Wireshark

Guía de uso de Wireshark - Parte 1: Descripción, Instalación y Configuración

1. ¿Qué es Wireshark?

Wireshark es una herramienta de análisis de protocolos de red (network protocol analyzer) que permite capturar y examinar los paquetes de datos que circulan en una red. Es muy utilizada para solucionar problemas de red, análisis de seguridad, depuración de protocolos, y más. Wireshark permite visualizar cada paquete capturado de forma detallada, lo que la convierte en una herramienta indispensable para administradores de redes y expertos en seguridad.

Principales características de Wireshark:

Captura en tiempo real de paquetes de red.

Análisis profundo de protocolos.

Visualización detallada y filtrada de tráfico.

Posibilidad de guardar, analizar y compartir capturas.

Soporta cientos de protocolos y se actualiza constantemente para incorporar nuevos.

2. Instalación de Wireshark

a. Instalación en Windows:

1. **Descarga Wireshark** desde el sitio oficial: **Wireshark**.
2. Ejecuta el archivo `**.exe**` descargado.
3. Durante la instalación, asegúrate de marcar la opción para instalar **Npcap**, ya que este controlador permite la captura de paquetes en Windows.
4. Completa la instalación siguiendo las instrucciones del asistente.
5. Una vez finalizada la instalación, abre Wireshark.

b. Instalación en Linux (Debian/Ubuntu):

1. Abre una terminal.
2. Ejecuta el siguiente comando para actualizar los repositorios e instalar Wireshark:

```bash
sudo apt update
sudo apt install wireshark
```

3. Durante la instalación, se te preguntará si deseas permitir a los usuarios no root capturar paquetes. Responde "Sí" si quieres permitirlo.
4. Añade tu usuario al grupo `**wireshark**` para permitir la captura sin privilegios de root:

```bash
sudo usermod -aG wireshark $USER
```

5. Cierra la sesión y vuelve a iniciarla para que los cambios surtan efecto. c. Instalación en macOS:

1. **Instala Homebrew** si no lo tienes instalado. En la terminal, ejecuta:

```bash
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

2. Luego, instala Wireshark con el siguiente comando:

```bash
brew install wireshark
```

3. Si deseas permitir la captura sin privilegios root, sigue las instrucciones que Homebrew proporciona después de la instalación.

3. Configuración de Wireshark

Una vez que Wireshark está instalado, el siguiente paso es configurarlo para capturar tráfico de red de manera eficiente.

a. Interfaz de red

Cuando inicies Wireshark, lo primero que verás es una lista de las interfaces de red disponibles en tu sistema, como la conexión Ethernet o Wi-Fi.

Selecciona la interfaz correcta: En la pantalla inicial, selecciona la interfaz de red que deseas monitorizar. Esto es importante, ya que solo capturarás tráfico de la red correspondiente a la interfaz seleccionada.

Si estás en una red Wi-Fi, elige la interfaz Wi-Fi (usualmente mostrada como **`wlan0`** en Linux o **`Wi-Fi`** en Windows).

b. Filtros de Captura

Wireshark te permite capturar todo el tráfico en la red o solo paquetes que cumplan con ciertos criterios específicos. Utilizar filtros de captura es importante para no obtener una cantidad de datos excesiva y difícil de analizar.

Para establecer un filtro de captura:

1. En el campo de texto que aparece junto al botón de inicio de captura, escribe un filtro. Algunos ejemplos comunes son:

Capturar solo paquetes ICMP (Ping):

```bash
icmp
```

Capturar solo tráfico HTTP:

```bash
tcp port 80
```

Capturar solo tráfico de una IP específica:

```bash
host 192.168.1.100
```

c. Inicio y Detención de la Captura Para comenzar a capturar paquetes:

1. Selecciona la interfaz de red.
2. Haz clic en el botón de "Iniciar Captura" (ícono de tiburón verde en la barra de herramientas) o presiona `Ctrl + E`.

Wireshark comenzará a capturar todos los paquetes que circulan por la interfaz seleccionada.

Para detener la captura:

Haz clic en el botón de "Detener Captura" (ícono de un cuadro rojo) o presiona `**Ctrl + E**` nuevamente.

4. Uso Básico de Wireshark

Ahora que tienes Wireshark instalado y configurado, puedes comenzar a capturar tráfico en tu red local. Vamos a explorar cómo hacerlo con un ejemplo simple.

Ejemplo: Captura de un ping

Vamos a capturar paquetes ICMP (tráfico de ping) entre tu máquina y otro dispositivo en la red.

Pasos a seguir:

1. **Inicia la captura** en la interfaz correcta (por ejemplo, `**Wi-Fi**`).

2. Abre una terminal (Linux/macOS) o el símbolo del sistema (Windows) y envía un ping a un dispositivo en tu red. Por ejemplo:

```bash
ping 192.168.1.1
```

Esto enviará paquetes ICMP a la dirección IP `**192.168.1.1**`.

3. En Wireshark, verás que se empiezan a capturar paquetes. Si estás capturando mucho tráfico, puedes usar un **filtro de visualización** (que no afecta la captura, solo lo que se muestra en pantalla) para ver solo los paquetes ICMP:

```bash
icmp
```

4. Detén la captura después de unos segundos.

Resultados esperados:

En la ventana de Wireshark, deberías ver una lista de paquetes ICMP que corresponden al ping que ejecutaste.

Columna de Protocolo: Deberías ver la palabra "ICMP" en varios paquetes.

Columna de Información: Aquí verás información adicional, como "Echo (ping) request" y "Echo (ping) reply", que indican el envío y recepción del ping.

Puedes seleccionar cualquier paquete en la lista para ver más detalles. Abajo, en el panel de detalle de paquetes, verás la estructura de cada paquete, incluyendo encabezados IP y datos específicos de ICMP.

Guía de uso de Wireshark - Parte 2: Filtros, Análisis y Exportación de Capturas

En esta segunda parte, exploraremos cómo aplicar filtros avanzados para analizar el tráfico de red de manera eficiente, cómo interpretar los datos capturados, y cómo exportar y compartir capturas para análisis posterior.

1. Uso de Filtros en Wireshark

Wireshark permite aplicar filtros de visualización y de captura para aislar el tráfico de interés. Los **filtros de visualización** no afectan la captura en sí, sino que te permiten enfocar el análisis en ciertos paquetes una vez que la captura ha sido realizada. Los **filtros de captura** limitan los paquetes que se capturan desde el principio.

a. Filtros de Visualización

Los filtros de visualización se aplican en la barra de filtros, justo debajo de la barra de herramientas. Estos filtros son sumamente útiles para reducir la cantidad de datos que ves en la pantalla. Aquí tienes algunos filtros comunes:

Filtrar por protocolo:

Para mostrar solo paquetes de un protocolo específico, como HTTP, ICMP o TCP:

```bash
http
icmp
tcp
```

Filtrar por dirección IP específica:

Muestra solo el tráfico de una dirección IP de origen o destino.

```bash
ip.src == 192.168.1.10
ip.dst == 192.168.1.100
```

Filtrar por puerto:

Muestra solo paquetes relacionados con un puerto específico.

```bash
tcp.port == 80  # Puerto 80 (HTTP) udp.port
== 53  # Puerto 53 (DNS)
```

Filtrar por direcciones MAC:

Para analizar tráfico basado en direcciones MAC:

```bash
eth.src == 00:1A:2B:3C:4D:5E eth.dst
== 00:1F:2C:3B:4A:5D
```

b. Filtros de Captura

A diferencia de los filtros de visualización, los filtros de captura limitan los paquetes que Wireshark captura desde el inicio. Se aplican antes de iniciar la captura, en el campo de "Filtros de Captura". Algunos ejemplos comunes:

Capturar solo tráfico de una dirección IP específica:

```bash
host 192.168.1.10
```

Capturar solo tráfico en un puerto:

```bash
port 80
```

Capturar solo paquetes TCP o UDP:

```bash
tcp
udp
```

2. Análisis de Paquetes Capturados

Una vez que tienes una captura, el siguiente paso es analizar el contenido de los paquetes. Wireshark organiza la información capturada en tres paneles: la lista de paquetes, los detalles del paquete seleccionado, y la visualización de los datos sin procesar.

a. Lista de Paquetes

Cada línea en la lista de paquetes representa un paquete individual. Las columnas más importantes son:

No.: Número del paquete capturado.

Time: Hora en que se capturó el paquete.

Source: Dirección IP o MAC de origen.

Destination: Dirección IP o MAC de destino.

Protocol: Protocolo utilizado (TCP, UDP, ICMP, HTTP, etc.).

Length: Longitud del paquete en bytes.

Info: Información adicional del paquete, como el tipo de mensaje ICMP o el puerto de conexión TCP.

b. Detalles del Paquete

Selecciona un paquete para ver los detalles en el panel intermedio. Aquí verás una representación jerárquica del paquete, que incluye:

Ethernet Header: Detalles sobre la capa de enlace de datos (direcciones MAC de origen y destino).

IP Header: Información sobre la capa de red (dirección IP de origen y destino, tiempo de vida (TTL), etc.).

Protocolo: Información sobre el protocolo específico (TCP, UDP, ICMP, HTTP, etc.). c. Datos en bruto (Hexadecimal)

En el panel inferior, puedes ver el paquete en formato hexadecimal y ASCII, que muestra la información tal como fue transmitida por la red. Este nivel de detalle es útil para análisis avanzados o depuración de protocolos.

3. Interpretación de Paquetes y Protocolos

Veamos algunos ejemplos de cómo interpretar los paquetes más comunes en una red. a. Análisis de Paquetes ICMP (Ping)

Si capturaste tráfico ICMP (como en el ejemplo de la Parte 1 con el comando **`ping`**), puedes ver dos tipos de mensajes:

Echo Request (Solicitud de eco): Es el mensaje enviado por el comando **`ping`** para probar la conectividad con el host destino.

Echo Reply (Respuesta de eco): Es la respuesta del host destino indicando que ha recibido el ping.

En los detalles del paquete ICMP, verás algo como esto:

Type: 8 (Echo Request) o 0 (Echo Reply).

Checksum: Verifica la integridad del mensaje.

b. Análisis de Tráfico HTTP

Si capturaste tráfico HTTP (por ejemplo, al navegar por una página web), los paquetes TCP asociados con el protocolo HTTP incluirán solicitudes y respuestas HTTP.

HTTP Request: Verás información sobre la solicitud de un navegador web, como el método (GET, POST), la URL solicitada, y los encabezados HTTP.

HTTP Response: Aquí se incluye el código de estado de la respuesta (200 OK, 404 Not Found), junto con los encabezados y posiblemente el contenido de la página.

En los detalles del paquete HTTP, verás:

Host: La dirección web solicitada.

User-Agent: El navegador o cliente utilizado para hacer la solicitud.

Content-Type: Tipo de contenido de la respuesta (HTML, imagen, etc.). c. Análisis de Paquetes TCP

Los paquetes TCP son fundamentales para el análisis de conexiones en red. Un ejemplo clásico es el **proceso de establecimiento de una conexión TCP de tres vías**:

1. **SYN (Synchronization)**: El cliente envía un paquete SYN al servidor para iniciar una conexión.
2. **SYN-ACK**: El servidor responde con un SYN-ACK.
3. **ACK**: El cliente envía un paquete ACK para completar el establecimiento de la conexión.

Puedes identificar estos paquetes buscando los campos **SYN**, **ACK** y **FIN** en los detalles del paquete.

4. **Exportación y Compartición de Capturas**

Wireshark permite exportar los resultados de las capturas para su análisis posterior o para compartir con otros. Aquí te explico cómo hacerlo:

a. Guardar Capturas

Una vez que has terminado de capturar el tráfico, puedes guardar los paquetes capturados en un

archivo `**.pcap**`, que es el formato estándar para capturas de red.

1. Ve a **File > Save As**.
2. Elige la ubicación donde deseas guardar el archivo y asegúrate de que el formato seleccionado sea `.pcap`.
3. Haz clic en **Guardar**.

b. Exportar Paquetes Específicos

También puedes exportar solo ciertos paquetes de interés:

1. Aplica un filtro de visualización para aislar los paquetes que deseas exportar.
2. Ve a **File > Export Specified Packets**.
3. Elige si deseas exportar todos los paquetes mostrados o solo aquellos seleccionados.
4. Guarda el archivo en formato `**.pcap**`.

c. Exportar como Texto o CSV

Para reportes o análisis más simples, puedes exportar la captura en formato de texto o CSV:

1. Ve a **File > Export Packet Dissections > As Plain Text** o **As CSV**.
2. Elige los detalles que deseas incluir y guarda el archivo.

Ejemplo Práctico: Captura de una Solicitud HTTP

Escenario: Imagina que estás capturando tráfico en una red y deseas analizar una solicitud HTTP realizada desde tu navegador.

1. Inicia la captura en Wireshark.
2. Abre tu navegador y visita un sitio web sencillo, como `**http://example.com**`.
3. Detén la captura en Wireshark.
4. Aplica un filtro de visualización para mostrar solo tráfico HTTP:

```bash
http
```

5. Examina los paquetes filtrados. Deberías ver varios paquetes HTTP. Selecciona uno de los paquetes de solicitud HTTP (`**GET**`) para ver los detalles de la solicitud:

Host: `example.com`

User-Agent: Detalles sobre tu navegador.

GET / HTTP/1.1: La solicitud para obtener la página de inicio.

6. Examina también la respuesta HTTP desde el servidor, que debería incluir un código de estado HTTP (200 OK) y posiblemente el contenido HTML de la página.

Guía de uso de Wireshark - Parte 3: Funciones Avanzadas y Herramientas de Análisis

En esta última parte de la guía, exploraremos las características avanzadas de Wireshark que te permitirán realizar un análisis más profundo del tráfico de red. Veremos cómo generar gráficos, analizar flujos TCP, usar estadísticas, y cómo aprovechar algunas herramientas adicionales para mejorar tu análisis.

1. Análisis de Flujos TCP

El análisis de flujos TCP es una de las funciones más potentes de Wireshark, ya que permite ver todo el intercambio de paquetes entre dos puntos finales de una conexión TCP (cliente y servidor). Esto es especialmente útil para solucionar problemas relacionados con la conectividad o el rendimiento. a. ¿Qué es un flujo TCP?

Un flujo TCP representa la secuencia completa de paquetes entre dos hosts durante el tiempo que dura una conexión. Wireshark puede ensamblar y mostrar este intercambio de manera continua, ayudando a identificar problemas como retrasos, retransmisiones o pérdida de paquetes. b. Cómo seguir un flujo TCP

1. Inicia Wireshark y captura tráfico TCP (puedes usar un filtro de captura como **`tcp`** para concentrarte en paquetes TCP).
2. Detén la captura una vez que hayas reunido suficientes datos.
3. Identifica un paquete relacionado con la conexión que deseas analizar (por ejemplo, una solicitud HTTP o un paquete de inicio de sesión SSH).
4. Haz clic derecho en el paquete y selecciona **"Follow" > "TCP Stream"**.

Wireshark abrirá una nueva ventana con el flujo TCP completo, mostrando el tráfico en texto claro, organizado cronológicamente entre el cliente y el servidor. Aquí verás el contenido de los mensajes, lo que te permite analizar cómo se comporta la conexión.

Color Rojo: Datos enviados por el cliente.

Color Azul: Datos enviados por el servidor.

c. Análisis de retransmisiones y pérdida de paquetes

En la ventana principal de Wireshark, puedes identificar problemas de retransmisión o pérdida de paquetes buscando ciertos indicadores en la columna "Info", como:

[TCP Retransmission]: Esto indica que un paquete TCP ha sido retransmitido, lo que puede significar congestión o problemas en la red.

[TCP Dup ACK]: Un "Acknowledgment" duplicado puede indicar la pérdida de paquetes.

Estos eventos suelen estar relacionados con problemas de rendimiento de la red.

2. Generación de Gráficos y Estadísticas

Wireshark no solo captura y analiza paquetes, sino que también permite generar gráficos y estadísticas útiles para visualizar el tráfico de red.

a. Gráficos de tiempo (IO Graphs)

Los gráficos de entrada/salida (Input/Output Graphs) permiten visualizar el volumen de tráfico a lo largo del tiempo. Son especialmente útiles para identificar picos o anomalías en el tráfico de red.

1. Ve a **Statistics > IO Graph**.
2. Aparecerá una ventana con un gráfico que muestra el tráfico en función del tiempo.
3. Puedes ajustar el intervalo de tiempo y aplicar filtros para concentrarte en tipos de tráfico específicos.

Por ejemplo, puedes crear un gráfico que muestre solo el tráfico HTTP aplicando el filtro **`http`** en uno de los gráficos disponibles.

b. Flujo de protocolos (Protocol Hierarchy)

Esta opción te muestra un desglose jerárquico de todos los protocolos capturados en una sesión, lo que te permite entender qué protocolos están consumiendo más ancho de banda.

1. Ve a **Statistics > Protocol Hierarchy**.
2. Verás una lista de todos los protocolos, organizados jerárquicamente (capa de enlace, capa de red, capa de transporte, etc.).
3. Para cada protocolo, se muestra el número de paquetes y el porcentaje de tráfico que representa.

c. Conversaciones y puntos finales (Conversations and Endpoints)

Wireshark también puede mostrar las conversaciones activas entre hosts (intercambio de paquetes entre direcciones IP específicas) y los puntos finales (endpoints), que son las direcciones IP únicas que han sido capturadas.

Conversaciones: Ve a **Statistics > Conversations**. Aquí puedes ver todas las conversaciones (comunicaciones entre dos IPs) para protocolos como Ethernet, IP, TCP, UDP, etc.

Puntos finales: Ve a **Statistics > Endpoints**. Esta opción te muestra todas las direcciones IP capturadas, junto con estadísticas como el número de paquetes enviados y recibidos por cada una.

3. Usando Capturas de Paquetes para Detectar Problemas de Seguridad

Wireshark es ampliamente utilizado en seguridad informática para detectar anomalías o posibles ataques en la red. Aquí te mostramos algunos ejemplos de cómo identificar problemas de seguridad comunes.

a. Detección de escaneos de puertos

Los escaneos de puertos son uno de los métodos más comunes utilizados por atacantes para descubrir servicios vulnerables en un sistema. Puedes identificarlos en Wireshark buscando patrones anormales en las conexiones.

1. Captura tráfico de red y busca múltiples intentos de conexión a diferentes puertos desde una misma dirección IP.
2. Aplica el filtro:

```bash
tcp.flags.syn == 1 and tcp.flags.ack == 0
```

Esto mostrará solo los paquetes SYN, que son usados en el inicio de las conexiones TCP. Si ves muchos SYN desde la misma IP a diferentes puertos, esto puede ser un indicio de escaneo. b. Detección de ataques de denegación de servicio (DoS)

Un ataque de denegación de servicio (DoS) intenta abrumar un sistema enviando una gran cantidad de tráfico. Para detectar posibles ataques DoS, busca patrones de tráfico inusualmente altos desde una o varias IPs.

Ve a **Statistics > IO Graphs** y observa los picos de tráfico.

Identifica las IPs responsables de generar grandes volúmenes de tráfico en **Statistics > Endpoints**.

c. Análisis de tráfico cifrado y no cifrado

Otra preocupación de seguridad es si el tráfico sensible está siendo transmitido sin cifrar. Puedes usar Wireshark para verificar si las comunicaciones están usando protocolos seguros como HTTPS o TLS.

1. Captura tráfico en una red donde sepas que se están realizando transacciones sensibles (como inicios de sesión o transferencias de archivos).

2. Filtra por `**http**` para ver si hay tráfico HTTP no cifrado.

3. Si encuentras tráfico HTTP, examina los detalles del paquete para verificar si se están transmitiendo credenciales o información sensible sin cifrado.

4. Captura y Análisis de Tráfico en Redes Inalámbricas (Wi-Fi)

La captura de tráfico en redes Wi-Fi es algo más compleja debido a la naturaleza del medio inalámbrico. Sin embargo, Wireshark puede capturar paquetes de redes Wi-Fi si se configura correctamente.

a. Configuración de la captura en Wi-Fi

1. Asegúrate de que tu tarjeta de red inalámbrica admite el modo de monitor, que es necesario para capturar tráfico en redes Wi-Fi.

2. En sistemas Linux, usa el siguiente comando para activar el modo de monitor:

```bash
sudo iwconfig wlan0 mode monitor
```

En Windows, Wireshark usa el controlador Npcap para habilitar este modo en ciertas tarjetas de red.

b. Filtros para redes inalámbricas

Wireshark tiene varios filtros que te permiten analizar tráfico en redes Wi-Fi, como:

Tráfico de gestión de Wi-Fi: Muestra paquetes relacionados con la gestión de la red inalámbrica, como solicitudes de asociación y desasociación.

```bash
wlan.fc.type_subtype == 0x00 or wlan.fc.type_subtype == 0x01
```

Tráfico de datos de Wi-Fi: Para analizar el tráfico de datos entre dispositivos.

```bash
wlan.fc.type == 0x02
```

5. Plugins y Scripts

Wireshark admite la extensión de su funcionalidad mediante plugins y scripts, escritos en Lua o Python, lo que permite automatizar tareas complejas o añadir soporte para nuevos protocolos. a. Uso de Lua Scripts

Lua es un lenguaje de scripting soportado por Wireshark que te permite escribir tus propios analizadores o filtros. Puedes cargar scripts Lua en Wireshark desde el menú **Edit > Preferences > Protocols > Lua**.

b. Herramientas complementarias

Además de Wireshark, existen herramientas que se integran bien con él para realizar análisis más avanzados:

TShark: Es la versión de línea de comandos de Wireshark, útil para capturas remotas o automatización.

tcpdump: Otra herramienta de línea de comandos para capturar tráfico de red, que puede exportar capturas en formato `**.pcap**` para analizar en Wireshark.

Ejercicio Práctico: Análisis de un Escaneo de Puertos

Escenario: Supongamos que has capturado tráfico en tu red y sospechas que ha habido un escaneo de puertos. Te gustaría confirmar si realmente se produjo el escaneo.

Pasos a seguir:

1. Inicia Wireshark y carga la captura de tráfico.
2. Usa el filtro de captura para mostrar solo paquetes SYN sin ACK:

```bash
tcp.flags.syn == 1 and tcp.flags.ack == 0
```

3. Examina los resultados. Si ves múltiples solicitudes SYN desde la misma dirección IP a varios puertos diferentes, es una fuerte indicación de un escaneo de puertos.
4. Para obtener más información, selecciona un paquete SYN y ve a **Statistics > Conversations**. Aquí puedes ver todas las direcciones IP y puertos involucrados en el escaneo.

Solución:

Este ejercicio te permite practicar cómo identificar patrones típicos de un ataque de escaneo de puertos. Al usar filtros específicos y herramientas de análisis, puedes confirmar la actividad sospechosa en la red.

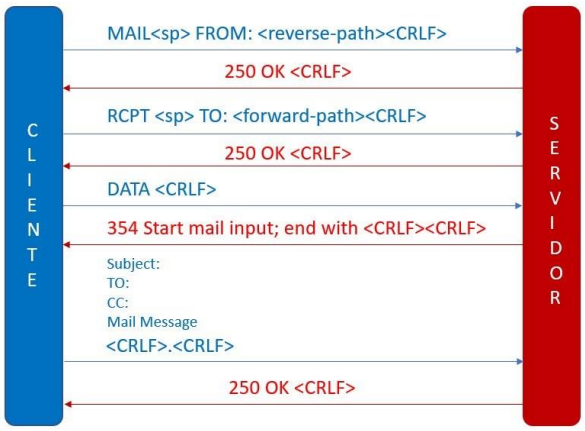
Uso básico de Wireshark - **Ejercicio práctico**

1. ANÁLISIS SMTP

El protocolo SMTP (Simple Mail Transfer Protocol), es un protocolo muy utilizado y asociado a un RFC que data de 1982. Este protocolo se usa para el envío de correo electrónico. La mayoría de los ataques maliciosos, están relacionados con los ficheros adjuntos que son enviados o los enlaces embebidos en el cuerpo del mensaje.

Como se puede apreciar en la siguiente imagen, el protocolo SMTP es un protocolo orientado a línea donde se intercambian mensajes entre el servidor (en rojo) y el cliente (en azul).

El protocolo orientado a línea es un protocolo donde se utiliza un retorno de carro y una línea nueva por cada comando intercambiado.



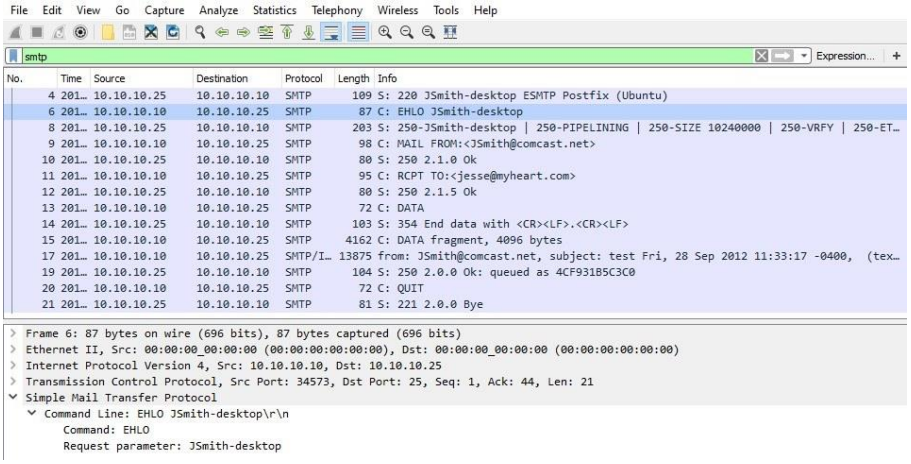
1 - Figura 1 Intercambio de Comandos SMTP

Una transacción SMTP empieza con el comando MAIL desde el cliente hasta el servidor, junto con el parámetro FROM donde se indica a donde se deben enviar los posibles fallos en el envío del correo. SMTP finaliza con un retorno de carro y salto de línea (CRLF). Si el comando MAIL se acepta por parte del servidor, el servidor responderá con el código «250 OK», y esto significa que esta todo correcto.

El siguiente paso es enviar el comando RCPT para identificar una o varias direcciones de correo de destinatario. Si el servidor SMTP acepta esto, responderá «250 OK», en caso contrario responderá con un «550 No such user here», indicando que no ha podido encontrar al destinatario.

El cliente de correo envía el comando DATA para indicar que el siguiente comando será el mensaje. El servidor responde con un «354 Start mail input, end with <CRLF>.<CRLF>», donde el servidor indica al cliente como debe terminar el mensaje, en nuestro caso una línea que contenga «.».Si todo es correcto el servidor devolverá un 250 OK.

Analicemos una captura que contiene una comunicación SMTP:

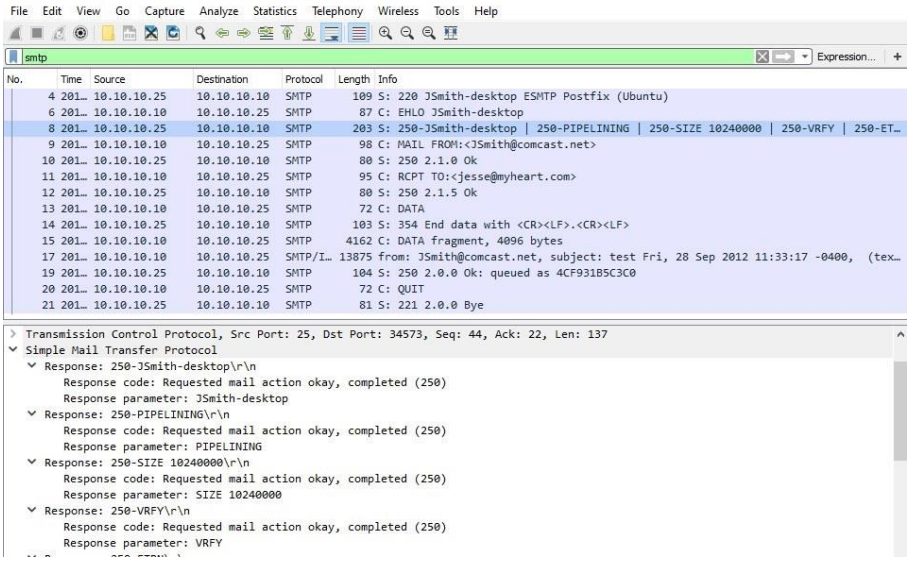


2 - Figura 2 Análisis de SMTP desde Wireshark

Como vemos en la anterior captura, la comunicación empieza previamente con un nuevo comando HELO o EHLO.

En la actualidad ambos se utilizan, pero al utilizar EHLO el servidor responde con características adicionales como PIPELINING, SIZE, HELP and

ENHACEDSTATUSCODES como se puede identificar en la siguiente captura:

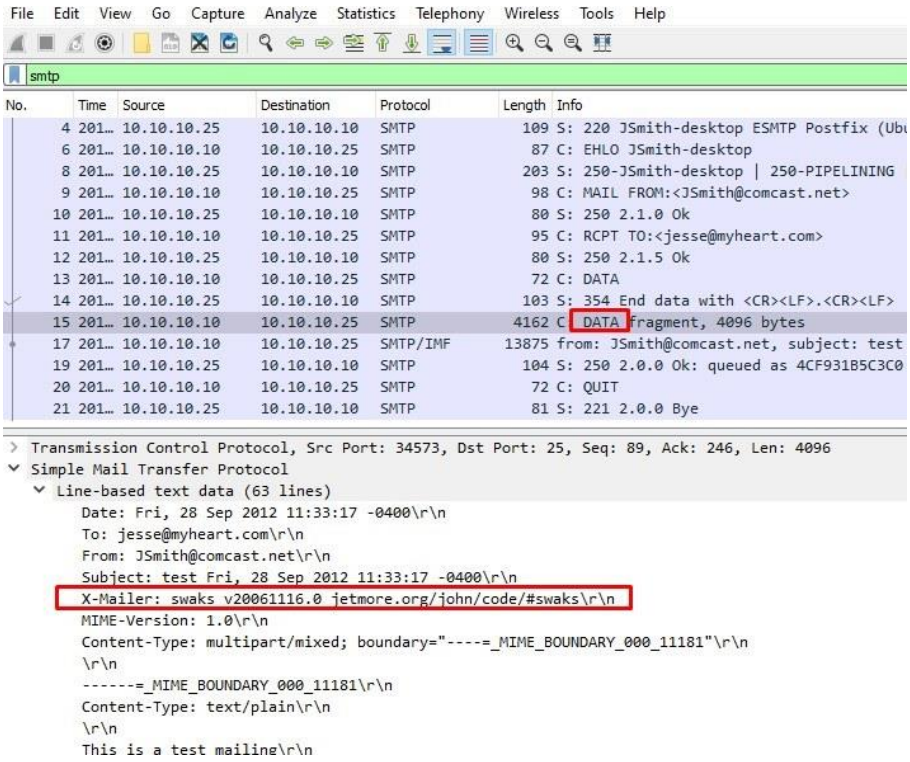


3 - Figura 3 SMTP en profundidad

La sesión comienza con un *three-way handshake* TCP hacia el servidor SMTP, y el servidor responde con un 220 como se puede apreciar en el paquete número 4. El cliente continúa con el mensaje, en el paquete número 6 con un EHLO identificándose como JSmithdesktop y el servidor responde con un «250 OK».

Una vez explicado cómo funciona SMTP te proponemos el siguiente ejercicio, dado la captura smtp_muestra.pcap ¿Sabías decirnos cual es cliente de correo electrónico que supuestamente fue utilizado para enviar el correo electrónico en cuestión?

Sabiendo que el cliente tiene que enviar el mensaje DATA con la información del correo electrónico, incluido las cabeceras, se podría obtener dicha información:



4 - Figura 4 Versión del cliente de correo utilizado

La respuesta sería: «X-Mailer: swaks v20061116.0 jetmore.org/john/code/#swaks»

2. ANÁLISIS HTTP

«Hypertext Transfer Protocol» (HTTP) es un protocolo mundialmente conocido donde inicialmente los ataques fueron perpetrados contra los servidores, pero hoy en día a menudo el objetivo es el navegador para comprometer el host del equipo.

Como veras, HTTP es un protocolo de formato simple, pero el formato de cuerpo de una petición o respuesta HTTP se puede complicar. Los servidores y navegadores son susceptibles de muchas vulnerabilidades y tipos de ataque. HTTP es un protocolo sin estado por que el servidor no mantiene el estado entre las transacciones realizadas en una sesión.

Una petición HTTP comienza con una «Start Line» o línea de empiece que incluye un método, una URL, la versión HTTP y termina con un retorno de carro y nueva línea (CRLF). Existen diferentes métodos HTTP donde el más usado es GET.

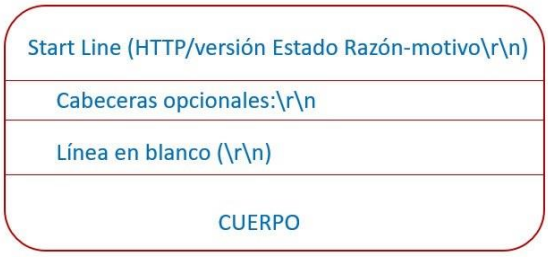
5 - Figura 5 Petición HTTP

Un método GET realiza peticiones de algún tipo de recurso o documento identificado por la URL del servidor. Otro tipo de método es POST, el cual envía datos hacia un servidor especificado por una URL. Puede haber cabeceras opcionales desde que contenido se acepta, idiomas o codificaciones y algunas también por seguridad. Después de las cabeceras podremos encontrar una línea den blanco con un CRLF y finalmente el cuerpo del mensaje.

HTTP y SMTP son protocolos orientados a línea, esto significa que el protocolo utiliza una nueva línea para delimitar los distintos elementos que conforman la petición.

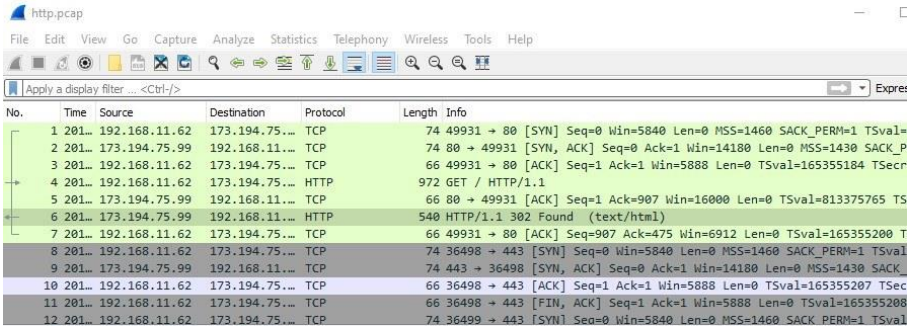
El formato de la respuesta HTTP no es muy diferente al de petición. La única diferencia notable es que la línea de comienzo es la versión, código de estado y la razón-motivo. El campo versión nos indica la versión soportada por el servidor HTTP. El código de estado son 3 dígitos que indican el resultado de la petición. El primer digito indica la clase de códigos tales como éxito o error. El campo razón-motivo indica de manera explícita lo que indica el código de estado. Finalmente, la línea de empiece debe finalizar con CRLF.

Las cabeceras son opcionales de nuevo, pero la mayoría de los servidores las incluyen. Estas deben ser seguidas por una línea en blanco y el cuerpo que es opcional.



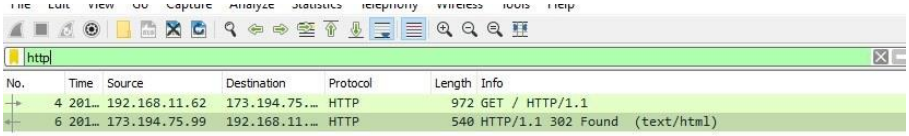
6 - Figura 6 Mensaje HTTP

Dando el fichero http.cap ¿podrías indicarnos que cual sería el servidor involucrado en la petición, es decir, el tipo de servidor utilizado?



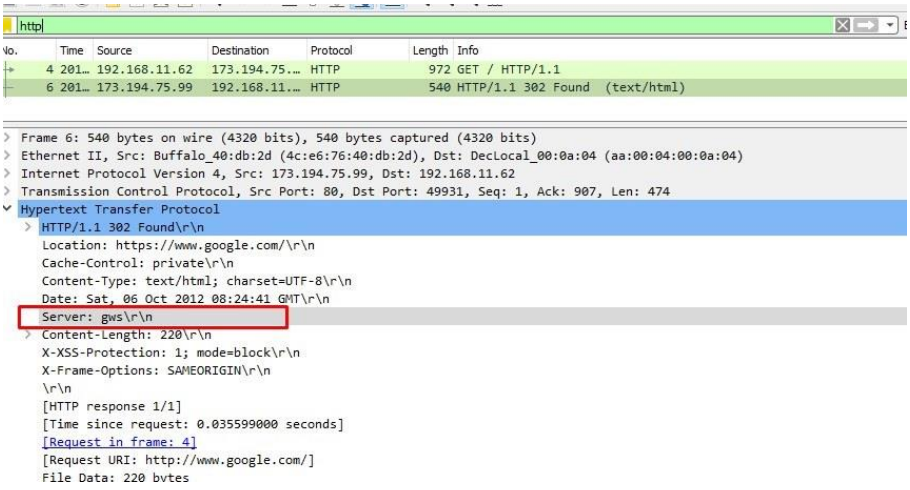
7 - **Figura 7 Análisis HTTP desde Wireshark**

El primer paso sería filtrar por el protocolo HTTP:



8 - Figura 8 Filtro HTTP

Una vez filtrado nos quedarían dos paquetes HTTP donde deberemos analizar la respuesta del servidor. Abrimos el paquete número 6 podemos ver las cabeceras de la respuesta:



9 - **Figura 9 Ampliación sobre la pila de protocolos**

Finalmente podemos ver en las cabeceras el servidor «gws».

Práctica con WireShark

Ejercicio 1

La aplicación WireShark se encuentra ya instalada en los equipos del laboratorio. Por seguridad, los analizadores de protocolos requieren permisos de administrador del sistema para poder realizar capturas del tráfico de la red. Por ello, para que el programa permita la selección de la interfaz sobre la que se van a realizar las capturas a cualquier usuario del sistema, es necesario ejecutar

chmod u+s /usr/bin/dumpcap

ya que dumpcap es la utilidad de la que depende Wireshark y, de esta forma, hacemos que el proceso dumpcap sea propiedad del *root.*

Tras seleccionar la Interfaz sobre la que deberá realizar capturas, arranque una consola y ejecute el comando ping. Este comando genera paquetes de tipo ICMP *Request* y ICMP *Reply.* Configure Wireshark para que capture 10 paquetes cualesquiera que circulen por la red, que no capture en modo promiscuo e inicie una captura.

Una vez se han capturado los 10 paquetes, suspenda la ejecución de ping (Ctrl-C) y observe los paquetes ICMP recibidos, identificando todos los elementos del protocolo.

Responda a las siguientes cuestiones:

- *Datos del paquete*: número, bytes de longitud, protocolos contenidos
- *Trama Ethernet*: Localice únicamente donde se identifica el protocolo de nivel IP que está encapsulado en la zona de datos de la trama Ethernet.
- *Trama IP*: Localice la longitud, origen, destino (¿en qué formato se especifica?) y tipo de protocolo que viaja en la parte de datos: cuál es su código correspondiente?
- *Trama ICMP*: ¿que es ICMP, qué código ("tipo") corresponde a un "Echo Request" y a un "Echo Reply".
- Observe el número de secuencia comparando dos paquetes uno de Request y otro de Reply.

Ejercicio 2

Elimine la limitación de capturas para detener el proceso manualmente y seleccione el modo promiscuo. Arranca una nueva captura. Vuelva a escribir el comando ping anterior. Arranque un navegador Internet y realice algunos accesos con él a cualquier dirección web.

Detenga la captura y observe los diferentes tipos de paquetes, contrastando con el ejercicio anterior. En el menú S*tatistics-Protocol Hierarchy* obtendrá una lista porcentual de los diferentes paquetes capturados. Indique cuántos paquetes de cada uno de estos tipos han sido recibidos: ARP_____, IP_____, TCP_____, UDP_____.

Defina un filtro de presentación escribiendo el nombre del protocolo que deseas ver en la barra de filtro*. Filtre* para ICMP y luego para TCP. Es posible ver mensajes dirigidos a otras direcciones de la LAN al captar en modo promiscuo y también mensajes TCP con errores y recuperación de errores (en negro)

Ejercicio 3

El protocolo DNS se utiliza para poder denominar a los computadores mediante nombres simbólicos en lugar de utilizar las direcciones IP más difíciles de recordar. Los servidores DNS se encargan de responder con la dirección IP buscada. El servicio DNS emplea el puerto 53 de UDP.

Realice una captura para estudiar los paquetes generados por la orden

$ ping -c 4 www.google.com

Examine el contenido del protocolo de la aplicación DNS para la petición de resolución de nombres. Localice el servidor DNS en la primera petición y el contenido de la consulta Query en el cuerpo del mensaje DNS. Eche un vistazo al contenido del datagrama UDP y observe los números de puerto implicados en la transmisión.

Observe que hay dos consultas: la primera obtiene la dirección IP dado un nombre de dominio, la segunda hace lo contrario; localice los parámetros y contraste con el caso anterior. Analice las conversaciones entre diferentes hosts que muestra la ventana *Menú Statistics – Conversations*. Se muestra para protocolos de diferentes niveles Ehernet, IPV (direcciones IP) y UDP. Otras opciones que refieren a conversaciones muestran el mismo tipo de información.

Podemos ver en Wireshark, las secuencia de intercambio de paquetes en un diagrama: *Menú Statistics – Flow Graph*.

Realice de nuevo la misma captura, pero proporcionando la dirección IP de Google al comando ping en lugar de su nombre de host y dominio. ¿Será necesario hacer consultas al servicio DNS?

Ejercicio 4

Mediante el comando telnet, o ssh nos podremos conectar a un usuario de otra máquina de nuestra red. este comando generará tráfico TCP para así poder analizarlo. Será útil designar un filtro como "*host dir_ip and not arp*" siendo *dir_ip* la IP del host en que nos situamos.

Analice las tramas, localice los puertos que conectan en el nivel de transporte y observe el gráfico de flujo como en el caso anterior.

Ejercicio 5

Al igual que con los comandos telnet, ftp o ssh, el protocolo HTTP también usa TCP en el nivel de transporte. Podemos hacer una petición de una página web desde el navegador a www.google.com o/y www.uah.es y capturar los paquetes generados para identificar algunos parámetros de este protocolo. Será conveniente usar el filtro predefinido

"*port 80*" que tiene el nombre "*TCP or UDP port 80 (HTTP)*". Porqué es *port 80*?

En el contenido de los datos http podrá localizar la orden GET / HTTP 1.0 e igualmente podrá localizar el texto de una página html que envía el servidor como respuesta a la petición Get. El seguimiento de las acciones también puede observarse desde *Menu Analyze-Follow TCP stream.*

TAREAS

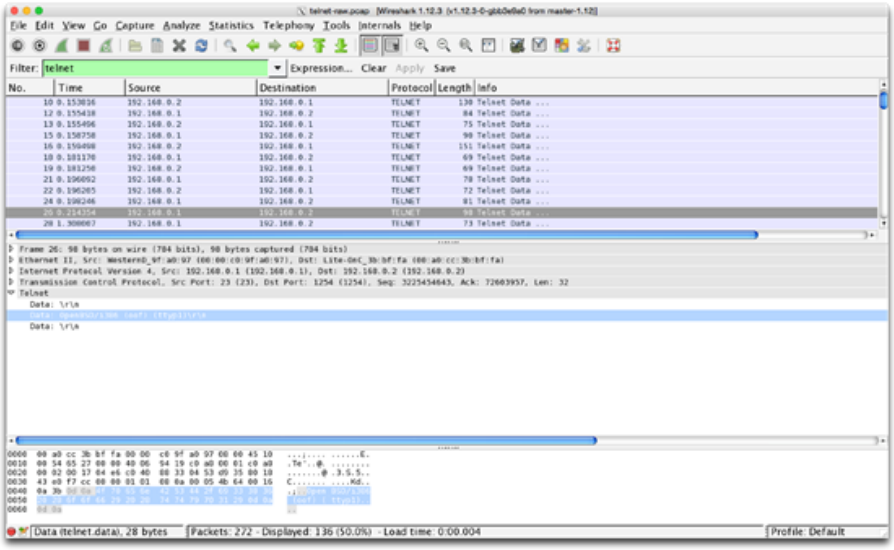
TAREA 1: Capturando tráfico con Wireshark

Primera parte: analizando un protocolo inseguro – Telnet.

En esta tarea no vamos a realizar capturas en vivo de tráfico, sino que vamos a analizar trazas (capturas) ya realizadas con anterioridad y salvadas en archivos. En este caso, vamos a usar la traza telnet-raw.pcap, del repositorio de capturas disponible en Wireshark.

Descárgate la traza en tu ordenador y ábrela con Wireshark. Esta traza ha capturado el tráfico de una sesión de Telnet entre el cliente y el servidor.

Un consejo: para observar mejor el tráfico de Telnet, puedes usar un filtro muy sencillo de visualización, como puedes ver en la imagen:



Para responder las siguientes preguntas, lo que se hizo fue utilizar la opción Follow TCP Stream dentro de Wireshark, la cual nos muestra la transmisión TCP en el orden que ocurrió desde el cliente hasta el servidor.

¿Qué usuario y contraseña se ha utilizado para acceder al servidor de Telnet?

Como podemos ver en esta captura de datos de Wireshark el usuario utilizado en la sesion de telnet es **fake** y la contraseña **user**

```
login: .."........"ffaakkee
.
Password:user
```

¿Qué sistema operativo corre en la máquina?

Sistema Operativo Open BSD

```
OpenBSD/i386 (oof) (ttyp1)
```

¿Qué comandos se ejecutan en esta sesión?

Como podemos ver en las siguientes imágenes, en la sesión de **Telnet** capturada se ejecutaron los siguientes comandos:

ls -a

```
$ llss
.
$ llss  --aa
```

ping a yahoo.com

```
.           ..          .cshrc     .login     .mailrc    .profile   .rhosts
$ //ssbbiinn//ppiinngg  wwwwww..yyaahhoooo..ccoomm
.
PING www.yahoo.com (204.71.200.74): 56 data bytes
64 bytes from 204.71.200.74: icmp_seq=0 ttl=239 time=73.569 ms
64 bytes from 204.71.200.74: icmp_seq=1 ttl=239 time=71.099 ms
64 bytes from 204.71.200.74: icmp_seq=2 ttl=239 time=68.728 ms
64 bytes from 204.71.200.74: icmp_seq=3 ttl=239 time=73.122 ms
64 bytes from 204.71.200.74: icmp_seq=4 ttl=239 time=71.276 ms
64 bytes from 204.71.200.74: icmp_seq=5 ttl=239 time=75.831 ms
64 bytes from 204.71.200.74: icmp_seq=6 ttl=239 time=70.101 ms
64 bytes from 204.71.200.74: icmp_seq=7 ttl=239 time=74.528 ms
64 bytes from 204.71.200.74: icmp_seq=9 ttl=239 time=74.514 ms
64 bytes from 204.71.200.74: icmp_seq=10 ttl=239 time=75.188 ms
64 bytes from 204.71.200.74: icmp_seq=11 ttl=239 time=72.925 ms
...^C
.--- www.yahoo.com ping statistics ---
13 packets transmitted, 11 packets received, 15% packet loss
round-trip min/avg/max = 68.728/72.807/75.831 ms
```

exit

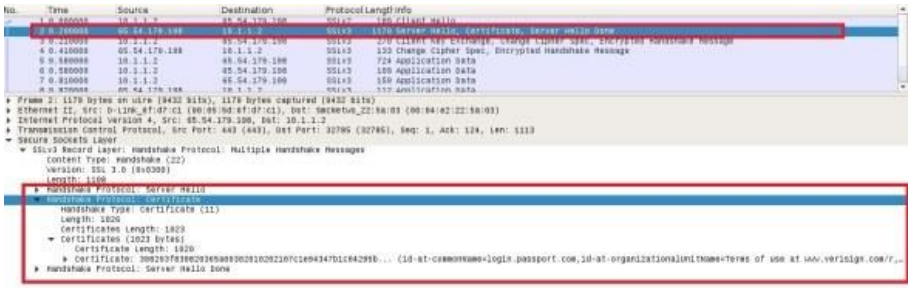
```
$ eexxiitt
```

Segunda parte: analizando SSL.

Para la realización de este ejercicio, descarga esta traza con tráfico SSL y ábrela con Wireshark. SSL es un protocolo seguro que utilizan otros protocolos de aplicación como HTTP. Usa certificados digitales X.509 para asegurar la conexión.

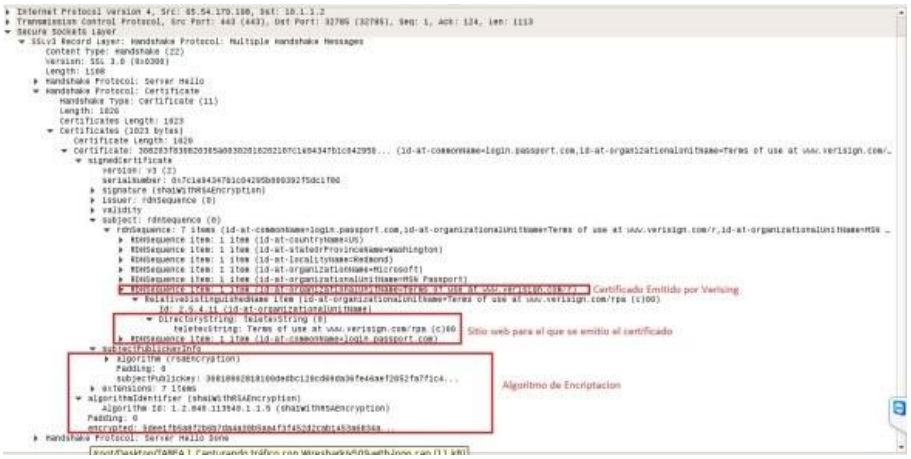
¿Puedes identificar en qué paquete de la trama el servidor envía el certificado?

Como podemos ver en la siguiente imagen el certificado se envía en el segundo paquete



¿El certificado va en claro o está cifrado? ¿Puedes ver, por ejemplo, qué autoridad ha emitido el certificado?

Los datos del certificado lo podemos ver en la siguiente imagen, la entidad que emitió el certificado es Verising



¿Qué asegura el certificado, la identidad del servidor o del cliente?

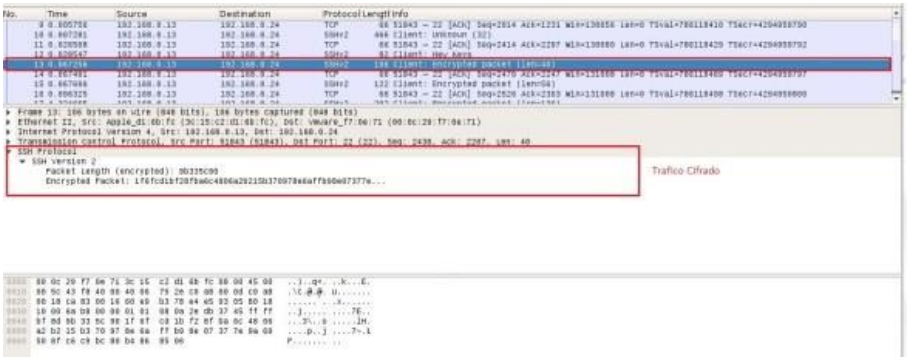
El certificado es asegurado por el servidor, como se pudo notar en la primera imagen.

Tercera parte: analizando SSH.

En la primera parte de este ejercicio hemos visto un protocolo no seguro, como Telnet. Una alternativa a usar Telnet a la hora de conectarnos a máquinas remotas es SSH, que realiza una negociación previa al intercambio de datos de usuario. A partir de esta negociación, el tráfico viaja cifrado. Descarga esta traza con tráfico SSH y ábrela con Wireshark.

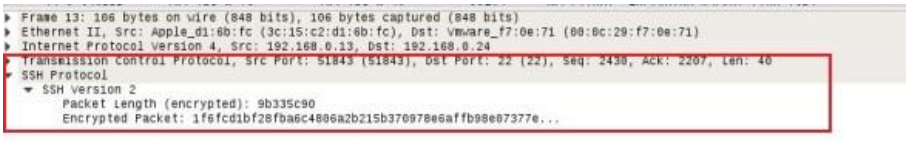
¿Puedes ver a partir de qué paquete comienza el tráfico cifrado?

El trafico cifrado comienza en el paquete 13, ya que es donde se envía el 1er paquete encriptado, como podemos ver en la siguiente imagen:



¿Qué protocolos viajan cifrados, todos (IP, TCP…) o alguno en particular?

El protocolo que viaja cifrado es el SSH, el cual va después de la capa TCP como podemos ver en la siguiente imagen:



¿Es posible ver alguna información de usuario como contraseñas de acceso?

En el protocolo SSH los datos viajen cifrado, lo que quiere decir que no es posible ver información tal como contraseñas o nombres de usuario.

https://wiki.wireshark.org/samplecaptures