DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

LABORATORIO #1: INTRODUCCIÓN A REDES DE DATOS

ISIS3204 – INFRAESTRUCTURA DE COMUNICACIONES

PROFESOR

CARLOS LOZANO

GRUPO 6

MARÍA LUCÍA BENAVIDES DOMÍNGUEZ – 202313423

DANIEL CAMILO QUIMBAY VELÁSQUEZ - 202313861

NIKOL RODRIGUEZ ORTIZ – 202317538

2025-20

TABLA DE CONTENIDOS

Laboratorio 2 – Análisis de Protocolos con Wireshark

1. Objetivo

Describir brevemente que el objetivo es analizar el tráfico de red para diferentes protocolos de capa de aplicación (DNS, HTTP/HTTPS, FTP, VoIP, RTMP) usando Wireshark, tal como exige el laboratorio.

2. Capturas y Análisis Wireshark

2.1 Prueba de Conectividad (Ping)

2.1.1 Prueba para Ping_DNS_IP

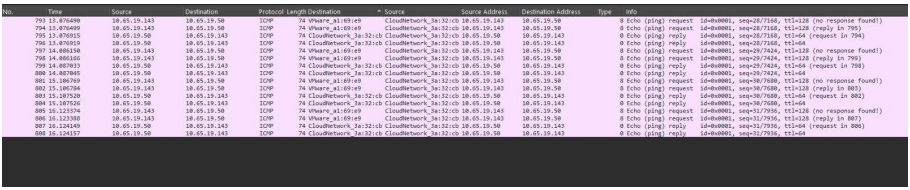


Figura 2.1.1.1 Wireshark llamadas DNS filtrando por ICMP con source, destination e info

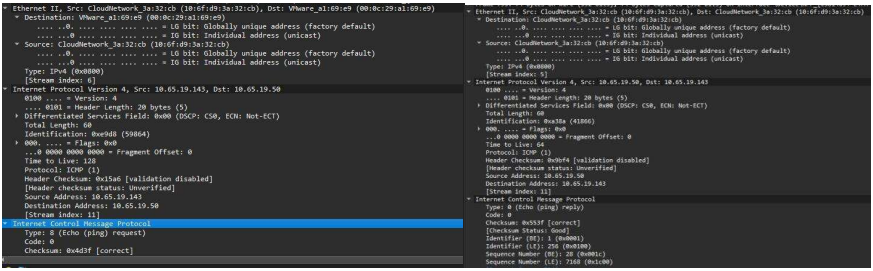
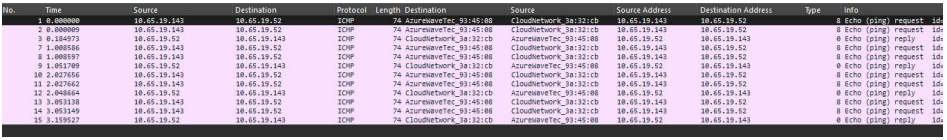


Figura 2.1.1.2 Imágenes de información de mensajes de request y response

En las imagenes se puede apreciar de manera clara la forma de las peticiones cuando se inspeccionan a profundidad y con la información de estas, llegamos a la siguiente tabla

Campo	**Request**	**Reply**
IP origen	10.65.19.143	10.65.19.50
IP destino	10.65.19.50	10.65.19.143
MAC cliente	10:6f:d9:3a:32:cb	10:6f:d9:3a:32:cb
MAC servidor	00:0c:29:a1:69:e9	10:6f:d9:3a:32:cb

2.1.2 Prueba para Ping_FTP_IP



No.	Time	Source	Destination	Protocol	Length	Destination	Source	Source Address	Destination Address	Type	Info
1	0.000000	10.65.19.143	10.65.19.52	ICMP	74	AzureWaveTec_93:45:08	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.52		8 Echo (ping) request id=
2	0.000809	10.65.19.52	10.65.19.143	ICMP	74	CloudNetwork_3a:32:cb	AzureWaveTec_93:45:08	10.65.19.52	10.65.19.143		0 Echo (ping) reply id=
3	0.184973	10.65.19.143	10.65.19.52	ICMP	74	AzureWaveTec_93:45:08	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.52		8 Echo (ping) request id=
7	1.000586	10.65.19.52	10.65.19.143	ICMP	74	CloudNetwork_3a:32:cb	AzureWaveTec_93:45:08	10.65.19.52	10.65.19.143		0 Echo (ping) reply id=
8	1.000597	10.65.19.143	10.65.19.52	ICMP	74	AzureWaveTec_93:45:08	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.52		8 Echo (ping) request id=
9	1.051709	10.65.19.52	10.65.19.143	ICMP	74	CloudNetwork_3a:32:cb	AzureWaveTec_93:45:08	10.65.19.52	10.65.19.143		0 Echo (ping) reply id=
10	2.027656	10.65.19.143	10.65.19.52	ICMP	74	AzureWaveTec_93:45:08	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.52		8 Echo (ping) request id=
11	2.027662	10.65.19.143	10.65.19.52	ICMP	74	AzureWaveTec_93:45:08	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.52		8 Echo (ping) request id=
12	2.048664	10.65.19.52	10.65.19.143	ICMP	74	CloudNetwork_3a:32:cb	AzureWaveTec_93:45:08	10.65.19.52	10.65.19.143		0 Echo (ping) reply id=
13	3.053138	10.65.19.143	10.65.19.52	ICMP	74	AzureWaveTec_93:45:08	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.52		8 Echo (ping) request id=
14	3.053149	10.65.19.143	10.65.19.52	ICMP	74	AzureWaveTec_93:45:08	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.52		8 Echo (ping) request id=
15	3.159527	10.65.19.52	10.65.19.143	ICMP	74	CloudNetwork_3a:32:cb	AzureWaveTec_93:45:08	10.65.19.52	10.65.19.143		0 Echo (ping) reply id=

Figura 2.1.2.1 Wireshark llamadas FTP filtrando por ICMP con source, destination e info

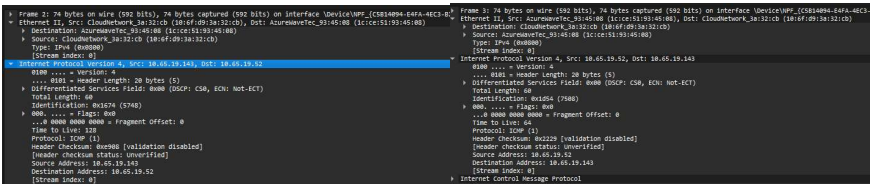


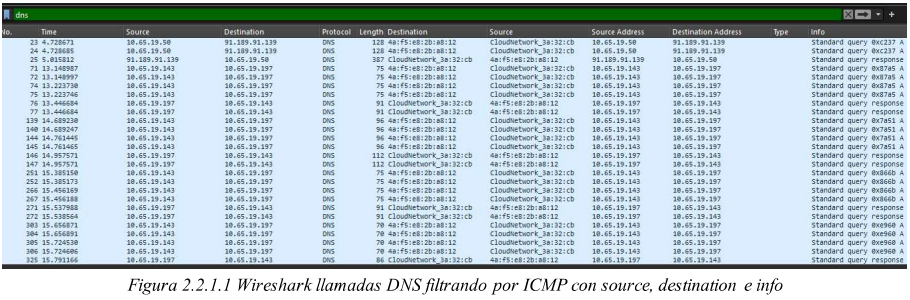
Figura 2.1.1.2 Imágenes de información de mensajes de request y response

En las imagenes se puede apreciar de manera clara la forma de las peticiones cuando se inspeccionan a profundidad y con la información de estas, llegamos a la siguiente tabla

Campo	**Request**	**Reply**
IP origen	10.65.19.143	10.65.19.52
IP destino	10.65.19.52	10.65.19.143
MAC cliente	10:6f:d9:3a:32:cb	1c:ce:51:93:45:08
MAC servidor	1c:ce:51:93:45:08	10:6f:d9:3a:32:cb

2.2 Servicio DNS

2.2.1 Archivo Ping_WEB_IP



No.	Time	Source	Destination	Protocol	Length	Destination	Source	Source Address	Destination Address	Type	Info
23	4.728671	10.65.19.50	91.189.91.139	DNS	128	4a:f5:e8:2b:a8:12	CloudNetwork_3a:32:cb	10.65.19.50	91.189.91.139		Standard query 0xc237 A
24	4.728685	10.65.19.50	91.189.91.139	DNS	128	4a:f5:e8:2b:a8:12	CloudNetwork_3a:32:cb	10.65.19.50	91.189.91.139		Standard query 0xc237 A
25	5.015812	91.189.91.139	10.65.19.50	DNS	387	CloudNetwork_3a:32:cb	4a:f5:e8:2b:a8:12	91.189.91.139	10.65.19.50		Standard query response
71	13.148987	10.65.19.143	10.65.19.197	DNS	75	4a:f5:e8:2b:a8:12	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.197		Standard query 0x87a5 A
72	13.148997	10.65.19.143	10.65.19.197	DNS	75	4a:f5:e8:2b:a8:12	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.197		Standard query 0x87a5 A
74	13.223730	10.65.19.143	10.65.19.197	DNS	75	4a:f5:e8:2b:a8:12	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.197		Standard query 0x87a5 A
75	13.223746	10.65.19.143	10.65.19.197	DNS	75	4a:f5:e8:2b:a8:12	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.197		Standard query 0x87a5 A
76	13.446684	10.65.19.197	10.65.19.143	DNS	91	CloudNetwork_3a:32:cb	4a:f5:e8:2b:a8:12	10.65.19.197	10.65.19.143		Standard query response
77	13.446684	10.65.19.197	10.65.19.143	DNS	91	CloudNetwork_3a:32:cb	4a:f5:e8:2b:a8:12	10.65.19.197	10.65.19.143		Standard query response
139	14.689230	10.65.19.143	10.65.19.197	DNS	96	4a:f5:e8:2b:a8:12	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.197		Standard query 0x7a51 A
140	14.689247	10.65.19.143	10.65.19.197	DNS	96	4a:f5:e8:2b:a8:12	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.197		Standard query 0x7a51 A
144	14.761445	10.65.19.143	10.65.19.197	DNS	96	4a:f5:e8:2b:a8:12	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.197		Standard query 0x7a51 A
145	14.761465	10.65.19.143	10.65.19.197	DNS	96	4a:f5:e8:2b:a8:12	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.197		Standard query 0x7a51 A
146	14.957571	10.65.19.197	10.65.19.143	DNS	112	CloudNetwork_3a:32:cb	4a:f5:e8:2b:a8:12	10.65.19.197	10.65.19.143		Standard query response
147	14.957571	10.65.19.197	10.65.19.143	DNS	112	CloudNetwork_3a:32:cb	4a:f5:e8:2b:a8:12	10.65.19.197	10.65.19.143		Standard query response
251	15.385150	10.65.19.143	10.65.19.197	DNS	75	4a:f5:e8:2b:a8:12	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.197		Standard query 0x866b A
252	15.385173	10.65.19.143	10.65.19.197	DNS	75	4a:f5:e8:2b:a8:12	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.197		Standard query 0x866b A
266	15.456169	10.65.19.143	10.65.19.197	DNS	75	4a:f5:e8:2b:a8:12	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.197		Standard query 0x866b A
267	15.456188	10.65.19.143	10.65.19.197	DNS	75	4a:f5:e8:2b:a8:12	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.197		Standard query 0x866b A
271	15.537988	10.65.19.197	10.65.19.143	DNS	91	CloudNetwork_3a:32:cb	4a:f5:e8:2b:a8:12	10.65.19.197	10.65.19.143		Standard query response
272	15.538564	10.65.19.197	10.65.19.143	DNS	91	CloudNetwork_3a:32:cb	4a:f5:e8:2b:a8:12	10.65.19.197	10.65.19.143		Standard query response
303	15.656871	10.65.19.143	10.65.19.197	DNS	70	4a:f5:e8:2b:a8:12	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.197		Standard query 0xe968 A
304	15.656891	10.65.19.143	10.65.19.197	DNS	70	4a:f5:e8:2b:a8:12	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.197		Standard query 0xe968 A
305	15.724530	10.65.19.143	10.65.19.197	DNS	70	4a:f5:e8:2b:a8:12	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.197		Standard query 0xe968 A
306	15.724606	10.65.19.143	10.65.19.197	DNS	70	4a:f5:e8:2b:a8:12	CloudNetwork_3a:32:cb	10.65.19.143	10.65.19.197		Standard query 0xe968 A
325	15.791166	10.65.19.197	10.65.19.143	DNS	86	CloudNetwork_3a:32:cb	4a:f5:e8:2b:a8:12	10.65.19.197	10.65.19.143		Standard query response

Figura 2.2.1.1 Wireshark llamadas DNS filtrando por ICMP con source, destination e info

Dentro de la imagen se puede ver de manera clara las request y los replies que se hicieron con el protocolo DNS con su información más importante. Estos son todos los archivos capturados con DNS.

La información de la capa de aplicación que se puede obtener de este apartado es:

1. Las peticiones

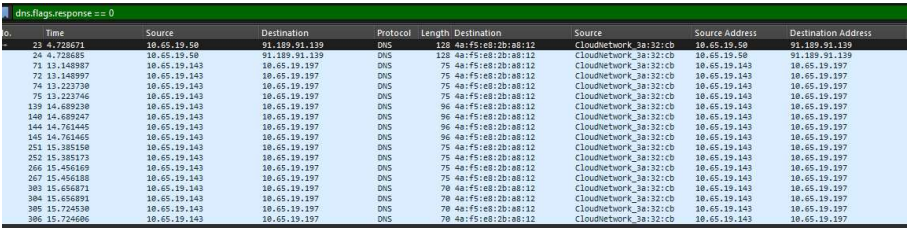


Figura 2.2.1.2 Wireshark filtrado para ver solo peticiones

Campo	**Request**	**Reply**
IP origen	10.65.19.143	10.65.19.50
IP destino	10.65.19.50	10.65.19.143
MAC cliente	10:6f:d9:3a:32:cb	10:6f:d9:3a:32:cb
MAC servidor	00:0c:29:a1:69:e9	10:6f:d9:3a:32:cb

2. Responses

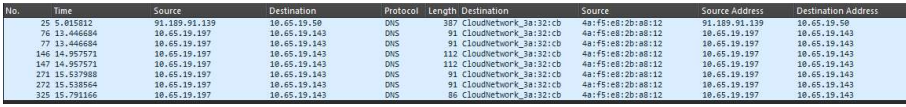


Figura 2.2.1.3 Wireshark filtrado para ver solo respuestas

Dentro de las peticiones y respuestas, se pudo obtener información como la siguiente:

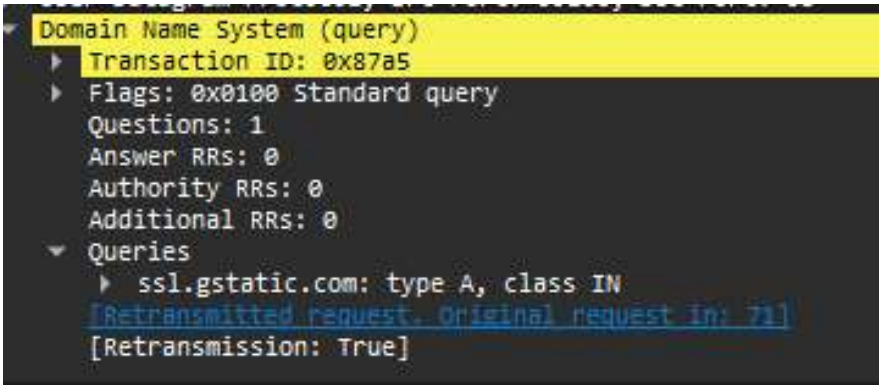


Figura 2.2.1.3 Información para capa de aplicación

Dentro de esta imagen, se puede observar la información de la capa de aplicación que aparece en los paquetes capturados que están relacionados con el servicio DNS. Tenemos un id para cada consulta/respuesta, el dominio solicitad (query name), el query type y Answers en donde podemos ver Tipo de registro, Dirección IP devuelta y TTL.

El protocolo de transporte identificado que se genera en las peticiones es UDP, ya que dns necesita ser rápido se usa este protocolo.

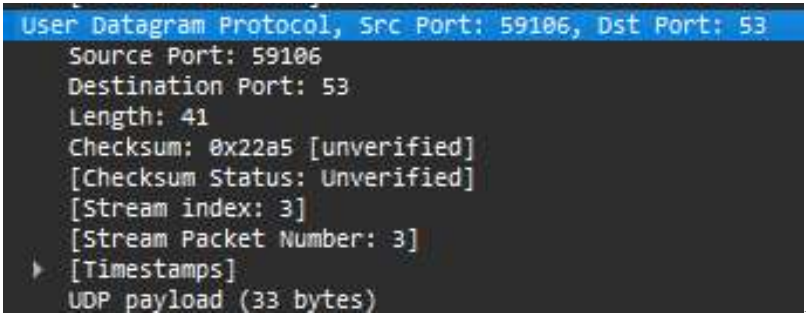


Figura 2.2.1.3 Información para capa de transporte y puerto

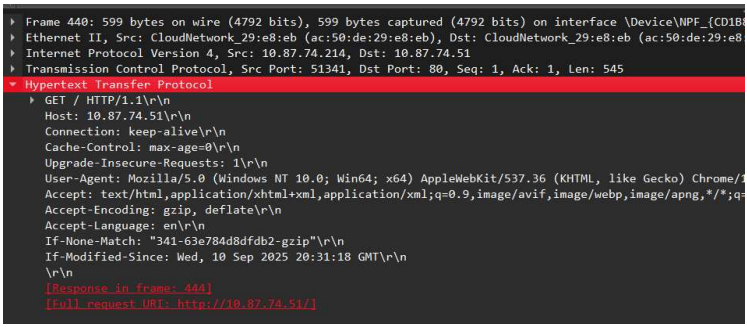
Así mismo en la imagen se puede ver que se hace uso del puerto 53, el cual siempre será el Destination Port. En la imagen para la petición escogida el puerto source fue el 59106, este se asigna como un numero aleatorio del cliente.

Campo	Request	Reply
IP origen	10.65.19.139	10.65.19.50
IP destino	10.65.19.50	10.65.19.139
MAC cliente	a:f5:e8:2b:a8:12	10:6f:d9:3a:32:cb
MAC servidor	00:0c:29:a1:69:e9	a:f5:e8:2b:a8:12

2.3 Servicio Web (HTTP/HTTPS)

2.3.1 HTTP

- Identifique la información de la capa de aplicación que aparecen en los paquetes capturados que estén relacionados con el servicio web:

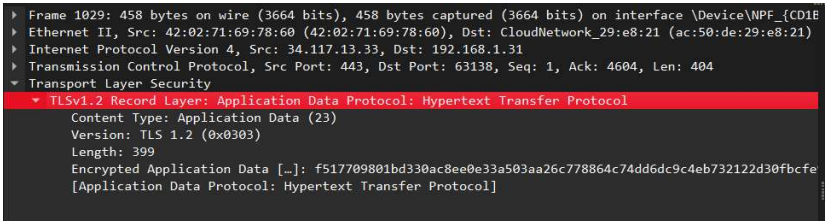


Para la información de la capa de aplicación es el mismo protocolo HTTP, vemos información como el destino, el tipo de conexión, información del navegador, etc.

- Identifique el protocolo de la capa de transporte generado por las peticiones al servidor web.
  En la misma captura de pantalla podemos observar que el protocolo de la capa de transporte es TCP
- Identifique los puertos utilizados por el servicio web
  En la misma linea que nos indica el protocolo tcp podemos ver que el servicio web usó el puerto 80. Además, el cliente usó el 51341

2.3.2 HTTPS

- Identifique la información de la capa de aplicación que aparecen en los paquetes capturados que estén relacionados con el servicio web:



Para la información de la capa vemos el TLS, ya que al ser HTTPS el contenido del HTTP está cifrado. Podemos ver es la información encriptada del http.

- Identifique el protocolo de la capa de transporte generado por las peticiones al servidor web.
  En la misma captura de pantalla podemos observar que el protocolo de la capa de transporte es TCP
- Identifique los puertos utilizados por el servicio web
  En la misma línea que nos indica el protocolo TCP podemos ver que el servicio web usó el puerto 443. Además, el cliente usó el 63168

- Datos principales, puertos 80/443, etc.

2.4 Servicio FTP

3.4.1 Análisis de tráfico FTP – Descarga de archivo

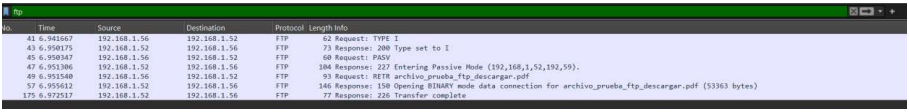


Ilustración 3.4.1 captura de pantalla del archivo FTP_download.pcap

Capa de aplicación:

En la captura se observa que el cliente cambia el modo de transferencia a binario mediante el comando TYPE I, lo cual es confirmado por el servidor con el mensaje "200 Type set to I". Luego, el cliente solicita el modo pasivo (PASV) y el servidor responde con "227 Entering Passive Mode (192,168,1,52,192,59)", indicando la IP y el puerto que se usarán para la conexión de datos. Posteriormente, el cliente envía el comando RETR archivo_prueba_ftp_descargar.pdf para descargar el archivo. El servidor confirma la apertura de la conexión con el mensaje "150 Opening BINARY mode data connection… (53363 bytes)" y, finalmente, cierra la operación con "226 Transfer complete", confirmando que la descarga fue exitosa.

Capa de transporte:

El protocolo de transporte utilizado es **TCP**, ya que FTP requiere una conexión confiable para el envío y recepción de archivos.

Puertos utilizados:

El puerto **21/TCP** se emplea como canal de control para los comandos y respuestas. Además, en la transferencia de datos se abre un puerto dinámico en modo pasivo. Según la respuesta del servidor, el puerto calculado es **49211/TCP**, que se usó para enviar el archivo al cliente.

3.4.2 Análisis de tráfico FTP – Subida de archivo

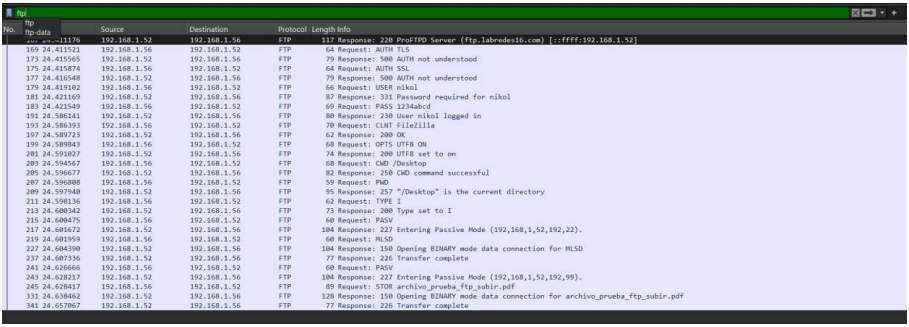


Ilustración 3.4.2 captura de pantalla del archivo FTP_upload.pcap

Capa de aplicación:

En este caso, el cliente inicia sesión con USER nikol y PASS 1234abcd, logrando autenticarse de manera exitosa con la respuesta "230 User nikol logged in". Luego, cambia al directorio de trabajo con CWD /Desktop, consulta la ubicación actual con PWD, y establece el modo binario con TYPE I. Posteriormente, el cliente solicita el modo pasivo (PASV), y el servidor responde con "227 Entering Passive Mode (192,168,1,52,192,22)". Con esta información, el cliente lista el directorio mediante MLSD, operación que es confirmada por el servidor con "150 Opening BINARY mode data connection for MLSD" y "226 Transfer complete". Más adelante, se vuelve a activar el modo pasivo (PASV), esta vez con la respuesta "227 Entering Passive Mode (192,168,1,52,192,99)". Con este nuevo canal de datos, el cliente inicia la subida del archivo con STOR archivo_prueba_ftp_subir.pdf. El servidor abre la conexión con "150 Opening BINARY mode data connection for archivo_prueba_ftp_subir.pdf" y confirma la finalización con "226 Transfer complete".

Capa de transporte:

El protocolo utilizado es **TCP**, garantizando una comunicación confiable entre cliente y servidor durante toda la transferencia.

Puertos utilizados:

El puerto **21/TCP** se emplea para el canal de control. Para los datos, el servidor asigna puertos dinámicos en modo pasivo: en este caso, el puerto **49214/TCP** se utilizó para el listado de archivos y el puerto **49283/TCP** para la transferencia del archivo cargado.

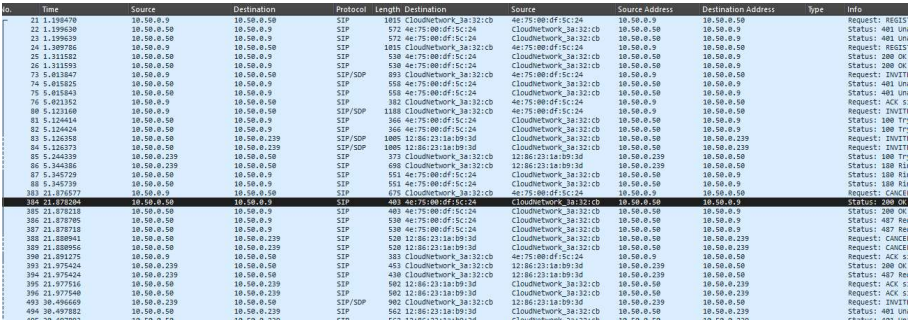
2.5 Servicio VoIP

Archivo .pcap: VoIP_view.pcap

Filtros usados: sip, rtp, udp.port==5060

Dentro de VoIp se hace uso de 2 protocolos, SIP y RTP. Dentro de estos protocolos en capa de aplicación podemos ver lo siguiente en sus paquetes capturados

SIP:



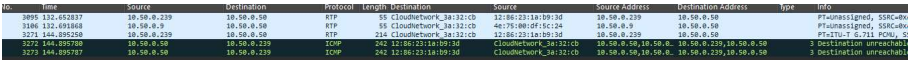
No.	Time	Source	Destination	Protocol	Length	Destination	Source	Source Address	Destination Address	Type	Info
21	1.198470	10.50.0.9	10.50.0.50	SIP	1015	CloudNetwork_3a:32:cb	4e:75:00:df:5c:24	10.50.0.9	10.50.0.50		Request: REGIS
22	1.199630	10.50.0.50	10.50.0.9	SIP	572	4e:75:00:df:5c:24	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.9		Status: 401 Un
23	1.199639	10.50.0.50	10.50.0.9	SIP	572	4e:75:00:df:5c:24	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.9		Status: 401 Un
24	1.309786	10.50.0.9	10.50.0.50	SIP	1015	CloudNetwork_3a:32:cb	4e:75:00:df:5c:24	10.50.0.9	10.50.0.50		Request: REGIS
25	1.311582	10.50.0.50	10.50.0.9	SIP	530	4e:75:00:df:5c:24	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.9		Status: 200 OK
26	1.311593	10.50.0.50	10.50.0.9	SIP	530	4e:75:00:df:5c:24	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.9		Status: 200 OK
73	5.013847	10.50.0.9	10.50.0.50	SIP/SDP	893	CloudNetwork_3a:32:cb	4e:75:00:df:5c:24	10.50.0.9	10.50.0.50		Request: INVIT
74	5.015825	10.50.0.50	10.50.0.9	SIP	558	4e:75:00:df:5c:24	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.9		Status: 401 Un
75	5.015843	10.50.0.50	10.50.0.9	SIP	558	4e:75:00:df:5c:24	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.9		Status: 401 Un
76	5.021352	10.50.0.9	10.50.0.50	SIP	382	CloudNetwork_3a:32:cb	4e:75:00:df:5c:24	10.50.0.9	10.50.0.50		Request: ACK s
80	5.123160	10.50.0.9	10.50.0.50	SIP/SDP	1188	CloudNetwork_3a:32:cb	4e:75:00:df:5c:24	10.50.0.9	10.50.0.50		Request: INVIT
81	5.124414	10.50.0.50	10.50.0.9	SIP	366	4e:75:00:df:5c:24	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.9		Status: 100 Tr
82	5.124424	10.50.0.50	10.50.0.9	SIP	366	4e:75:00:df:5c:24	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.9		Status: 100 Tr
83	5.126358	10.50.0.50	10.50.0.239	SIP/SDP	1005	12:86:23:1a:b9:3d	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.239		Request: INVIT
84	5.126373	10.50.0.50	10.50.0.239	SIP/SDP	1005	12:86:23:1a:b9:3d	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.239		Request: INVIT
85	5.244339	10.50.0.239	10.50.0.50	SIP	373	CloudNetwork_3a:32:cb	12:86:23:1a:b9:3d	10.50.0.239	10.50.0.50		Status: 100 Tr
86	5.344386	10.50.0.239	10.50.0.50	SIP	698	CloudNetwork_3a:32:cb	12:86:23:1a:b9:3d	10.50.0.239	10.50.0.50		Status: 180 Ri
87	5.345729	10.50.0.50	10.50.0.9	SIP	551	4e:75:00:df:5c:24	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.9		Status: 180 Ri
88	5.345739	10.50.0.50	10.50.0.9	SIP	551	4e:75:00:df:5c:24	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.9		Status: 180 Ri
383	21.876577	10.50.0.9	10.50.0.50	SIP	675	CloudNetwork_3a:32:cb	4e:75:00:df:5c:24	10.50.0.9	10.50.0.50		Request: CANCE
384	21.878204	10.50.0.50	10.50.0.9	SIP	403	4e:75:00:df:5c:24	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.9		Status: 200 OK
385	21.878218	10.50.0.50	10.50.0.9	SIP	403	4e:75:00:df:5c:24	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.9		Status: 200 OK
386	21.878705	10.50.0.50	10.50.0.9	SIP	530	4e:75:00:df:5c:24	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.9		Status: 487 Re
387	21.878718	10.50.0.50	10.50.0.9	SIP	530	4e:75:00:df:5c:24	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.9		Status: 487 Re
388	21.880941	10.50.0.50	10.50.0.239	SIP	520	12:86:23:1a:b9:3d	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.239		Request: CANCE
389	21.880956	10.50.0.50	10.50.0.239	SIP	520	12:86:23:1a:b9:3d	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.239		Request: CANCE
390	21.891275	10.50.0.9	10.50.0.50	SIP	383	CloudNetwork_3a:32:cb	4e:75:00:df:5c:24	10.50.0.9	10.50.0.50		Request: ACK s
393	21.975424	10.50.0.239	10.50.0.50	SIP	453	CloudNetwork_3a:32:cb	12:86:23:1a:b9:3d	10.50.0.239	10.50.0.50		Status: 200 OK
394	21.975424	10.50.0.239	10.50.0.50	SIP	430	CloudNetwork_3a:32:cb	12:86:23:1a:b9:3d	10.50.0.239	10.50.0.50		Status: 487 Re
395	21.977516	10.50.0.50	10.50.0.239	SIP	502	12:86:23:1a:b9:3d	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.239		Request: ACK s
396	21.977540	10.50.0.50	10.50.0.239	SIP	502	12:86:23:1a:b9:3d	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.239		Request: ACK s
493	30.496669	10.50.0.50	10.50.0.239	SIP/SDP	902	CloudNetwork_3a:32:cb	12:86:23:1a:b9:3d	10.50.0.239	10.50.0.50		Request: INVIT
494	30.497882	10.50.0.50	10.50.0.239	SIP	562	12:86:23:1a:b9:3d	CloudNetwork_3a:32:cb	10.50.0.50	10.50.0.239		Status: 401 Un

Figura 2.5.1 Imagen de filtro de SIP en wireshark

Figura 2.5.2 Imagen de capa de aplicación de paquetes con SIP

Dentro de esta imagen podemos ver el request line con OK, las cabeceras con un CALL-ID e información de los usuarios que se estaban conectando con el FROM y el TO. Se pueden ver tambien de manera clara la parte de VIA qeu tiene el puerto diferente información de envio.

RTP:



No.	Time	Source	Destination	Protocol	Length	Destination	Source	Source Address	Destination Address	Type	Info
3095	132.652837	10.50.0.239	10.50.0.50	RTP	55	CloudNetwork_3a:32:cb	12:86:23:1a:b9:3d	10.50.0.239	10.50.0.50		PT=unassigned, SSRC=0x
3106	132.691868	10.50.0.9	10.50.0.50	RTP	55	CloudNetwork_3a:32:cb	4e:75:00:df:5c:24	10.50.0.9	10.50.0.50		PT=unassigned, SSRC=0x
3271	144.895250	10.50.0.239	10.50.0.50	RTP	214	CloudNetwork_3a:32:cb	12:86:23:1a:b9:3d	10.50.0.239	10.50.0.50		PT=ITU-T G.711 PCMU, S
3272	144.895780	10.50.0.50	10.50.0.239	ICMP	242	12:86:23:1a:b9:3d	CloudNetwork_3a:32:cb	10.50.0.50,10.50.0.	10.50.0.239,10.50.0.50		3 Destination unreachabl
3273	144.895787	10.50.0.50	10.50.0.239	ICMP	242	12:86:23:1a:b9:3d	CloudNetwork_3a:32:cb	10.50.0.50,10.50.0.	10.50.0.239,10.50.0.50		3 Destination unreachabl

Figura 2.5.1 Imagen de filtro de RTP en wireshark

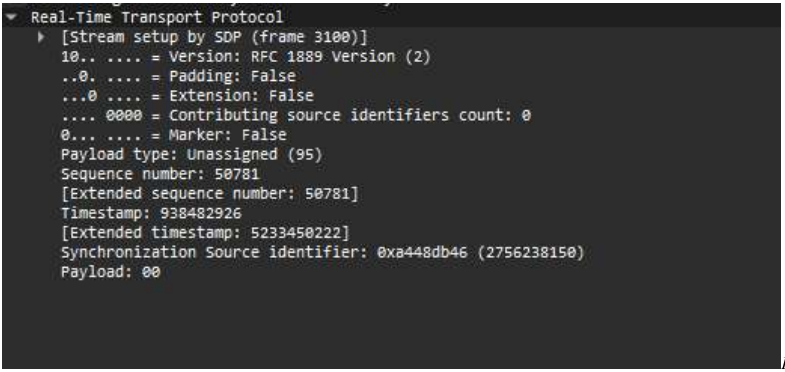


Figura 2.5.2

Imagen de capa de aplicación de paquetes con RTP

Dentro de esta imagen podemos ver información como el PayloadType que es unassigned, el sequence number, timestamp y tambien el SSRI que representa la información más importante de la capa de aplicación.

Por otro lado para la capa de transporte, podemos ver que dentro de la cabecera de una petición SIP y una petición RTP el protocolo es UDP ya que es un protocolo rapido de usar.

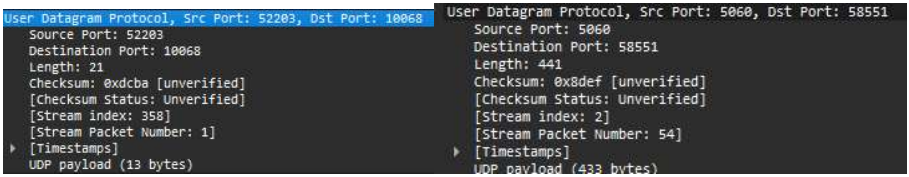


Figura 2.5.3 Imagen de capa de transporte en peticiones tanto RTP como SIP respectivamente

Dentro de las imagenes anteriores se puede ver que el puerto de SIP es 5060 y el puerto de RTP es 52203. El puerto SIP es definido, mientras que el puerto RTP esta en un rango dinamico dependiente del softphone.

2.6 Servicio RTMP

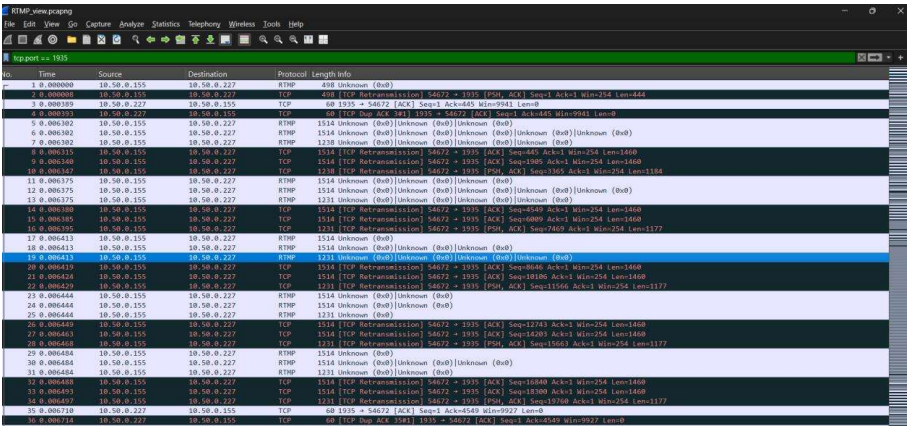


Ilustración 3.1 captura de pantalla del archivo RTMP_view.pcap

Capa de aplicación:

En la captura se observa tráfico identificado como **RTMP**. Sin embargo, no todo el tráfico aparece bajo ese filtro porque Wireshark no siempre reconoce automáticamente este protocolo. Por eso, en lugar de filtrar con `rtmp`, es más efectivo aplicar el filtro por **puerto**, en este caso `tcp.port == 1935`, que es el puerto estándar de RTMP. Una vez hecho esto, se puede distinguir el intercambio de paquetes relacionados con la transmisión en vivo.

En cuanto a la información que se aprecia, los paquetes contienen tramas propias de la transmisión de video en tiempo real, donde se observa una gran cantidad de mensajes “Unknown” o sin decodificar por parte de Wireshark. Esto ocurre porque RTMP encapsula datos multimedia (video y audio) que se envían de manera continua en bloques pequeños, por lo cual aparecen cientos o miles de paquetes durante la transmisión.

Capa de transporte:
El protocolo de transporte utilizado es **TCP**, ya que RTMP requiere un canal confiable para garantizar que los fragmentos de video y audio lleguen en orden y sin pérdidas al reproductor del cliente.

Puertos utilizados:
El puerto estándar utilizado para RTMP es el **1935/TCP**, que aparece claramente en la captura. Este es el canal por el cual se transporta la transmisión de video desde el cliente (OBS u otra aplicación emisora) hacia el servidor, y luego se redistribuye al receptor (por ejemplo, VLC).