

Anexo. Uso de Wireshark

1. Introducción a Wireshark

1.1 ¿Qué es Wireshark?

Descripción general

Wireshark es una herramienta de análisis de red de código abierto que permite capturar y examinar en tiempo real el tráfico de datos que circula por una red.

Funcionalidad clave: Captura cada "paquete" de datos que viaja a través de la red y permite a los usuarios analizar el contenido de estos paquetes de forma detallada.

Usos de Wireshark

Resolución de problemas de red: Identificación de cuellos de botella, errores de red, y problemas de configuración.

Análisis de tráfico sospechoso: Monitorización de tráfico para detectar comportamientos inusuales que podrían indicar la presencia de malware, intentos de hacking, o accesos no autorizados.

Auditorías de seguridad: Revisión del tráfico de red para asegurar que no se están transmitiendo datos sensibles sin cifrar o detectar la fuga de datos.

Importancia en seguridad informática

Detección de intrusiones: Wireshark puede ayudar a identificar patrones de ataque en la red, como escaneos de puertos o intentos de explotación.

Análisis forense: Después de un incidente de seguridad, Wireshark permite revisar capturas de tráfico para entender cómo se produjo la intrusión y qué datos fueron comprometidos.

1.2 Entorno de Wireshark

Interfaz gráfica

Familiarización con la interfaz

Menú principal: Acceso a funciones de captura, análisis, estadísticas y configuración.

Barras de herramientas: Accesos directos a funciones comunes como iniciar/parar capturas, aplicar filtros y analizar tráfico.

Pantallas principales:

Lista de paquetes capturados: Muestra todos los paquetes capturados en tiempo real.

Detalles del paquete: Muestra la estructura del paquete seleccionado, dividido en diferentes capas del modelo OSI.

Datos en bruto: Muestra los datos del paquete en formato hexadecimal.

Ventana de captura

Explicación de las interfaces de red disponibles:

Al iniciar Wireshark, se presenta una lista de las interfaces de red disponibles en el sistema, como Ethernet, Wi-Fi, y adaptadores virtuales.

Selección de la interfaz correcta: Selecciona la interfaz que corresponde a la conexión de red activa (ej. Wi-Fi si estás conectado de forma inalámbrica).

Filtros de captura

Introducción básica a los filtros de captura y visualización:

Filtros de captura: Se configuran antes de iniciar la captura y determinan qué tipos de tráfico se capturan.

Filtros de visualización: Se aplican después de la captura para enfocar el análisis en tipos específicos de tráfico.

2. Configuración Inicial y Captura de Tráfico

2.1 Instalación de Wireshark

Pasos para instalar Wireshark en Windows y Linux:

Windows

1. Visita el sitio oficial de Wireshark (<https://www.wireshark.org/>) y descarga el instalador para Windows.
2. Ejecuta el instalador y sigue las instrucciones en pantalla.
3. Durante la instalación, se te ofrecerá la opción de instalar **WinPcap** o **Npcap** (controladores necesarios para la captura de tráfico en Windows). Asegúrate de instalar uno de ellos.
4. Finaliza la instalación y abre Wireshark.

Linux

1. Abre la terminal.
2. Dependiendo de la distribución, usa uno de los siguientes comandos:
 - **Debian/Ubuntu:** `sudo apt-get update && sudo apt-get install wireshark`
 - **Fedora:** `sudo dnf install wireshark`
 - **Arch Linux:** `sudo pacman -S wireshark-gtk`
3. Durante la instalación, se te preguntará si deseas que los usuarios no root puedan capturar tráfico. Responde sí si lo deseas.
4. Una vez instalado, abre Wireshark desde la terminal o el menú de aplicaciones.

Permisos necesarios:

- **Windows:** En Windows, es recomendable ejecutar Wireshark como administrador para asegurar que tenga acceso completo a todas las interfaces de red. Haz clic derecho en el icono de Wireshark y selecciona "Ejecutar como administrador".
- **Linux:** En Linux, si no diste permisos de captura a usuarios no root durante la instalación, tendrás que ejecutar Wireshark como superusuario utilizando sudo wireshark.

2.2 Captura de Tráfico

Selección de la interfaz de red:

Identificación de interfaces:

Una vez abierto Wireshark, en la pantalla principal verás una lista de todas las interfaces de red disponibles en tu sistema (Ethernet, Wi-Fi, adaptadores virtuales, etc.).

Cada interfaz mostrará un gráfico que indica la cantidad de tráfico en tiempo real, lo que puede ayudarte a identificar la interfaz activa.

Seleccionar la interfaz adecuada:

Para capturar tráfico de una red específica, selecciona la interfaz correspondiente. Por ejemplo:

Si estás conectado a una red Wi-Fi, selecciona la interfaz Wi-Fi.

Si estás conectado por cable, selecciona la interfaz Ethernet.

Iniciar y detener una captura:

Iniciar una captura:

Una vez seleccionada la interfaz, haz clic en el botón de la lupa con el icono de la red para iniciar la captura.

Wireshark comenzará a capturar todos los paquetes que pasan a través de la interfaz seleccionada, mostrando cada paquete en la ventana principal.

Detener una captura:

Para detener la captura, haz clic en el botón cuadrado rojo en la barra de herramientas.

Después de detener la captura, podrás examinar los paquetes capturados en detalle.

2.3 Análisis de Tráfico Capturado

Examinar los paquetes

Visualización de los detalles

Cada paquete capturado se muestra en una línea en la parte superior de la pantalla de Wireshark.

Haz clic en cualquier paquete para ver más detalles. La pantalla se divide en tres partes:

- **Lista de paquetes:** Muestra una vista general de todos los paquetes capturados.
- **Detalles del paquete:** Muestra información detallada del paquete seleccionado, desglosada en las capas del modelo OSI (físico, enlace de datos, red, transporte, aplicación).
- **Datos en bruto:** Muestra el contenido del paquete en formato hexadecimal.

Captura de tráfico HTTP

Ejemplo práctico

Captura de tráfico HTTP: Abre un navegador web y visita un sitio web no cifrado (HTTP). Simultáneamente, inicia una captura en Wireshark.

Análisis: Después de detener la captura, filtra el tráfico capturado utilizando http en el campo de filtros. Esto mostrará solo el tráfico HTTP.

Identificación de solicitudes GET y POST: En los paquetes HTTP capturados, busca las solicitudes GET y POST. Explica cómo GET se utiliza para solicitar datos de un servidor y POST para enviar datos.

Análisis de las solicitudes GET y POST

Selecciona una solicitud GET y observa los detalles en la sección de capa de aplicación. Destaca la URL solicitada, los encabezados HTTP, y otros metadatos.

Luego, selecciona una solicitud POST y analiza los datos enviados al servidor, como formularios o datos de inicio de sesión (en sitios no cifrados).

3. Filtros en Wireshark

Wireshark permite a los usuarios capturar y analizar grandes volúmenes de tráfico de red. Sin embargo, analizar todos los paquetes capturados puede ser abrumador. Los filtros en Wireshark son herramientas poderosas que ayudan a los usuarios a centrarse en los datos más relevantes, ya sea durante la captura del tráfico o mientras se examina el tráfico capturado.

3.1 Filtros de Captura vs. Filtros de Visualización

Diferencias clave

Filtros de Captura: Se aplican antes o durante la captura del tráfico y determinan qué paquetes se capturan. Si se utiliza un filtro de captura, solo se capturarán los paquetes que coincidan con ese filtro. Esto es útil para reducir la cantidad de datos que Wireshark necesita manejar y para centrarse en un tipo específico de tráfico.

Filtros de Visualización: Se aplican después de que el tráfico ha sido capturado y permiten a los usuarios filtrar la vista para mostrar solo los paquetes que coinciden con criterios específicos. Esto no afecta a los datos capturados, sino que simplemente cambia la manera en que se presentan en la interfaz.

Ejemplo práctico: Configuración de un filtro de captura para capturar solo tráfico ICMP (ping):

1. Aplicar un filtro de captura ICMP

- Antes de iniciar la captura, abre Wireshark y selecciona la interfaz de red adecuada.
- En el campo de filtro de captura, escribe icmp para filtrar solo los paquetes ICMP (que incluyen las solicitudes y respuestas de ping).
- Inicia la captura.

2. Generar tráfico ICMP

- En una terminal o línea de comandos, ejecuta un comando ping hacia una dirección IP o dominio, por ejemplo: ping www.google.com.
- Wireshark capturará únicamente el tráfico ICMP generado por el comando ping.

3. Detener la captura y revisar los resultados

- Detén la captura en Wireshark y examina los paquetes capturados. Solo deberías ver solicitudes y respuestas ICMP, lo que simplifica el análisis si solo estás interesado en el tráfico de ping.

3.2 Uso de Filtros de Visualización

Sintaxis de Filtros

Wireshark utiliza una sintaxis específica para los filtros de visualización, que es poderosa pero sencilla una vez que se comprende. Puedes utilizar filtros para mostrar solo los paquetes que cumplen con ciertos criterios, como un protocolo específico, direcciones IP de origen o destino, puertos, etc.

Algunos ejemplos básicos de filtros de visualización son:

ip.addr == 192.168.1.1: Muestra solo los paquetes donde la dirección IP de origen o destino es 192.168.1.1.

tcp.port == 80: Muestra solo los paquetes TCP que están usando el puerto 80 (HTTP).

http: Muestra solo el tráfico HTTP.

Ejemplo práctico: Aplicar un filtro de visualización para mostrar solo tráfico TCP en el puerto 80 (HTTP):

1. Capturar tráfico sin filtros de captura:

- Inicia Wireshark y selecciona la interfaz de red adecuada.
- Inicia la captura de tráfico sin aplicar ningún filtro de captura.

2. Generar tráfico HTTP:

- Abre un navegador y visita un sitio web que utilice HTTP (no HTTPS). Por ejemplo, puedes intentar con un sitio interno o de prueba que aún use HTTP.

3. Aplicar un filtro de visualización para tráfico TCP en el puerto 80:

- Después de detener la captura, escribe el filtro `tcp.port == 80` en el campo de filtros de Wireshark y presiona Enter.
- Wireshark ahora mostrará solo los paquetes que corresponden al tráfico TCP en el puerto 80, lo que facilita el análisis de las comunicaciones HTTP.

3.3 Filtros Avanzados

Los filtros en Wireshark son herramientas esenciales que permiten a los analistas de seguridad centrarse en el tráfico relevante para su investigación. Saber cuándo y cómo utilizar filtros de captura y visualización puede hacer que el análisis de grandes volúmenes de tráfico sea más manejable y efectivo

Filtros Complejos

Los filtros en Wireshark también permiten la creación de expresiones más complejas usando operadores lógicos como **AND**, **OR** y **NOT**. Esto te permite combinar varios criterios en un solo filtro.

Operadores Lógicos

AND (&&): Muestra paquetes que cumplen con ambos criterios.

OR (| |): Muestra paquetes que cumplen con cualquiera de los criterios.

NOT (!): Excluye los paquetes que cumplen con un criterio específico.

Ejemplo práctico: Filtrar el tráfico para mostrar solo paquetes HTTP que contengan un determinado texto en su contenido:

1. Capturar tráfico HTTP:

- Inicia Wireshark y selecciona la interfaz de red adecuada.
- Inicia la captura de tráfico.

2. Aplicar un filtro avanzado:

- Detén la captura después de haber navegado por algunos sitios web que usan HTTP.
- Para buscar tráfico HTTP que contenga un texto específico en su contenido, como password, utiliza el siguiente filtro:

sql

Copiar código

http contains "password"

- Este filtro mostrará solo los paquetes HTTP que contengan la palabra "password" en su carga útil.

3. Análisis detallado:

- Examina los paquetes filtrados para entender cómo se transmiten los datos. Este tipo de análisis es útil para detectar la transmisión de información sensible en texto claro, lo que puede ser un problema de seguridad.

4. Análisis de Protocolos

El análisis de protocolos es una parte fundamental en la utilización de Wireshark, ya que permite entender cómo se comportan las aplicaciones y los servicios en una red. A continuación, exploraremos cómo analizar algunos de los protocolos más comunes: HTTP, DNS y ARP.

4.1 Análisis de Tráfico HTTP

Descripción del protocolo HTTP

HTTP (Hypertext Transfer Protocol) es un protocolo de la capa de aplicación utilizado para la transmisión de documentos hipermedia, como HTML. Es la base de la comunicación de datos para la World Wide Web.

Los métodos más comunes en HTTP son **GET** y **POST**:

GET: Solicita datos de un servidor.

POST: Envía datos al servidor, generalmente desde un formulario.

Identificación de solicitudes y respuestas HTTP:

Las solicitudes y respuestas HTTP consisten en una línea de inicio, una serie de encabezados (headers) y un cuerpo de mensaje (opcional).

Solicitudes GET: Comienzan con "GET /path HTTP/1.1".

Solicitudes POST: Comienzan con "POST /path HTTP/1.1".

Respuestas HTTP: Comienzan con una línea de estado como "HTTP/1.1 200 OK".

Ejemplo práctico: Identificar y analizar una solicitud GET y la respuesta correspondiente desde un servidor web:

1. Captura del tráfico HTTP:

- Inicia Wireshark y selecciona la interfaz de red correcta (Wi-Fi o Ethernet).
- Comienza una captura de tráfico.
- Abre un navegador y visita un sitio web no cifrado (HTTP, no HTTPS).

2. Filtrar el tráfico HTTP:

- En Wireshark, usa el filtro http para mostrar solo el tráfico HTTP.
- Identifica una solicitud **GET** en la lista de paquetes. Esto se mostrará como "GET /path" en la columna de información.

3. Análisis de la solicitud GET:

- Haz clic en la solicitud GET para ver los detalles en el panel inferior.
- Examina los encabezados HTTP, que incluyen información como la URL solicitada, el agente de usuario (User-Agent) y las cookies.

4. Análisis de la respuesta HTTP:

- Encuentra la respuesta correspondiente a la solicitud GET seleccionada. Generalmente, se muestra como "HTTP/1.1 200 OK".
- Haz clic en la respuesta para analizar el contenido, incluyendo los encabezados y el cuerpo del mensaje (que podría contener el HTML de la página).

4.2 Análisis de Tráfico DNS

Descripción de cómo funciona el protocolo DNS

DNS (Domain Name System) es el protocolo utilizado para resolver nombres de dominio en direcciones IP. Es un servicio crucial que permite a los usuarios acceder a sitios web a través de nombres de dominio en lugar de direcciones IP numéricas.

El flujo típico de DNS incluye una consulta de nombre de dominio (DNS query) y una respuesta con la dirección IP correspondiente (DNS response).

Ejemplo práctico: Capturar y analizar el tráfico DNS mientras se resuelve el nombre de un sitio web

1. Captura de tráfico DNS:

- Inicia una captura en Wireshark como se ha explicado anteriormente.
- En el navegador, accede a un sitio web (puede ser cualquier sitio web).

2. Filtrar el tráfico DNS:

- En Wireshark, usa el filtro dns para mostrar solo el tráfico DNS.
- Identifica las consultas DNS (DNS queries) que aparecen como "Standard query A example.com".

3. Análisis de una consulta DNS:

- Haz clic en una consulta DNS para ver los detalles.
- Observa el nombre de dominio solicitado y el tipo de consulta (por ejemplo, tipo A para direcciones IPv4).

4. Análisis de una respuesta DNS:

- Encuentra la respuesta correspondiente a la consulta DNS. Esto generalmente se muestra como "Standard query response A example.com" seguido de una dirección IP.
- Analiza la respuesta para ver la dirección IP asociada al nombre de dominio.

4.3 Análisis de Tráfico ARP

Explicación del protocolo ARP y su uso en redes

ARP (Address Resolution Protocol) se utiliza para mapear direcciones IP a direcciones MAC en una red local. Es fundamental para la comunicación en redes Ethernet.

Un dispositivo envía una solicitud ARP preguntando "¿Quién tiene esta IP?" y recibe una respuesta con la dirección MAC correspondiente.

Ejemplo práctico: Captura de tráfico ARP para identificar un ataque de ARP spoofing en una red local

1. Captura de tráfico ARP:

- Inicia Wireshark y comienza una captura en la red local.
- En el filtro de Wireshark, escribe arp para mostrar solo el tráfico ARP.

2. Análisis de solicitudes y respuestas ARP:

- Observa las solicitudes ARP (ARP requests) que aparecen como "Who has 192.168.1.1? Tell 192.168.1.100".
- Examina las respuestas ARP (ARP replies) que indican "192.168.1.1 is at 00:11:22:33:44:55".

3. Detección de un ataque ARP spoofing:

- Durante un ataque de ARP spoofing, podrías ver múltiples respuestas ARP que asocian una dirección IP con diferentes direcciones MAC.

- Si identificas respuestas ARP contradictorias, esto podría indicar un intento de ataque de ARP spoofing, donde un atacante intenta interceptar o redirigir el tráfico en la red local.

5. Detección de Actividades Sospechosas

Wireshark es una herramienta poderosa que permite a los profesionales de la seguridad informática detectar actividades sospechosas en la red, como escaneos de puertos, ataques de Man-in-the-Middle (MitM), y tráfico malicioso. En este apartado, exploraremos cómo identificar estos tipos de actividades utilizando Wireshark.

La capacidad de detectar actividades sospechosas con Wireshark es crucial para la seguridad de la red. A través de la identificación de patrones de tráfico inusuales y el análisis detallado de los paquetes, los profesionales pueden identificar y mitigar amenazas antes de que comprometan los sistemas.

5.1 Detección de Intentos de Escaneo de Puertos

Explicación sobre el escaneo de puertos y sus implicaciones en seguridad

Escaneo de Puertos: El escaneo de puertos es una técnica utilizada por atacantes para descubrir servicios activos en un host. Un atacante envía solicitudes a diferentes puertos de un sistema con el fin de identificar cuáles están abiertos y qué servicios están corriendo. Esta información puede ser utilizada para lanzar ataques dirigidos a los servicios descubiertos.

Implicaciones en Seguridad: Detectar un escaneo de puertos es crucial, ya que es un indicio de que un atacante está intentando mapear la red para descubrir vulnerabilidades. Un escaneo de puertos podría ser el primer paso en una cadena de ataques más sofisticados.

Ejemplo práctico: Captura y análisis de un escaneo de puertos utilizando Nmap y cómo detectarlo en Wireshark

1. Configuración del entorno

Asegúrate de tener Wireshark instalado y que el host de destino y la herramienta Nmap estén configurados en la misma red local.

2. Realizar un escaneo de puertos con Nmap

En una terminal, ejecuta el siguiente comando para realizar un escaneo rápido de puertos contra un host específico:

```
nmap -sS <IP_del_host>
```

Este comando realiza un escaneo SYN, que es uno de los tipos más comunes de escaneo de puertos.

3. Capturar tráfico en Wireshark:

Abre Wireshark y selecciona la interfaz de red adecuada.

Inicia la captura de tráfico antes de ejecutar el escaneo con Nmap.

4. Detectar el escaneo de puertos:

Una vez que el escaneo haya terminado, detén la captura en Wireshark.

Filtra los paquetes SYN no respondidos (indicativos de un escaneo de puertos) utilizando el siguiente filtro de visualización:

```
tcp.flags.syn == 1 && tcp.flags.ack == 0
```

Este filtro mostrará solo los paquetes SYN enviados sin una respuesta ACK, lo que es típico en un escaneo de puertos SYN.

5. Análisis:

- Revisa la lista de puertos escaneados y determina si existe un patrón que sugiera un escaneo sistemático. La presencia de múltiples paquetes SYN sin respuestas de puertos sucesivos es un claro indicador de un escaneo de puertos.

5.2 Detección de Ataques de Man-in-the-Middle (MitM)

Descripción del ataque MitM y sus peligros

Man-in-the-Middle (MitM): Es un tipo de ataque en el que un atacante intercepta y posiblemente altera la comunicación entre dos partes sin que ninguna de ellas se dé cuenta. El atacante puede leer, modificar o inyectar datos en la comunicación, lo que lo convierte en un ataque muy peligroso.

Peligros: Un ataque MitM puede llevar a la exposición de información confidencial, manipulación de datos y, en algunos casos, a la completa compromisión de los sistemas involucrados.

Ejemplo práctico: Captura y análisis de un ataque MitM simple en una red local y cómo identificar señales de este ataque en Wireshark

1. Preparar un entorno para el ataque MitM:

Utiliza herramientas como **Ettercap** o **Bettercap** para realizar un ataque MitM en tu red local. Este ataque se puede llevar a cabo usando técnicas como ARP spoofing para interceptar el tráfico entre dos dispositivos en la red.

2. Capturar tráfico con Wireshark:

Abre Wireshark en la misma red donde se realizará el ataque.

Inicia la captura de tráfico antes de realizar el ataque MitM.

3. Realizar el ataque MitM:

Ejecuta el ataque utilizando la herramienta elegida (por ejemplo, Ettercap). Este ataque envenena la tabla ARP de la víctima, redirigiendo su tráfico a través del atacante.

4. Identificar señales del ataque MitM:

En Wireshark, utiliza el siguiente filtro para identificar paquetes ARP sospechosos:

```
arp.duplicate-address-frame
```

Este filtro puede revelar duplicaciones de direcciones IP o cambios inesperados en las direcciones MAC asociadas, lo que es indicativo de ARP spoofing.

5. Analizar el tráfico:

Revisa si hay tráfico sospechoso como respuestas ARP inesperadas o paquetes que parecen estar manipulados. También puedes buscar tráfico HTTPS que no esté cifrado, lo que podría indicar que un atacante ha establecido un proxy MitM para descifrar y analizar el tráfico.

5.3 Análisis de Tráfico Malicioso

Identificación de patrones comunes en tráfico malicioso:

Tráfico No Cifrado: La falta de cifrado en comunicaciones sensibles es un fuerte indicio de tráfico malicioso. Tráfico HTTP, en lugar de HTTPS, para transmisión de credenciales o datos personales es una señal de advertencia.

Direcciones IP Sospechosas: Conexiones a IPs conocidas por estar asociadas con actividades maliciosas, como redes de botnets o servidores de comando y control (C2), son claras indicaciones de tráfico malicioso.

Ejemplo práctico: Análisis de una captura de tráfico que contiene un intento de explotación de vulnerabilidades en un servidor web:

1. Capturar tráfico de un servidor web vulnerable:

Si tienes acceso a un entorno de pruebas, configura un servidor web vulnerable y realiza un ataque de explotación conocido, como un SQL Injection o un intento de ejecución de comandos remotos.

2. Iniciar captura con Wireshark:

Captura todo el tráfico hacia y desde el servidor web durante el intento de explotación.

3. Aplicar filtros para analizar el tráfico:

Usa el siguiente filtro para detectar solicitudes HTTP sospechosas:

```
http.request.method == "POST" && http contains "union select"
```

Este filtro muestra solicitudes POST que contienen la cadena "union select", común en ataques SQL Injection.

4. Analizar el tráfico malicioso:

Revisa las solicitudes y respuestas HTTP para identificar comportamientos anómalos, como solicitudes que intentan explotar vulnerabilidades conocidas. Busca patrones inusuales en la estructura de los parámetros de las URLs o en los datos enviados en las solicitudes POST.

6. Exportación y Documentación

Una vez que has capturado y analizado el tráfico de red con Wireshark, es crucial saber cómo exportar estos datos y documentar tus hallazgos. Esto no solo te permite compartir tus descubrimientos con otros profesionales, sino que también es una parte vital de la realización de auditorías, investigaciones forenses y análisis de seguridad.

Saber cómo exportar datos y documentar hallazgos es esencial para un análisis de red eficaz y profesional. La exportación permite compartir y revisar capturas más tarde, mientras que la documentación asegura que los análisis sean comprensibles y útiles para otros

6.1 Exportar Datos Capturados

Explicación sobre la exportación de datos:

Propósito de la Exportación: La exportación de datos capturados permite almacenar la información para análisis futuros, compartirla con colegas, o importar los datos en otras herramientas para un análisis más detallado.

Formatos de Exportación: Wireshark soporta una variedad de formatos de exportación, que son útiles para diferentes propósitos. Los formatos comunes incluyen:

PCAP (Packet Capture): Este es el formato más común y contiene todos los detalles de los paquetes capturados. Es ampliamente compatible con otras herramientas de análisis de red.

TXT (Texto): Exporta los datos en un formato de texto plano, útil para reportes simples o para importarlos en aplicaciones de procesamiento de texto.

CSV (Comma-Separated Values): Este formato organiza los datos en un formato tabular, ideal para análisis en hojas de cálculo o bases de datos.

Ejemplo práctico: Exportar un conjunto de paquetes filtrados a un archivo para análisis posterior:

1. Filtrar el tráfico deseado:

Abre Wireshark y carga una captura de tráfico.

Aplica un filtro para seleccionar los paquetes que desees exportar. Por ejemplo, si solo te interesan los paquetes HTTP, utiliza:

```
http
```

2. Exportar los paquetes filtrados:

Ve al menú **File** y selecciona **Export Specified Packets**.

Asegúrate de seleccionar la opción "Displayed" en la sección de rango de paquetes, para exportar solo los paquetes que coinciden con tu filtro actual.

Elige el formato de exportación:

- Si eliges **PCAP**, guarda el archivo como .pcap para futuras capturas.
- Si eliges **CSV**, el archivo se guardará como .csv, que puedes abrir en programas como Excel.

Asigna un nombre al archivo y guarda.

3. Verificar la exportación:

Abre el archivo exportado con una herramienta adecuada para asegurarte de que los datos se exportaron correctamente.

6.2 Creación de Informes

Documentar hallazgos:

Importancia de la Documentación: La documentación es crucial para comunicar los resultados de tu análisis a otros profesionales o equipos. Un informe bien estructurado facilita la comprensión de los eventos observados y las conclusiones obtenidas.

Elementos de un Informe Efectivo:

Resumen Ejecutivo: Una descripción breve del objetivo del análisis, incluyendo hallazgos clave.

Metodología: Explicación de cómo se realizó la captura de tráfico, incluyendo los filtros utilizados y las herramientas complementarias.

Resultados: Detalles de los hallazgos, con capturas de pantalla de Wireshark y explicaciones sobre patrones sospechosos o problemas de red.

Recomendaciones: Sugerencias para mitigar los problemas encontrados o acciones a tomar basadas en el análisis.

Apéndices: Incluir datos exportados, configuraciones de filtros, y otros detalles técnicos relevantes.

Ejemplo práctico: Crear un informe detallado basado en la captura de un ataque DDoS simulado:

1. Preparar la captura:

Realiza una simulación de un ataque DDoS (Denial of Service) utilizando una herramienta como **LOIC** o **Hping3** para generar tráfico anómalo en una red controlada.

2. Capturar y analizar el tráfico:

Inicia Wireshark y captura el tráfico durante la simulación.

Analiza el tráfico usando filtros como:

```
ip.src == <IP_origen_ataque>
```

Este filtro te ayudará a identificar el tráfico proveniente de la fuente del ataque.

3. Documentar los hallazgos:

Crea un nuevo documento en un procesador de texto.

Comienza con un **Resumen Ejecutivo** que explique el contexto del ataque simulado y los objetivos del análisis.

En la sección de **Metodología**, describe cómo realizaste la captura de tráfico, mencionando los filtros utilizados y las herramientas empleadas.

En **Resultados**, inserta capturas de pantalla de Wireshark que muestren el tráfico anómalo, con explicaciones sobre las características del ataque (por ejemplo, múltiples paquetes SYN enviados en un corto periodo de tiempo).

En **Recomendaciones**, sugiere acciones como la implementación de sistemas de detección de intrusiones (IDS) para identificar y mitigar ataques similares.

Incluye un **Apéndice** con el archivo PCAP exportado y los comandos utilizados en la simulación.

4. **Revisión y entrega del informe:**

Revisa el informe para asegurar que esté claro y completo.

Exporta el documento en formato PDF o Word y compártelo con el equipo o instructor.

7. Buenas Prácticas y Consideraciones Éticas

El uso de Wireshark, como cualquier herramienta poderosa en el ámbito de la seguridad informática, requiere una comprensión sólida no solo de sus capacidades técnicas, sino también de las implicaciones legales y éticas de su utilización. En este apartado, aprenderás cómo aplicar buenas prácticas y actuar de manera ética y legal al utilizar Wireshark.

Es esencial que los profesionales y estudiantes de seguridad informática utilicen Wireshark de manera legal y ética, respetando la privacidad de los usuarios y cumpliendo con las regulaciones aplicables. Además, la configuración de alertas en Wireshark, aunque limitada, puede ser una herramienta útil para la detección temprana de actividades sospechosas.

7.1 Legalidad y Ética en el Uso de Wireshark

Explicación sobre la legalidad y la ética:

Aspectos Legales:

Permisos y Consentimiento: En la mayoría de las jurisdicciones, es ilegal capturar tráfico de red sin el consentimiento explícito de los propietarios de la red o de los participantes en la comunicación. La interceptación de comunicaciones sin autorización puede ser considerada una violación de leyes como el **Wiretap Act** en los Estados Unidos o leyes de privacidad en la Unión Europea.

Regulaciones y Normativas: Algunas normativas, como el **Reglamento General de Protección de Datos (GDPR)** en Europa, imponen estrictas reglas sobre la recopilación y manejo de datos personales. Cualquier captura de tráfico que involucre datos personales debe cumplir con estas regulaciones.

Aspectos Éticos:

Responsabilidad Profesional: Los profesionales de la seguridad informática deben adherirse a códigos de ética, como los propuestos por organizaciones como **(ISC)²** o **ISACA**, que subrayan la importancia de proteger la privacidad y actuar en el mejor interés de la sociedad.

Uso Responsable: Es fundamental utilizar Wireshark y herramientas similares exclusivamente para fines autorizados y beneficiosos, como la auditoría de seguridad de redes bajo tu control o la investigación académica con el debido consentimiento.

Ejemplo práctico: Escenarios legales y éticos en la captura de tráfico en redes corporativas:

Escenario Legal: Un administrador de red desea monitorear el tráfico en una red corporativa para detectar posibles ataques. Sin embargo, para cumplir con la legislación local, el administrador primero debe:

- **Obtener el consentimiento de los empleados** mediante una política de uso aceptable firmada.
- **Informar a los empleados** sobre el alcance del monitoreo y cómo se protegerán sus datos personales.

Escenario Ético: Un estudiante de seguridad informática está tentado a utilizar Wireshark para capturar tráfico en una red Wi-Fi pública, sin tener en cuenta que podría estar accediendo a información sensible de otros usuarios. **La acción correcta** sería abstenerse de capturar el tráfico, ya que hacerlo sin el consentimiento de los usuarios de la red sería un acto no ético y posiblemente ilegal.

7.2 Configuración de Alertas y Notificaciones

Uso de Wireshark para configurar alertas sobre actividades sospechosas:

Propósito de las Alertas: Configurar alertas en Wireshark puede ayudarte a identificar y responder rápidamente a actividades sospechosas en una red, como intentos de ataque o el envío de datos sensibles en texto claro.

Tipos de Alertas: Aunque Wireshark no es un sistema de detección de intrusiones (IDS) en sí, puedes usar **scripts personalizados o filtros** para generar alertas basadas en patrones específicos de tráfico.

Ejemplo práctico: Configurar una alerta para notificar cuando se detecte tráfico HTTP que contenga datos de inicio de sesión en texto claro:

1. Identificar el tráfico de interés:

En Wireshark, utiliza el siguiente filtro de visualización para identificar tráfico HTTP que contiene datos sensibles:

```
http.request.method == "POST" && http contains "password"
```

Este filtro mostrará todas las solicitudes HTTP POST que contienen la palabra "password", comúnmente utilizada en formularios de inicio de sesión.

2. Crear una alerta personalizada:

- Aunque Wireshark no envía alertas por defecto, puedes usar herramientas como **tshark** (la versión de línea de comandos de Wireshark) combinada con un script en **Python** o **Bash** que monitoree el tráfico en tiempo real y envíe una notificación (por ejemplo, un correo electrónico) cuando se detecte tráfico que coincida con el filtro anterior.

3. Implementar y probar la alerta:

- Ejecuta el script en un entorno de prueba y realiza una captura de tráfico simulando una solicitud de inicio de sesión insegura. Verifica que la alerta se active correctamente y que la notificación se envíe al destinatario designado.

8. Conclusión y resumen

Después de haber explorado y trabajado con Wireshark en diversas áreas, desde la captura de tráfico hasta la detección de actividades sospechosas, es crucial recapitular los conocimientos adquiridos y asegurarse de que los estudiantes estén preparados para aplicar estos conceptos en escenarios reales. En esta última sección, resumiremos los puntos clave, realizaremos un ejercicio práctico final, y ofreceremos sugerencias para ejercicios adicionales que los estudiantes puedan realizar por su cuenta para seguir profundizando en el uso de Wireshark.

8.1 Resumen

Resumen de las características principales de Wireshark y su aplicación en seguridad informática:

Wireshark como Herramienta Fundamental:

Captura de Tráfico en Tiempo Real: Wireshark permite a los usuarios capturar y analizar tráfico de red en tiempo real, proporcionando una visión detallada de lo que ocurre en una red.

Análisis de Protocolos: Es posible analizar una amplia variedad de protocolos de red, lo que permite a los profesionales de seguridad identificar vulnerabilidades, errores de configuración y actividades sospechosas.

Filtros Potentes: Los filtros de captura y de visualización son herramientas esenciales para enfocar la atención en el tráfico de interés, simplificando el análisis y permitiendo la identificación rápida de problemas específicos.

Documentación y Reportes: Wireshark facilita la exportación de datos y la generación de informes detallados, cruciales para la documentación de incidentes de seguridad y la toma de decisiones informadas.

Ejemplo práctico: Realizar una captura de tráfico final donde los estudiantes aplican los conocimientos adquiridos para identificar y documentar un problema de red.

Objetivo del Ejercicio:

Realizar una captura de tráfico en una red local.

Identificar y documentar un problema de red utilizando los conocimientos adquiridos en el curso.

Instrucciones:

1. Preparación de la Captura:

Selecciona la interfaz de red adecuada y comienza la captura de tráfico utilizando Wireshark.

2. Aplicación de Filtros:

Utiliza filtros de visualización para enfocar la atención en el tráfico relevante. Por ejemplo, si se sospecha de un problema relacionado con HTTP, aplica el filtro:

```
tcp.port == 80
```

Examina los paquetes capturados para identificar solicitudes y respuestas HTTP sospechosas.

3. Análisis de los Resultados:

Inspecciona los paquetes filtrados para detectar anomalías como retrasos en las respuestas, errores HTTP, o datos sensibles transmitidos en texto claro.

Documenta los hallazgos utilizando la función de exportación de Wireshark para guardar los paquetes relevantes y genera un informe detallado que describa el problema encontrado, las posibles causas, y las recomendaciones para resolverlo.

4. **Presentación de Resultados:**

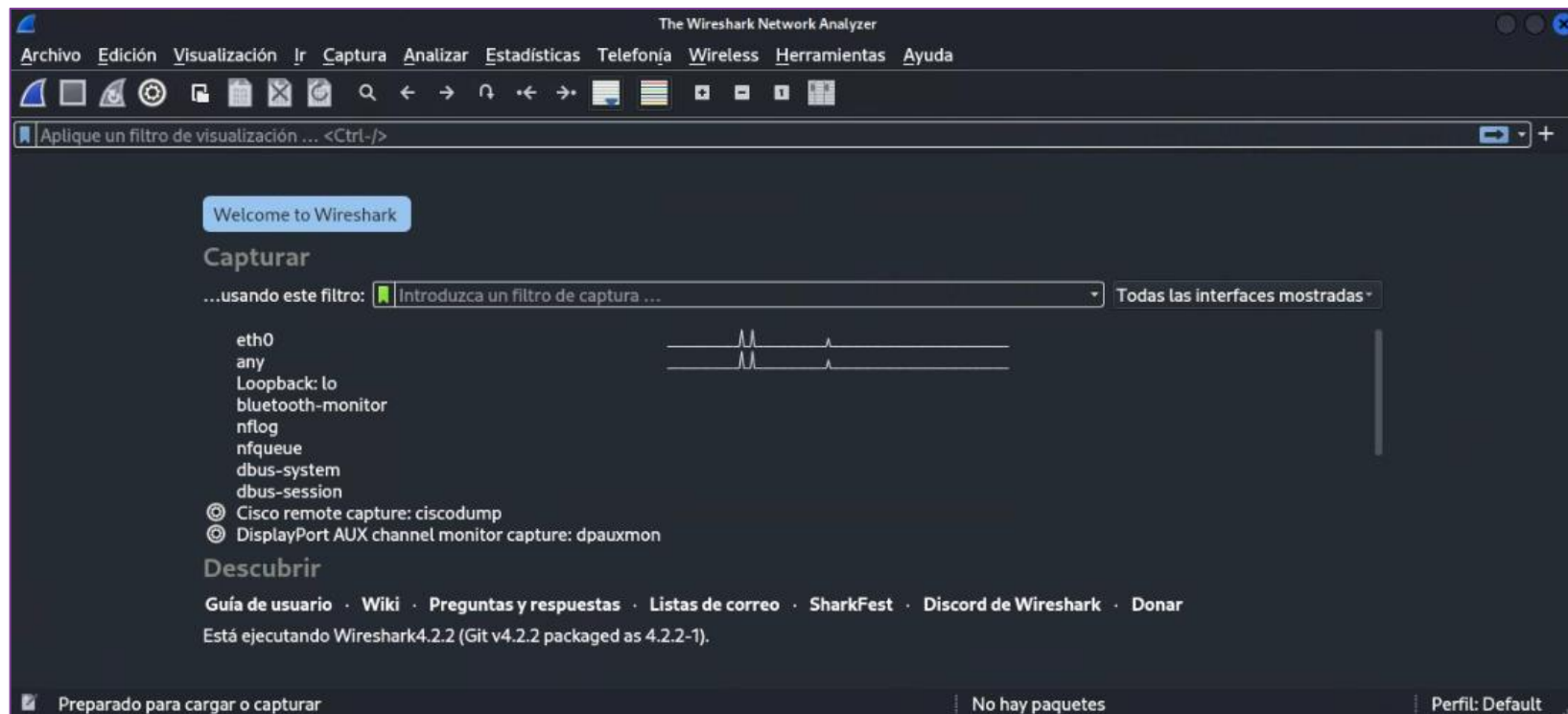
- Pide a los estudiantes que compartan sus hallazgos y discutan las posibles soluciones al problema detectado.

9. Ejercicios

Abrir Wireshark:

```
(root@Newkali)~[/home/pru]
# wireshark
** (wireshark:2125) 11:50:02.400323 [GUI WARNING] -- QStandardPaths: XDG_
RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
** (wireshark:2125) 11:50:06.099369 [Main MESSAGE] -- Wireshark is up and
ready to go, elapsed time 3,714s
```

Se abre el programa y muestra la siguiente interfaz:

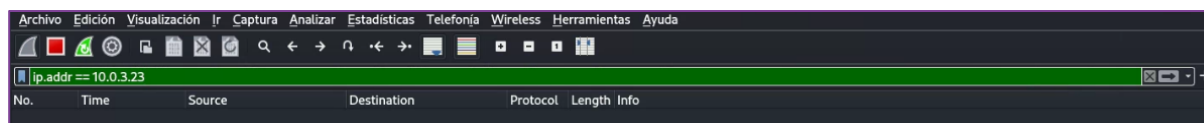


Para la actividad contamos con una máquina **Windows** (10.0.3.22), una máquina **Linux Ubuntu** (10.0.3.23) y la máquina **metasploitable2** (10.0.3.25)

1. PING Comando de análisis de captura de paquetes de red

El comando **PING** funciona según el protocolo **ICMP**, enviando paquetes y devolviendo paquetes normalmente. Tomando el host 10.0.3.25 como ejemplo, los principales pasos experimentales son:

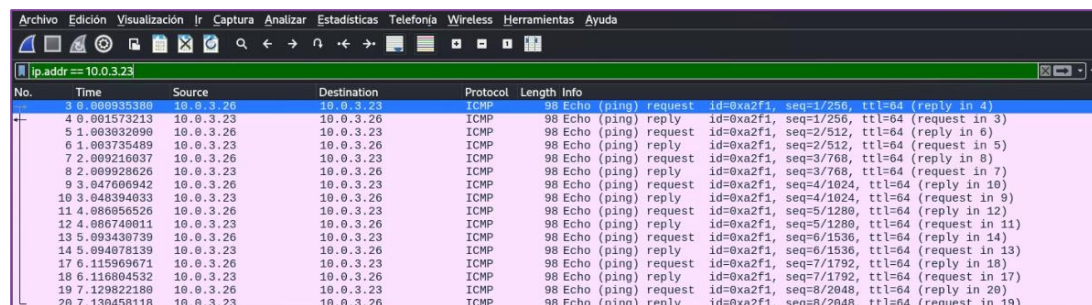
- (1) Configure el "filtro de captura": complete el host 10.0.2.10 en Filtro de captura;
- (2) Comience a capturar paquetes;



- (3) Ejecute el comando PING

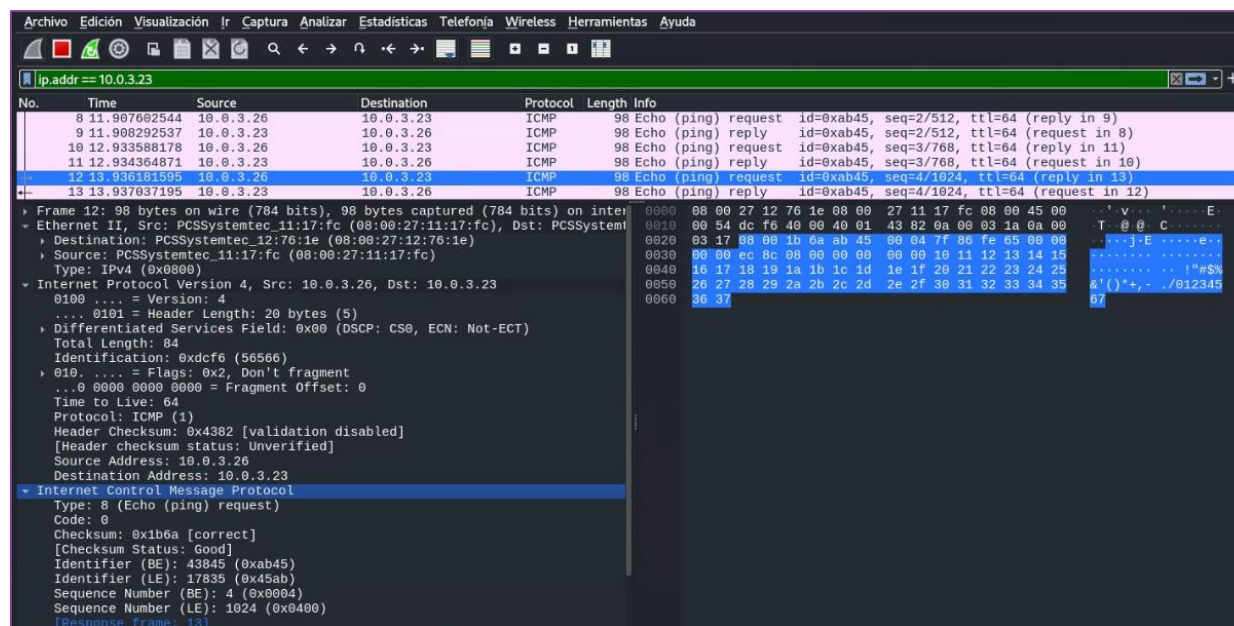
```
(pru@kali)-[~]
$ ping 10.0.3.23
PING 10.0.3.23 (10.0.3.23) 56(84) bytes of data:
64 bytes from 10.0.3.23: icmp_seq=1 ttl=64 time=1.59 ms
64 bytes from 10.0.3.23: icmp_seq=2 ttl=64 time=0.738 ms
64 bytes from 10.0.3.23: icmp_seq=3 ttl=64 time=0.733 ms
64 bytes from 10.0.3.23: icmp_seq=4 ttl=64 time=0.807 ms
64 bytes from 10.0.3.23: icmp_seq=5 ttl=64 time=0.699 ms
64 bytes from 10.0.3.23: icmp_seq=6 ttl=64 time=0.668 ms
64 bytes from 10.0.3.23: icmp_seq=7 ttl=64 time=0.859 ms
64 bytes from 10.0.3.23: icmp_seq=8 ttl=64 time=0.654 ms
^C
— 10.0.3.23 ping statistics —
8 packets transmitted, 8 received, 0% packet loss, time
7130ms
rtt min/avg/max/mdev = 0.654/0.842/1.585/0.287 ms
```

(4) Dejar de capturar paquetes.



No.	Time	Source	Destination	Protocol	Length	Info
3	0.000935380	10.0.3.26	10.0.3.23	ICMP	98	Echo (ping) request id=0xa2f1, seq=1/256, ttl=64 (reply in 4)
4	0.001573213	10.0.3.23	10.0.3.26	ICMP	98	Echo (ping) reply id=0xa2f1, seq=1/256, ttl=64 (request in 3)
5	1.003932090	10.0.3.26	10.0.3.23	ICMP	98	Echo (ping) request id=0xa2f1, seq=2/512, ttl=64 (reply in 6)
6	1.003735469	10.0.3.23	10.0.3.26	ICMP	98	Echo (ping) reply id=0xa2f1, seq=2/512, ttl=64 (request in 5)
7	2.009216037	10.0.3.26	10.0.3.23	ICMP	98	Echo (ping) request id=0xa2f1, seq=3/768, ttl=64 (reply in 8)
8	2.009928626	10.0.3.23	10.0.3.26	ICMP	98	Echo (ping) reply id=0xa2f1, seq=3/768, ttl=64 (request in 7)
9	3.047606942	10.0.3.26	10.0.3.23	ICMP	98	Echo (ping) request id=0xa2f1, seq=4/1024, ttl=64 (reply in 10)
10	3.048394033	10.0.3.23	10.0.3.26	ICMP	98	Echo (ping) reply id=0xa2f1, seq=4/1024, ttl=64 (request in 9)
11	4.080856526	10.0.3.26	10.0.3.23	ICMP	98	Echo (ping) request id=0xa2f1, seq=5/1280, ttl=64 (reply in 12)
12	4.080749011	10.0.3.23	10.0.3.26	ICMP	98	Echo (ping) reply id=0xa2f1, seq=5/1280, ttl=64 (request in 11)
13	5.093430739	10.0.3.26	10.0.3.23	ICMP	98	Echo (ping) request id=0xa2f1, seq=6/1536, ttl=64 (reply in 14)
14	5.094078139	10.0.3.23	10.0.3.26	ICMP	98	Echo (ping) reply id=0xa2f1, seq=6/1536, ttl=64 (request in 13)
17	6.115909671	10.0.3.26	10.0.3.23	ICMP	98	Echo (ping) request id=0xa2f1, seq=7/1792, ttl=64 (reply in 18)
18	6.116804532	10.0.3.23	10.0.3.26	ICMP	98	Echo (ping) reply id=0xa2f1, seq=7/1792, ttl=64 (request in 17)
19	7.129822180	10.0.3.26	10.0.3.23	ICMP	98	Echo (ping) request id=0xa2f1, seq=8/2048, ttl=64 (reply in 20)
20	7.130458118	10.0.3.23	10.0.3.26	ICMP	98	Echo (ping) reply id=0xa2f1, seq=8/2048, ttl=64 (request in 19)

(5) Seleccione un determinado paquete de datos, concéntrese en analizar la parte del protocolo, especialmente el contenido del encabezado del protocolo.



Frame 12: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: PCSSystemtec 11:17:fc (08:00:27:11:17:fc), Dst: PCSSystemtec 12:76:1e (08:00:27:12:76:1e)
Destination: PCSSystemtec 12:76:1e (08:00:27:12:76:1e)
Source: PCSSystemtec 11:17:fc (08:00:27:11:17:fc)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 10.0.3.26, Dst: 10.0.3.23
0100 = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 64
Identification: 0xdef6 (56566)
010. = Flags: 0x2, Don't fragment
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 64
Protocol: ICMP (1)
Header Checksum: 0x4382 [validation disabled]
[Header checksum status: Unverified]
Source Address: 10.0.3.26
Destination Address: 10.0.3.23
Internet Control Message Protocol
Type: 8 (Echo (ping) request)
Code: 0
Checksum: 0x1b6a [correct]
[Checksum Status: Good]
Identifier (BE): 43845 (0xab45)
Identifier (LE): 17835 (0x45ab)
Sequence Number (BE): 4 (0x0004)
Sequence Number (LE): 1024 (0x0400)
[Response frame: 13]

0000 08 00 27 12 76 1e 08 00 27 11 17 fc 08 00 45 00 ... v ... E
0010 00 54 dc f6 40 00 40 01 43 82 0a 00 03 1a 0a 00 ... T _ @ C
0020 03 17 08 00 1b 6a ab 45 00 04 7f 86 fe 65 00 00 j . E
0030 00 00 ec 8c 08 00 00 00 00 00 10 11 12 13 14 15
0040 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25
0050 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 &'()*+,-./012345
0060 36 37

Mediante un análisis simple, podemos ver que el comando **Ping** utiliza el protocolo **ICMP**. Cada vez que se envía un paquete a 10.0.3.23, se recibirá un paquete de respuesta (reply).

12	13.936181595	10.0.3.26	10.0.3.23	ICMP	98 Echo (ping) request	id=0xab45, seq=4/1024, ttl=64 (reply in 13)
13	13.937037195	10.0.3.23	10.0.3.26	ICMP	98 Echo (ping) reply	id=0xab45, seq=4/1024, ttl=64 (request in 12)

2. TRACERT

Desde la máquina **Windows** vamos a hacer un **Tracert** 142.250.201.67. Primero ponemos a capturar a **Wireshark**

```
C:\Users\pru>tracert 142.250.201.67

Trazo a la dirección mad07s25-in-f3.1e100.net [142.250.201.67]
sobre un máximo de 30 saltos:

  1  <1 ms    <1 ms    <1 ms    10.0.3.1
  2   9 ms     5 ms     5 ms     192.168.1.1
  3   8 ms     8 ms     7 ms     192.168.144.1
  4   7 ms     7 ms     8 ms     133.red-81-41-252.staticip.rima-tde.net [81.41.252.133]
  5   *        *        *        Tiempo de espera agotado para esta solicitud.
  6   *        *        *        Tiempo de espera agotado para esta solicitud.
  7  31 ms    31 ms    31 ms    176.52.253.97
  8  32 ms    33 ms    32 ms    google-ae4-0-grcmadjv2.net.telefonicaglobalsolutions.com [
213.140.50.41]
  9   36 ms    32 ms    43 ms    142.251.231.147
 10   33 ms    31 ms    31 ms    74.125.37.87
 11   32 ms    32 ms    31 ms    mad07s25-in-f3.1e100.net [142.250.201.67]

Trazo completa.

C:\Users\pru>
```

Wireshark packet capture showing ICMP Echo (ping) requests to 142.250.201.67. The packet list shows several failed ping attempts with 'Time to live exceeded' or 'no response found!'. The packet details pane shows the structure of an ICMP Echo request, including the header and data fields.

Filter: ip.dst_host==142.250.201.67

No.	Time	Source	Destination	Protocol	Length	Info
7	11.946056741	10.0.3.22	142.250.201.67	ICMP	106	Echo (ping) request id=0x0001, seq=143/36088, ttl=1 (no response found!)
8	11.946056947	10.0.3.1	10.0.3.22	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
9	11.949004776	10.0.3.22	142.250.201.67	ICMP	106	Echo (ping) request id=0x0001, seq=144/36864, ttl=1 (no response found!)
10	11.949004868	10.0.3.1	10.0.3.22	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
12	11.952370958	10.0.3.22	142.250.201.67	ICMP	106	Echo (ping) request id=0x0001, seq=145/37120, ttl=1 (no response found!)
13	11.952371231	10.0.3.1	10.0.3.22	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
28	17.610131252	10.0.3.22	142.250.201.67	ICMP	106	Echo (ping) request id=0x0001, seq=146/37376, ttl=2 (no response found!)
29	17.614873037	192.168.1.1	10.0.3.22	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
30	17.617668752	10.0.3.22	142.250.201.67	ICMP	106	Echo (ping) request id=0x0001, seq=147/37632, ttl=2 (no response found!)
31	17.623515874	192.168.1.1	10.0.3.22	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
32	17.625933503	10.0.3.22	142.250.201.67	ICMP	106	Echo (ping) request id=0x0001, seq=148/37888, ttl=2 (no response found!)
33	17.626003927	192.168.1.1	10.0.3.22	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)

Frame 7: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface 0
Ethernet II, Src: PCSSystemtec_03:92:75 (08:00:27:03:92:75), Dst: 52:54:00:12:35:00 (52:54:00:12:35:00)
Source: PCSSystemtec_03:92:75 (08:00:27:03:92:75)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 10.0.3.22, Dst: 142.250.201.67
0100 = Version: 4
... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 92
Identification: 0xede7 (58607)
000. = Flags: 0x0
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 1
Protocol: ICMP (1)
Header Checksum: 0x6f5e [validation disabled]
[Header checksum status: Unverified]
Source Address: 10.0.3.22
Destination Address: 142.250.201.67
Internet Control Message Protocol
Type: 8 (Echo (ping) request)
Code: 0
Checksum: 0xf76f [correct]
[Checksum status: Good]

Destination Address (ip.dst), 4 byte(s) | Paquetes: 115 - Mostrado: 57 (49.6%) | Perfil: Default

3. FTP

ftp [ftp.dlptest.com](ftp://ftp.dlptest.com)

Los datos del Servidor FTP son:

URL: <ftp://ftp.dlptest.com/> Usuario: dlpuser Contraseña:
rNrKYTX9g7z3RgJRmxWuGHbeu

```
C:\Users\pru>ftp ftp.dlptest.com
Conectado a ftp.dlptest.com.
220 Welcome to the DLP Test FTP Server
200 Always in UTF8 mode.
Usuario (ftp.dlptest.com:(none)): dlpuser
331 Please specify the password.
Contraseña:
230 Login successful.
ftp> ls
500 Illegal PORT command.
425 Use PORT or PASV first.
ftp> list
Comando no válido.
ftp> pwd
257 "/"
ftp> ls
425 Use PORT or PASV first.
ftp> quit
221 Goodbye.
```

ftp

No.	Time	Source	Destination	Protocol	Length	Info
4	0.428169935	44.241.66.173	10.0.3.22	FTP	94	Response: 220 Welcome to the DLP Test FTP Server
5	0.453736403	10.0.3.22	44.241.66.173	FTP	68	Request: OPTS UTF8 ON
7	0.666053281	44.241.66.173	10.0.3.22	FTP	80	Response: 200 Always in UTF8 mode.
11	6.184548680	10.0.3.22	44.241.66.173	FTP	68	Request: USER dlpuser
13	6.397102580	44.241.66.173	10.0.3.22	FTP	88	Response: 331 Please specify the password.
44	20.122899284	10.0.3.22	44.241.66.173	FTP	86	Request: PASS rNrKYTX9g7z3RgJRmxWuGHbeu
47	20.583330168	44.241.66.173	10.0.3.22	FTP	77	Response: 230 Login successful.
50	25.114086958	10.0.3.22	44.241.66.173	FTP	77	Request: PORT 10,0,3,22,195,24

> Frame 11: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface
 > Ethernet II, Src: PCSSystemtec_03:92:75 (08:00:27:03:92:75), Dst: 52:54:00:12:35:00
 > Internet Protocol Version 4, Src: 10.0.3.22, Dst: 44.241.66.173
 > Transmission Control Protocol, Src Port: 49943, Dst Port: 21, Seq: 15, Ack: 10000
 > File Transfer Protocol (FTP)
 > USER dlpuser\r\n
 Request command: USER
 Request arg: dlpuser
 [Current working directory:]

0000 52 54 00 12 35 00 08 00 27 03 92 75 08 00 45 00 RT...5...!...u...E...
 0010 00 36 5d 34 40 00 80 06 20 da 0a 00 03 16 2c f1 H]4@... ..,
 0020 42 ad c3 17 00 15 f3 3f 78 79 00 16 e7 1d 50 18 B...? xy...P...
 0030 1f be ed c8 00 00 55 53 45 52 20 64 6c 70 75 73US ER dlpus
 0040 65 72 0d 0a er...

ftp

No.	Time	Source	Destination	Protocol	Length	Info
4	0.428169935	44.241.66.173	10.0.3.22	FTP	94	Response: 220 Welcome to the DLP Test FTP Server
5	0.453736403	10.0.3.22	44.241.66.173	FTP	68	Request: OPTS UTF8 ON
7	0.666053281	44.241.66.173	10.0.3.22	FTP	80	Response: 200 Always in UTF8 mode.
11	6.184548680	10.0.3.22	44.241.66.173	FTP	68	Request: USER dlpuser
13	6.397102580	44.241.66.173	10.0.3.22	FTP	88	Response: 331 Please specify the password.
44	20.122899284	10.0.3.22	44.241.66.173	FTP	86	Request: PASS rNrKYTX9g7z3RgJRmxWuGHbeu
47	20.583330168	44.241.66.173	10.0.3.22	FTP	77	Response: 230 Login successful.
50	25.114086958	10.0.3.22	44.241.66.173	FTP	77	Request: PORT 10,0,3,22,195,24

> Frame 44: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface
 > Ethernet II, Src: PCSSystemtec_03:92:75 (08:00:27:03:92:75), Dst: 52:54:00:12:35:00
 > Internet Protocol Version 4, Src: 10.0.3.22, Dst: 44.241.66.173
 > Transmission Control Protocol, Src Port: 49943, Dst Port: 21, Seq: 29, Ack: 10000
 > File Transfer Protocol (FTP)
 > PASS rNrKYTX9g7z3RgJRmxWuGHbeu\r\n
 Request command: PASS
 Request arg: rNrKYTX9g7z3RgJRmxWuGHbeu
 [Current working directory:]

0000 52 54 00 12 35 00 08 00 27 03 92 75 08 00 45 00 RT...5...!...u...E...
 0010 00 48 5d 36 40 00 80 06 20 c6 0a 00 03 16 2c f1 H]6@... ..,
 0020 42 ad c3 17 00 15 f3 3f 78 87 00 16 e7 3f 50 18 B...? x...?P...
 0030 1f 9c 44 7c 00 00 50 41 53 53 20 72 4e 72 4b 59 ..D|..PA SS rNrKY
 0040 54 58 39 67 37 7a 33 52 67 4a 52 6d 78 57 75 47 TX9g7z3R gJRmxWuG
 0050 48 62 65 75 0d 0a Hbeu...

4. Nmap

Nmap -T4 -A 10.0.3.25

```
C:\Users\pru>nmap -T4 -A 10.0.3.25
Starting Nmap 7.94 ( https://nmap.org ) at 2024-03-23 09:30 Hora estándar romance
Stats: 0:02:48 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 99.43% done; ETC: 09:33 (0:00:01 remaining)
Nmap scan report for 10.0.3.25
Host is up (0.0011s latency).
Not shown: 977 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ftp-syst:
|   STAT:
|   FTP server status:
|       Connected to 10.0.3.22
|       Logged in as ftp
|       TYPE: ASCII
|       No session bandwidth limit
|       Session timeout in seconds is 300
|       Control connection is plain text
|       Data connections will be plain text
|       vsFTPd 2.3.4 - secure, fast, stable
|_End of status
|_ms-sql-info: ERROR: Script execution failed (use -d to debug)
|_ms-sql-ntlm-info: ERROR: Script execution failed (use -d to debug)
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
|_ssh-hostkey:
|   1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|   2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
|_ms-sql-ntlm-info: ERROR: Script execution failed (use -d to debug)
|_ms-sql-info: ERROR: Script execution failed (use -d to debug)
23/tcp    open  telnet       Linux telnetd
```

Aplicamos el filtro ip.dst ==10.0.3.25 || tcp.port ==80

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

ip.dst==10.0.3.25 || tcp.port==80

No.	Time	Source	Destination	Protocol	Length	Info
4303	142.361380214	10.0.3.22	10.0.3.25	SMB	203	Session Setup AndX Request, NTLMSSP_NEGOTIATE
4305	142.363898020	10.0.3.22	10.0.3.25	TCP	60	50114 → 445 [FIN, ACK] Seq=203 Ack=471 Win=2101760 Len=0
4306	142.365829042	10.0.3.22	10.0.3.25	UDP	60	60697 → 1434 Len=1
4307	142.365829202	10.0.3.22	10.0.3.25	ICMP	71	Destination unreachable (Port unreachable)
4309	142.372339017	10.0.3.22	10.0.3.25	TCP	60	50114 → 445 [ACK] Seq=204 Ack=472 Win=2101760 Len=0
4310	152.406151438	10.0.3.22	10.0.3.25	TCP	66	50115 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
4312	152.406629684	10.0.3.22	10.0.3.25	TCP	60	50115 → 445 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
4313	152.408186062	10.0.3.22	10.0.3.25	SMB	107	Negotiate Protocol Request
4316	152.409921516	10.0.3.22	10.0.3.25	SMB	203	Session Setup AndX Request, NTLMSSP_NEGOTIATE
4318	152.411939031	10.0.3.22	10.0.3.25	TCP	60	50115 → 445 [FIN, ACK] Seq=203 Ack=471 Win=2101760 Len=0
4319	152.413254597	10.0.3.22	10.0.3.25	UDP	60	60698 → 1434 Len=1
4320	152.413254719	10.0.3.22	10.0.3.25	ICMP	71	Destination unreachable (Port unreachable)
4322	152.417121421	10.0.3.22	10.0.3.25	TCP	60	50115 → 445 [ACK] Seq=204 Ack=472 Win=2101760 Len=0
4323	152.634048761	10.0.3.22	10.0.3.25	TCP	60	49958 → 1099 [RST, ACK] Seq=9 Ack=17 Win=0 Len=0
4324	162.436933658	10.0.3.22	10.0.3.25	TCP	66	50116 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
4326	162.437499070	10.0.3.22	10.0.3.25	TCP	60	50116 → 445 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
4327	162.438918111	10.0.3.22	10.0.3.25	SMB	107	Negotiate Protocol Request
4330	162.440862202	10.0.3.22	10.0.3.25	SMB	203	Session Setup AndX Request, NTLMSSP_NEGOTIATE
4332	162.442873197	10.0.3.22	10.0.3.25	TCP	60	50116 → 445 [FIN, ACK] Seq=203 Ack=471 Win=2101760 Len=0
4333	162.444206993	10.0.3.22	10.0.3.25	UDP	60	60699 → 1434 Len=1
4334	162.444463994	10.0.3.22	10.0.3.25	ICMP	71	Destination unreachable (Port unreachable)
4336	162.453124988	10.0.3.22	10.0.3.25	TCP	60	50116 → 445 [ACK] Seq=204 Ack=472 Win=2101760 Len=0



▶ Frame 4288: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface
 ▶ Ethernet II, Src: PCSSystemtec_id:ae:28 (08:00:27:1d:ae:28), Dst: PCSSystemtec_id:ae:28 (08:00:27:1d:ae:28)
 ▶ Internet Protocol Version 4, Src: 10.0.3.22, Dst: 10.0.3.25
 ▶ Internet Control Message Protocol
 Type: 3 (Destination unreachable)
 Code: 3 (Port unreachable)
 Checksum: 0x1746 [correct]
 [Checksum Status: Good]
 Unused: 00000000
 ▶ Internet Protocol Version 4, Src: 10.0.3.22, Dst: 10.0.3.25
 ▶ User Datagram Protocol, Src Port: 60696, Dst Port: 1434
 ▶ Data (1 byte)

eth0: <live capture in progress> Paquetes: 4336 · Mostrado: 2305 (53.2%) Perfil: Default

5. Http

http://www.bopsantacruzdetenerife.es

← ↻ No seguro | www.bopsantacruzdetenerife.es/bopsc2/index.php

 **BOLETIN OFICIAL**
DE LA PROVINCIA DE SANTA CRUZ DE TENERIFE 

Inicio Archivos Buscar Suscripción Contacto Editorial Ayuda **Sábado, 23 de marzo de 2024** 09 : 40 : 27

Marzo 2024
L M X J V S D
1 2 3
4 5 6 7 8 9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31


Febrero 2024
L M X J V S D
1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29

Sumario del último Boletín, número 36, de fecha 22-3-24

Descargar Boletín




II. ADMINISTRACIÓN DE LA COMUNIDAD AUTÓNOMA
CONSEJERÍA DE TURISMO Y EMPLEO
- Convenio Colectivo de la empresa Clínica Vintersol
- Convenio Colectivo de la empresa Dravo, S.A.

III. ADMINISTRACIÓN LOCAL
CABILDO INSULAR DE TENERIFE
- Anuncio relativo a la relación de aprobados y oferta de puestos de trabajo en la convocatoria para la cobertura por personal laboral fijo, de cuatro plazas de Peón Agrícola, en ejecución de la Oferta de Empleo Público de 2016.
- Anuncio relativo a la publicación del aspirante que ha superado el proceso selectivo en la convocatoria pública para la cobertura, por personal laboral fijo, de una plaza de Técnico/a de Grado Medio (a extinguir), sujeta al proceso extraordinario de estabilización de empleo temporal de larga duración.
- Anuncio relativo a información pública de la Carrera de Montaña Tenerife Bluetrail 2024 en el Parque Nacional del Teide.
INSTITUTO INSULAR DE ATENCIÓN SOCIAL Y SOCIO SANITARIA (IASIS)
- Anuncio relativo a la aprobación de la relación definitiva de aspirantes excluidas, en la convocatoria pública para la cobertura de plazas con carácter fijo, mediante el sistema de concurso y por el turno de acceso libre, de seis (6) plazas de Médico/a Adjunto/a, Grupo A1, vacantes en la plantilla de Personal Laboral del Organismo Autónomo IASIS, derivadas del proceso extraordinario de estabilización de empleo temporal de larga duración e incluidas en la Oferta de Empleo Público del Instituto del año 2022.
ORGANISMO AUTÓNOMO DE MUSEOS Y CENTROS (ISLA DE TENERIFE)
- Anuncio relativo a la convocatoria pública aprobada por Decreto de la Presidencia nº 21/24, de 18 de marzo de 2024, por el que se aprueban las Bases específicas para la provisión, mediante concurso, de los puestos de trabajo vacantes pertenecientes a las plazas de Auxiliar Administrativo/a de la Plantilla de Personal Laboral en la relación de puestos de trabajo del Organismo Autónomo de Museos y Centros del Cabildo Insular de Tenerife.
CABILDO INSULAR DE EL HIERRO
- Anuncio relativo a modificación de la composición de la Mesa de Contratación Permanente del Cabildo Insular de El Hierro (órgano de asistencia de la Presidencia como órgano de contratación).

¿Como adquirir?
BUSCAR UN NUMERO
2024 
Busqueda Avanzada

Haciendo ping a bopsantacruzdetenerife.es [91.126.176.238] con 32 bytes de datos:
Respuesta desde 91.126.176.238: bytes=32 tiempo=61ms TTL=52
Respuesta desde 91.126.176.238: bytes=32 tiempo=60ms TTL=52
Respuesta desde 91.126.176.238: bytes=32 tiempo=60ms TTL=52
Respuesta desde 91.126.176.238: bytes=32 tiempo=61ms TTL=52

Estadísticas de ping para 91.126.176.238:
Paquetes: enviados = 4, recibidos = 4, perdidos = 0
(0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
Mínimo = 60ms, Máximo = 61ms, Media = 60ms

tcp.stream eq 4

No.	Time	Source	Destination	Protocol	Length	Info
88	15.298249271	10.0.3.22	91.126.176.238	TCP	60	50256 → 80 [ACK] Seq=1496 Ack=4850 Win=64240 Len=0
89	15.298249350	91.126.176.238	10.0.3.22	HTTP	400	HTTP/1.1 200 OK (text/html)
97	15.342288806	10.0.3.22	91.126.176.238	TCP	60	50256 → 80 [ACK] Seq=1496 Ack=5196 Win=63894 Len=0
98	15.345301387	10.0.3.22	91.126.176.238	HTTP	623	GET /bopsc2/cab.php HTTP/1.1
107	15.409577576	91.126.176.238	10.0.3.22	TCP	4434	80 → 50256 [ACK] Seq=5196 Ack=2065 Win=32199 Len=4380 [TCP segment of a ...
108	15.409780741	10.0.3.22	91.126.176.238	TCP	60	50256 → 80 [ACK] Seq=2065 Ack=9576 Win=64240 Len=0
109	15.409780816	91.126.176.238	10.0.3.22	HTTP	452	HTTP/1.1 200 OK (text/html)

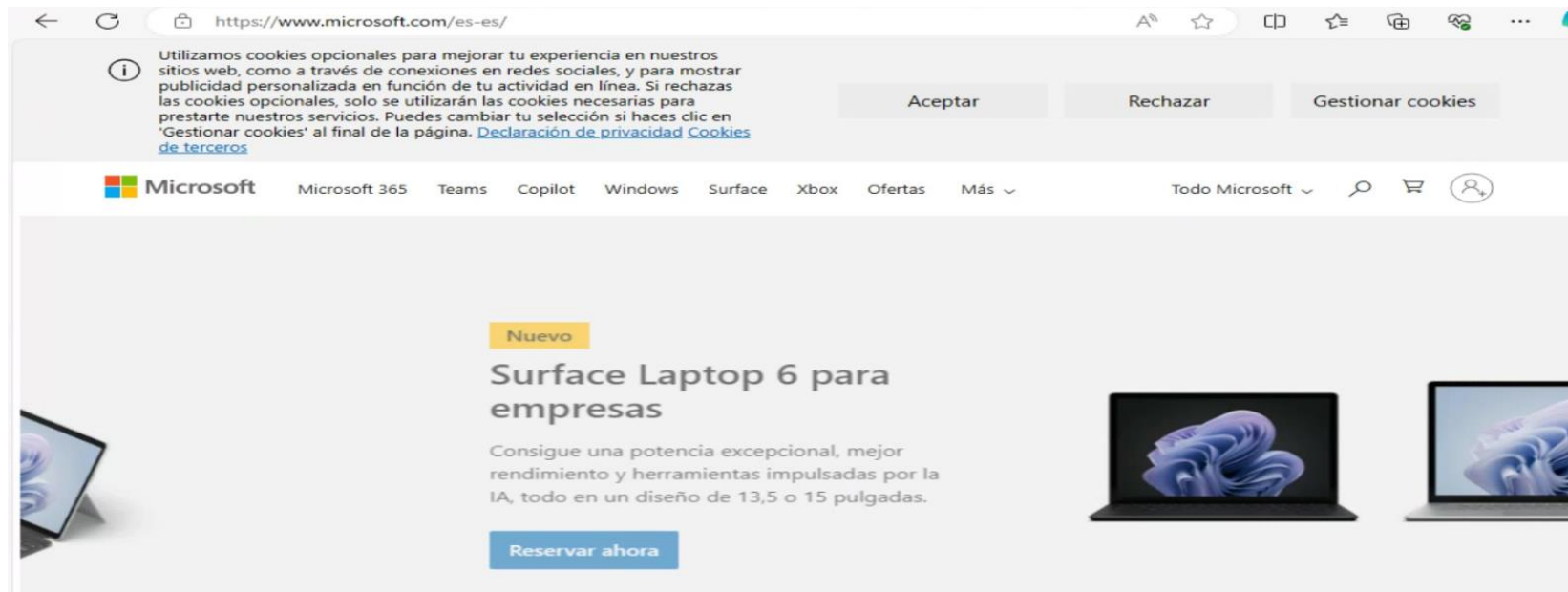
▶ Frame 98: 623 bytes on wire (4984 bits), 623 bytes captured (4984 bits) on interface
 ▶ Ethernet II, Src: PCSSystemtec_03:92:75 (08:00:27:03:92:75), Dst: 52:54:00:12:35:00
 ▶ Internet Protocol Version 4, Src: 10.0.3.22, Dst: 91.126.176.238
 ▶ Transmission Control Protocol, Src Port: 50256, Dst Port: 80, Seq: 1496, Ack: 5196, Win: 63894, Len: 623
 ▶ Hypertext Transfer Protocol
 ▶ GET /bopsc2/cab.php HTTP/1.1\r\n
 Host: www.bopsantacruzdetenerife.es\r\n
 Connection: keep-alive\r\n
 Upgrade-Insecure-Requests: 1\r\n
 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36 Edg/122.0.0.0\r\n
 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9\r\n
 Referer: http://www.bopsantacruzdetenerife.es/bopsc2/index.php\r\n
 Accept-Encoding: gzip, deflate\r\n
 Accept-Language: es,es-ES;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6\r\n
 \r\n
 [Full request URI: http://www.bopsantacruzdetenerife.es/bopsc2/cab.php]
 [HTTP request 4/5]
 [Prev request in frame: 82]
 [Response in frame: 109]
 [Next request in frame: 116]

0030 f9 96 ae ff 00 00 47 45 54 20 2f 62 6f 70 73 63 ... GET /bopsc
 0040 32 2f 63 61 62 2e 70 68 70 20 48 54 54 50 2f 31 2/cab.php HTTP/1
 0050 2e 31 0d 0a 48 6f 73 74 3a 20 77 77 77 2e 62 6f .1 Host: www.bo
 0060 70 73 61 6e 74 61 63 72 75 7a 64 65 74 65 6e 65 psantacruzdetene
 0070 72 69 66 65 2e 65 73 0d 0a 43 6f 6e 6e 65 63 74 rife.es Connect
 0080 69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 65 0d ion: keep-alive
 0090 0a 55 70 67 72 61 64 65 2d 49 6e 73 65 63 75 72 Upgrade-Insecur
 00a0 65 2d 52 65 71 75 65 73 74 73 3a 20 31 0d 0a 55 e-Requests: 1
 00b0 73 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c ser-Agent: Mozil
 00c0 6c 61 2f 35 2e 30 20 28 57 69 6e 64 6f 77 73 20 la/5.0 (Windows
 00d0 4e 54 20 31 30 2e 30 3b 20 57 69 6e 36 34 3b 20 NT 10.0; Win64;
 00e0 78 36 34 29 20 41 70 70 6c 65 57 65 62 4b 69 74 x64) AppleWebKit
 00f0 2f 35 33 37 2e 33 36 20 28 4b 48 54 4d 4c 2c 20 /537.36 (KHTML,
 0100 6c 69 6b 65 20 47 65 63 6b 6f 29 20 43 68 72 6f like Gecko) Chro
 0110 6d 65 2f 31 32 32 2e 30 2e 30 2e 30 20 53 61 66 me/122.0.0.0 Saf
 0120 61 72 69 2f 35 33 37 2e 33 36 20 45 64 67 2f 31 ari/537.36 Edg/1
 0130 32 32 2e 30 2e 30 2e 30 0d 0a 41 63 63 65 70 74 22.0.0.0 Accept
 0140 3a 20 74 65 78 74 2f 68 74 6d 6c 2c 61 70 70 6c : text/html,appl
 0150 69 63 61 74 69 6f 6e 2f 78 68 74 6d 6c 2b 78 6d ication/xhtml+xml
 0160 6c 2c 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 6d l,application/xm
 0170 6c 3b 71 3d 30 2e 39 2c 69 6d 61 67 65 2f 61 76 l;q=0.9, image/av
 0180 69 66 2c 69 6d 61 67 65 2f 77 65 62 70 2c 69 6d if,image/webp,im
 0190 61 67 65 2f 61 70 6e 67 2c 2a 2f 2a 3b 71 3d 30 age/apng,*/*;q=0
 01a0 2e 38 2c 61 70 70 6c 69 63 61 74 69 6f 6e 2f 73 .8,application/s
 01b0 69 67 6e 65 64 2d 65 78 63 68 61 6e 67 65 3b 76 igned-exchange;v

6. Https

<https://www.microsoft.es>

```
Haciendo ping a microsoft.es [20.112.250.133] con 32 bytes de datos:  
Respuesta desde 20.112.250.133: bytes=32 tiempo=175ms TTL=109  
Respuesta desde 20.112.250.133: bytes=32 tiempo=175ms TTL=109  
Respuesta desde 20.112.250.133: bytes=32 tiempo=174ms TTL=109  
Respuesta desde 20.112.250.133: bytes=32 tiempo=175ms TTL=109  
  
Estadísticas de ping para 20.112.250.133:  
Paquetes: enviados = 4, recibidos = 4, perdidos = 0  
(0% perdidos),  
Tiempos aproximados de ida y vuelta en milisegundos:  
Mínimo = 174ms. Máximo = 175ms. Media = 174ms
```



tls && ip.addr==20.112.250.133

No.	Time	Source	Destination	Protocol	Length	Info
192	12.181862016	10.0.3.22	20.112.250.133	TLSv1.3	586	Client Hello (SNI=microsoft.es)
198	12.199429016	10.0.3.22	20.112.250.133	TLSv1.3	650	Client Hello (SNI=microsoft.es)
211	12.356310568	20.112.250.133	10.0.3.22	TLSv1.3	1464	Server Hello, Change Cipher Spec, Application Data
215	12.358412036	20.112.250.133	10.0.3.22	TLSv1.3	5894	Application Data
217	12.358934340	20.112.250.133	10.0.3.22	TLSv1.3	199	Application Data, Application Data
219	12.362596871	10.0.3.22	20.112.250.133	TLSv1.3	118	Change Cipher Spec, Application Data
220	12.362596977	10.0.3.22	20.112.250.133	TLSv1.3	146	Application Data
222	12.362828841	10.0.3.22	20.112.250.133	TLSv1.3	555	Application Data
226	12.380396118	20.112.250.133	10.0.3.22	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data
229	12.383129613	20.112.250.133	10.0.3.22	TLSv1.3	4434	Application Data
231	12.383649649	20.112.250.133	10.0.3.22	TLSv1.3	149	Application Data, Application Data
233	12.384233794	10.0.3.22	20.112.250.133	TLSv1.3	118	Change Cipher Spec, Application Data
238	12.532864226	20.112.250.133	10.0.3.22	TLSv1.3	532	Application Data, Application Data
239	12.555274313	20.112.250.133	10.0.3.22	TLSv1.3	532	Application Data, Application Data
241	12.582924386	20.112.250.133	10.0.3.22	TLSv1.3	327	Application Data, Application Data, Application Data, Application Data
242	12.584588374	10.0.3.22	20.112.250.133	TLSv1.3	85	Application Data
2660	193.939726433	20.112.250.133	10.0.3.22	TLSv1.3	93	Application Data

Content Type: Handshake (22)
Version: TLS 1.0 (0x0301)
Length: 591
Handshake Protocol: Client Hello
Handshake Type: Client Hello (1)
Length: 587
Version: TLS 1.2 (0x0303)
Random: adb8350f458331c91a25b6514e245d166b7961863a2820890c541a46ed9f3
Session ID Length: 32
Session ID: 60ed2b3d3a413c936ff49c028a4e941dac7d0f51daa42f3426b7131b
Cipher Suites Length: 32
Cipher Suite: Reserved (GREASE) (0xbaba)
Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)

Paquetes: 2664 - Mostrado: 17 (0.6%)

Wireshark - Flow - eth0

Intervalo	10.0.3.22	91.126.176.238	185.43.181.50	10.0.3.23	Comentario
0.000000000	50267	50267 → 80 [ACK] Seq=1 Ack=1 Win=642	80		TCP: 50267 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=1
0.000010677	50266	50266 → 80 [ACK] Seq=1 Ack=1 Win=642	80		TCP: 50266 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=1
0.000010700	50267	80 → 50267 [ACK] Seq=1 Ack=2 Win=327...	80		TCP: 80 → 50267 [ACK] Seq=1 Ack=2 Win=32768 Len=0
0.000010719	50266	80 → 50266 [ACK] Seq=1 Ack=2 Win=327...	80		TCP: 80 → 50266 [ACK] Seq=1 Ack=2 Win=32768 Len=0
4.123896720	50266	50266 → 80 [FIN, ACK] Seq=2 Ack=1 Win...	80		TCP: 50266 → 80 [FIN, ACK] Seq=2 Ack=1 Win=64240 ...
4.123897074	50266	80 → 50266 [ACK] Seq=1 Ack=3 Win=327...	80		TCP: 80 → 50266 [ACK] Seq=1 Ack=3 Win=32767 Len=0
4.124817144	50267	50267 → 80 [FIN, ACK] Seq=2 Ack=1 Win...	80		TCP: 50267 → 80 [FIN, ACK] Seq=2 Ack=1 Win=64240 ...
4.124817341	50254	50254 → 443 [FIN, ACK] Seq=1 Ack=1 Win=63718 Len=0	443		TCP: 50254 → 443 [FIN, ACK] Seq=1 Ack=1 Win=63718 ...
4.124817364	50267	80 → 50267 [ACK] Seq=1 Ack=3 Win=327...	80		TCP: 80 → 50267 [ACK] Seq=1 Ack=3 Win=32767 Len=0
4.124817382	50254	50254 → 443 [RST, ACK] Seq=2 Ack=1 Win=0 Len=0	443		TCP: 50254 → 443 [RST, ACK] Seq=2 Ack=1 Win=0 Len=0
4.124817403	50254	443 → 50254 [ACK] Seq=1 Ack=2 Win=31678 Len=0	443		TCP: 443 → 50254 [ACK] Seq=1 Ack=2 Win=31678 Len=0
4.125650993	50268	50268 → 80 [SYN] Seq=0 Win=64240 Le...	80		TCP: 50268 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS...
4.125651112	50269	50269 → 80 [SYN] Seq=0 Win=64240 Le...	80		TCP: 50269 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS...
4.183224011	50268	80 → 50268 [SYN, ACK] Seq=0 Ack=1 Win...	80		TCP: 80 → 50268 [SYN, ACK] Seq=0 Ack=1 Win=32768...
4.183628238	50268	50268 → 80 [ACK] Seq=1 Ack=1 Win=642...	80		TCP: 50268 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
4.192916396	50269	80 → 50269 [SYN, ACK] Seq=0 Ack=1 Win...	80		TCP: 80 → 50269 [SYN, ACK] Seq=0 Ack=1 Win=32768...