**Practicando con Wireshark**



Daniel Escaño Hernández CET Ciberseguridad

# **1. Índice:**

[**1. Índice:**](#_lmwzrhyaxgtw) **1**

[**2. Ejercicio 1**](#_g59i8ewd3ed3) **2**

[**2.1 Apartado A:**](#_5d38uxjyswtl) **3**

[**2.2 Apartado B:**](#_jq8dj8r5cbgp) **3**

[**3. Ejercicio 2**](#_t6onuyod54or) **4**

[3.1 Apartado A:](#_k0fvdjnlsnzw) 5

[**3.2 Apartado B:**](#_yp9m375ktb2k) **6**

[**3.3 Apartado C:**](#_l1rlxn82xfk9) **6**

[3.4 Apartado D:](#_p5chs2qzd7pu) 7

[**4. Ejercicio 3**](#_c93cx0ohjm3r) **8**

[**4.1 Apartado A:**](#_ymmhubt7rkcc) **8**

[**4.2 Apartado B:**](#_jhmz0bo2y62k) **9**

[4.3 Apartado C:](#_askpl98ot4qo) 10

[**5. Ejercicio 4**](#_50vn0cuj6k6n) **11**

[**5.1 Apartado A:**](#_hfeys8fihfr7) **11**

[**5.2 Apartado B:**](#_y1pemnrgvywz) **12**

[**5.3 Apartado C:**](#_odizc09qbzsh) **14**

[**5.4 Apartado D:**](#_ap6hxzy44o7e) **14**

[**5.5 Apartado E:**](#_gmfxr1sravjm) **15**

[5.6 Apartado F:](#_iy1w44r2roms) 16

[**6. Ejercicio 6**](#_yx65r5b141uq) **17**

[**6.1 Apartado A:**](#_2mvpbimirk6d) **17**

[6.2 Apartado B:](#_gdl6lg5c2qk8) 18

[**6.3 Apartado C:**](#_q3l14mjxzqrx) **18**

[6.4 Apartado D:](#_b8cy5xgailmz) 20

[**6. Ejercicio 7**](#_valsymudoh7k) **23**

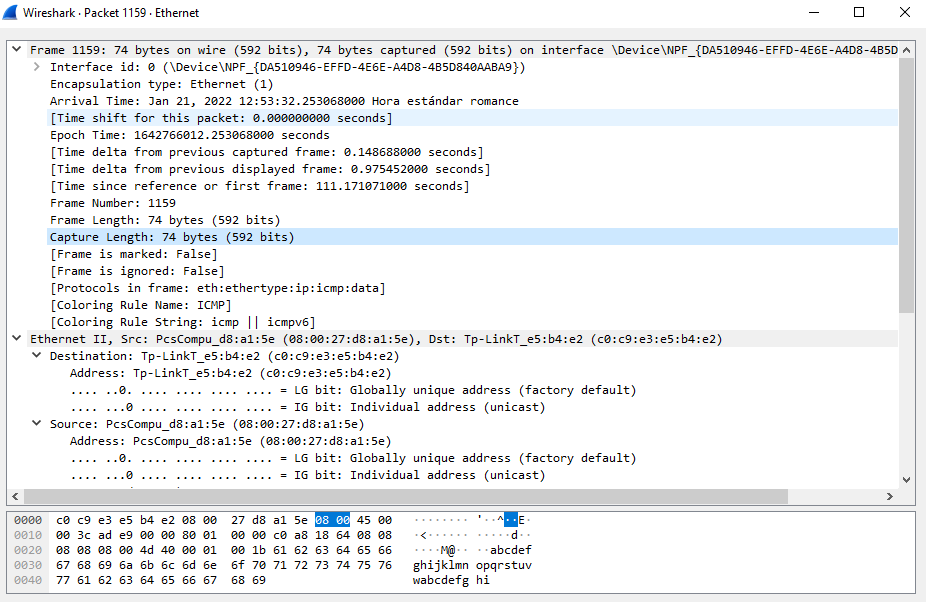
# **2. Ejercicio 1**

Abre el WireShark y selecciona la interfaz de red adecuada. Configura para que tome una

captura de 100 paquetes sin el modo promiscuo activo. Lanza un ping constante contra una

dirección IP pública. Filtra por los paquetes ICMP recibidos e identifica la siguiente

información para rellenar la tabla:



| Datos del paquete | Número | 1159 |
| --- | --- | --- |
| Bytes de longitud | 74 |
| Protocolos contenidos | ICMP |
| Trama IP | Longitud | 60 |
| Origen | 08:00:27:d8:a1:5e |
| Destino | c0:c9:e3:e5:b4:e2 |
| Tipo de protocolo | IPv4 |
| TTL | 128 |
| Trama ICMP | Código del Tipo | 8 |
| Tiempo de respuesta | 1164 |

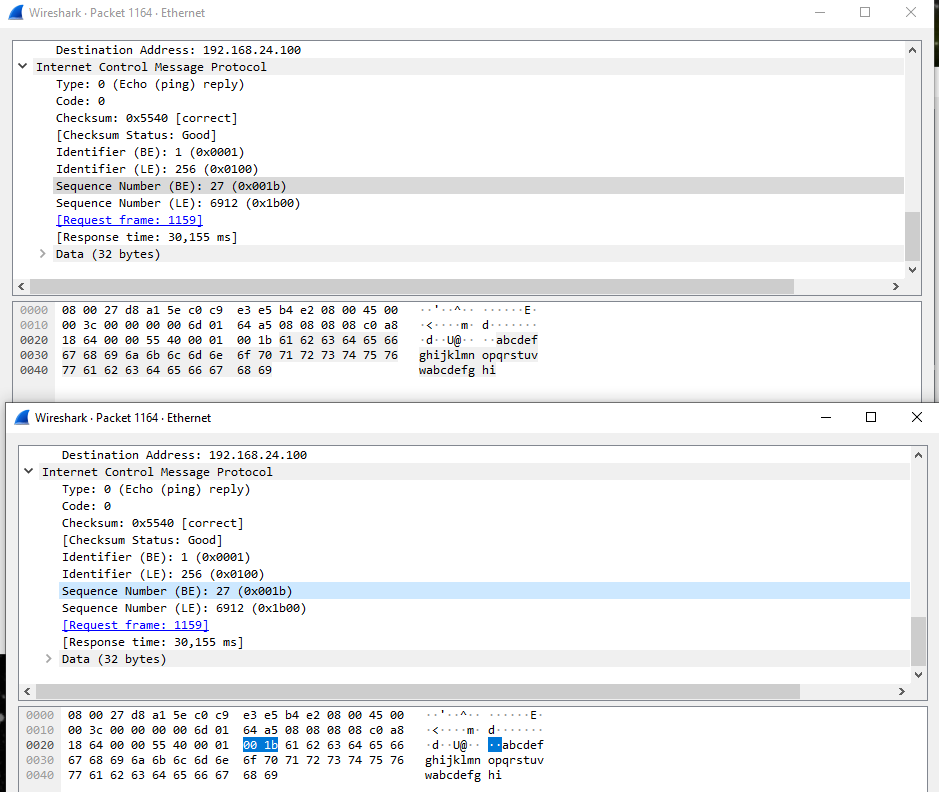
### **2.1 Apartado A:**

¿En el caso de un Echo Reply, qué código ICMP aparece?

Aparece el código 0

### **2.2 Apartado B:**

Comprueba que para un paquete de reply con un número de secuencia determinada, hay un paquete de request con el mismo número de secuencia.

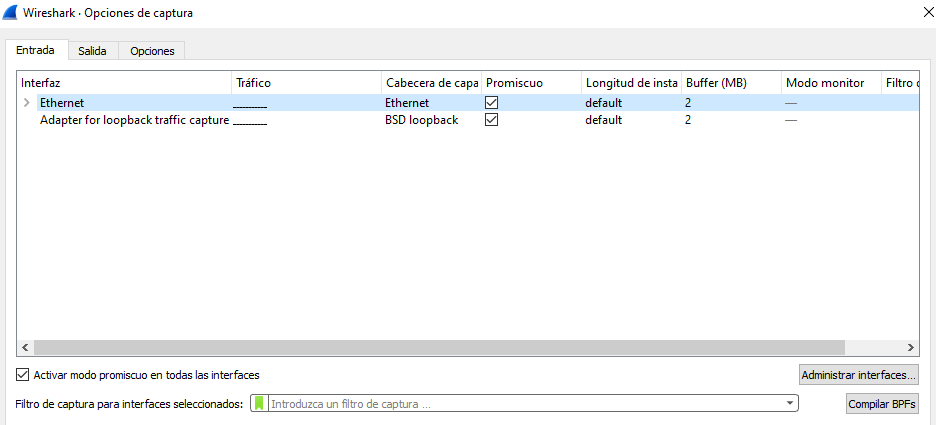


# 

# 

# **3. Ejercicio 2**

Realiza una nueva captura con el modo promiscuo activo y sin limitación en la recogida de paquetes. Cuando inicies la captura, realiza el mismo ping que hiciste en el ejercicio anterior y además abre el navegador con cualquier dirección web. Comprueba y realiza lo siguiente:



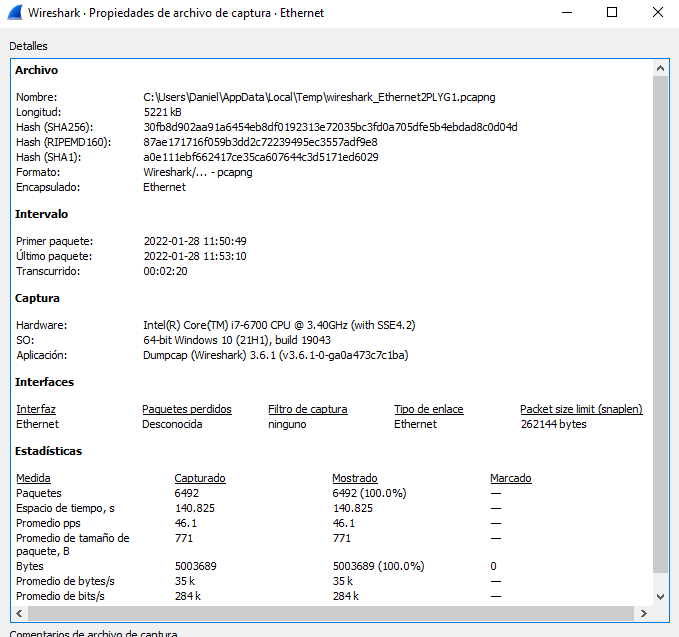
### 

### 

### 

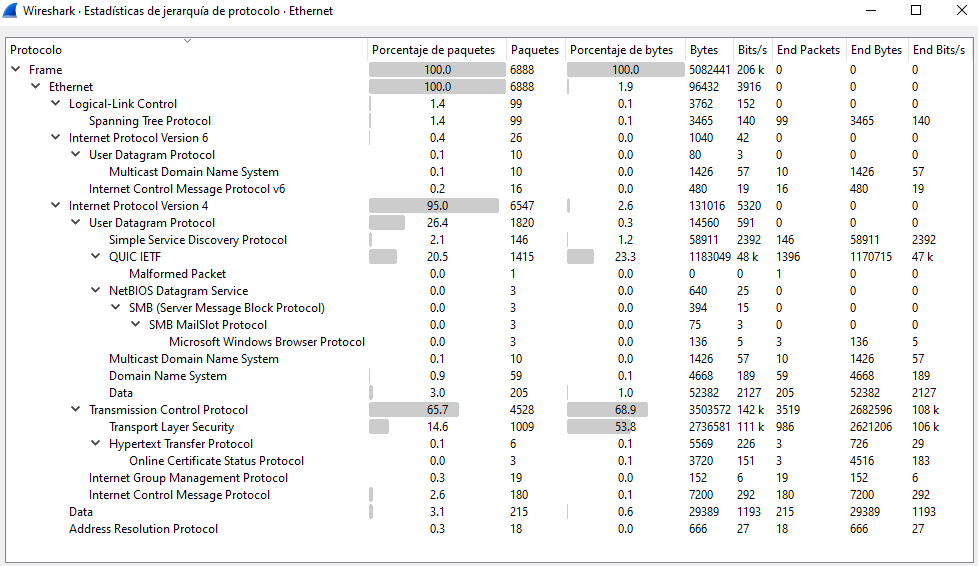
### **3.1 Apartado A:**

Estadísticas – Propiedades de la Captura tienes información sobre la captura generada



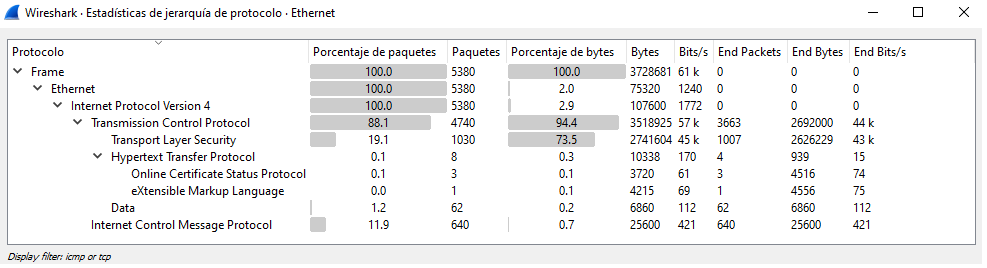
### **3.2 Apartado B:**

Estadísticas – Jerarquía de Protocolos tienes los % de los diferentes paquetes capturados



### **3.3 Apartado C:**

Haz un filtro que sólo muestre protocolo ICMP y TCP. Luego accede a la estadística de jerarquía de protocolos para comprobar que sólo te muestra datos sobre el filtro creado.



### 

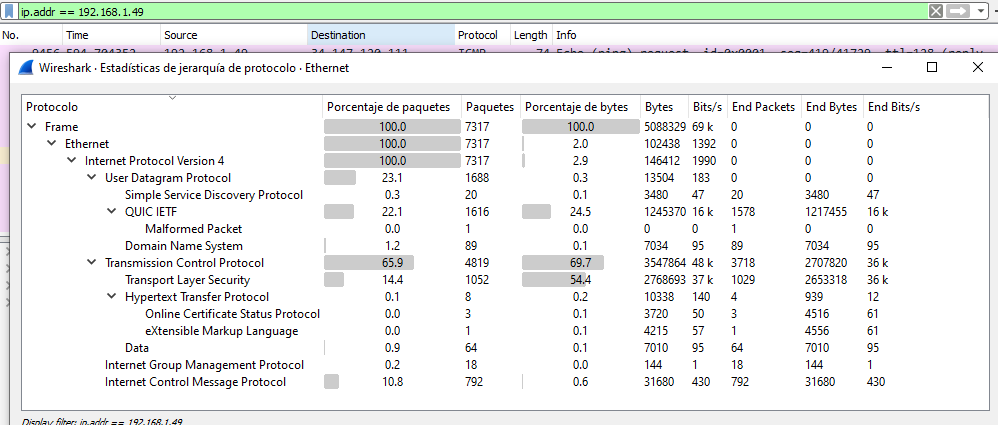
### 

### 

### 

### **3.4 Apartado D:**

Al haber activado el modo promiscuo, es posible que salgan direcciones IP de destino de la red local distintas a tu equipo. Filtra para ver cualquier IP perteneciente al direccionamiento local de tu red.

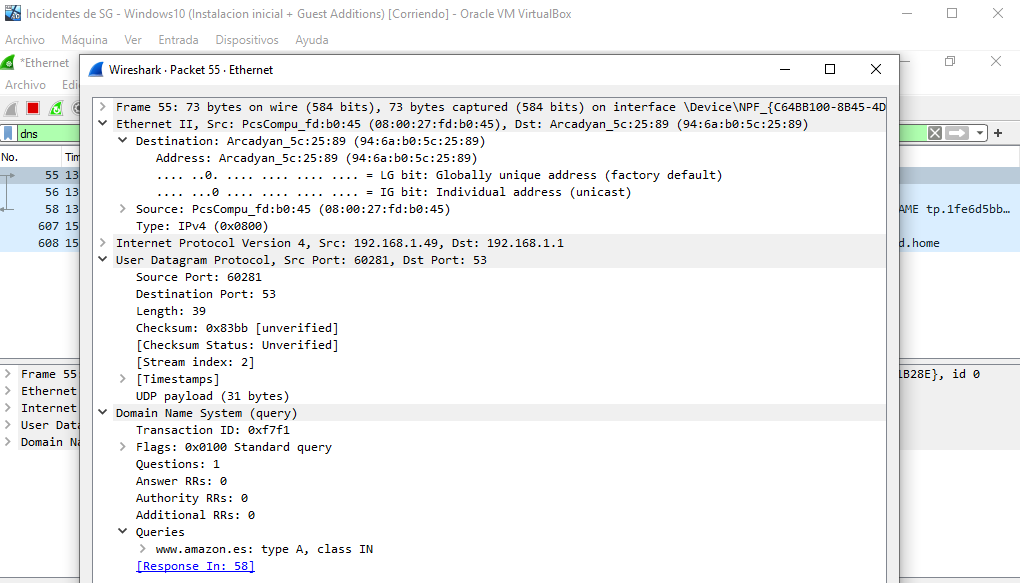


# **4. Ejercicio 3**

Realiza una nueva captura, enviando desde la línea de comandos 6 pings con resolución DNS al nombre www.amazon.es. Luego comprueba lo siguiente:

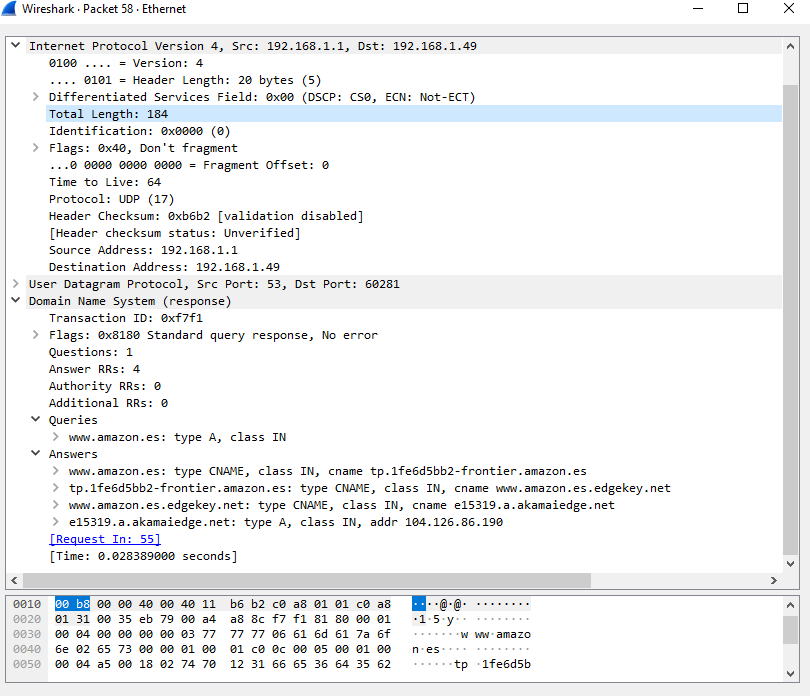
### **4.1 Apartado A:**

Localiza la primera petición DNS que se realiza y el servidor de destino. Dentro de la información del paquete, comprueba que aparece en queries el texto del nombre del dominio. Fijate que también te indica el número de paquete con la respuesta. ¿Qué número de puerto origen y destino aparece en la información UDP?



### **4.2 Apartado B:**

En el paquete de respuesta localiza la IP que se corresponde con el dominio solicitado



### 

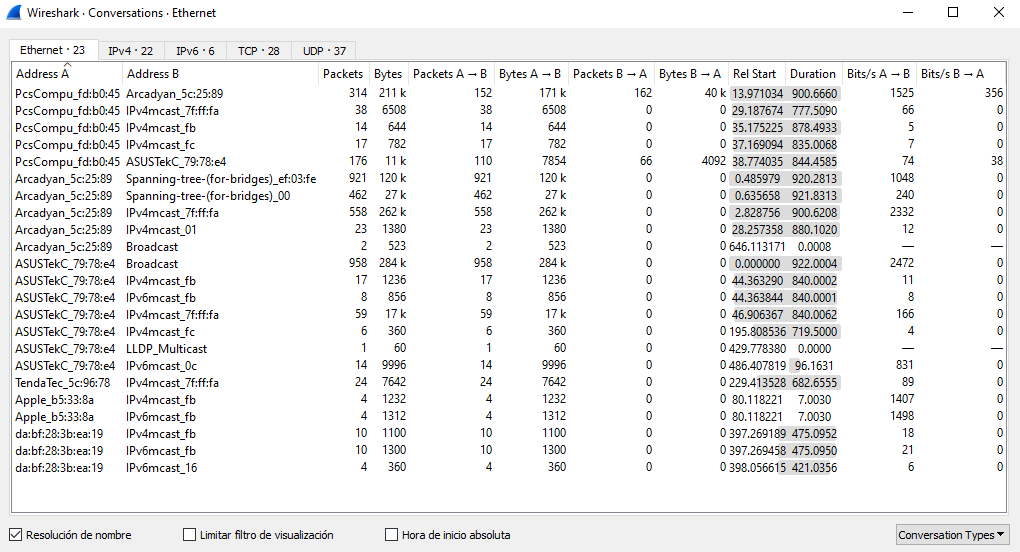
### 

### 

### 

### **4.3 Apartado C:**

Accede a Estadísticas – Conversaciones y podrás ver detalles de las comunicaciones a pares.



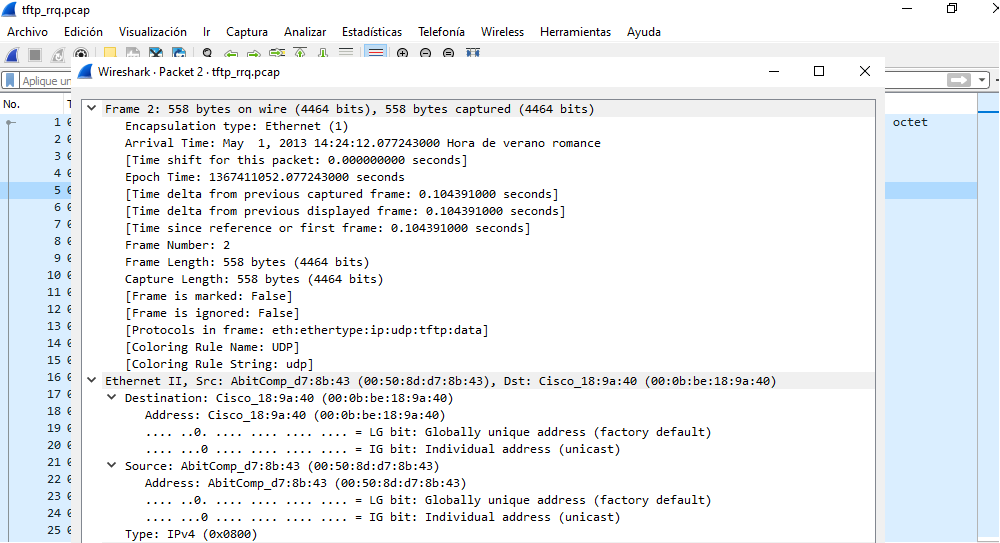
# **5. Ejercicio 4**

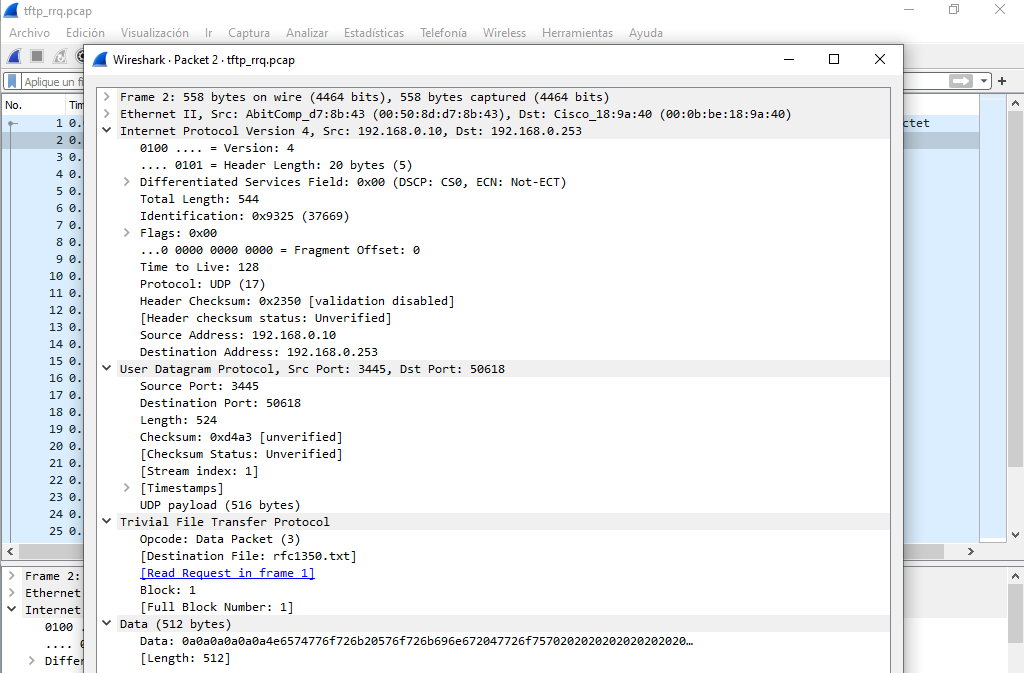
Analiza el fichero tftp\_rrq.pcap

### **5.1 Apartado A:**

¿De qué trata la comunicación? Analiza cualquier paquete de tipo Data Packet para ver que datos contiene

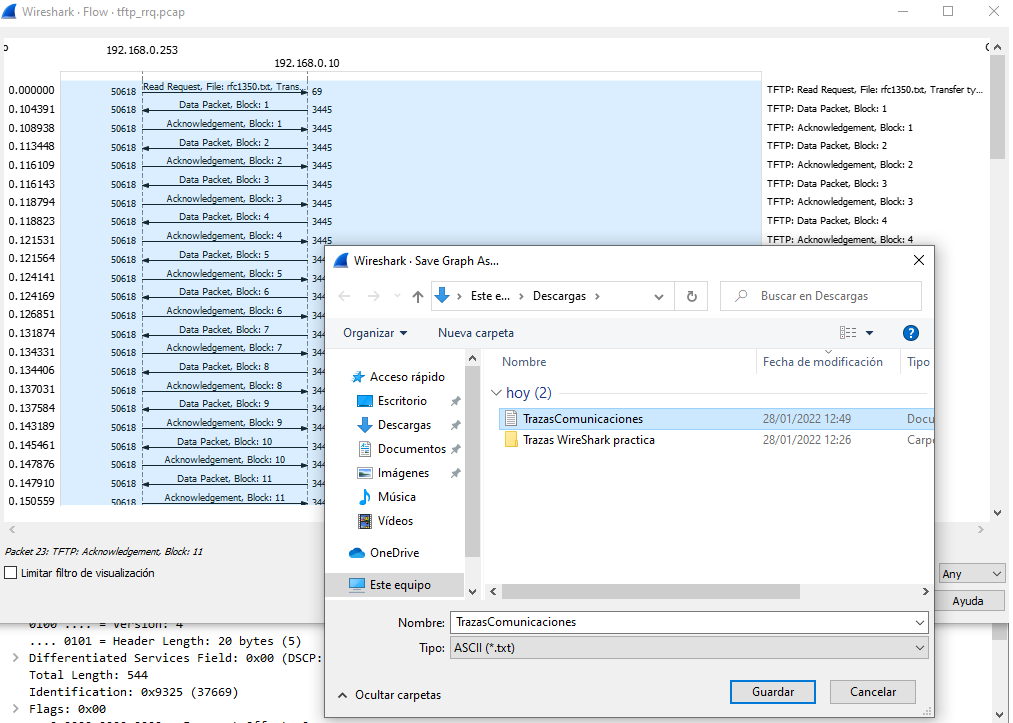
Se trata de una comunicación FTP.

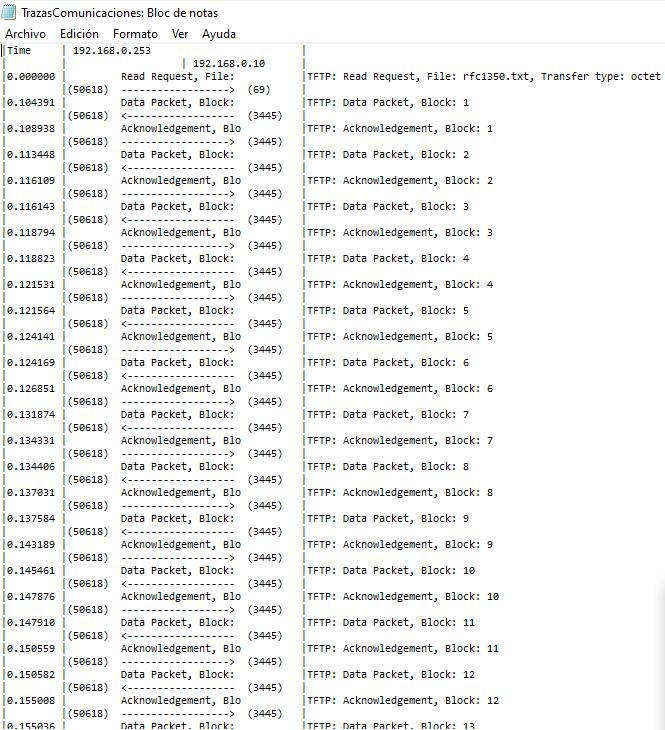




### **5.2 Apartado B:**

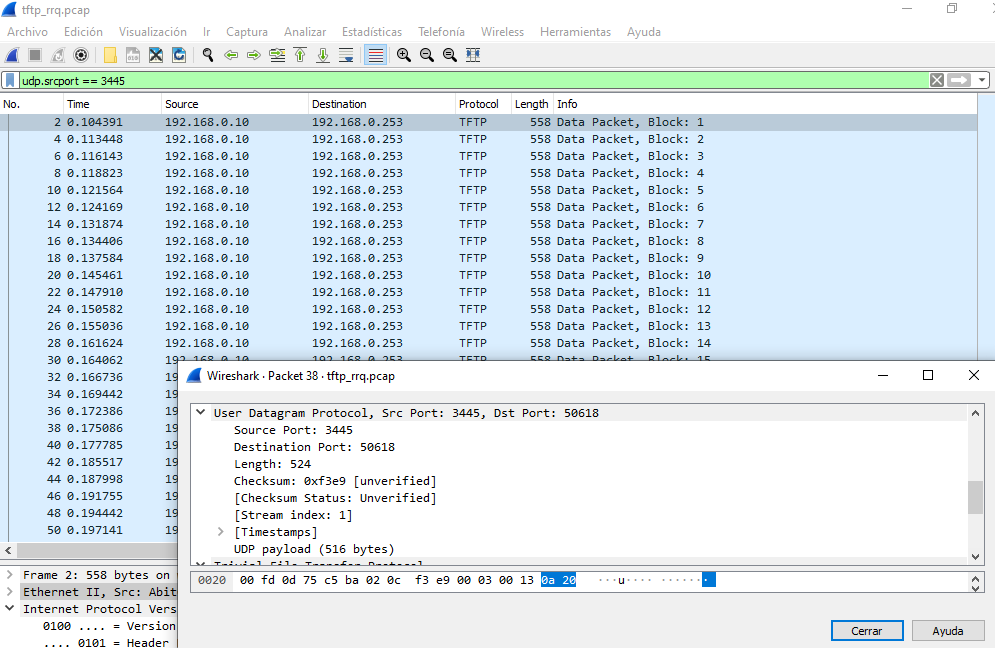
Obtén el fichero de texto que contiene las trazas de comunicaciones





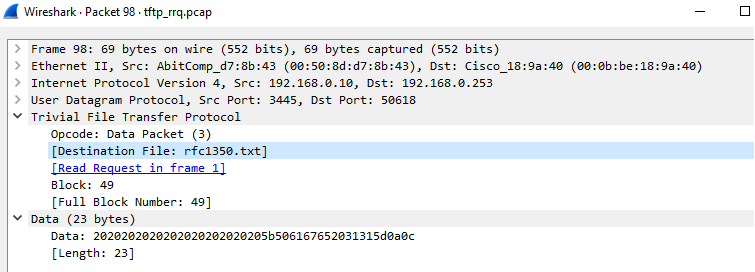
### **5.3 Apartado C:**

¿Qué puerto de comunicaciones aparece en el flujo de la descarga? Establece un filtro de visualización con el puerto fijo que has encontrado.



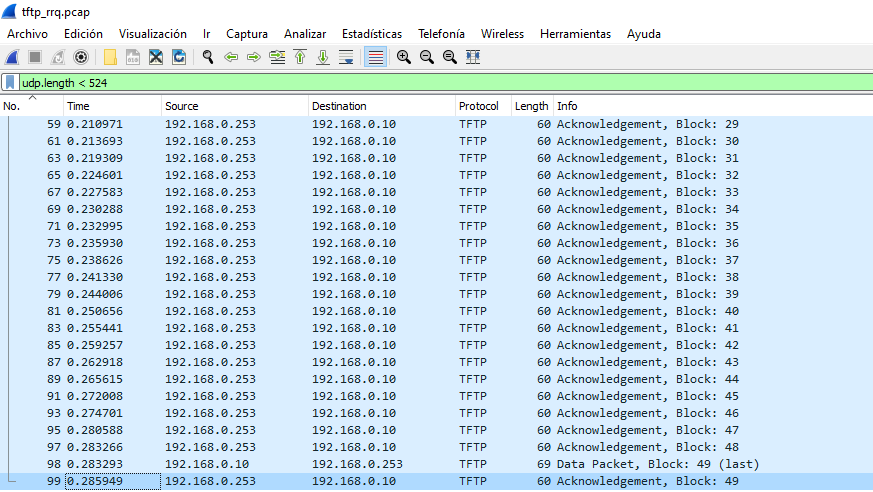
### **5.4 Apartado D:**

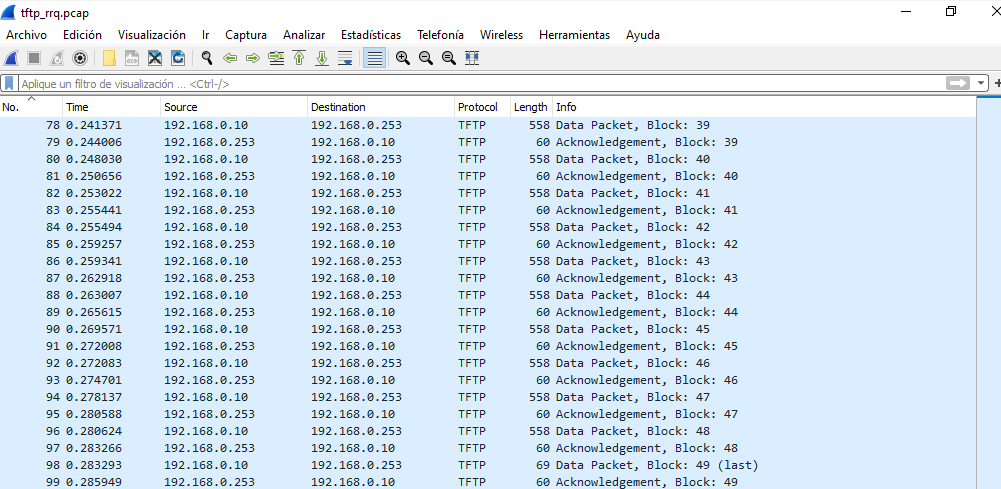
¿Cuál es el nombre del fichero que se estaba descargando?



### **5.5 Apartado E:**

¿Qué filtro utilizarías para visualizar solo los paquetes UDP que tengan una longitud menor a 524 bytes? ¿Cuántos paquetes hay en la captura con esa longitud? Niega el filtro que has establecido y comprueba que el número de paquetes totales es correcto.





### 

### 

### 

### 

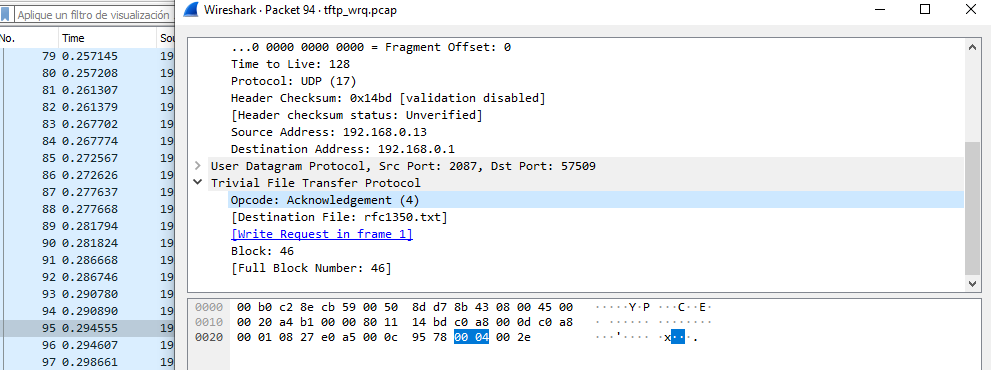
### 

### **5.6 Apartado F:**

Abre el fichero de captura tftp\_wrq.pcap Respecto a la otra traza, ¿Se estaba realizando el

mismo tipo de operación? ¿Con qué código se distingue cada una de las operaciones?

Se utiliza el mismo protocolo y se concluye la descarga del archivo con el mismo nombre, pero el OPcode es diferente, en este siendo 4 y en la anterior traza contiene el valor 3.



# 

# 

# 

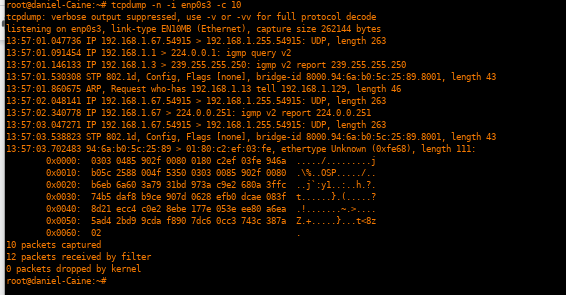
# **6. Ejercicio 6**

WireShark es una utilidad gráfica con gran potencia como habrás podido ver. En muchos

servidores Linux que no tenemos entorno gráfico, existe el sniffer TCPDump que nos permitirá capturar tráfico y además guardarlo (pcap) para luego ser examinado con WireShark

### **6.1 Apartado A:**

Utiliza una máquina Linux para lanzar TCPDump y ver las trazas de comunicaciones por consola. Usa los modificadores -n (no resolver IP a nombres) -i (selecciona la interfaz) y -c (captura 10 paquetes).

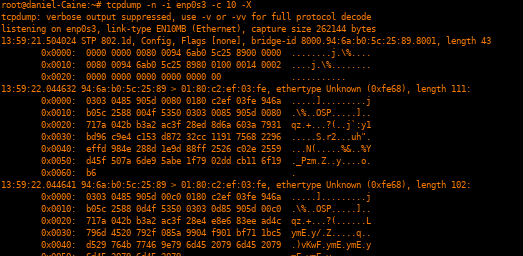


### 

### 

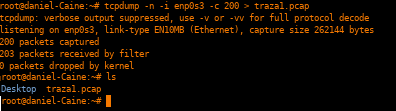
### **6.2 Apartado B:**

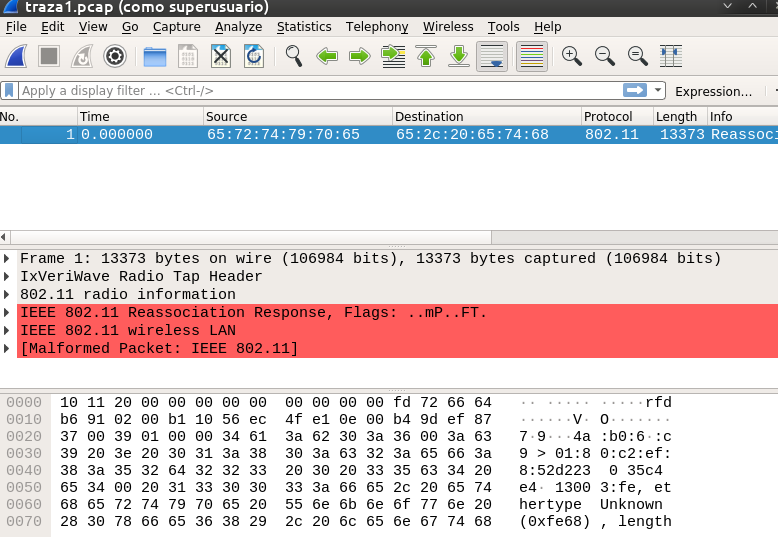
¿Qué modificador utilizarías para ver el contenido de la traza en hexadecimal? ¿Y para que salga en formato ASCII al lado de los valores en hexadecimal? Compruébalo.



### **6.3 Apartado C:**

Utiliza el modificador adecuado para guardar una captura en formato .pcap que contenga 200 paquetes. Abre la captura por pantalla utilizando TCPDump. Luego ábrela con WireShark para comprobar que se ha guardado en el formato correcto. Para verla por pantalla:





### 

### 

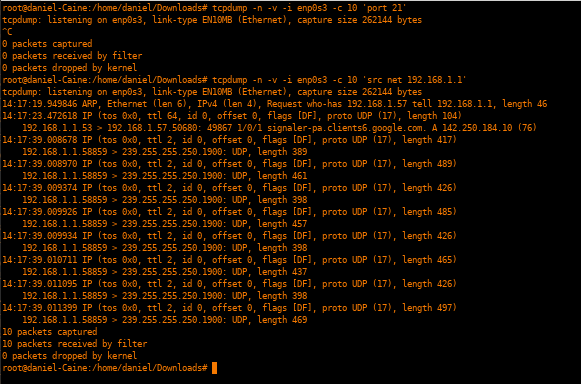
### 

### **6.4 Apartado D:**

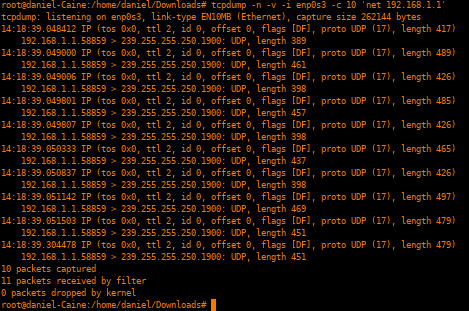
Probaremos algunos filtros en TCPDump. La sintaxis que sigue es tcpdump [opciones] [patrones]. La documentación sobre los patrones que podemos utilizar.

Capturar todos los paquetes dirigidos al puerto 21 mostrando toda la información adicional de los paquetes

Capturar todos los paquetes cuyo host de origen sea la puerta de enlace de tu red.

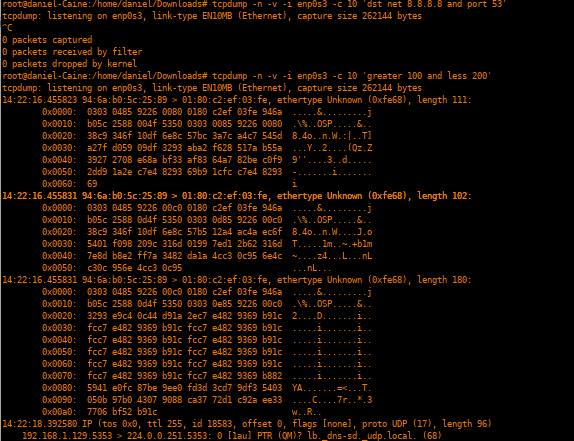


Capturar todos los paquetes para los cuales la puerta de enlace de la red sea origen o destino.



Aquellos paquetes cuyo destino sea 8.8.8.8 y el puerto utilizado sea 53

Queremos capturar todos los paquetes cuyo tamaño se encuentre en el rango 100 a 200 bytes



# 

# **6. Ejercicio 7**

Conociendo ahora WireShark y TCPDump, te habrá quedado claro que es mucho más fácil analizar los paquetes usando WireShark. En ocasiones puede ser muy útil, redirigir la salida del TCPDump al WireShark en tiempo real. Por ejemplo, si tenemos un servidor Ubuntu sin modo gráfico y estamos analizando el tráfico de red. En el siguiente artículo se explica como se puede llevar acabo. Probarlo en una máquina con Ubuntu Server redireccionando la traza a otra máquina con WireShark.

