Wireshark Analysis

26th November 2024

COMP3343 – Network Project Fall 2024

Table of contents

1. Introduction
2. Analyzing Internet Traffic
   1. Domain Name System
   2. Internet Protocols / Packets Analysis

2.2.1 Address Resolution Protocol

2.2.2 Transmission Control Protocol & Internet Protocol

2.2.3 Transport Layer Security

2.2.3.1 QUIC

2.2.4 User Datagram Protocol

2.3 Sample Malware Traffic Analysis

2.4 Decoding Packets

1. References

# Introduction

# Analyzing Internet Traffic

We’ve gathered a sample internet traffic in Wireshark using Wi-Fi interface in my home. When live capturing packets, I also made sure to do simple searches on the web. Based on the traffic captured, the traffic mainly consists of packets going through 2 protocols, User Diagram Protocol (UDP) & Transmission Control Protocol (TCP).

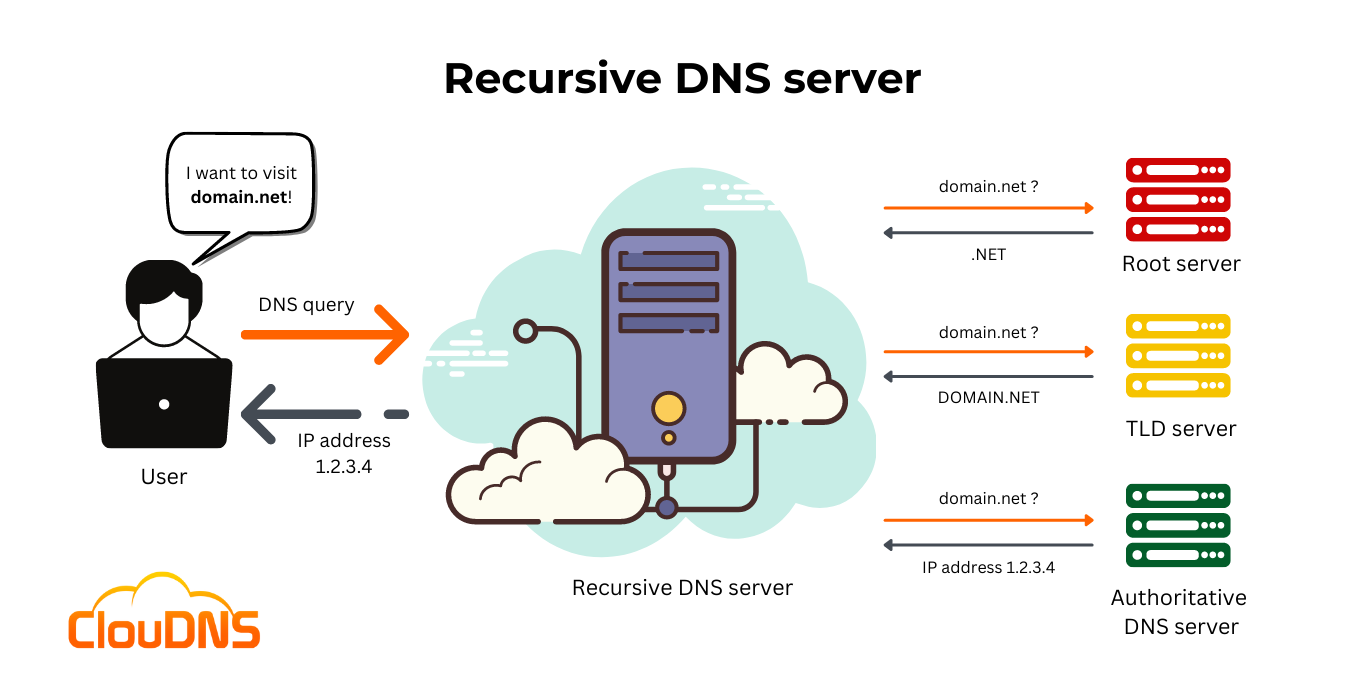
A screen shot of a computer

Description automatically generated

Upon further inspection, captured network traffic reveals a predominance of packets utilizing two primary protocols: User Datagram Protocol (UDP) & Transmission Control Protocol (TCP). Upon closer analysis, I noticed that a few traffic queries are sent to a server called gstatic.com using DNS protocol. These protocols serve distinct purposes as in network communication whereas TCP are designed to establish reliable and connection-oriented channel for data exchange while UDP focuses on speed over reliability which suits real time applications. Apart from these 2 protocols, notice there’s also some traffic using DNS protocol which plays a vital role in web browsing, email, cloud services and many more. Before we start looking through UDP TCP packets, we first understand how DNS queries works.

* 1. Domain Name System

Domain Name System, also known as the phonebook of the Internet. The idea is that it maps human readable URLs or host names to the IP addresses of the server that hosts that site. For example, when you type a URL or a name into the browser, it’ll make a DNS query to figure out which unique IP address is associated with the URL. But before it send querys to request information to other servers, it first checks the cache of your browser, if there’s isn’t, then it’ll check the phonebook which is the job of Recursive DNS resolver.



As the name suggests, this server sends multiple requests to other servers, starting with the root server, TLD server then authoritative DNS server.

* Root server
  + This server doesn’t store any actual website data but instead they know the next server you should head to get this information. Think of this server as a navigation app.
* Top level domain (TLD) server
  + This server is a root server for all domain names (.com, .org, .net, .edu), and contains the location of all TLDs servers. It can direct you to them. For example, you’re looking for google.com, root server will direct you to a “.com” server which is an authoritative server, where the server contains the actual IP address.
* Authoritative DNS server
  + This server contains all IP address of domain names and it provides the final response to a query about a domain name.
  1. Protocols / Packets analysis

The captured network traffic, as analyzed using Wireshark, provides a detailed breakdown of each packet. The information for each packet is presented in the following format as shown below:

[Number of packets], [Time], [Source], [Destination], [Protocol], [Length], [Info].

A screenshot of a computer

Description automatically generated

Sample capture

To get a summary of what our captured traffic mostly consists of, we take a look at Statistic protocol hierarchy as shown below.

A screenshot of a computer

Description automatically generated

Protocol Hierarchy

Our traffic mostly consist of IPv4 and IPv6 traffic, where the most dominant protocol used are UDP and TCP, with a few packets using ARP, DNS and TLS.

2.2.1 Address Resolution Protocol

Upon closer inspection, we noticed there’s a protocol which contains very human readable information. And all the packets use a protocol called ARP.

* Address Resolution Protocol (ARP): This protocol translates IP address to MAC address so LAN endpoints can communicate with one another. This usually happens within a smaller area network. For example, at home you have a computer who could send data to your printer, the computer would try to locate the IP address of your printer first, once a connection is established, user could transfer data between these devices.

A screenshot of a computer

Description automatically generated

The filtered network traffic reveals a series of ARP (Address Resolution Protocol) packets. In particular,

* Who has 192.168.2.102? Tell 192.168.2.1

Packet 251 is an ARP request broadcast by a device with the MAC address SagemcomBroa\_6f:82:bc. This device is seeking the MAC address of the device with the IP address 192.168.2.102.

* 192.168.2.102 is at cc:2f:71:dd:ea:50

In response, packet 252 is an ARP reply from a device with the IP address 192.168.2.1 (likely a router). This device provides its MAC address, encrypted as cc:2f:71:dd:ea:50.

This ARP exchange demonstrates the fundamental process of mapping IP addresses to MAC addresses on a local network, allowing devices to communicate with each other.

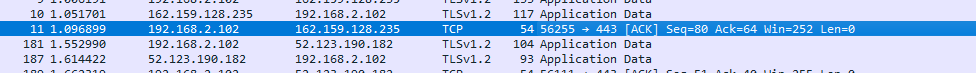
As for the remaining packets in the capture are ARP requests searching for a device with the IP address 192.168.2.215. However, no ARP replies were received, indicating that this device is likely not present on the network.

2.2.2 Transmission Control Protocol / Internet Protocol

TCP, a protocol standard that enables communication between devices and applications, where IP, a unique identifier assigned to each device, this made communications between other devices possible. There are two mainly types of IP versions:

* IPv4
  + 32-bit address in the format of 192.0.2.146 where each column can go up to 255. This 32-bit address allows up to 4 billion unique addresses
* IPv6
  + 128-bit address in eight columns format using hexadecimal digits (2001:0000:130F:0000:0000:09C0:876A:130B). This allows up to trillions of unique addresses, way more than IPv4 addresses.

TCP and IP work together to provide a reliable and efficient way for devices to communicate with each other. Often abbreviated as TCP/IP. Let’s take a closer look at sample TCP packet.



We already know what the first 5 columns are. Let’s understand what the data is,

* 54 – Length of the full packet in bytes
* 56255 🡺 443 – Source port to destination port
* [ACK] – acknowledgement packet, typically used to confirm that receiver successfully received the data sent by the sender.
* Seq=80 / Ack=1 – sequence number of the packet and acknowledgement number.
* Win=252 – size of data its willing to accept before it acknowledges the data received. Usually located in the header of packet. One of the most crucial aspects in data flow control.
* Len=0 – payload length of the length. Different from length of packets. Length of packets include header all the other extra information, payload length is the actual data length.

2.2.3 Transport Layer Security

In addition to TCP protocol packets, we also observed several packets using the TLSv1.2 protocol, primarily involving handshakes, authentication methods, ciphering, and other related processes. There’s many more version of TLS but we’re focused on TLSv1.2.

A screen shot of a computer

Description automatically generated

TLS is a protocol designed to ensure privacy and data security over a network. It is commonly used for securing internet connection and other services such as email, money transfer and web browsing. However, the packet information shown here does not provide much insight as the data is encrypted.

Also known as cryptographic protocol, TLS established a connection between a client and a server through a process called handshake:

* + 1. Client initiates a connection to the server.
    2. Server response with a digital certificate.
    3. Client identifies the certificate to ensure its validity and confirms that it was issued by a trusted Certificate Authority (CA).
    4. Both client and server generate a shared secret key using public key cryptography.

Following the handshake, encryption takes place: both hosts use symmetric-key cryptography to encrypt and decrypt data. Only the client and the server can decrypt the data using the shared key, ensuring the confidentiality of the information being transferred.

Now a secure connection is established, to resume it,

1. The client sends a “Client hello” message together with session ID to the server.
2. Server verification: Server checks its cache folder whether a matching ID is found, if a match is found the server will send a “Server Hello” message with session ID. (If a match not found, it’ll have to perform the full handshake)
3. Encryption Key exchange: Change the encryption keys with each other by exchanging messages “Change Cipher spec”, “Client finished”, “Server finished”.
4. Client and server resume application data exchange.

We can clearly see this happening in packets 2303, 2305, 2306.

2.2.3.1 QUIC

The green-highlighted rows in WireShark capture indicate packets utilizing the QUIC protocol. These packets exhibit distinctive characteristics, including unique ID numbers and keywords like “CRYPTO”, “PADDING”, and “PING” in the information column. QUIC or Quick UDP Internet Connection is a modern network protocol designed to improve [performance and security of internet connections. Similar to TLS, it establishes encrypted, connection-oriented communication. However, unlike traditional TLS, which relies on TCP, QUIC leverages UDP, a connectionless protocol. This fundamental difference simplifies the protocol’s operation, resulting in faster connection establishment and reduced latency. QUIC ewmploys TLS 1.3 fpr encryption and authentication, this shows some resemblance of TLS traffic, this is my assumption on why its shown under the filter of TLS even though it operated in UDP.

A diagram of a server and a server

Description automatically generated

Comparison between TLS & QUIC

QUIC optimizes the connection setup process by simultaneously exchanging digital certificates, sharing public keys, and generating a shared symmetric key. This merging approach minimized the required network interactions, resulting in faster connection establishment. Besides, QUIC encrypts a greater amount of data than TLS, providing a more robust and secure communication channel.

A screenshot of a computer program

Description automatically generated

QUIC encrypts more data than TLS

Both QUIC and TCP protocols use unencrypted headers to transmit essential control information. However, QUIC encrypts a larger portion of the actual data payload, enhancing security compared to TCP. Moreover, QUIC frames include more metadata, providing richer information about the data being transferred. We can confirm this by comparing it to the packet captured in our sample in Wireshark.

2.2.4 User Datagram Protocol

According to our protocol hierarchy summary, packets using IPv4 are primarily composed of 95.9% of UDP traffic. User Datagram Protocol is a communications protocol designed for time-sensitive applications. Its main advantage lies in its speed and efficiency. Unlike other protocols, which need to establish a connection before transferring data, UDP operates without a connection allowing it to transmit data much faster.

The process behind UDP is straightforward. First, it confirms the destination address and then sends data packets called datagram, to the target. Unlike TCP, which relies on acknowledgement packets and a defined window size to manage dataflow and ensure reliable delivery, UDP does not include any of these mechanisms. Once the datagrams are sent, there is no further tracking or confirmation, they are simply transmitted without any follow-up.

This makes UDP an unreliable protocol. While it excels in speed, it comes with risks, as the packets sent can be lost, duplicated, or corrupted during transmission. Although it prioritizes speed and efficiency, it sacrifices reliability and data integrity. Below is a summary comparison between TCP and UDP.

|  |  |  |
| --- | --- | --- |
|  | TCP | UDP |
| Connection-oriented | Yes | No |
| Reliable | Yes | No |
| Speed | Slow | Fast |
| Congestion control | Yes | No |
| Order of delivery | Guaranteed | Not guaranteed |
| Real life applications | Web Browsing, Email, File transfer | Streaming services, Gaming platforms, routing update protocols |

Ultimately, both protocols have their advantages and disadvantages. For instance, TCP is like a reliable postal service that ensures your packages and letters arrive in the correct destination without any damage, but it takes more time. On the other hand, UDP is like sending a postcard, it’s fast, but it sacrifices reliability and makes no guarantees that it will reach the intended destination.

* 1. Malware Traffic Analysis

Now we understand protocols, we will study how to identify abnormal traffic on Wireshark. A open a sample pcap file and alert file that contains the malware.

A screenshot of a computer

Description automatically generated

Image shows the alert file

There are several alerts with different types of alerts. The format are as follows:

[Total count] [Date/Time] [Source IP] [Source Port] [Destination IP] [Destination Port] [Protocol] [Event Message].

Now we investigate the suspicious event message that could tell us any information about any malware. Notice how at the last row, message stating TROJAN Win32/Koi Stealer CnC checkin (POST) M2. The word TROJAN is already suspicious enough, lets look up the web and see if we can find any information related to Win32/Koi Stealer.

I came across a web page (pygrum) providing information about a malware that steals sensitive data from infected computers, including system information, browsing history, and private files. We have gathered details about this malware to gain a deeper understanding of its behavior and impact.

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A close-up of a text

Description automatically generated

Theoretically if we can identify any relevant details related to the malware, such as the ID, server URL, or source format, we can pinpoint the packet that needs to be mitigated. Since we know that the TLSv1.2 protocol performs a handshake with websites to establish a secure connection, let’s apply this filter and see what we can uncover.

A close-up of a screen

Description automatically generated

Among the packets, we observed several establishing secure connections to servers such as “Microsoft.com”, “godaddy.com”, “login.microsoft.com”, and “officeapps.live.com”. Additionally, we noticed some requests directed to the server “www.bellantonicioccolato.it”, which is one of the pieces of information we uncovered during our web searches.

This is the first piece of evidence indicating that malware is present in these packet captures. Next, we can gather additional information by exporting HTTP objects.

A screenshot of a computer

Description automatically generated

A window will appear displaying a list of packets containing HTTP objects. At that point, I immediately noticed the keyword “subid” which is another piece of information we found during our web search, appearing as a parameter in a download string.

A screenshot of a computer

Description automatically generated

HTTP Object List

We could select that and save that in your local machine and open it using a text file. The text.file saved should look like this:

A screen shot of a computer

Description automatically generated

Subid file

Though the IP is not the same as the server endpoint we found on the web, we can try look up a website scanner for this URL provided

A blue and black rectangular object with red text

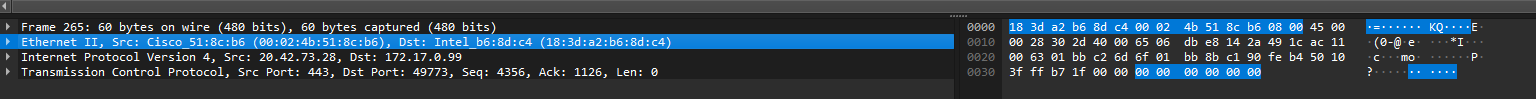
Description automatically generated with medium confidence

As we can see this URL is malicious with a score of only 11 out of 96.

To conclude, these are just a few examples of how we can detect malware in internet traffic. While we could go further by writing code to decrypt the payload, that’s not the main focus here. However, these findings provide valuable insights, as we now understand that while the TLS protocol is used to establish secure connections, it can also be exploited to transmit malicious packets through those connections.

* 1. Decoding packets

Select a random packet to view the packet’s details and packets bytes.



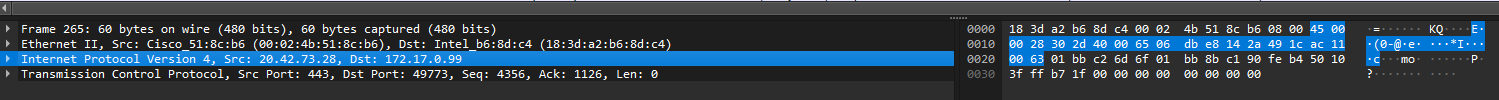
The first 28 bytes are dedicated to the ethernet frame, with the first 12 bytes allocated for destination MAC address, and the next 12 bytes for the source MAC address and the last 4 bytes indicating the protocol type and version used. The highlighted bytes in the last 12 bytes are padding, added to ensure the total packet length aligns with the required bit size. We can confirm this by expanding the drop-down arrow which shows the details.

A screen shot of a computer

Description automatically generated

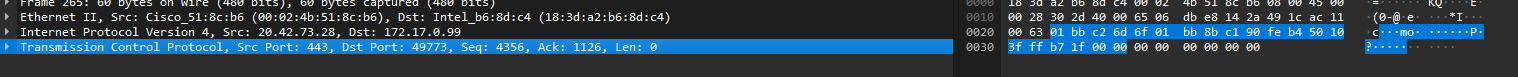
Ethernet

Next, it’s the internet protocol bytes 36 bytes (18x2)



Internet Protocol

The last part of the bytes stream is the TCP protocol bytes, the first 8 bytes are the destination port and source port. Next 8 bytes are for sequence number, in our case its 4356. Next 8 bytes are the acknowledgement number (1126). 2 bytes for header length, 4 bytes for window size (4194048). 4 bytes for checksum values.



TCP

By looking at the sequence of the bytes and how there are arranged, we can derive the following formula:

* 0 – 12 bytes: destination MAC address
* 13 – 24 bytes: source MAC address
* 25 – 28 bytes: types & protocol used
* 29 - 30 bytes: Header length
* 31 -34 bytes: Total packets length
* 35 – 38 bytes: Identification
* 39 – 42 bytes: Flags & fragments
* 43 – 46 bytes: Connection time & protocol
* 47 - 50 bytes: Header checksum
* 51 – 58 bytes: Source Address
* 59 – 66 bytes: Destination address
* 67 – 70 bytes: Source port
* 71 – 74 bytes: Destination port
* 75 - 82 bytes: Relative sequence number & raw sequence number
* 83 - 90 bytes: Acknowledgement number
* 91 – 92 bytes: TCP header length
* 93 – 96 bytes: Acknowledgement flags
* 97 – 100 bytes: window scaling factor, window size.
* 101 - 104 bytes: Checksum.
* 105 – 120 bytes: Padding

Note: All bytes are encoded in hexadecimal or “Hex” which is a base 16-system. The formula derived refers to the placement of packets bytes displayed in Wireshark. Additionally, we can view the encrypted version alongside the bytes.

References:

1. Pramatarov, M. 2024. What is a Recursive DNS server. ClouDNS.

<https://www.cloudns.net/blog/recursive-dns-server/>

1. Alvinashcraft. V-kents. DCtheGeek. Drewbatgit. Msatranjr. 2021. TLS Handshake Protocol. Windows App Development.

<https://learn.microsoft.com/en-us/windows/win32/secauthn/tls-handshake-protocol>

1. Haynes, R. 2023. A Primer on QUIC Networking and Encryption in NGINX. NGINX.

<https://www.f5.com/company/blog/nginx/primer-quic-networking-encryption-in-nginx>

1. What is User Datagram Protocol (UDP)?. FORTINET

<https://www.fortinet.com/resources/cyberglossary/user-datagram-protocol-udp#:~:text=User%20Datagram%20Protocol%20(UDP)%20is,destination%20before%20transferring%20the%20data>.

1. TRAFFIC ANALYSIS EXERCISE: BIG FISH IN A LITTLE POND. Malware-traffic-analysis.net.

<https://malware-traffic-analysis.net/2024/09/04/index.html>

1. A Destruction of Koi Stealer – Malware Analyst. Pygrum.

<https://pygrum.github.io/posts/koi-stealer/>

1. VIRUSTOTAL (Website Scanner)

<https://www.virustotal.com/gui/home/upload>