Isfahan University Of Technology
Faculty of Electrical and Computer Engineering

Network Security Lab

Write-up Lab 2

Alireza Raisi

Instructor

Dr. Ali Fanian

Introduction

The Address Resolution Protocol (ARP) is essential for mapping IP addresses to MAC addresses in local networks. However, its lack of security makes it vulnerable to **ARP Cache Poisoning**, where attackers manipulate ARP tables to intercept or alter communication. This lab explores ARP spoofing techniques using ARP requests, responses, and gratuitous packets, and demonstrates a **Man-in-the-Middle (MITM)** attack on Telnet and Netcat sessions. Using a custom Python script with Scapy, we intercept and modify TCP payloads to highlight the risks of unencrypted protocols. The lab underscores the importance of securing ARP and using encryption to mitigate such attacks.

1 Methods for ARP Cache Poisoning

ARP cache poisoning can be achieved through various techniques, each exploiting the trust-based nature of the ARP protocol. Below are the primary methods used to corrupt ARP tables:

1.1 ARP Request Spoofing

In this method, the attacker sends a forged **ARP request** to the victim, claiming that the attacker's MAC address corresponds to the IP address of another host. While ARP requests are typically used to resolve IP-to-MAC mappings, malicious requests can trick the victim into updating its ARP cache with incorrect information.

1.2 ARP Reply Spoofing

An **ARP reply** is sent by the attacker without waiting for a corresponding ARP request. The forged reply claims that the attacker's MAC address is associated with the IP address of a legitimate host. This method is effective because ARP replies are trusted by default, and victims update their ARP caches based on the received reply.

1.3 Gratuitous ARP

A **gratuitous ARP** packet is a special type of ARP announcement where a device proactively informs the network about its IP-to-MAC mapping. An attacker can exploit this by sending a gratuitous ARP packet with a spoofed source IP and MAC address. This causes all devices on the network to update

their ARP caches with the attacker's information, redirecting traffic to the attacker.

Key Consideration

For **ARP Reply** attacks to be effective, the victim must already have an ARP entry for the target IP. Without a pre-existing entry, the victim may ignore the spoofed reply or send an ARP request to verify the mapping, rendering the attack ineffective. This restriction does not apply to ARP requests or gratuitous ARP, as they can create or overwrite ARP entries without requiring a prior record.

2 Attacks and Results

This section demonstrates the practical execution of ARP cache poisoning attacks and their impact on network communication. The attacks include ARP reply spoofing, Telnet session hijacking, and Netcat payload manipulation.

2.1 ARP Reply Spoofing

Using the provided Python code, an ARP reply spoofing attack was executed. The attacker sent a forged ARP reply to the victim, associating the attacker's MAC address with the IP address of a legitimate host. This caused the victim to update its ARP cache, redirecting traffic intended for the legitimate host to the attacker.

2.2 Telnet Session Hijacking

The attacker intercepted a Telnet session between two hosts, as shown in Figure 1. The Python script modified the payload of the Telnet packets, replacing all characters with "z". This resulted in the victim receiving garbled text (e.g., "zzzzz_") instead of the expected commands, effectively disrupting the session.

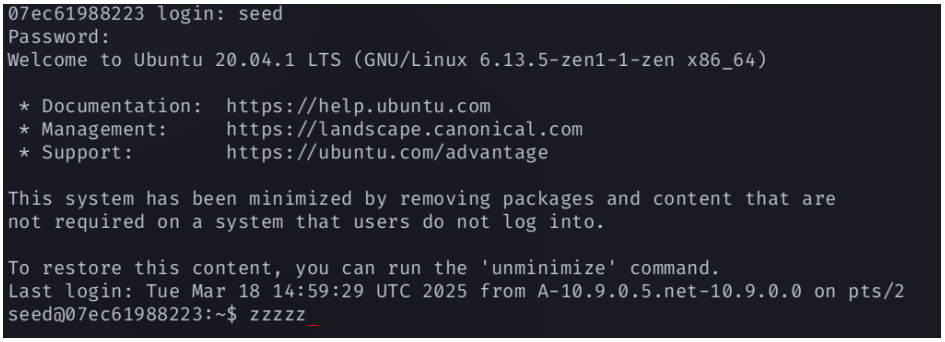

Figure 1: Telnet session hijacking showing modified payload.

2.3 Netcat Payload Manipulation

The attacker also intercepted and manipulated Netcat communication. As shown in Figure 2, the original payload (”ls, pwd”) was modified to ”ll” and ”ppp”. The victim received the manipulated payload, as seen in Figure 3, while the attacker sent the original commands, as shown in Figure 4.

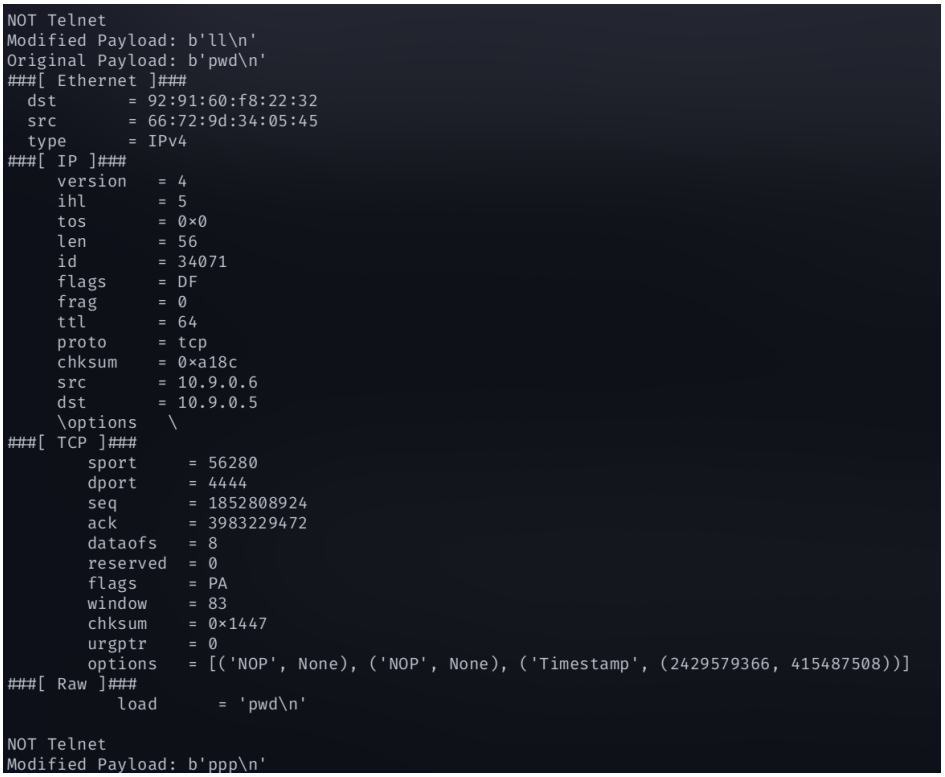

Figure 2: Attacker’s view showing original and modified payloads.

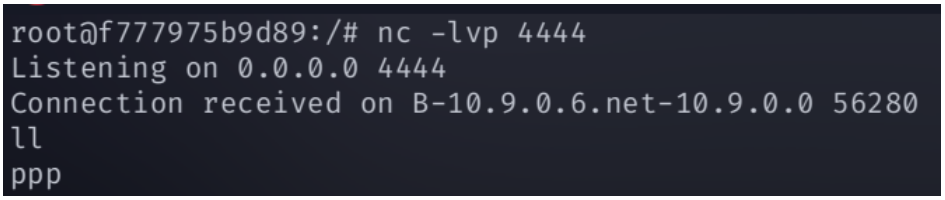

Figure 3: Victim A's view receiving manipulated Netcat payload.

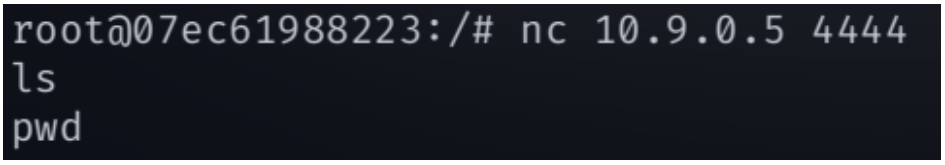

Figure 4: Victim B's view sending original Netcat commands.

Key Observations

- ARP cache poisoning effectively redirects traffic, enabling the attacker to intercept and manipulate communication.
- Unencrypted protocols like Telnet and Netcat are highly vulnerable to such attacks.
- The Python script successfully modified payloads in real-time, demonstrating the risks of unsecured network protocols.