Módulo 1. Visibilidad de red

CÓRDOBA CREA FUTURO

Introducción

La visibilidad de red es la base para entender qué sucede en la infraestructura de una empresa. Consiste en poder monitorear y analizar el tráfico de datos que circula por los dispositivos y enlaces de la red. De esta forma, se conoce el funcionamiento normal de una determinada red, lo cual resulta fundamental, dado que todas las redes son distintas. Esto permite identificar comportamientos anómalos, resolver problemas de conectividad y asegurar que los sistemas se comuniquen de forma correcta y segura.

En entornos de pymes (pequeñas y medianas empresas) con poco personal técnico, lograr visibilidad de red resulta especialmente importante, ya que muchas veces no se cuenta con herramientas costosas ni con un equipo especializado. Por este motivo, herramientas gratuitas como Wireshark se vuelven aliados fundamentales.

Wireshark es un analizador de paquetes de red ampliamente utilizado a nivel mundial. Su gran ventaja es que permite capturar todo el tráfico que entra y sale por una interfaz de red y detallarlo por capas y protocolos, de forma comprensible. Gracias a herramientas como esta, incluso un operador de seguridad con recursos limitados puede inspeccionar lo que ocurre en la red de su organización. Por ejemplo, con Wireshark es posible detectar cuellos de botella, configuraciones incorrectas o comportamientos anómalos en los dispositivos conectados.

A lo largo de este módulo adoptaremos un estilo técnico-pedagógico claro, explicando paso a paso cómo usar Wireshark para lograr dicha visibilidad, con un enfoque práctico orientado a pymes.

Objetivos del módulo

Al finalizar el módulo, serás capaz de realizar capturas de tráfico en tu red, filtrar y analizar esos datos para obtener información relevante, reconocer patrones normales (DNS, HTTP, sesiones TLS) y detectar posibles anomalías. Se incluirán

laboratorios guiados con ejercicios prácticos, utilizando solo herramientas gratuitas, y un caso de estudio final que integrará todos los conceptos aprendidos. También se proporcionarán referencias académicas y técnicas actualizadas para profundizar en los temas tratados.

CONTINUAR

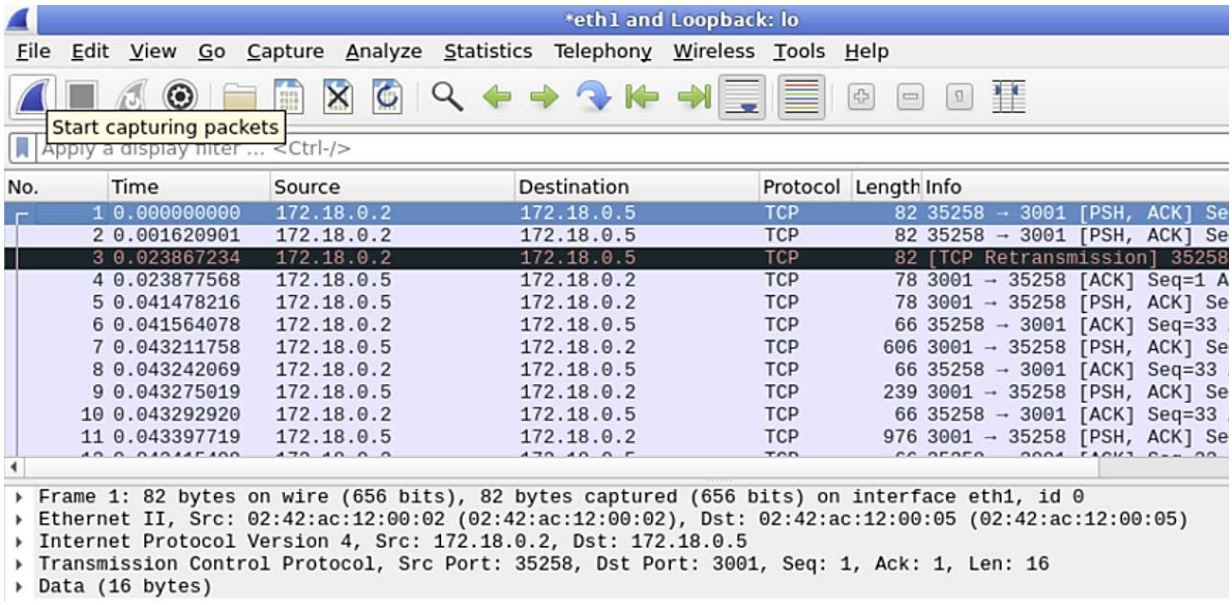
Unidad 1. Monitoreo de redes de datos. Wireshark

En esta primera unidad nos enfocaremos en los fundamentos del monitoreo de tráfico de red usando Wireshark, la herramienta por excelencia para inspeccionar paquetes. Veremos cómo realizar capturas y guardar esos datos en archivos PCAP, cómo usar filtros de visualización para encontrar información específica y cómo personalizar la interfaz mediante perfiles y columnas para facilitar el análisis. Por último, abordaremos las opciones de exportación de datos y buenas prácticas para registrar evidencia, algo importante cuando los hallazgos pudieran usarse en diagnósticos o investigaciones de seguridad.

Captura y PCAP

En la siguiente figura se muestra la lista de interfaces de red disponibles para captura y el botón «Iniciar captura» resaltado.

Figura 1: Interfaz principal de Wireshark (ejemplo en Linux)



Fuente: captura de pantalla de Wireshark (Wireshark Foundation, 2025)

Capturar tráfico con Wireshark implica colocar la tarjeta de red en modo promiscuo para interceptar las tramas que atraviesan dicha interfaz. Al abrir Wireshark, lo primero que se muestra es la lista de interfaces de red disponibles en el sistema. Para comenzar, se debe seleccionar la interfaz adecuada (por ejemplo, la interfaz Ethernet o wifi activa) y luego hacer clic en el botón de captura (el ícono de aleta de tiburón verde). A partir de ese momento, Wireshark

registrará todos los paquetes entrantes y salientes por esa interfaz.

Es importante tener en cuenta que, en muchos sistemas, se requieren privilegios de administrador para capturar paquetes. En Windows, por ejemplo, la instalación de Wireshark incluye la librería Npcap, que permite esta función; en Linux, suele ser necesario agregar el usuario al grupo «wireshark» o ejecutar la aplicación con «sudo» para obtener permiso de captura.

Cada captura de Wireshark puede guardarse en un archivo con formato PCAP (*packet capture*) o en su versión extendida, PCAPNG (*PCAP next generation*). Un archivo PCAP es, básicamente, un contenedor que almacena las tramas capturadas byte por byte, junto con su correspondiente marca de tiempo (*timestamp*) y los metadatos de la captura. Este formato es estándar y es compatible con diversas herramientas, como tcpdump o Snort, lo que permite que las capturas realizadas en un sistema puedan analizarse posteriormente en otro. Wireshark, por defecto, utiliza «.pcapng», una versión más moderna que admite funcionalidades adicionales, como marcas de tiempo con mayor precisión o comentarios del analista. En la práctica, ambos formatos cumplen con el

propósito fundamental: permitir que posteriormente se reexamine el tráfico grabado como si se estuviera «reproduciendo» la comunicación original.

Al iniciar la captura, se recomienda cerrar aplicaciones que puedan generar ruido o tráfico no relevante, de modo que la recopilación se centre en lo que resulta de interés. Una vez comenzada, Wireshark irá listando en tiempo real los paquetes: cada fila representa una trama capturada, con columnas como número, hora, origen, destino, protocolo, longitud e información breve. Se puede dejar correr la captura el tiempo necesario para observar el fenómeno en cuestión (por ejemplo, realizar una prueba de conexión o reproducir un error). Una vez finalizada, se debe presionar el botón «Stop» (cuadro rojo) para detener la captura. Es fundamental guardar el archivo de inmediato: se debe ir al menú «File» → «Save» (o hacer clic en el ícono de guardar) y almacenar el archivo «.pcapng» en un lugar seguro.

GUARDAR LAS CAPTURAS OFRECE VARIAS VENTAJAS. A CONTINUACIÓN, SE ENUMERAN ALGUNAS DE ELLAS:

FILTROS DE CAPTURA VS. FILTROS DE VISUALIZACIÓN

1. Permiten analizarlas con detenimiento más adelante, aplicando distintos filtros sin riesgo de perder información. Esto resulta

indispensable en auditorías, donde el analista recolecta las evidencias —como el tráfico de red— y luego las examina.

2. Facilitan compartir el archivo con colegas o personas expertas de confianza para obtener ayuda, siempre teniendo en cuenta que el archivo PCAP puede contener datos sensibles.

3. Funcionan como evidencia documentada de lo ocurrido en la red en una fecha y hora determinadas. En entornos de ciberseguridad, conservar la integridad de estos archivos es importante. Se aconseja no modificarlos y, de ser posible, calcular un *hash* (MD5, SHA-256) si se van a presentar como evidencia forense, con el fin de demostrar que no han sido alterados.

Nota: Wireshark captura todo el tráfico del adaptador seleccionado, salvo que se configuren filtros de captura (ver más abajo). Esto incluye, potencialmente, información sensible —como credenciales en texto plano o datos personales— si dicha información circula por la red sin cifrar. Por lo tanto, los archivos PCAP deben manejarse con precaución y compartirse únicamente por medios seguros. En una pyme, una práctica recomendada es almacenarlos en ubicaciones accesibles solo para el personal autorizado de TI o seguridad.

GUARDAR LAS CAPTURAS OFRECE VARIAS VENTAJAS. A CONTINUACIÓN, SE ENUMERAN ALGUNAS DE ELLAS:

FILTROS DE CAPTURA VS. FILTROS DE VISUALIZACIÓN

Wireshark permite establecer un filtro en el momento de capturar, conocido como *capture filter*, para limitar qué paquetes se guardan en el archivo PCAP. Estos filtros utilizan la sintaxis de BPF (Berkeley Packet Filter), similar a la de «tcpdump», y resultan útiles si se sabe de antemano que solo interesa cierto tipo de tráfico. Por ejemplo, se podría utilizar el comando `host 192.168.1.5 and port 80` para capturar únicamente tráfico web de un host específico. Sin embargo, un error en un filtro de captura podría provocar la pérdida de información importante. En la mayoría de los casos en pymes, suele preferirse capturar todo (particularmente si no se trata de un volumen excesivo de datos) y luego aplicar filtros durante el análisis. Por esta razón, en la siguiente sección se abordarán los filtros de visualización, que no descartan paquetes, sino que permiten mostrarlos u ocultarlos dinámicamente según criterios definidos.

En resumen, para realizar una captura efectiva, se deben seguir los siguientes pasos:

- seleccionar la interfaz correcta;
- iniciar la captura en el momento oportuno;
- detenerla cuando se disponga de suficiente información;

- guardar el resultado en un archivo PCAP.

Este archivo es nuestra «grabación» del tráfico, que podrá reproducirse y analizarse tantas veces como sea necesario. Un detalle adicional es que muchos equipos de red —como *firewalls* o *routers*— permiten exportar capturas en formato PCAP. Por ejemplo, *firewalls* con firmware como pfSense cuentan con utilidades integradas para capturar paquetes en una interfaz, incluso aplicando allí un filtro básico por dirección IP o puerto; luego ofrecen descargar el archivo «.pcap» resultante. Esto resulta especialmente útil: se podría capturar tráfico directamente en el *router* de la empresa (para observar qué sale hacia Internet) y luego abrir ese archivo en Wireshark para realizar un análisis detallado. De este modo, al combinar las capacidades de los dispositivos de red con las de Wireshark, se obtiene una visibilidad completa de lo que ocurre.

Filtros de visualización

Al capturar paquetes, es común que se acumulen rápidamente cientos o miles de entradas en pantalla. ¿Cómo enfocarse en lo relevante? Aquí es donde intervienen los filtros de visualización (*display filters*) de Wireshark. Estos

filtros se aplican después de la captura y permiten mostrar únicamente aquellos paquetes que cumplan cierta condición, ocultando el resto. Los datos siguen estando en memoria; simplemente no se listan mientras el filtro esté activo. La sintaxis de los filtros de visualización es propia de Wireshark —distinta de la de los filtros de captura— y ofrece un alto grado de flexibilidad.

Figura 2. Ejemplos de filtros de visualización

Fuente: Cisco, 2025, https://short.do/12mDnT

La barra de filtros de Wireshark aparece en la parte superior de la ventana principal, con el texto «Apply a display filter». Allí se pueden escribir las expresiones de filtrado. Cuando la

sintaxis es válida, el recuadro se muestra en verde; si se introduce algo incorrecto, aparecerá en rojo para indicar un error.

1. Filtrar por protocolo

Simplemente, se debe escribir el nombre del protocolo. Por ejemplo:

- `dns` muestra solo paquetes DNS.
- `http` muestra únicamente tráfico HTTP.
- `tcp` muestra solo segmentos TCP.

También se pueden combinar protocolos. Por ejemplo, `http || dns` mostrará paquetes HTTP o DNS (el operador `||` representa una disyunción lógica).

De forma análoga, se puede utilizar && para conjunción y ! (o not) para negación. Por ejemplo, `tcp && !http` indica que se deben mostrar solo los paquetes TCP que no sean HTTP.

2. Filtrar por dirección IP

Se utilizan los campos `ip.src` (IP de origen), `ip.dst` (IP de destino) o `ip.addr` (cualquiera de las dos). Por ejemplo, `ip.addr == 192.168.1.40` permitirá ver todo el tráfico hacia o desde la IP 192.168.1.40. Si solo interesa como destino: `ip.dst == 192.168.1.40`.

También es posible excluir tráfIco con `!=`. Por ejemplo, `ip.addr != 192.168.1.1` ocultará cualquier paquete donde la IP de origen o destino sea 192.168.1.1, lo cual puede ser útil para ignorar, por ejemplo, el tráfico del *router*.

3. Filtrar por puerto

Se utilizan los campos `tcp.port` o `udp.port`, que aplican tanto a puertos de origen como de destino. Por ejemplo, `tcp.port == 80` mostrará todos los segmentos TCP con puerto 80, es decir, tráfico web sin cifrar.

También se pueden combinar con direcciones IP. Por ejemplo, `ip.addr == 10.0.0.5 && tcp.port == 443` mostrará el tráfico entre la IP 10.0.0.5 y el puerto 443, es decir, tráfico HTTPS desde o hacia esa máquina. Para UDP se aplica del mismo modo: `udp.port == 53` filtrará tráfico DNS, ya que DNS suele utilizar UDP en el puerto 53.

4. Búsqueda de contenido

Una función útil es el operador `contains`, que permite buscar una cadena dentro del contenido del paquete. Por ejemplo:

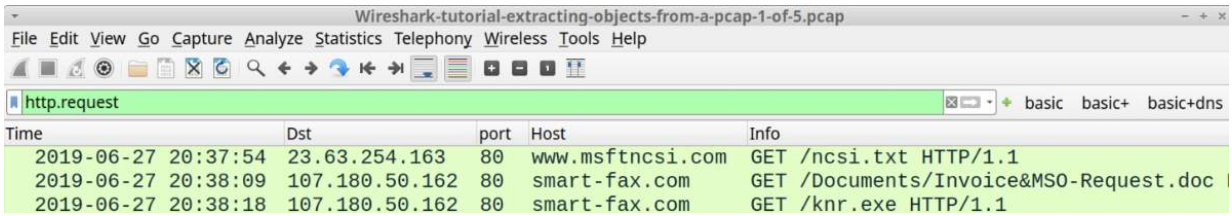
- `http contains "User-Agent"` buscará paquetes HTTP que contengan la cadena "User-Agent", probablemente en el encabezado HTTP.
- `frame contains "MiEmpresa"` buscará cualquier aparición de "MiEmpresa" en el contenido bruto de cualquier paquete (el campo `frame` representa todo el contenido del paquete).

Este tipo de filtro es útil para localizar datos específicos, aunque puede ser más lento si la captura es muy grande.

En la siguiente figura se observa un ejemplo real de filtrado. El filtro `http.request` es un atajo proporcionado por Wireshark, equivalente a `http.request.method == "GET"`, que muestra únicamente paquetes correspondientes a peticiones HTTP (generalmente del tipo GET cuando se accede a páginas web). En la lista de paquetes filtrados, la

columna *Info* muestra, por ejemplo, `GET /ncsi.txt HTTP/1.1` dirigido al *host* www.msftncsi.com (tráfico generado por Windows para comprobar conectividad), junto con otras dos solicitudes GET al dominio smart-fax.com para descargar un documento `.doc` y un ejecutable `.exe`. Este filtro permitió aislar rápidamente las acciones HTTP dentro del volumen general de tráfico.

Figura 3. Uso de un filtro de visualización en Wireshark



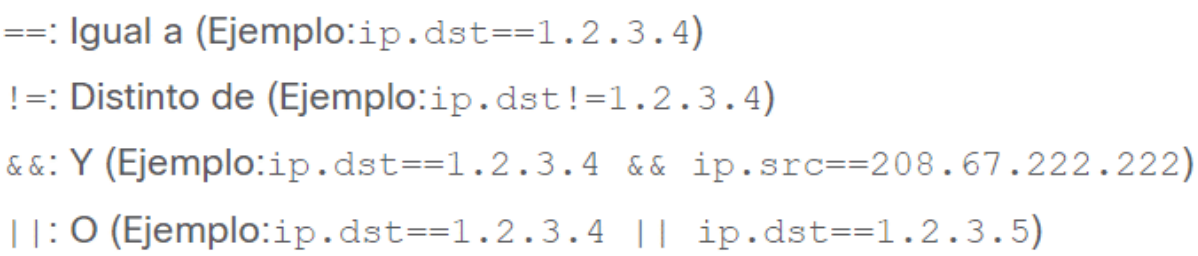
Fuente: Unit 42, s.f., https://short.do/5HR7Jh

Combinación y sintaxis

Cuerpo del texto: 18 ptos. #000000. Then we show up to the present moment with all of our senses, we invite the world to fill us with joy. The pains of the past are behind us. The future has yet to unfold. But the now is full of beauty simply waiting for our attention.Los filtros de visualización admiten operadores de comparación (`==`, `!=`, `>`, `<`, etc.) y operadores lógicos (`and`, `or`, `not`). Esto permite construir expresiones muy

precisas para ajustar la visualización según los criterios deseados.

Figura 4. Operadores en filtros de visualización



Fuente: Cisco, 2025, https://short.do/12mDnT

A continuación, se presentan algunos casos específicos que muestran el uso combinado de campos y operadores en filtros de visualización:

- `tcp.flags.reset == 1 && tcp.port == 80` mostrará paquetes TCP en el puerto 80 que tengan el *flag* RST activo, es decir, conexiones HTTP que fueron reseteadas. Esto resulta útil para diagnosticar caídas de conexión.

- `dns.flags.rcode == 3` permitirá encontrar respuestas DNS que indican que el nombre de

dominio no fue encontrado (código 3 en DNS, correspondiente a NXDomain).

Wireshark también ofrece filtros predeterminados, como `tcp.analysis.retransmission` o `tcp.analysis.flags`, que permiten resaltar retransmisiones, duplicados y otros eventos detectados automáticamente por el programa.

Diferencia con filtros de captura —

Como se ha mencionado anteriormente, un filtro de captura determina qué paquetes se guardan en el archivo, mientras que el filtro de visualización solo afecta lo que se muestra en pantalla en un momento determinado. Si se olvida aplicar un filtro de captura al grabar, siempre es posible filtrar posteriormente mediante visualización; pero si se aplica un filtro de captura demasiado restrictivo, no será posible recuperar los paquetes que nunca llegaron a guardarse. Por esta razón, salvo que existan limitaciones de espacio o se esté diagnosticando algo muy puntual, en pymes suele preferirse capturar todo y filtrar después.

Aplicar múltiples filtros secuencialmente —

Es posible cambiar el filtro cuantas veces sea necesario durante el análisis. En Wireshark, al borrar o modificar la expresión y pulsar el botón «Apply», la lista de paquetes se actualiza de forma inmediata.

Además, se pueden preparar filtros frecuentes como botones de acceso rápido, disponibles en la barra de herramientas superior a través del menú «Analyze» → «Display Filter Quick Buttons». También se incluye, por defecto, un diálogo de ayuda para construir filtros: el botón «Expression...» permite acceder a una lista de campos disponibles según los protocolos presentes en la captura, lo que facilita crear filtros sin necesidad de recordar toda la sintaxis.

A continuación, se presentan algunos ejemplos:

Tabla 1. Ejemplos de filtros de captura y visualización en Wireshark

Expresión	Descripción
`not arp`	Captura todos los tipos de paquetes excepto ARP.
`port 22`	Captura paquetes cuyo puerto de origen o destino sea el 22, sin importar si es TCP o UDP.
`tcp port 443`	Captura solo paquetes TCP cuyo puerto de origen o destino sea el 443.

`not port 25 and not port 53`	Ignora todos los paquetes TCP o UDP cuyo puerto sea el 25 o el 53.
`tcp src port 80`	Captura paquetes TCP cuyo puerto de origen sea el 80.
`tcp[tcpflags] == tcp-syn`	Captura los paquetes TCP con la *flag* SYN.
`tcp[tcpflags] == (tcp-syn + tcp-ack)`	Captura los paquetes TCP con las *flags* SYN+ACK.
`tcp[tcpflags] == tcp-rst`	Captura los paquetes TCP con la *flag* RST.
`ip.addr == IP`	Filtra por la dirección IP indicada, ya sea origen o destino.
`ip.dst == IP_DST && ip.src == IP_SRC`	Filtra por las direcciones IP indicadas en origen y destino.
`http or dns`	Muestra todos los paquetes HTTP o DNS capturados.

`tcp.port == PORT`	Muestra los paquetes TCP cuyo puerto de origen o destino coincida con el especificado.
`tcp.flags.reset == 1`	Muestra todos los paquetes TCP que tengan la *flag* RST activada.
`tcp contains TERMINO`	Muestra paquetes TCP que contengan el término especificado.
`!(arp or icmp or dns or ssdp or udp)`	Muestra solo los paquetes que no utilizan los protocolos ARP, ICMP, DNS, SSDP o UDP.
`tcp.port in {80 25}`	Muestra todos los paquetes cuyo puerto TCP en origen o destino sea el 80 o el 25.
`!(tcp.port in {22 443})`	Muestra todos los paquetes cuyo puerto TCP en origen o destino no sea el 22 o el 443.
`host 192.168.1.20`	Captura todos los paquetes con origen y destino en 192.168.1.20.

`src host 192.168.1.1`	Captura todos los paquetes cuyo origen sea 192.168.1.1.
`dst host 192.168.1.1`	Captura todos los paquetes cuyo destino sea 192.168.1.1.
`dst host SERVER-1`	Captura todos los paquetes con destino en el host SERVER-1.
`host` http://www.terra.com	Captura todos los paquetes con origen y destino en http://www.terra.com.

Fuente: elaboración propia

En resumen, los filtros de visualización funcionan como binoculares para navegar por la captura. Dominarlos permite ahorrar una gran cantidad de tiempo y facilita encontrar la proverbial aguja en el pajar dentro de gigabytes de tráfico. A medida que se gana práctica, la persona operadora irá memorizando los filtros más útiles para su entorno (DNS, HTTP, errores TCP, entre otros). En los anexos y referencias se pueden consultar listas de filtros comunes recomendados para tareas de administración y *pentesting*.

Perfiles y columnas

Wireshark ofrece muchas opciones de personalización de la interfaz para facilitar el análisis según nuestras necesidades. Dos de las más útiles para un operador de seguridad son la configuración de **perfiles** y la customización de **columnas** en la vista de paquetes.

PERFILES DE CONFIGURACIÓN | COLUMNAS PERSONALIZADAS | EJEMPLO PRÁCTICO

Un perfil en Wireshark es un conjunto de preferencias y ajustes (colores, columnas visibles, reglas de resaltado, etc.) que se pueden guardar con un nombre, de manera que sea posible cambiar rápidamente entre distintas configuraciones. Por ejemplo, se podría tener un perfil por defecto con la visualización estándar, y otro perfil llamado «DNS-HTTP», en el que se hayan agregado columnas especiales y reglas de color para esos protocolos. Al cambiar de perfil, la interfaz de Wireshark se adapta inmediatamente al caso de uso.

En entornos de pyme, los perfiles permiten que un mismo operador cambie su modo de trabajo: quizá un día esté analizando tráfico web, y al siguiente esté investigando latencia en enlaces; cada tarea podría beneficiarse de un perfil distinto.

Wireshark incluye un perfil por defecto, y es posible crear nuevos desde el menú «Editar» → «Perfiles de configuración» (o «Edit» → «Configuration Profiles» en inglés). Desde allí se puede copiar el perfil por defecto y luego modificarlo a gusto, para no empezar desde cero. No hay riesgo en experimentar con perfiles: siempre se puede volver al perfil por defecto para restablecer la vista original.

Por defecto, la lista de paquetes en Wireshark muestra columnas como «No.», «Time», «Source», «Destination», «Protocol», «Length» e «Info». Estas resultan adecuadas para muchas situaciones, pero en algunos casos puede ser útil mostrar otros campos directamente como columnas, para evitar revisar el panel de detalles en cada paquete. Por ejemplo, en un análisis DNS se podría querer una columna que muestre el nombre de dominio consultado en cada paquete DNS; o, en un análisis TCP, agregar una columna para el número de secuencia o el TTL de los paquetes IP.

Wireshark permite agregar, quitar y reordenar columnas libremente. Para hacerlo, se debe seguir este procedimiento:

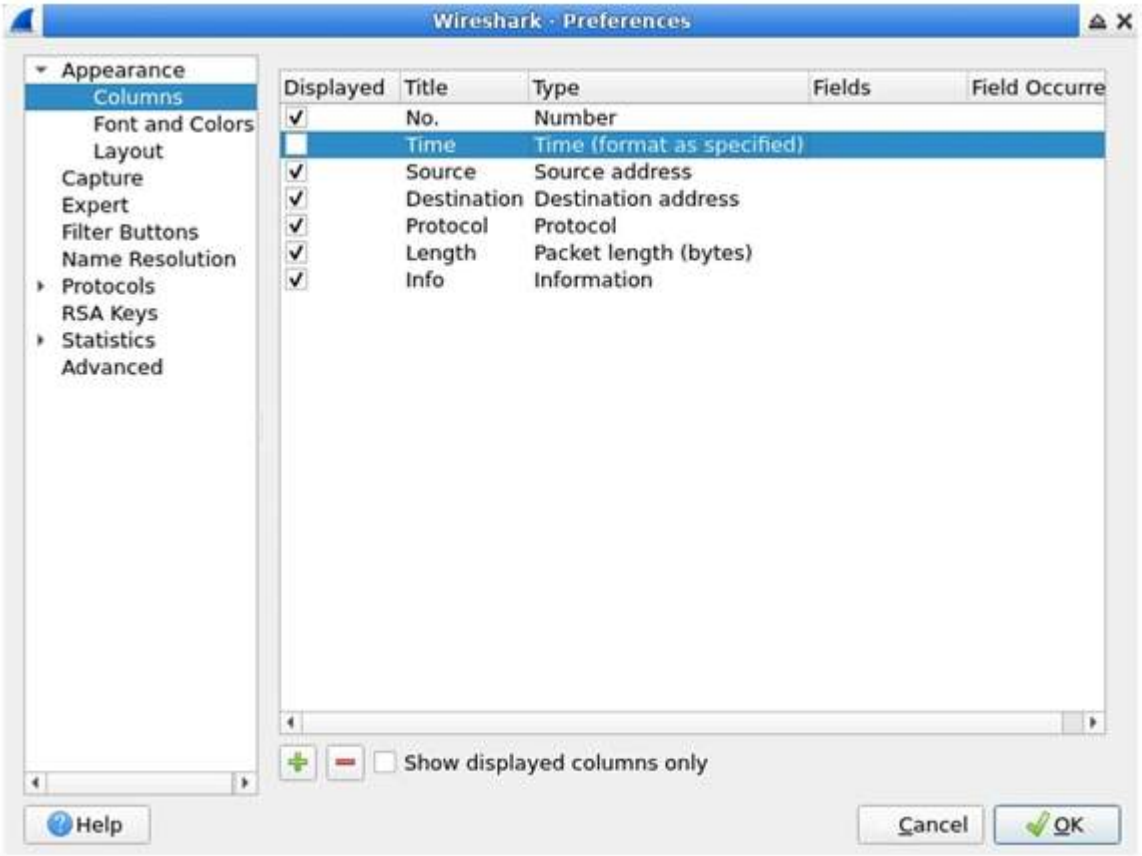
- Hacer clic derecho sobre cualquier encabezado de columna y elegir «Preferencias de columnas». Se abrirá una ventana con la lista de columnas actuales y botones para añadir (+), eliminar (−) o mover (flechas arriba/abajo) columnas.
- Al hacer clic en el botón de agregar (+), se puede seleccionar un tipo de columna predefinido o elegir «Custom» (Personalizado) para especificar un campo arbitrario. Luego se asigna el campo deseado y un título para la columna. Por ejemplo, si se quiere una columna con la IP de origen, se debe escoger el tipo «Personalizado» y, en «Field name», escribir `ip.src` (campo de dirección IP de origen), y asignar como título *IP Origen*.
- Al hacer clic en «Aceptar», la nueva columna aparecerá en la lista. Se puede mover a la posición deseada (por ejemplo, junto a la columna Source original o en reemplazo de esta).

Una vez aplicados los cambios, la ventana principal mostrará esa información en la tabla de paquetes.

En la siguiente figura se muestra la configuración por defecto: columnas de «No.», «Time», «Source», «Destination», «Protocol», «Length» e «Info». Desde esta ventana, el usuario puede añadir nuevas columnas (botón +), eliminarlas (−) o editar las existentes mediante doble clic. También se pueden reordenar con las flechas.

Los perfiles de Wireshark almacenan estos ajustes de columnas, lo que permite tener distintas vistas de datos según el análisis que se realice.

Figura 5. Ventana de preferencias de columnas en Wireshark



Fuente: captura de pantalla de Wireshark (Wireshark Foundation, 2025)

Supongamos que se está investigando tráfico de correo electrónico SMTP en texto plano. Se podría agregar una columna con `smtp.rcpt_to` para ver directamente las direcciones de correo destinatarias a las que se envía información, o `smtp.req.command` para observar los comandos SMTP. En un análisis de *malware* basado en DNS, agregar una columna con `dns.qry.name` (nombre de consulta DNS) permite ver rápidamente qué dominios se están consultando en cada paquete, sin necesidad de expandir el detalle uno por uno. Esta personalización ahorra tiempo y brinda visibilidad inmediata de datos clave.

Además de las columnas, también se pueden personalizar colores. Wireshark incluye un esquema de coloreado donde, por ejemplo, TCP aparece en azul claro, HTTP en verde y DNS en morado dentro de la lista de paquetes. Estas reglas de color se editan desde el menú «Ver» → «Reglas de color» (en inglés, «View» → «Coloring Rules»). Si se desea, es posible resaltar paquetes ICMP en naranja o tráfico sospechoso en rojo, definiendo reglas basadas en filtros de visualización asociados a colores. Estas reglas también se guardan por perfil, por lo que un perfil orientado a seguridad podría resaltar condiciones como `tcp.flags.syn == 1 and tcp.flags.ack == 0` (posibles SYN scans) en rojo, por ejemplo.

Lo más importante es comprender que Wireshark se adapta a nuestras necesidades: su interfaz es altamente configurable.

Conforme se trabaje con ciertos protocolos o incidentes recurrentes, conviene invertir unos minutos en crear un perfil con columnas y colores que faciliten esas tareas. Un analista en una pyme podría tener un perfil «Performance», en el que agregue columnas con el delta de tiempo entre paquetes y el tamaño de la ventana TCP para diagnosticar lentitud; y otro perfil «Seguridad», que resalte paquetes potencialmente malformados o tráfico no autorizado. Cambiar entre perfiles es tan sencillo como seleccionarlo en la esquina inferior derecha: la barra de estado muestra el perfil activo y permite cambiar con clic derecho.

Por último, cabe mencionar que todos estos cambios en la interfaz no afectan el contenido de la captura. Son simplemente distintas formas de visualizar los mismos datos. Si algo no funciona como se esperaba o ya no se necesitan ciertas columnas, siempre es posible volver al perfil predeterminado para recuperar la vista estándar. No debe temerse la experimentación con la personalización: forma parte del aprovechamiento pleno de Wireshark. Estas habilidades para usar y adaptar Wireshark se vuelven especialmente útiles a medida que se avanza en el análisis de redes y seguridad, ya que permiten responder con flexibilidad a cada situación utilizando la herramienta.

Exportación y evidencia

Exportación

Una vez que se ha capturado y filtrado tráfico de interés, muchas veces será necesario exportar ciertos datos fuera de Wireshark. Por ejemplo, puede identificarse un archivo malicioso que se transfiere por HTTP y que se desea extraer para analizar con un antivirus, o tal vez se quiera guardar un segmento de la captura para adjuntarlo en un reporte de incidente. Wireshark ofrece diversas funcionalidades de exportación:

- **Guardar paquetes seleccionados o filtrados**

Desde el menú «File» → «Save As...» se puede guardar la captura completa o, al marcar la opción «Displayed» (mostrados), guardar únicamente los paquetes que cumplen con el filtro de visualización activo. También es posible seleccionar manualmente ciertos paquetes y, luego, desde «File» → «Export Specified Packets», exportar solo esa selección.

Esta función resulta útil cuando se desea compartir únicamente la parte relevante del tráfico, reduciendo el ruido o el volumen de datos. Por ejemplo, se puede exportar solo la secuencia de tres

pasos de un *handshake* TCP problemático, en lugar de toda la captura. El formato de exportación normalmente será PCAP o PCAPNG, y podrá abrirse posteriormente en Wireshark u otras herramientas compatibles.

- **Seguir flujo (*follow stream*)**

Una de las funciones más potentes de Wireshark es «Follow TCP Stream» (y sus variantes para UDP, TLS, etc.). Al hacer clic derecho sobre un paquete de una conexión y seleccionar «Seguir» → «Flujo TCP», Wireshark mostrará en una ventana aparte la reconstrucción completa de los datos de aplicación intercambiados en esa conexión, ordenados por tiempo. Por ejemplo, en una sesión HTTP se visualiza la petición completa y la respuesta del servidor de forma legible.

Desde esa ventana de seguimiento de flujo es posible guardar el contenido en un archivo de texto. Esta herramienta resulta especialmente útil para extraer, por ejemplo, el texto de correos SMTP capturados, el contenido de una conversación HTTP (HTML, JSON, etc.) o cualquier otra comunicación basada en texto.

Si la comunicación está cifrada (HTTPS/TLS), lo que se verá en «Follow TLS Stream» será binario ininteligible, a menos que se

cuente con las claves necesarias para descifrarla, lo cual excede el alcance de este módulo.

- **Exportar objetos**

Wireshark incluye una opción para extraer archivos u objetos transferidos a través de ciertos protocolos. Por ejemplo, si se transmitieron archivos mediante HTTP, FTP, SMB, entre otros, la herramienta puede reconstruirlos. Esta opción se encuentra en el menú «File» → «Export Objects», que despliega submenús según los protocolos presentes en la captura (por ejemplo, HTTP, SMB, etc.). Al seleccionarla, aparece una lista de objetos detectados en ese tráfico, con columnas como host de origen, tipo de contenido, tamaño y nombre del archivo (cuando es deducible). Luego, se puede seleccionar un objeto y guardarlo.

Un caso de uso común: si se detecta actividad sospechosa y se observa que un equipo descargó un archivo por HTTP, se puede aplicar un filtro `http.request` para identificar la petición GET a un archivo `.exe` (como en la figura 2, donde se solicita `knr.exe`). A continuación, al utilizar la opción «Export Objects» → «HTTP...», Wireshark mostrará ese archivo en la lista, y podrá guardarse localmente para su análisis con antivirus o herramientas forenses.

En la figura 6, observamos que Wireshark identificó que en la captura hubo contenido HTTP transferido: se muestra un archivo de

texto plano («ncsi.txt» de Microsoft de 14 *bytes*) y los dos elementos mencionados anteriormente (el archivo «Invoice&MSO-Request.doc» de tipo *application/msword* y el «knr.exe» de tipo *application/x-msdownload*). Con solo seleccionar cada uno y hacer clic en «Save», es posible recuperar esos archivos tal como fueron recibidos por el cliente durante la captura. Esta capacidad de extraer archivos directamente desde las comunicaciones de red facilita considerablemente la tarea de obtener muestras de malware o documentos para su análisis.

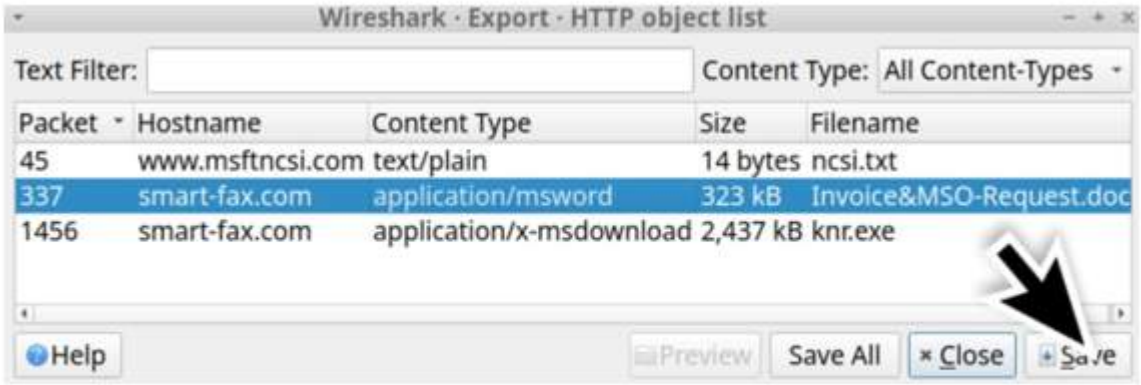
- **Exportar a texto o CSV**

Para propósitos de documentación, a veces se requiere incluir en un informe la lista de ciertas comunicaciones. Wireshark permite exportar el contenido visible de los paquetes a formatos de texto plano, CSV, JSON, entre otros.

Por ejemplo, se puede exportar la lista de paquetes filtrados a CSV para abrirla en Excel, o guardar el flujo seguido de TCP en un archivo «.txt» para adjuntarlo.

Estas opciones se encuentran en el menú «File» → «Export Packet Dissections» (como «Resumen», «Todos los paquetes», etc.) y permiten generar archivos legibles.

Figura 6. Lista de objetos HTTP exportables en Wireshark



Fuente: Unit 42, s.f., https://short.do/5HR7Jh

Manejo de evidencia

En contextos de seguridad, cualquier hallazgo importante conviene respaldarlo con evidencia. Una captura de Wireshark, por sí sola, puede constituir evidencia de que ocurrió cierta comunicación. Al exportar objetos o extraer información, se debe mantener la cadena de custodia si se anticipa un posible uso legal o disciplinario.

En pymes, típicamente no se llega a instancias judiciales, pero vale la pena adoptar buenas prácticas: guardar los archivos PCAP originales en un almacenamiento seguro, anotar la fecha y la persona que realizó la captura, y trabajar siempre sobre copias cuando se vayan a realizar análisis destructivos (por ejemplo, sanitizar datos sensibles antes de compartir con un proveedor).

Si se va a compartir un archivo PCAP con un tercero (por ejemplo, soporte técnico externo), conviene considerar la anonimización de

ciertas partes: Wireshark incluye herramientas (y existen scripts adicionales) para reemplazar direcciones IP reales por direcciones ficticias de forma consistente, protegiendo así la privacidad de la red analizada. Como alternativa, se puede utilizar un filtro de visualización para exportar solo lo relevante y minimizar la exposición de datos.

Por último, recordemos que un archivo PCAP es una representación completa del tráfico en un determinado lapso. Si ese tráfico estaba cifrado (TLS, WPA2, IPsec), los datos de capa de aplicación no serán legibles en la captura, a menos que se proporcionen a Wireshark las claves de descifrado correspondientes. Esto se aplica, por ejemplo, al tráfico wifi protegido (donde, con la contraseña inalámbrica, es posible descifrar el contenido si se ha capturado el *handshake* completo) o a sesiones TLS, si se dispone de la clave privada del servidor o de las *session keys*.

Aunque esto excede el alcance de este módulo, es importante mencionarlo: no siempre será posible ver todo el contenido, lo cual, desde la perspectiva de seguridad, es positivo (el cifrado funciona). Sin embargo, al menos se contará con los metadatos (direcciones IP, puertos, tamaños, patrones de tiempos), que muchas veces son suficientes para realizar un análisis de visibilidad.

Resumen de la unidad 1

En esta unidad aprendimos a capturar tráfico con Wireshark y tomar las precauciones necesarias al hacerlo. También vimos cómo utilizar filtros de visualización para examinar grandes volúmenes de paquetes en busca de información relevante, y cómo personalizar la herramienta mediante perfiles y columnas para trabajar de forma más eficiente. Finalmente, exploramos distintas formas de exportar información clave, ya sea para analizarla por separado o para presentarla como evidencia. Con estos fundamentos, en la unidad 2 aplicaremos lo aprendido al reconocimiento de patrones comunes de tráfico y a la identificación de posibles anomalías de seguridad.

CONTINUAR

Unidad 2. Patrones comunes

En esta unidad profundizaremos en cómo se presentan en la red ciertos patrones típicos (como las consultas DNS, las transferencias web HTTP, el establecimiento de sesiones seguras TLS, entre otros) y cómo Wireshark nos ayuda a interpretarlos. Conocer lo «normal» es fundamental para poder identificar lo «anormal». Por eso, primero describiremos brevemente cada uno de estos flujos comunes y luego abordaremos técnicas para detectar irregularidades o indicadores de compromiso en el tráfico. Todo con un enfoque práctico y adaptado a entornos de pequeñas empresas.

DNS y HTTP

DNS (DOMAIN NAME SYSTEM)	HTTP

Es el sistema de resolución de nombres de dominio a direcciones IP. En la red de una pyme, cada vez que un equipo intenta conectarse a un sitio web (por ejemplo, «www.ejemplo.com») o a algún servicio en Internet, lo habitual es que realice una consulta DNS para obtener la dirección IP de ese dominio. En Wireshark, el tráfico DNS aparece identificado con el protocolo DNS, que utiliza típicamente el puerto UDP 53. Un intercambio típico consta de un paquete de consulta (*standard query*) y su correspondiente respuesta (*standard query response*). Podemos aplicar filtros como `dns o udp.port == 53` para aislar fácilmente estas consultas.

En la siguiente figura se aprecia una «standard query» (consulta estándar) para el nombre «www.redeszone.net» (tipo A, clase IN) enviada desde la IP 10.11.1.2, seguida por la «standard query response» correspondiente, emitida por el servidor DNS 10.11.1.1, que indica una respuesta sin error (*No error*) junto con las direcciones IP asociadas a ese dominio.

Debajo de la lista de paquetes, en el panel de detalles, Wireshark desglosa la respuesta DNS mostrando el nombre consultado y la(s) dirección(es) IP en los registros de respuesta.

Como se observa:

- El cliente (10.11.1.2) realizó una consulta: «¿Cuál es la dirección IP de www.redeszone.net?» (consulta tipo A).
- El servidor DNS local (10.11.1.1) respondió con una respuesta que incluye, entre otros datos, una dirección IPv4 (por ejemplo, «185.99.186.179») asociada a ese nombre.
- Wireshark muestra esta información en el panel inferior de detalles: bajo la capa DNS aparece la sección «Answers», que contiene los registros tipo A con sus respectivos valores IP.

¿Qué patrones son normales en DNS?

Por lo general, un *host* realiza la consulta DNS antes de iniciar una conexión TCP o UDP hacia el destino. Por ejemplo, si abrimos el navegador y escribimos «www.google.com», primero veremos una o varias consultas DNS desde nuestro equipo al servidor DNS —ya sea el de la empresa o el del proveedor de servicios de Internet— para resolver «google.com». Luego, vendrán los paquetes TCP hacia la dirección IP resultante, normalmente en el puerto 443 (HTTPS).
Las consultas DNS típicas tienen ciertas características: el campo «Transaction ID» suele ser un número aleatorio por consulta; la mayoría de las consultas son de tipo A (IPv4) o AAAA (IPv6) para resolver nombres de dominio; y las respuestas suelen ser relativamente pequeñas (unas pocas direcciones, tal vez algún

registro adicional). También pueden aparecer consultas PTR (búsqueda inversa de IP a nombre) y otros tipos, aunque en una red de pyme el tráfico DNS se compone mayormente de resoluciones de nombres para navegación web, correo, entre otros servicios.

Wireshark para diagnosticar problemas DNS

Por ejemplo, si un usuario reporta que «no puede acceder a X sitio», podríamos aplicar un filtro `dns` y verificar si:

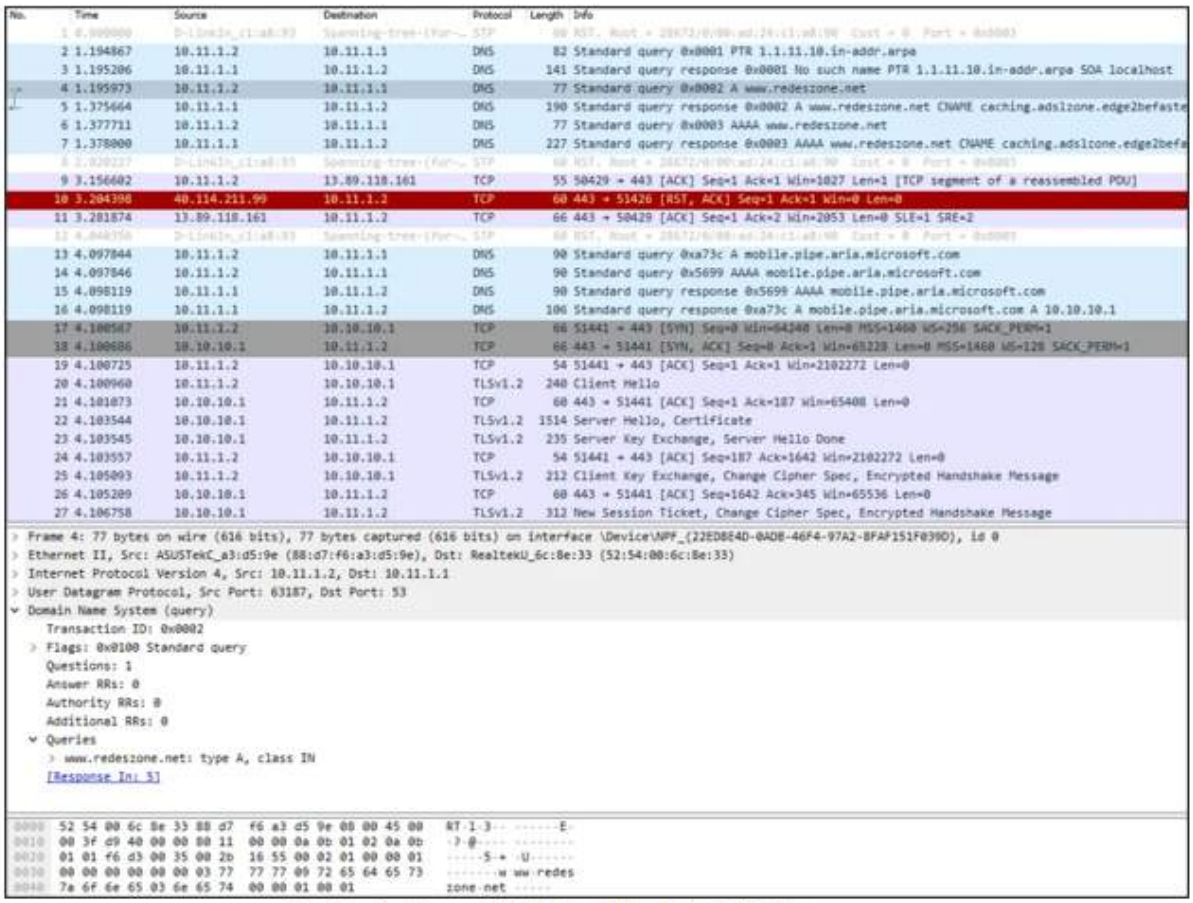
- el equipo realizó la consulta;
- el servidor DNS respondió o no;
- respondió con un código de error (como NXDomain para dominio no existente, o ServFail).

En caso de que no haya respuesta, podríamos observar retransmisiones de la consulta. Wireshark las marca en la columna «Info» como «standard query, retry» o similar, o bien en el panel «Expert Info» como advertencias de retransmisión DNS. Esto podría indicar un problema de conectividad con el servidor DNS.

De hecho, Wireshark es capaz de señalar eventos inusuales como «DNS query retransmission» cuando una consulta se repite sin haber recibido respuesta, lo cual ofrece pistas útiles sobre posibles fallos.

Otro uso práctico es detectar tráfico DNS malicioso o anómalo. El tráfico DNS normal suele apuntar a dominios reconocibles (empresas, servicios habituales) y no satura la red. Si se observa un host realizando cientos de consultas por segundo a nombres extraños (por ejemplo, «una-cadena-aleatoria.xyz»), podría tratarse de malware que usa DNS para comunicarse (túneles DNS, exfiltración o *command and control*). Este punto se ampliará en la sección 2.4 sobre anomalías.

Figura 7. Ejemplo de tráfico DNS capturado



Fuente: Red Zone, 2025, https://short.do/G9hNl2

HTTP es el protocolo de la web que opera en el puerto 80, sin cifrado. Aunque hoy la mayoría del tráfico web utiliza HTTPS (es decir, HTTP sobre TLS), conviene comprender el funcionamiento básico de HTTP porque:

1. Algunas aplicaciones internas todavía lo emplean.
2. Muchos dispositivos en redes pequeñas (como routers o cámaras) ofrecen interfaces web a través de HTTP.
3. Al analizar tráfico HTTPS en Wireshark, solo veremos la capa TLS, no el contenido. En cambio, si alguna comunicación ocurre en texto claro mediante HTTP, Wireshark la mostrará completa. Esto, además, representa un riesgo de seguridad si en esa comunicación se transmiten credenciales.

Flujo HTTP típico

Primero se establece una conexión TCP de tres pasos (SYN, SYN-ACK, ACK) hacia el puerto 80 del servidor. Luego, el cliente envía una petición HTTP que comienza con una línea como `GET /recurso HTTP/1.1`, acompañada de encabezados como `Host`, `User-Agent`,

entre otros. El servidor responde con un código de estado (por ejemplo, «200 OK» si todo está correcto, o «404 Not Found» si el recurso no existe), seguido de los datos, como el HTML de una página o el archivo solicitado.

Wireshark muestra estos encabezados y datos en la pestaña de detalles bajo la capa HTTP, y en la columna «Info» suele ofrecer un resumen. Por ejemplo, en un paquete de respuesta, «Info» podría mostrar «HTTP/1.1 200 OK (text/html)», indicando un código 200 y el tipo de contenido.

Se puede filtrar el tráfico HTTP con `http`, o de forma más específica con `http.request` (solo peticiones), `http.response` (respuestas) o por métodos: `http.request.method == "POST"`. En el ejemplo de la figura 3, se filtraron las peticiones y se observaron varios GET. Si se hubiera filtrado con `http.response`, se habrían mostrado las respuestas con sus respectivos códigos.

En el ámbito de las pymes, analizar tráfico HTTP puede servir para observar qué páginas están solicitando los usuarios o qué datos se envían y reciben en un servicio interno. Por ejemplo, si se sospecha que una aplicación web transmite contraseñas sin cifrar, se podría capturar su tráfico HTTP y buscar esas credenciales dentro del contenido (utilizando `contains` en el filtro).

HTTP como vector de ataque

Muchos ataques se manifiestan en el tráfico HTTP. Por ejemplo, un escaneo de vulnerabilidades web podría verse como decenas de peticiones GET a archivos comunes (login.php, admin.html, etc.); un *malware* podría descargar una segunda etapa desde un URL; o un atacante podría utilizar HTTP para extraer datos (exfiltración). Wireshark no «detecta» estas amenazas por sí mismo (no es un sistema de detección de intrusos con firmas), pero proporciona la visibilidad necesaria para identificarlas manualmente.

Una estrategia posible es aplicar filtros orientados a ciertos patrones, como buscar cadenas sospechosas en la columna «Info». Además, el panel «Expert Info» de Wireshark clasifica algunos códigos HTTP 4xx y 5xx como «Note» o «Warn» en determinados contextos (por ejemplo, un «404 Not Found» es común pero destacable; un «500 Internal Server Error» puede indicar un problema más serio).

En síntesis, DNS y HTTP son dos de los protocolos más frecuentes que se observarán. DNS convierte nombres en direcciones IP y suele preceder a otras conexiones; HTTP, al no estar cifrado, aún aparece en entornos pequeños o como paso inicial de ciertas comunicaciones antes de redirigir a HTTPS. Con Wireshark es posible diagnosticar problemas (como un DNS que no responde o errores en HTTP), así como identificar comportamientos (qué dominios se consultan, a qué servidores web se conecta la red).

En una pyme, este análisis puede revelar, por ejemplo, si un equipo está haciendo solicitudes inusuales (¿accediendo a sitios no relacionados con el trabajo?) o si algún servicio interno intenta conectarse a servidores externos mediante HTTP.

Una recomendación práctica es utilizar filtros combinados. Por ejemplo, para analizar la experiencia de navegación de un usuario, podríamos aplicar el filtro `ip.addr == IP_del_usuario && (dns || http || tls)`. Esto mostraría toda la secuencia de ese usuario: la resolución de nombres, las conexiones por HTTP o TLS, y el tráfico asociado.

Veríamos, por ejemplo: una consulta DNS, su respectiva respuesta, luego posiblemente un `Client Hello` de TLS y, a continuación, el tráfico TLS cifrado. Esa secuencia confirmaría que la navegación fue exitosa.

En cambio, si se observa una consulta DNS sin respuesta (o repetida), eso sugiere un posible problema en la resolución de nombres. Si el *handshake* TLS no se completa (por ejemplo, si aparece un `RST`), podría tratarse de un inconveniente con los certificados o con el protocolo.

TLS y *handshake*

Hoy en día, gran parte de las comunicaciones se realiza de forma segura mediante TLS (*Transport Layer Security*), sucesor de SSL. Este protocolo se utiliza en HTTPS (navegación web segura), muchos servicios de correo (IMAPS, SMTPS), VPN, entre otros. Comprender a grandes rasgos el proceso de *handshake* TLS nos permite diagnosticar conexiones cifradas de manera más efectiva.

Cuando se observa tráfico TLS en Wireshark, suele aparecer identificado como protocolo TLSv1.2 o TLSv1.3 (según la versión), aunque en algunas versiones antiguas podría seguir figurando como «SSL». Lo primero que se produce es un intercambio inicial de *handshake* entre cliente y servidor, compuesto por varios subtipos de paquetes «Handshake Protocol».

El flujo básico de TLS 1.2 es el siguiente:

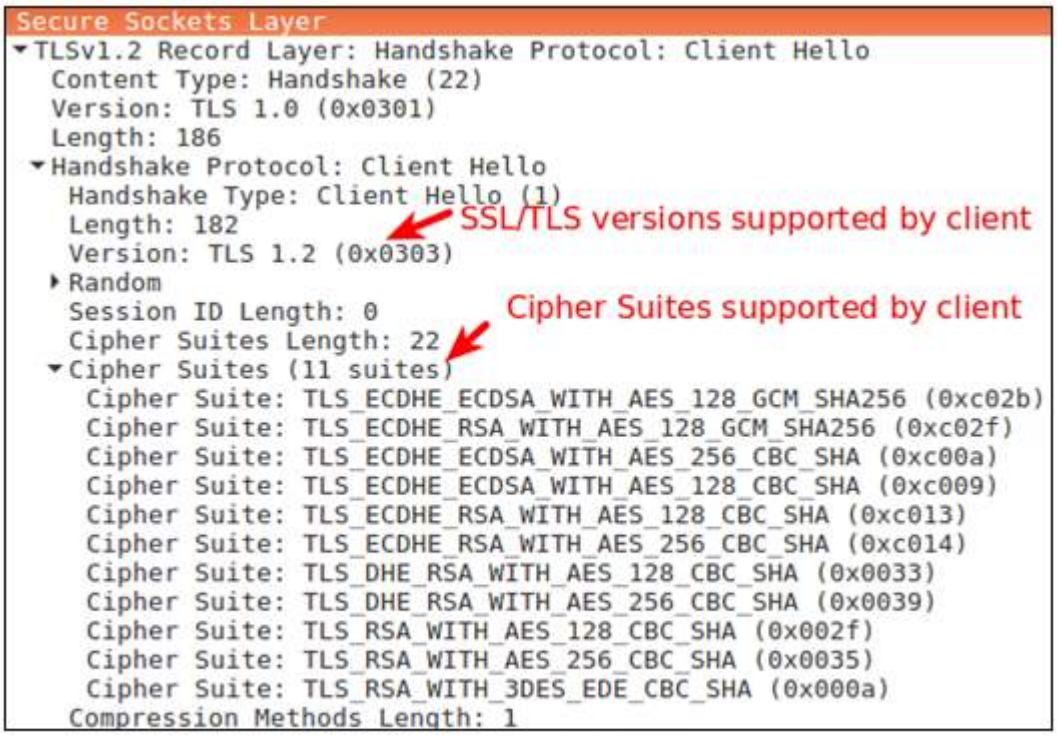
1. Client Hello

El cliente inicia el proceso con un mensaje en el que indica que desea establecer una sesión TLS. Incluye la versión máxima de TLS que soporta, una lista de *cipher suites* (algoritmos de cifrado) compatibles y algunos datos aleatorios.

En Wireshark, dentro del mensaje «Client Hello» se pueden ver detalles como la versión TLS propuesta, la longitud de la lista de cifrados y los cifrados específicos ofrecidos (por ejemplo, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, entre otros).

En la siguiente imagen se observa que el cliente ofrece la versión TLS 1.2 (0x0303) junto con una serie de *cipher suites* (en este caso, 11 opciones) que declara como compatibles. Las flechas rojas agregadas señalan estos campos clave: la versión del protocolo y los cifrados ofrecidos por el cliente.

Figura 8. Detalle de un paquete TLS Client Hello en Wireshark (vista expandida en el panel de detalles)



Fuente: Harshkapadia, 2023, https://short.do/b6Hah4

2. Server Hello

El servidor responde eligiendo la versión de TLS y el *cipher suite* que se utilizará (normalmente, el mejor que ambas partes soportan). También envía un número aleatorio y, en algunos casos, un identificador de sesión. Justo después, el servidor transmite su certificado público (*X.509*) en el paquete «Certificate», y dependiendo del *cipher suite* seleccionado, puede incluir un paquete «Server Key Exchange». Finalmente, envía un «Server Hello Done» para indicar que ha finalizado su parte del *handshake*.

En Wireshark, se pueden ver los paquetes «Certificate», donde es posible inspeccionar el certificado del servidor: nombre común (*common name*), emisor, período de validez, entre otros. La figura 9 muestra un ejemplo de este tipo de paquete.

Como observamos, Wireshark indica el tamaño total de los certificados transmitidos (3054 *bytes* en este caso) y permite visualizar la cadena completa. Por ejemplo, en este caso se observa que el certificado incluye un *Common Name* que contiene «cloudfaressl.com» y que fue emitido por «COMODO CA». Este es el certificado público del servidor.

Figura 9. Paquete TLS «Certificate» enviado por el servidor durante el *handshake*

Fuente: Harshkapadia, 2023, https://short.do/b6Hah4

3. Intercambio de claves —

En TLS 1.2, típicamente el cliente envía un «Client Key Exchange», que contiene la información necesaria (por ejemplo, una clave *premaster* RSA cifrada con la clave pública del servidor, o parámetros de Diffie-Hellman) para acordar la clave simétrica final. Luego, tanto el cliente como el servidor envían un mensaje «Change Cipher Spec» para indicar que, a partir de ese punto, se utilizará el cifrado acordado. A continuación, se transmite un mensaje «Finished» cifrado, que actúa como una verificación del *handshake*: es un resumen en forma de *hash* de toda la negociación previa, lo que permite detectar si hubo alteraciones.

En TLS 1.3, el proceso se simplifica: hay menos rondas y el intercambio de claves ocurre más temprano, aunque se mantiene el concepto general de *Hello* e intercambio de claves.

Para un operador de ciberseguridad, ¿qué resulta relevante del *handshake*? Entre los aspectos más importantes se encuentran los siguientes:

- **Verificar la versión TLS utilizada.** Por políticas internas, puede que la empresa no permita TLS 1.0 o 1.1 por considerarlas inseguras. Si al capturar observamos esas versiones en los mensajes «Client Hello» o «Server Hello», sería un hallazgo. Wireshark incluso puede advertirnos si se emplea un cifrado débil (aunque TLS 1.0 completo ya se considera débil).
- **Inspeccionar el certificado presentado por el servidor.** Si se sospecha de un sitio web falso, podemos capturar la conexión y revisar el *Common Name* del certificado y la entidad emisora. Por ejemplo, si accedemos a «banco.com» pero el certificado es autofirmado o emitido por una autoridad desconocida, esto llamará la atención.
- **Diagnóstico de problemas.** Si ocurre un fallo en la negociación TLS, es posible ver un mensaje «Alert» (por ejemplo, «Handshake Failure»). Wireshark lo resaltará en rojo como error. Esto podría indicar incompatibilidades de cifrado, certificados vencidos, etc. Un caso típico es cuando el servidor no confía en el cliente y responde con un «Alert» de nivel *fatal* tras el «Client Hello».
- **Medir la latencia del *handshake*.** El establecimiento de una sesión TLS introduce cierta demora (una ida y vuelta inicial). En

> redes de baja latencia, normalmente no representa un problema, pero Wireshark permite medir ese retraso. Por ejemplo, podemos calcular el RTT (tiempo de ida y vuelta) entre el «Client Hello» y el «Server Hello». En la pestaña «Statistics → Summary» a veces se muestra esta duración, o también puede deducirse observando las marcas de tiempo de esos paquetes.

La mayoría del tráfico TLS en Wireshark, una vez completado el *handshake*, aparece simplemente como «Application Data» cifrado. No es posible ver su contenido —lo cual es positivo desde el punto de vista de la privacidad—, pero sí se puede acceder a metadatos como direcciones IP, puertos, tamaños de paquetes, tiempos y el SNI (Server Name Indication), que a veces es visible en el «Client Hello». Este último campo indica el nombre del servidor al que se intenta conectar y puede ofrecer pistas valiosas: por ejemplo, si un equipo realiza múltiples conexiones TLS hacia un mismo dominio —y ese dominio es malicioso conocido—, esto podría representar un Indicador de Compromiso (IoC), siempre que el SNI no esté cifrado mediante *ESNI* (Encrypted SNI).

Desde una perspectiva de seguridad defensiva en pymes, verificar que los handshakes TLS se completen correctamente y sin demoras puede formar parte de los controles mínimos de rendimiento y seguridad. En algunos ataques, un adversario puede intentar forzar el uso de versiones antiguas como TLS 1.0 (downgrade attack). Si

esto ocurre, podríamos detectarlo fácilmente observando la versión negociada en Wireshark, aunque hoy TLS incluye mecanismos de protección contra ese tipo de ataques.

En resumen, Wireshark nos permite visualizar el «apretón de manos» que da inicio a una conexión segura. Aunque el contenido posterior esté cifrado, comprender el inicio del flujo TLS nos ayuda a verificar el nivel de seguridad empleado y a diagnosticar fallos de conexión. En el contexto de una pyme, quizá no se analicen *handshakes* a diario, pero es importante saber que si únicamente vemos tráfico TLS sin HTTP, eso indica que la comunicación está cifrada —lo que puede ser un alivio desde el punto de vista de la confidencialidad—. También, si un servicio no responde, capturar el *handshake* podría revelar un error de certificado o de negociación.

Finalmente, Wireshark incluye una sección en «Statistics → TLS» donde se resumen aspectos clave como los cipher suites utilizados en cada sesión, lo cual resulta útil para reportar la solidez de las conexiones observadas en la red.

Trazas y latencia

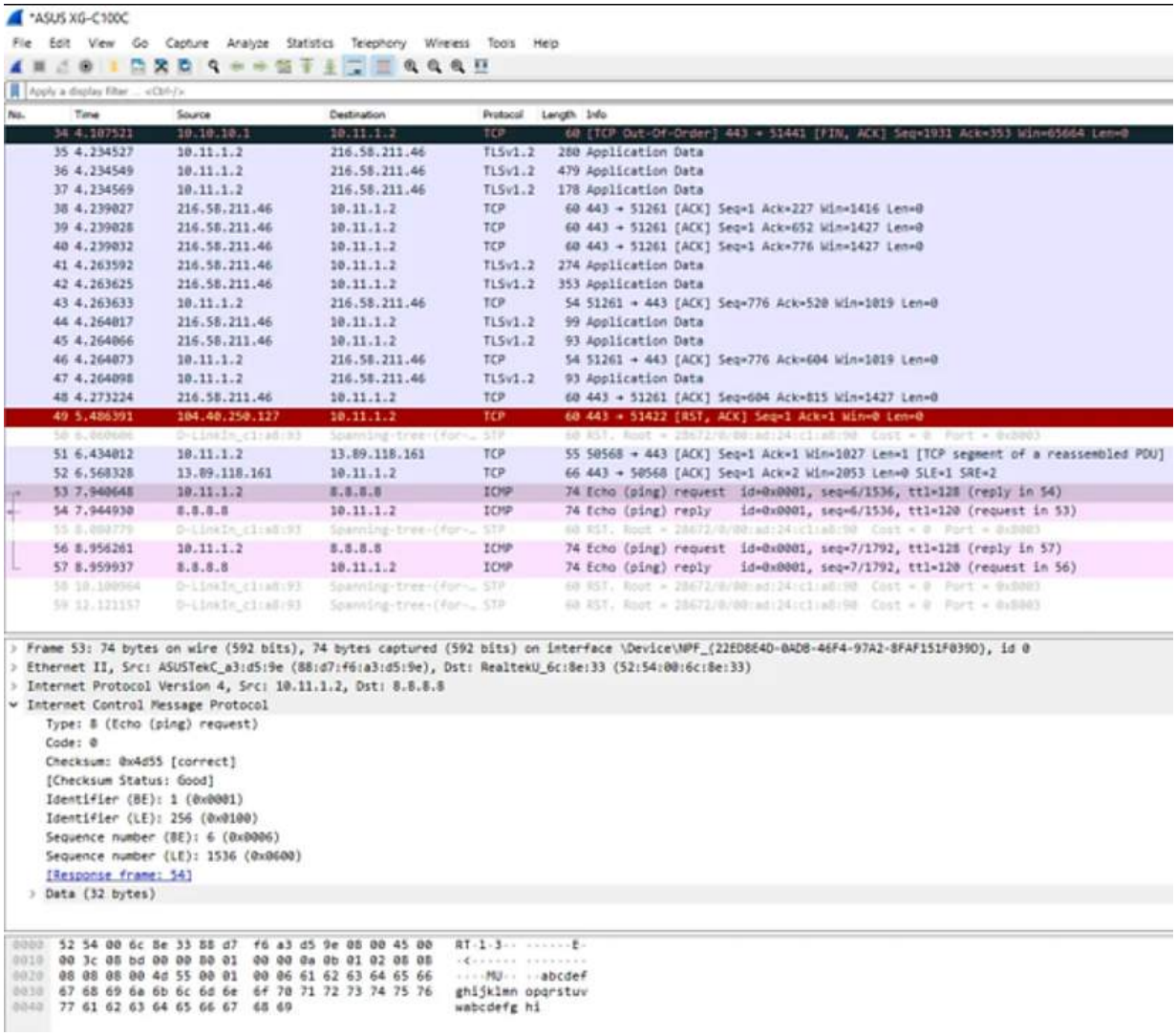
El término «traza» en redes a veces se refiere al resultado de herramientas como *traceroute* (que muestra los saltos intermedios hacia un destino) o a la propia captura de paquetes (traza de paquetes). Aquí abordaremos cpo analizar las rutas y tiempos de respuesta en la red: básicamente, cómo identificar latencia (demoras) y su posible origen mediante Wireshark u otras utilidades, sin necesidad de infraestructura compleja.

Latencia es el tiempo que tarda un paquete en ir y venir (RTT, *round-trip time*). En Wireshark, cada paquete tiene un *timestamp*, y podemos calcular diferencias. Si queremos saber cuánto tardó una respuesta DNS, por ejemplo, podemos mirar el tiempo del *query* vs. el *response*. Wireshark incluso tiene una función: clic derecho en el paquete de consulta DNS, opción «Set/Unset Time Reference» para marcarlo como t_0, luego mirar el *timestamp* relativo del *response*. También existen estadísticas: en «Statistics → Service Response Time → DNS» Wireshark puede calcular automáticamente tiempos promedio de respuesta DNS, HTTP, etc.

Para diagnosticar latencia en la red local vs. Internet, a menudo usamos *ping* (hace solicitudes ICMP Echo y mide

respuesta). Wireshark, capturando ICMP, nos permite ver esos *echo request/reply* con sus tiempos. En la figura 10 se ilustra un *ping*.

Figura 10. Captura de paquetes ICMP (*ping*) hacia 8.8.8.8



Fuente: Red Zone, 2025, https://short.do/G9hNl2

Cada par *request-reply* de la figura 10 corresponde a un ping. Si habilitamos en Wireshark la columna «Time» en modo delta (tiempo desde el paquete anterior), o simplemente calculamos, podemos obtener los milisegundos que tardó cada respuesta. Sin embargo, para medir latencia suele ser más sencillo usar las herramientas del sistema (por ejemplo, *ping* ya lo calcula). El valor añadido de Wireshark es que nos deja ver dónde puede estar el retraso si hay múltiples pasos.

Ahí entra *traceroute*: esta herramienta envía paquetes con TTL crecientes para provocar respuestas «TTL exceeded» de routers intermedios, revelando la ruta. Si capturamos mientras hacemos un *traceroute*, veremos muchos mensajes ICMP «Time to live exceeded» de diversos *routers* (cada uno indicando un salto). Wireshark también puede ordenar por TTL en la vista de detalles IP. Además, muchos *traceroute* utilizan UDP o ICMP para destapar la ruta hacia un destino. Podemos filtrar este tipo de tráfico en Wireshark utilizando expresiones como `icmp` o filtrando por los puertos UDP altos que emplea la herramienta. Además, en la vista «Expert Info», Wireshark puede resaltar paquetes con TTL muy bajos (por ejemplo, TTL=1) con un color distintivo, marcándolos como una nota. Esto indica que son paquetes con una vida útil muy limitada, característicos del funcionamiento de *traceroute*.

En el caso de una pyme, *traceroute* resulta útil si se sospecha que la lentitud proviene del exterior (como del proveedor de servicios de Internet o de la propia Internet). Por ejemplo, si la conexión a un servidor remoto es lenta, un *traceroute* puede mostrar en qué salto se incrementa la latencia. Wireshark puede utilizarse en conjunto con *traceroute*: iniciamos un *traceroute* hacia servidor.com y, al mismo tiempo, capturamos tráfico con Wireshark aplicando un filtro como `icmp or udp.port == 33434` (puerto inicial usado por *traceroute* con UDP). Así podemos almacenar toda la secuencia de saltos (*hops*). Luego, al analizar, podremos ver el salto 1 (generalmente el router local) con un tiempo de respuesta X ms, el salto 2 (un nodo del ISP) con Y ms, y así sucesivamente.

Incluso sin ejecutar un *traceroute* explícito, cualquier tráfico puede darnos pistas: el campo TTL (Time To Live) de las cabeceras IP entrantes sugiere cuántos saltos ha atravesado un paquete (aunque esto depende del valor inicial asignado por el sistema operativo). Por ejemplo, si en una respuesta de *ping* observamos TTL=57 y sabemos que el sistema de destino inicializa en 64, podríamos estimar que hay unos 7 saltos.

Wireshark también incluye herramientas de análisis de flujo TCP en el menú «Statistics» → «TCP Stream Graphs», como Round Trip Time Graph o Time-Sequence Graph. Por ejemplo, el gráfico de tiempo de ida y vuelta traza el RTT estimado de cada segmento (basado en los ACKs), lo cual permite identificar si la latencia es constante o si presenta variaciones.

Cuando hay pérdida de paquetes, Wireshark —a través de «Expert Info»— marcará eventos como retransmisiones o «DUP ACKs», que suelen indicar problemas posiblemente causados por latencia o congestión. Si detectamos muchos "TCP Retransmission" en la captura, eso sugiere que los paquetes no están llegando a destino a tiempo y se reenvían, lo cual es un síntoma de una red inestable o saturada. En «Expert Info» veríamos líneas en la sección «Sequence» (en amarillo) enumerando las retransmisiones.

Con Wireshark podemos visualizar interrupciones de sesiones, retransmisiones, pérdida de paquetes o latencia excesiva entre saltos, para así distinguir si el origen del problema está en el cliente, en el servidor o en algún punto intermedio. En una red pequeña, ese punto intermedio probablemente sea el router o el proveedor de servicios de Internet. Por ejemplo:

- si la captura hecha cerca del cliente muestra que este ni siquiera envía la petición, es un problema en el cliente.

- si el cliente envía la petición y el servidor responde rápido, pero la respuesta se ve demorada llegando al cliente, quizás la red local tiene un cuello de botella;

- si la respuesta tarda mucho en venir del servidor (tiempo largo entre solicitud y respuesta, medido en Wireshark), el problema puede estar en el servidor o en su red.

Ejemplo práctico

Supongamos que empleados se quejan de que el sistema CRM web (en la nube) está lento. Capturamos tráfico hacia el dominio del CRM. Vemos: el DNS responde rápido con la IP, la conexión TLS se establece en, digamos, 50 ms (*handshake*), pero luego cada petición «POST» tarda 2 segundos en recibir respuesta. Viendo los *timestamps*, confirmamos que el servidor CRM se toma aproximadamente 2 segundos en responder. La red (*traceroute*) muestra latencias menores a 100 ms. Entonces,

concluimos que la latencia se debe al procesamiento del servidor, no a la red. En cambio, si viéramos que el *handshake* tarda 1 segundo y cada paquete presenta enormes lapsos entre sí, podríamos sospechar de una red saturada o con pérdida (y ver retransmisiones que lo confirmen). Así, la visibilidad temporal de Wireshark nos ayuda a localizar el segmento problemático.

En el caso de las pymes, a veces se detectan problemas como «la copia de archivos al servidor es lenta». Al capturar el tráfico FTP o SMB, quizá se observe un patrón de *window size* reduciéndose y muchas esperas, lo que indica congestión. O bien, si «la VPN se cae», tal vez se detecte un ICMP «Fragmentation needed» no atendido o una alerta TLS.

Para finalizar, conviene destacar que Wireshark complementa, pero no reemplaza, las herramientas especializadas de monitoreo continuo de latencia. Sin embargo, para un análisis puntual y detallado, resulta muy útil: permite ver, desde la capa física hasta la de aplicación, todo en uno, y con ello diagnosticar problemas de rendimiento y rutas ineficientes.

Detección de anomalías

Llegamos a una cuestión relevante: **¿cómo podemos identificar comportamientos anómalos o maliciosos en la red usando Wireshark?** Si ya tenemos una idea clara de lo «normal» —patrones DNS típicos, volumen de HTTP razonable, tiempos de respuesta habituales—, cualquier desviación significativa puede ser una anomalía que valga la pena analizar. A continuación, presentamos algunas situaciones comunes en pymes:

Comunicaciones repetitivas o inusuales —

Por ejemplo, un equipo que normalmente realiza unas decenas de consultas DNS por hora, de pronto ejecuta miles de consultas por minuto a dominios con nombres aleatorios (ejemplo: ajdkslfjsdlfj.attackersite.com). Esto podría indicar la presencia de un túnel DNS —el *malware* encapsula datos en consultas DNS— o algún tipo de *beaconing* encubierto.

Con Wireshark, podemos identificar ese patrón: al filtrar por la IP del equipo y por DNS, se mostraría la secuencia anómala. También en «Expert Info» podríamos ver numerosos «Standard query response no such name», si el *malware* intenta acceder a subdominios inexistentes.

Comportamientos de alta frecuencia o direccionales —es decir, siempre hacia un mismo *host*— constituyen una señal de alerta. Un patrón sospechoso puede consistir en múltiples conexiones salientes a puertos no estándar desde una computadora interna; esto podría indicar un escaneo o la actividad de un *bot*.

Escaneos de puertos o conexiones no autorizadas —

Si alguien lanza un *port scan* en la red, Wireshark podría revelarlo como múltiples intentos de conexión desde una IP de origen hacia varios destinos o puertos secuenciales. Por ejemplo, veríamos muchos paquetes SYN sin completar. En el campo «Info» podrían listarse como SYN repetidos, y quizás también aparecerían *flags* RST posteriormente. Además, Wireshark colorea por defecto los SYN (azul oscuro) y los RST (rojo) en su esquema, por lo que el patrón se destacaría visualmente. Estas conexiones repetitivas se mencionan en la bibliografía como indicios de actividad sospechosa.

ARP spoofing / duplicados —

En una LAN pequeña, un atacante interno podría intentar un *ARP spoof* para interceptar tráfico. Un síntoma de esto es que Wireshark, con el filtro «arp», podría mostrar dos dispositivos reclamando la misma IP (respuestas ARP duplicadas). De hecho, Wireshark cuenta con una detección interna denominada «Duplicate IP address configured», que se marca en «Expert Info» con nivel de advertencia. Si en la captura aparecen paquetes ARP en los que dos direcciones MAC diferentes responden a la misma IP, se trata de un

comportamiento anómalo. Asimismo, muchas peticiones ARP sin respuesta también pueden indicar problemas.

Errores recurrentes

No siempre una anomalía indica un ataque; también puede tratarse de un error de configuración. Por ejemplo, una máscara de subred incorrecta. En «Expert Info» podrían observarse mensajes ICMP «Destination unreachable» de forma reiterada, o paquetes TCP con RST continuo cuando un *host* intenta conectarse a un destino inexistente. Wireshark también permite identificar errores de protocolo: por ejemplo, si un servidor DHCP está mal configurado, veremos múltiples solicitudes DHCP Discover sin respuesta, lo cual resulta anómalo desde el punto de vista del funcionamiento normal de la red.

Tráfico fuera de horario o voluminoso

En el contexto de las pymes, si a las 3 de la mañana un equipo comienza a enviar gigabytes de datos a una IP externa, se trata de un comportamiento anómalo. Con Wireshark, podemos filtrar por la IP en cuestión y analizar el tipo de tráfico involucrado: ¿es FTP?, ¿es HTTP con método PUT?, ¿es una serie de segmentos TLS? El patrón temporal también resulta clave: Wireshark permite ordenar los paquetes por tiempo y detectar, por ejemplo, que una computadora inició una conexión justo después de un evento, como un inicio de sesión. Los gráficos de entrada/salida («Statistics» → «I/O Graph») pueden evidenciar picos inusuales de tráfico.

Uso de protocolos inseguros inesperados

Si la política de red establece que todas las comunicaciones deben estar cifradas mediante TLS, la detección de tráfico en texto plano —como HTTP, FTP no seguro o Telnet— constituye una anomalía. Podemos aplicar un filtro http sobre toda la captura; cualquier resultado podría indicar un uso inapropiado, por ejemplo, un usuario descargando contenido mediante HTTP cuando debería utilizar HTTPS. Asimismo, la presencia de protocolos obsoletos, como SMBv1 o SSLv3, es relevante. Wireshark resalta algunas de estas condiciones: un SMBv1, por ejemplo, podría aparecer como NBSS, y al saber que está en desuso, corresponde marcarlo como comportamiento no deseado.

Indicadores específicos de ataques

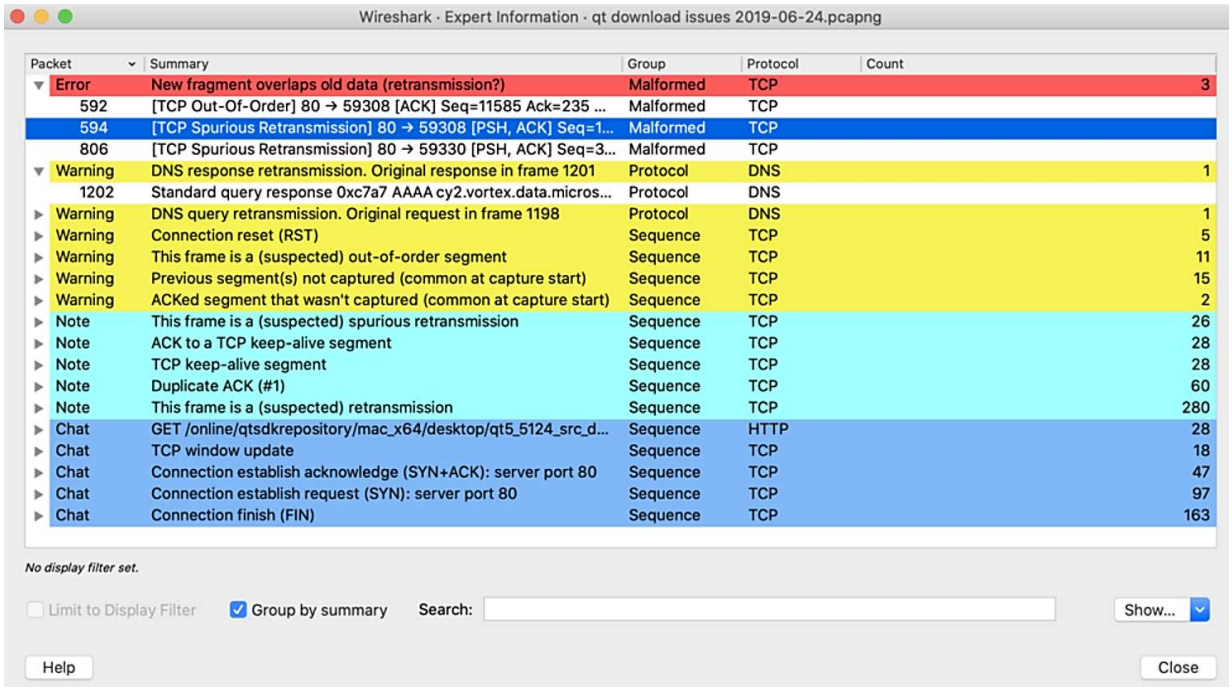
Existen ciertos patrones ampliamente reconocidos. Por ejemplo, en casos de *DNS tunneling*, las consultas DNS suelen contener cadenas codificadas en los subdominios y un dominio final constante. Wireshark no identificará explícitamente que se trata de *DNS tunneling*, pero si observamos nombres de dominio muy largos, inusuales y con alta frecuencia, junto con numerosos registros `TXT` o `CNAME`, podemos sospechar esta técnica.

Otro caso es el SMB Ghost scanning, donde se evidencian múltiples negociaciones SMB anómalas. Un ejemplo más sencillo: cierto *malware* del tipo *ransomware* realiza pings constantes a un

controlador de dominio antes de iniciar el ataque; esto puede reflejarse como tráfico ICMP inusual. Asimismo, el tráfico ICMP con contenido no típico (*payload*) puede ser indicio de un canal encubierto (*covert channel*).

Para ilustrar, la figura 11 muestra el panel «Expert Information» de Wireshark, que resume distintos eventos detectados en una captura problemática:

Figura 11. Pantalla «Expert Information» de Wireshark, con elementos notables agrupados por nivel de severidad



Fuente: Wireshark, s.f., https://short.do/JrQj1s

En rojo («Error») se observan anomalías TCP graves —fragmentos fuera de orden que se sobreponen—; en amarillo («Warning»), retransmisiones DNS y mensajes TCP inusuales; en cian («Note»), eventos como ACK duplicados o retransmisiones sospechosas; y en azul («Chat»), eventos normales como el establecimiento y la finalización de conexiones. Esta vista general nos ayuda a identificar rápidamente posibles anomalías sin necesidad de revisar paquete por paquete.

La figura resulta reveladora: por ejemplo, muestra una línea «DNS response retransmission. Original response in frame 1201» marcada como «Warning», lo que indica que un servidor DNS envió la misma respuesta dos veces —quizá porque el cliente no la recibió a tiempo y volvió a consultarla—. También se observan múltiples «TCP spurious retransmission» y «Previous segment not captured», lo que sugiere pérdidas o una captura incompleta. Ante tanto color amarillo y rojo, un analista podría inferir que hay un problema en esa comunicación, como congestión o incluso un posible ataque. En cambio, las líneas azules

correspondientes a paquetes SYN o FIN son normales y esperables.

La herramienta «Expert Info» de Wireshark es muy útil para una detección inicial de irregularidades. Sin embargo, como indica su documentación, se trata solo de un punto de partida: luego debemos investigar cada caso en profundidad.

Recapitulando, detectar anomalías requiere primero establecer una línea base del comportamiento normal y, a partir de allí, utilizar las funciones de Wireshark para destacar lo que se aparta de ese patrón. No se trata de un IDS que emite alertas automáticamente —aunque existen integraciones posibles—, pero con búsquedas adecuadas podemos identificar desde intrusiones hasta errores de configuración.

A continuación, enumeramos algunos indicadores concretos que deberíamos considerar al utilizar Wireshark en tareas de ciberseguridad:

- **Conexiones a dominios o IP maliciosas conocidas**

Si contamos con listas de referencia, podemos aplicar filtros para detectar coincidencias (aunque Wireshark no realiza análisis de reputación por sí mismo). Alternativamente, es posible exportar las IP con las que se ha comunicado la red y verificar su presencia en *feeds* de indicadores de compromiso (*IOC*).

- **Exfiltración de datos**

Tráfico de salida inusualmente voluminoso, fuera de horario o mediante protocolos no habituales puede ser señal de exfiltración. Por ejemplo, si una computadora del área contable sube 500 MB a un servidor FTP en Rusia, resulta sospechoso. Filtros por tamaño —como `tcp.len > 0` junto con herramientas de estadísticas— permiten identificar estos volúmenes.

- **Suplantación o ataques de intermediario (MITM).** Además del ARP duplicado ya mencionado, es importante estar atentos a certificados extraños en sesiones TLS, lo cual puede indicar un proxy no autorizado o un

ataque. Si durante un *TLS handshake* se detecta un certificado emitido por una autoridad desconocida para un sitio popular, se trata de una señal de alerta.

Cabe mencionar que existen *scripts* y herramientas de análisis de archivos PCAP —como Tshark, PyShark e incluso enfoques basados en *machine learning*— que permiten automatizar la búsqueda de anomalías, aunque ese tipo de soluciones excede los objetivos de este módulo. Con los conocimientos adquiridos, un operador en una pyme puede lograr un análisis bastante completo de forma manual.

Por último, es recomendable documentar cada hallazgo y correlacionarlo con registros de otras fuentes. Wireshark proporciona una visión directa del tráfico, pero en algunos casos conviene contrastar la información con los registros del *firewall*, un SIEM u otras herramientas, para confirmar si una anomalía en la red corresponde a un evento legítimo. Por ejemplo, un volumen elevado de tráfico HTTP hacia un dominio determinado puede coincidir con una actualización en segundo plano autorizada.

Conclusión de la unidad 2

En esta unidad revisamos cómo identificar el tráfico legítimo de DNS, HTTP y TLS, y cómo utilizar ese conocimiento para establecer un punto de referencia. Luego exploramos de qué manera Wireshark puede ayudarnos a detectar señales de alerta en la red: desde retransmisiones y errores hasta patrones sospechosos que podrían indicar un ataque o un mal funcionamiento. En entornos pequeños, estas habilidades permiten que un operador junior de seguridad identifique incidentes incipientes o problemas de red antes de que escalen, y actúe con rapidez para contener una amenaza o corregir una configuración incorrecta.

A continuación, pondremos en práctica estos conocimientos en un laboratorio guiado y presentaremos un caso de estudio integrador.

Laboratorio guiado: analizando tráfico real con Wireshark

En este laboratorio realizaremos una captura de tráfico en un entorno controlado y aplicaremos los filtros y técnicas aprendidos para extraer información útil. No se requiere infraestructura costosa, solo una computadora con Wireshark instalado y conexión a Internet o a una red local de pruebas. Los pasos a seguir son los siguientes:

1. Preparación del entorno

Si aún no lo has hecho, descarga e instala Wireshark en tu computadora (está disponible para Windows, Linux o macOS desde su sitio web oficial). Cierra las aplicaciones que puedan generar mucho tráfico de fondo —como navegadores con múltiples pestañas abiertas o procesos de actualización— para que la captura sea lo más limpia posible. Asegúrate de tener privilegios de administrador, ya que Wireshark puede requerirlos para iniciar la captura.

Gaining Insight

Abre Wireshark y selecciona la interfaz de red que esté conectada a Internet (por ejemplo, Ethernet o wifi). Haz doble clic sobre ella para comenzar la captura. Deberías ver cómo empiezan a aparecer paquetes en la lista —tráfico rutinario de la red, ARP, alguna consulta DNS, entre otros—.

Luego, genera de forma intencional algo de tráfico de prueba: por ejemplo, abre el navegador web e ingresa la URL http://neverssl.com (un sitio diseñado para responder por HTTP sin cifrado). También puedes abrir una terminal o símbolo del sistema y hacer *ping* a 8.8.8.8 (servidor DNS público de Google, útil para observar tráfico ICMP). Si tienes más tiempo, intenta realizar una consulta DNS manual con `nslookup` o `dig` sobre un dominio de tu elección. Después de aproximadamente un minuto, vuelve a Wireshark y presiona el botón rojo «Stop» para detener la captura.

—

Con la captura detenida, desplázate por la lista de paquetes. Deberías ver lo siguiente:

- Tráfico ARP con mensajes del tipo «Who has …? Tell …», generado al hacer *ping*, probablemente para resolver la dirección MAC del *gateway*.
- Paquetes ICMP Echo Request y Echo Reply.
- Una secuencia de paquetes TCP hacia neverssl.com (puerto 80) y, seguidamente, paquetes HTTP. Es posible que el sitio haya redirigido a otro destino; si hubo una redirección a HTTPS, tal vez veas primero un mensaje «HTTP/1.1 301 Moved Permanently»

y luego un *handshake* TLS. Pero, para este ejercicio, asumimos que visualizaste contenido HTTP en texto plano.

- Consultas DNS generadas por la petición web (por ejemplo, para resolver la IP de neverssl.com) y posiblemente otras consultas propias del sistema operativo (como las que realiza Windows a msftncsi.com, como se mostró en ejemplos anteriores).

Sin aplicar filtros aún, intenta identificar a simple vista la siguiente información:

- La dirección IP de neverssl.com (¿a qué IP se conectó tu computadora en el puerto 80?).
- El tiempo que tardó el *ping* (revisa la diferencia entre las marcas de tiempo del Echo Request y el Echo Reply, o bien observa el valor en milisegundos que aparece en la consola del *ping*).
- Si hay un paquete HTTP, haz clic sobre él y revisa el panel inferior: ¿qué código de respuesta devolvió el servidor? ¿Qué contenido HTML o texto envió? (Podría aparecer un mensaje como «You are not using SSL»).

4. Aplicar filtros básicos —

Ahora usa la barra de filtros para realizar los siguientes análisis:

- **Filtra `dns` y pulsa «Apply».** Deberías ver las consultas DNS. ¿Aparece una consulta para neverssl.com? ¿Qué respuesta se dio (qué dirección IP devolvió)?
- **Filtra `icmp`.** Ahora solo verás los paquetes del *ping*. Identifica el «Identifier» y el «Sequence number» en los detalles del protocolo ICMP. Son valores numéricos que permiten correlacionar las solicitudes («Request») con sus respuestas (Reply). Comprueba que coincidan.
- **Filtra `http`**. Wireshark mostrará únicamente los paquetes HTTP. Si la página redirigió a HTTPS, es posible que no aparezca mucho tráfico; en ese caso, filtra `tcp.port == 80` para visualizar todo el flujo que usó ese puerto. Si encuentras un paquete HTTP GET, selecciónalo y, en el panel de detalles, expande la sección «Hypertext Transfer Protocol». Podrás leer los encabezados que envió tu navegador (como *Host*, *User-Agent*, etc.). Luego, expande el paquete de respuesta HTTP y observa el *Status Code*. Anota cuál fue (por ejemplo, 200 OK, 301 Moved Permanently, etc.).
- **Experimenta con un filtro de visualización que combine condiciones.** Por ejemplo, `ip.addr == 8.8.8.8 or dns`. Este filtro debería mostrar cualquier comunicación con 8.8.8.8 (incluido el *ping*) o cualquier tráfico DNS. Es decir, dos tipos de tráfico

distintos al mismo tiempo. Deberías ver tanto el ICMP como las consultas DNS en una misma vista.

5. Seguir una sesión TCP

Ubica en la lista un paquete que sea parte de la conexión a neverssl.com (puedes filtrar con `ip.addr == [IP_de_neverssl]` para facilitar la búsqueda). Haz clic derecho sobre uno de esos paquetes y elige «Follow» → «TCP Stream». Se abrirá una ventana que muestra todo el flujo TCP reensamblado en orden. Lo que veas dependerá de la interacción, pero probablemente aparecerá la petición GET enviada por tu navegador y la respuesta del servidor, en texto legible. Esta vista concatenada es muy útil para analizar el intercambio completo.

En la parte inferior, donde dice «Entire conversation», cambia el desplegable a «ASCII» si prefieres ver solo texto plano. Puedes guardar el contenido con la opción «Save as...» si deseas conservarlo.

¿Aparecen en este flujo las palabras que viste en la página web? (Por ejemplo, si el sitio devolvió un mensaje como «OK» o algún fragmento de HTML).

6. Marcar y comentar (opcional)

Wireshark permite agregar comentarios a los paquetes. Para hacerlo, haz clic derecho sobre un paquete y selecciona la opción «Packet Comment». Puedes probar añadiendo un comentario al primer paquete HTTP de la captura, indicando: «Inicio de conversación HTTP». Este comentario queda almacenado dentro del archivo PCAP (si lo guardas) y estará visible para cualquier persona que abra ese archivo con Wireshark.

7. Exportar objetos (opcional)

Si obtuviste tráfico HTTP con contenido descargable —por ejemplo, un archivo PDF descargado mediante HTTP—, puedes ir a «File» → «Export Objects» → «HTTP» y ver los objetos listados. En este laboratorio, es posible que aparezca `ncsi.txt` (un archivo de prueba utilizado por Microsoft) o algún `favicon`. Si alguno de estos objetos está disponible, intenta exportarlo: selecciónalo, haz clic en «Save» y luego ábrelo con un editor de texto para verificar su contenido.

8. Finalizar y limpiar

Con esto concluye el laboratorio. Puedes guardar la captura para referencia futura, por ejemplo, con el nombre «lab1.pcapng». Si capturaste tráfico en la red real de tu empresa, no compartas este archivo públicamente, ya que puede contener información sensible sobre tu entorno de red. Si trabajaste en un entorno de pruebas, no hay problema en conservarlo o compartirlo con fines educativos.

Resultados esperados

Tras completar este laboratorio, deberías sentirte cómodo realizando una captura básica, aplicando filtros por protocolo y utilizando la función «Follow TCP Stream» para reconstruir conversaciones. Habrás identificado los siguientes patrones:

- La consulta DNS previa a una conexión HTTP.
- El *handshake* TCP (SYN, SYN/ACK, ACK), visible en la lista de paquetes.
- La secuencia de solicitud y respuesta HTTP.
- El *ping* ICMP y su correspondiente respuesta.

Preguntas de reflexión

- ¿Qué diferencias notas entre una comunicación en texto plano (HTTP) y una cifrada (si probaste con HTTPS)? ***Tip***: en HTTPS no se puede ver el contenido, solo el *handshake* TLS.
- ¿Cómo podrías distinguir en Wireshark un tráfico normal de navegación web de un posible tráfico de *malware* encubierto? Piensa en el volumen, los destinos o los horarios inusuales.

Si un usuario reporta que «Internet no funciona», ¿qué pasos seguirías con Wireshark para determinar si se trata de un problema de DNS, de puerta de enlace, o de otro tipo?

Este laboratorio desarrolló un caso sencillo. En situaciones reales, las capturas pueden ser mucho más extensas y complejas. La clave está en comenzar con filtros amplios e ir afinando progresivamente. También es importante apoyarte en las funcionalidades vistas —como los esquemas de colores, los perfiles personalizados si los configuraste, y la herramienta «Expert Info»— para no perder el rumbo en el análisis. Con la práctica, muchos patrones se volverán familiares y las anomalías serán más fáciles de identificar.

Conclusiones

En este módulo desarrollamos las bases necesarias para obtener visibilidad de red en entornos de pymes utilizando herramientas accesibles. Comenzamos por dominar las funciones esenciales de Wireshark: captura de tráfico, aplicación de filtros, personalización de vistas y exportación de datos relevantes. Luego, aplicamos ese conocimiento para comprender los patrones típicos de tráfico —DNS, HTTP, TLS,

entre otros— y, a partir de ellos, detectar desviaciones que podrían señalar problemas de rendimiento o incidentes de seguridad.

Es importante destacar que la visibilidad, por sí sola, no detiene las amenazas, pero constituye el primer paso indispensable para poder reaccionar ante ellas. Con una formación adecuada, podemos diagnosticar rápidamente una amplia variedad de situaciones: desde una simple falla de configuración —como un DNS mal configurado— hasta una intrusión silenciosa, como la que exploramos en el caso de estudio. En ausencia de soluciones de monitoreo costosas, estas habilidades representan nuestra primera línea de defensa en una pequeña empresa.

Te recomendamos seguir practicando en tu propio entorno. Cada red tiene su personalidad, y conocer qué es lo normal en la tuya te permitirá desarrollar alertas mentales cuando algo fuera de lo común ocurra. Además, mantenerse actualizado es fundamental: los protocolos evolucionan —como la adopción de DNS sobre HTTPS o TLS 1.3—, al igual que las tácticas de ataque. Por eso, complementar este aprendizaje con la lectura de casos actuales y la participación en comunidades —foros, blogs, entre otros— te ayudará a afinar tu criterio como analista.

Finalmente, las referencias proporcionadas a lo largo del texto (ver sección siguiente) incluyen guías en español e inglés, documentación oficial de Wireshark y análisis técnicos que puedes consultar para profundizar en aspectos específicos. La ciberseguridad en redes es un campo vasto, pero al dominar herramientas como Wireshark ya cuentas con una ventaja fundamental: la capacidad de ver y comprender lo que ocurre bajo la superficie de los cables y señales de tu organización. Esa claridad es el cimiento sobre el cual se construyen redes más seguras y resilientes.

CONTINUAR

Referencias

Cisco, (2025). *Capture y analice el tráfico de red con Wireshark para diagnósticos.* https://www.cisco.com/c/es_mx/support/docs/security/umbrella/225250-capture-and-analyze-network-traffic.html

Harshkapadia, (2023). *TLS*. https://networking.harshkapadia.me/tls.html

Red Zone, (2025). *Cómo usar Wireshark para capturar y analizar el tráfico de red.* https://www.redeszone.net/tutoriales/redes-cable/wireshark-capturar-analizar-trafico-red/

Unit 42, (s.f.). *Tutorial de Wireshark: Exportación de objetos desde un PCAP.* https://unit42.paloaltonetworks.com/using-wireshark-exporting-objects-from-a-pcap/

Wireshark, (s.f.). *Información de expertos.* https://www.wireshark.org/docs/wsug_html_chunked/ChAdvExpert.html

Wireshark Foundation, (2025). *Wireshark* [analizador de paquetes].

Referencias bibliográficas de consulta

Campus de Seguridad, (s.f.). *Guía completa de Wireshark para análisis y monitoreo de redes.* https://www.campusciberseguridad.com/blog/guia-completa-de-wireshark-para-analisis-y-monitoreo-de-redes/

Echeverri, E. (2024). *Filtros útiles en Wireshark para administración y pentesting*. https://thehackerway.es/2024/02/26/filtros-utiles-en-wireshark-para-administracion-y-pentesting/

Torres, A. (2021). Network Traffic Analysis: ¿Qué es un archivo PCAP? *LinkedIn*. https://www.linkedin.com/pulse/network-traffic-analysis-qu%C3%A9-es-un-archivo-pcap-arturo-e-torres/

Wireshark, (s.f.). *Comprendiendo la pantalla inicial.* https://labex.io/es/tutorials/wireshark-explore-and-customize-wireshark-interface-415949

CONTINUAR