Wireshark Analysis

26th November 2024

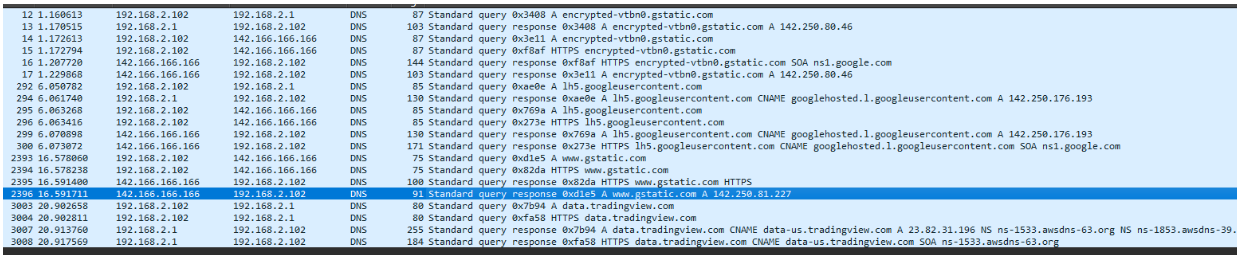
COMP3343 – Network Project Fall 2024

Table of contents

1. Introduction

2. 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).

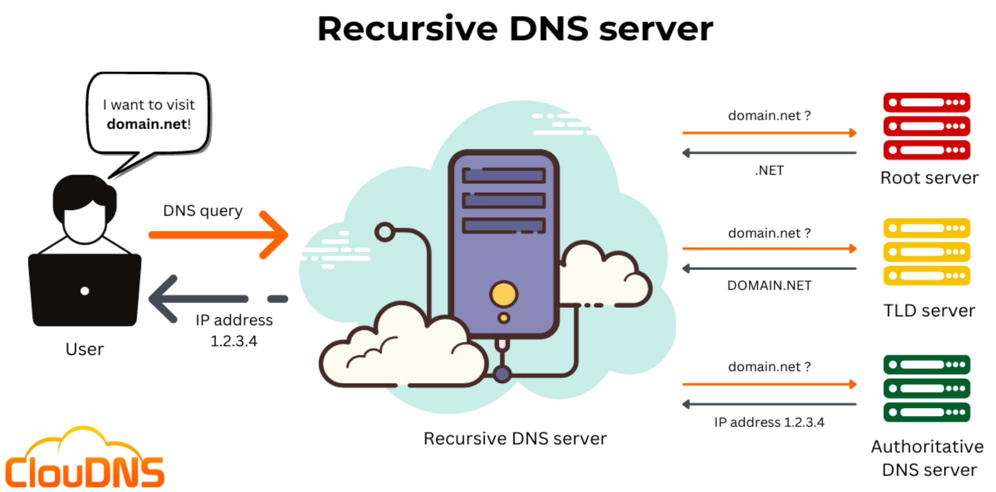


12	1.160613	192.168.2.102	192.168.2.1	DNS	87	Standard query 0x3408 A encrypted-vtbn0.gstatic.com
13	1.170515	192.168.2.1	192.168.2.102	DNS	103	Standard query response 0x3408 A encrypted-vtbn0.gstatic.com A 142.250.80.46
14	1.172613	192.168.2.102	142.166.166.166	DNS	87	Standard query 0x3e11 A encrypted-vtbn0.gstatic.com
15	1.172794	192.168.2.102	142.166.166.166	DNS	87	Standard query 0xf8af HTTPS encrypted-vtbn0.gstatic.com
16	1.207720	142.166.166.166	192.168.2.102	DNS	144	Standard query response 0xf8af HTTPS encrypted-vtbn0.gstatic.com SOA ns1.google.com
17	1.229868	142.166.166.166	192.168.2.102	DNS	103	Standard query response 0x3e11 A encrypted-vtbn0.gstatic.com A 142.250.80.46
292	6.050782	192.168.2.102	192.168.2.1	DNS	85	Standard query 0xae0e A lh5.googleusercontent.com
294	6.061740	192.168.2.1	192.168.2.102	DNS	130	Standard query response 0xae0e A lh5.googleusercontent.com CNAME googlehosted.l.googleusercontent.com A 142.250.176.193
295	6.063268	192.168.2.102	142.166.166.166	DNS	85	Standard query 0x769a A lh5.googleusercontent.com
296	6.063416	192.168.2.102	142.166.166.166	DNS	85	Standard query 0x273e HTTPS lh5.googleusercontent.com
299	6.070898	142.166.166.166	192.168.2.102	DNS	130	Standard query response 0x769a A lh5.googleusercontent.com CNAME googlehosted.l.googleusercontent.com A 142.250.176.193
300	6.073072	142.166.166.166	192.168.2.102	DNS	171	Standard query response 0x273e HTTPS lh5.googleusercontent.com CNAME googlehosted.l.googleusercontent.com SOA ns1.google.com
2393	16.578060	192.168.2.102	142.166.166.166	DNS	75	Standard query 0xd1e5 A www.gstatic.com
2394	16.578238	192.168.2.102	142.166.166.166	DNS	75	Standard query 0x82da HTTPS www.gstatic.com
2395	16.591400	142.166.166.166	192.168.2.102	DNS	100	Standard query response 0x82da HTTPS www.gstatic.com HTTPS
2396	16.591711	142.166.166.166	192.168.2.102	DNS	91	Standard query response 0xd1e5 A www.gstatic.com A 142.250.81.227
3003	20.902658	192.168.2.102	192.168.2.1	DNS	80	Standard query 0x7b94 A data.tradingview.com
3004	20.902811	192.168.2.102	192.168.2.1	DNS	80	Standard query 0xfa58 HTTPS data.tradingview.com
3007	20.913760	192.168.2.1	192.168.2.102	DNS	255	Standard query response 0x7b94 A data.tradingview.com CNAME data-us.tradingview.com A 23.82.31.196 NS ns-1533.awsdns-63.org NS ns-1853.awsdns-39.
3008	20.917569	192.168.2.1	192.168.2.102	DNS	184	Standard query response 0xfa58 HTTPS data.tradingview.com CNAME data-us.tradingview.com SOA ns-1533.awsdns-63.org

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.

2.1 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.

2.2 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].

	Time	Source	Destination	Protocol	Length	Info
1	0.000000	66.22.231.39	192.168.2.102	RTCP	94	Receiver Report
2	0.302258	192.168.2.102	66.22.231.39	RTCP	70	Receiver Report
3	0.894602	192.168.2.102	92.38.168.199	TCP	1514	56275 → 443 [ACK] Seq=1 Ack=1 Win=255 Len=1460 [TCP
4	0.894602	192.168.2.102	92.38.168.199	TLSv1.2	181	Application Data
5	0.932669	92.38.168.199	192.168.2.102	TCP	54	443 → 56275 [ACK] Seq=1 Ack=1588 Win=250 Len=0
6	0.939298	92.38.168.199	192.168.2.102	TLSv1.2	338	Application Data
7	0.986840	192.168.2.102	92.38.168.199	TCP	54	56275 → 443 [ACK] Seq=1588 Ack=285 Win=254 Len=0
8	0.998784	66.22.231.39	192.168.2.102	RTCP	94	Receiver Report
9	1.006191	192.168.2.102	162.159.128.235	TLSv1.2	133	Application Data
10	1.051701	162.159.128.235	192.168.2.102	TLSv1.2	117	Application Data
11	1.096899	192.168.2.102	162.159.128.235	TCP	54	56255 → 443 [ACK] Seq=80 Ack=64 Win=252 Len=0

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.

Wireshark · Protocol Hierarchy Statistics · 1.pcapng

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	3043	100.0	2386309	875 k	0	0	0	3043
Ethernet	100.0	3043	1.8	42662	15 k	0	0	0	3043
Internet Protocol Version 6	0.1	2	0.0	80	29	0	0	0	2
User Datagram Protocol	0.1	2	0.0	16	5	0	0	0	2
Multicast Domain Name System	0.1	2	0.0	230	84	2	230	84	2
Internet Protocol Version 4	99.7	3034	2.5	60680	22 k	0	0	0	3034
User Datagram Protocol	95.9	2917	1.0	23336	8564	0	0	0	2917
Real-time Transport Control Protocol	0.9	27	0.0	823	302	26	687	252	31
Malformed Packet	0.0	1	0.0	0	0	1	0	0	1
QUIC IETF	9.8	298	8.7	207686	76 k	298	206748	75 k	302
NetBIOS Name Service	0.0	1	0.0	68	24	1	68	24	1
Multicast Domain Name System	0.2	5	0.0	581	213	5	581	213	5
HiPerConTracer Trace Service	0.0	1	0.1	1246	457	1	1246	457	1
Domain Name System	0.7	20	0.1	1397	512	20	1397	512	20
Data	84.3	2565	85.1	2029680	744 k	2565	2029680	744 k	2565
Transmission Control Protocol	3.8	117	0.1	2460	902	54	1200	440	117
Transport Layer Security	2.1	63	0.6	14705	5396	63	14705	5396	63
Address Resolution Protocol	0.2	7	0.0	196	71	7	196	71	7

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.

o.	Time	Source	Destination	Protocol	Length	Info
251	4.094531	SagemcomBroa_6f:82:…	Intel_dd:ea:50	ARP	52	Who has 192.168.2.102? Tell 192.168.2.1
252	4.094547	Intel_dd:ea:50	SagemcomBroa_6f:82:…	ARP	42	192.168.2.102 is at cc:2f:71:dd:ea:50
2253	13.925183	b2:15:54:b4:86:4f	Broadcast	ARP	52	Who has 192.168.2.1? Tell 192.168.2.93
2961	17.997125	9a:58:e7:80:25:30	Broadcast	ARP	52	Who has 192.168.2.215? Tell 192.168.2.250
2973	18.983172	9a:58:e7:80:25:30	Broadcast	ARP	52	Who has 192.168.2.215? Tell 192.168.2.250
2999	19.979487	9a:58:e7:80:25:30	Broadcast	ARP	52	Who has 192.168.2.215? Tell 192.168.2.250
3013	20.981619	9a:58:e7:80:25:30	Broadcast	ARP	52	Who has 192.168.2.215? Tell 192.168.2.250

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.

10	1.051701	162.159.128.235	192.168.2.102	TLSv1.2	117	Application Data
11	1.096899	192.168.2.102	162.159.128.235	TCP	54	56255 → 443 [ACK] Seq=80 Ack=64 Win=252 Len=0
181	1.552990	192.168.2.102	52.123.190.182	TLSv1.2	104	Application Data
187	1.614422	52.123.190.182	192.168.2.102	TLSv1.2	93	Application Data

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.

2282	15.297157	23.82.31.235	192.168.2.102	TLSv1.2	78	Application Data
2283	15.297339	192.168.2.102	23.82.31.235	TLSv1.2	82	Application Data
2303	15.878943	192.168.2.102	104.22.0.235	TLSv1.2	430	Client Hello (SNI=api.reasonsecurity.com)
2305	15.911572	104.22.0.235	192.168.2.102	TLSv1.2	169	Server Hello, Change Cipher Spec, Encrypted Handshake Message
2306	15.912221	192.168.2.102	104.22.0.235	TLSv1.2	675	Change Cipher Spec, Encrypted Handshake Message, Application Data
2307	15.944712	104.22.0.235	192.168.2.102	TLSv1.2	108	Application Data
2308	15.944896	192.168.2.102	104.22.0.235	TLSv1.2	164	Application Data
2310	15.994802	104.22.0.235	192.168.2.102	TLSv1.2	711	Application Data
2311	15.994802	104.22.0.235	192.168.2.102	TLSv1.2	119	Application Data, Encrypted Alert
2397	16.593041	192.168.2.102	142.250.81.227	QUIC	1292	Initial, DCID=2b45270d4b0f6928, PKN: 1, CRYPTO
2398	16.593106	192.168.2.102	142.250.81.227	QUIC	1292	Initial, DCID=2b45270d4b0f6928, PKN: 2, CRYPTO, PADDING, CRYPTO, PING, PADDING
2406	16.626464	142.250.81.227	192.168.2.102	QUIC	1292	Initial, SCID=eb45270d4b0f6928, PKN: 3, CRYPTO, PADDING
2598	16.982380	192.168.2.102	205.196.6.133	TLSv1.2	108	Application Data
2714	17.186263	192.168.2.102	52.123.190.181	TLSv1.2	111	Application Data
2753	17.271118	52.123.190.181	192.168.2.102	TLSv1.2	100	Application Data
2951	17.762016	162.159.130.234	192.168.2.102	TLSv1.2	107	Application Data

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:

a. Client initiates a connection to the server.

b. Server response with a digital certificate.

c. Client identifies the certificate to ensure its validity and confirms that it was issued by a trusted Certificate Authority (CA).

d. 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,

a. The client sends a “Client hello” message together with session ID to the server.

b. 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)

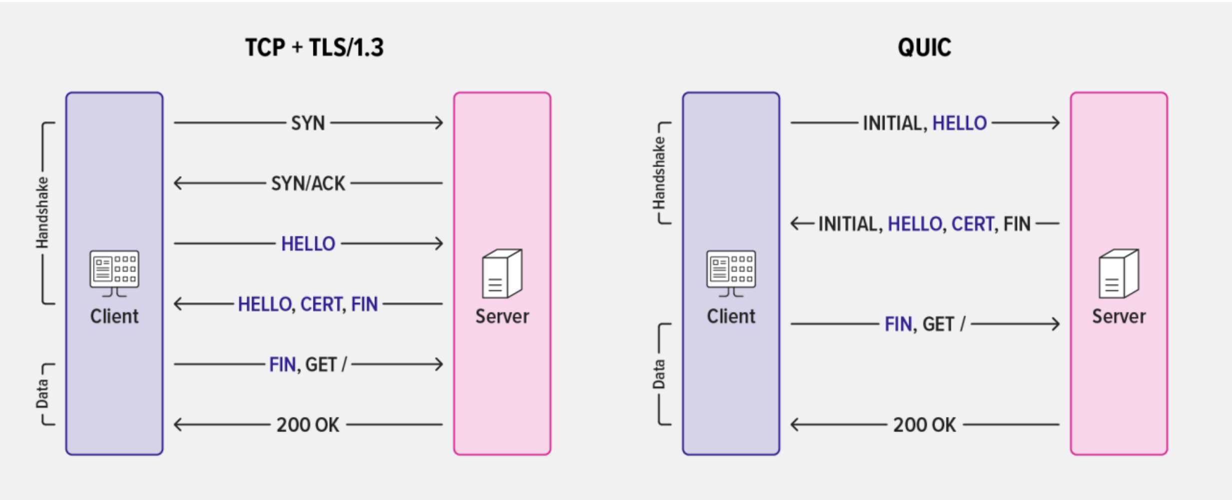
c. Encryption Key exchange: Change the encryption keys with each other by exchanging messages “Change Cipher spec”, “Client finished”, “Server finished”.

d. Client and server resume application data exchange.

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

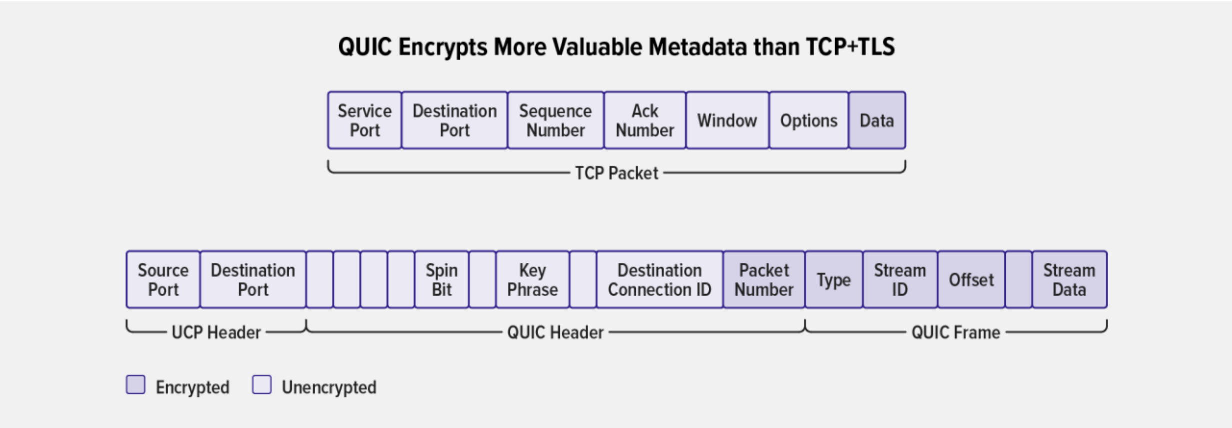
2.2.3.1 QUIC

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.



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.



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.

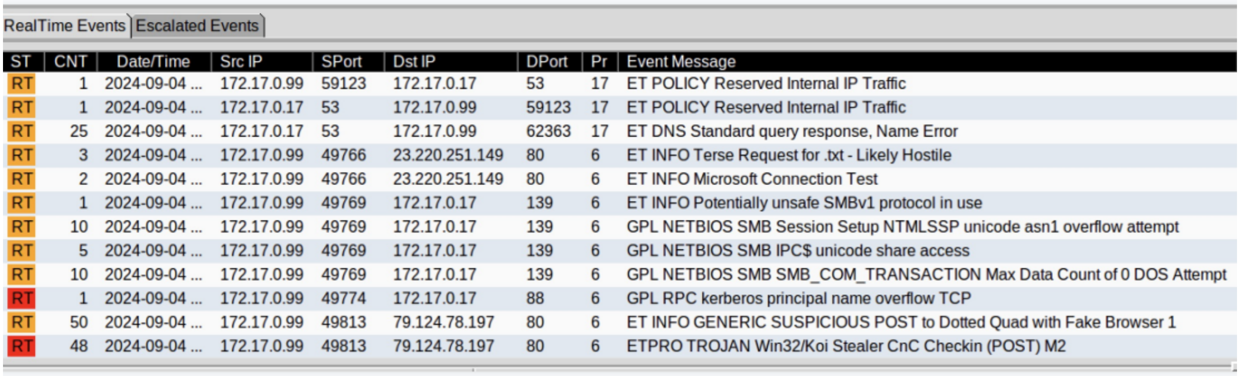
This makes UDP an unreliable protocol. While it excels in speed, it comes with risks, as the packets sent can be lost, duplicated, or corrupted during transmission. Although it prioritizes speed and efficiency, it sacrifices reliability and data integrity. Below is a summary comparison between TCP and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

2.3 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.



RealTime Events | Escalated Events

ST	CNT	Date/Time	Src IP	SPort	Dst IP	DPort	Pr	Event Message
RT	1	2024-09-04 ...	172.17.0.99	59123	172.17.0.17	53	17	ET POLICY Reserved Internal IP Traffic
RT	1	2024-09-04 ...	172.17.0.17	53	172.17.0.99	59123	17	ET POLICY Reserved Internal IP Traffic
RT	25	2024-09-04 ...	172.17.0.17	53	172.17.0.99	62363	17	ET DNS Standard query response, Name Error
RT	3	2024-09-04 ...	172.17.0.99	49766	23.220.251.149	80	6	ET INFO Terse Request for .txt - Likely Hostile
RT	2	2024-09-04 ...	172.17.0.99	49766	23.220.251.149	80	6	ET INFO Microsoft Connection Test
RT	1	2024-09-04 ...	172.17.0.99	49769	172.17.0.17	139	6	ET INFO Potentially unsafe SMBv1 protocol in use
RT	10	2024-09-04 ...	172.17.0.99	49769	172.17.0.17	139	6	GPL NETBIOS SMB Session Setup NTMLSSP unicode asn1 overflow attempt
RT	5	2024-09-04 ...	172.17.0.99	49769	172.17.0.17	139	6	GPL NETBIOS SMB IPC$ unicode share access
RT	10	2024-09-04 ...	172.17.0.99	49769	172.17.0.17	139	6	GPL NETBIOS SMB SMB_COM_TRANSACTION Max Data Count of 0 DOS Attempt
RT	1	2024-09-04 ...	172.17.0.99	49774	172.17.0.17	88	6	GPL RPC kerberos principal name overflow TCP
RT	50	2024-09-04 ...	172.17.0.99	49813	79.124.78.197	80	6	ET INFO GENERIC SUSPICIOUS POST to Dotted Quad with Fake Browser 1
RT	48	2024-09-04 ...	172.17.0.99	49813	79.124.78.197	80	6	ETPRO TROJAN Win32/Koi Stealer CnC Checkin (POST) M2

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.

Details

Details about the main implant:

* Sample source: https://app.any.run/tasks/0c21f3f8-9f51-44ae-9d5e-5a67a29cd9f9/
* Sample type: Intel 80386 (x86) .NET Assembly
* First submission (VT): 2024-03-28 14:47:02 UTC

We can deduct what each item in the configuration represents:

* `CG9GFcU9muFt8RFrooHX`: XOR key for decrypting the .NET payload
* `ENmpj9mb`: possibly an ID for this particular infection. This may have been the value used to determine the correct encryption key to send. Notice that it was used as the `subid` parameter in the download string
* `http://195.123.220.40/index.php`: The malicious server endpoint. This is passed as an argument to the malware so that it knows where to call back to

Using this configuration, we can write a generic script to decrypt and dump the PE ourselves without running it.

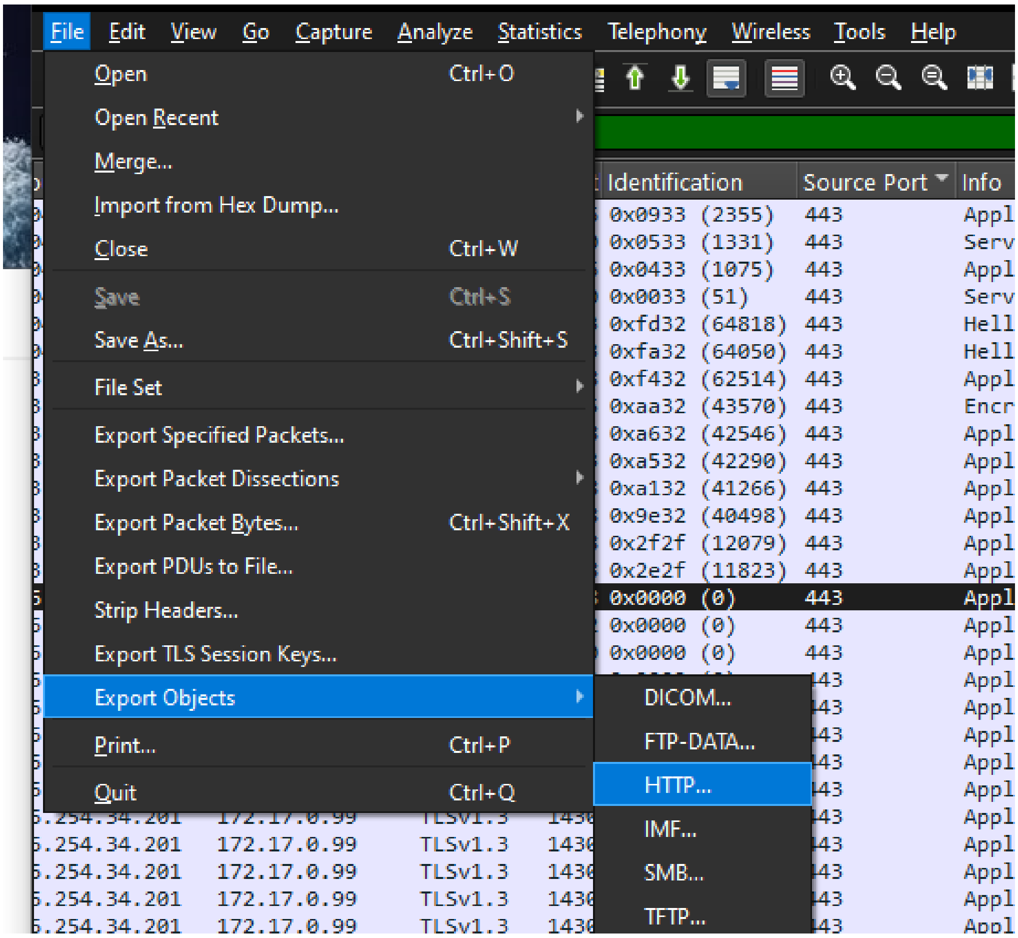
HTTPS traffic to www.bellantonicioccolato.it is also in this pcap, and it is likely associated with this infection. If you search Google for the domain plus the term "malware" you should find sandbox analysis and other entries that indicate the site is associated with Koi Loader/Koi Stealer activity.

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.

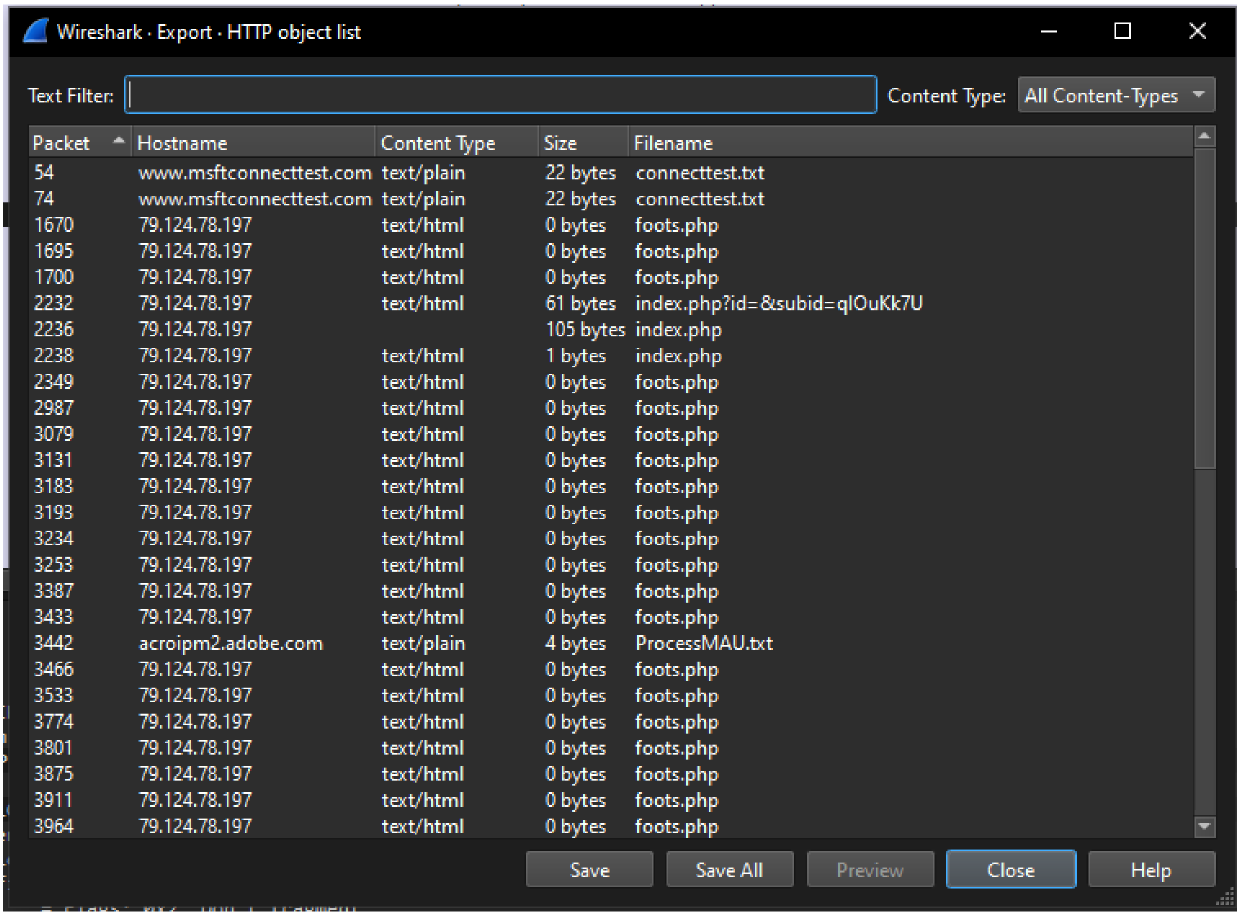
1212	129.372802	23.195.212.189	172.17.0.99	TLSv1.2	1430	0x472e	(18222)	443	Certificate Status
1213	129.372889	23.195.212.189	172.17.0.99	TLSv1.2	78	0x482e	(18478)	443	Server Key Exchange, Server Hello Done
1215	129.380955	172.17.0.99	23.195.212.189	TLSv1.2	180	0xc36f	(50031)	49804	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
1216	129.381071	172.17.0.99	23.195.212.189	TLSv1.2	432	0xc370	(50032)	49804	Application Data
1217	129.427485	23.195.212.189	172.17.0.99	TLSv1.2	328	0x492e	(18734)	443	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
1222	129.488558	23.195.212.189	172.17.0.99	TLSv1.2	398	0x4c2e	(19502)	443	Application Data
1249	145.434801	172.17.0.99	13.107.246.57	TLSv1.3	366	0xb605	(46597)	49805	Client Hello (SNI=inputsuggestions.msdxcdn.microsoft.com)
1251	145.468288	13.107.246.57	172.17.0.99	TLSv1.3	153	0x562e	(22062)	443	Hello Retry Request, Change Cipher Spec
1253	145.469838	172.17.0.99	13.107.246.57	TLSv1.3	405	0xb607	(46599)	49805	Change Cipher Spec, Client Hello (SNI=inputsuggestions.msdxcdn.microsoft.com)
1254	145.508271	13.107.246.57	172.17.0.99	TLSv1.3	1430	0x582e	(22574)	443	Server Hello, Application Data
1257	145.508583	13.107.246.57	172.17.0.99	TLSv1.3	1018	0x5b2e	(23342)	443	Application Data, Application Data, Application Data
1261	145.515603	172.17.0.99	13.107.246.57	TLSv1.3	128	0xb60b	(46603)	49805	Application Data
1262	145.523266	172.17.0.99	13.107.246.57	TLSv1.3	134	0xb60c	(46604)	49805	Application Data
1263	145.527267	172.17.0.99	13.107.246.57	TLSv1.3	245	0xb60d	(46605)	49805	Application Data
1264	145.541149	13.107.246.57	172.17.0.99	TLSv1.3	133	0x5c2e	(23598)	443	Application Data
1265	145.541149	13.107.246.57	172.17.0.99	TLSv1.3	133	0x5d2e	(23854)	443	Application Data
1266	145.541149	13.107.246.57	172.17.0.99	TLSv1.3	116	0x5e2e	(24110)	443	Application Data
1268	145.544294	172.17.0.99	13.107.246.57	TLSv1.3	85	0xb60f	(46607)	49805	Application Data
1269	145.552766	13.107.246.57	172.17.0.99	TLSv1.3	85	0x612e	(24878)	443	Application Data
1271	145.567918	13.107.246.57	172.17.0.99	TLSv1.3	361	0x622e	(25134)	443	Application Data
1279	153.533734	172.17.0.99	46.254.34.201	TLSv1.3	332	0x9632	(38450)	49806	Client Hello (SNI=www.bellantonicioccolato.it)
1281	153.712832	46.254.34.201	172.17.0.99	TLSv1.3	1430	0x6e2e	(28206)	443	Server Hello, Change Cipher Spec, Application Data
1283	153.712967	46.254.34.201	172.17.0.99	TLSv1.3	523	0x702e	(28718)	443	Application Data, Application Data, Application Data
1285	153.748774	172.17.0.99	46.254.34.201	TLSv1.3	369	0x9634	(38452)	49806	Change Cipher Spec, Application Data, Application Data
1286	153.918777	46.254.34.201	172.17.0.99	TLSv1.3	357	0x712e	(28974)	443	Application Data
1287	153.918777	46.254.34.201	172.17.0.99	TLSv1.3	357	0x722e	(29230)	443	Application Data

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.

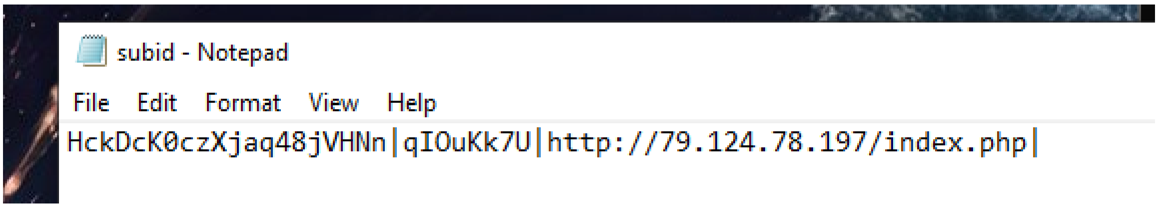


A window will appear displaying a list of packets containing HTTP objects. At that point, I immediately noticed the keyword “subid” which is another piece of information we found during our web search, appearing as a parameter in a download string.



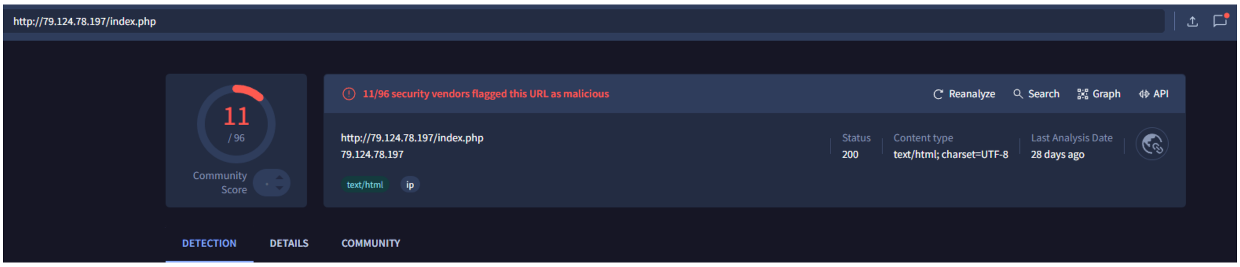
HTTP Object List

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:



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



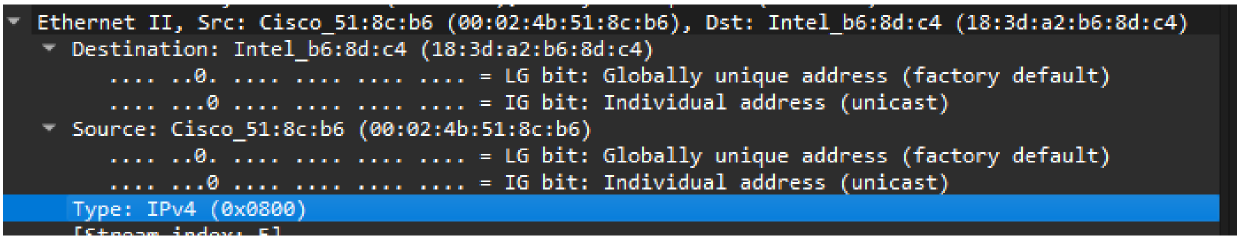
As we can see this URL is malicious with a score of only 11 out of 96.

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.

2.4 Decoding packets

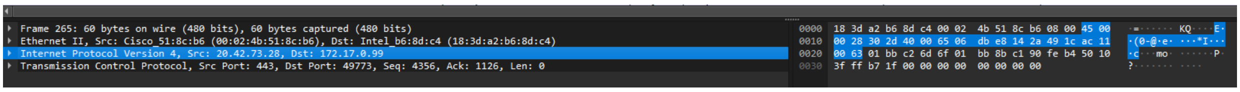
Select a random packet to view the packet's details and packets bytes.

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.



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/

2. 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

3. 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

4. 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.

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

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

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

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

7. VIRUSTOTAL (Website Scanner)

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