Sesión de Laboratorio L2

Herramientas básicas y HTTP

Lectura previa:

- Kurose Cap. 2.2

Información de consulta:

- HTTP: http://www.rfc-editor.org/rfc/rfc2616.txt
- Guía usuario Wireshark: http://www.wireshark.org/docs/

El entorno de trabajo

- Acceso al PC: Linux. Arrancar en la partición “Prácticas de Red-GII”.
- Servidores: zoltar.redes.upv.es, www.upv.es

Uso de la herramienta netcat (ver Apéndice A)

Esta herramiento nos permite enviar y recibir información utilizando sockets TCP y UDP mediante sencillas órdenes desde el *shell*. Por lo tanto necesitarás utilizarla desde un terminal. El apéndice A muestra información detallada de los parámetros que acepta.

Demostraciones por parte del profesor:

- Conexión como cliente a `zoltar.redes.upv.es`:
  - Servicio de eco (puerto 7):
    - Conexión a zoltar mediante la orden `> nc zoltar.redes.upv.es 7`

      Después se puede teclear lo que uno quiera, tras darle al retorno de carro se enviarán los datos y el servidor devuelve la misma frase que hayamos enviado.
  - Nueva prueba con el servicio de daytime (puerto 13):
    - `> nc zoltar.redes.upv.es 13`
- Uso cliente-servidor: “chat” entre PCs:

  servidor: nc -l 9999

  cliente: nc localhost 9999

- **Ejercicio 1:** envía mediante netcat una petición HTTP de tipo GET al servidor web de la UPV (www.upv.es) para obtener su página principal. Prueba con HTTP/1.0 y HTTP/1.1. Por ejemplo, para el primer caso (1.0) teclea:

  ```
  > nc www.upv.es 80
  ```

Con esta orden estás conectandote al servidor www.upv.es (que es el servidor de web de la UPV) al puerto 80, que es donde se encuentra este servicio habitualmente (*well-known port*) mediante el protocolo TCP, que es que por defecto utiliza la herramienta nc. Una vez que has conectado debes solicitar el recurso:

```
GET / HTTP/1.0
```

Atención: Hay que pulsar la tecla de retorno de carro **dos veces** después de la segunda línea.
Como resultado de esta acción se recibirá un texto parecido al siguiente:

```
HTTP/1.1 200 OK
Date: Thu, 06 Oct 2016 17:13:16 GMT
Server: Apache
Accept-Ranges: bytes
X-UA-Compatible: IE=EmulateIE7, IE=9
Connection: close
Content-Type: text/html; charset=ISO-8859-1
Content-Language: es
      ...
```

El contenido recibido se ha mostrado por la salida estándar, que era en este caso la pantalla. Si se desea capturar este texto se puede redirigir la salida a un archivo mediante el operador “>”.

Como el servidor habrá cerrado la conexión tras enviarte el recurso. Vuelve a establecer una nueva conexión con nc y solicita ahora de nuevo la página, esta vez utilizando la versión 1.1 del protocolo HTTP.

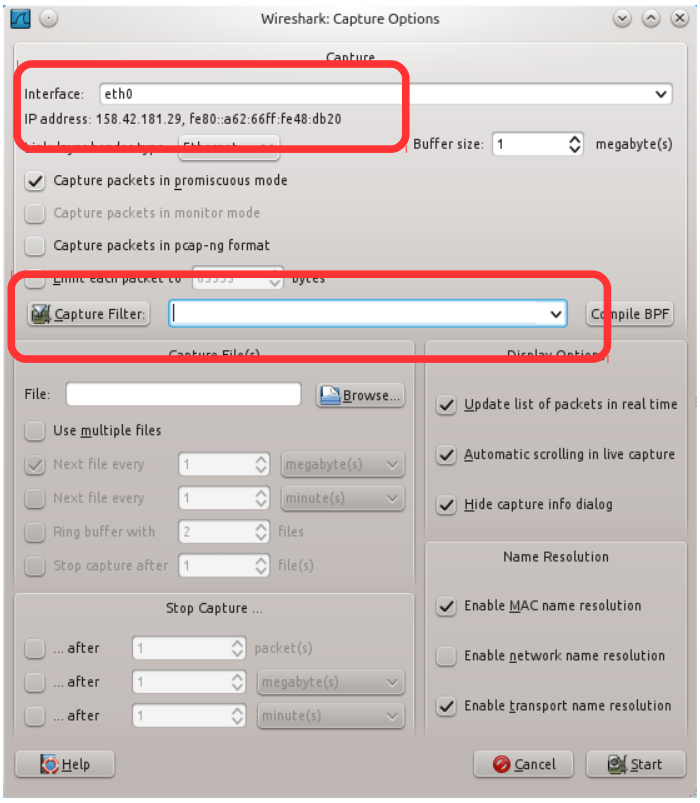
- ¿Cómo se indica el fin de las cabeceras en la petición del cliente y en la respuesta del servidor?
- ¿Qué cabeceras son obligatorias en la petición del cliente?

Recapitulando, lo que estamos haciendo es crear un socket en nuestro ordenador. Ese socket, que actúa como cliente, lo conectamos al servidor de web de la UPV y le solicitamos que nos envíe el contenido de su página web inicial. La conexión iniciada por la orden **`nc`** se realiza al puerto 80 del servidor www.upv.es y dura sólo lo indispensable hasta que se entrega la página web solicitada. Es importante destacar que la respuesta del servidor contiene una información del protocolo HTTP (o cabecera) a la que sigue, después de una línea en blanco, el código HTML de la página solicitada. Tras enviar esa información el servidor cierra la conexión, con lo cual la ejecución de la orden **`nc`** finaliza.

Usos de los analizadores de protocolos

- En esta parte de la sesión pasaremos a trabajar con el analizador de protocolos Wireshark que nos permitirá analizar el tráfico HTTP intercambiado entre nuestro navegador y los servidores web a los que nos conectemos.
- Recuerda que antes generar el tráfico (solicitar las páginas web indicadas) debes iniciar las capturas con el Wireshark. Para ellos recuerda un par de pasos básicos:
  - Es conveniente que compruebes que el interfaz sobre el que vas a capturar es el adecuado. Para ello accede al menú *Capture* y selecciona la opción *Options*. Te aparecerá una nueva ventana similar a la mostrada en la figura siguiente, llamada *Capture Options*. Comprueba que la interfaz tiene una dirección IP asociada que

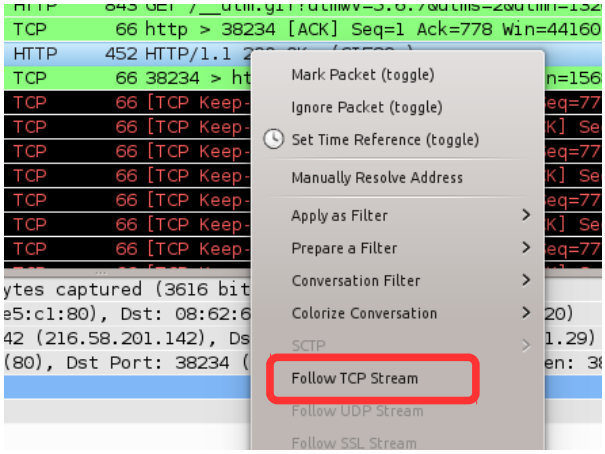
empiece por 158.42 y si no fuera así selecciona la interfaz que tenga ese tipo de dirección asociada. Se trata de la dirección IP pública que nos permite acceder a Internet.



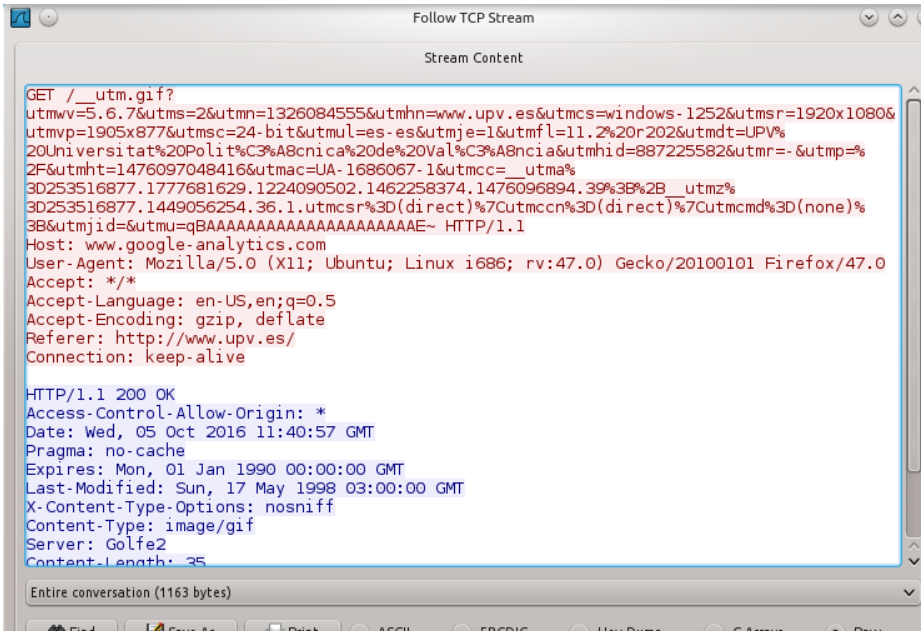
Como vamos a capturar tráfico HTTP en el apartado “Capture Filter” introduce “port 80”.

- HTTP con Wireshark
  - Utilidad de la orden *Follow TCP Stream*

    Al trabajar con protocolos de aplicación basados en TCP puede resultar muy útil poder ver los datos intercambiados en una conexión TCP de la misma forma que los ve el nivel de aplicación. Especialmente cuando en la captura aparecen intercalados los paquetes asociados a diferentes conexiones TCP. Para ello el programa wireshark proporciona la opción “*Follow TCP Stream*”. Al seleccionar un paquete de una conexión determinada, en la que estemos interesados, y seleccionar la opción “*Follow TCP Stream*”, del menú “*Analyze*”, wireshark aplica automáticamente un filtro de visualización y abre una ventana adicional mostrando en orden todos los datos intercambiados a nivel de aplicación por el cliente y el servidor. También puedes activarlo mediante el menú que se abre al pulsar el botón derecho del ratón, después de haber seleccionado el paquete correspondiente:



- Además, para facilitar su interpretación utiliza diferente color para los datos de cada uno de los extremos de la comunicación:



- **Ejercicio 2**. Configura el wireshark para capturar el tráfico del puerto 80 mediante un filtro de captura (sintaxis “port 80”). Inicia la captura y utilizando tu navegador accede a esta página:

  http://www.redes.upv.es/redes/Wireshark/HTTP-wireshark-file1.html

  Analiza el tráfico capturado y contesta a las siguientes preguntas:

  - Tu navegador, ¿utiliza HTTP 1.0 o 1.1? ¿Qué versión de HTTP está utilizando el servidor?
  - ¿Qué idioma está indicando tu navegador al servidor que puede aceptar?
  - ¿Cuál es la dirección IP de tu ordenador? ¿Y la del servidor?
  - ¿Cuándo ha sido modificado por ultima vez el fichero html en el servidor?¿Qué diferencia hay entre las cabeceras *Date* y *Last-modified*?
  - ¿Cuál es el tamaño de la página que envía el servidor?

- **Ejercicio 3**. Repite la captura anterior recargando la página y observa las cabeceras de la petición GET enviada y la respuesta del servidor.

  - ¿Qué diferencias observas con las del ejercicio anterior?

- **Ejercicio 4**. Ahora analiza el acceso a esta nueva página:

http://www.redes.upv.es/redes/Wireshark/HTTP-wireshark-file3.html

Cuando haya terminado la transferencia de datos deja la captura abierta hasta que aparezcan los segmentos marcados con la etiqueta FIN (cierre de la conexión TCP). Contesta las siguientes preguntas:

- ¿Cuántos HTTP GET ha enviado tu navegador?
- ¿Cuántos segmentos TCP han sido necesarios para llevar la respuesta HTTP?

Ejercicios avanzados:

(cierra el navegador y borra su historial antes de proceder)

- **Ejercicio 5**. Analiza el acceso a esta página:
  http://www.redes.upv.es/redes/Wireshark/HTTP-wireshark-file5.html
  y contesta las siguientes preguntas:
  - ¿Cuántos HTTP GET ha enviado tu navegador? ¿A qué direcciones de Internet?
  - ¿Puedes decir si tu navegador ha descargado las dos imágenes de forma secuencial o en paralelo desde los dos servidores web?
  - ¿Se trata de una conexión persistente?
  - ¿Quién cierra la conexión, el cliente o el servidor?

- **Ejercicio 6**. En Firefox teclea **about:config** en la barra de direcciones y busca el atributo **network.http.version**. Cambia el valor a 1.0. Repite el ejercicio anterior e indica las diferencias que observas.

- **Ejercicio 7**. Accede ahora al servidor `www.upv.es`. Analiza las diferencias de comportamiento en cuanto a la persistencia de las conexiones.

Apéndice A

La herramienta netcat: usos y parámetros

usage: nc [-46DdhklnrStUuvzC] [-i interval] [-P proxy_username] [-p source_port]
[-s source_ip_address] [-T ToS] [-w timeout] [-X proxy_protocol]
[-x proxy_address[:port]] [hostname] [port[s]]

Command Summary:

-4	Use IPv4
-6	Use IPv6
-D	Enable the debug socket option
-d	Detach from stdin
-h	This help text
-i secs	Delay interval for lines sent, ports scanned
-k	Keep inbound sockets open for multiple connects
-l	Listen mode, for inbound connects
-n	Suppress name/port resolutions
-P proxyuser	Username for proxy authentication
-p port	Specify local port for remote connects
-q secs	quit after EOF on stdin and delay of secs (-1 to not quit)
-r	Randomize remote ports
-S	Enable the TCP MD5 signature option
-s addr	Local source address
-T ToS	Set IP Type of Service
-C	Send CRLF as line-ending
-t	Answer TELNET negotiation
-U	Use UNIX domain socket
-u	UDP mode
-v	Verbose
-w secs	Timeout for connects and final net reads
-X proto	Proxy protocol: "4", "5" (SOCKS) or "connect"
-x addr[:port]	Specify proxy address and port
-z	Zero-I/O mode [used for scanning]

Port numbers can be individual or ranges: lo-hi [inclusive]