****

**Redes de Computadoras**

**Práctico 6: Análisis de tráfico TCP y UDP en GNU/Linux.**

**Alumnos:**

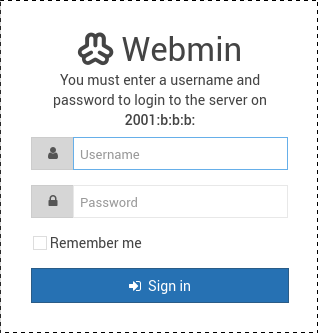
* **Heredia, Marco.**
* **Yepez Hinostroza, Franz.**

### Ejercicio 1: Análisis de tráfico TCP sobre servidor HTTP.

1.1.- Para crear e iniciar un container para ser utilizado como servidor DNS, utilizamos la imagen “sameesbn/bind:latest” que viene con “bind” como servicio DNS y “webmin” para la administración remota a través del navegador.

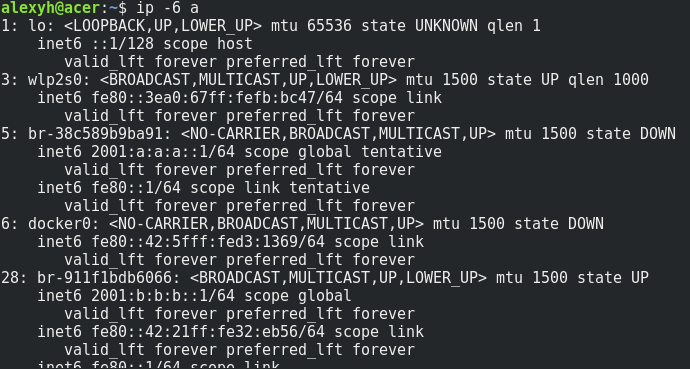
1.2.- Configuramos webmin para poder acceder a través de IPv6, para eso editamos el Dockerfile para configurar que una vez que levante el contenedor instale automáticamente el paquete libsocket6-perl. En nuestro caso no utilizamos las IP de red solicitada en la consigna 2001:a:a:a::/64 ya que nos daba un problema de configuracion, lo reemplazamos por la red 2001:b:b:b::/64

Por lo tanto para conectarnos a webmin por el navegador ingresamos en la siguiente dirección: [*https://[2001:b:b:b::2]:10000*](https://%5B2001:b:b:b::2%5D:10000)



2.1.- Realizamos el análisis del tráfico HTTPS con wireshark.

Primero, identificamos la interfaz del bridge que creó el docker que está en la misma red que el container.



Para este caso, vemos que la dirección ip 2001:b:b:b::1 está en la interfaz cuyo nombre es br-911f1bdb6066.

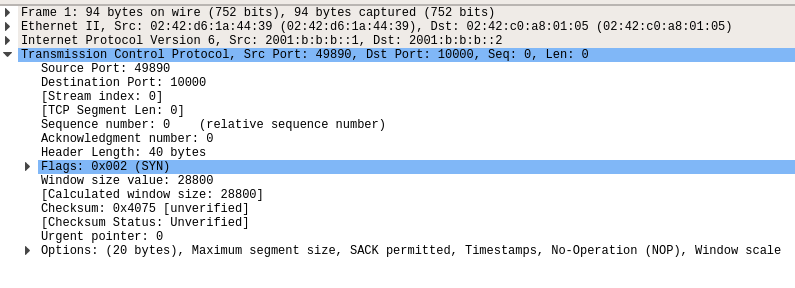
Procedemos capturar paquetes con el siguiente comando:

$ sudo tcpdump -i br-911f1bdb6066 -w <nombre\_registro>

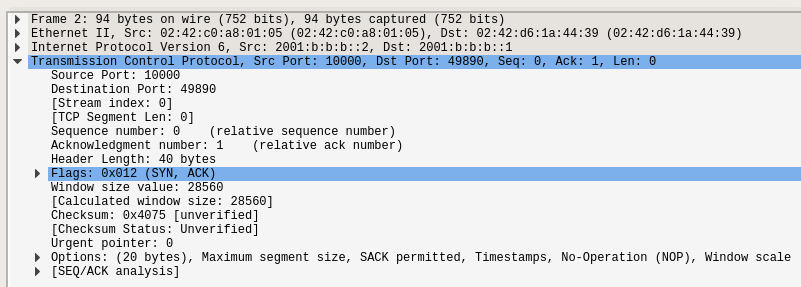
2.3.- La **secuencia de iniciación de sesión** tcp se denomina “Three-Way Handshake”. En términos generales el establecimiento de conexión es lo siguiente:



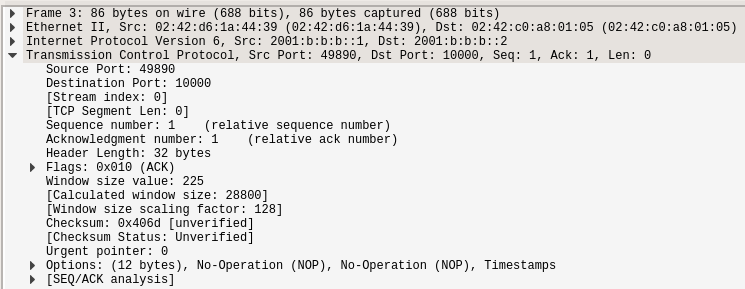
* El cliente envía un segmento con el flag o indicador SYN activado.



* El servidor responde un segmento con el flag o indicador SYN/ACK activado.



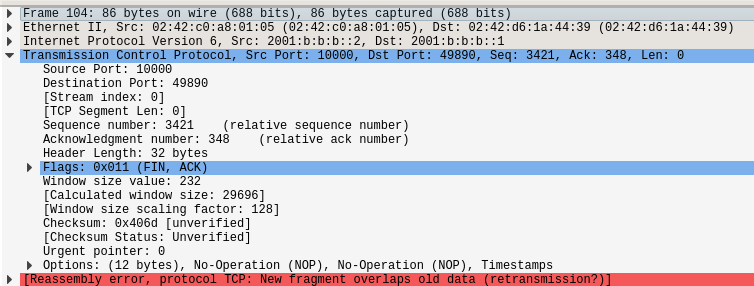
* El cliente envía un segmento con el flag o indicador ACK activado.



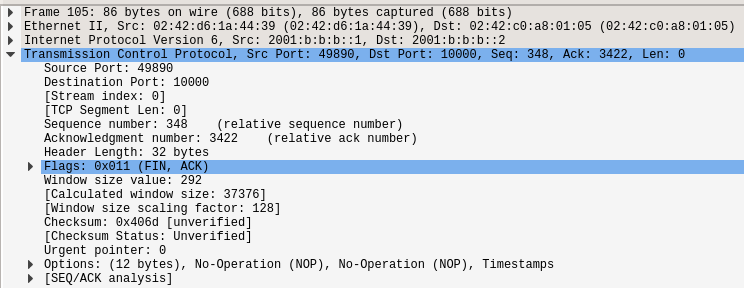
La **secuencia de finalización de sesión** tcp es una secuencia de 4 pasos, que de forma resumida es la siguiente:



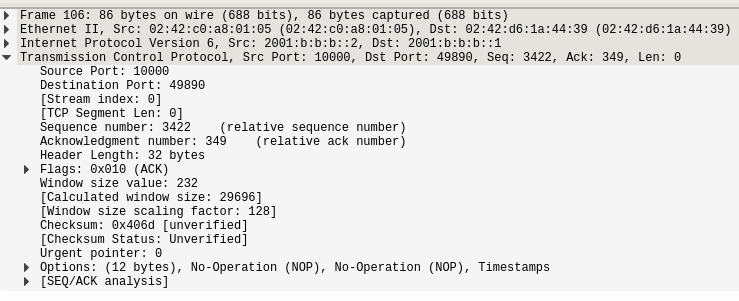
* El cliente envía un segmento solicitando el cierre de la conexión, este segmento no tiene datos, solo la cabecera con el flag FIN activado.



* El servidor confirma con un ACK la recepción del FIN. El cliente ya no puede enviar más datos nuevos, la conexión se cierra solo en un sentido.
* Ahora el servidor envía un segmento solicitando el cierre de la conexión, este segmento no tiene datos, solo la cabecera con el flag FIN activado.



* El cliente confirma este segundo FIN con un ACK. Por si ese último ACK se pierde, el cliente espera un tiempo por si lo tiene que reenviar y se finaliza la conexión TCP



La **ventana en la sesión tcp** se utiliza para el control de flujo. Existen 2 ventanas, una ventana de recepción (RW) y una ventana de congestión (CW). El número máximo de bytes que puede enviar el emisor (AW, Allowed Window) es el mínimo entre ambas ventanas.

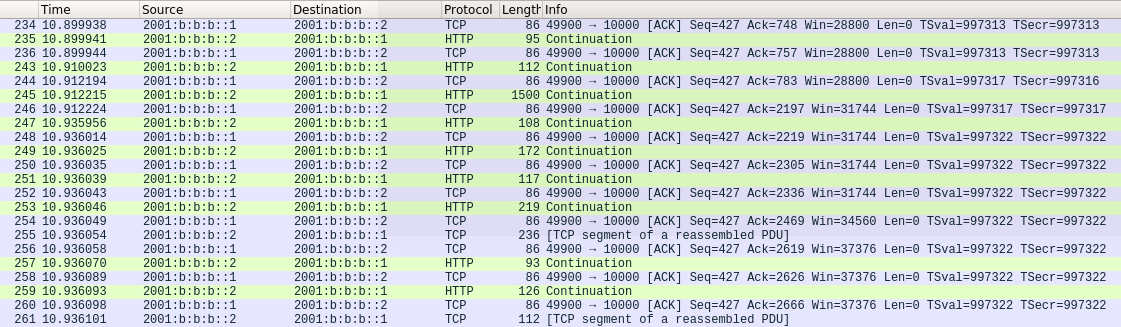
La ventana de congestión es complementaria a la ventana de recepción

usada para el control de flujo. En una situación de no congestión (sin pérdida o retraso de segmentos) la ventana de congestión alcanza el mismo tamaño que la ventana de recepción.

La transmisión comienza con un tamaño de ventana de congestión CW = 1 y va aumentando, pasando por tres fases distintas:

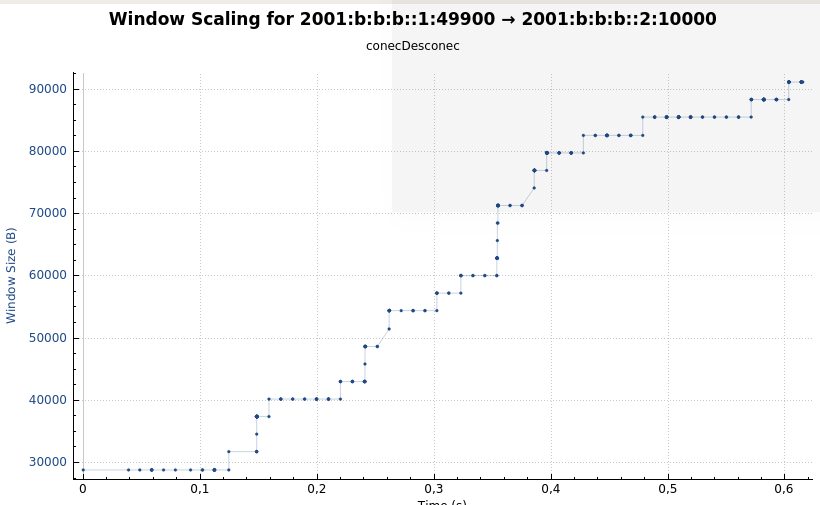
* Fase de arranque lento: la CW se incrementa en uno por cada segmento enviado y confirmado, esto provoca un crecimiento exponencial. Esta fase termina cuando el tamaño de CW alcanza un cierto umbral, denominado umbral de arranque lento
* Fase de evitación de congestión: a partir del umbral de arranque lento, la CW se incrementa en 1 cada vez que se envía y se confirma una ventana completa, esto provoca un crecimiento lineal. Esta fase termina cuando la CW alcanza el tamaño de la ventana de recepción (RW)
* Fase constante: en esta fase, la CW se mantiene a un valor constante (CW = RW)

En la siguiente captura podemos ver como se indica el tamaño de la ventana en el paquete y cómo va variando:



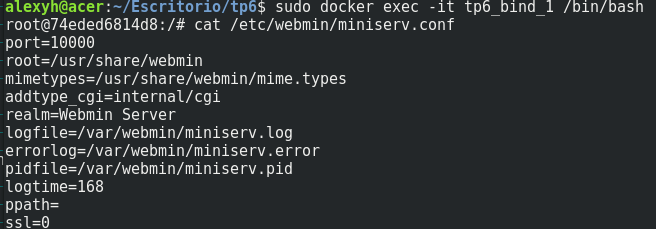
El **tamaño de ventana cero** nos indica que el receptro no puede recibir mas paquetes en ese momento, y la comunicación TCP se tiene que pausar hasta que pueda volver a procesar los paquetes que contiene en el buffer receptor.

Filtramos una sesión tcp en wireshark con el comando tcp.port == 49900 y podemos ver como varía el tamaño de la ventana durante la sesión. Podemos ver que al principio la ventana empieza en un valor de 28800 B y que va aumentando en un principio de forma exponencial, hasta llegar a un momento en el que aumenta de forma lineal. Al final de la sesión tcp la ventana llegó a un valor de 91136. Podemos observar el crecimiento de la ventana en el siguiente gráfico.



3.1

Para que webmin trabaje sin ssl fue necesario modificar el valor que venía por defecto. En la línea última línea mostrada el valor de fue cambiado de 1 a 0.



3.2

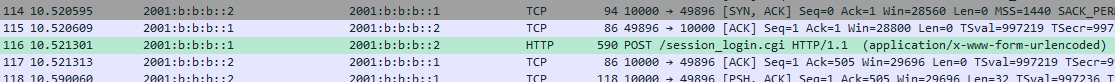
Para analizar el tráfico se utilizó la herramienta tcpdump para que escuchara en la interfaz bridge creada por docker para conectarse con el contenedor.

3.3

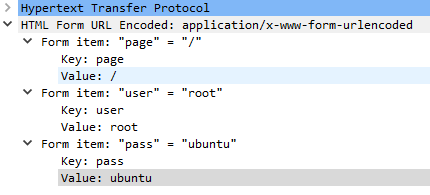
Se puede leer todo el contenido de los paquetes ya que éstos no se encuentran encriptados.

A continuación se muestra el paquete http que usa el método POST.

Para filtrar en wireshark:http.request.method=="POST"



A continuación su contenido:



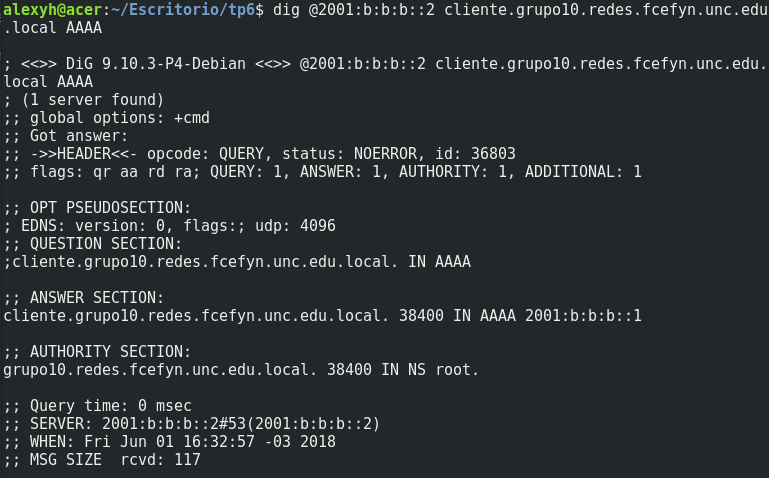
### Ejercicio 2: Análisis de tráfico UDP sobre servidor DNS.

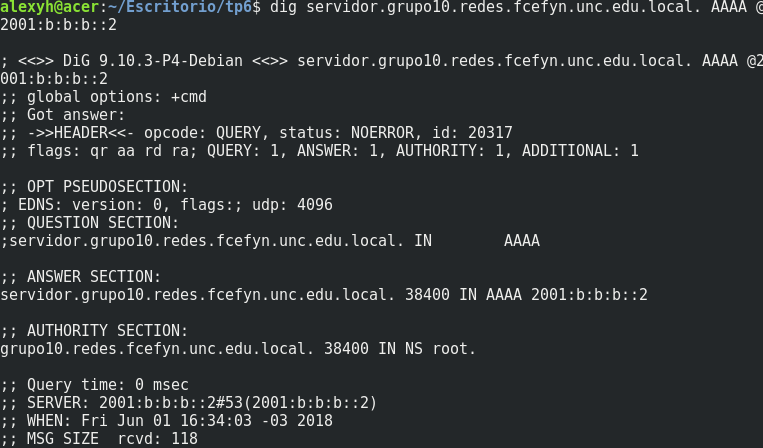
1.1.- Para que se guarden los archivos de configuración DNS de forma persistente agregamos las siguiente líneas al docker-compose.yml:

volumes:

- ./volumes/data:/data:rw

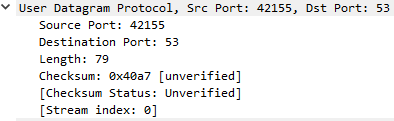
2.1





2.2

En la siguiente imagen se muestra la consulta dns realizada.



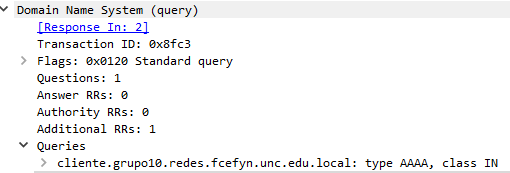
Como se puede observar, el encabezado de la capa de transporte es mucho más simplificado, ya que corresponde al protocolo udp.

Se puede apreciar el número de puerto de origen, en cual es un número aleatorio. El número de puerto de destino no lo es, ya que al ser una consulta dns, el valor por defecto establecido es 53.

A su vez, también se encuentra el campo Lenght(Longitud). Éste indica la longitud en bytes de la cabecera udp como de su contenido.

El campo checksum proporciona información relativa a la suma de comprobación, la cual es de uso opcional, en este caso como dice unverified no se verifica.

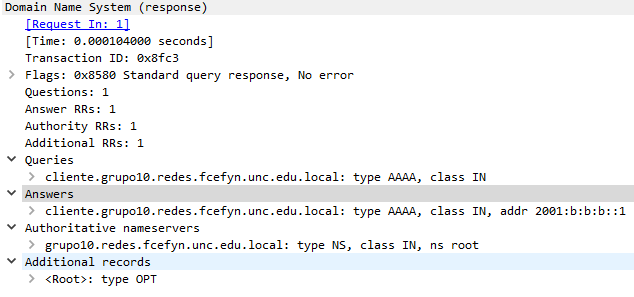
En la siguiente imagen se muestra la consulta DNS.



En la sección de Queries, se puede apreciar cuál fue el nombre de dominio del cual se solicita conocer su dirección IP.

Cabe aclarar que cuando se usó el comando dig, se especificó que fuera una consulta sobre la dirección ip versión 6,por lo que en esta sección aparece en type AAAA.

El contenido de la respuesta dns:



Esta información encapsulada en paquetes udp corresponden a la capa de aplicación.

Zona de autoridad: es el conjunto de dominios sobre el cual un servidor DNS es responsable.

Para nuestro caso, la zona de autoridad corresponde al siguiente:



Como se dijo antes, la consulta es transportada sobre un paquete udp, por lo tanto es una comunicación no orientada a la conexión, por lo que si la consulta no llegó a destino no vuelve a ser transmitida, por lo tanto, el reenvío de la consulta debe ser manejado por el programador de la aplicación.