weaknesses of ARP protocol that allow us to run ARP spoofing attacks

Technology:

The required Python packages are ”scapy”, and “time”. The source code to run this program can be found at [13]

‘’’

def spoof(target\_ip\_sp, spoof\_ip):  
 target\_mac = get\_mac(target\_ip\_sp)  
 # pdst: IP address of the victim, hwdst: MAC address of the victim, psrc: IP address of the  
 packet = scapy.ARP(op=2, pdst=target\_ip\_sp, hwdst=target\_mac, psrc=spoof\_ip)  
 scapy.send(packet, verbose=False)

‘’’

### 4. Packet Sniffer

In the previous section, I have a program that will put me in the middle of the connection but I still can’t read the information is flowing through my computer. Since data is transferred in packets and unreadable, I'll need a packet sniffer to read this information. The Packet Sniffer is a program that reads packets or data that flow through an interface.

My Packet Sniffer will allow me to read the data that flows through a certain interface. Then once I use the tool that I've built previously to become the man in the middle the data will flow through my interface by default. Whenever I am the man in the middle I'll be able to read all the information that flows through our computer and will be able to see all the usernames passwords URLs and so on that any target computer sends or receives.

Hackers can use this information to create more attack plans. Either manual or automatic attack depends on the hacker’s skill. The gathered information can also help the virus to understand the user behavior as such more complicated plans can be created on the go.

Technology:

The required Python packages are ”scapy.all”, and “scapy.layers”. The source code to run this program can be found at [13]

```

def sniff(interface):  
 scapy.sniff(iface=interface, store=False, prn=process\_sniffed\_packet)  
  
  
def get\_url(packet):  
 return packet[http.HTTPRequest].Host + packet[http.HTTPRequest].Path

```

### 5. DNS Spoofer (DNS Cache Poisoning)

I can use scapy to modify this data. For example, I want to redirect any request made by our victim to some other place (a fake website, or a website that asks the victim to download a backdoor…). I can do that using scapy once I get the packet. Using my computer I can create another packet which will be the modified packet and then send that to its destination. The problem with this implementation is that Scapy cannot be used to intercept or drop packets.

When I’m receiving the request, I am creating a copy of that request or modified request and then both will be sent to the target. Therefore, I can't stop the original request from being sent to the target. The result is that the target is going to receive two requests: the original one and the modified one. Then it'll decide which one it'll execute. In reality, it usually executes the one that it receives first. Since modifying a request might take some time (half of a second), chances are the target will never execute my modified request. It will always receive the original request because the original request will always come first.

A better implementation is creating a queue in my hacker machine and trapping packets inside that queue. Whenever I get a request, I'll put it in my Queue and never send it to its target. Then I'll access my Queue from my Python Program and modify the packets as I want. After I finish modifying the packet I will send the request. As a result, the target will only receive one packet which is the modified packet (or the modified request).

The same way can be used to modify responses. Whenever I trap responses in my Queue, I access my Queue from my Python program. After modifying the packet, my program (virus) will forward it to the destination.

Technology:

The required Python packages are ”scapy.all”, and “netfilterqueue”. The source code to run this program can be found at [13]

```

def process\_packet(packet):  
 scapy\_packet = scapy.IP(packet.get\_payload())  
 if scapy\_packet.haslayer(scapy.DNSR):  
 print(scapy\_packet.show())  
 packet.accept()  
  
  
queue = netfilterqueue.NetfilterQueue()  
# call back function use in every sing queue  
queue.bind(0, process\_packet)  
queue.run()

```

### 1. Environment:

- Perform Man-In-The-Middle (MITM) attack on a virtual private network

- Established by using VM Ware

- Hardware:

- PC 64 bits, Intel Core i7 10th Gen, 36 GB of ram

- Software:

- Microsoft Window 10 (host)

- VM Ware Sphere (or free version)

- Kali Linux (free)

## Results:

## Conclusion/Future work

There are monitoring

## Keywords:

**Address Resolution Protocol** (ARP) is a protocol or procedure that connects an ever-changing Internet Protocol (IP) address to a fixed physical machine address

## IV. Reference

- [IEEE: Advance Technology for Humanity](https://www.ieee.org/) [1]

- [ISO 9000-3: Quality management and quality assurance standards](https://www.iso.org/standard/26364.html) [2]

- [SWEBOK V3.0: Guide to the Software Engineering Body of Knowledge](https://ieeecs-media.computer.org/media/education/swebok/swebok-v3.pdf) [3]

- [Tutorial Point: STLC Tutorial](https://www.tutorialspoint.com/stlc/index.htm) [4]

- [Software Testing | Security Testing](https://www.geeksforgeeks.org/software-testing-security-testing/?ref=lbp) [5]

- [Security Testing: Types, Tools, and Best Practices](https://www.neuralegion.com/blog/security-testing/) [6]

- [API Security: The Complete Guide](https://www.neuralegion.com/blog/api-security/) [7]

- [STLC (Software Testing Life Cycle) Phases, Entry, Exit Criteria (guru99.com)](https://www.guru99.com/software-testing-life-cycle.html) [8]

- Fink, G. & Bishop, M. "Property-Based Testing: A New Approach to Testing for Assurance." ACM SIGSOFT Software Engineering Notes 22, 4 (July 1997): 74-80.[9]

- McGraw, Gary & Potter, Bruce. "Software Security Testing." IEEE Security and Privacy 2, 5 (Sept.-Oct. 2004): 81-85. [10a]

- McGraw, Gary. "Application Security Testing Tools: Worth the Money?" Network Magazine, November 1, 2004. (2004). [10b]

- [Kali Linux](https://www.kali.org/?msclkid=ccbd3c3faa2511ecbe541363c15a4582) [11]

- [Python Scapy Package](https://pypi.org/project/scapy/?msclkid=33343ba2aa2611eca8b9c3abfd8b35c1) [12]

- [Truc Huynh: Ethical Hacking Using Python](https://github.com/jackyhuynh/ethical-hacking-using-python) [13]

- [Udemy: Learn Python and Ethical Hacking from Scratch](https://www.udemy.com/course/learn-python-and-ethical-hacking-from-scratch/learn/lecture/10800892#overview) [14]