Infraestructura y Tecnología de Redes
Curso 2020-2021

Práctica 2: *Analizadores de red*

Introducción

A la hora de trabajar con una red, es muy importante la información que se pueda obtener de esta, una información de calidad nos permitirá gestionar la red de forma sólida y evitar problemas.

Para obtener esta información, un tipo de herramienta imprescindible son los analizadores de red. Un analizador de red normalmente es un programa (también los hay de hardware) que intercepta y registra el tráfico que circula por una red. Habitualmente, los analizadores de red trabajan en tiempo real y entienden las cabeceras de los datos, de forma que son capaces de mostrar el contenido de estas, los diferentes campos de cada paquete, el protocolo al que pertenecen, etc.

1. Herramienta utilizada: Wireshark

Wireshark [1] es uno de los analizadores de red más poderosos del mercado. Además, es gratuito (tiene licencia GPL) y multi-plataforma. Wireshark se considera habitualmente como el estándar de facto de la industria. Entre sus características se encuentran: captura en tiempo real y análisis offline, navegador de paquetes, interfaz gráfica muy potente y fácil de usar (también existe una versión TTY llamada tshark), filtros de captura y de pantalla, análisis de VoIP, lee y escribe diferentes formatos de archivo de captura de tráfico, es compatible con Ethernet, 802.11, PPP / HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, etc.

Para que Wireshark pueda capturar paquetes utilizando directamente las interfaces de red, es necesario que se ejecute con privilegios de administrador.

2. Funcionamiento de Wireshark

2.1. Captura

Capturar tráfico con Wireshark es muy sencillo, basta con abrirlo con una cuenta de root, configurar "**Capture**" $\rightarrow$ "**Options**" con la interfaz que se quiere

utilizar y pulsar "**Capture**" → "**Start**".

2.2. Filtro

Es probable que al capturar tráfico de cualquier red, una gran cantidad de los paquetes no nos interesen. Estos paquetes dificultan el análisis y aumentan de forma innecesaria el tamaño de los archivos de captura. Para evitarlo, es necesario filtrar esta información.

Para ello, disponemos de dos alternativas:

1. Filtro de captura, el propio programa descarta los paquetes que no interesan, se configura en "**Capture**" → "**Options**" antes de empezar a capturar.

2. Filtro de pantalla, el programa acepta y registra todos los paquetes, pero sólo se muestran por pantalla aquellos que interesan. Estos filtros se configuran en la parte inferior de la ventana de Wireshark. Este tipo de filtrado se puede combinar con la anterior.

La sintaxis de los filtros de captura es la misma que utiliza `tcpdump` (la sintaxis de los filtros de pantalla es ligeramente diferente, y se puede utilizar un asistente para configurarlos).

2.2.1. Sintaxis de los filtros de captura

Los filtros definen el tipo de paquetes que se capturan, de forma que un paquete será descartado si la expresión del filtro evalúa **false** y será capturado si se evalúa **true**. Las expresiones de filtrado se escriben siempre en minúsculas y consisten en una serie de primitivas conectadas mediante operadores lógicos (and, or, not).

Cada primitiva consta de un identificador (un identificador puede ser una dirección física, una dirección de red o el identificador especial **broadcast**) precedido por, como mínimo, uno de los siguientes calificadores:

- Calificadores de tipo:
  - **host**: Hace referencia a un nodo concreto. Este es el calificador por defecto, Wireshark lo utiliza en el caso de no especificar un calificador.
  - **net**: Hace referencia a una red.
  - **port**: Hace referencia a un puerto cualquiera (tanto puede ser TCP como UDP).
- Calificadores de dirección:
  - **src**: Hace referencia al origen del paquete.

- **dst**: Hace referencia al destino del paquete.
- **src or dst**: Hace referencia al origen o el destino del paquete, indistintamente.
  - Es el calificador de dirección por defecto.
- **src and dst**: Hace referencia al origen y al destino del paquete, simultáneamente.

- Calificadores de protocolo:
  - **arp, ip, icmp, tcp, udp, ether, etc.** Hacen referencia al protocolo en cuestión. No hay calificador para los protocolos de capa de aplicación.

A continuación vemos algunos ejemplos de filtros de captura:

`host 158.42.180.62` captura todas las tramas dirigidas a o enviadas por el nodo con esta dirección.

`src net 158.42` captura todas las tramas enviadas por nodos de la red 158.42.0.0/16.

`src or dst port 25` captura todos los paquetes enviados al puerto 25 o desde el puerto 25 (UDP o TCP).

`udp and src 158.42.148.3` captura todos los paquetes UDP enviados desde del host 158.42.148.3.

`dst 158.42.181.4 and (udp or icmp)` captura todos los paquetes UDP o ICMP dirigidos al host 158.42.181.4.

2.2.2. Sintaxis de los filtros de pantalla

Wireshark utiliza filtros de pantalla para el filtrado general de paquetes durante la visualización.

El filtro de pantalla no es un filtro de captura. Los filtros de captura (como tcp port 80) no deben confundirse con los filtros de pantalla (como tcp.port == 80).

Ejemplos:

- Mostrar solo tráfico SMTP (puerto 25) e ICMP :

  ```
  tcp.port eq 25 or icmp
  ```

- Mostrar solo el tráfico en la LAN (192.168.xx), entre estaciones de trabajo y servidores, sin Internet:

  ```
  ip.src==192.168.0.0/16 and ip.dst==192.168.0.0/16
  ```

- Restringir la visualización a solo paquetes de un fabricante de dispositivo específico. Por ejemplo, solo para máquinas DELL:

  ```
  eth.addr[0:3]==00:06:5B
  ```

- Coincidir con las solicitudes HTTP donde los últimos caracteres de la uri son los caracteres "gl = se":

  ```
  http.request.uri matches "gl=se$"
  ```

  Nota: El carácter $ es un carácter de puntuación PCRE que coincide con el final de una cadena, en este caso el final del campo http.request.uri.

- Filtrar por un protocolo (por ejemplo, SIP) y filtrar las direcciones IP no deseadas:

  ```
  ip.src != xxx.xxx.xxx.xxx  ip.dst != xxx.xxx.xxx.xxx
  sip
  ```

Gotchas:

- Algunos campos de filtro coinciden con varios campos de protocolo . Por ejemplo, `ip.addr` coincide con las direcciones IP de origen y destino en el encabezado IP. Lo mismo es cierto para `tcp.port`, `udp.port`, `eth.addr` y otros. Es importante tener en cuenta que

  ```
  ip.addr == 10.43.54.65
  ```

  es equivalente a

  ```
  ip.src == 10.43.54.65 or ip.dst == 10.43.54.65
  ```

- Esto puede ser contrario a la intuición en algunos casos. Suponga que queremos filtrar el tráfico hacia o desde 10.43.54.65. Podríamos intentar lo siguiente:

  ```
  ip.addr! = 10.43.54.65
  ```

  que es equivalente a

  ```
  ip.src! = 10.43.54.65 or ip.dst! = 10.43.54.65
  ```

  Esto se traduce en pasar todo el tráfico excepto el tráfico con una dirección IPv4 de origen de 10.43.54.65 y una dirección IPv4 de destino de 10.43.54.65, que no es lo que queríamos.

  En su lugar, necesitamos negar la expresión, así:

  ```
  ! (ip.addr == 10.43.54.65)
  ```

  que es equivalente a

  ```
  ! (ip.src == 10.43.54.65 or ip.dst == 10.43.54.65)
  ```

3. Guión de la práctica

Todas las respuestas deben ir acompañadas de las gráficas correspondientes que demuestren los resultados de lo que se está explicando, configurando y/o requiriendo. Excepto por las preguntas que se encuentran en el cuestionario de este práctica

Siga el siguiente guión y responda a las preguntas que se plantean:

3.1. Diseño de filtros

Cuestionario

A continuación deberá de escribir unos filtros de captura para Wireshark que cumplan determinadas condiciones, es muy recomendable que compruebe que su funcionamiento es correcto antes de entregarlos.

1. **(1 punto)** Escriba un filtro que capture solo el tráfico DNS.

2. **(1 punto)** Escriba un filtro que capture solo el tráfico de la red del laboratorio.

3. **(1 punto)** Escriba un filtro que capture el tráfico HTTP hostname o domain que terminen en .org.

3.2. Tráfico HTML

Cuestionario

1. Ponga Wireshark a capturar tráfico (si utiliza un filtro de captura o de pantalla le será más fácil localizar lo que busca, sobre todo si está en el laboratorio).

2. Abra su navegador y conéctese a http://www.hypexr.org/

3. Detener la captura de tráfico e inspeccione los paquetes capturados para encontrar:

   - **(1 punto)** ¿Cuál es el servidor DNS que resuelve www.hypexr.org y cuál es el nameserver en el cual está registrado esta página Web?.

   - **(1 punto)** ¿Qué servidor Web HTTP utiliza?.

3.3. Tráfico SSH

Informe

1. **(1 punto)** Filtre en Wireshark para que capture solo tráfico SSH. Conéctese a través de SSH a otra máquina con `ssh usuario@DEIC-DCX`. Una vez iniciada sesión detenga la captura. Utilice la herramienta Flow Graph, haga una captura del gráfico desde el inicio de la negociación TCP hasta el envío de los 3 primeros datos cifrados. Resalte el método (dentro de un canal público) de intercambio de llaves utilizado.

2. **(1 punto)** ¿Cuál es el puerto local que se utiliza para la conexión SSH?

3. **(1 punto)** ¿Qué versión de SSH, cifrado, Message Authentication Code (MAC) y compresión se negoció?. Colocar una imagen.

3.4. Tráfico HTTPS

Informe

A continuación vamos a crear un servidor que sirve una página Web en HTTPS y utilizaremos Wireshark para ver cómo se puede capturar esta información y qué información podemos obtener.

1. Comenzaremos creando una llave RSA y un certificado para el servidor, abrimos una consola y los generamos con el siguiente comando:

   ```
   openssl req -new -x509 -out server.crt -nodes -keyout
   server.pem -subj /CN=localhost
   ```

   Con este pedido solicitamos a OpenSSL que nos genere un cerificado (**-NEW**) autofirmado (**-x509**) que se guardará en el archivo (**-out server.crt**), pedimos que nos genere una llave privada (**-keyout server.pem**) y que no la cifre (**-nodes**) y especificamos el CommonName del certificado (**-subj / CN = localhost**).

2. Una vez que tengamos el certificado y la llave privada, podemos abrir el servidor con el siguiente comando:

   ```
   openssl s_server -tls1_2 -www -cipher AES256-SHA -key
   server.pem -cert server.crt
   ```

   En este caso estamos requiriendo a OpenSSL que nos abra un servidor que utilizará AES para cifrar (**-cipher AES256-SHA**) que utilizará la llave privada (**-key server.pem**) y el certificado (**-cierto server.crt**) que acabamos de crear y que nos responderá a las peticiones con un mensaje de estado (**-www**).

3. Ahora ejecutamos Wireshark para capturar el tráfico que se genere hacia y desde **localhost**. Se recomienda aplicar como filtro de pantalla **"ssl"**

4. Una vez iniciemos la captura del tráfico, en una nueva consola haremos la petición Web con el comando:

   ```
   printf 'GET / HTTP/1.0\r\n\r\n' | openssl s_client -ign_eof
   ```

5. **(1 punto)** Explica lo que has obtenido en esta captura. Colocar una imagen.

6. Ahora le diremos a Wireshark cuál es la llave privada que se está utilizando en esta comunicación, para hacer esto iremos a "**Edit**"→"**Preferences**", dentro de **Protocolos** buscaremos **SSL** y modificaremos la **RSA keys lists**, con los parámetros: **IP address - 127.0.0.1**, **Puerto - 4433**, **Protocolo - http** y **Key File - server.pem**. (En caso de que no se descifren los mensajes cambiar la IP address por **::1**). Finalmente al darle doble click sobre uno de los paquetes cifrados se habilitará una nueva pestaña **Decrypted SSL** al lado de **Frame**

7. **(1 punto)** ¿Qué cambios se han producido en los paquetes? ¿Cuál es la respuesta del servidor a la petición GET?. Colocar la imagen obtenida al ingresar a Follow SSL Stream.

4. Calendario

A continuación se describe el calendario de los hitos relativos a la práctica:

- **Prática**: El 12/03/2021.
- **Entrega**: El 18/03/2021 hasta las 23:55.

5. Condiciones de entrega

- La entrega de la práctica se hará a través del campus virtual.
- **No** se aceptarán informes entregados fuera de plazo.
- Cada grupo debe entregar un informe en formato **PDF que contenga el número de práctica, el número de grupo y el primer apellido de cada alumno** (ej. p1-a1-carpio-miranda.pdf) y las respuestas a los diferentes apartados de la práctica. En caso de **no** seguir el formato se **restará 1 punto de la nota**.

Referencias

[1] Riverbed Technologies. Wireshark. `http://www.wireshark.org/`.