1. USANDO WIRESHARK

Para el uso del programa, puede consultar las páginas de manual del sistema o la documentación en línea (Wireshark Documentation), disponible en http://www.wireshark.org/docs/

La herramienta de captura `wireshark` se encuentra disponible para múltiples sistemas operativos y puede utilizarse para analizar capturas que hayan sido guardadas con anterioridad. Alternativamente, puede resultarle interesante utilizar la herramienta `cloudshark` (http://www.cloudshark.org/) para analizar capturas guardadas sin tener `wireshark` instalado en su PC.

1.1 Captura de tráfico

Para capturar el tráfico que está ocurriendo en este momento en la red, hay que usar el menú *"Capture"*, primero definiendo la interface o interfaces donde se va quiere capturar el tráfico y luego iniciando la captura.

En ese momento comienza la captura de tráfico. Una ventana informa de los datos fundamentales de la captura según ésta se va produciendo (número total de tramas y número de tramas por cada uno de los principales protocolos)

Para detener la captura, pulse *'Stop'* en la ventana de información sobre el proceso de captura. Las tramas capturadas se cargan automáticamente en la pantalla principal de `wireshark`.

Puede guardar la captura realizada en un fichero (para su posterior análisis utilizando `wireshark` o `cloudshark`).

1.2 Examinando tramas

La lista de tramas capturadas aparece en la parte superior de la pantalla principal de `wireshark`, indicando origen, destino, protocolo, etc. Al seleccionar una trama, en la parte inferior puede verse el detalle de la trama, incluyendo los datos de cada cabecera de cada trama. Wireshark ofrece dos visiones del contenido de la trama, una decodificada y otra con el contenido en hexadecimal. En la decodificada se ofrece la información dividida por cabecera, con lo que es fácil ir a la cabecera concreta que quiera analizar. Por ejemplo, en una trama podemos ver cabeceras Ethernet, IP, TCP y HTTP. Desplegando la información de una cabecera se pueden ver los distintos campos de dicha cabecera. Si se pulsa en un campo cualquiera se resalta dicho campo en la secuencia hexadecimal que representa la trama en la otra ventana. Los campos entre corchetes, que aparecen en la trama decodificada, no son datos incluidos en la trama sino información adicional que wireshark ha podido deducir a partir de la trama. Por ejemplo, en un Echo request se incluye un [Response In:] que indica el número de paquete en la captura wireshark en la que se puede encontrar el Echo reply correspondiente (si no aparece es que wireshark no ha podido determinarlo o que la trama del Echo reply no se ha capturado). Y en el Echo reply hay un [Response to] en el que se indica el número de paquete en la captura wireshark en la que encontrar el Echo request correspondiente.

1.3 Filtros

En `wireshark` existen dos tipos de filtros: captura y visualización, que además tienen una sintaxis diferente.

Los filtros de captura en `wireshark` no son más que un texto que sigue el formato del comando *'tcpdump'*. En el apéndice I describimos ligeramente las características de este lenguaje de definición de filtros. Este tipo de filtros se emplean para discriminar qué tipos de paquetes son capturados (y guardados en el fichero de captura) y cuáles no. Estos filtros se utilizan normalmente cuando se va a capturar tráfico

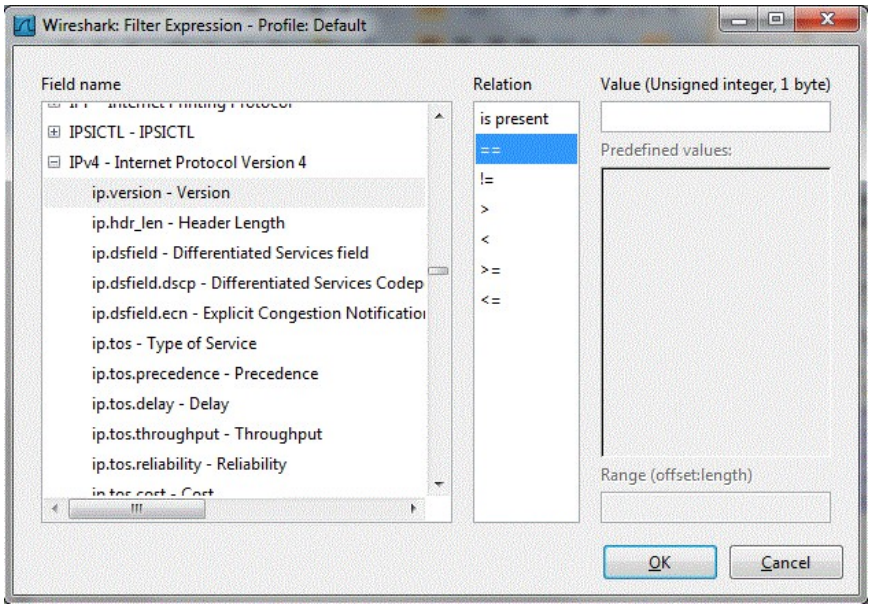
durante un periodo largo, y el tamaño de lo que se va a capturar puede convertirse en un problema por lo que es mejor discriminar antes de capturar.

Cuando se hacen capturas pequeñas, suele ser mejor capturar todos los paquetes y establecer un filtro de visualización para ver solo aquellos que nos interesan. Esto tiene la ventaja de poder cambiar el filtro y ver otros paquetes si nos damos cuenta de que nuestra selección inicial no era acertada (cosa que con los filtros de captura no podemos hacer porque ya no hemos capturado esos otros paquetes). En principio durante las prácticas usaremos filtros de visualización. El filtro de visualización se escribe en una ventana de "Filtro" en la parte superior de la ventana Wireshark. La sintaxis de los filtros de visualización (que es diferente de la de los de captura) se describe en el siguiente enlace:

https://www.wireshark.org/docs/wsug_html_chunked/ChWorkDisplayFilterSection.html

Es importante leer dicha página y las dos siguientes enlazadas desde la misma para entender cómo trabajar con filtros de visualización. También en el propio programa wireshark, en Help->Manual Pages, se puede acceder a información de los filtros wireshark. Una referencia más detallada de los filtros de visualización que podemos hacer en wireshark es: https://www.wireshark.org/docs/dfref/ (que se da como referencia, aunque no es necesario verlo en detalle de cara a la práctica).

Wireshark también ofrece una ventana de diálogo para ayudar a crear filtros de visualización. Al lado de la ventana de filtro de visualización hay un botón "Expression" y al pulsarlo aparece una ventana de diálogo. La ventana es básicamente una lista de protocolos, cada protocolo se puede extender para ver campos dentro del protocolo:



Para ir directamente a un protocolo se puede teclear las primeras letras del mismo. Y luego permite comprobar el valor de un campo respecto a un cierto valor. Al hacer OK la expresión aparecerá en la ventana de filtro de visualización. Así podemos crear cualquier expresión para incluir en un filtro. Luego podemos combinarla con otras expresiones usando los operadores que se describen en las páginas de ayuda enlazadas arriba (*and, or, not*, etc.). Por lo tanto, esta ayuda es muy útil para aprender a crear filtros, aunque luego normalmente con la práctica escribamos el filtro directamente sin la ayuda de la ventana de "Filter Expression".

APÉNDICE I: Filtros de captura en Wireshark

Los filtros de captura siguen la sintaxis de *'tcpdump'* y pueden hacerse por protocolo, por host, por puerto, etc. Los filtros pueden ser combinados mediante operadores booleanos ('and', 'or' y 'not'). Para eliminar problemas de precedencia de operadores pueden utilizarse paréntesis.

Una expresión en un filtro consiste en una o más primitivas. Las primitivas consisten normalmente de un identificador (nombre) precedido de uno o más calificadores que parametrizan el funcionamiento de la primitiva.

Los posibles calificadores son de uno de los tipos siguientes:
1- *Type*: Identifican si hablamos de una dirección de host, de red o de un número de puerto.
2- *Dir*: Especifica una dirección de transferencia desde o hacia el identificador. Las posibles direcciones son 'src' (fuente), 'dst' (destino), 'src or dst' (o bien fuente o bien destino), 'src and dest' (fuente y destino)
3- *Proto*: Restringe la captura a un protocolo particular. Posibles valores son 'ether', 'fddi', 'ip', 'rarp', 'tcp, 'udp', etc.

Las principales primitivas válidas son:
1- `dst host <host>`: verdadera si la dirección IP de destino del paquete es el host especificado, que puede ser un nombre o una dirección IP. P.e: '`dst host it015`' ó '`dst host 163.117.244.212`'
2- `src host <host>`: análogo al anterior pero con la dirección IP fuente del paquete.
3- `host <host>`: verdadera si la dirección IP fuente o destino del paquete es host.
4- `ether dst <host>, ether src <host>, ether host <host>`: análogos a los tres anteriores pero con direcciones ethernet.
5- `dst net <net>, src net <net>, net <net>`: análogas a las tres primeras primitivas pero con direcciones de red en lugar de direcciones de host.
6- `dst port <port>, src port <port>, port <port>`: análogas a las tres primeras primitivas pero con números de puerto en lugar de direcciones de host.
7- `ip proto <protocol>`: verdadera si el paquete es un paquete ip llevando por encima el protocolo de nivel superior `protocol`. `Protocol` puede ser icmp, igrp, udp o tcp. Como tcp, udp e icmp son también identificadores, deben ser escapados con '\'. Por ejemplo, para quedarnos con los paquetes ip que por encima usan tcp, escribiríamos '`(ip proto \tcp)`'.
8- `ether broadcast` o `ip broadcast`: verdaderas si el paquete es un broadcast ethernet o ip, respectivamente.
9- `ether multicast` o `ip multicast`: verdaderas si el paquete es un multicast ethernet o ip, respectivamente.
10- `ether proto <protocol>`: verdadera si el paquete es un paquete ethernet llevando por encima el protocolo de nivel superior protocol. Protocol puede ser ip, arp o rarp. Como ip, arp y rarp son también identificadores, deben ser escapados con '\'. Por ejemplo, para quedarnos con los paquetes ethernet que por encima usan ip, escribiríamos '(ether proto \ip)'.
11- `ip, arp, rarp`: abreviaturas para '`ether proto <protocol>`', donde <protocol> es ip, arp o rarp.
12- `tcp, udp, icmp`: abreviaturas para '`ip proto <protocol>`', donde <protocol> es tcp, udp o icmp.

Si, por ejemplo, queremos quedarnos sólo con los paquetes tcp con fuente it012 y destino it015, pondríamos '`(src host it012) and (dest host it015) and (ip proto \tcp)`'.
Una vez activado un filtro, si efectuamos una captura sólo se capturará el tráfico que cumpla las condiciones de filtrado. Podemos hacerlo actuar sobre el tráfico de un fichero ya cargado simplemente usando la opción *'Reload'* del menú *'File'*.

Por otro lado tenemos los filtros de visualización, que no influyen sobre qué paquetes se capturan, pero sí sobre qué paquetes son mostrados en la ventana principal de `wireshark`. Su sintaxis no es idéntica a la que se emplea para los filtros de captura (puede consultar la ayuda de *Wireshark: Wireshark Filter* para obtener más información). En general los filtros de captura son importantes cuando se captura tráfico durante largos periodos de tiempo, con lo que se quiere evitar tener que guardar muchos datos no necesarios. Pero si se comete un error con el filtro, nos podemos encontrar con que no tenemos el tráfico

que necesitamos. Los filtros de visualización son más prácticos cuando el almacenamiento de la captura no es un problema (como las capturas sencillas durante tiempo reducido de esta práctica) pues tenemos todo el tráfico y podemos cambiar el filtro si no estamos viendo lo que esperábamos.