CS342 Assignment 2 - SKYPE
Kousik Rajesh

Drive link : https://drive.google.com/drive/folders/1qGgb_gRRICL2G0hQnZQV5cJc8-hDm5h-?usp=sharing

Q.1 Protocols used

TCP: Used to communicate to skype servers, call initiation, termination, messaging
UDP: Used for audio/video call from one node to another
STUN: Used for a binding request
TLSv1.2: Used for secure encrypted communication with the server
DNS: Used for a DNS lookup of skype servers

Link Layer

Ethernet II:

1. **Source IP**
2. **Destination IP**
3. **Source MAC**
4. **Destination MAC**

Network Layer

IPV4:

1. **Differentiated Services Code Point (DSCP):** specifies differentiated services per RFC 2474.
2. **Explicit Congestion Notification (ECN):** End to end congestion notification
3. **Time To Live (TTL):** Time packet persists in the network
4. **Protocol:** The protocol used in the data portion
5. **Header Checksum:** Checksum to verify packet integrity
6. **Source address**

Transport Layer

TCP/UDP

1. **Source port**
2. **Destination port**
3. **Checksum:** Checksum to verify packet integrity
4. **Checksum status:** Verified/unverified
5. **ACK and Sequence number(TCP)**

```
▾ Ethernet II, Src: sloth.local (0c:54:15:bd:36:20), Dst: HuaweiTe_86:a0:6f (8c:fd:18:86:a0:6f)
  ▾ Destination: HuaweiTe_86:a0:6f (8c:fd:18:86:a0:6f)
      Address: HuaweiTe_86:a0:6f (8c:fd:18:86:a0:6f)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  ▾ Source: sloth.local (0c:54:15:bd:36:20)
      Address: sloth.local (0c:54:15:bd:36:20)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
▾ Internet Protocol Version 4, Src: 192.168.1.10, Dst: 52.149.21.60
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1278
    Identification: 0x6acc (27340)
  ▸ Flags: 0x4000, Don't fragment
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0xbfaa [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.1.10
    Destination: 52.149.21.60
▾ Transmission Control Protocol, Src Port: 45100, Dst Port: 443, Seq: 1413, Ack: 3052, Len: 1238
    Source Port: 45100
    Destination Port: 443
    [Stream index: 3]
    [TCP Segment Len: 1238]
    Sequence number: 1413    (relative sequence number)
    Sequence number (raw): 1186816195
    [Next sequence number: 2651    (relative sequence number)]
    Acknowledgment number: 3052    (relative ack number)
    Acknowledgment number (raw): 141372387
    0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x018 (PSH, ACK)
    Window size value: 1421
    [Calculated window size: 1421]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x51f0 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ [SEQ/ACK analysis]
  ▸ [Timestamps]
    TCP payload (1238 bytes)
    TCP segment data (1238 bytes)
▸ [2 Reassembled TCP Segments (2650 bytes): #37(1412), #38(1238)]
▾ Transport Layer Security
  ▾ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 2645
      Encrypted Application Data: 000000000000007e663c159b37aa9cddc6c3ce2bf06a8133…
```

Application Layer

TLSv1.2 (Transport Layer Security)

Used to exchange application data or handshakes securely

1. **Content-type**: **Application data**/ handshake data
2. **Version:** The TLS version
3. **Length:** Length of the packet
4. **Encrypted Application Data**

DNS (Domain Name System)

DNS lookup used to get domain of skype server

Query/Response

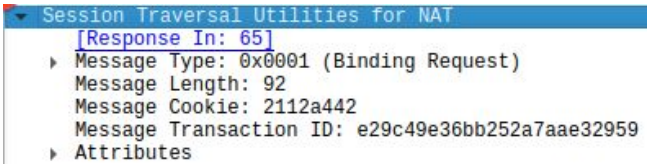
1. **Transaction ID :** A unique id identifying the request
2. **Flags:** Flags for dns request
3. **Questions:** the number of questions asked in the message
4. **Queries:** All the queries for DNS lookup
5. **RRs:** resource records

```
▸ Frame 53: 104 bytes on wire (832 bits), 104 bytes captured (832 b
▸ Ethernet II, Src: sloth.local (0c:54:15:bd:36:20), Dst: HuaweiTe_8
▸ Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.1.1
▸ User Datagram Protocol, Src Port: 58394, Dst Port: 53
▾ Domain Name System (query)
    Transaction ID: 0x70d5
  ▸ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
  ▾ Queries
    ▸ broker-asse-03-skype.cloudapp.net: type AAAA, class IN
  ▸ Additional records
    [Response In: 60]
```

STUN (Session Traversal Utilities for NAT)

STUN is used to resolve the public IP of a device running behind a NAT used in VoIP calls and to make binding requests

1. **Message Type:** Allocate/Binding/Refresh request
2. **Length**
3. **Cookie**
4. **Transaction ID**



Q.2 **Functionalities**

Skype has a persistent TCP connection to the server which it maintains at all times. Hence most information can be conveyed through this TCP stream itself.

1. **Login authentication:**
   Uses **DNS** on a UDP packet to lookup domain name for server, A **TCP** handshake follows and **TLS** is used to create a secure encrypted channel for authentication
2. **Add contact:**
   TCP - The persistent connection can be used, using **UDP** we would need to worry about packet loss and retransmission
3. **Schedule call : TCP**
4. **Initiate/terminate call : TCP, UDP, P2P**
   TCP is used to initiate the call and **UDP** is used to transfer call data as it is a loss tolerant application and maintaining a connection is expensive
5. **Send message**
   TCP is used since we cannot afford loses
6. **Create a poll : TCP**
7. **View profile : TCP**
8. **Notifications : TCP**
9. **Private chat (E2E encrypted) : TCP**
   (**UDP** in case of private video/audio call)
10. **VoIP call : P2P, UDP, TCP**
11. **VoIP SMS : P2P, UDP, TCP**

Skype also pickles randomly from a wide range of ports so it is harder to be blocked by a firewall. This also makes it difficult to do port filtering on wireshark to remove noise from other running background processes.

Q.3

Application 1: Voice/Video call

Skype uses a proprietary video conferencing protocol based on **P2P(Peer to Peer) networking.** Assume a client **X** calls **Y**

Each logged-in user maintains a **persistent TCP connection** established with skype servers. When **X** makes a call to **Y,** the IP address of user **Y** is looked up from data stored on the server.
For **Y** this persistent connection is used to sound a ring when a call comes.

When a new call is initiated by **X** it **creates a new TCP connection** in addition to the persistent connection to the server, which will be **closed when the call is terminated**. This connection could be used as a **heartbeat mechanism to check if the call is still alive**. This is necessary as, during the call due to the **P2P** policy none of the packets go through the server, and hence it has no way of knowing if the call is still alive.

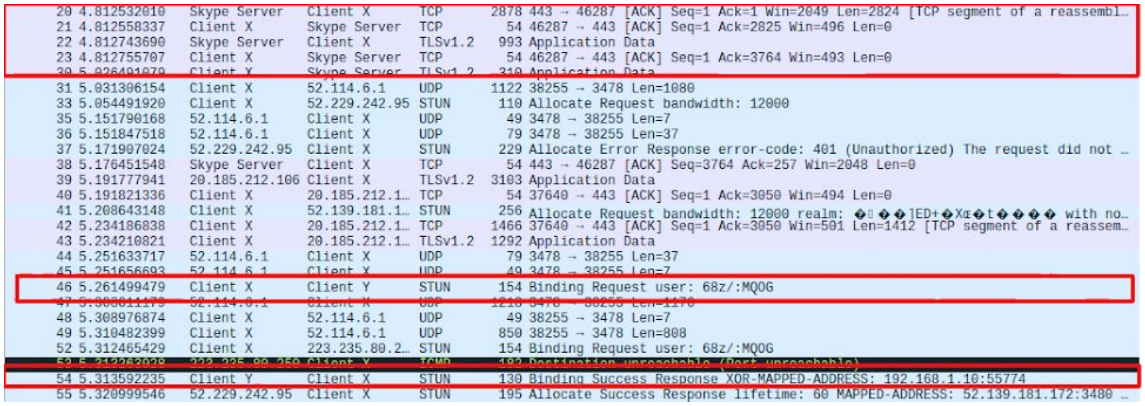
Handshake

57	5.321542893	Client X	broker-asse-02-skype.…	TCP	74 38629 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=13740890…
62	5.392350715	broker-asse-02-skype…	Client X	TCP	66 443 → 38629 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 WS=256 SACK_PER…
63	5.392379648	Client X	broker-asse-02-skype.…	TCP	54 38629 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0
64	5.392610527	Client X	broker-asse-02-skype.…	TLSv1.2	353 Client Hello
70	5.462016102	20.185.212.106	Client X	TCP	54 443 → 37640 [ACK] Seq=3050 Ack=2651 Win=2051 Len=0
71	5.464474149	broker-asse-02-skype…	Client X	TCP	2878 443 → 38629 [ACK] Seq=1 Ack=300 Win=524800 Len=2824 [TCP segment of a reass…
72	5.464490209	Client X	broker-asse-02-skype.…	TCP	54 38629 → 443 [ACK] Seq=300 Ack=2825 Win=63488 Len=0
73	5.481539809	broker-asse-02-skype…	Client X	TLSv1.2	1394 Server Hello, Certificate, Server Key Exchange, Server Hello Done
74	5.481553410	Client X	broker-asse-02-skype.…	TCP	54 38629 → 443 [ACK] Seq=300 Ack=4165 Win=63104 Len=0
75	5.484859460	Client X	broker-asse-02-skype.…	TLSv1.2	212 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
79	5.571500245	broker-asse-02-skype…	Client X	TLSv1.2	105 Change Cipher Spec, Encrypted Handshake Message
80	5.571514812	Client X	broker-asse-02-skype.…	TCP	54 38629 → 443 [ACK] Seq=458 Ack=4216 Win=64128 Len=0
81	5.571686143	Client X	broker-asse-02-skype.…	TLSv1.2	647 Application Data

Shown above is the trace for connection initiation which starts with a **SYN packet (used for initiating a TCP connection)(Packet 57)** sent by client to the server. This is followed by the **Client Hello (Packet 64)** message to initiate the key exchange and the server responds with a **Server Hello(Packet 73)** which contains a **Certificate** to verify the authenticity of the server and completes the key exchange process. The encrypted channel is now established and a **new encrypted handshake is done through the encrypted channel** which exchanges important server information **(Packet 79)**.

Skype uses encryption on its calls hence **X** establishes a secure encrypted channel with **Y** with the help of the new encrypted TCP connection we just created using TLS.

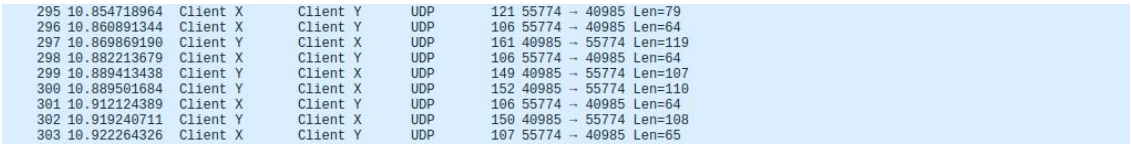
Next, **X** makes a **STUN** binding request to **Y,** and **Y** returns a response with an **XOR mapped address.**
Now the connection is established.
Now the call has started. Using a **P2P protocol** the **UDP** packets of the call carrying video/audio are routed from **X** to **Y** without any server intervention.
(Numeric IP's have been renamed for clarity)



Client X initiates TCP connection with the server and communicates with the server to initiate a call to Y

Binding Request by X to Y

Binding Response by Y to X

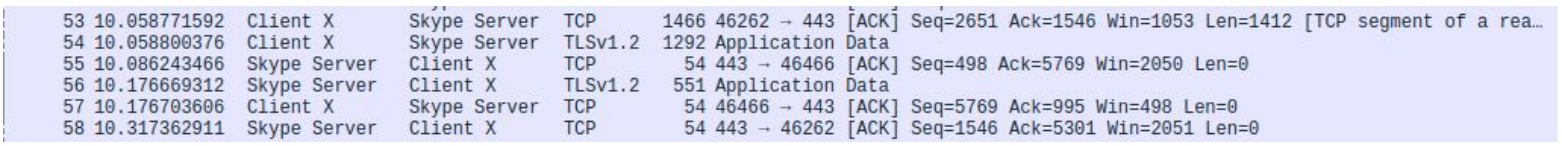


Now since the connection is established the call continue using purely UDP packets

Termination
The trace showing the connection termination is given below

The connection is terminated when the call ends by sending a **TCP [FIN] packet.**
The **TCP RST packet** is indicating that a packet was attempted to be sent to a host connection which was previously closed/terminated.

Application 2: Messaging
For messaging skype **does not** use a **P2P policy. All communication happens solely with the server.**
The client **uses the already established encrypted persistent TCP connection** with the server to send messages which further sends it to the receiver using its secure TCP connection. No handshakes are observed as no new TCP connections are created



As shown above, the client sends the encrypted message using the secure TLS channel and receives ACK and some packets back from the server(Could possibly contain information such as sent/read receipts)

Skype also offers an **End-to-End encrypted** private channel for messaging/calls. The encryption and messaging mechanism is **based on the Signal Protocol**. The difference in the E2E channel is that initially when setup, it does a key exchange using DHKE to setup a shared secret channel that is closed to the server. From my analysis, I found that once such a channel is established the packet sequences are similar for both calls and messaging as the one given above. (The only difference is that the messages are encrypted with the new E2E key which is not known to the server)

Q.4

Statistics	Time		
	14:00	16:30	19:20
Throughput(bytes/s)	160k	27k	134k
RTT(ms)	184	230	107
Packet size(Bytes)	676	247	611
Number of packets lost	3	7	3
UDP Packets	10076	4267	9088
TCP Packets	264	249	259
Response ratio	0.77	0.89	0.84

Q.5
Yes, multiple sources exist. Some of them are

Servers facilitating persistent connections	Servers facilitating video/audio calls
20.185.212.106	52.114.6.1
40.77.18.167	52.114.15.57
40.83.97.152	52.114.15.65
40.114.211.99	52.114.77.164
	52.114.128.71
	52.149.21.60
	52.114.159.22
	52.247.2.90

By observing I noticed that the sources on **the left** were used for the **persistent TCP connections** and the **right** ones(starting with 52) were initiated when a call was made.
Reasons for multiple sources:

- Skype has **multiple parallel TCP connections** for facilitating activities such as messaging, call alerts, video calls
- By **differentiating servers based on the type of requests**(messaging/calls) they receive, it gives more opportunity for them to **optimize the server** to handle those requests better
- Multiple servers also provide **load balancing**, prevent congestion on one server, and improves the quality of service
- A singles server is a **Single Point of Failure** and should be avoided

Some Interesting Observations
Since Skype uses a P2P protocol during a call between two devices on the same local network, connection to the Internet is not required. I tested this by creating a call between two devices connected to my WIFI router(locally connected). Wireshark showed the Local IP Address of both devices. Once the call was initiated (This part requires the internet as a new TCP connection is created with skype server) I turned off the internet access of my WIFI router and the call continued to work with no issues.
This kind of protocol removes unnecessary load from skype servers.

Files
Dump_1400.pcapng
Dump_1630.pcapng
Dump_1920.pcapng