

UNIVERSIDAD DEL VALLE DE GUATEMALA

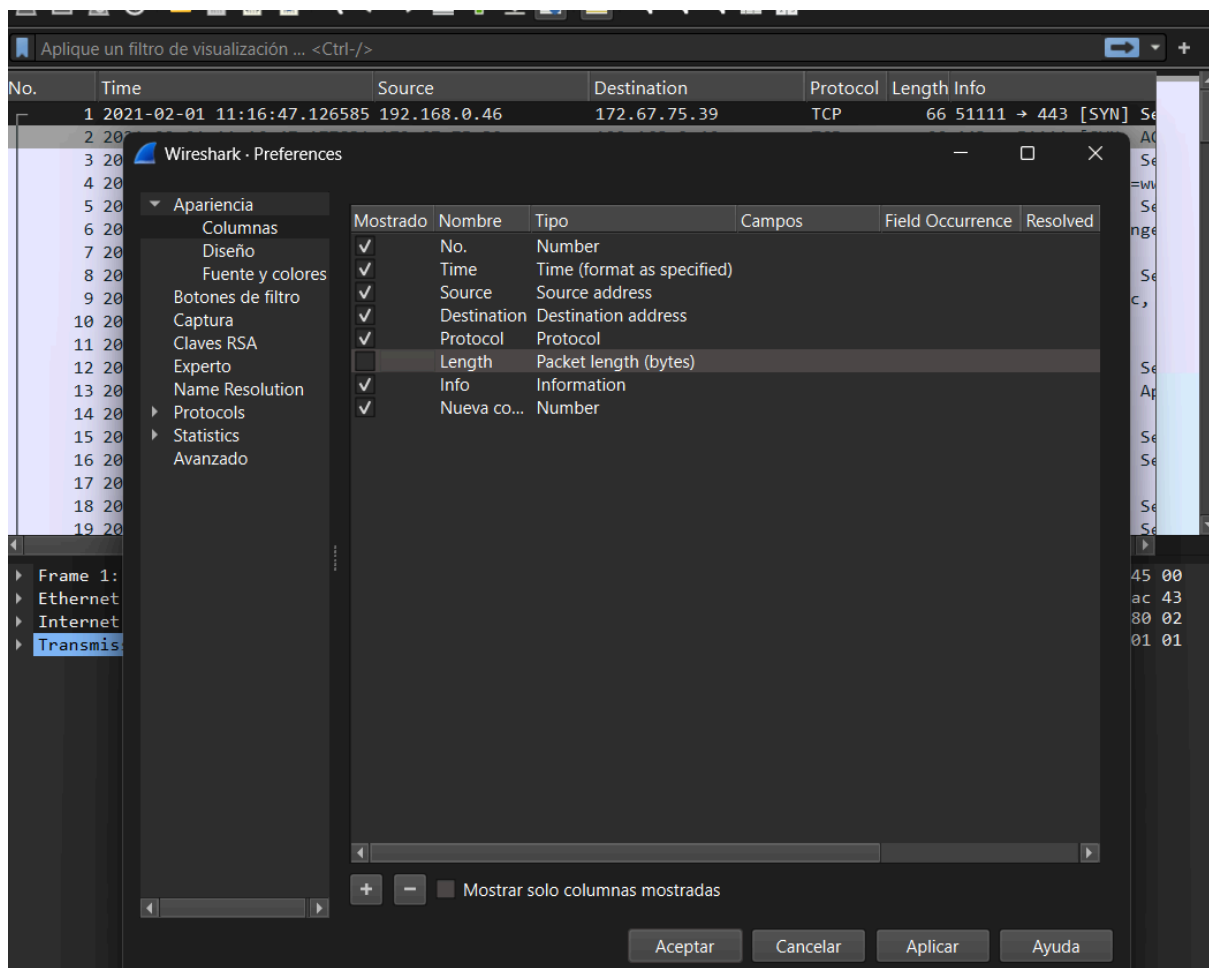
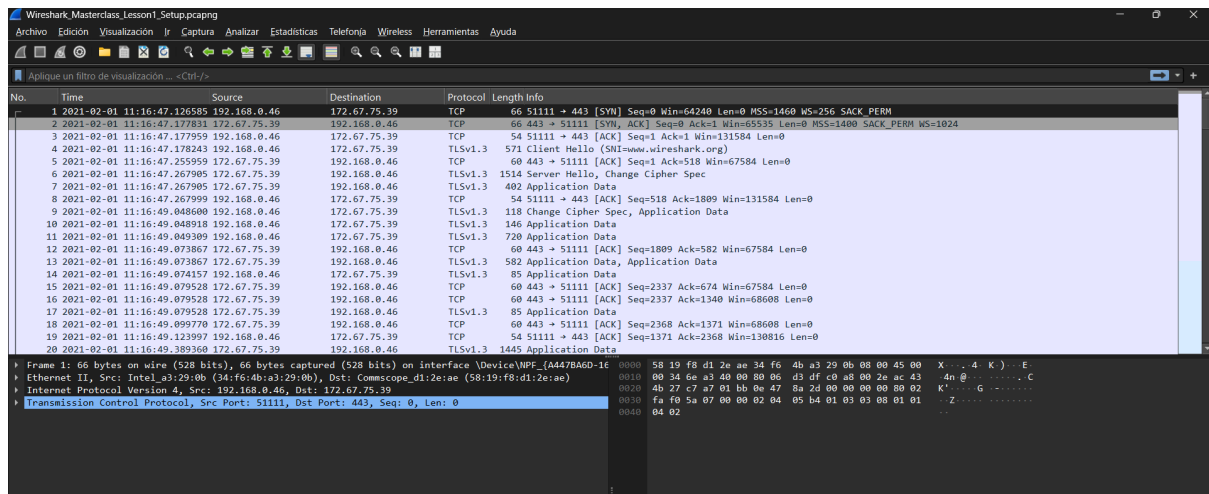


Laboratorio 1 - Reporte Individual

Sergio Orellana - 221122

Redes

1.1 Personalización de entorno



Aplique un filtro de visualización ... <Ctrl>F						
No.	Time	Source	Destination	Protocol	Info	Nueva columna
1	2021-02-01 11:16:47.126585	192.168.0.46	172.67.75.39	TCP	51111 → 443 [SYN] Seq=0 Win=...	
2	2021-02-01 11:16:47.177831	172.67.75.39	192.168.0.46	TCP	443 → 51111 [SYN, ACK] Seq=...	
3	2021-02-01 11:16:47.177959	192.168.0.46	172.67.75.39	TCP	51111 → 443 [ACK] Seq=1 Ack=...	
4	2021-02-01 11:16:47.178243	192.168.0.46	172.67.75.39	TLSv1.3	Client Hello (SN=sw.wires...	
5	2021-02-01 11:16:47.255959	172.67.75.39	192.168.0.46	TCP	443 → 51111 [ACK] Seq=1 Ack=...	
6	2021-02-01 11:16:47.267905	172.67.75.39	192.168.0.46	TLSv1.3	Server Hello, Change Cipher...	
7	2021-02-01 11:16:47.267905	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
8	2021-02-01 11:16:47.267999	192.168.0.46	172.67.75.39	TCP	51111 → 443 [ACK] Seq=518 A...	
9	2021-02-01 11:16:49.048600	192.168.0.46	172.67.75.39	TLSv1.3	Change Cipher Spec, Applica...	
10	2021-02-01 11:16:49.048918	192.168.0.46	172.67.75.39	TLSv1.3	Application Data	
11	2021-02-01 11:16:49.049309	192.168.0.46	172.67.75.39	TLSv1.3	Application Data	
12	2021-02-01 11:16:49.073867	172.67.75.39	192.168.0.46	TCP	443 → 51111 [ACK] Seq=1809 ...	
13	2021-02-01 11:16:49.073867	172.67.75.39	192.168.0.46	TLSv1.3	Application Data, Applicati...	
14	2021-02-01 11:16:49.074157	192.168.0.46	172.67.75.39	TLSv1.3	Application Data	
15	2021-02-01 11:16:49.079528	172.67.75.39	192.168.0.46	TCP	443 → 51111 [ACK] Seq=2337 ...	
16	2021-02-01 11:16:49.079528	172.67.75.39	192.168.0.46	TCP	443 → 51111 [ACK] Seq=2337 ...	
17	2021-02-01 11:16:49.079528	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
18	2021-02-01 11:16:49.099770	172.67.75.39	192.168.0.46	TCP	443 → 51111 [ACK] Seq=2368 ...	
19	2021-02-01 11:16:49.123997	192.168.0.46	172.67.75.39	TCP	51111 → 443 [ACK] Seq=1371 ...	
20	2021-02-01 11:16:49.389360	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
21	2021-02-01 11:16:49.389360	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
22	2021-02-01 11:16:49.389360	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
23	2021-02-01 11:16:49.389360	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
24	2021-02-01 11:16:49.389360	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
25	2021-02-01 11:16:49.389360	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
26	2021-02-01 11:16:49.389360	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
27	2021-02-01 11:16:49.389546	192.168.0.46	172.67.75.39	TCP	51111 → 443 [ACK] Seq=1371 ...	
28	2021-02-01 11:16:49.461370	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
29	2021-02-01 11:16:49.464847	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
30	2021-02-01 11:16:49.464847	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
31	2021-02-01 11:16:49.464847	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
32	2021-02-01 11:16:49.464847	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
33	2021-02-01 11:16:49.464847	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
34	2021-02-01 11:16:49.464847	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
35	2021-02-01 11:16:49.464847	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
36	2021-02-01 11:16:49.464847	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
37	2021-02-01 11:16:49.464847	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
38	2021-02-01 11:16:49.464847	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
39	2021-02-01 11:16:49.464847	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
40	2021-02-01 11:16:49.464847	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	
41	2021-02-01 11:16:49.464847	172.67.75.39	192.168.0.46	TLSv1.3	Application Data	

Wireshark_Masterclass_Lesson1_Setup.pcapng

Paquetes: 89

Perfil: Sergio Orellana

Wireshark · Reglas de coloreado Sergio Orellana

Nombre	Filtro
✓ Bad TCP	tcp.analysis.flags && !tcp.analysis.window_update && !tcp.analysis
✓ HSRP State Change	hsrp.state != 8 && hsrp.state != 16
✓ Spanning Tree Topology Change	stp.type == 0x80
✓ OSPF State Change	ospf.msg != 1
✓ ICMP errors	icmp.type in { 3..5, 11 } icmpv6.type in { 1..4 }
✓ ARP	arp
✓ ICMP	icmp icmpv6
✓ TCP RST	tcp.flags.reset eq 1
✓ SCTP ABORT	sctp.chunk_type eq ABORT
✓ IPv4 TTL low or unexpected	(ip.dst != 224.0.0.0/4 && ip.ttl < 5 && !(pim ospf eigrp bgp
✓ IPv6 hop limit low or unexpected	(ipv6.dst != ff02::/8 && ipv6.hlim < 5 && !(ospf bgp tcp.port
✓ Checksum Errors	eth.fcs.status=="Bad" ip.checksum.status=="Bad" tcp.checks
✓ SMB	smb nbss nbns netbios
✓ HTTP	http tcp.port == 80 http2
✓ DCERPC	dcerpc
✓ Routing	hsrp eigrp ospf bgp cdp vrrp carp gvrp igmp ism
✓ TCP SYN/FIN	tcp.flags & 0x02 tcp.flags.fin == 1
✓ TCP	tcp
✓ UDP	udp
✓ Broadcast	eth[0] & 1
✓ System Event	systemd_journal sysdig

Doble clic para editar. Arrastrar para mover. Las reglas son procesadas en orden hasta que una coincidencia es encontrada.

+

-

Primer plano

Fondo

Aplicar como filtro

C:\Users...rfilters

Aceptar

Copiar desde ▾

Cancelar

Importar...

Exportar...

Ayuda

Wireshark Masterclass Lesson1_Setup.pcapng

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

tcp.flags & 0x02 || tcp.flags.fin == 1

No.	Time	Source	Destination	Protocol	Info	Nueva columna
1	2021-02-01 11:16:47.126585	192.168.0.46	172.67.75.39	TCP	51111 → 443 [SYN] Seq=0 Win=...	
2	2021-02-01 11:16:47.177831	172.67.75.39	192.168.0.46	TCP	443 → 51111 [SYN, ACK] Seq=...	

Frame 2: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Ethernet II, Src: Commscope_d1:2e:ae (58:19:f8:d1:2e:ae), Dst: Intel_a3:29:29 (82:5f:ea:a3:29:29)
Internet Protocol Version 4, Src: 172.67.75.39, Dst: 192.168.0.46
Transmission Control Protocol, Src Port: 443, Dst Port: 51111, Seq: 0, Len: 0

0000 34 f6 4b a3 29 0b 58 19 f8 d1 2e ae 08 00 45 00 4 K) X ... E
0010 00 34 00 00 40 00 3b 06 87 83 ac 43 4b 27 c0 a8 4 @ ; ... CK'
0020 00 2e 01 bb c7 a7 c4 9f 88 86 0e 47 8a 2e 80 12 G..
0030 ff ff 87 fb 00 00 02 04 05 78 01 01 04 02 01 03 x.....
0040 03 0a

Wireshark Masterclass Lesson1_Setup.pcapng Paquetes: 89 - Displayed: 2 (2.2%) Perfil: Sergio Orellana

Wireshark Masterclass Lesson1_Setup.pcapng

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

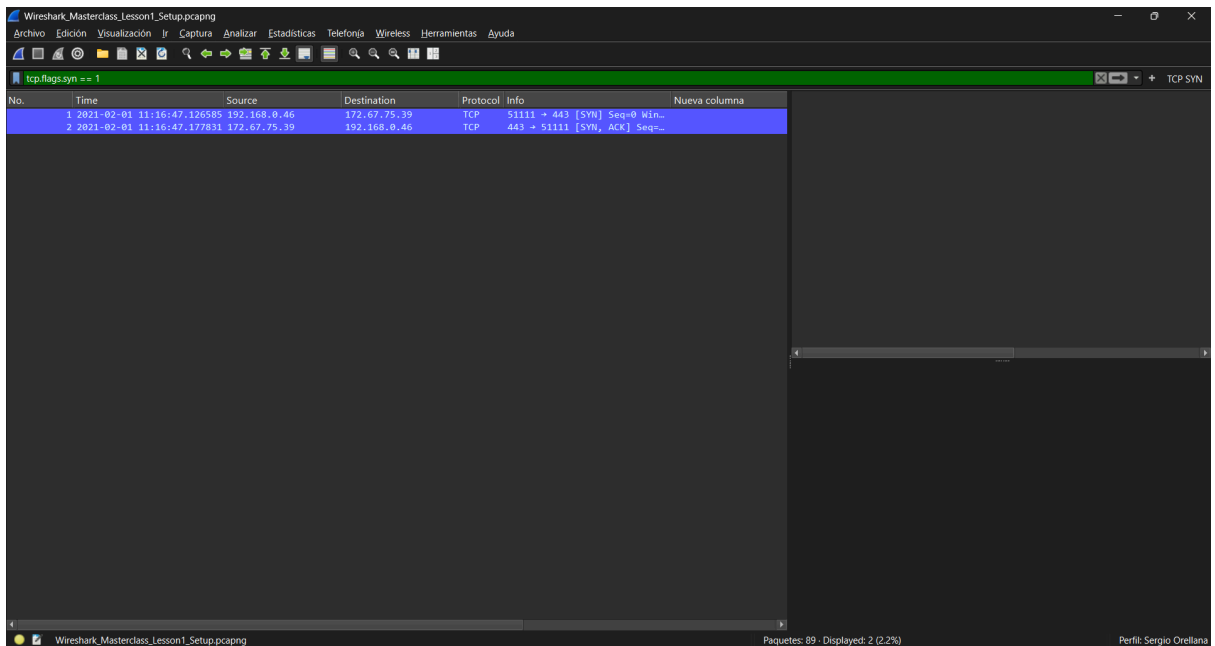
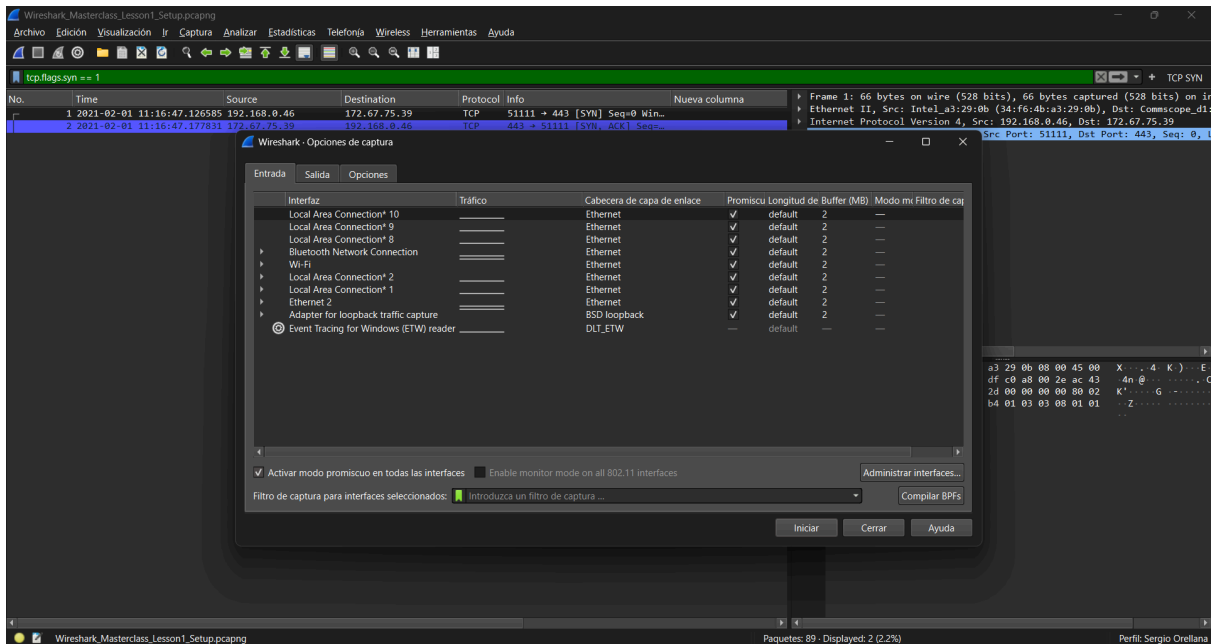
tcp.flags.syn == 1

No.	Time	Source	Destination	Protocol	Info	Nueva columna
1	2021-02-01 11:16:47.126585	192.168.0.46	172.67.75.39	TCP	51111 → 443 [SYN] Seq=0 Win=...	
2	2021-02-01 11:16:47.177831	172.67.75.39	192.168.0.46	TCP	443 → 51111 [SYN, ACK] Seq=...	

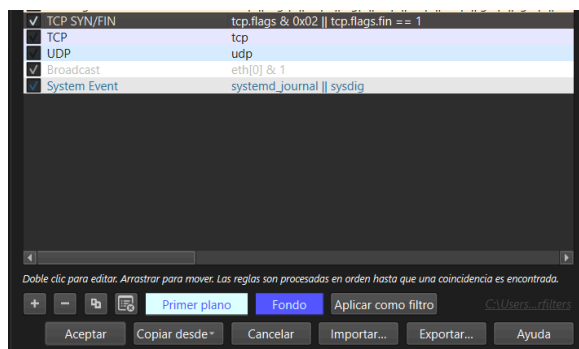
Frame 2: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Ethernet II, Src: Commscope_d1:2e:ae (58:19:f8:d1:2e:ae), Dst: Intel_a3:29:29 (82:5f:ea:a3:29:29)
Internet Protocol Version 4, Src: 172.67.75.39, Dst: 192.168.0.46
Transmission Control Protocol, Src Port: 443, Dst Port: 51111, Seq: 0, Len: 0

0000 34 f6 4b a3 29 0b 58 19 f8 d1 2e ae 08 00 45 00 4 K) X ... E
0010 00 34 00 00 40 00 3b 06 87 83 ac 43 4b 27 c0 a8 4 @ ; ... CK'
0020 00 2e 01 bb c7 a7 c4 9f 88 86 0e 47 8a 2e 80 12 G..
0030 ff ff 87 fb 00 00 02 04 05 78 01 01 04 02 01 03 x.....
0040 03 0a

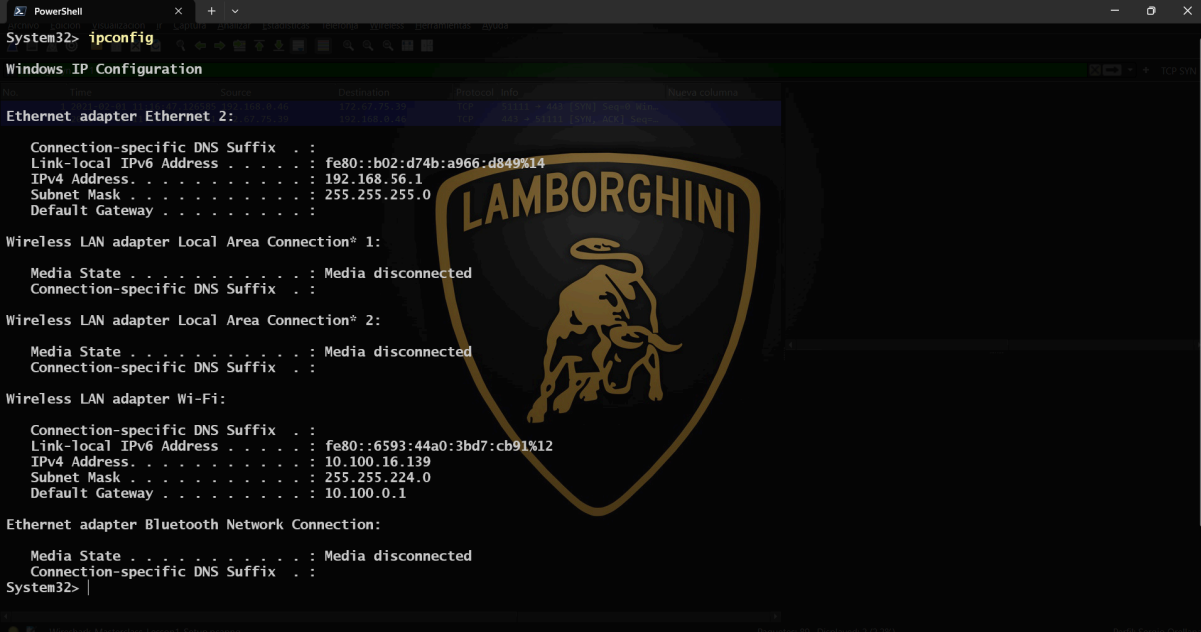
Wireshark Masterclass Lesson1_Setup.pcapng Paquetes: 89 - Displayed: 2 (2.2%) Perfil: Sergio Orellana



Le cambié el color de la fuente, por eso aparece color azul claro.



1.2 Configuración de la captura de paquetes



```
PowerShell
System32> ipconfig

Windows IP Configuration

Ethernet adapter Ethernet 2:

    Connection-specific DNS Suffix  . : 
    Link-Local IPv6 Address . . . . . : fe80::b02:d74b:a966:d849%14
    IPv4 Address. . . . . : 192.168.56.1
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 

Wireless LAN adapter Local Area Connection* 1:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

Wireless LAN adapter Local Area Connection* 2:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

Wireless LAN adapter Wi-Fi:

    Connection-specific DNS Suffix  . : 
    Link-Local IPv6 Address . . . . . : fe80::6593:44a0:3bd7:cb91%12
    IPv4 Address. . . . . : 10.100.16.139
    Subnet Mask . . . . . : 255.255.224.0
    Default Gateway . . . . . : 10.100.0.1

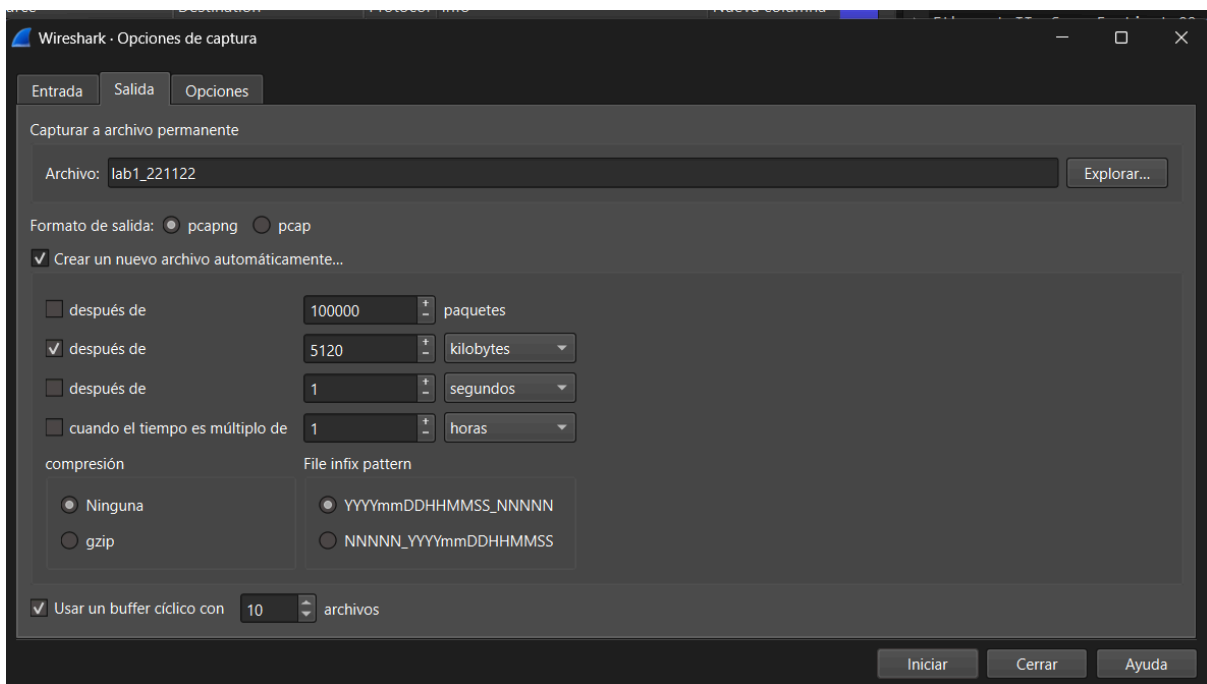
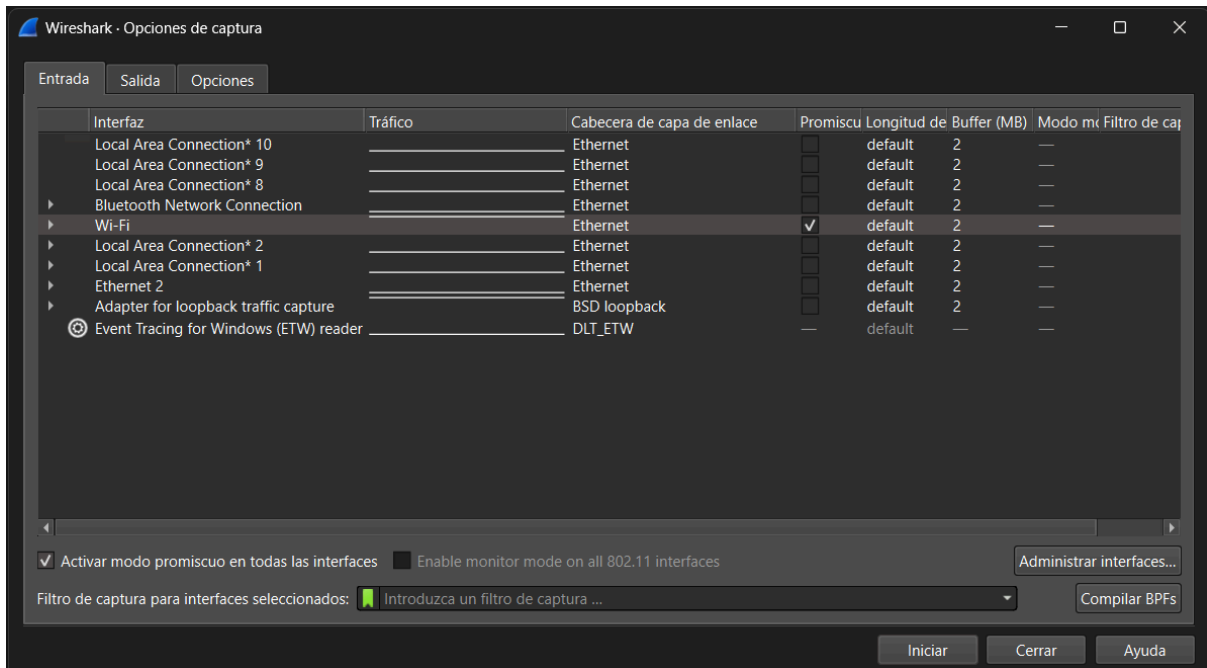
Ethernet adapter Bluetooth Network Connection:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 
System32> |
```

Al ejecutar el comando `ipconfig` en la terminal de Windows, observé varias interfaces de red con diferentes estados y configuraciones.

- La interfaz Ethernet 2 muestra una dirección IPv4 192.168.56.1 y una máscara 255.255.255.0, sin puerta de enlace. Esto indica que es probablemente una red local virtual (por ejemplo, de una máquina virtual), ya que no está conectada a internet.
- Las interfaces Local Area Connection 1* y 2 aparecen como “Media disconnected”, lo que significa que el adaptador físico existe pero no tiene conexión activa.
- La interfaz Wi-Fi está correctamente conectada: tiene IP 10.100.16.139, máscara 255.255.224.0 y puerta de enlace predeterminada 10.100.0.1. Esto confirma que es la interfaz que se está utilizando para navegar por la red y es la adecuada para realizar nuestra captura de paquetes.
- El adaptador Bluetooth Network Connection también aparece como “Media disconnected”, por lo que no está en uso.

Para entender mejor estos resultados, consulté la documentación oficial de Microsoft y fuentes adicionales. Aprendí que una dirección Link-local IPv6 (como `fe80::...`) es una dirección automática asignada por el sistema para comunicación local en la misma subred, y no se utiliza para acceso a internet. Además, la presencia de varias interfaces “disconnected” es normal en laptops con múltiples adaptadores inalámbricos o virtuales que no están activos.

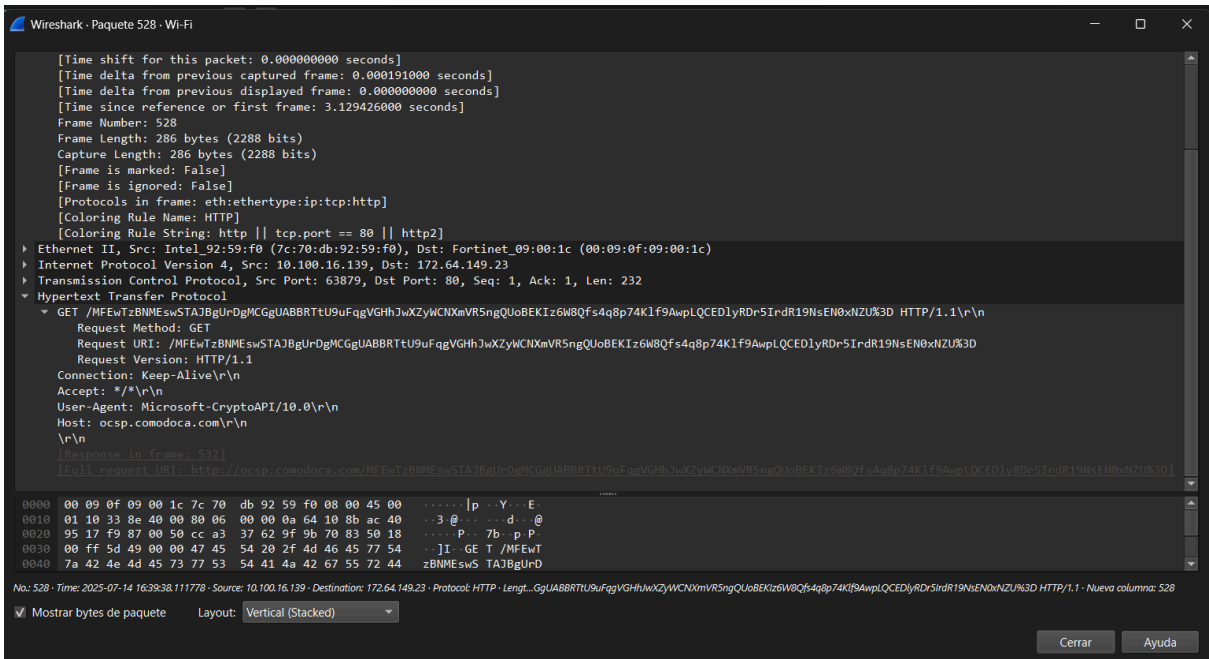
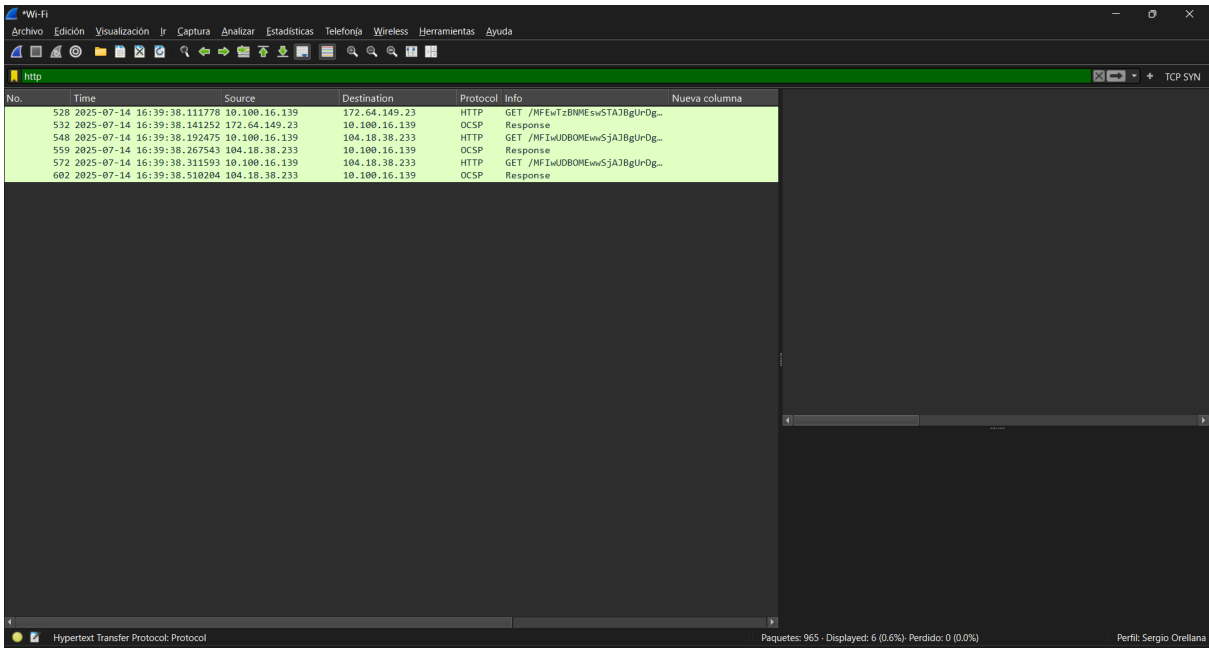


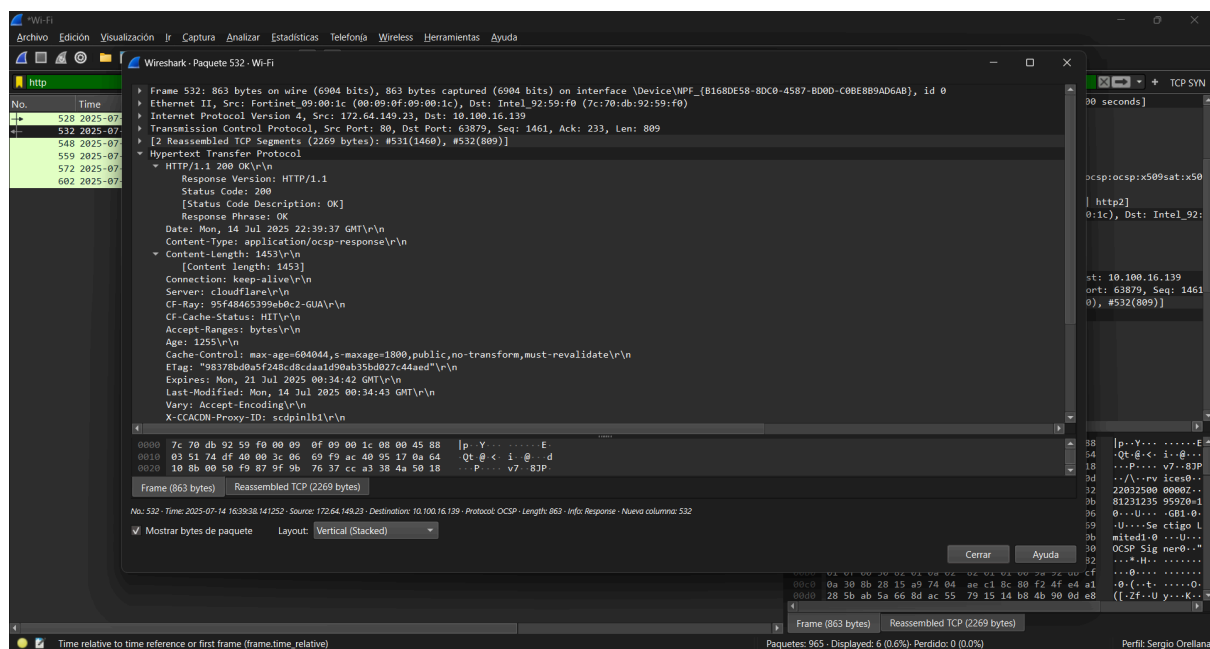
Name	Status	Date modified	Type	Size
lab1_221122		14/07/2025 16:01	File	5,002 KB
lab1_221122_20250714160355_00004		14/07/2025 16:04	File	5,001 KB
lab1_221122_20250714160409_00005		14/07/2025 16:04	File	5,002 KB
lab1_221122_20250714160411_00006		14/07/2025 16:04	File	5,001 KB
lab1_221122_20250714160418_00007		14/07/2025 16:04	File	5,001 KB
lab1_221122_20250714160434_00008		14/07/2025 16:04	File	5,001 KB
lab1_221122_20250714160436_00009		14/07/2025 16:04	File	5,001 KB
lab1_221122_20250714160438_00010		14/07/2025 16:04	File	5,001 KB
lab1_221122_20250714160458_00011		14/07/2025 16:04	File	5,001 KB
lab1_221122_20250714160459_00012		14/07/2025 16:05	File	5,001 KB
lab1_221122_20250714160501_00013		14/07/2025 16:05	File	3,209 KB

ns | 10 items selected 47.0 MB | Sync pending |

Verifiqué la creación de los archivos revisando la carpeta destino. A medida que la captura avanzaba y se alcanzaban los 5 MB, Wireshark generó nuevos archivos automáticamente, como se esperaba. Validé el tamaño de los archivos en el sistema de archivos y comprobé que el buffer cíclico funcionó correctamente hasta el límite de 10 archivos y cuando llegaba a la cantidad máxima elimina el archivo más antiguo que se había generado.

1.3. Análisis de paquetes





Responda las siguientes preguntas:

a. ¿Qué versión de HTTP está ejecutando su navegador?

Esto indica que el navegador está utilizando HTTP/1.1 para realizar la solicitud. “GET ... HTTP/1.1”

b. ¿Qué versión de HTTP está ejecutando el servidor?

La respuesta del servidor muestra “HTTP/1.1 200 OK” lo que confirma que el servidor también opera con HTTP/1.1.

c. ¿Qué lenguajes (si aplica) indica el navegador que acepta a el servidor?

Esto significa que el navegador acepta cualquier tipo de contenido (cualquier MIME type), representado por el comodín “/*/*”.

d. ¿Cuántos bytes de contenido fueron devueltos por el servidor?

El servidor devolvió exactamente 1453 bytes de contenido en esa respuesta.

e. En el caso que haya un problema de rendimiento mientras se descarga la página, ¿en que elementos de la red convendría “escuchar” los paquetes? ¿Es conveniente instalar Wireshark en el servidor? Justifique.

Para diagnosticar posibles problemas de rendimiento al descargar la página, considero fundamental capturar tráfico en varios puntos de la red. En primer lugar, capturaría en mi propio cliente para medir los tiempos desde que envío la solicitud hasta que recibo la respuesta, identificando pausas o retardos visibles para el usuario. Luego, incluiría una captura en el router o punto intermedio; esto me permitiría detectar pérdidas de paquetes, retransmisiones o congestión en el trayecto. Por último, si tengo acceso a un entorno de prueba del servidor, también realizaría una captura allí para analizar posibles cuellos de botella en el procesamiento interno del servicio HTTP.

Sin embargo, tengo en cuenta que no es recomendable instalar Wireshark directamente en servidores de producción reales, ya que puede comprometer la seguridad al capturar tráfico sensible, además de consumir recursos del sistema, lo que podría afectar el rendimiento bajo carga elevada.

Discusión sobre la actividad y hallazgos

Durante la primera parte del laboratorio, dedicada a los esquemas de comunicación (Morse y Baudot), pude comparar directamente la facilidad de uso y la propensión a errores de ambos sistemas. Mi experiencia con Baudot fue complicada: interpretar secuencias de cinco bits sin ninguna lógica fonética presente resultó incómodo, especialmente al realizar la transmisión mediante audio por Zoom. En varias ocasiones me confundí entre secuencias de bits, lo cual se evidenció en los mensajes errados que transmitieron mis compañeros (por ejemplo, “Ceneral Kenomi” en lugar de “General Kenobi”).

En contraste, el código Morse se mostró mucho más intuitivo. Me bastó con diferenciar entre sonidos cortos y prolongados y respetar pausas claras entre letras. Aunque también hubo errores (como transformar “Mensaje difícil” en “Mensape difícil”), estos fueron mínimos y fáciles de corregir.

En la segunda parte, de transmisión empaquetada por WhatsApp, el desafío se intensificó: sin poder pedir aclaraciones en tiempo real, una sola pausa mal hecha podía cambiar el significado de un mensaje. En mi caso, cometí hasta 15 errores en una transmisión, ya que al no haber pausas distinguidas, fue complicado delimitar los límites de cada letra. Esta experiencia reforzó lo que ya habíamos visto: Morse es más tolerante y Baudot, aún menos viable cuando se pierde la retroalimentación instantánea.

En la introducción a Wireshark, me enfrenté a un entorno completamente nuevo. Primero identifiqué mis interfaces con ipconfig, confirmé que la interfaz Wi-Fi era activa (tenía dirección 10.100.16.139), y procedí a configurar un perfil personalizado en Wireshark con reglas de color y filtros de TCP-SYN. Después configuré la captura con ring buffer de 5 MB

por archivo, hasta 10 archivos, generando tráfico para llenar los archivos. Finalmente, realizando una captura HTTP, identifiqué que tanto navegador como servidor usaban HTTP/1.1, encontré encabezados como Accept: */*, y determiné que el servidor devolvió 1 453 bytes de contenido.

Comentarios

- Sobre la dinámica de Morse vs Baudot, me pareció muy ilustrativo ver cómo la complejidad del protocolo influye directamente en la facilidad de aprendizaje y precisión. Morse se siente más natural, especialmente al tener retroalimentación inmediata.
- La transmisión empaquetada por WhatsApp evidenció lo importante que es la claridad y planificación al enviar mensajes codificados: sin feedback, la tasa de error sube dramáticamente.
- En cuanto al análisis de tráfico con Wireshark, me gustó ver lo sencillo que es configurar reglas de color, filtros y layout personalizado. La experiencia me permitió entender mejor cómo monitorear tráfico en tiempo real y diagnosticar protocolos como HTTP.

Conclusiones

- El código Morse es significativamente más práctico y confiable que Baudot para transmisiones orales, en tiempo real o empaquetadas, especialmente por su simplicidad y naturalidad.
- El uso de captura por ring buffer en Wireshark permite recolectar tráfico de forma controlada, separada en archivos manejables, lo que facilita su análisis posterior.
- La investigación de las solicitudes HTTP me permitió confirmar versiones de protocolo, encabezados clave (Accept, Content-Length), y validar la comunicación entre cliente y servidor.
- Finalmente, adquirí habilidades útiles: creación de perfiles personalizados, reglas de color para tráfico específico y filtros interactivos. Esto refuerza la importancia de herramientas como Wireshark para el análisis de redes, sin necesidad de instalar software pesado en servidores de producción.

Referencias

- Stevewhims. (s. f.). IPv6 Link-local and Site-local Addresses - Win32 apps. Microsoft Learn.
<https://learn.microsoft.com/en-us/windows/win32/winsock/link-local-and-site-local-addresses-2>
- runebook.dev. (s. f.). Troubleshooting Accept Header Errors: Ensuring Smooth Data Flow in Web Applications. <https://runebook.dev/en/articles/http/headers/accept>
- EricLaw. (2013). What does 'Accept: */*' mean under Client section of Request Headers?
<https://stackoverflow.com/questions/14772634/what-does-accept-mean-under-client-section-of-request-headers>
- Cyberly. (2024, 3 diciembre). What are the limitations of using Wireshark for network analysis?
<https://www.cyberly.org/en/what-are-the-limitations-of-using-wireshark-for-network-analysis/index.html>