Purpose of Wireshark

Wireshark is a network protocol analyzer. It captures network packets in real time and allows you to inspect them in detail. In simple words, it lets you see what is happening on a network at the packet level.

Network communication happens in small units called packets. These packets contain source and destination addresses, protocols, payload data, and many other fields. Normally, this traffic is invisible to users. Wireshark makes it visible.

Wireshark is commonly used for:

- Troubleshooting network connectivity problems
- Detecting suspicious or malicious traffic
- Understanding how protocols work
- Debugging application communication issues
- Learning networking concepts practically

For example, if a website is not loading, Wireshark can help determine whether the issue is DNS resolution, TCP handshake failure, or server response delay.

How to Approach a Packet Capture

Opening a .pcap file in Wireshark without a plan can be overwhelming. A packet capture may contain thousands or even millions of packets. So, a structured approach is important.

Step 1: Understand the Context

Before analyzing, ask:

- What is the problem?
- What time did it occur?
- What system or IP address is involved?
- Is this normal traffic or suspicious activity?

Without context, analysis becomes guesswork.

Step 2: Check Basic Information

After opening the capture file:

- Look at the **time range**

- Check the **protocol hierarchy** (Statistics → Protocol Hierarchy)
- Identify dominant protocols (DNS, TCP, HTTP, TLS, etc.)
- Observe top talkers (Statistics → Conversations)

This gives a high-level overview of what kind of traffic is inside the capture.

Step 3: Apply Display Filters

Instead of scrolling manually, use display filters:

- dns → shows DNS traffic only
- tcp → shows TCP traffic
- http → shows HTTP traffic
- ip.addr == 10.1.1.97 → filter specific IP
- tcp.port == 443 → filter HTTPS

Filters reduce noise and help focus on relevant packets.

Step 4: Follow Streams

For deeper inspection:

- Right-click a TCP packet
- Click **Follow → TCP Stream**

This reconstructs the full communication between client and server.

Step 5: Analyze Packet Details

Each packet has three panes:

1. Packet List Pane
2. Packet Details Pane
3. Packet Bytes Pane

In the Packet Details pane, expand protocol layers:

- Frame
- Ethernet
- IP
- TCP/UDP

- Application Layer (DNS, HTTP, etc.)

Understanding these layers helps you analyze communication step-by-step.

Protocol Deep Dive – DNS (Domain Name System)

I chose **DNS** because it is one of the most important protocols in network communication and often appears in both normal and malicious traffic.

What is DNS?

DNS translates domain names into IP addresses.

For example:

When you type:

www.google.com

Your system does not understand domain names directly. It needs an IP address like:

142.250.190.78

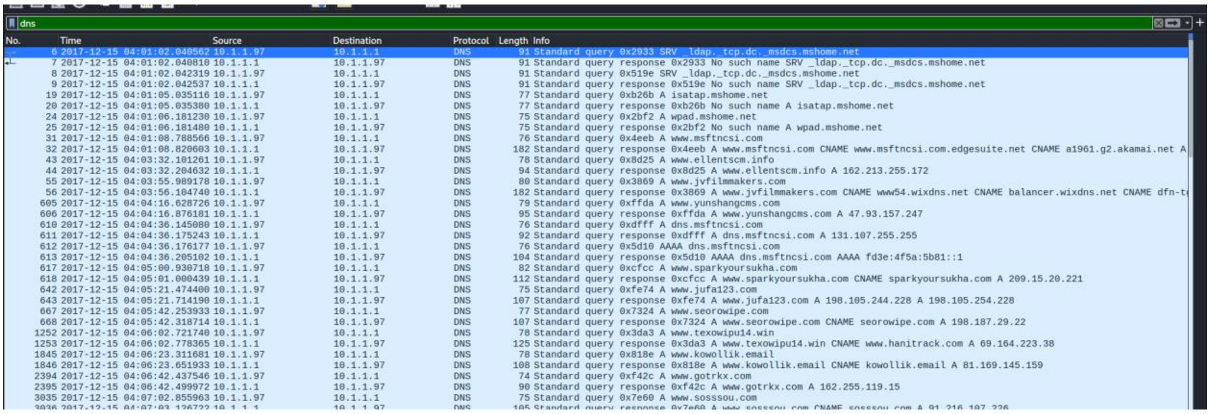
DNS performs this translation.

How DNS Works (Basic Flow)

1. Client sends a DNS query to a DNS server.
2. DNS server responds with the IP address.
3. Client uses that IP to connect to the server.

DNS Packet Structure in Wireshark

For this analysis, I used the packet capture from a malicious email attachment incident. After the attachment was opened, the infected system generated outbound DNS queries to external domains. By applying the dns filter in Wireshark, I identified suspicious domain lookups initiated by the victim machine. These DNS requests appeared immediately after the attachment execution, indicating possible command-and-control communication. This analysis helped in understanding how the malware attempted to establish external connectivity.

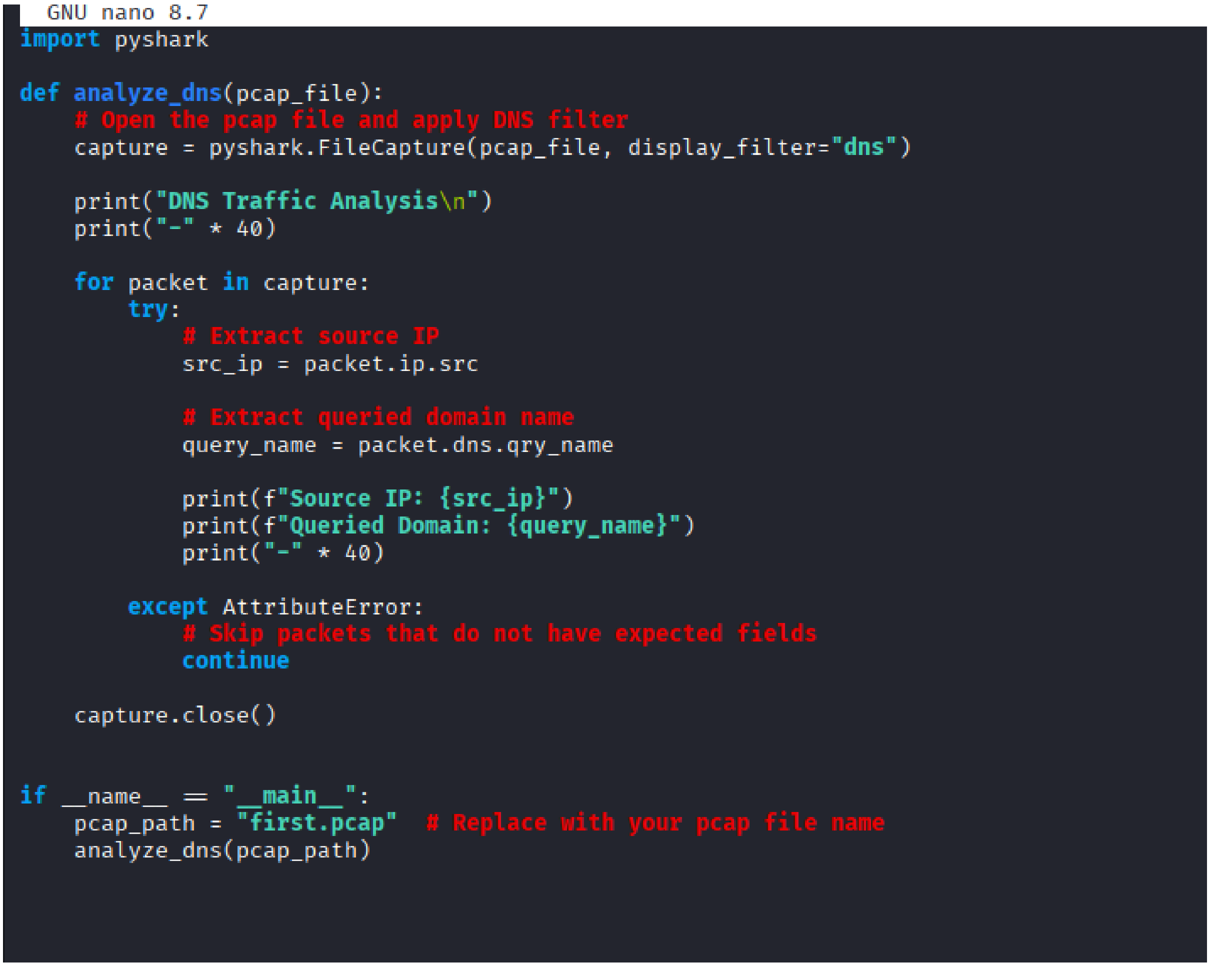


No.	Time	Source	Destination	Protocol	Length	Info
6	2017-12-15 04:01:02.040562	10.1.1.97	10.1.1.1	DNS	91	Standard query 0x2933 SRV _ldap._tcp.dc._msdcs.mshome.net
7	2017-12-15 04:01:02.040810	10.1.1.1	10.1.1.97	DNS	91	Standard query response 0x2933 No such name SRV _ldap._tcp.dc._msdcs.mshome.net
8	2017-12-15 04:01:02.042319	10.1.1.97	10.1.1.1	DNS	91	Standard query 0x519e SRV _ldap._tcp.dc._msdcs.mshome.net
9	2017-12-15 04:01:02.042537	10.1.1.1	10.1.1.97	DNS	91	Standard query response 0x519e No such name SRV _ldap._tcp.dc._msdcs.mshome.net
19	2017-12-15 04:01:05.035116	10.1.1.97	10.1.1.1	DNS	77	Standard query 0xb26b A isatap.mshome.net
20	2017-12-15 04:01:05.035380	10.1.1.1	10.1.1.97	DNS	77	Standard query response 0xb26b No such name A isatap.mshome.net
24	2017-12-15 04:01:06.181230	10.1.1.97	10.1.1.1	DNS	75	Standard query 0x2bf2 A wpad.mshome.net
25	2017-12-15 04:01:06.181480	10.1.1.1	10.1.1.97	DNS	75	Standard query response 0x2bf2 No such name A wpad.mshome.net
31	2017-12-15 04:01:08.788566	10.1.1.97	10.1.1.1	DNS	76	Standard query 0x4eeb A www.msftncsi.com
32	2017-12-15 04:01:08.820603	10.1.1.1	10.1.1.97	DNS	182	Standard query response 0x4eeb A www.msftncsi.com CNAME www.msftncsi.com.edgesuite.net CNAME a1961.g2.akamai.net A
43	2017-12-15 04:03:32.101261	10.1.1.97	10.1.1.1	DNS	78	Standard query 0x8d25 A www.ellentscm.info
44	2017-12-15 04:03:32.204632	10.1.1.1	10.1.1.97	DNS	94	Standard query response 0x8d25 A www.ellentscm.info A 162.213.255.172
55	2017-12-15 04:03:55.989178	10.1.1.97	10.1.1.1	DNS	80	Standard query 0x3869 A www.jvfilmmakers.com
56	2017-12-15 04:03:56.104740	10.1.1.1	10.1.1.97	DNS	182	Standard query response 0x3869 A www.jvfilmmakers.com CNAME www54.wixdns.net CNAME balancer.wixdns.net CNAME dfn-t
605	2017-12-15 04:04:16.628726	10.1.1.97	10.1.1.1	DNS	79	Standard query 0xffda A www.yunshangcms.com
606	2017-12-15 04:04:16.876181	10.1.1.1	10.1.1.97	DNS	95	Standard query response 0xffda A www.yunshangcms.com A 47.93.157.247
610	2017-12-15 04:04:36.145080	10.1.1.97	10.1.1.1	DNS	76	Standard query 0xdfff A dns.msftncsi.com
611	2017-12-15 04:04:36.175243	10.1.1.1	10.1.1.97	DNS	92	Standard query response 0xdfff A dns.msftncsi.com A 131.107.255.255
612	2017-12-15 04:04:36.176177	10.1.1.97	10.1.1.1	DNS	76	Standard query 0x5d10 AAAA dns.msftncsi.com
613	2017-12-15 04:04:36.205102	10.1.1.1	10.1.1.97	DNS	104	Standard query response 0x5d10 AAAA dns.msftncsi.com AAAA fd3e:4f5a:5b81::1
617	2017-12-15 04:05:00.930718	10.1.1.97	10.1.1.1	DNS	82	Standard query 0xcfcc A www.sparkyoursukha.com
618	2017-12-15 04:05:01.000439	10.1.1.1	10.1.1.97	DNS	112	Standard query response 0xcfcc A www.sparkyoursukha.com CNAME sparkyoursukha.com A 209.15.20.221
642	2017-12-15 04:05:21.474400	10.1.1.97	10.1.1.1	DNS	75	Standard query 0xfe74 A www.jufa123.com
643	2017-12-15 04:05:21.714190	10.1.1.1	10.1.1.97	DNS	107	Standard query response 0xfe74 A www.jufa123.com A 198.105.244.228 A 198.105.254.228
667	2017-12-15 04:05:42.253933	10.1.1.97	10.1.1.1	DNS	77	Standard query 0x7324 A www.seorowipe.com
668	2017-12-15 04:05:42.318714	10.1.1.1	10.1.1.97	DNS	107	Standard query response 0x7324 A www.seorowipe.com CNAME seorowipe.com A 198.187.29.22
1252	2017-12-15 04:06:02.721740	10.1.1.97	10.1.1.1	DNS	78	Standard query 0x3da3 A www.texowipu14.win
1253	2017-12-15 04:06:02.778365	10.1.1.1	10.1.1.97	DNS	125	Standard query response 0x3da3 A www.texowipu14.win CNAME www.hanitrack.com A 69.164.223.38
1845	2017-12-15 04:06:23.311681	10.1.1.97	10.1.1.1	DNS	78	Standard query 0x818e A www.kowollik.email
1846	2017-12-15 04:06:23.651933	10.1.1.1	10.1.1.97	DNS	108	Standard query response 0x818e A www.kowollik.email CNAME kowollik.email A 81.169.145.159
2394	2017-12-15 04:06:42.437546	10.1.1.97	10.1.1.1	DNS	74	Standard query 0xf42c A www.gotrkx.com
2395	2017-12-15 04:06:42.499972	10.1.1.1	10.1.1.97	DNS	90	Standard query response 0xf42c A www.gotrkx.com A 162.255.119.15
3035	2017-12-15 04:07:02.855963	10.1.1.97	10.1.1.1	DNS	75	Standard query 0x7e60 A www.sosssou.com
3036	2017-12-15 04:07:03.126722	10.1.1.1	10.1.1.97	DNS	105	Standard query response 0x7e60 A www.sosssou.com CNAME sosssou.com A 91.216.107.226

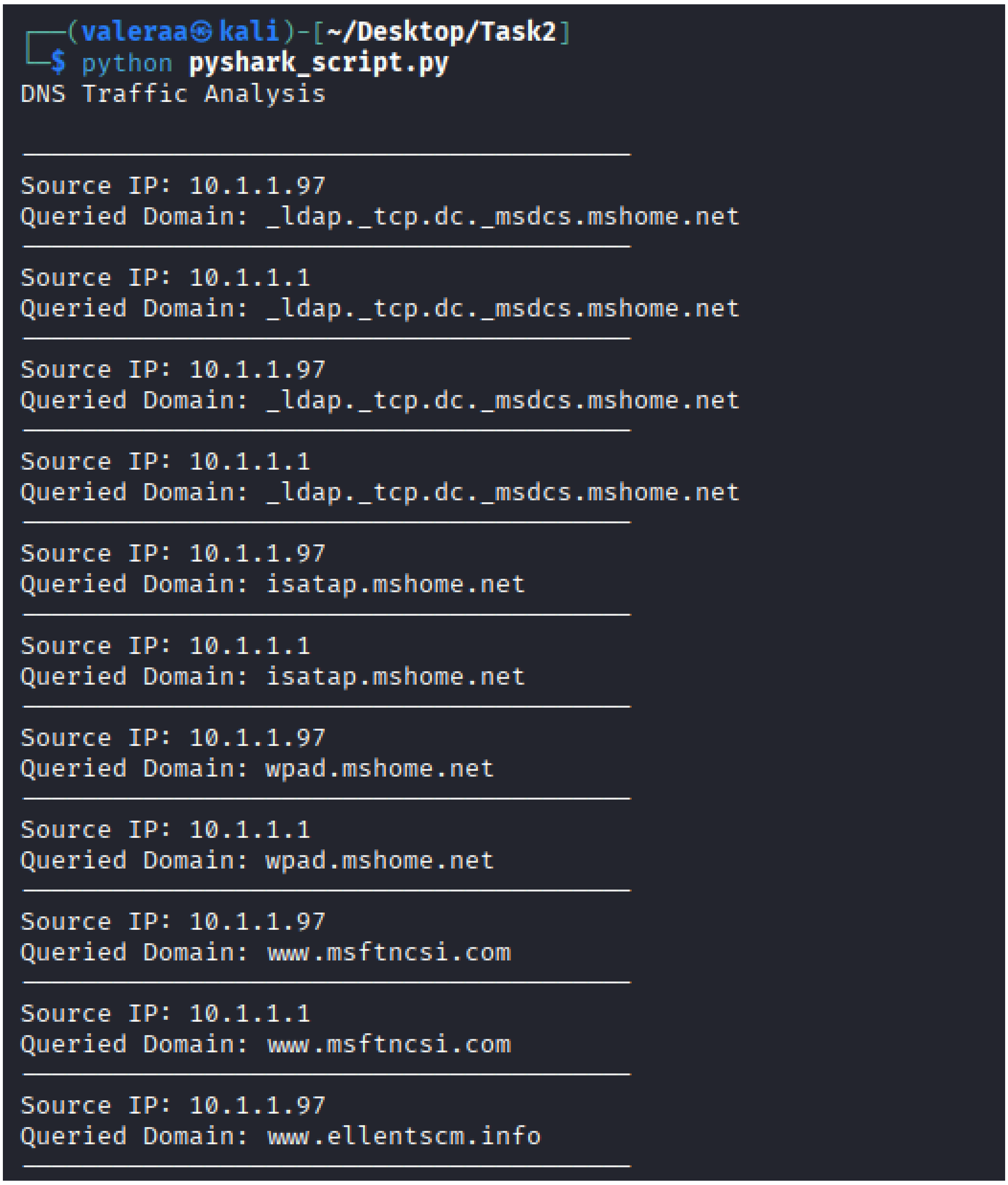
Python Script using PyShark

Below is a simple Python script that:

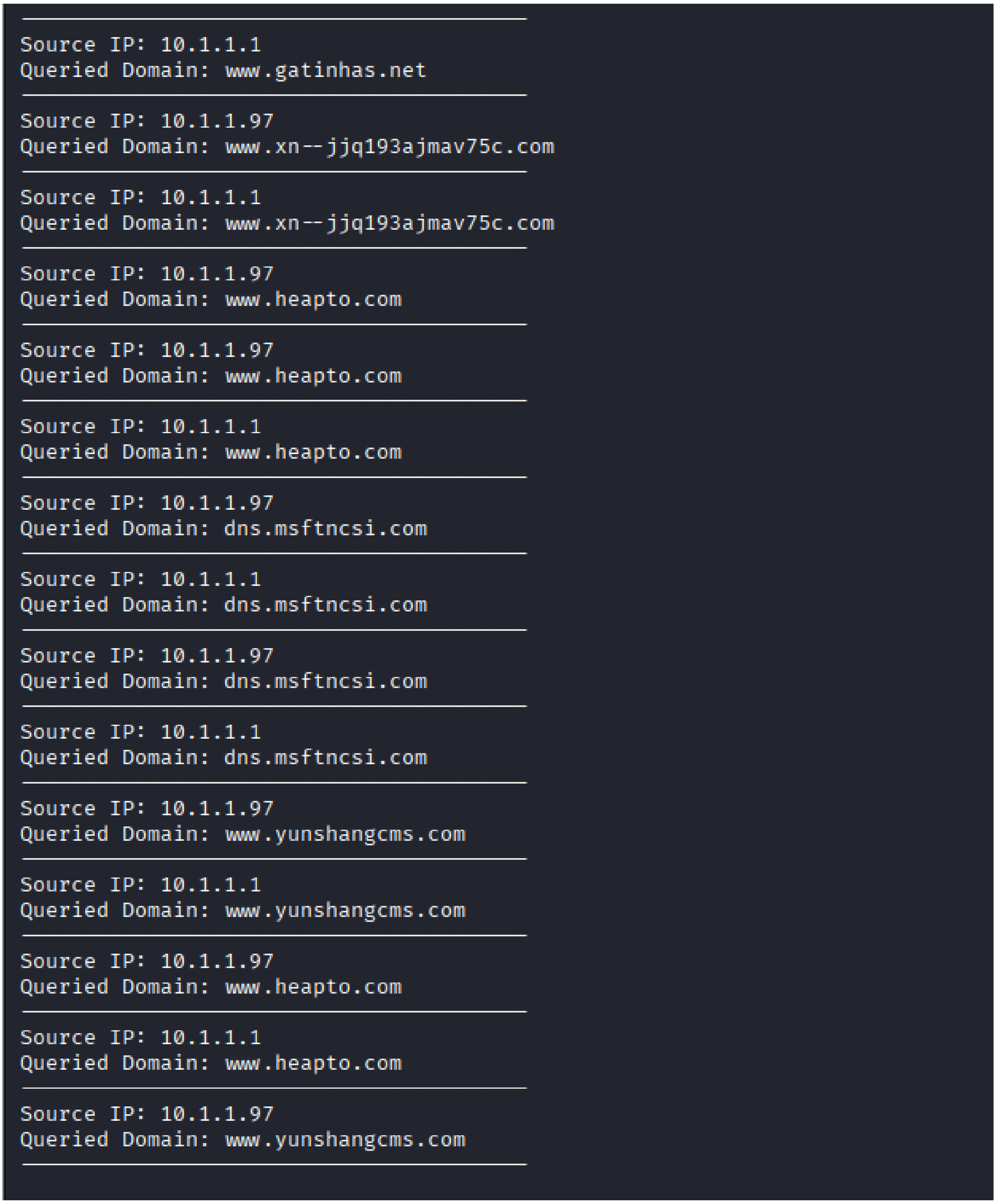
- Opens a .pcap file
- Filters only DNS traffic
- Prints:
  - Source IP
  - Queried domain name



```
import pyshark

def analyze_dns(pcap_file):
    # Open the pcap file and apply DNS filter
    capture = pyshark.FileCapture(pcap_file, display_filter="dns")

    print("DNS Traffic Analysis\n")
    print("-" * 40)

    for packet in capture:
        try:
            # Extract source IP
            src_ip = packet.ip.src

            # Extract queried domain name
            query_name = packet.dns.qry_name

            print(f"Source IP: {src_ip}")
            print(f"Queried Domain: {query_name}")
            print("-" * 40)

        except AttributeError:
            # Skip packets that do not have expected fields
            continue

    capture.close()


if __name__ == "__main__":
    pcap_path = "first.pcap"  # Replace with your pcap file name
    analyze_dns(pcap_path)
```



```
┌──(valeraa㉿kali)-[~/Desktop/Task2]
└─$ python pyshark_script.py
DNS Traffic Analysis

________________________________________
Source IP: 10.1.1.97
Queried Domain: _ldap._tcp.dc._msdcs.mshome.net
________________________________________
Source IP: 10.1.1.1
Queried Domain: _ldap._tcp.dc._msdcs.mshome.net
________________________________________
Source IP: 10.1.1.97
Queried Domain: _ldap._tcp.dc._msdcs.mshome.net
________________________________________
Source IP: 10.1.1.1
Queried Domain: _ldap._tcp.dc._msdcs.mshome.net
________________________________________
Source IP: 10.1.1.97
Queried Domain: isatap.mshome.net
________________________________________
Source IP: 10.1.1.1
Queried Domain: isatap.mshome.net
________________________________________
Source IP: 10.1.1.97
Queried Domain: wpad.mshome.net
________________________________________
Source IP: 10.1.1.1
Queried Domain: wpad.mshome.net
________________________________________
Source IP: 10.1.1.97
Queried Domain: www.msftncsi.com
________________________________________
Source IP: 10.1.1.1
Queried Domain: www.msftncsi.com
________________________________________
Source IP: 10.1.1.97
Queried Domain: www.ellentscm.info
________________________________________
```



```
────────────────────────────────────────
Source IP: 10.1.1.1
Queried Domain: www.gatinhas.net
────────────────────────────────────────
Source IP: 10.1.1.97
Queried Domain: www.xn--jjq193ajmav75c.com
────────────────────────────────────────
Source IP: 10.1.1.1
Queried Domain: www.xn--jjq193ajmav75c.com
────────────────────────────────────────
Source IP: 10.1.1.97
Queried Domain: www.heapto.com
────────────────────────────────────────
Source IP: 10.1.1.97
Queried Domain: www.heapto.com
────────────────────────────────────────
Source IP: 10.1.1.1
Queried Domain: www.heapto.com
────────────────────────────────────────
Source IP: 10.1.1.97
Queried Domain: dns.msftncsi.com
────────────────────────────────────────
Source IP: 10.1.1.1
Queried Domain: dns.msftncsi.com
────────────────────────────────────────
Source IP: 10.1.1.97
Queried Domain: dns.msftncsi.com
────────────────────────────────────────
Source IP: 10.1.1.1
Queried Domain: dns.msftncsi.com
────────────────────────────────────────
Source IP: 10.1.1.97
Queried Domain: www.yunshangcms.com
────────────────────────────────────────
Source IP: 10.1.1.1
Queried Domain: www.yunshangcms.com
────────────────────────────────────────
Source IP: 10.1.1.97
Queried Domain: www.heapto.com
────────────────────────────────────────
Source IP: 10.1.1.1
Queried Domain: www.heapto.com
────────────────────────────────────────
Source IP: 10.1.1.97
Queried Domain: www.yunshangcms.com
────────────────────────────────────────
```

From the output, we can clearly see two internal IP addresses involved:

- **10.1.1.97**
- **10.1.1.1**

This suggests:

- 10.1.1.97 → likely a client machine
- 10.1.1.1 → likely the local DNS server or gateway

We can see a pattern:

Client (10.1.1.97) → DNS Query
DNS Server (10.1.1.1) → DNS Forwarding / Response

Suspicious / Potentially Malicious Domains

We see domains like:

- www.ellentscm.info
- www.jufa123.com
- www.seorowipe.com
- www.texowipu14.win
- www.kowollik.email
- www.sosssou.com
- www.cerebrumfriend.info
- www.heapto.com
- www.xn--jjq193ajmav75c.com

These domains raise red flags because:

Random-looking names
Strange TLDs (.win, .email, .info)
Punycode domain (xn--jjq193ajmav75c.com)
Multiple uncommon domains queried in sequence

This behavior is commonly seen in:

- Adware infections
- Malware beaconing
- Malicious browser extensions
- DNS-based malware callbacks