**Exploring DNS Traffic Using Wireshark**

In today’s interconnected world, the Domain Name System (DNS) plays a critical role in resolving human-readable domain names into IP addresses, allowing users to access websites and services effortlessly. However, beneath this simplicity lies a complex exchange of query and response packets that can reveal much about network behaviour and security.

This lab focuses on using Wireshark, a powerful open-source packet capture and analysis tool, to inspect DNS traffic in detail. Wireshark enables users to visualize the entire network protocol stack, filter traffic based on protocol types, and analyse individual packet contents—making it invaluable for network troubleshooting, protocol analysis, and even cybersecurity forensics.

The objective of this lab is to explore how DNS traffic flows through the network. By capturing and analysing DNS packets, we gain insights into how queries are sent to DNS servers and how responses are returned. This is essential knowledge for network professionals and cybersecurity practitioners.

Objectives

* Part 1: Capture DNS Traffic
* Part 2: Explore DNS Query Traffic
* Part 3: Explore DNS Response Traffic

Required Resources

* A PC with internet access
* Wireshark installed and configured

**PART 1: CAPTURE DNS TRAFFIC**

STEPS THAT I FOLLOWED IN THIS CAPTURE

1. Start Wireshark. Select an active interface with traffic for pocket capture
2. Clear the DNS cache

Since I was using Windows, I entered **ipconfig /flushdns** in the command prompt

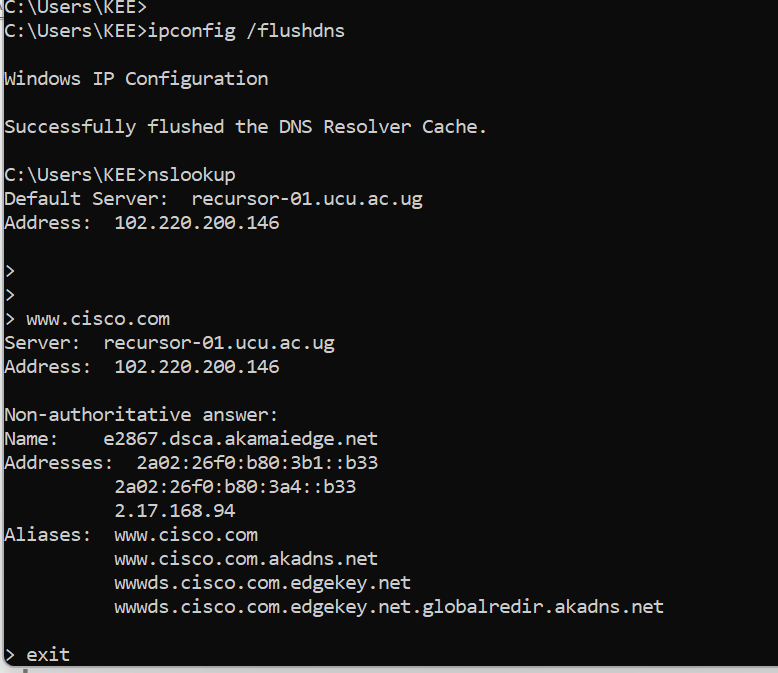
For the majority of Linux distributions, one of the following utilities is used for DNS caching: Systemd -Resolved, DNSMasq, and NSCD. If your Linux distribution does not use one of the listed utilities, please perform an internet search for the DNS caching utility for your Linux distribution.

Identify the utility used in your Linux distribution by checking the status:

1. Systemd-Resolved: **systemctl status systemd-resolved.service**
2. DNSMasq:       **systemctl status dnsmasq.service**
3. NSCD:        **systemctl status nscd.service**

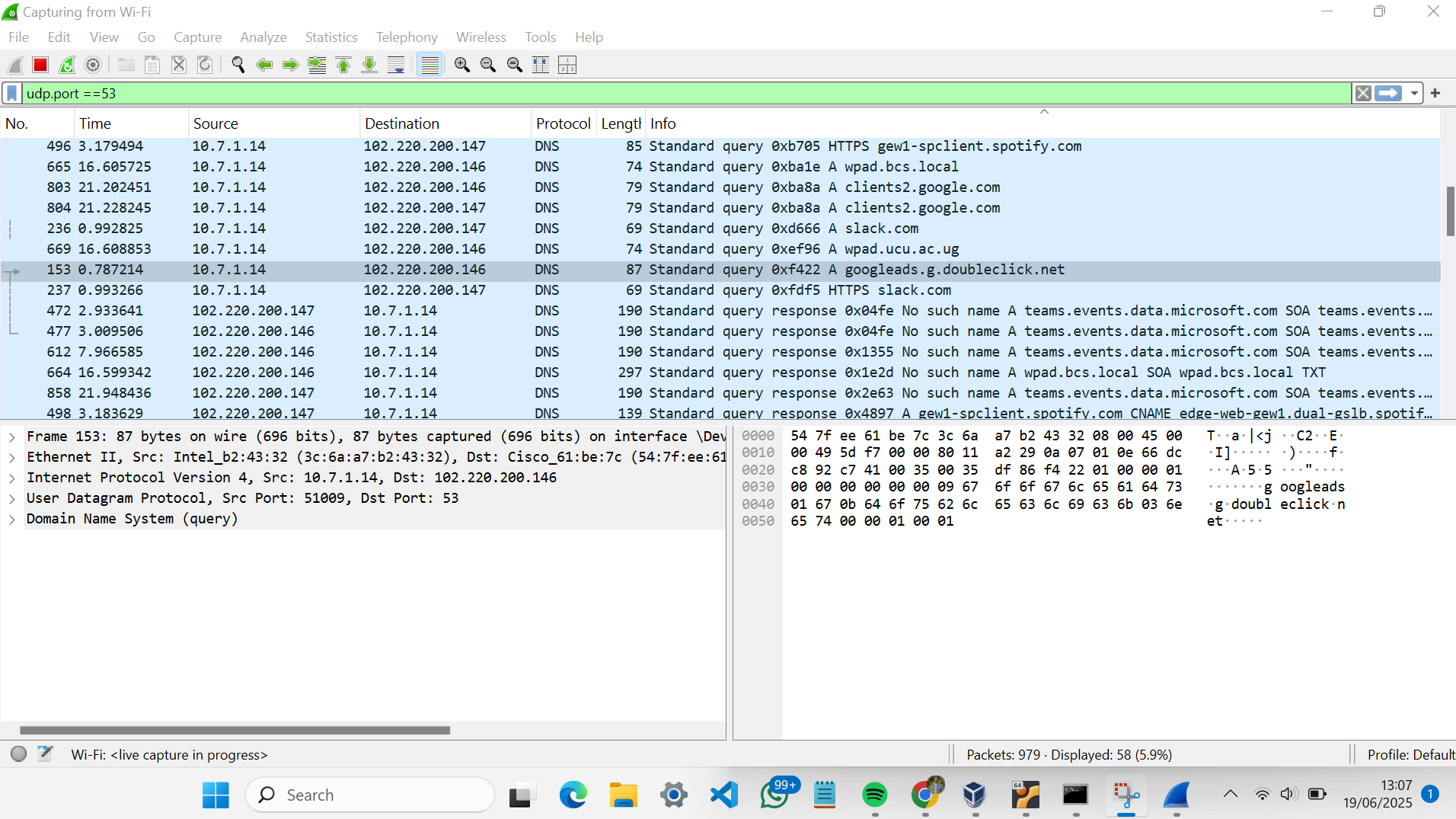
If you are using system-resolved, enter systemd-resolve --flush-caches to flush the cache for Systemd-Resolved before restarting the service. The following commands restart the associated service using elevated privileges:

1. Systemd-Resolved: **sudo systemctl restart systemd-resolved.service**
2. DNSMasq:        **sudo systemctl restart dnsmasq.service**
3. NSCD:        **sudo systemctl restart nscd.service**
4. For the macOS, enter **sudo killall -HUP mDNSResponder** to clear the DNS cache in the Terminal. Perform an internet search for the commands to clear the DNS cache for an older OS.
5. At a command prompt or terminal, type **nslookup** enter the interactive mode.
6. Enter the domain name of a website. The domain name I used in this case was, [www.cisco.com](http://www.cisco.com)
7. Type **exit** when finished. Close the command prompt.
8. Click**Stop capturing packets**to stop the Wireshark capture.

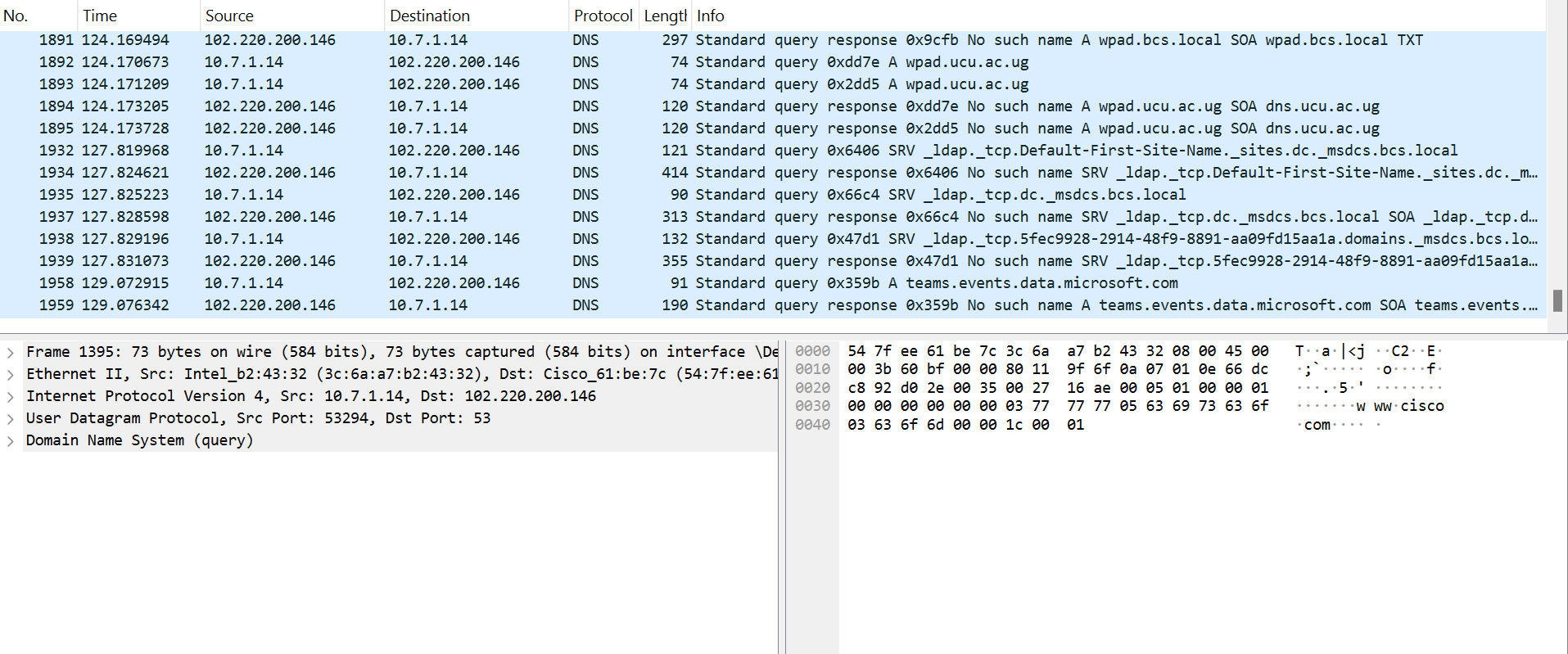


**Part 2: Explore DNS Query Traffic**

1. Observe the traffic captured in the Wireshark Packet List pane. Enter **udp.port == 53** in the filter box and click the arrow (or press enter) to display only DNS packets.**Note:** The provided screenshots are just examples. Your output maybe slightly different.



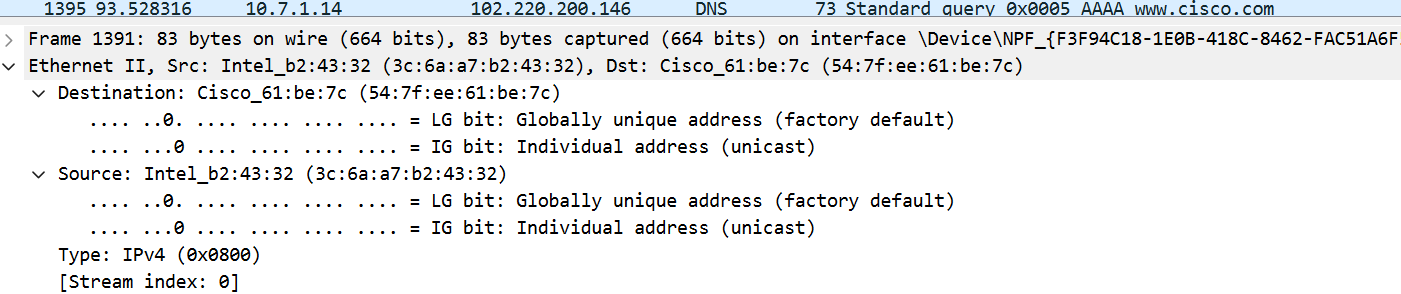
1. Select the DNS packet contains **Standard query** and **A www.cisco.com**in the Info column.
2. In the Packet Details pane, notice this packet has Ethernet II, Internet Protocol Version 4, User Datagram Protocol and Domain Name System (query).
3. Expand **Ethernet II** to view the details. Observe the source and destination fields.

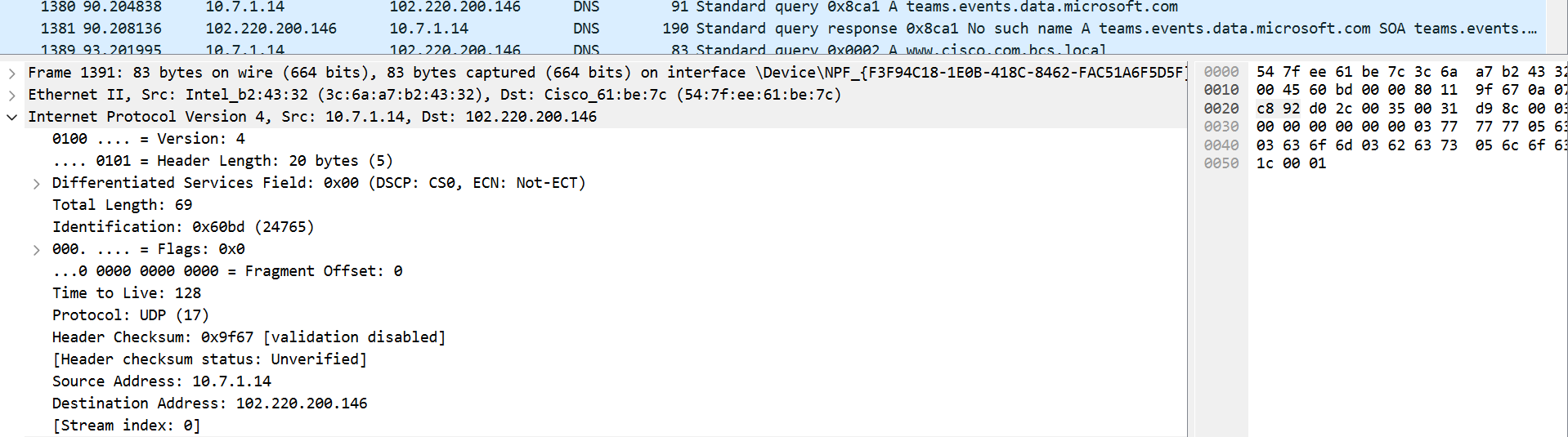


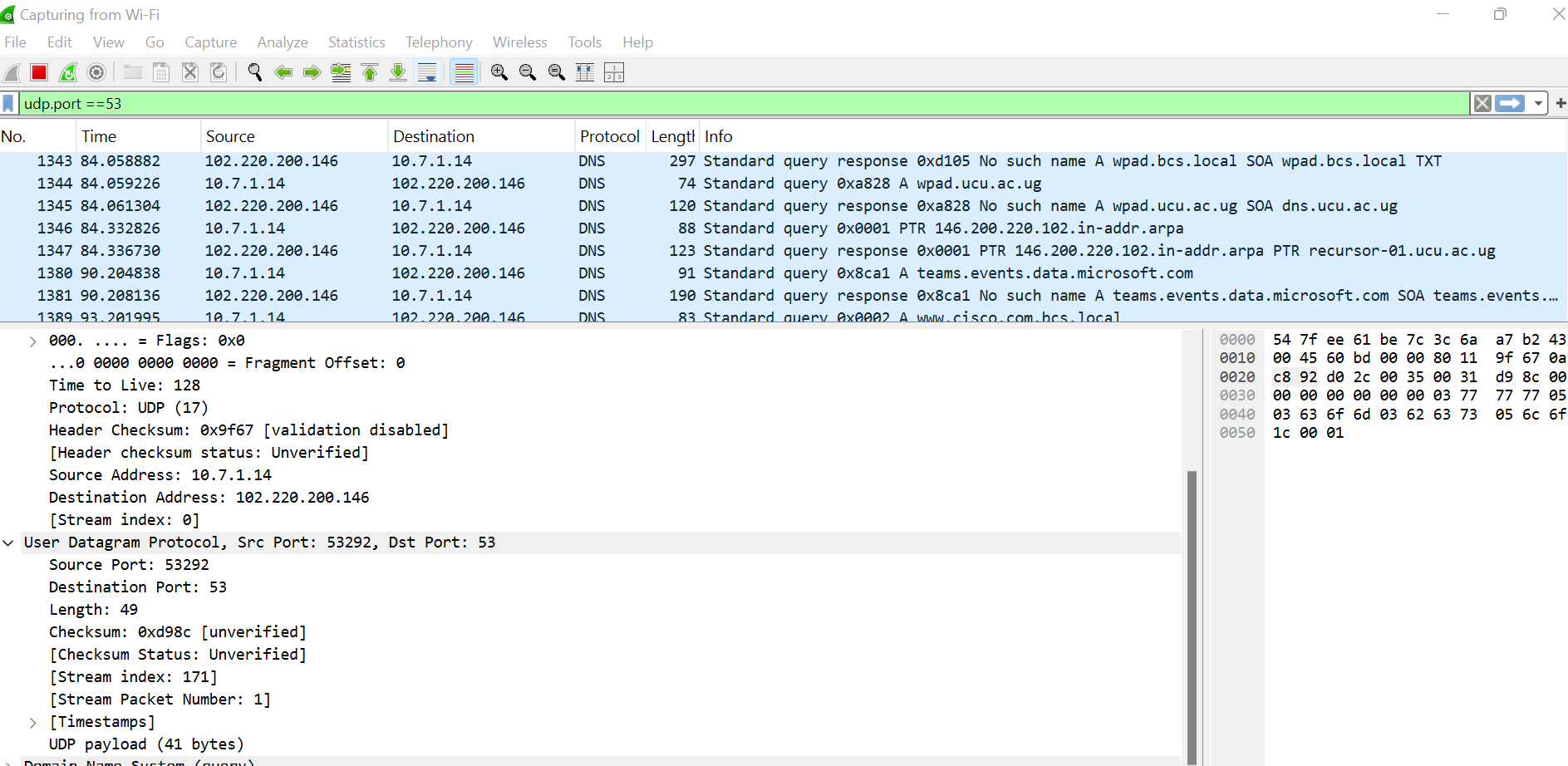
I then identified the source and destination MAC addresses and the network Interfaces whic this MAC addresses are associated with:

NOTE: *In this example, the source MAC address is associated with the NIC on the PC and the destination MAC address is associated with the default gateway. If there is a local DNS server, the destination MAC address would be the MAC address of the local DNS server.*

1. Expand**Internet Protocol Version 4**. Observe the source and destination IPv4 addresses.







SOURCE PORT : 53292

DESTINATION PORT : 53

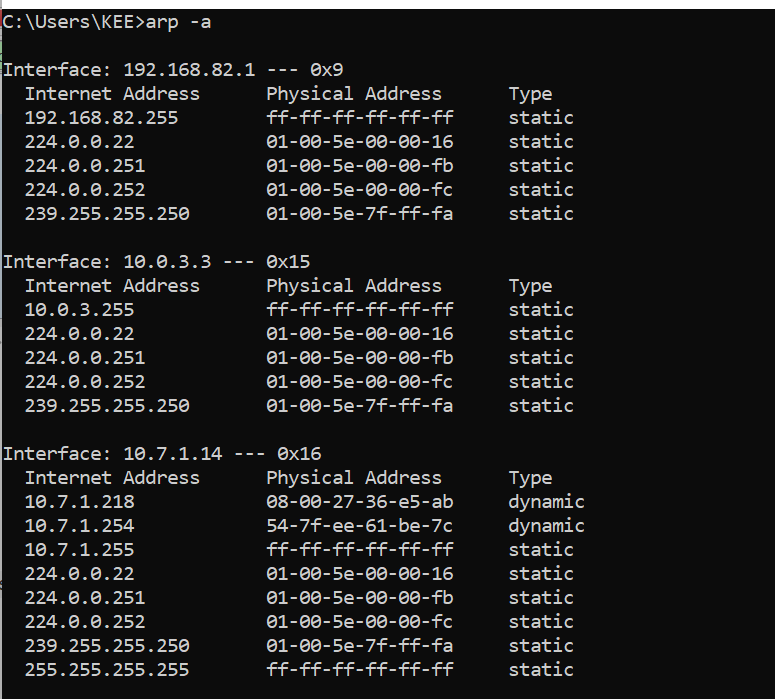
The destination Port is also serving as the default DNS Port number.

DETERMINING THE IP AND MAC ADDRESS OF THE PC.

1. In a Windows command prompt, enter**arp –a** and **ipconfig /all** to record the MAC and IP addresses of the PC.
2. For Linux and macOS PC, enter**ifconfig**or **ip address** in a terminal.

***Comparing the MAC and IP addresses in the Wireshark results to the IP and MAC addresses.***

The observation with this was ***The IP and MAC addresses captured in the Wireshark results are the same as the addresses listed in ipconfig /all command.***

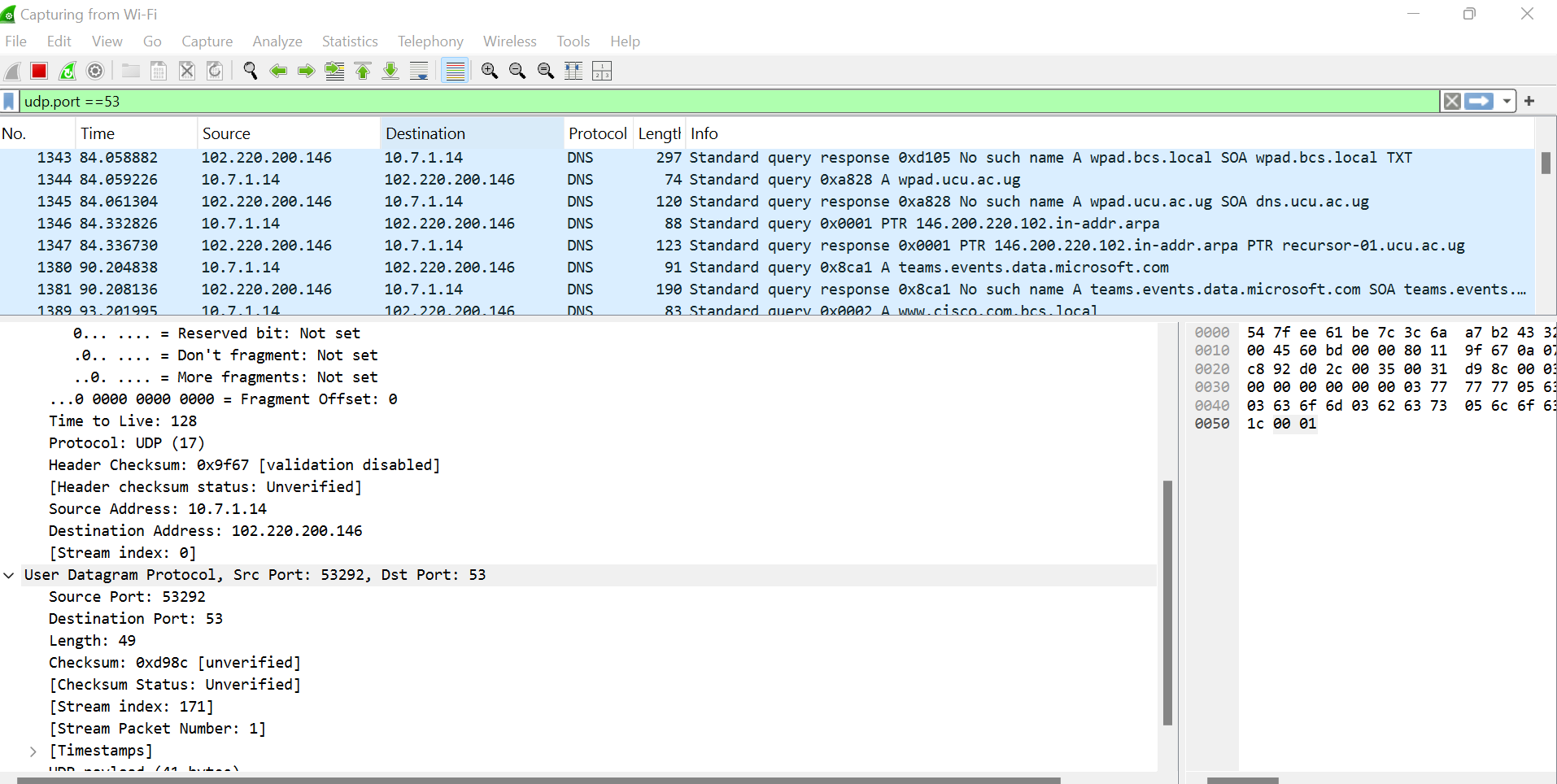


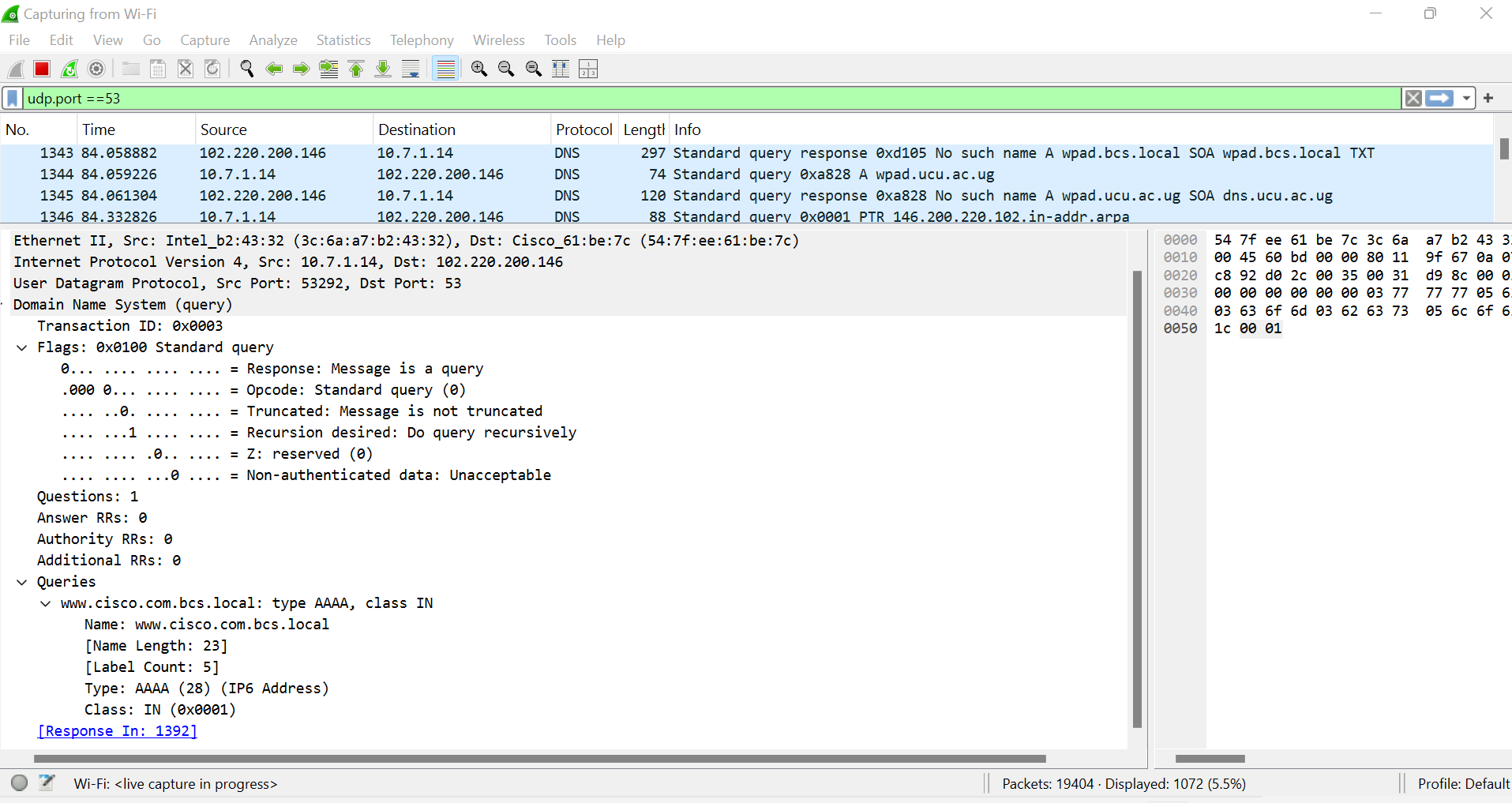
1. Expand**ing Domain Name System (query)** in the Packet Details pane. Then expand the **Flags**and **Queries**.
2. Observe the results. The flag is set to do the query recursively to query for the IP address to [www.cisco.com](http://www.cisco.com).

**PART 3: EXPLORING DNS RESPONSE TRAFFIC**

I selected the corresponding response DNS packet which has standard query response and a A [www.cisco.com](http://www.cisco.com) in the Info column.

The source IP, MAC address, and port number in the query packet are now destination addresses. The destination IP, MAC address, and port number in the query packet are now source addresses***.***





Without the filters, the results display other packets, such as DHCP and ARP. From these packets and the information contained within these packets, you can learn about other devices and their functions within the LAN.

**Conclusion**

If network traffic is not encrypted, an attacker on the same Local Area Network (LAN) can exploit tools like **Wireshark** to capture and inspect packet data in real time. This allows them to observe sensitive information such as domain names, IP addresses, and even credentials or personal data transmitted in plaintext. This emphasizes the critical importance of using encryption protocols—such as **HTTPS**, **DNS over HTTPS (DoH)**, or **VPNs**—to safeguard communication and maintain network confidentiality and integrity