

UNIVERSIDAD DE LOS ANDES
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y
COMPUTACIÓN



LABORATORIO #2: ANÁLISIS DE PROTOCOLOS DE LA CAPA DE
APLICACIÓN - PRUEBA DE CONECTIVIDAD

ISIS 3204 - INFRAESTRUCTURA DE COMUNICACIONES

Nathalia Quiroga

Grupo 7

David Quiroga 202310820

Nicolas Gonzalez 202310041

Samuel Rodríguez Torres 202310140

2025-20

Contenido

Actividad 8.1: Pruebas Ping	3
a. Servicio de DNS por IP	3
b. Servicio de FTP por IP	3
c. Filtro ICMP	4
Actividad 8.2: Análisis de tráfico del Servidor DNS	5
a. Prueba de conectividad al Servidor WEB por IP	5
b. Caché del DNS	5
c. Borrar caché del DNS	6
d. Prueba de conectividad al Servidor WEB - URL	6
e. Análisis del tráfico DNS	7
Actividad 8.3: Análisis de tráfico del Servidor FTP	9
a. Acceder al servidor FTP	9
b. Interacciones con el servidor FTP	10
c. Análisis del tráfico FTP	11
Actividad 8.4: Análisis de tráfico del Servidor WEB	12
a. Conexión al servidor HTTP	13
b. Análisis del tráfico HTTP	13
Actividad 8.5: Análisis del protocolo HTTPS realizando navegación de sitios web	15
a. Conexión a YouTube	15
b. Conexión a sitios web	15
c. Análisis del tráfico HTTPS	16
Actividad 8.6: Análisis del protocolo VoIP	17
Actividad 8.7: Análisis del protocolo RTMP	18
Conclusión	21

1. Actividad 8.1 Pruebas Ping:

a. Servicio de DNS por IP

Verificando la conectividad de los servidores y los equipos de la red, se hizo una prueba de ping al servicio de DNS y se capturó el tráfico por medio de un sniffer, Wireshark. Se guardó la captura del tráfico bajo el nombre de **Ping_DNS_IP.pcap** como lo indicaba la guía y se ve a continuación:

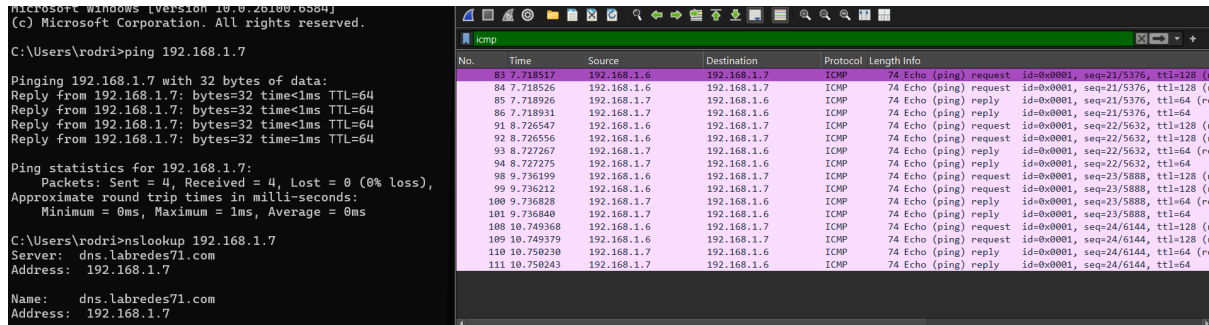


Imagen 1: Captura de paquetes del ping en Ping_DNS_IP.pcap

En la parte izquierda se observa el ping hacia la dirección IP 192.168.1.7. Posteriormente se ejecuta un nslookup sobre esa misma dirección como prueba de que está asociada al servidor DNS. El resultado del ping confirma que todos los paquetes fueron recibidos exitosamente. En el analizador de tráfico, filtrando por el protocolo ICMP, se visualiza la comunicación entre la IP del cliente y la del servidor, intercambiando paquetes Echo Request y Echo Reply, cada uno con su número de secuencia y su campo TTL (Time To Live) correspondiente. Analizándolo por capas, desde la capa de aplicación el usuario ejecuta el comando ping, en transporte no interviene TCP ni UDP, ya que ICMP es un protocolo de control que se ubica directamente sobre IP, a nivel de red el mensaje ICMP se encapsulan en un paquete IP que incluye direcciones de origen y destino, en enlace de datos: dicho paquete se transmite a través de Wi-Fi en forma de tramas con direcciones MAC, física: las tramas se convierten en señales eléctricas o de radio en el medio.

Durante la comunicación, el cliente envía paquetes Echo Request con un número de secuencia generado por el sistema operativo. Este suele incluir un Identificador único (para distinguir múltiples procesos de ping simultáneos) y un contador incremental que inicia en 0 o 1. El servidor responde con un Echo Reply que conserva estos valores, permitiendo al cliente verificar qué paquetes llegaron y detectar posibles pérdidas.

b. Servicio de FTP por IP

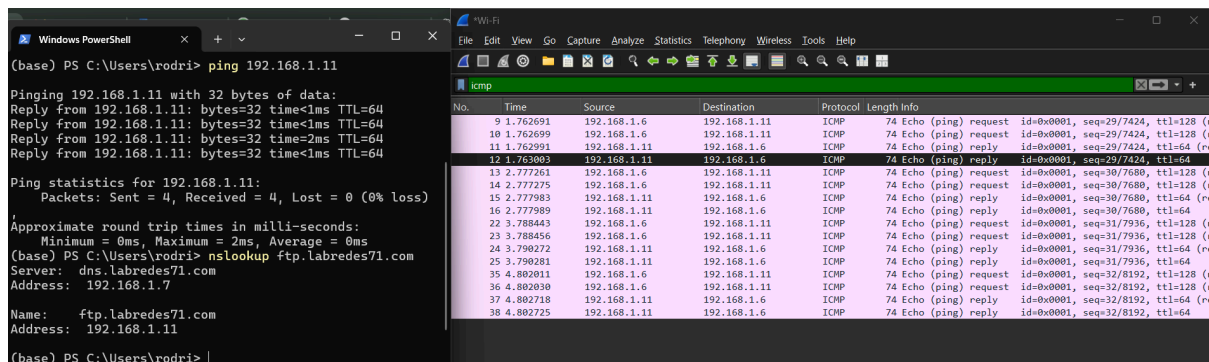


Imagen 2: Captura de paquetes del ping en Ping_FTP_IP.pcap

Para verificar la conectividad con el servidor FTP se realizó un ping utilizando directamente su dirección IP, siguiendo el mismo procedimiento descrito en el apartado a) Servicio de DNS por IP. Al

ejecutar un ping, el comportamiento es independiente del servidor al que se dirija la prueba, ya que este se basa en un estándar de comunicación definido por el protocolo ICMP, el cual envía paquetes de tipo Echo Request y espera las correspondientes respuestas Echo Reply. Por esta razón, la captura de tráfico mostrada en la Imagen 1 para el ping al servidor DNS resulta prácticamente idéntica a la obtenida con el servidor FTP, diferenciándose únicamente en las direcciones IP involucradas. El detalle de la conexión y el protocolo FTP se verá y analizará en su respectiva sección.

c. Filtro ICMP

Wireshark es un sniffer que registra cada paquete y permite filtrarlos por protocolo, puerto, ips, contenido, etc. Al seleccionar los paquetes, se puede obtener el proceso que este estuvo siguiendo a lo largo de las capas del modelo TCP/IP, de lo cual se puede ver la información solicitada por el enunciado como se muestra en la imagen 3 con el filtro por protocolo de icmp:

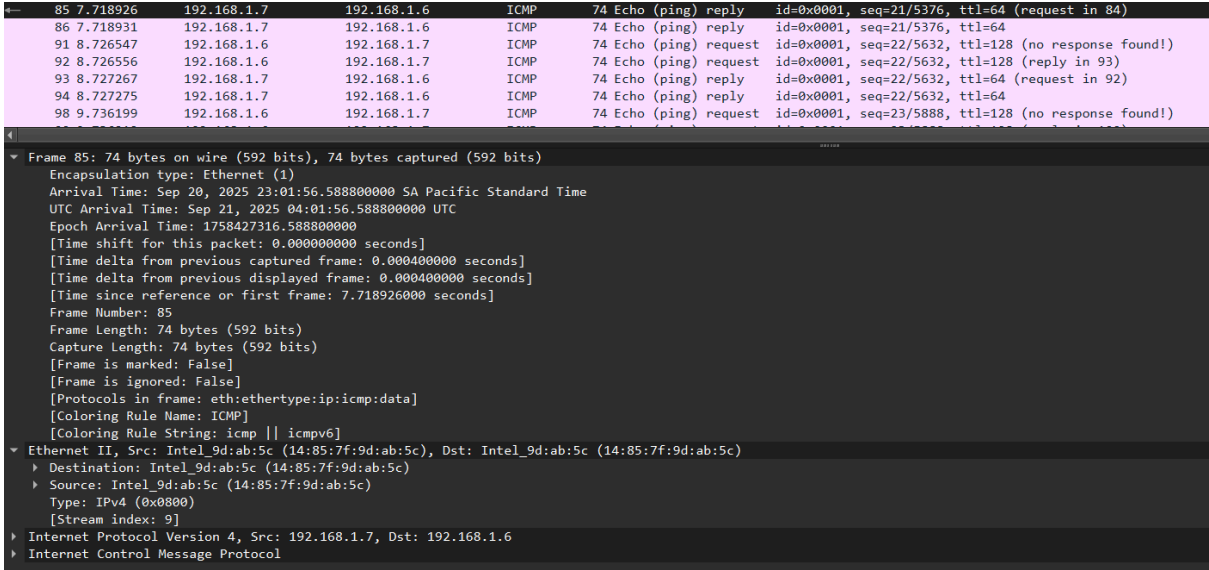


Imagen 3: Viaje del paquete a través de las capas

Archivo Ping_DNS_IP.pcap

IP origen Request	192.168.1.6
IP destino Request	192.168.7
IP origen reply	192.168.1.7
IP destino reply	192.168.1.6
MAC Cliente	14:85:7f:9d:ab:5c
MAC Servidor	00:0c:29:8d:07:ce

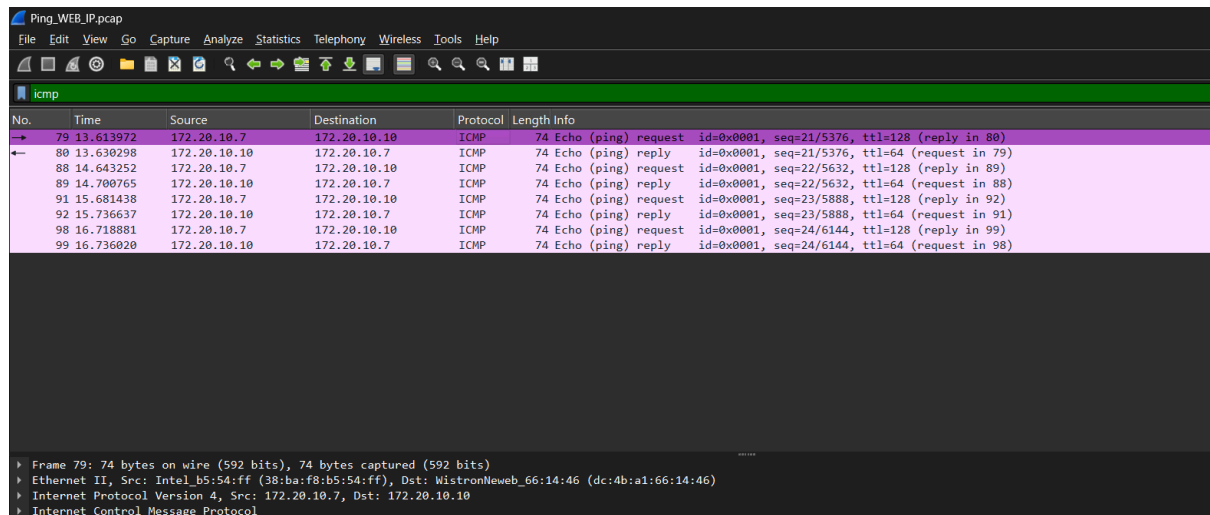
Archivo Ping_FTP_IP.pcap

IP origen Request	192.168.1.6
IP destino Request	192.168.11
IP origen reply	192.168.1.11

IP destino reply	192.168.1.6
MAC Cliente	14:85:7f:9d:ab:5c
MAC Servidor	00:0c:29:40:cf:72

2. Actividad 8.2: Análisis de tráfico del Servidor DNS.

a. Prueba de conectividad al Servidor WEB por IP



No.	Time	Source	Destination	Protocol	Length	Info
79	13.613972	172.20.10.7	172.20.10.10	ICMP	74	Echo (ping) request id=0x0001, seq=21/5376, ttl=128 (reply in 80)
80	13.630298	172.20.10.10	172.20.10.7	ICMP	74	Echo (ping) reply id=0x0001, seq=21/5376, ttl=64 (request in 79)
88	14.643252	172.20.10.7	172.20.10.10	ICMP	74	Echo (ping) request id=0x0001, seq=22/5632, ttl=128 (reply in 89)
89	14.700765	172.20.10.10	172.20.10.7	ICMP	74	Echo (ping) reply id=0x0001, seq=22/5632, ttl=64 (request in 88)
91	15.681438	172.20.10.7	172.20.10.10	ICMP	74	Echo (ping) request id=0x0001, seq=23/5888, ttl=128 (reply in 92)
92	15.736637	172.20.10.10	172.20.10.7	ICMP	74	Echo (ping) reply id=0x0001, seq=23/5888, ttl=64 (request in 91)
98	16.718881	172.20.10.7	172.20.10.10	ICMP	74	Echo (ping) request id=0x0001, seq=24/6144, ttl=128 (reply in 99)
99	16.736020	172.20.10.10	172.20.10.7	ICMP	74	Echo (ping) reply id=0x0001, seq=24/6144, ttl=64 (request in 98)

Frame 79: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
 Ethernet II, Src: Intel_b5:54:ff (38:ba:f8:b5:54:ff), Dst: WistronNeweb_66:14:46 (dc:4b:a1:66:14:46)
 Internet Protocol Version 4, Src: 172.20.10.7, Dst: 172.20.10.10
 Internet Control Message Protocol

Imagen 4: Captura de paquetes del ping en Ping_WEB_IP.pcap

Al igual que en las anteriores, por medio de la consola se realizó una prueba de conexión con ping al servidor web, donde se pueden ver los 4 paquetes enviados por el cliente y los 4 paquetes de respuesta del servidor, viendo el mismo comportamiento en DNS, FTP y ahora en WEB. Con esto, sabemos que el servidor DNS, FTP y WEB son accesibles a nivel de red para sus clientes. Cada uno de estos archivos, como se indica en la descripción de la imagen es Ping_<Servidor>_IP.pcap y se encuentran subidos en el mismo repositorio donde se encuentra este archivo.

b. Caché del DNS

Como se indica en la guía, se usa el comando `ipconfig /displaydns` para ver el caché existente del dns en este momento; esto es importante porque puede afectar el posterior análisis del tráfico si una consulta ya se encuentra en el caché, de modo que no se podrá analizar el tráfico generado hacia y desde el servidor DNS.

```
C:\Users\nicog>ipconfig /displaydns

Configuración IP de Windows

www.bing.com
-----
Nombre de registro . . : www.bing.com
Tipo de registro . . . : 5
Período de vida . . . : 54729
Longitud de datos . . : 8
Sección . . . . . : respuesta
Registro CNAME. . . . : www-www.bing.com.trafficmanager.net

Nombre de registro . . : www-www.bing.com.trafficmanager.net
Tipo de registro . . . : 5
Período de vida . . . : 54729
Longitud de datos . . : 8
Sección . . . . . : respuesta
Registro CNAME. . . . : www.bing.com.edgekey.net

Nombre de registro . . : www.bing.com.edgekey.net
Tipo de registro . . . : 5
Período de vida . . . : 54729
Longitud de datos . . : 8
Sección . . . . . : respuesta
Registro CNAME. . . . : e86303.dscx.akamaiedge.net

Nombre de registro . . : e86303.dscx.akamaiedge.net
```

Imagen 5: comando ipconfig /displaydns

Concordando con lo esperado, se muestra el caché del dispositivo donde se pueden ver una cadena de registros CNAME que empieza en Bing y termina en Akamai, lo que confirma que el sistema almacena temporalmente las resoluciones DNS para optimizar futuras consultas y agilizar la conexión a los servicios.

c. Borrar caché del DNS

Anteriormente se mencionó la razón por la cual el cache del dns puede afectar el análisis de tráfico a nuestros servidores, por lo que se usa el comando ipconfig /flushdns para vaciar el caché; finalmente se usa displaydns para verificar que el caché está vacío como se puede observar en la siguiente imagen:

```
login.microsoftonline.com
-----
No hay registros de tipo A

C:\Users\nicog>nslookup
Servidor predeterminado: dns.labredes71.com
Address: 172.20.10.5

>
C:\Users\nicog>ipconfig /flushdns

Configuración IP de Windows

Se vació correctamente la caché de resolución de DNS.

C:\Users\nicog>ipconfig /displaydns

Configuración IP de Windows
```

Imagen 6: comando ipconfig /flushdns

d. Prueba de conectividad al Servidor WEB - URL

Se repitió el mismo procedimiento para realizar una prueba de conectividad pero con la excepción de que ahora se accede al servidor WEB por medio de la URL. Esto es un cambio significativo en el tráfico que se debe analizar ya que ahora el servidor DNS deberá hacer la traducción de la url a la IP del servidor WEB como se ve a continuación:

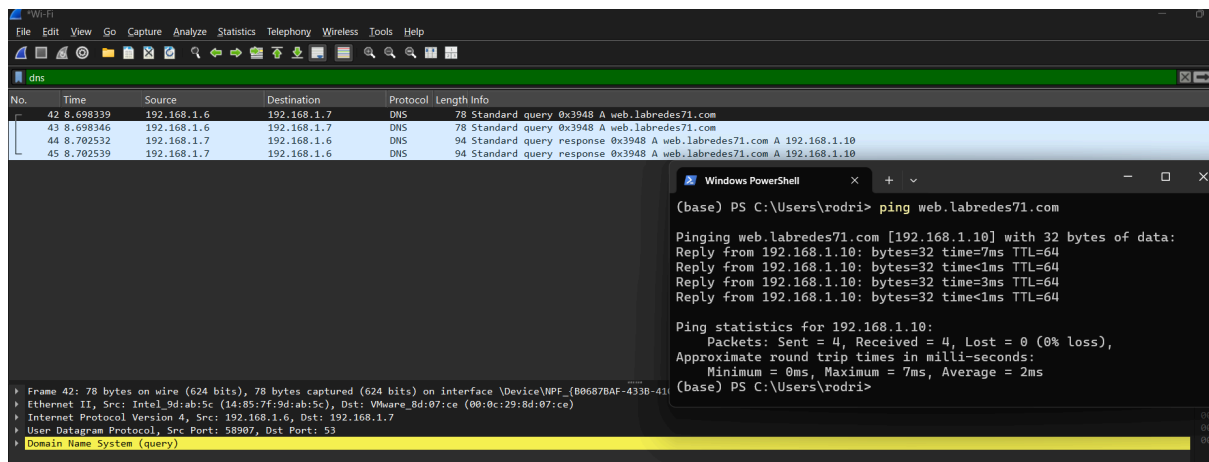


Imagen 7: Tráfico del archivo Ping_WEB.pcap filtrado por DNS

En la imagen se puede ver el comando ping realizado y el tráfico generado con el DNS para traducir la url a su IP. En la siguiente sección se detalla más sobre el tráfico que se envía y se recibe del DNS.

e. Análisis del tráfico DNS

Abriendo el paquete con la descripción sobre el registro tipo A de la url del dominio del servidor WEB a su respectiva IP se pudo obtener la información de la siguiente imagen:

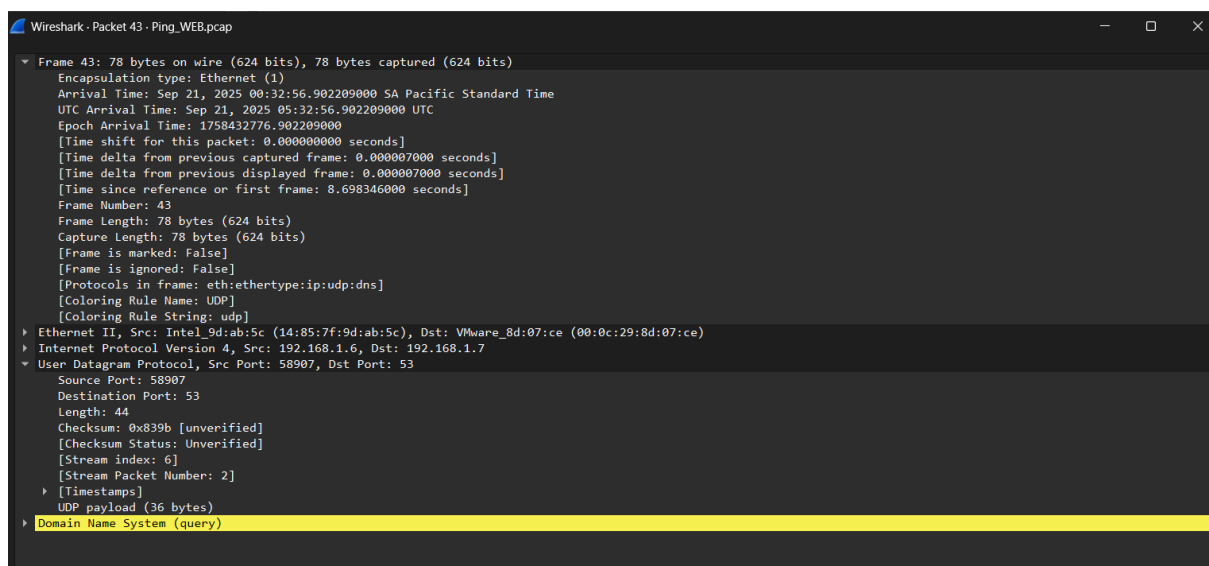


Imagen 8: Muestra de un paquete relacionado con el DNS por las diferentes capas.

A partir del análisis de este tráfico se logró identificar en la capa de aplicación el tipo de registro solicitado (A), el dominio consultado (web.labredes71.com), la IP con la que respondió el servidor y las flags que muestran detalladamente la respuesta, en la imagen 9 se puede ver que la respuesta fue autoritativa, se hizo de manera recursiva y sin errores

```

Frame 45: 94 bytes on wire (752 bits), 94 bytes captured (752 bits)
Ethernet II, Src: Intel_9d:ab:5c (14:85:7f:9d:ab:5c), Dst: Intel_9d:ab:5c (14:85:7f:9d:ab:5c)
Internet Protocol Version 4, Src: 192.168.1.7, Dst: 192.168.1.6
User Datagram Protocol, Src Port: 53, Dst Port: 58907
Domain Name System (response)
Transaction ID: 0x3948
Flags: 0x8580 Standard query response, No error
1... .. = Response: Message is a response
.000 0... .. = Opcode: Standard query (0)
.... 1... .. = Authoritative: Server is an authority for domain
.... 0... .. = Truncated: Message is not truncated
.... 1... .. = Recursion desired: Do query recursively
.... 1... .. = Recursion available: Server can do recursive queries
.... 0... .. = Z: reserved (0)
.... 0... .. = Answer authenticated: Answer/authority portion was not authenticated by the server
.... 0... .. = Non-authenticated data: Unacceptable
.... 0000 = Reply code: No error (0)
Questions: 1
Answer RRs: 1
Authority RRs: 0
Additional RRs: 0
Queries
  web.labredes71.com: type A, class IN
    Name: web.labredes71.com
    [Name Length: 18]
    [Label Count: 3]
    Type: A (1) (Host Address)
    Class: IN (0x0001)
Answers
  web.labredes71.com: type A, class IN, addr 192.168.1.10
  [Retransmitted response. Original response in: 44]
  [Retransmission: True]

```

Imagen 9: Capa de aplicación de un paquete de DNS.

Por el lado de la capa de transporte, como se ve en la imagen 10, este usa UDP, desde el puerto de origen 53 al puerto 58907 del cliente (esto debido a que es la respuesta del DNS al cliente). Adicionalmente, tiene una longitud de 60 bytes, 52 del payload y 8 de la cabecera, se verifica la integridad del paquete con el checksum, en este caso Wireshark no lo validó porque la tarjeta de red no lo pasó aún verificado (offloading). El puerto 58907 es un puerto efímero usado únicamente por el cliente para esta conexión y se hace por UDP ya que se prefiere tener una alta velocidad en las respuestas y la cantidad de información no es mucha, por lo que es poco usual ver un fallo o que un paquete se pierda.

```

User Datagram Protocol, Src Port: 53, Dst Port: 58907
Source Port: 53
Destination Port: 58907
Length: 60
Checksum: 0xfa8e [unverified]
[Checksum Status: Unverified]
[Stream index: 6]
[Stream Packet Number: 4]
[Timestamps]
  [Time since first frame: 0.004200000 seconds]
  [Time since previous frame: 0.000007000 seconds]
UDP payload (52 bytes)

```

Imagen 10: Capa de transporte de un paquete de DNS.

Cabe resaltar que el puerto 53 es el puerto utilizado por el servicio de DNS tanto para la respuesta como para las consultas. Finalmente, con esta información pudimos completar la tabla para el servidor WEB:

Archivo Ping_WEB.pcap

IP origen Request	192.168.1.6
IP destino Request	192.168.10
IP origen reply	192.168.1.10
IP destino reply	192.168.1.6
MAC Cliente	14:85:7f:9d:ab:5c
MAC Servidor	02:AB:3F:7C:91:D4

3. Actividad 8.3: Análisis de tráfico del Servidor FTP.

Para analizar el tráfico del servidor FTP se vació el caché por las razones mencionadas previamente, repitiendo todos los pasos y se accedió al servicio por medio de FileZilla para interactuar con el servidor subiendo y descargando archivos.

a. Acceder al servidor FTP

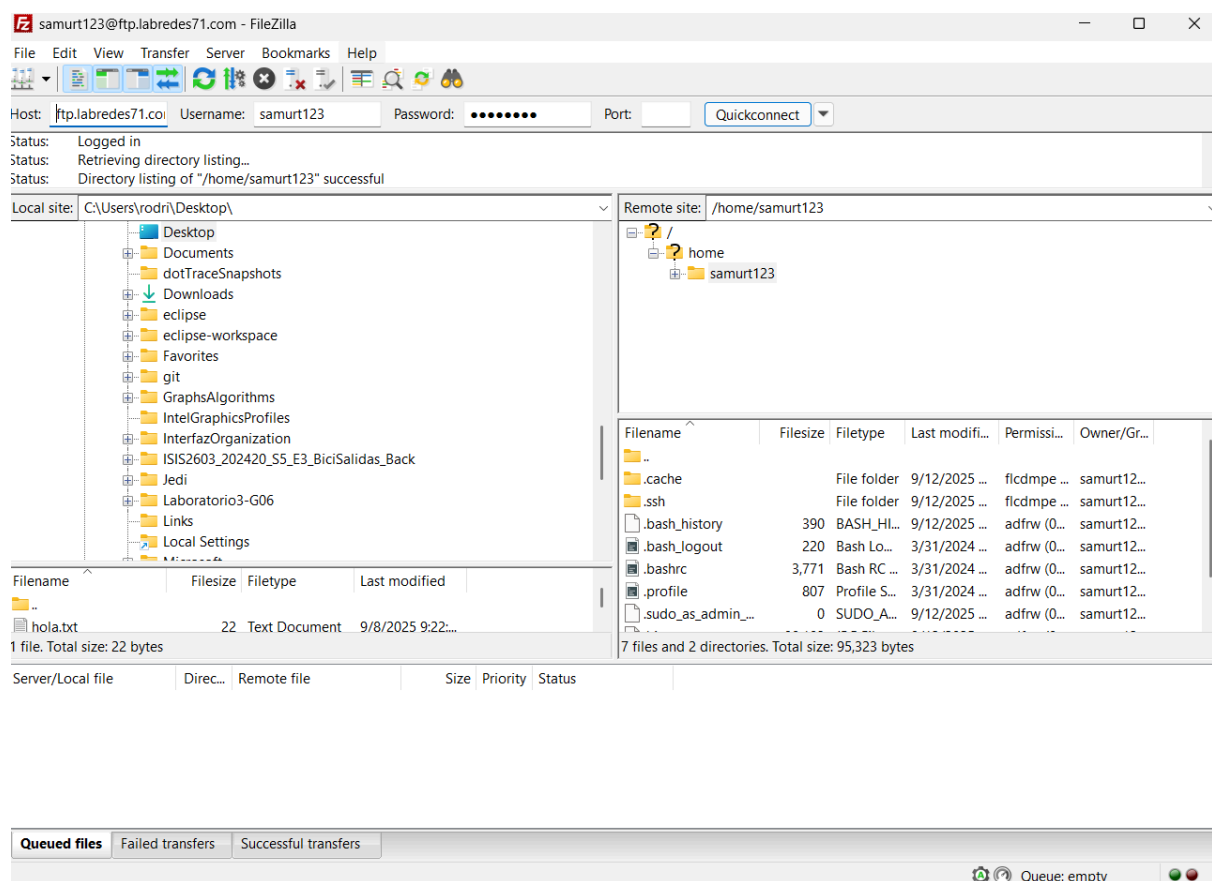


Imagen 11: Conexión con FileZilla al servidor FTP

Usando FileZilla, decidimos acceder al servicio por medio de la URL para añadir tráfico DNS a la captura de Wireshark, luego accedimos con un Username y un Password válidos definidos en el servidor FTP, el cual es samurt123 y password, en el puerto pusimos 21 ya que es el puerto del protocolo FTP para control y sesión. Posterior a la conexión se llega al estado que se muestra en la

imagen 11, donde ya no aparece el puerto utilizado pero si se mantiene el host, el username y password. Con respecto a la imagen, del lado izquierdo se puede observar los archivos del dispositivo cliente y del lado derecho, los del servidor.

b. Interacciones con el servidor FTP

Por medio de FileZilla, desde el equipo cliente se descarga un archivo del servidor FTP, 1.png, y se capturó el tráfico con wireshark como se ve a continuación:

No.	Time	Source	Destination	Protocol	Length	Info
258	28.479572	192.168.1.11	192.168.1.6	FTP	76	Response: 500 No entiendo AUTH
260	28.482450	192.168.1.6	192.168.1.11	FTP	70	Request: USER samurt123
262	28.483107	192.168.1.11	192.168.1.6	FTP	96	Response: 331 Contraseña necesaria para samurt123
264	28.483383	192.168.1.6	192.168.1.11	FTP	69	Request: PASS password
266	28.495853	192.168.1.11	192.168.1.6	FTP	87	Response: 230 Usuario samurt123 conectado
268	28.496000	192.168.1.6	192.168.1.11	FTP	70	Request: CLNT FileZilla
270	28.498000	192.168.1.11	192.168.1.6	FTP	62	Response: 200 OK
272	28.498200	192.168.1.6	192.168.1.11	FTP	68	Request: OPTS UTF8 ON
274	28.498747	192.168.1.11	192.168.1.6	FTP	76	Response: 200 Configurado UTF8
276	28.503992	192.168.1.6	192.168.1.11	FTP	75	Request: CWD /home/samurt123
278	28.505303	192.168.1.11	192.168.1.6	FTP	93	Response: 250 orden CWD ejecutada correctamente
280	28.505476	192.168.1.6	192.168.1.11	FTP	59	Request: PWD
282	28.505863	192.168.1.11	192.168.1.6	FTP	101	Response: 257 "/home/samurt123" es el directorio actual
284	28.511099	192.168.1.6	192.168.1.11	FTP	62	Request: TYPE I
286	28.511668	192.168.1.11	192.168.1.6	FTP	81	Response: 200 Tipo establecido en I
288	28.511822	192.168.1.6	192.168.1.11	FTP	60	Request: PASV
290	28.515054	192.168.1.11	192.168.1.6	FTP	105	Response: 227 Entering Passive Mode (192,168,1,11,147,151).
292	28.515402	192.168.1.6	192.168.1.11	FTP	66	Request: RETR 1.jpg
300	28.518496	192.168.1.11	192.168.1.6	FTP	119	Response: 150 Opening BINARY mode data connection for 1.jpg (90103 bytes)
496	28.533085	192.168.1.11	192.168.1.6	FTP	84	Response: 226 Transferencia completada

Transmission Control Protocol, Src Port: 51515, Dst Port: 21, Seq: 122, Ack: 367, Len: 12 Source Port: 51515 Destination Port: 21 [Stream index: 8] [Stream Packet Number: 51] [Conversation completeness: Incomplete, DATA (15)] [TCP Segment Len: 12] Sequence Number: 122 (relative sequence number) Sequence Number (raw): 2971071619 [Next Sequence Number: 134 (relative sequence number)] Acknowledgment Number: 367 (relative ack number) Acknowledgment number (raw): 3456798754 0101 ... = Header Length: 20 bytes (5) Flags: 0x018 (PSH, ACK) Window: 254 [Calculated window size: 65024] [Window size scaling factor: 256] Checksum: 0x8388 [unverified] [Checksum Status: Unverified] Urgent Pointer: 0

Imagen 12: Parte de la captura de tráfico del archivo FTP_download.pcap

Del tráfico encontrado en el archivo FTP_download se puede destacar, entre otras cosas, la sesión FTP en la que el cliente (192.168.1.6) se conecta al servidor ProFTPD en Debian (192.168.1.11) usando el puerto 21 para el canal de control. Posteriormente se recibe el banner 220, intenta usar AUTH TLS/SSL pero obtiene 500 porque no hay soporte de FTPS, luego inicia sesión con el usuario dicho previamente (331 pide contraseña y 230 confirmó acceso), consulta el tipo de sistema (215 UNIX Type: L8), activa UTF-8 (200), verifica el directorio (257 /home/samurt123), cambia el modo de transferencia a binario (TYPE I, 200 OK) y abre el canal de datos en modo pasivo (227 Entering Passive Mode) para listar archivos (150 apertura, 226 transferencia completada); todo viaja en texto plano (usuario, contraseña y comandos), evidenciando que FTP es inseguro.

Por otra parte, se cargó un archivo en el servidor FTP, “Adivinar el pais”.pptx, y se capturo el tráfico como se ve a continuación:

No.	Time	Source	Destination	Protocol	Length	Info
110	16.089772	192.168.1.6	192.168.1.11	FTP	69	Request: PASS password
112	16.103379	192.168.1.11	192.168.1.6	FTP	87	Response: 230 Usuario samurt123 conectado
114	16.103630	192.168.1.6	192.168.1.11	FTP	70	Request: CLNT FileZilla
116	16.104046	192.168.1.11	192.168.1.6	FTP	62	Response: 200 OK
118	16.104188	192.168.1.6	192.168.1.11	FTP	68	Request: OPTS UTF8 ON
120	16.104538	192.168.1.11	192.168.1.6	FTP	76	Response: 200 Configurado UTF8
122	16.106282	192.168.1.6	192.168.1.11	FTP	75	Request: CWD /home/samurt123
124	16.106755	192.168.1.11	192.168.1.6	FTP	93	Response: 250 orden CWD ejecutada correctamente
128	16.179527	192.168.1.6	192.168.1.11	FTP	62	Request: TYPE I
130	16.180143	192.168.1.11	192.168.1.6	FTP	81	Response: 200 Tipo establecido en I
132	16.180461	192.168.1.6	192.168.1.11	FTP	60	Request: PASV
134	16.184030	192.168.1.11	192.168.1.6	FTP	105	Response: 227 Entering Passive Mode (192,168,1,11,155,243).
136	16.184509	192.168.1.6	192.168.1.11	FTP	83	Request: STOR Adivinar el país.pptx
138	16.195741	192.168.1.11	192.168.1.6	FTP	138	Response: 150 Abriendo conexión de datos en modo BINARY para Adivinar el país.pptx
13294	28.589225	192.168.1.11	192.168.1.6	FTP	84	Response: 226 Transferencia completada
13296	28.604460	192.168.1.6	192.168.1.11	FTP	60	Request: PASV
13298	28.605900	192.168.1.11	192.168.1.6	FTP	104	Response: 227 Entering Passive Mode (192,168,1,11,131,21).
13300	28.606147	192.168.1.6	192.168.1.11	FTP	60	Request: MLSD
13308	28.607283	192.168.1.11	192.168.1.6	FTP	112	Response: 150 Abriendo conexión de datos en modo BINARY para MLSD
13320	28.609126	192.168.1.11	192.168.1.6	FTP	84	Response: 226 Transferencia completada

```

> Frame 136: 83 bytes on wire (664 bits), 83 bytes captured (664 bits)
> Ethernet II, Src: Intel_9d:ab:5c (14:85:7f:9d:ab:5c), Dst: Vmware_40:cf:72 (00:0c:29:40:cf:72)
> Internet Protocol Version 4, Src: 192.168.1.6, Dst: 192.168.1.11
> Transmission Control Protocol, Src Port: 56108, Dst Port: 21, Seq: 117, Ack: 320, Len: 29
> File Transfer Protocol (FTP)
  [Current working directory: /home/samurt123]
  [Command response frames: 4208]
  [Command response bytes: 38324529]
  [Command response first frame: 144]
  [Command response last frame: 13284]
  [Response duration: 12401ms]
  [Response bitrate: 24723Kbps]
  [Setup frame: 134]

```

Imagen 13: Parte de la captura del tráfico del archivo FTP_upload.pcap

De esto, se puede decir que la diferencia principal entre subir y descargar un archivo está en el comando y en la dirección del flujo de datos: al subir se usa STOR, lo que hace que el cliente envíe el archivo hacia el servidor (flujo cliente → servidor), mientras que al descargar se usa RETR, con el servidor enviando el archivo al cliente (flujo servidor → cliente); en ambos casos el control va por el puerto 21 y el servidor responde con códigos como 150 al abrir la conexión de datos y 226 al confirmar la transferencia, pero en la subida los bytes viajan desde el cliente y en la descarga desde el servidor.

c. Análisis del tráfico FTP

Después de haber hecho un uso general del servidor FTP y haber capturado el tráfico se pudo observar que:

- La capa de aplicación en FTP tiene varios comandos que le indican al servidor que hacer, en este sentido, se identificó que RETR <archivo> se usa para descargar un archivo, mientras que STOR <archivo> se usa para subir un archivo como se muestra en la imagen:

No.	Time	Source	Destination	Protocol	Length	Info
292	66 bytes on wire (528 bits), 66 bytes captured (528 bits)					
		Ethernet II, Src: Intel_9d:ab:5c (14:85:7f:9d:ab:5c), Dst: Intel_9d:ab:5c (14:85:7f:9d:ab:5c)				
		Internet Protocol Version 4, Src: 192.168.1.6, Dst: 192.168.1.11				
		Transmission Control Protocol, Src Port: 51515, Dst Port: 21, Seq: 122, Ack: 367, Len: 12				
		File Transfer Protocol (FTP)				
		Request command: RETR				
		Request arg: 1.jpg				
		[Current working directory: /home/samurt123]				
		[Command response frames: 62]				
		[Command response bytes: 90103]				
		[Command response first frame: 302]				
		[Command response last frame: 484]				
		[Response duration: 4ms]				
		[Response bitrate: 180206Kbps]				
		[Setup frame: 230]				

```

> Frame 136: 83 bytes on wire (664 bits), 83 bytes captured (664 bits)
> Ethernet II, Src: Intel_9d:ab:5c (14:85:7f:9d:ab:5c), Dst: Vmware_40:cf:72 (00:0c:29:40:cf:72)
> Internet Protocol Version 4, Src: 192.168.1.6, Dst: 192.168.1.11
> Transmission Control Protocol, Src Port: 56108, Dst Port: 21, Seq: 117, Ack: 320, Len: 29
> File Transfer Protocol (FTP)
  Request command: STOR
  Request arg: Adivinar el país.pptx
  [Current working directory: /home/samurt123]
  [Command response frames: 4208]
  [Command response bytes: 38324529]
  [Command response first frame: 144]
  [Command response last frame: 13284]
  [Response duration: 12401ms]
  [Response bitrate: 24723Kbps]
  [Setup frame: 134]

```

Imagen 14: Comandos de la capa de aplicación de FTP.

Adicionalmente, junto a los comandos viaja información para ejecutar el servicio, esta es, la dirección y la carpeta a donde debe ir el archivo junto con datos como la duración de la respuesta, el bitrate, bytes, frames, etc.

Por otra parte, el protocolo de la capa de transporte utilizado por las peticiones del servidor FTP corresponde a TCP (Transmission Control Protocol). Esto se debe a que FTP requiere una comunicación confiable, en la que todos los segmentos enviados deben llegar de manera ordenada y sin pérdidas.

Al analizar los paquetes capturados se pueden observar las banderas (flags) de TCP —como SYN, ACK, FIN o PSH—, que cumplen funciones específicas en el control de la conexión. Estas flags permiten establecer el 3-way-handshake, mantener la sesión activa y, finalmente, cerrarla de manera ordenada. Además, reflejan en cada instante el estado de la comunicación entre cliente y servidor.

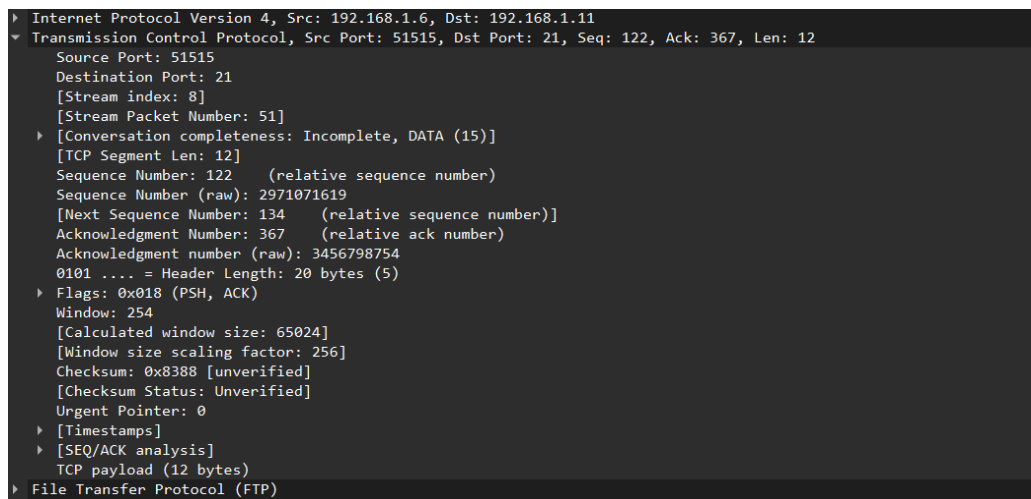
De igual forma, se identifican los campos de encabezado característicos de TCP:

- Número de secuencia (Sequence Number): indica el orden de los bytes enviados, asegurando que los datos lleguen de manera secuencial.
- Número de acuse de recibo (Acknowledgment Number): confirma la correcta recepción de la información, indicando el siguiente byte que espera recibir el host.
- Longitud de los datos (Length o Payload Length): muestra cuántos bytes útiles se transportan dentro de un segmento.

Durante la transmisión estos valores van cambiando dinámicamente, ya que reflejan el flujo de datos en tiempo real: cuando un segmento se envía, el número de secuencia avanza; cuando se recibe correctamente, el acknowledgment responde con la confirmación correspondiente.

*** NOTA: La explicación detallada de TCP y uso del DNS solo se hará en esta sección ya que es la misma para todos los protocolos que usan TCP.**

Adicionalmente, FTP usa 2 canales paralelos de TCP que corresponden a sus 2 puertos, 1 canal para control y otro para datos. A continuación se puede ver la descripción provista por Wireshark de la capa de transporte:



```
Internet Protocol Version 4, Src: 192.168.1.6, Dst: 192.168.1.11
Transmission Control Protocol, Src Port: 51515, Dst Port: 21, Seq: 122, Ack: 367, Len: 12
  Source Port: 51515
  Destination Port: 21
  [Stream index: 8]
  [Stream Packet Number: 51]
  [Conversation completeness: Incomplete, DATA (15)]
  [TCP Segment Len: 12]
  Sequence Number: 122 (relative sequence number)
  Sequence Number (raw): 2971071619
  [Next Sequence Number: 134 (relative sequence number)]
  Acknowledgment Number: 367 (relative ack number)
  Acknowledgment number (raw): 3456798754
  0101 .... = Header Length: 20 bytes (5)
  Flags: 0x018 (PSH, ACK)
  Window: 254
  [Calculated window size: 65024]
  [Window size scaling factor: 256]
  Checksum: 0x8388 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  [Timestamps]
  [SEQ/ACK analysis]
  TCP payload (12 bytes)
File Transfer Protocol (FTP)
```

Imagen 15: Capa de transporte de un paquete con protocolo FTP en Wireshark.

Con el tráfico y las imágenes dadas se puede identificar que el servicio FTP efectivamente usa el puerto 20 para transferir datos y el puerto 21 para control. No obstante, FileZilla usa FTP Pasivo, como se ve la imagen 13, por lo que abre un puerto efímero y se lo comunica al cliente.

No.	Time	Source	Destination	Protocol	Length	Info
77	16.997578	192.168.1.11	192.168.1.1	DNS	84	Standard query 0xd46 PTR 6.1.168.192.in-addr.arpa
78	16.997589	192.168.1.11	192.168.1.1	DNS	84	Standard query 0xd46 PTR 6.1.168.192.in-addr.arpa
79	17.005777	192.168.1.11	192.168.1.1	DNS	109	Standard query response 0xd46 PTR 6.1.168.192.in-addr.arpa PTR 192.168.1.6
245	28.451514	192.168.1.11	192.168.1.1	DNS	84	Standard query 0xf0ee PTR 6.1.168.192.in-addr.arpa
246	28.451531	192.168.1.11	192.168.1.1	DNS	84	Standard query 0xf0ee PTR 6.1.168.192.in-addr.arpa
247	28.473233	192.168.1.1	192.168.1.11	DNS	109	Standard query response 0xf0ee PTR 6.1.168.192.in-addr.arpa PTR 192.168.1.6

Imagen 15.2: Tráfico DNS durante las pruebas con el servidor DNS

Como apartado especial, se documenta también el uso de DNS en las pruebas realizadas sobre el servidor FTP, donde se evidencia que, de forma subyacente, este servicio depende igualmente de la resolución de nombres. Cabe resaltar que esta explicación corresponde a la última ocasión en que se analiza el funcionamiento de nuestro servidor DNS dentro del laboratorio: la primera se presentó en la prueba de conectividad al servidor WEB en apartados anteriores y la segunda, y última, en este escenario con FTP. Fuera de estas dos instancias, no se emplea nuevamente el DNS propio en el desarrollo del informe.

4. Actividad 8.4: Análisis de tráfico del Servidor WEB.

Como se ha realizado previamente, se preparó el ambiente para capturar el tráfico para el servidor WEB, esto es, vaciar el caché del DNS y cerrar todas las aplicaciones que puedan generar ruido durante la captura.

a. Conexión al servidor HTTP

A pesar de haber autofirmado nuestro servidor WEB para generar un certificado y tener habilitados los puertos 80 (HTTP) y 443 (HTTPS), para esta parte del informe solo se analizó el protocolo solicitado en el enunciado, es decir, HTTP, lo que da razón a que en el tráfico capturado no existan registros para el puerto 443.

No.	Time	Source	Destination	Protocol	Length	Info
899	7.899528	192.168.1.6	192.168.1.10	HTTP	652	GET / HTTP/1.1
993	8.075704	192.168.1.10	192.168.1.6	HTTP	594	HTTP/1.1 200 OK (text/html)

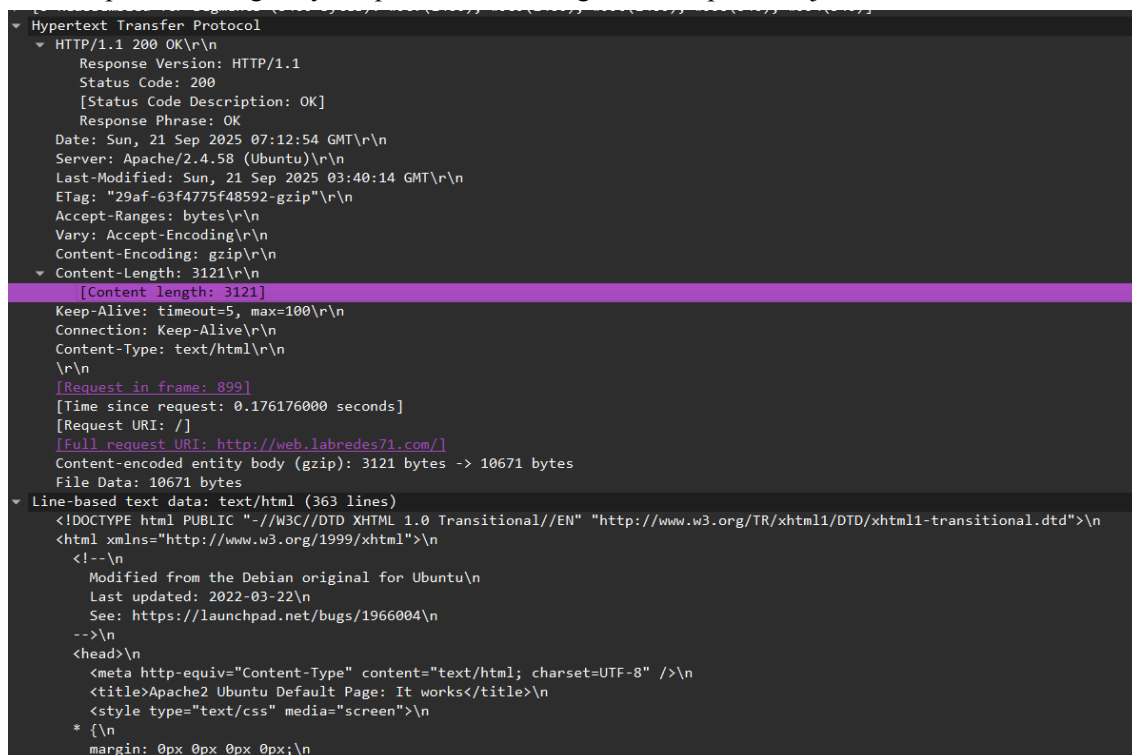
Imagen 16: Archivo HTTP_view filtrado por el protocolo http

De esta manera, la imagen 14 muestra los paquetes HTTP usados durante la comunicación y se puede ejemplificar las respuestas del protocolo HTTP (200 para OK), las peticiones GET, el archivo html, el puerto, TCP, entre otras cosas que se detallarán en la siguiente sección.

b. Análisis del tráfico HTTP

Para analizar el archivo HTTP_view.pcap que cuenta con el tráfico generado de acceder al servicio web se filtró por el protocolo HTTP, dejando los 2 paquetes de la imagen 16, los cuales corresponden a la petición HTTP enviada por el cliente 192.168.1.6 al servidor 192.168.1.10, utilizando el método GET para solicitar el recurso principal (/) bajo el protocolo HTTP/1.1. Posteriormente, el servidor responde desde la dirección 192.168.1.10 hacia el cliente con un mensaje HTTP/1.1 200 OK, indicando que la solicitud fue exitosa, y comienza a transferir el contenido de la página en formato HTML (text/html). En más detalle, analizando las capas de cada paquete se identificó que:

- La capa de aplicación de los paquetes capturados contiene la información solicitada así como aspectos técnicos, entre los cuales se encuentran el código de estado de la solicitud, el servidor, la última modificación (importante para el GET condicional del objeto solicitado), y el archivo comprimido (gzip) para hacerlo más ligero y agilizar el tiempo de transmisión. Adicionalmente, cabe resaltar que el .html viaja en texto plano, corroborando que HTTP no es un protocolo seguro y ataques como sniffing o MITM pueden ejecutarse fácilmente.



```
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    Response Version: HTTP/1.1
    Status Code: 200
    [Status Code Description: OK]
    Response Phrase: OK
    Date: Sun, 21 Sep 2025 07:12:54 GMT\r\n
    Server: Apache/2.4.58 (Ubuntu)\r\n
    Last-Modified: Sun, 21 Sep 2025 03:40:14 GMT\r\n
    ETag: "29af-63f4775f48592-gzip"\r\n
    Accept-Ranges: bytes\r\n
    Vary: Accept-Encoding\r\n
    Content-Encoding: gzip\r\n
  Content-Length: 3121\r\n
  [Content length: 3121]
  Keep-Alive: timeout=5, max=100\r\n
  Connection: Keep-Alive\r\n
  Content-Type: text/html\r\n
  \r\n
  [Request in frame: 899]
  [Time since request: 0.176176000 seconds]
  [Request URI: /]
  [Full request URI: http://web.labredes71.com/]
  Content-encoded entity body (gzip): 3121 bytes -> 10671 bytes
  File Data: 10671 bytes
Line-based text data: text/html (363 lines)
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">\n
<html xmlns="http://www.w3.org/1999/xhtml">\n
  <!--\n
    Modified from the Debian original for Ubuntu\n
    Last updated: 2022-03-22\n
    See: https://launchpad.net/bugs/1966004\n
  -->\n
  <head>\n
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />\n
    <title>Apache2 Ubuntu Default Page: It works</title>\n
    <style type="text/css" media="screen">\n
  * {\n
    margin: 0px 0px 0px 0px;\n
```

Imagen 17: Capa de aplicación de un paquete de http vista en wireshark.

- En la captura se observa la comunicación de TCP entre el servidor (puerto 80) y el cliente, donde se envían 540 bytes de datos correspondientes al contenido HTTP; los números de secuencia y ACK evidencian cómo TCP garantiza la entrega ordenada. Además, Wireshark indica que este segmento forma parte del reensamblaje de varios fragmentos, lo que ilustra la función de TCP de dividir y reconstruir la información; finalmente, el RTT, calculado como el tiempo entre el envío de un segmento y la recepción de su ACK, refleja la eficiencia de la red, pues un RTT bajo como el observado en la red local asegura una comunicación ágil y confiable.

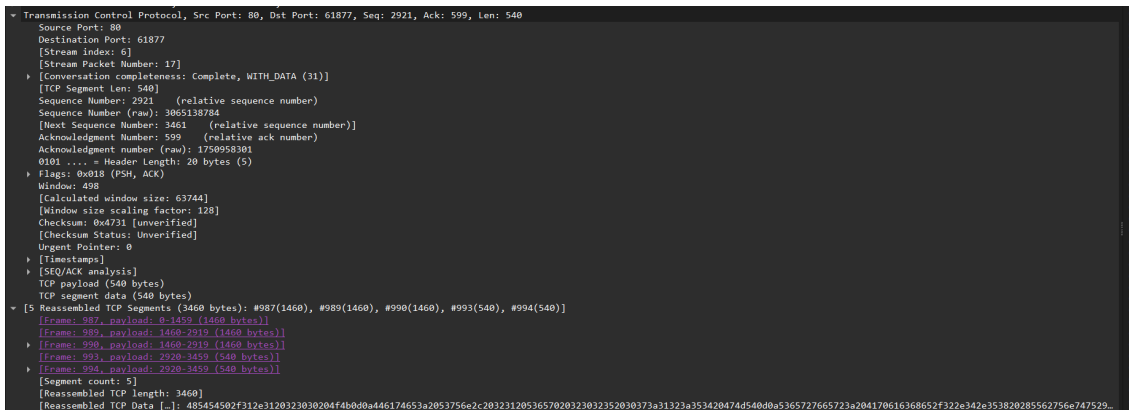


Imagen 18: Capa de transporte de un paquete de http vista en wireshark.

- En cuanto a los puertos, a modo de resumen, el servicio web utiliza el puerto 80, siendo así que si se filtraran los paquetes en wireshark no por el protocolo http sino por `tcp.port == 80`, el resultado sería el mismo.

5. Actividad 8.5: Análisis del protocolo HTTPS realizando navegación de sitios web.

Una vez más, se vació el caché del DNS, se cerraron las aplicaciones que consumen recursos de red y se dejó todo preparado para iniciar las diferentes capturas de tráfico.

[illegible]

Imagen 19: Uso de los servidores DNS en HTTPS.

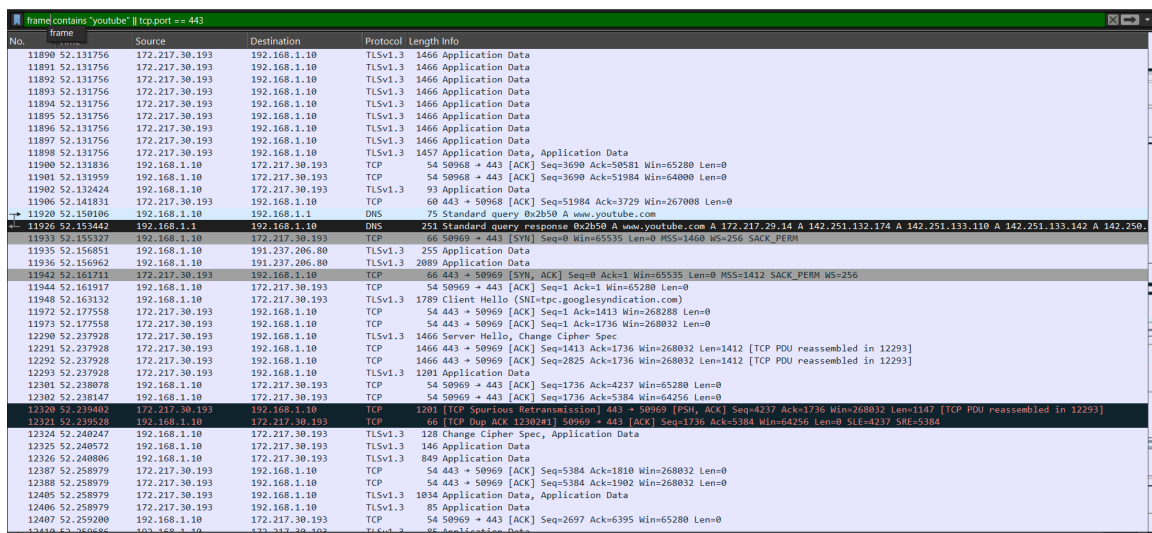
En la imagen se muestran dos capturas de Wireshark filtradas por DNS, correspondientes a los archivos YouTube_view.pcap y HTTPS_view.pcap. En ambos casos se observa cómo, antes de establecer conexiones HTTPS, los clientes deben resolver los nombres de dominio de los distintos servicios a sus respectivas direcciones IP. En la traza de YouTube se evidencian consultas a dominios como youtube.com, ytimg.com, googleapis.com o doubleclick.net, todos necesarios para la reproducción de videos y carga de contenido asociado. En la traza de navegación HTTPS aparecen resoluciones hacia elespectador.com, uniandes.edu.co, eltiempo.com, bancolombia.com y otros

servicios externos, reflejando cómo el DNS actúa como paso previo indispensable para acceder a recursos seguros.

En ambos casos, el servidor DNS consultado corresponde a la dirección 192.168.1.1, que en este escenario funciona como el servidor DNS local de la red (posiblemente el router o gateway del laboratorio), encargado de recibir las peticiones de resolución de nombres y reenviarlas a servidores autoritativos externos en caso de ser necesario.

a. Conexión a YouTube.

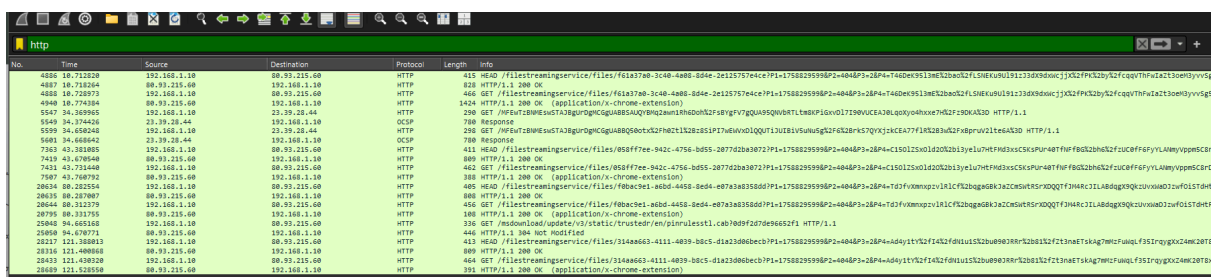
En una primera instancia se estableció una conexión a YouTube desde un navegador que nunca había sido utilizado previamente. Esto permitió iniciar sesión en Gmail y acceder a YouTube sin contar con caché de dominios ni cookies habituales, garantizando que todo el tráfico HTTPS generado pudiera observarse de manera limpia en Wireshark. A partir de allí se reprodujeron tres videos y se publicaron comentarios breves en cada uno, con el fin de analizar la generación y captura de paquetes durante dichas interacciones como se ve a continuación:



No.	frame	Source	Destination	Protocol	Length	Info
11890	52.131756	172.217.30.193	192.168.1.10	TLSv1.3	1466	Application Data
11891	52.131756	172.217.30.193	192.168.1.10	TLSv1.3	1466	Application Data
11892	52.131756	172.217.30.193	192.168.1.10	TLSv1.3	1466	Application Data
11893	52.131756	172.217.30.193	192.168.1.10	TLSv1.3	1466	Application Data
11894	52.131756	172.217.30.193	192.168.1.10	TLSv1.3	1466	Application Data
11895	52.131756	172.217.30.193	192.168.1.10	TLSv1.3	1466	Application Data
11896	52.131756	172.217.30.193	192.168.1.10	TLSv1.3	1466	Application Data
11897	52.131756	172.217.30.193	192.168.1.10	TLSv1.3	1466	Application Data
11898	52.131756	172.217.30.193	192.168.1.10	TLSv1.3	1457	Application Data, Application Data
11900	52.131836	192.168.1.10	172.217.30.193	TCP	54	50968 → 443 [ACK] Seq=3690 Ack=50581 Win=65280 Len=0
11901	52.131959	192.168.1.10	172.217.30.193	TCP	54	50968 → 443 [ACK] Seq=3690 Ack=51984 Win=64000 Len=0
11902	52.132424	192.168.1.10	172.217.30.193	TLSv1.3	93	Application Data
11906	52.141831	172.217.30.193	192.168.1.10	TCP	60	443 → 50968 [ACK] Seq=51984 Ack=3729 Win=267008 Len=0
11920	52.150106	192.168.1.10	192.168.1.1	DNS	75	Standard query 0x2b50 A www.youtube.com
11926	52.153442	192.168.1.1	192.168.1.10	DNS	251	Standard query response 0x2b50 A www.youtube.com A 172.217.29.14 A 142.251.132.174 A 142.251.133.110 A 142.251.133.142 A 142.250.11933
11933	52.155327	192.168.1.10	172.217.30.193	TCP	66	50969 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
11935	52.156851	192.168.1.10	191.237.206.80	TLSv1.3	255	Application Data
11936	52.156962	192.168.1.10	191.237.206.80	TLSv1.3	2089	Application Data
11942	52.161711	172.217.30.193	192.168.1.10	TCP	66	443 → 50969 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM WS=256
11944	52.161917	192.168.1.10	172.217.30.193	TCP	54	50969 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0
11948	52.161313	172.217.30.193	192.168.1.10	TLSv1.3	1789	Client Hello (SHA256+AESGCM+TLSv1.3)
11972	52.177558	172.217.30.193	192.168.1.10	TCP	54	443 → 50969 [ACK] Seq=1 Ack=1413 Win=268288 Len=0
11973	52.177558	172.217.30.193	192.168.1.10	TCP	54	443 → 50969 [ACK] Seq=1 Ack=1736 Win=268032 Len=0
12290	52.237928	172.217.30.193	192.168.1.10	TLSv1.3	1466	Server Hello, Change Cipher Spec
12291	52.237928	172.217.30.193	192.168.1.10	TCP	1466	443 → 50969 [ACK] Seq=1413 Ack=1736 Win=268032 Len=1412 [TCP PDU reassembled in 12293]
12292	52.237928	172.217.30.193	192.168.1.10	TCP	1466	443 → 50969 [ACK] Seq=2825 Ack=1736 Win=268032 Len=1412 [TCP PDU reassembled in 12293]
12293	52.237928	172.217.30.193	192.168.1.10	TLSv1.3	1201	Application Data
12301	52.238078	192.168.1.10	172.217.30.193	TCP	54	50969 → 443 [ACK] Seq=1736 Ack=4237 Win=65280 Len=0
12303	52.238147	172.217.30.193	192.168.1.10	TCP	54	50969 → 443 [ACK] Seq=1736 Ack=5384 Win=64256 Len=0
12320	52.239402	172.217.30.193	192.168.1.10	TCP	1201	[TCP Spurious Retransmission] 443 → 50969 [PSH, ACK] Seq=4237 Ack=1736 Win=268032 Len=1147 [TCP PDU reassembled in 12293]
12321	52.239528	192.168.1.10	172.217.30.193	TCP	66	[TCP Dup ACK 12302#1] 50969 → 443 [ACK] Seq=1736 Ack=5384 Win=64256 Len=0 SIF=4237 SRE=5384
12324	52.240247	192.168.1.10	172.217.30.193	TLSv1.3	128	Change Cipher Spec, Application Data
12325	52.240572	192.168.1.10	172.217.30.193	TLSv1.3	146	Application Data
12326	52.240806	192.168.1.10	172.217.30.193	TLSv1.3	849	Application Data
12387	52.258979	172.217.30.193	192.168.1.10	TCP	54	443 → 50969 [ACK] Seq=5384 Ack=1810 Win=268032 Len=0
12388	52.258979	172.217.30.193	192.168.1.10	TCP	54	443 → 50969 [ACK] Seq=5384 Ack=1902 Win=268032 Len=0
12405	52.258979	172.217.30.193	192.168.1.10	TLSv1.3	1034	Application Data, Application Data
12406	52.258979	172.217.30.193	192.168.1.10	TLSv1.3	85	Application Data
12407	52.259200	192.168.1.10	172.217.30.193	TCP	54	50969 → 443 [ACK] Seq=2697 Ack=6395 Win=65280 Len=0
12410	52.260666	192.168.1.10	172.217.30.193	TLSv1.3	85	Application Data

Imagen 20: Parte del tráfico del archivo YouTube_view.pcap

En la Imagen se observan paquetes correspondientes a solicitudes de información de páginas, recursos asociados y fragmentos de video, visibles mediante métodos como GET y HEAD; sin embargo, no aparecen acciones interactivas como comentarios, inicio de sesión o “me gusta”, ya que YouTube protege estos procesos con HTTPS, de modo que los métodos POST y otros que transmiten datos sensibles viajan dentro de un túnel TLS cifrado, lo cual impide ver su contenido en texto plano en Wireshark y garantiza la seguridad y confidencialidad de la comunicación.



No.	Time	Source	Destination	Protocol	Length	Info
4886	18.712820	192.168.1.10	192.168.1.10	HTTP	415	HEAD /filestreamservice/files/61a17b0-3c40-4a80-8d4e-2e125757eace?P1=175829598P2=4048P3=28P4=740de95138e3ba2cf1508kuh13113383ducj1x2?PK32y32fcqgvtfa13toemjvvsj HTTP/1.1 200 OK
4887	18.712824	192.168.1.10	192.168.1.10	HTTP	415	HEAD /filestreamservice/files/61a17b0-3c40-4a80-8d4e-2e125757eace?P1=175829598P2=4048P3=28P4=740de95138e3ba2cf1508kuh13113383ducj1x2?PK32y32fcqgvtfa13toemjvvsj HTTP/1.1 200 OK
4888	18.712828	192.168.1.10	192.168.1.10	HTTP	415	HEAD /filestreamservice/files/61a17b0-3c40-4a80-8d4e-2e125757eace?P1=175829598P2=4048P3=28P4=740de95138e3ba2cf1508kuh13113383ducj1x2?PK32y32fcqgvtfa13toemjvvsj HTTP/1.1 200 OK
4940	18.774304	88.33.215.60	192.168.1.10	HTTP	1424	HTTP/1.1 200 OK (application/x-chrome-extension)
5547	34.209955	192.168.1.10	192.168.1.10	HTTP	200	GET /filestreamservice/files/61a17b0-3c40-4a80-8d4e-2e125757eace?P1=175829598P2=4048P3=28P4=740de95138e3ba2cf1508kuh13113383ducj1x2?PK32y32fcqgvtfa13toemjvvsj HTTP/1.1 200 OK
5549	34.274426	23.19.28.44	192.168.1.10	OCSP	788	Response
5550	34.468442	192.168.1.10	23.19.28.44	HTTP	200	GET /filestreamservice/files/61a17b0-3c40-4a80-8d4e-2e125757eace?P1=175829598P2=4048P3=28P4=740de95138e3ba2cf1508kuh13113383ducj1x2?PK32y32fcqgvtfa13toemjvvsj HTTP/1.1 200 OK
5601	34.468442	23.19.28.44	192.168.1.10	OCSP	788	Response
7363	43.381885	192.168.1.10	88.33.215.60	HTTP	415	HEAD /filestreamservice/files/61a17b0-3c40-4a80-8d4e-2e125757eace?P1=175829598P2=4048P3=28P4=740de95138e3ba2cf1508kuh13113383ducj1x2?PK32y32fcqgvtfa13toemjvvsj HTTP/1.1 200 OK
7413	43.470640	88.33.215.60	192.168.1.10	HTTP	415	HEAD /filestreamservice/files/61a17b0-3c40-4a80-8d4e-2e125757eace?P1=175829598P2=4048P3=28P4=740de95138e3ba2cf1508kuh13113383ducj1x2?PK32y32fcqgvtfa13toemjvvsj HTTP/1.1 200 OK
7431	43.731440	192.168.1.10	88.33.215.60	HTTP	415	HEAD /filestreamservice/files/61a17b0-3c40-4a80-8d4e-2e125757eace?P1=175829598P2=4048P3=28P4=740de95138e3ba2cf1508kuh13113383ducj1x2?PK32y32fcqgvtfa13toemjvvsj HTTP/1.1 200 OK
7507	43.760792	88.33.215.60	192.168.1.10	HTTP	415	HEAD /filestreamservice/files/61a17b0-3c40-4a80-8d4e-2e125757eace?P1=175829598P2=4048P3=28P4=740de95138e3ba2cf1508kuh13113383ducj1x2?PK32y32fcqgvtfa13toemjvvsj HTTP/1.1 200 OK
28634	88.225554	192.168.1.10	88.33.215.60	HTTP	415	HEAD /filestreamservice/files/61a17b0-3c40-4a80-8d4e-2e125757eace?P1=175829598P2=4048P3=28P4=740de95138e3ba2cf1508kuh13113383ducj1x2?PK32y32fcqgvtfa13toemjvvsj HTTP/1.1 200 OK
28635	88.225607	88.33.215.60	192.168.1.10	HTTP	415	HEAD /filestreamservice/files/61a17b0-3c40-4a80-8d4e-2e125757eace?P1=175829598P2=4048P3=28P4=740de95138e3ba2cf1508kuh13113383ducj1x2?PK32y32fcqgvtfa13toemjvvsj HTTP/1.1 200 OK
28644	88.232779	192.168.1.10	88.33.215.60	HTTP	415	HEAD /filestreamservice/files/61a17b0-3c40-4a80-8d4e-2e125757eace?P1=175829598P2=4048P3=28P4=740de95138e3ba2cf1508kuh13113383ducj1x2?PK32y32fcqgvtfa13toemjvvsj HTTP/1.1 200 OK
28795	88.231755	192.168.1.10	192.168.1.10	HTTP	180	HTTP/1.1 200 OK (application/x-chrome-extension)
29445	84.465165	88.33.215.60	192.168.1.10	HTTP	330	HEAD /filestreamservice/files/61a17b0-3c40-4a80-8d4e-2e125757eace?P1=175829598P2=4048P3=28P4=740de95138e3ba2cf1508kuh13113383ducj1x2?PK32y32fcqgvtfa13toemjvvsj HTTP/1.1 200 OK
29650	84.470771	88.33.215.60	192.168.1.10	HTTP	444	HTTP/1.1 304 Not Modified
29217	121.308813	192.168.1.10	88.33.215.60	HTTP	415	HEAD /filestreamservice/files/61a17b0-3c40-4a80-8d4e-2e125757eace?P1=175829598P2=4048P3=28P4=740de95138e3ba2cf1508kuh13113383ducj1x2?PK32y32fcqgvtfa13toemjvvsj HTTP/1.1 200 OK
28316	121.488658	88.33.215.60	192.168.1.10	HTTP	880	HTTP/1.1 200 OK
28433	121.498308	192.168.1.10	88.33.215.60	HTTP	464	GET /filestreamservice/files/61a17b0-3c40-4a80-8d4e-2e125757eace?P1=175829598P2=4048P3=28P4=740de95138e3ba2cf1508kuh13113383ducj1x2?PK32y32fcqgvtfa13toemjvvsj HTTP/1.1 200 OK
28439	121.520508	192.168.1.10	88.33.215.60	HTTP	395	HTTP/1.1 200 OK (application/x-chrome-extension)

Imagen 21: Filtro por http del archivo YouTube_view.pcap

b. Conexión a sitios web

En segunda instancia, con el mismo navegador se visitaron los sitios web solicitados en la guía, elespectador, uniandes, el tiempo y bancolombia y de la misma manera, se obtuvo el siguiente tráfico donde se puede ver intercambio de paquetes TCP y TLS (todas estas paginas usan https) y en el primer paquete el ejemplo con elespectador:

No.	Time	Source	Destination	Protocol	Length	Info
335	22.101412	192.168.1.10	181.54.160.137	TLSv1.3	1848	Client Hello (SHA1=www.elespectador.com)
336	22.111424	192.168.1.10	181.54.160.137	TCP	1438	[TCP Retransmission] 51034 → 443 [PSH, ACK] Seq=411 Ack=1 Win=65280 Len=1384
338	22.116518	181.54.160.137	192.168.1.10	TCP	54	443 → 51034 [ACK] Seq=1 Ack=1385 Win=67072 Len=0
339	22.116518	181.54.160.137	192.168.1.10	TCP	54	443 → 51034 [ACK] Seq=1 Ack=1795 Win=69888 Len=0
340	22.116518	181.54.160.137	192.168.1.10	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data
341	22.116518	181.54.160.137	192.168.1.10	TCP	1514	443 → 51034 [PSH, ACK] Seq=1461 Ack=1795 Win=69888 Len=1460 [TCP PDU reassembled in 342]
342	22.116518	181.54.160.137	192.168.1.10	TLSv1.3	451	Application Data, Application Data, Application Data
343	22.116659	192.168.1.10	181.54.160.137	TCP	54	51034 → 443 [ACK] Seq=1795 Ack=3318 Win=65280 Len=0
344	22.120645	181.54.160.137	192.168.1.10	TCP	66	[TCP Dup ACK 339#1] 443 → 51034 [ACK] Seq=3318 Ack=1795 Win=69888 Len=0 SIF=411 SRE=1795
347	22.122483	192.168.1.10	20.42.65.85	TLSv1.2	452	Application Data
348	22.122579	192.168.1.10	20.42.65.85	TLSv1.2	93	Application Data
349	22.122616	192.168.1.10	20.42.65.85	TLSv1.2	7902	Application Data
350	22.123748	192.168.1.10	181.54.160.137	TCP	66	51035 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
351	22.124055	192.168.1.10	20.42.65.85	TLSv1.2	411	Application Data
352	22.124123	192.168.1.10	20.42.65.85	TLSv1.2	8352	Application Data
355	22.128088	181.54.160.137	192.168.1.10	TCP	66	443 → 51035 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1384 SACK_PERM WS=128
356	22.128888	192.168.1.10	181.54.160.137	TCP	54	51035 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0
357	22.129403	192.168.1.10	181.54.160.137	TLSv1.3	1816	Client Hello (SHA1=www.elespectador.com)
360	22.137263	192.168.1.10	181.54.160.137	TLSv1.3	134	Change Cipher Spec, Application Data
361	22.137574	192.168.1.10	181.54.160.137	TLSv1.3	146	Application Data
362	22.137991	192.168.1.10	181.54.160.137	TLSv1.3	555	Application Data
363	22.139112	181.54.160.137	192.168.1.10	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data
364	22.139112	181.54.160.137	192.168.1.10	TCP	1514	443 → 51035 [PSH, ACK] Seq=1461 Ack=1763 Win=64128 Len=1460 [TCP PDU reassembled in 365]
365	22.139112	181.54.160.137	192.168.1.10	TLSv1.3	451	Application Data, Application Data, Application Data
366	22.139206	192.168.1.10	181.54.160.137	TCP	54	51035 → 443 [ACK] Seq=1763 Ack=3318 Win=65280 Len=0
367	22.139342	181.54.160.137	192.168.1.10	TCP	54	443 → 51034 [ACK] Seq=3318 Ack=1875 Win=69888 Len=0
368	22.139342	181.54.160.137	192.168.1.10	TLSv1.3	357	Application Data
369	22.139342	181.54.160.137	192.168.1.10	TLSv1.3	357	Application Data
370	22.139388	192.168.1.10	181.54.160.137	TCP	54	51034 → 443 [ACK] Seq=2468 Ack=3924 Win=64768 Len=0
371	22.140919	192.168.1.10	181.54.160.137	TLSv1.3	134	Change Cipher Spec, Application Data
372	22.141584	181.54.160.137	192.168.1.10	TCP	54	443 → 51034 [ACK] Seq=3924 Ack=1967 Win=69888 Len=0
373	22.142187	181.54.160.137	192.168.1.10	TCP	54	443 → 51034 [ACK] Seq=3924 Ack=2468 Win=72576 Len=0
375	22.143860	181.54.160.137	192.168.1.10	TCP	54	443 → 51035 [ACK] Seq=3318 Ack=1843 Win=64128 Len=0
377	22.143964	192.168.1.10	20.201.14.155	TCP	66	51036 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
378	22.145334	181.54.160.137	192.168.1.10	TLSv1.3	357	Application Data
379	22.145334	181.54.160.137	192.168.1.10	TLSv1.3	357	Application Data
380	22.145407	192.168.1.10	181.54.160.137	TCP	54	51035 → 443 [ACK] Seq=1843 Ack=3924 Win=64768 Len=0
381	22.165919	181.54.160.137	192.168.1.10	TLSv1.3	115	Application Data
382	22.165919	181.54.160.137	192.168.1.10	TLSv1.3	85	Application Data

Imagen 22: Parte del tráfico del archivo HTTPS_view.pcap

c. Análisis del tráfico HTTPS

Para analizar el tráfico se aplicó el filtro correspondiente al puerto de https por medio de la instrucción tcp.port == 443. Con esto, se logró obtener la siguiente información:

- Con relación a la navegación segura, dentro de la capa de aplicación se pudo observar que TLS cifra los datos, de manera que en la imagen a continuación se ve como Encrypted Application Data posee una línea de caracteres de la cual no sé puede saber ni qué tipo de objeto es, ni lo que contiene.

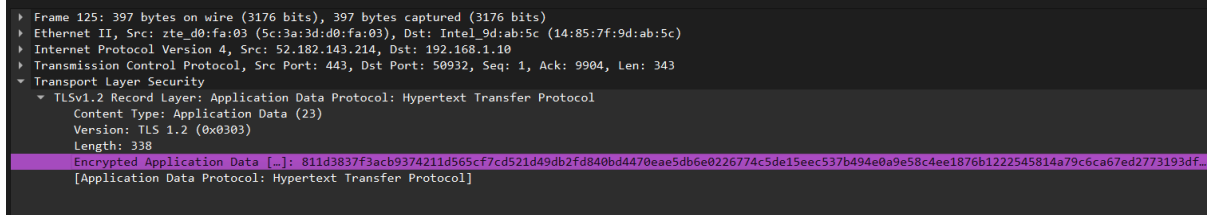


Imagen 23: Capa de aplicación de un paquete TLSv1.2 en Youtube_view.pcap

Adicionalmente, se observa que HTTP está encapsulado sobre TLS, lo cual confirma lo visto en clase en líneas como: “Application Data Protocol: Hypertext Transfer Protocol”. De esta manera, el trabajo de cifrar los datos y gestionar aspectos como el control de sesión corresponde a TLS. Según el modelo lógico OSI, estas funciones se ubican en la capa de Presentación, pero en el modelo práctico TCP/IP se consideran parte de la capa de aplicación. Por ello, este análisis se incluye en este apartado.

- En la capa de transporte, tanto HTTP como HTTPS utilizan TCP, que asegura la entrega confiable y ordenada de datos entre aplicaciones, además del control de flujo y la corrección de errores. La diferencia es que en HTTPS, sobre esa misma base de TCP, se añade TLS para manejar las llaves de sesión y el cifrado, mientras que las banderas y funciones de TCP se mantienen sin cambios.

203	4.273743	192.168.1.10	191.237.206.80	TCP	66	50928 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
204	4.300239	172.217.30.174	192.168.1.10	TLSv1.3	1466	Server Hello, Change Cipher Spec
205	4.300239	172.217.30.174	192.168.1.10	TCP	1466	443 → 50937 [ACK] Seq=1413 Ack=1790 Win=268032 Len=1412 [TCP PDU reassembled in 209]
206	4.300239	172.217.30.174	192.168.1.10	TCP	1466	443 → 50937 [PSH, ACK] Seq=2825 Ack=1790 Win=268032 Len=1412 [TCP PDU reassembled in 209]
207	4.300239	172.217.30.174	192.168.1.10	TCP	1466	443 → 50937 [ACK] Seq=4237 Ack=1790 Win=268032 Len=1412 [TCP PDU reassembled in 209]
208	4.300239	172.217.30.174	192.168.1.10	TCP	1466	443 → 50937 [ACK] Seq=5649 Ack=1790 Win=268032 Len=1412 [TCP PDU reassembled in 209]

Imagen 24: Captura de tráfico de conexión segura TLS sobre TCP.

- Finalmente, se puede notar que el puerto 80 es reemplazado por el puerto 443 que es el estándar global para HTTPS y se usa en toda la comunicación.

6. Actividad 8.6: Análisis del protocolo VoIP.

Se repite el proceso de las actividades anteriores con las aplicaciones que consumen recursos de red y se inició una llamada entre 2 clientes del servidor VoIP mientras se capturaba el tráfico con Wireshark como se puede ver a continuación:

No.	Time	Source	Destination	Protocol	Length	Info
41	1.225648	172.20.10.3	13.107.136.10	TCP	1393	64731 → 443 [ACK] Seq=7054 Ack=5219 Win=8191 Len=1339 [TCP PDU reassembled in 43]
42	1.225648	172.20.10.3	13.107.136.10	TCP	1393	64731 → 443 [ACK] Seq=8393 Ack=5219 Win=8191 Len=1339 [TCP PDU reassembled in 43]
43	1.225648	172.20.10.3	13.107.136.10	TLSv1.2	1378	Application Data
44	1.225685	172.20.10.3	13.107.136.10	TCP	1393	[TCP Retransmission] 64731 → 443 [ACK] Seq=7054 Ack=5219 Win=8191 Len=1339
45	1.225710	172.20.10.3	13.107.136.10	TCP	1393	[TCP Retransmission] 64731 → 443 [ACK] Seq=8393 Ack=5219 Win=8191 Len=1339 [TCP PDU reassembled in 43]
46	1.225725	172.20.10.3	13.107.136.10	TCP	1378	[TCP Retransmission] 64731 → 443 [PSH, ACK] Seq=9732 Ack=5219 Win=8191 Len=1324 [TCP PDU reassembled in 43]
47	1.234556	184.84.145.42	172.20.10.3	TCP	54	80 → 60207 [FIN, ACK] Seq=1 Ack=2 Win=501 Len=0
48	1.234556	172.217.28.99	172.20.10.3	TCP	54	80 → 60206 [FIN, ACK] Seq=1 Ack=2 Win=1053 Len=0
49	1.234556	199.232.178.172	172.20.10.3	TCP	54	80 → 60209 [ACK] Seq=1 Ack=2 Win=288 Len=0
50	1.234556	199.232.178.172	172.20.10.3	TCP	54	80 → 60209 [FIN, ACK] Seq=1 Ack=2 Win=288 Len=0
51	1.234603	172.20.10.3	184.84.145.42	TCP	54	60207 → 80 [ACK] Seq=2 Ack=2 Win=254 Len=0
52	1.234608	172.20.10.3	184.84.145.42	TCP	54	[TCP Dup ACK 51#1] 60207 → 80 [ACK] Seq=2 Ack=2 Win=254 Len=0
53	1.234653	172.20.10.3	172.217.28.99	TCP	54	60206 → 80 [ACK] Seq=2 Ack=2 Win=251 Len=0
54	1.234656	172.20.10.3	172.217.28.99	TCP	54	[TCP Dup ACK 53#1] 60206 → 80 [ACK] Seq=2 Ack=2 Win=251 Len=0
55	1.234677	172.20.10.3	199.232.178.172	TCP	54	60209 → 80 [ACK] Seq=2 Ack=2 Win=255 Len=0
56	1.234679	172.20.10.3	199.232.178.172	TCP	54	[TCP Dup ACK 55#1] 60209 → 80 [ACK] Seq=2 Ack=2 Win=255 Len=0
57	1.345788	13.107.136.10	172.20.10.3	TCP	54	443 → 64731 [ACK] Seq=5219 Ack=8393 Win=16387 Len=0
58	1.345788	13.107.136.10	172.20.10.3	TCP	54	443 → 64731 [ACK] Seq=5219 Ack=11056 Win=16387 Len=0
59	2.704606	172.20.10.1	172.20.10.12	SIP	909	Request: REGISTER sip:172.20.10.12;transport=UDP (1 binding)
60	2.705375	172.20.10.12	172.20.10.1	SIP	617	Status: 401 Unauthorized
61	2.705387	172.20.10.12	172.20.10.1	SIP	617	Status: 401 Unauthorized
62	2.820650	172.20.10.1	172.20.10.12	SIP	909	Request: REGISTER sip:172.20.10.12;transport=UDP (1 binding)
63	2.821765	172.20.10.12	172.20.10.1	SIP	686	Request: OPTIONS sip:1010172.20.10.1:65489;rinstance=dbd05f0f605c769a;transport=UDP
64	2.821778	172.20.10.12	172.20.10.1	SIP	686	Request: OPTIONS sip:1010172.20.10.1:65489;rinstance=dbd05f0f605c769a;transport=UDP
65	2.822516	172.20.10.12	172.20.10.1	SIP	671	Status: 200 OK (REGISTER) (1 binding)
66	2.822524	172.20.10.12	172.20.10.1	SIP	671	Status: 200 OK (REGISTER) (1 binding)
67	2.830353	172.20.10.1	172.20.10.12	SIP	719	Status: 200 OK (OPTIONS)
68	3.463788	13.107.136.10	172.20.10.3	TCP	1393	443 → 64731 [ACK] Seq=5219 Ack=11056 Win=16387 Len=1339 [TCP PDU reassembled in 71]
69	3.463788	13.107.136.10	172.20.10.3	TCP	1393	443 → 64731 [ACK] Seq=6558 Ack=11056 Win=16387 Len=1339 [TCP PDU reassembled in 71]
70	3.463788	13.107.136.10	172.20.10.3	TCP	1393	443 → 64731 [ACK] Seq=7897 Ack=11056 Win=16387 Len=1339 [TCP PDU reassembled in 71]
71	3.463788	13.107.136.10	172.20.10.3	TLSv1.2	829	Application Data
72	3.463788	13.107.136.10	172.20.10.3	TCP	1393	443 → 64731 [ACK] Seq=10011 Ack=11056 Win=16387 Len=1339 [TCP PDU reassembled in 80]
73	3.463788	13.107.136.10	172.20.10.3	TCP	1393	443 → 64731 [ACK] Seq=11350 Ack=11056 Win=16387 Len=1339 [TCP PDU reassembled in 80]
74	3.463788	13.107.136.10	172.20.10.3	TCP	1393	443 → 64731 [ACK] Seq=12689 Ack=11056 Win=16387 Len=1339 [TCP PDU reassembled in 80]
75	3.463788	13.107.136.10	172.20.10.3	TCP	1393	443 → 64731 [ACK] Seq=14028 Ack=11056 Win=16387 Len=1339 [TCP PDU reassembled in 80]
76	3.463859	172.20.10.3	13.107.136.10	TCP	54	64731 → 443 [ACK] Seq=11056 Ack=15367 Win=8191 Len=0
77	3.463865	172.20.10.3	13.107.136.10	TCP	54	[TCP Dup ACK 76#1] 64731 → 443 [ACK] Seq=11056 Ack=15367 Win=8191 Len=0
78	3.467871	13.107.136.10	172.20.10.3	TCP	1393	443 → 64731 [ACK] Seq=15367 Ack=11056 Win=16387 Len=1339 [TCP PDU reassembled in 80]

Imagen 25: Parte del tráfico del archivo VoIP_view.pcap

- En un primer análisis se identificó que en la capa de aplicación se emplea el protocolo SIP (Session Initiation Protocol), utilizado inicialmente para la autenticación de clientes y, posteriormente, para la señalización de llamadas. Tal como se observa en la Imagen 24 y 25, el cliente envía un mensaje INVITE para iniciar la comunicación con otro usuario; dentro de este mensaje también se incluye información del protocolo SDP (Session Description Protocol), que especifica parámetros técnicos como las direcciones IP, los puertos UDP destinados a la transmisión de medios, así como los códecs de audio o video que se usarán durante la sesión. De esta manera, mientras SIP cumple la función de organizar y gestionar la llamada, SDP provee la descripción técnica necesaria para que los flujos de voz o video se transmitan adecuadamente sobre RTP, mostrando la estrecha relación entre protocolos de señalización y transporte en sistemas VoIP.

No.	SIP sipfrag	Source	Destination	Protocol	Length	Info
60	7.785375	172.20.10.1	172.20.10.1	SIP	909	Request: REGISTER sip:172.20.10.12;transport=UDP (1 binding)
61	2.705387	172.20.10.1	172.20.10.1	SIP	617	Status: 401 Unauthorized
62	2.628650	172.20.10.1	172.20.10.12	SIP	909	Request: REGISTER sip:172.20.10.12;transport=UDP (1 binding)
63	2.621765	172.20.10.1	172.20.10.1	SIP	686	Request: OPTIONS sip:100@172.20.10.1:65489;instance=99b3c9e76ea1278;transport=UDP
64	2.621776	172.20.10.1	172.20.10.1	SIP	686	Request: OPTIONS sip:100@172.20.10.1:65489;instance=99b3c9e76ea1278;transport=UDP
65	2.622516	172.20.10.1	172.20.10.1	SIP	671	Status: 200 OK (REGISTER) (1 binding)
66	2.622524	172.20.10.1	172.20.10.1	SIP	671	Status: 200 OK (REGISTER) (1 binding)
67	2.636353	172.20.10.1	172.20.10.12	SIP	719	Status: 200 OK (OPTIONS)
944	4.362237	172.20.10.1	172.20.10.12	SIP/SDP	975	Request: INVITE sip:100@172.20.10.12;transport=UDP
945	4.362895	172.20.10.1	172.20.10.1	SIP	596	Status: 401 Unauthorized
946	4.363110	172.20.10.1	172.20.10.1	SIP	596	Status: 401 Unauthorized
947	4.371313	172.20.10.1	172.20.10.12	SIP	367	Request: ACK sip:100@172.20.10.12;transport=UDP
949	4.408669	172.20.10.1	172.20.10.12	SIP/SDP	1146	Request: INVITE sip:100@172.20.10.12;transport=UDP
949	4.498089	172.20.10.1	172.20.10.1	SIP	574	Status: 100 Trying
950	4.498821	172.20.10.1	172.20.10.1	SIP	574	Status: 100 Trying
951	4.492628	172.20.10.12	172.20.10.2	SIP/SDP	909	Request: INVITE sip:100@172.20.10.2:55973;instance=99b3c9e76ea1278;transport=UDP
952	4.492650	172.20.10.12	172.20.10.2	SIP/SDP	909	Request: INVITE sip:100@172.20.10.2:55973;instance=99b3c9e76ea1278;transport=UDP
953	4.593272	172.20.10.12	172.20.10.2	SIP/SDP	909	Request: INVITE sip:100@172.20.10.2:55973;instance=99b3c9e76ea1278;transport=UDP
954	4.593286	172.20.10.12	172.20.10.2	SIP/SDP	909	Request: INVITE sip:100@172.20.10.2:55973;instance=99b3c9e76ea1278;transport=UDP
955	4.692960	172.20.10.2	172.20.10.12	SIP	346	Status: 100 Trying
975	5.226046	172.20.10.2	172.20.10.12	SIP	671	Status: 100 Ringing
976	5.227033	172.20.10.12	172.20.10.1	SIP	590	Status: 100 Ringing
975	5.227042	172.20.10.12	172.20.10.1	SIP	590	Status: 100 Ringing
1000	7.798410	172.20.10.2	172.20.10.12	SIP/SDP	957	Status: 200 OK (INVITE)
1001	7.791375	172.20.10.12	172.20.10.2	SIP	478	Request: ACK sip:100@172.20.10.2:55973
1002	7.791385	172.20.10.12	172.20.10.1	SIP	478	Request: ACK sip:100@172.20.10.2:55973
1003	7.792279	172.20.10.12	172.20.10.1	SIP/SDP	923	Status: 200 OK (INVITE)
1004	7.792288	172.20.10.12	172.20.10.1	SIP/SDP	923	Status: 200 OK (INVITE)
1006	7.794027	172.20.10.12	172.20.10.2	SIP/SDP	923	Request: INVITE sip:100@172.20.10.2:55973, in-dialog
1006	7.794040	172.20.10.12	172.20.10.2	SIP/SDP	923	Request: INVITE sip:100@172.20.10.2:55973, in-dialog
1008	7.856177	172.20.10.1	172.20.10.12	SIP	443	Request: ACK sip:100@172.20.10.12:5060
1010	7.851348	172.20.10.1	172.20.10.1	SIP/SDP	911	Request: INVITE sip:100@172.20.10.1:65489;transport=UDP, in-dialog
1011	7.851369	172.20.10.1	172.20.10.1	SIP/SDP	911	Request: INVITE sip:100@172.20.10.1:65489;transport=UDP, in-dialog
1014	7.856581	172.20.10.2	172.20.10.12	SIP/SDP	907	Status: 200 OK (INVITE)
1015	7.857604	172.20.10.12	172.20.10.2	SIP	478	Request: ACK sip:100@172.20.10.2:55973
1016	7.857612	172.20.10.12	172.20.10.2	SIP	478	Request: ACK sip:100@172.20.10.2:55973
1017	7.866372	172.20.10.1	172.20.10.12	SIP/SDP	977	Status: 200 OK (INVITE)
1018	7.865174	172.20.10.1	172.20.10.1	SIP	439	Request: ACK sip:100@172.20.10.1:65489;transport=UDP
1019	7.865187	172.20.10.12	172.20.10.1	SIP	439	Request: ACK sip:100@172.20.10.1:65489;transport=UDP
2201	21.668893	172.20.10.2	172.20.10.12	SIP	909	Request: REGISTER sip:172.20.10.12;transport=UDP (1 binding)
2202	21.689980	172.20.10.12	172.20.10.2	SIP	617	Status: 401 Unauthorized
2201	21.689911	172.20.10.12	172.20.10.2	SIP	617	Status: 401 Unauthorized
2204	21.818923	172.20.10.12	172.20.10.12	SIP	909	Request: REGISTER sip:172.20.10.12;transport=UDP (1 binding)
2206	21.812055	172.20.10.12	172.20.10.2	SIP	686	Request: OPTIONS sip:100@172.20.10.2:55973;instance=99b3c9e76ea1278;transport=UDP
2206	21.812069	172.20.10.12	172.20.10.2	SIP	686	Request: OPTIONS sip:100@172.20.10.2:55973;instance=99b3c9e76ea1278;transport=UDP
2207	21.812810	172.20.10.12	172.20.10.2	SIP	671	Status: 200 OK (REGISTER) (1 binding)
2208	21.812823	172.20.10.12	172.20.10.2	SIP	671	Status: 200 OK (REGISTER) (1 binding)
2209	21.855602	172.20.10.1	172.20.10.12	SIP	719	Status: 200 OK (OPTIONS)
3743	40.658988	172.20.10.2	172.20.10.12	SIP	408	Request: BYE sip:100@172.20.10.12:5060
3750	40.660008	172.20.10.12	172.20.10.2	SIP	566	Status: 200 OK (BYE)
3751	40.660015	172.20.10.12	172.20.10.2	SIP	566	Status: 200 OK (BYE)
3752	40.660880	172.20.10.12	172.20.10.1	SIP/SDP	936	Request: INVITE sip:100@172.20.10.1:65489;transport=UDP, in-dialog
3753	40.660893	172.20.10.12	172.20.10.1	SIP/SDP	936	Request: INVITE sip:100@172.20.10.1:65489;transport=UDP, in-dialog
3755	40.675451	172.20.10.1	172.20.10.12	SIP/SDP	977	Status: 200 OK (INVITE)
3755	40.675621	172.20.10.12	172.20.10.1	SIP	439	Request: ACK sip:100@172.20.10.1:65489;transport=UDP

Imagen 26: Tráfico filtrado por SIP en VoIP_view.pcap

Al igual que en varios protocolos, en SIP existen los estados, como en el paquete 955 donde el servidor muestra 100 Trying, es decir, el celular del otro cliente está verificando disponibilidad, en 180 está sonando, el 200 es de OK. Por otro lado, el protocolo RTP (Real-time Transport Protocol) es donde viaja el audio de la llamada VoIP con un identificador de la fuente RTP, un número de secuencia, una marca de tiempo para sincronizar el audio y una bandera que señala el inicio de una nueva trama de audio.

No.	Time	Source	Destination	Protocol	Length	Info
999	7.790410	172.20.10.2	172.20.10.12	RTP	55	PT=Unassigned, SSRC=0x2A926C32, Seq=61866, Time=2300307427
1007	7.827286	172.20.10.1	172.20.10.12	RTP	55	PT=Unassigned, SSRC=0x41A59E5D, Seq=26530, Time=837103597
1009	7.850177	172.20.10.2	172.20.10.12	RTP	214	PT=ITU-T G.711 PCMA, SSRC=0x2A926C32, Seq=61867, Time=2300307427, Mark
1012	7.851050	172.20.10.12	172.20.10.1	RTP	214	PT=ITU-T G.711 PCMA, SSRC=0x66070393, Seq=8035, Time=2300307432, Mark
1013	7.851857	172.20.10.12	172.20.10.1	RTP	214	PT=ITU-T G.711 PCMA, SSRC=0x66070393, Seq=8035, Time=2300307432, Mark
3754	40.675451	172.20.10.1	172.20.10.12	RTP	214	PT=ITU-T G.711 PCMA, SSRC=0x41A59E5D, Seq=28157, Time=837363757, Mark
3761	40.760015	172.20.10.1	172.20.10.12	RTP	214	PT=ITU-T G.711 PCMA, SSRC=0x41A59E5D, Seq=28158, Time=837363917
3762	40.760015	172.20.10.1	172.20.10.12	RTP	214	PT=ITU-T G.711 PCMA, SSRC=0x41A59E5D, Seq=28159, Time=837364077
3763	40.760015	172.20.10.1	172.20.10.12	RTP	214	PT=ITU-T G.711 PCMA, SSRC=0x41A59E5D, Seq=28160, Time=837364237

Imagen 27: Parte del tráfico en VoIP_view.pcap filtrado por rtp

Asimismo el protocolo de la capa de transporte que se usa es UDP ya que una llamada no puede tolerar retrasos en la comunicación en el puerto 5060 para SIP, mientras que RTC usar puertos dinámicos en UDP asignados en la etapa de preparación de la llamada con SIP y SDP (SDP no es un protocolo de transporte, no usa puertos).

7. Actividad 8.7: Análisis del protocolo RTMP.

Como en las otras actividades, se cerraron las aplicaciones que consumen recursos de red y se inició una transmisión en vivo a través del servidor RTMP implementado al mismo tiempo que se capturaba el tráfico como se muestra en la siguiente imagen:

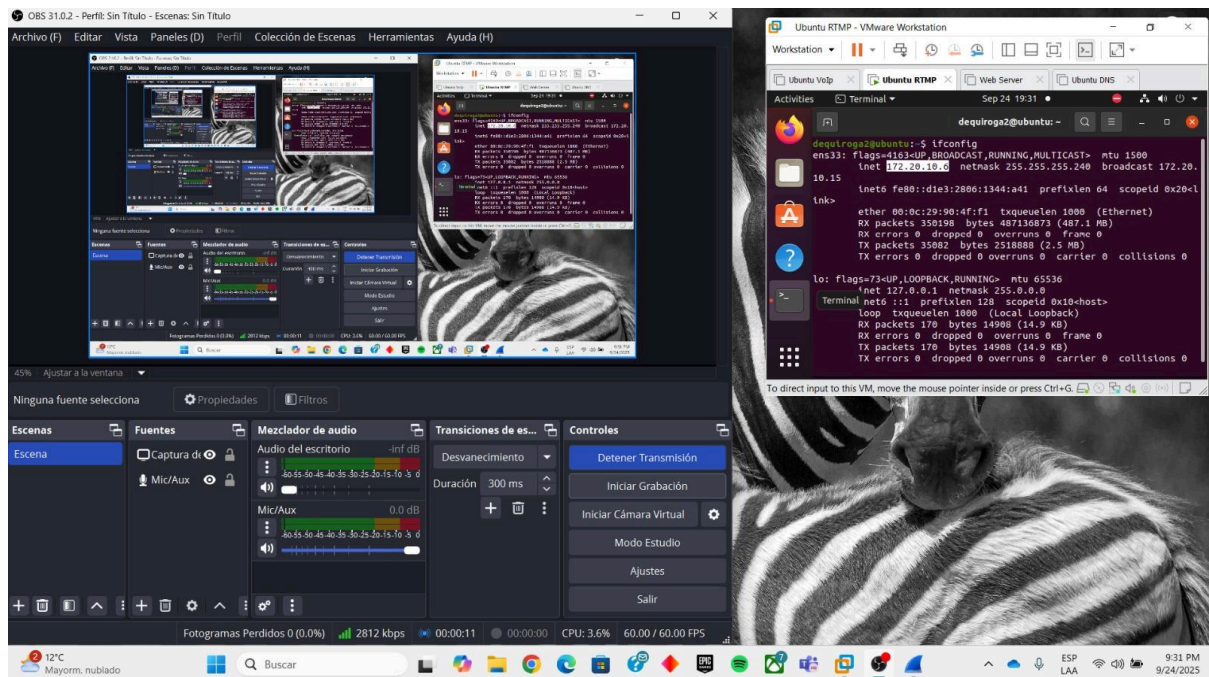


Imagen 28: Streaming con el servidor RTMP.

- En la capa de aplicación del tráfico generado por RTMP mostrado en la imagen 27 y 28, se puede observar un Handshake propio de este protocolo con C0, C1, C2 (pertenecientes al cliente), S0, S1, S2 (pertenecientes al servidor) para asegurar que el cliente y el servidor hablen el mismo idioma (versiones, timestamps, random data) antes de pasar chunks multimedia. Posteriormente se pasan los datos por medio del mismo protocolo y se manda audio y video por separado

50 1.322526	172.20.10.3	172.20.10.6	RTMP	131 Handshake C0+C1
62 1.323654	172.20.10.3	172.20.10.6	RTMP	130 Handshake C2
69 1.324123	172.20.10.6	172.20.10.3	RTMP	130 Handshake S0+S1+S2
75 1.324435	172.20.10.3	172.20.10.6	RTMP	247 Set Chunk Size 4096[connect('live')]
77 1.324932	172.20.10.6	172.20.10.3	RTMP	78 Window Acknowledgment Size 5000000
81 1.379259	172.20.10.6	172.20.10.3	RTMP	289 Set peer bandwidth 5000000,DynamicSet Chunk Size 4096[_result('NetConnection.Connect.Success')]
83 1.379408	172.20.10.3	172.20.10.6	RTMP	95 releaseStream('1234')
87 1.428183	172.20.10.3	172.20.10.6	RTMP	124 FCPublish('1234')[createStream()]
91 1.421281	172.20.10.6	172.20.10.3	RTMP	95 _result()
93 1.421281	172.20.10.3	172.20.10.6	RTMP	100 Publish('1234')
95 1.422093	172.20.10.6	172.20.10.3	RTMP	171 onStatus('NetStream.Publish.Start')
97 1.424314	172.20.10.3	172.20.10.6	RTMP	469 @setDataFrame()
101 1.703081	172.20.10.6	172.20.10.3	RTMP	73 Audio Data
103 1.703366	172.20.10.3	172.20.10.6	RTMP	1514 Video Data
497 1.715111	172.20.10.3	172.20.10.6	RTMP	1126 Video Data
509 1.717976	172.20.10.3	172.20.10.6	RTMP	703 Video Data
515 1.724462	172.20.10.6	172.20.10.3	RTMP	971 Video Data
519 1.751415	172.20.10.3	172.20.10.6	RTMP	241 Audio Data
523 1.762168	172.20.10.3	172.20.10.6	RTMP	916 Video Data
527 1.767382	172.20.10.3	172.20.10.6	RTMP	292 Audio Data
531 1.783536	172.20.10.3	172.20.10.6	RTMP	911 Video Data
535 1.784470	172.20.10.3	172.20.10.6	RTMP	251 Audio Data
539 1.801307	172.20.10.3	172.20.10.6	RTMP	906 Video Data
543 1.804874	172.20.10.3	172.20.10.6	RTMP	221 Audio Data
547 1.811766	172.20.10.3	172.20.10.6	RTMP	894 Video Data
551 1.825529	172.20.10.3	172.20.10.6	RTMP	754 Video Data
555 1.834273	172.20.10.3	172.20.10.6	RTMP	220 Audio Data
559 1.846400	172.20.10.3	172.20.10.6	RTMP	932 Video Data
563 1.851387	172.20.10.3	172.20.10.6	RTMP	230 Audio Data
567 1.867157	172.20.10.3	172.20.10.6	RTMP	911 Video Data
571 1.868423	172.20.10.3	172.20.10.6	RTMP	259 Audio Data
575 1.884379	172.20.10.3	172.20.10.6	RTMP	935 Video Data
579 1.889535	172.20.10.3	172.20.10.6	RTMP	296 Audio Data
583 1.902052	172.20.10.3	172.20.10.6	RTMP	947 Video Data
587 1.910215	172.20.10.3	172.20.10.6	RTMP	933 Video Data
591 1.917804	172.20.10.3	172.20.10.6	RTMP	322 Audio Data
595 1.932769	172.20.10.3	172.20.10.6	RTMP	979 Video Data
599 1.934110	172.20.10.3	172.20.10.6	RTMP	300 Audio Data
603 1.952956	172.20.10.3	172.20.10.6	RTMP	953 Video Data
607 1.954209	172.20.10.3	172.20.10.6	RTMP	280 Audio Data
611 1.968178	172.20.10.3	172.20.10.6	RTMP	932 Video Data
615 1.975394	172.20.10.3	172.20.10.6	RTMP	1282 Video Data
619 1.984648	172.20.10.3	172.20.10.6	RTMP	270 Audio Data
623 1.995559	172.20.10.3	172.20.10.6	RTMP	949 Video Data
627 2.002644	172.20.10.3	172.20.10.6	RTMP	240 Audio Data
631 2.017264	172.20.10.3	172.20.10.6	RTMP	949 Video Data
635 2.018623	172.20.10.3	172.20.10.6	RTMP	228 Audio Data
639 2.035990	172.20.10.3	172.20.10.6	RTMP	300 Video Data
643 2.039170	172.20.10.3	172.20.10.6	RTMP	225 Audio Data
647 2.052405	172.20.10.3	172.20.10.6	RTMP	951 Video Data
651 2.060310	172.20.10.3	172.20.10.6	RTMP	940 Video Data
655 2.068236	172.20.10.3	172.20.10.6	RTMP	222 Audio Data
659 2.081714	172.20.10.3	172.20.10.6	RTMP	346 Video Data
663 2.084279	172.20.10.3	172.20.10.6	RTMP	238 Audio Data
667 2.101436	172.20.10.3	172.20.10.6	RTMP	933 Video Data
671 2.102322	172.20.10.3	172.20.10.6	RTMP	292 Audio Data

Imagen 29: Parte del tráfico capturado en RTMP_view.pcap con el filtro del protocolo rtmp.

En la capa de transporte, el protocolo RTMP (Real Time Messaging Protocol) se implementa sobre TCP, utilizando por defecto el puerto 1935, como se observa en la Imagen 29. La elección de TCP se debe a que, aunque introduce cierta latencia por los mecanismos de control de congestión y retransmisión, garantiza una entrega ordenada y confiable de los datos, lo cual es esencial para mantener la estabilidad en la reproducción del video. De este modo, se prioriza la calidad y la ausencia de errores por encima de la inmediatez absoluta, algo especialmente relevante en transmisiones en vivo o streaming continuo donde la pérdida de paquetes degradará de manera notable la experiencia del usuario.

No.	Time	Source	Destination	Protocol	Length	Info
43	1.322125	172.20.10.3	172.20.10.6	TCP	6	62295 → 1935 [SYN] Seq=0 Win=0 Len=0 MSS=1460 WS=256 SACK_PERM
44	1.322130	172.20.10.3	172.20.10.6	TCP	66	[TCP Retransmission] 62295 → 1935 [SYN] Seq=0 Win=0 Len=0 MSS=1460 WS=256 SACK_PERM
45	1.322181	172.20.10.6	172.20.10.3	TCP	66	1935 → 62295 [SYN, ACK] Seq=0 Ack=1 Win=0 Len=0 MSS=1460 SACK_PERM WS=128
46	1.322186	172.20.10.6	172.20.10.3	TCP	6	[TCP Retransmission] 1935 → 62295 [ACK] Seq=0 Ack=1 Win=0 Len=0 MSS=1460 SACK_PERM WS=128
47	1.322444	172.20.10.3	172.20.10.6	TCP	54	62295 → 1935 [ACK] Seq=1 Ack=1 Win=0 Len=0
48	1.322445	172.20.10.3	172.20.10.6	TCP	54	[TCP Dup ACK 47x] 62295 → 1935 [ACK] Seq=1 Ack=1 Win=0 Len=0
49	1.322526	172.20.10.3	172.20.10.6	TCP	1514	62295 → 1935 [ACK] Seq=1 Ack=1 Win=0 Len=1460
50	1.322526	172.20.10.3	172.20.10.6	RTMP	131	Handshake C0=C1
51	1.322530	172.20.10.3	172.20.10.6	TCP	1514	[TCP Retransmission] 62295 → 1935 [ACK] Seq=1 Ack=1 Win=0 Len=1460
52	1.322563	172.20.10.3	172.20.10.6	TCP	131	[TCP Retransmission] 62295 → 1935 [PSH, ACK] Seq=1 Ack=1 Win=0 Len=77
53	1.322884	172.20.10.6	172.20.10.3	TCP	60	1935 → 62295 [ACK] Seq=1 Ack=1538 Win=0 Len=0
54	1.322884	172.20.10.6	172.20.10.3	TCP	60	[TCP Dup ACK 53x] 1935 → 62295 [ACK] Seq=1 Ack=1538 Win=0 Len=0
55	1.323352	172.20.10.6	172.20.10.3	TCP	1514	1935 → 62295 [ACK] Seq=1 Ack=1538 Win=0 Len=1460
56	1.323375	172.20.10.6	172.20.10.3	TCP	1514	[TCP Retransmission] 1935 → 62295 [ACK] Seq=1 Ack=1538 Win=0 Len=1460
57	1.323605	172.20.10.6	172.20.10.2	TCP	121	1935 → 62295 [PSH, ACK] Seq=1 Ack=1538 Win=0 Len=77
58	1.323607	172.20.10.6	172.20.10.3	TCP	131	[TCP Retransmission] 1935 → 62295 [PSH, ACK] Seq=1 Ack=1538 Win=0 Len=77
59	1.323629	172.20.10.3	172.20.10.6	TCP	54	62295 → 1935 [ACK] Seq=1538 Ack=1538 Win=0 Len=0
60	1.323631	172.20.10.3	172.20.10.6	TCP	54	[TCP Dup ACK 59x] 62295 → 1935 [ACK] Seq=1538 Ack=1538 Win=0 Len=0
61	1.323654	172.20.10.3	172.20.10.6	TCP	1514	62295 → 1935 [ACK] Seq=1538 Ack=1538 Win=0 Len=1460
62	1.323654	172.20.10.3	172.20.10.6	RTMP	130	Handshake C2
63	1.323667	172.20.10.3	172.20.10.6	TCP	1514	[TCP Retransmission] 62295 → 1935 [ACK] Seq=1538 Ack=1538 Win=0 Len=1460
64	1.323686	172.20.10.3	172.20.10.6	TCP	130	[TCP Retransmission] 62295 → 1935 [PSH, ACK] Seq=1538 Ack=1538 Win=0 Len=76
65	1.324037	172.20.10.6	172.20.10.3	TCP	1514	1935 → 62295 [ACK] Seq=1538 Ack=1538 Win=0 Len=1460
66	1.324040	172.20.10.6	172.20.10.3	TCP	105	[TCP Retransmission] 1935 → 62295 [ACK] Seq=1538 Ack=1538 Win=0 Len=1460
67	1.324076	172.20.10.3	172.20.10.6	TCP	54	62295 → 1935 [ACK] Seq=1538 Ack=1538 Win=0 Len=0
68	1.324081	172.20.10.3	172.20.10.6	TCP	54	[TCP Dup ACK 67x] 62295 → 1935 [ACK] Seq=1538 Ack=1538 Win=0 Len=0
69	1.324123	172.20.10.6	172.20.10.3	RTMP	130	Handshake Seq=1532

Imagen 30: Tráfico filtrado por el puerto 1935 en el archivo RTMP_view.pcap

En cuanto a los puertos utilizados, el 1935/TCP es el puerto estándar y más conocido para RTMP, pero en la práctica este protocolo puede también encapsularse en HTTP (puerto 80) o incluso en HTTPS (puerto 443), con el fin de atravesar firewalls y proxies que suelen bloquear el 1935 por motivos de seguridad. Esto le da flexibilidad a RTMP para adaptarse a distintos entornos de red, asegurando la disponibilidad del servicio de streaming aun cuando el puerto estándar no se encuentre habilitado, dando razón a que en el tráfico capturado existe RTMPT con la última T de Tunneling.

8. Conclusión

El análisis del laboratorio permitió comprender cómo los protocolos de la capa de aplicación (DNS, FTP, HTTP/HTTPS, VoIP y RTMP) dependen de la capa de transporte (TCP o UDP) para garantizar conectividad, control y transmisión de datos. Se evidenció que aplicaciones críticas como FTP, HTTP y SIP emplean TCP para fiabilidad y control, mientras que servicios sensibles a la latencia como VoIP se apoyan en UDP para priorizar inmediatez. Asimismo, RTMP recurre a TCP en el puerto 1935, priorizando estabilidad en la transmisión multimedia aun a costa de mayor latencia, aunque también puede encapsularse en HTTP/80 o HTTPS/443 para atravesar firewalls y proxies.

En cuanto a puertos y seguridad, se confirmó que el servicio web utiliza TCP:80 para HTTP y TCP:443 para HTTPS, este último con cifrado TLS que protege la confidencialidad de los datos. FTP usa TCP:21 para control y TCP:20 para datos, pero al transmitir en texto plano resulta inseguro frente a sniffing, razón por la cual su reemplazo más seguro, SFTP, opera sobre TCP:22 en un único canal cifrado y SSH, o FTPS con SSL/TLS. DNS funciona sobre UDP:53, priorizando velocidad y simplicidad. En VoIP, el protocolo SIP emplea UDP:5060 cuando no está cifrado o TCP:5061 en su variante segura SIPS, mientras que RTP utiliza puertos UDP dinámicos negociados con SDP. Finalmente, RTMP opera en TCP:1935. En conjunto, estos hallazgos muestran cómo la correcta elección de protocolo, puerto y mecanismo de seguridad es esencial para equilibrar rendimiento, confiabilidad y protección de la información en entornos de red modernos.