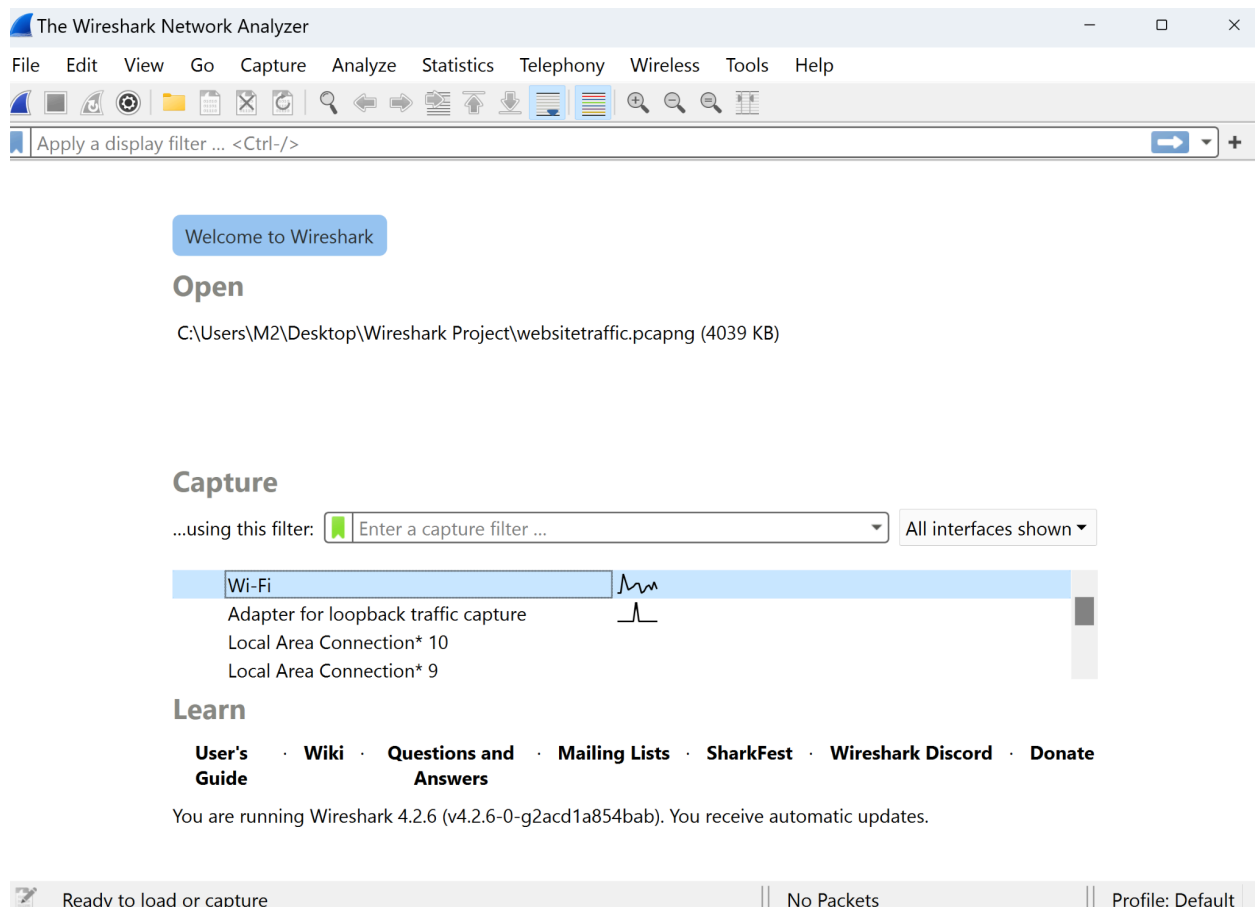


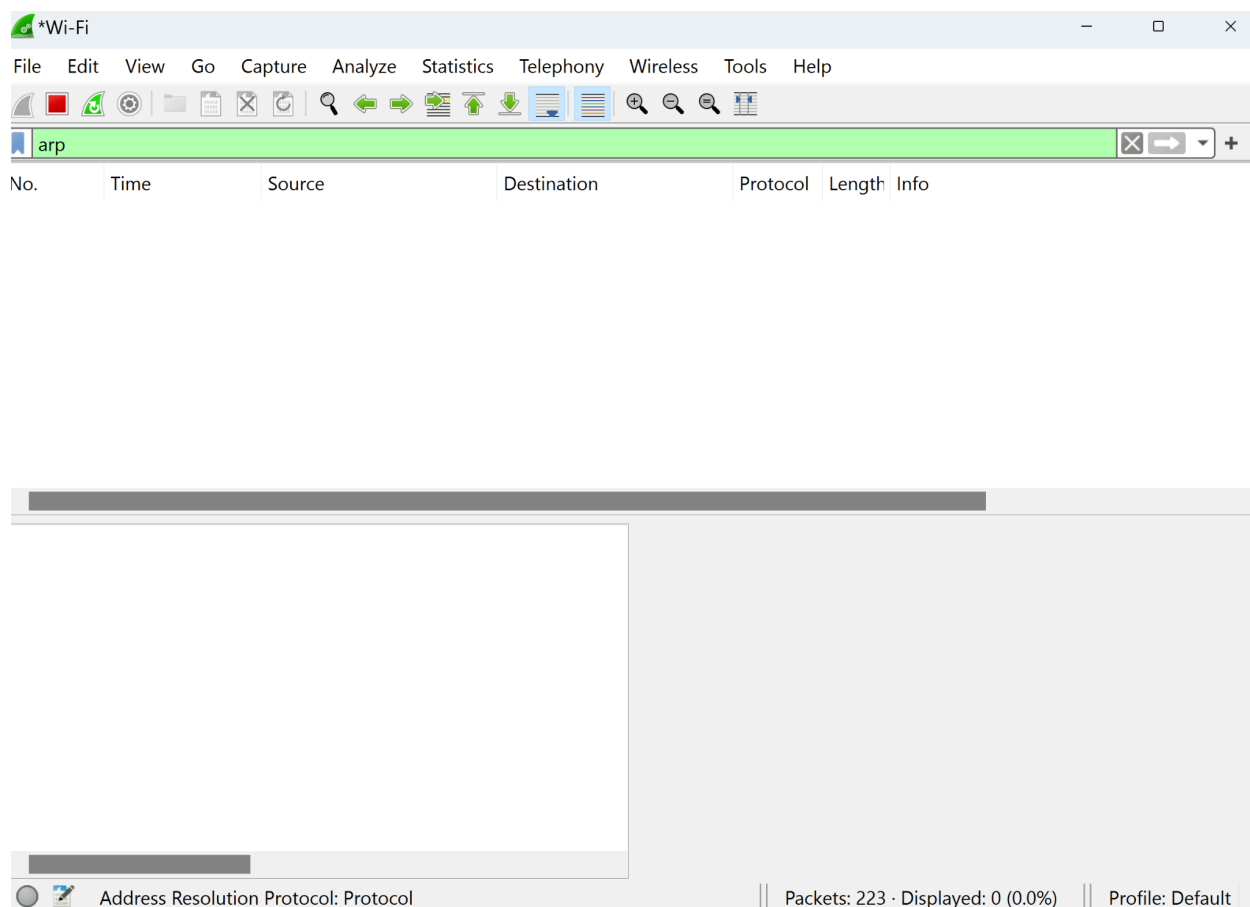
Using Nmap to Generate ARP Requests and Capturing Them with Wireshark by Matthew Miller 8/13/24

Step 1: Open Wireshark and Set Up for Capturing

1. **Launching Wireshark:** I opened Wireshark from my Start menu or desktop.
2. **Selecting a Network Interface:** Wireshark showed me a list of network interfaces, and I chose Wifi.



3. **Starting Capture:** I clicked the blue shark fin button to start capturing traffic on the selected interface.
4. **Filtering for ARP Traffic:** In the display filter bar at the top of Wireshark, I typed **arp** to filter only ARP traffic. This way, I could focus on ARP packets as they appeared.



Step 2: Use Nmap to Generate ARP Requests

1. **Opening Command Prompt:** I opened the Command Prompt on my Windows machine by pressing **Win + R**, typing **cmd**, and pressing **Enter**.



2. **Using Nmap for Scanning:** To generate ARP requests, I used Nmap to scan my local network. This scan sent ARP requests to identify live hosts on the network.

Basic ARP Scan: I ran the following command:

```
nmap -sn [my_network]/24
```

- I replaced `[my_network]` with my local network range.

Explanation:

- `nmap`: This is the command to invoke Nmap.
- `-sn`: This option tells Nmap to perform a "ping scan," which sends ARP requests without conducting a full port scan.
- `[my_network]/24`: This specifies the range of IP addresses on my local network. The `/24` subnet mask means I'm scanning all 256 addresses in a typical home network range.

3. **Running the Scan:** After typing the command, I pressed `Enter`, and Nmap started scanning the network, generating ARP requests in the process.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\M2> nmap -sn 192.168.1.0/24
Starting Nmap 7.95 ( https://nmap.org ) at 2024-08-13 15:03 Eastern Daylight Time
Nmap scan report for 192.168.1.1
Host is up (0.0075s latency).
MAC Address: 58:22:16:1E:1E:1E (TP-Link Technologies)
Nmap scan report for 192.168.1.2
Host is up (0.15s latency).
MAC Address: D8:9E:5C:12:12:12 (Samsung Electronics)
Nmap scan report for 192.168.1.3
Host is up (0.051s latency).
MAC Address: 28:9E:5C:12:12:12 (Amazon Technologies)
Nmap scan report for 192.168.1.4
Host is up (0.17s latency).
MAC Address: 8C:9E:5C:12:12:12 (Whisker Labs - Ting)
Nmap scan report for 192.168.1.5
Host is up (0.16s latency).
MAC Address: F0:9E:5C:12:12:12 (Cloud Network Technology Singapore PTE.)
Nmap scan report for 192.168.1.6
Host is up (0.15s latency).
MAC Address: 08:9E:5C:12:12:12 (Amazon Technologies)
Nmap scan report for 192.168.1.7
Host is up (0.15s latency).
MAC Address: 70:9E:5C:12:12:12 (Intel Corporate)
Nmap scan report for 192.168.1.8
Host is up (0.15s latency).
MAC Address: D8:9E:5C:12:12:12 (Shenzhen MTC)
Nmap scan report for 192.168.1.9
Host is up (0.18s latency).
```

Step 3: Capture ARP Requests in Wireshark

1. **Checking Wireshark for ARP Traffic:** I switched back to Wireshark and saw ARP requests appearing in real-time as a result of the Nmap scan.

*Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

arp

No.	Time	Source	Destination	Protocol	Length	Info
4672	146.270092	Microsoft_b4:26:53	ChinaDragonT_3d:a8:...	ARP	42	Who has 192.168.1.1? Tell me
4673	146.270124	ChinaDragonT_3d:a8:...	Microsoft_b4:26:53	ARP	42	192.168.1.1 is at 78:8a:86:3d:a8:d8
4675	146.388943	SamsungElect_21:0e:...	ChinaDragonT_3d:a8:...	ARP	42	Who has 192.168.1.1? Tell me
4676	146.388951	ChinaDragonT_3d:a8:...	SamsungElect_21:0e:...	ARP	42	192.168.1.1 is at 78:8a:86:3d:a8:d8
4677	146.563491	AmazonTechno_ce:39:...	ChinaDragonT_3d:a8:...	ARP	42	Who has 192.168.1.1? Tell me
4678	146.563511	ChinaDragonT_3d:a8:...	AmazonTechno_ce:39:...	ARP	42	192.168.1.1 is at 78:8a:86:3d:a8:d8
4683	146.795130	SamsungElect_21:0e:...	Broadcast	ARP	42	Who has 192.168.1.1? Tell me
4753	151.298009	SamsungElect_21:0e:...	Broadcast	ARP	42	Who has 192.168.1.1? Tell me
4789	152.321352	SamsungElect_21:0e:...	Broadcast	ARP	42	Who has 192.168.1.1? Tell me
4803	153.345247	SamsungElect_21:0e:...	Broadcast	ARP	42	Who has 192.168.1.1? Tell me

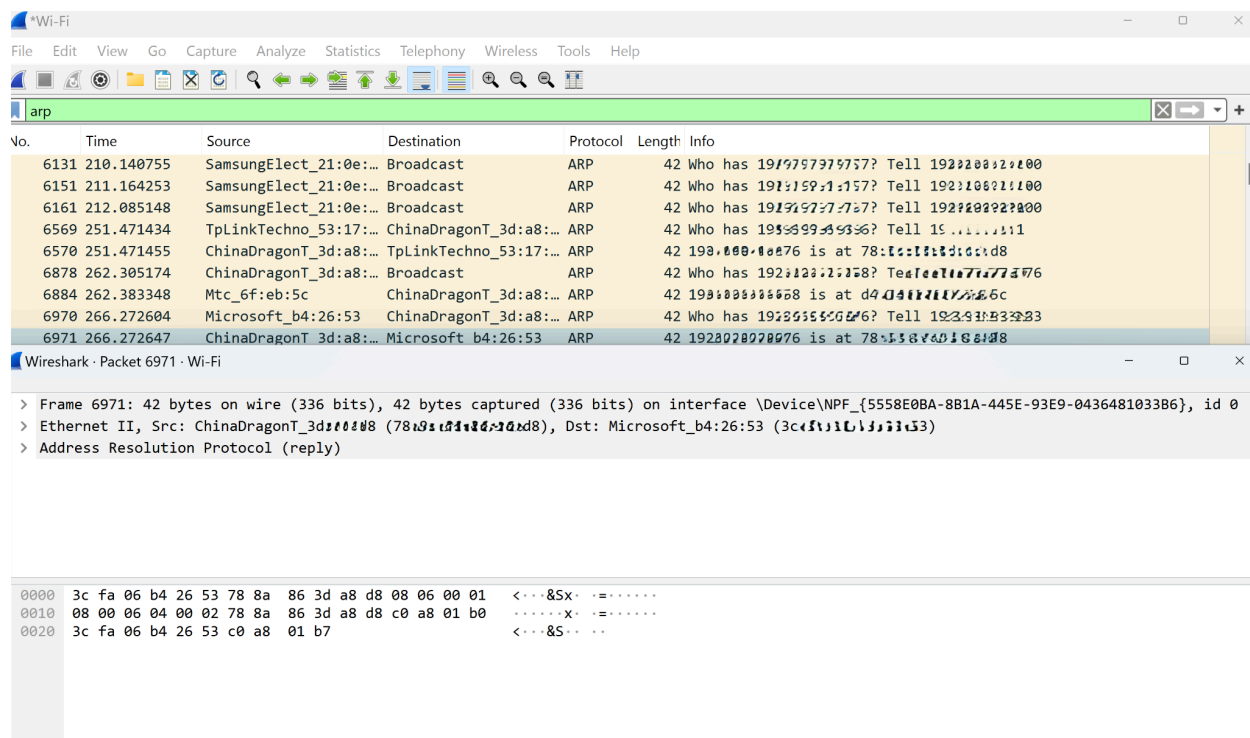
> Frame 486: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
 > Ethernet II, Src: SamsungElect_21:0e:a8 (d0:c2:4e:21:0e:a8), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 > Address Resolution Protocol (request)

0000 78 8a 86 3d a8 d8 d0 c2 4e 21 0e a8 08 06 00 01
 0010 08 00 06 04 00 01 d0 c2 4e 21 0e a8 c0 a8 01 64
 0020 00 00 00 00 00 00 c0 a8 01 b0

wireshark Wi-FiYG2JS2.pcapng | Packets: 11483 · Displayed: 621 (5.4%) · Dropped: 0 (0.0%) | Profile: Default

2. Inspecting the ARP Packets:

- I clicked on an ARP packet in the list to highlight it.
- The packet details pane showed me the ARP request and reply information.
- I could see who was asking for which IP address and the MAC address involved.



Step 4: Stop the Capture and Save the Data

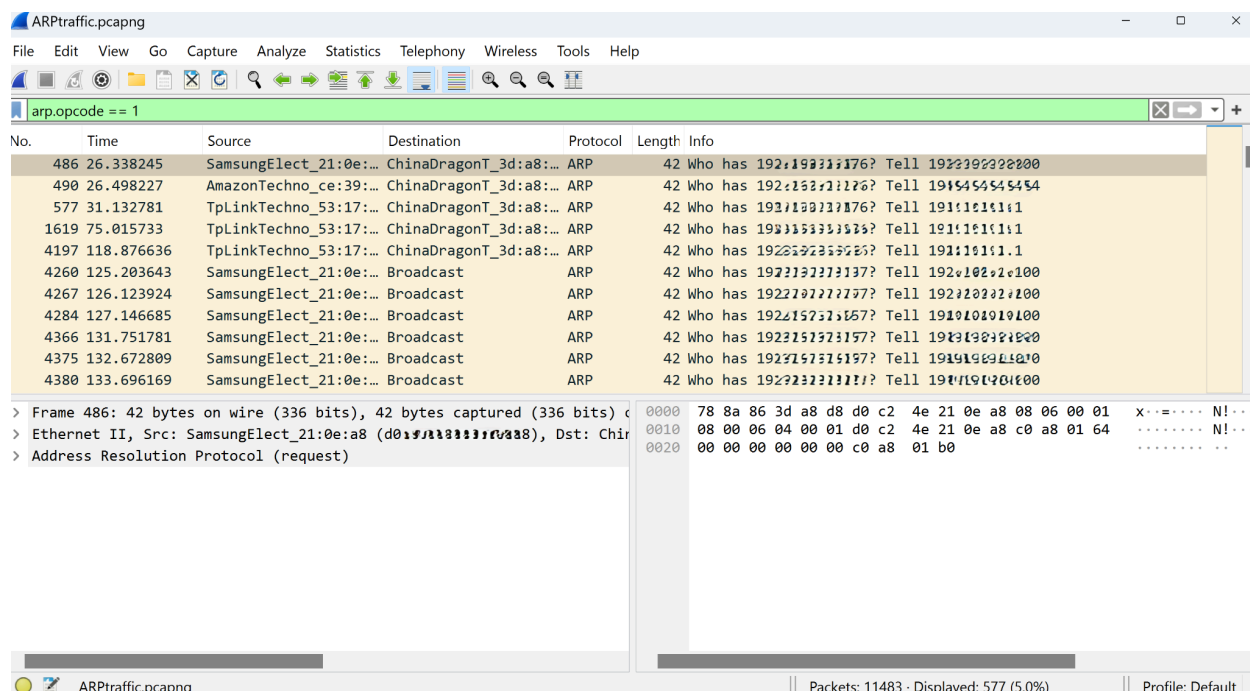
1. **Stopping Capture:** Once I had captured enough ARP requests, I clicked the red square button to stop the capture.
2. **Saving the Capture:**
 - I went to **File > Save As** and chose a location and filename to save my capture file.
 - I saved the file with a **.pcapng** extension.

Step 5: Further Analysis of ARP Packets

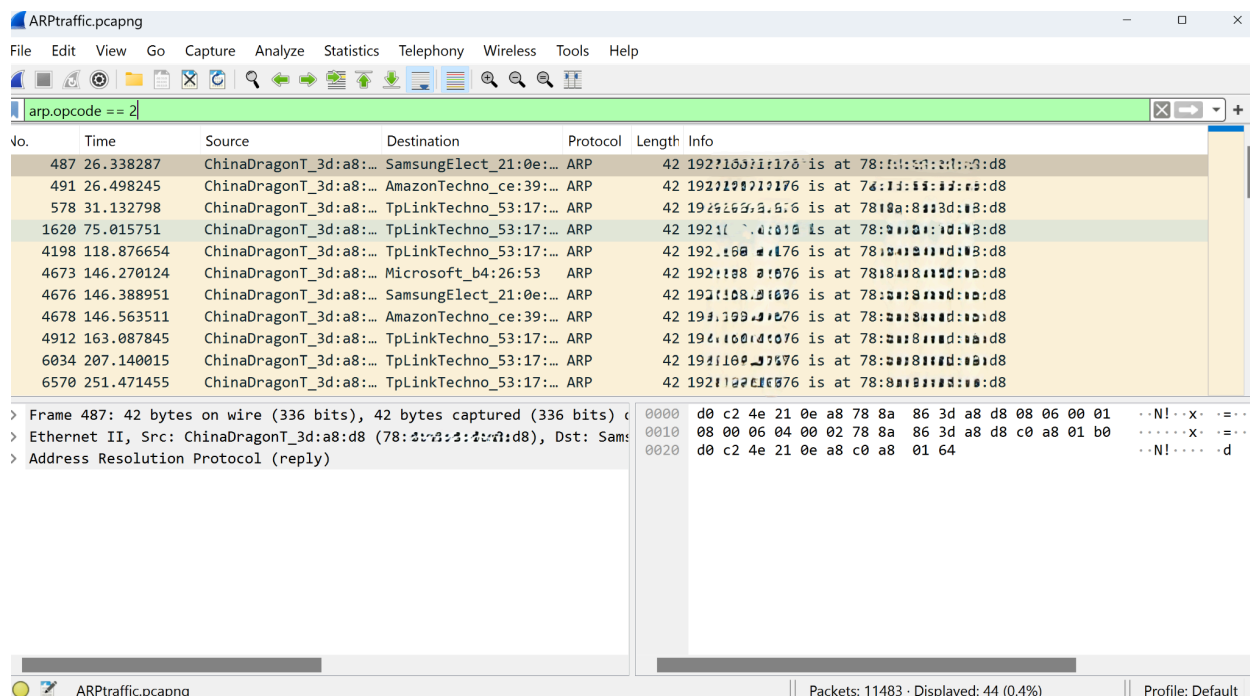
Here's how I did a deeper dive into the ARP packets I captured:

1. Filtering Specific ARP Packets

- **Filter for ARP Requests and Replies:**
 - If I want to isolate ARP requests, I use the filter **arp.opcode == 1**. This will show only the ARP requests where a device is asking "Who has this IP address?"

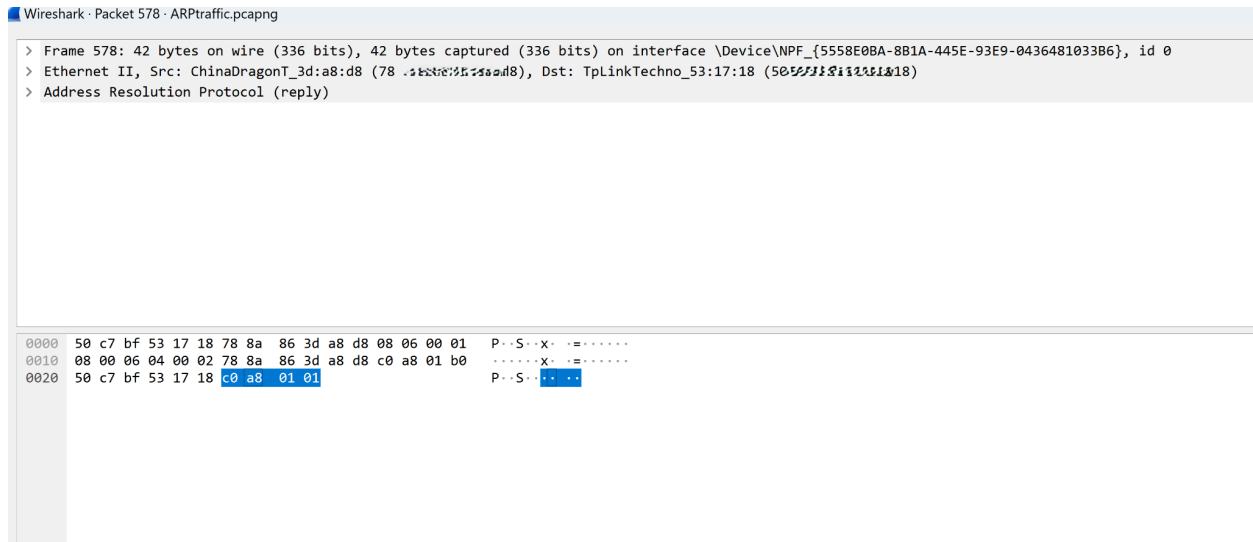


- To filter for ARP replies, I use `arp.opcode == 2`. This filter shows only the replies where a device responds, "I have this IP address."
- I enter these filters in the display filter bar at the top of Wireshark and press **Enter**.



2. Inspecting the ARP Packet Details

- Packet List Pane:
 - In the top pane, I see a list of captured packets. I double click on any ARP packet to highlight it. The details of this packet will appear in the middle pane.
- Packet Details Pane:
 - In the middle pane, I see a breakdown of the selected packet. This is where I can inspect the ARP details:



I clicked on an ARP reply packet (Packet 578). The packet has a length of 42 bytes and was captured on the interface with the identifier `\Device\NPF_{5558E0BA-8B1A-445E-93E9-0436481033B6}`.

The packet's source MAC address is `78*****d8`, identified as "ChinaDragonT_3d:a8", and the destination MAC address is `50:*****:18`, identified as "TpLinkTechno_53:17:18". The protocol used here is the Address Resolution Protocol (ARP), specifically an ARP reply.

The hex dump shows the structure of the packet, with the ARP protocol fields visible in the payload.

Inspect Ethernet Frame Details:

- Click on the "Ethernet II" line to expand it.
- Review the source and destination MAC addresses to confirm the devices involved.

```
Wireshark · Packet 578 · ARPtraffic.pcapng

> Frame 578: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF_{5558E0BA-8B1A-445E-93E9-0436481033B6}, id 0
> Ethernet II, Src: ChinaDragonT_3d:a8:d8 (78:8f:08:0d:09:d8), Dst: TplinkTechno_53:17:18 (50:13:00:00:00:18)
  ▾ Destination: TplinkTechno_53:17:18 (50:13:00:00:00:18)
    Address: TplinkTechno_53:17:18 (50:13:00:00:00:18)
    ....0. .... = LG bit: Globally unique address (factory default)
    ....0. .... = IG bit: Individual address (unicast)
  ▾ Source: ChinaDragonT_3d:a8:d8 (78:8f:08:0d:09:d8)
    Address: ChinaDragonT_3d:a8:d8 (78:8f:08:0d:09:d8)
    ....0. .... = LG bit: Globally unique address (factory default)
    ....0. .... = IG bit: Individual address (unicast)
  Type: ARP (0x0806)
> Address Resolution Protocol (reply)

0000  50 c7 bf 53 17 18 78 8a 86 3d a8 d8 08 06 00 01  P...S...X...=...
0010  08 00 06 04 00 02 78 8a 86 3d a8 d8 c0 a8 01 b0  ....X...=.....
0020  50 c7 bf 53 17 18 c0 a8 01 01  P..S.....
```

Analyze ARP Packet Content:

- Expand the "Address Resolution Protocol (reply)" section.
- Verify the ARP operation type (in this case, it should be "Reply").
- Examine the sender and target IP addresses within the ARP payload to understand the IP mapping involved in this exchange.

```
Wireshark · Packet 578 · ARPtraffic.pcapng

> Frame 578: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF_{5558E0BA-8B1A-445E-93E9-0436481033B6}, id 0
> Ethernet II, Src: ChinaDragonT_3d:a8:d8 (78:8f:08:0d:09:d8), Dst: TplinkTechno_53:17:18 (50:13:00:00:00:18)
> Address Resolution Protocol (reply)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: reply (2)
  Sender MAC address: ChinaDragonT_3d:a8:d8 (78:8f:08:0d:09:d8)
  Sender IP address: 192.168.1.176
  Target MAC address: TplinkTechno_53:17:18 (50:13:00:00:00:18)
  Target IP address: 192.168.1.1

0000  50 c7 bf 53 17 18 78 8a 86 3d a8 d8 08 06 00 01  P...S...X...=...
0010  08 00 06 04 00 02 78 8a 86 3d a8 d8 c0 a8 01 b0  ....X...=.....
0020  50 c7 bf 53 17 18 c0 a8 01 01  P..S.....

No: 578 · Time: 31.132798 · Source: ChinaDragonT_3d:a8:d8 · Destination: TplinkTechno_53:17:18 · Protocol: ARP · Length: 42 · Info: 192.168.1.176 is at 78:8a:86:3d:a8:d8
Show packet bytes
```

Cross-Referencing with Network Data

- **Identify Relationships:** I can cross-reference ARP packets with other captured data. For example, I can identify if the MAC address revealed in the ARP reply is consistent with what I see in other types of traffic from that IP.
- **Look for Anomalies:** I examine if there are unusual ARP requests, such as frequent or unexpected requests from unknown devices, which could indicate suspicious activity like ARP spoofing or a rogue device on the network.

Generate Reports: Wireshark also allows me to generate summary reports or statistics:

- I can go to **Statistics > Protocol Hierarchy** to get an overview of the ARP traffic compared to other protocols.

Wireshark · Protocol Hierarchy Statistics · ARPtraffic.pcapng

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDU's
▼ Frame	100.0	44	100.0	1848	34	0	0	0	44
▼ Ethernet	100.0	44	33.3	616	11	0	0	0	44
Address Resolution Protocol	100.0	44	66.7	1232	23	44	1232	23	44

Total Packets Captured:

- **44 Packets** were captured during this session, all of which fall under the Ethernet protocol.

Breakdown by Protocol:

- **Ethernet (100%):**
 - All 44 packets (100% of the captured traffic) are Ethernet frames. This indicates that all traffic in this capture was sent over Ethernet, which is the most common link-layer protocol.
- **Address Resolution Protocol (ARP):**
 - Within the Ethernet frames, all 44 packets (100% of the Ethernet traffic) are ARP packets. This means the entire capture consists of ARP traffic, which is used for resolving IP addresses to MAC addresses within a local network.
- **Bytes and Bandwidth:**
 - **Total Bytes:** The total amount of data captured is 1,848 bytes, with 1,232 bytes belonging to ARP (66.7% of total bytes) and 616 bytes to Ethernet (33.3% of total bytes).
 - **Bits/s:** The data transfer rate for ARP traffic is 23 bits per second, indicating the bandwidth consumption of ARP traffic within this capture.

- Using **Statistics > Endpoints**, I can view a list of IP addresses and MAC addresses, providing a broader picture of the devices involved in the ARP communication.

The screenshot shows the Wireshark 'Endpoints' window for the file 'ARPtraffic.pcapng'. The 'Endpoint Settings' panel on the left has 'Limit to display filter' checked. The main table displays statistics for the 'Ethernet II' protocol. The table has columns for Address, Packets, Bytes, Total Packets, Percent Filtered, Tx Packets, Tx Bytes, Rx Packets, and Rx Bytes. The data is as follows:

Address	Packets	Bytes	Total Packets	Percent Filtered	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
08:00:27:00:00:00:7e	5	210 bytes	45	11.11%	1	42 bytes	4	168 bytes
20:00:00:00:00:00:bd	1	42 bytes	1	100.00%	1	42 bytes	0	0 bytes
3c:00:00:00:00:00:53	8	336 bytes	110	7.27%	5	210 bytes	3	126 bytes
50:00:00:00:00:00:18	15	630 bytes	10,074	0.15%	2	84 bytes	13	546 bytes
70:00:00:00:00:00:c4	1	42 bytes	65	1.54%	1	42 bytes	0	0 bytes
78:00:00:00:00:00:d8	44	2 kB	10,374	0.42%	24	1 kB	20	840 bytes
80:00:00:00:00:00:a5	1	42 bytes	1	100.00%	1	42 bytes	0	0 bytes
d0:00:00:00:00:00:a8	5	210 bytes	474	1.05%	1	42 bytes	4	168 bytes
d4:00:00:00:00:00:5c	7	294 bytes	71	9.86%	7	294 bytes	0	0 bytes
f0:00:00:00:00:00:6f	1	42 bytes	1	100.00%	1	42 bytes	0	0 bytes

The status bar at the bottom indicates 'Address Resolution Protocol (arp), 28 bytes' and 'Packets: 11483 · Displayed: 44 (0.4%)'.

This section provides an analysis of the endpoints involved in the ARP traffic captured in the **ARPtraffic.pcapng** file. The analysis focuses on the Ethernet layer, detailing the packet and byte distribution across different MAC addresses (endpoints) within the network.

Key Observations:

- Total Packets and Bytes:**
 - The capture includes a total of **44 packets** with a combined size of approximately **2 kB** of data.
- Endpoints (MAC Addresses):**
 - 08:00:27:00:00:00:7e:**
 - Packets:** 5
 - Bytes:** 210 bytes
 - Total Packets:** 45 (11.11% of the filtered packets)
 - Tx Packets:** 1
 - Rx Packets:** 4
 - Description:** This MAC address exchanged a small amount of data, transmitting 42 bytes and receiving 168 bytes.
 - 3c:00:00:00:00:00:53:**
 - Packets:** 8

- **Bytes:** 336 bytes
 - **Total Packets:** 110 (7.27% of the filtered packets)
 - **Tx Packets:** 5
 - **Rx Packets:** 3
 - **Description:** This endpoint transmitted and received a moderate amount of data, indicating it was actively involved in ARP exchanges.
 - **50:*****:18:**
 - **Packets:** 15
 - **Bytes:** 630 bytes
 - **Total Packets:** 10,074 (0.15% of the filtered packets)
 - **Tx Packets:** 2
 - **Rx Packets:** 13
 - **Description:** This MAC address was a significant participant in the ARP communication, with a higher count of received packets (546 bytes).
 - **78*****:a8**
 - **Packets:** 44
 - **Bytes:** 2 kB
 - **Total Packets:** 10,374 (0.42% of the filtered packets)
 - **Tx Packets:** 24
 - **Rx Packets:** 20
 - **Description:** This endpoint is the primary communicator in the ARP traffic, responsible for the bulk of the data exchange.
3. **Endpoints with Single Packet Transmission:**
- **20:*****:b5**
, 70:***:55**
, 80:***:5f**
, f0:***:6f:** These addresses each transmitted a single packet with 42 bytes, indicating minimal interaction in the ARP communication.

Notes: tx means transmission, rx means receive

Summary:

The ARP traffic captured in this session involves multiple endpoints, with a few key addresses being more active in transmitting and receiving data. The endpoint **78:*****:a8**

stands out as the most active participant, responsible for a substantial portion of the data exchange, likely acting as a central node or router in the network. The presence of multiple endpoints with minimal packet exchange suggests either sporadic ARP requests/replies or the potential involvement of devices briefly connected to the network.

Conclusion

By following these steps, I successfully used Nmap to generate ARP requests and captured them using Wireshark and analyzed the packets. This exercise demonstrates my understanding of basic network scanning and packet analysis techniques.