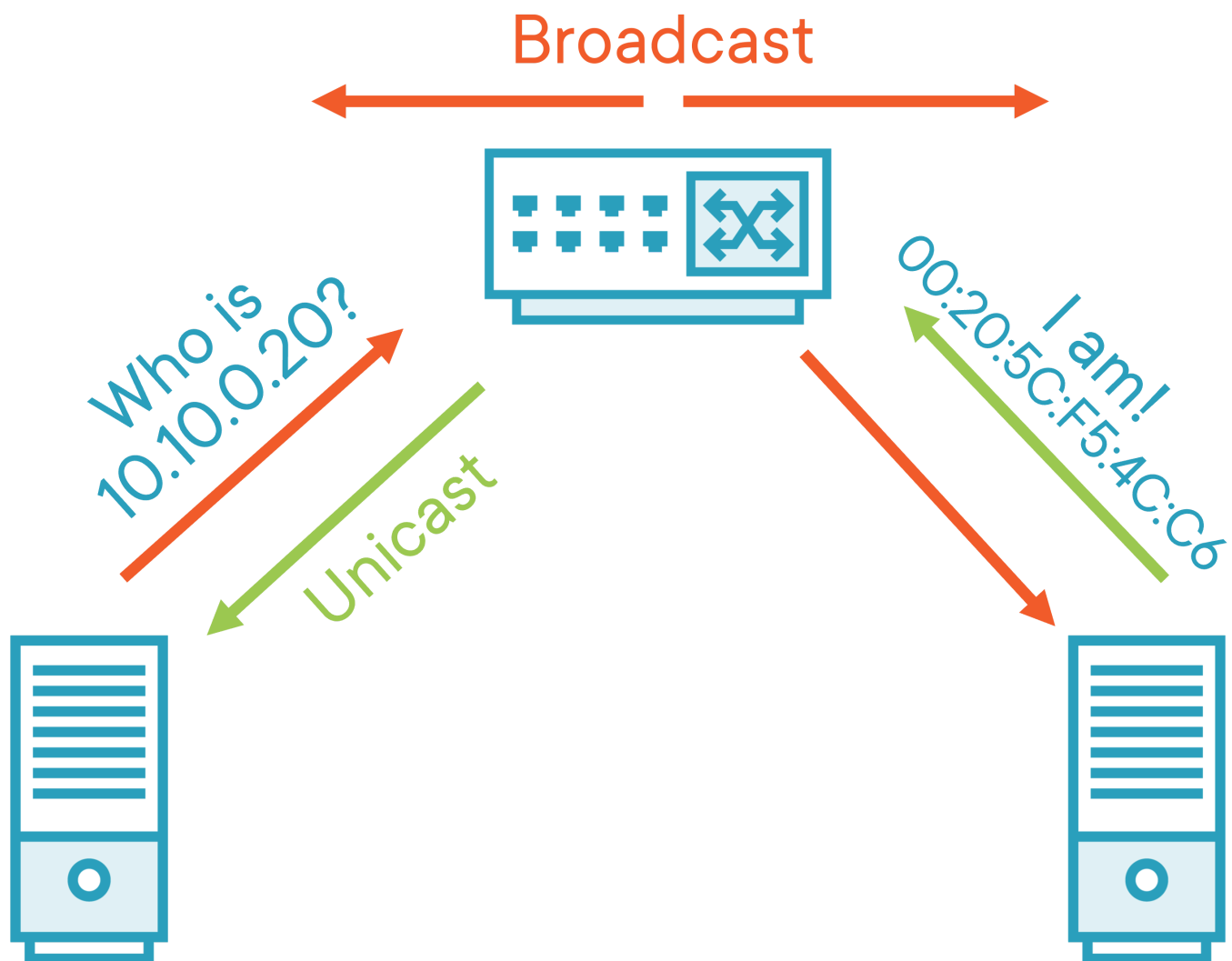


# Address Resolution Protocol (ARP)

When endpoints communicate across networks, they use logical IP addresses to keep track of where the requests come from and where the intended destination is. Once a packet arrives internal to an environment, networking devices need to convert that IP address to the more specific "physical" location the packets are destined for. That "physical" location is the MAC address you just analyzed in the last challenge. The Address Resolution Protocol (ARP) is the protocol which functions to make that translation.



**Note:** You must accomplish the steps in the previous challenge to continue with this challenge!

1. Continuing on the **Ubuntu Endpoint**, take a closer look at the first two packets:

```
tcpdump -nn -r ~/combined-analysis.pcap -e -c 2
```

**Note:** The tcpdump `-c` option allows you to specify a specific number of packets to read, starting at the top

```
02:42:9c:8f:b7:17 > ff:ff:ff:ff:ff:ff, ethertype ARP  
02:42:ac:11:00:02 > 02:42:9c:8f:b7:17, ethertype ARP
```

```
Request who-has 172.17.0.2 tell 172.17.0.1,  
Reply 172.17.0.2 is-at 02:42:ac:11:00:02, l
```

2. Let's take this analysis step-by-step. When you sent the curl request, the first thing that must take place is to figure out the intended destination. Where did the initial ARP request come from?

`ip link`

```
ubuntu@ip-172-31-24-10:~$ ip link  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500  
    link/ether 02:cf:3b:58:45:a5 brd ff:ff:ff:ff:ff:ff  
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500  
    link/ether ca:fe:c0:ff:ee:00 brd ff:ff:ff:ff:ff:ff  
5: vethf72823a@1f4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500  
    link/ether ea:42:30:a5:9d:33 brd ff:ff:ff:ff:ff:ff
```

**Analysis:** If you are taking good notes, you'll recognize that MAC address as the original MAC assigned to the docker interface before you changed it!

3. It looks like the first packet has a destination MAC of "ff:ff:ff:ff:ff:ff". What's that about? Since your endpoint doesn't know the destination MAC address, the first ARP packet is a broadcast.

```
02:42:9c:8f:b7:17 > ff:ff:ff:ff:ff:ff, ethertype ARP  
02:42:ac:11:00:02 > 02:42:9c:8f:b7:17, ethertype ARP
```

**Reference:** [https://en.wikipedia.org/wiki/Broadcast\\_address](https://en.wikipedia.org/wiki/Broadcast_address)

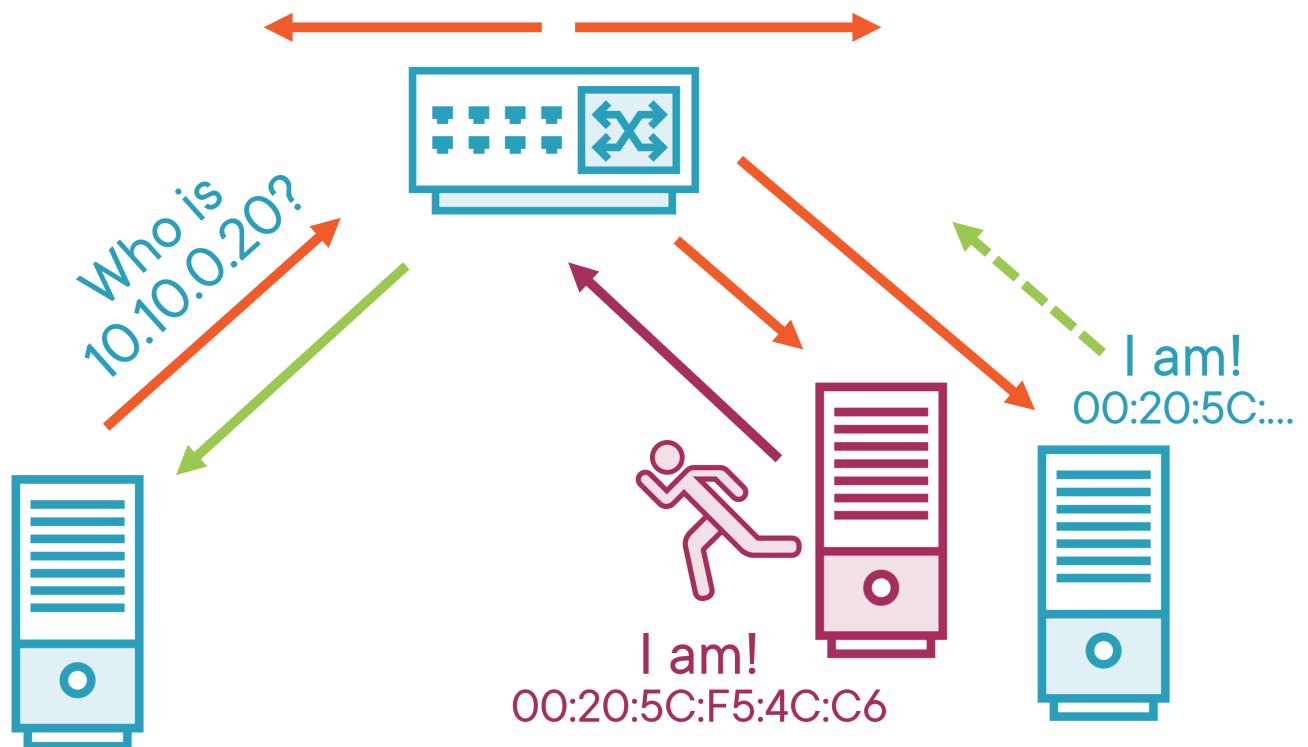
**Analysis:** A broadcasted packet will be sent to every host within the local network. Unfortunately, the ARP protocol was not developed with security in mind, so in most configurations, the first host to respond to the ARP request will be the "winner". This makes it very simple if an attacker controls a host within an

environment to spoof their own MAC, respond faster, and effectively perform a Man-In-The-Middle (MiTM) attack.

4. What IP address is the intended destination for this ARP request?

```
Request who-has 172.17.0.2 tell 172.17.0.1,  
Reply 172.17.0.2 is-at 02:42:ac:11:00:02, l
```

**Analysis:** The requesting IP address must be found in the payload of the packet. This is an important distinction, since most packets are returned to the requesting IP address found in the IPv4 header. This allows for adversaries to use attacks such as ARP spoofing and MAC flooding since the original requester doesn't have to be the intended destination.



5. Let's take a look at an example of one of these attacks. Using a similar method as the previous exercise, see if you can determine how many IP addresses are being targeted by a single MAC address in the **arp-storm.pcap** file:

```
tcpdump -nn -r ~/arp-storm.pcap 'arp' -e | cut -f 2,14 -d " " | sort | uniq  
-c | sort -n
```

```
reading from file arp-storm.pcap, link-type EN10MB (Ethernet)
  2 00:07:0d:af:f4:54 69.23.182.1,
  3 00:07:0d:af:f4:54 24.145.164.129,
  8 00:07:0d:af:f4:54 67.52.222.1,
  9 00:07:0d:af:f4:54 65.26.71.1,
 27 00:07:0d:af:f4:54 69.81.17.1,
 29 00:07:0d:af:f4:54 65.28.78.1,
 47 00:07:0d:af:f4:54 65.26.92.1,
205 00:07:0d:af:f4:54 69.76.216.1,
292 00:07:0d:af:f4:54 24.166.172.1,
```

**Analysis:** Here you can see a single MAC address pretending to be nine different IP addresses. As a simple attack, this can easily be scaled up to effectively accomplish a Denial of Service (DoS) attack or cause your networking devices and endpoints to drop the listing of what IP addresses are associated with each legitimate MAC address.

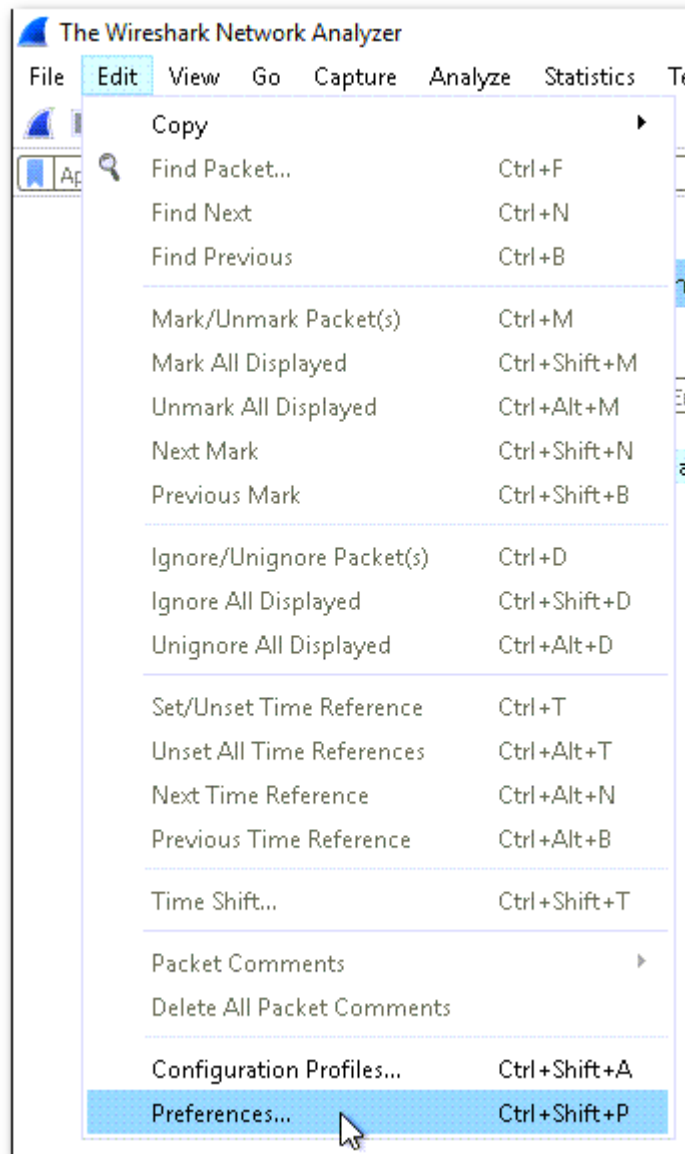
6. Let's take a look at another way you can analyze this traffic using Wireshark. Start by swapping to the **Windows 2019 Endpoint**.

To switch:

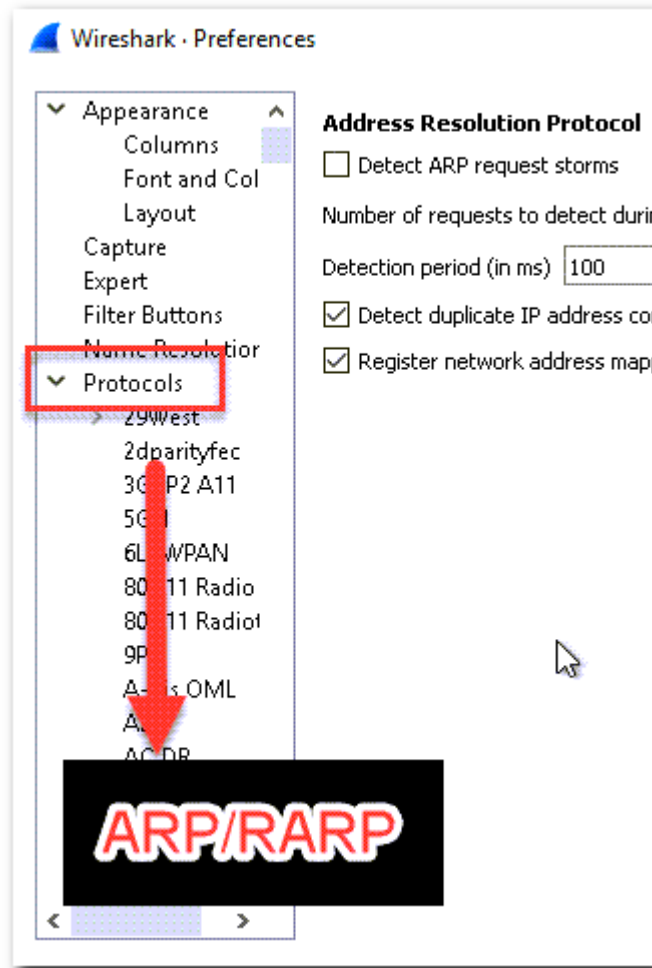
- **CTRL** + **ALT** + **SHIFT** (Windows)
- **CTRL** + **CMD** + **SHIFT** (macOS)

7. Open up Wireshark by double-clicking the icon on the desktop.
8. Wireshark contains adjustable preferences for each protocol. Navigate to the ARP/RARP options via the following menu options:

**Edit --> Preferences**



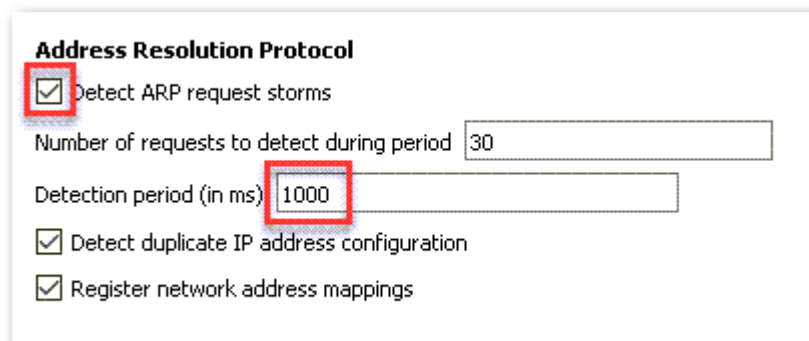
9. Expand the Protocols menu (on the left), scroll down, and select **ARP/RARP**



10. You'll need to make two changes here, then click **OK** to save your changes:

1. Check the box for **Detect ARP request storms**
2. Change the **Detection period (in ms)** to: **1000**

**Note:** Wireshark has the capability to attempt detection of an excessive rate of ARP requests. That's what you are taking advantage of here.



11. Now open the **arp-storm.pcap** file:

**File --> Open**, and double-click: **arp-storm.pcap**

**Note:** The file is located on the Desktop in a folder called **LAB\_FILES**.

12. This is the same PCAP you analyzed on the Ubuntu endpoint. Click on one of the packets and expand out the ARP section of the packet details pane to see a cleaner breakout of the ARP header information.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco251_a...	Broadcast	ARP	60	Who has 24.166.173.159? Tell 24.166.172.1
2	0.098594	Cisco251_a...	Broadcast	ARP	60	Who has 24.166.172.141? Tell 24.166.172.1
3	0.110617	Cisco251_a...	Broadcast	ARP	60	Who has 24.166.173.161? Tell 24.166.172.1
4	0.211791	Cisco251_a...	Broadcast	ARP	60	Who has 65.28.78.76? Tell 65.28.78.1
5	0.216744	Cisco251 a...	Broadcast	ARP	60	Who has 24.166.173.163? Tell 24.166.172.1

<

> Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface unknown, id 0

> Ethernet II, Src: Cisco251 af:f4:54 (00:07:0d:af:f4:54), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

✓ Address Resolution Protocol (request)

Hardware type: Ethernet (1)

Protocol type: IPv4 (0x0800)

Hardware size: 6

Protocol size: 4

Opcode: request (1)

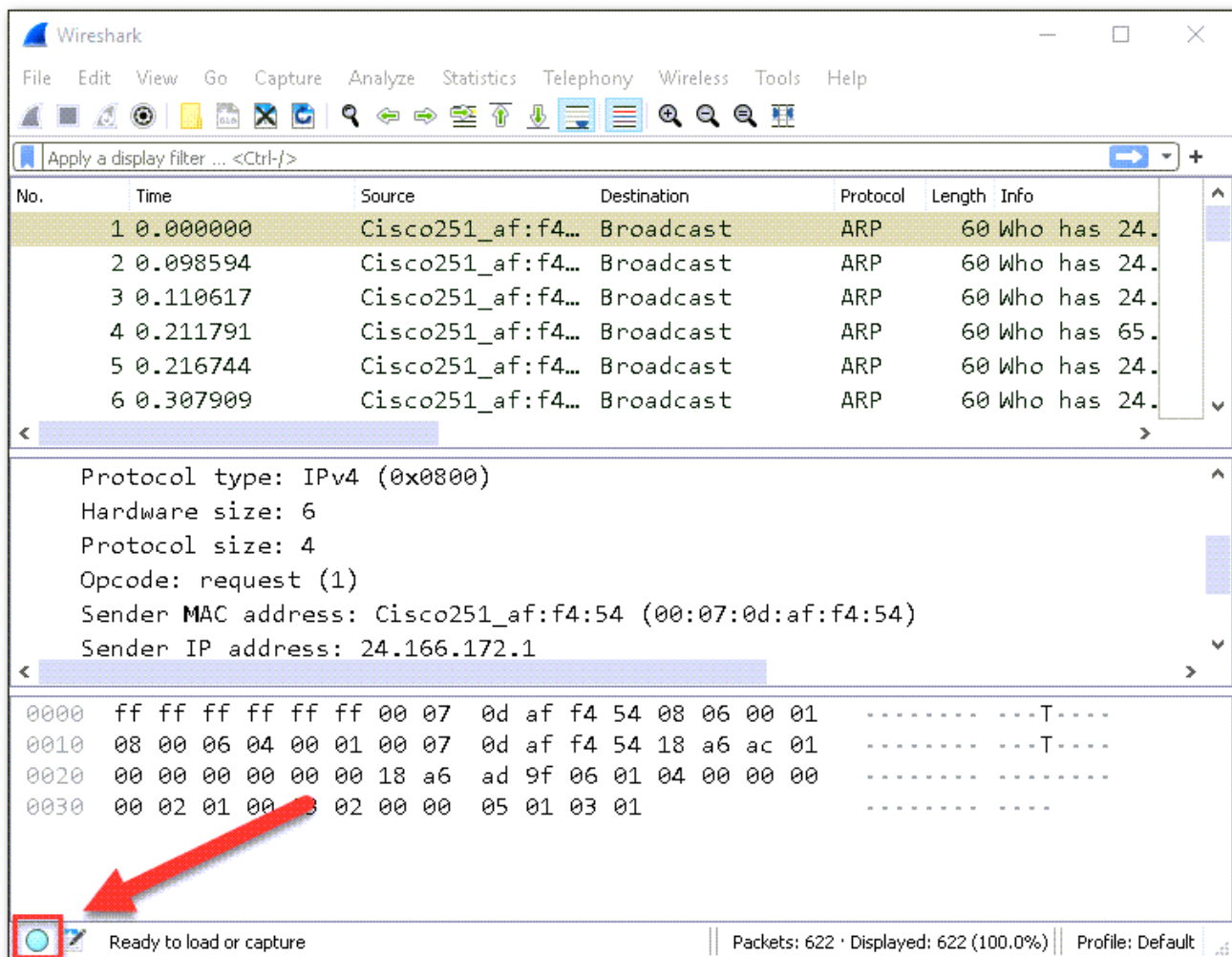
Sender MAC address: Cisco251\_af:f4:54 (00:07:0d:af:f4:54)

Sender IP address: 24.166.172.1

Target MAC address: 00:00:00\_00:00:00 (00:00:00:00:00:00)

Target IP address: 24.166.173.159

13. Now take advantage of the ARP-storm detection options by selecting the Expert Information Icon found in the bottom-left.



**Note:** Wireshark has a massive amount of customization available for analyzing various protocols. For more information, check out [Chris Greer's](#) course on [Wireshark Configuration for Cyber Security Analysis](#).

- Wireshark added a note here to let you know it detected three instances where an ARP packet storm was detected. Based on your configuration settings, at least 30 packets were seen destined for a specific IP address within one second.

The image shows the Wireshark Expert Information pane for the file arp-storm.pcap. It displays a table of notes generated by the tool.

Severity	Summary	Group	Protocol	Count
Note	ARP packet storm detected (30 packets in < 1000 ms)	Sequence	ARP/RARP	
57	Who has 69.76.218.63? Tell 69.76.216.1	Sequence	ARP/RARP	
88	Who has 24.166.172.169? Tell 24.166.172.1	Sequence	ARP/RARP	
137	Who has 69.76.223.251? Tell 69.76.216.1	Sequence	ARP/RARP	
Note	Didn't find padding of zeros, and an undecoded trailer exists...	Protocol	Ethertype	

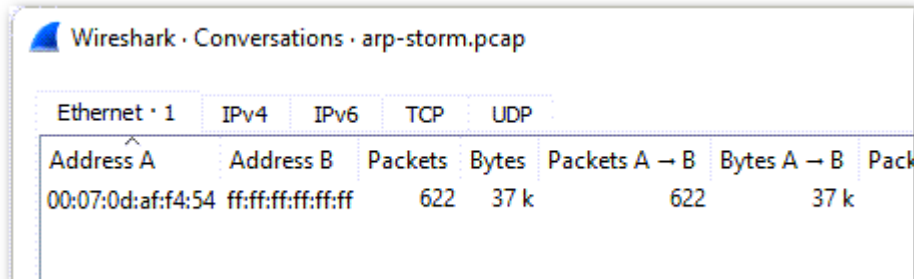
**Analysis:** Based on your analysis from the Ubuntu endpoint, you may wonder why only three of the IP addresses are marked despite you knowing the MAC address was spoofing a total of nine IP addresses. With the current settings, only three of those instances were above the "threshold" for Wireshark to make a note.



Considering a production network, you may need to adjust those settings based on the "noisiness" of your legitimate ARP broadcasts.

15. Another location you can detect this type of anomalous behavior is via the Conversations menu:

Statistics --> Conversations



The image shows the Wireshark 'Conversations' window for the file 'arp-storm.pcap'. The 'Ethernet 1' tab is selected. The table below displays the conversation data.

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A
00:07:0d:af:f4:54	ff:ff:ff:ff:ff:ff	622	37 k	622	37 k		

**Analysis:** Having a single MAC address broadcast 622 packets is a bit strange. Always consider the length of the overall PCAP, but seeing this type of activity can almost always lead you to some strange behavior.

Now that you have a fundamental understanding on how MAC addresses work and their link to IPs, let's move onto the next challenge where you will gain some further understanding of IP addressing.