

## Disecionamos el tráfico de red

# WIRESHARK

Cómo crear un registro de la actividad de nuestra red con Wireshark. **POR KURT SEIFRIED**

**W**ireshark [1], el sniffer de paquetes antes conocido como Ethereal, es una herramienta de tenencia obligada para cualquier administrador de sistemas. Para diagnosticar problemas de red u observar transacciones con servidores, supone la herramienta perfecta. Igual que la mayoría, si no todos, de los programas orientados a la captura de tráfico de red para Linux, Wireshark utiliza *libpcap*, que proporciona una interfaz independiente del sistema para la captura de paquetes, por lo que se evita la necesidad de escribir rutinas propias para cada sniffer de paquetes (*tcpdump*, *Snort*, *Wireshark*, etc.). En tanto en cuanto nuestro sistema operativo (Linux, \*BSD, HP-UX, Solaris, Windows, etc.) y el software soporten *libpcap*, se podrán capturar los paquetes.

## Instalación de libpcap y Wireshark

Libpcap viene incluido en la mayoría de los sistemas operativos. Wireshark también está incluido casi siempre (al menos en Linux y BSD). A veces está dividido en dos paquetes: uno que comprende utilidades de backend como *tshark* o *mergcap*, y otro que consiste en la interfaz gráfica (GUI). En caso de no estar instalado, en Fedora y similares basta con ejecutar

```
yum install libpcap
yum install wireshark-gnome
```

En Debian es igual de sencillo

```
apt-get install libpcap0
apt-get install wireshark
```

Uno de los problemas derivados de usar las versiones de Wireshark proporcionadas por las distribuciones, es que suelen ser bastante antiguas (por ejemplo, Fedora 11 viene con la v1.1.3). Aún con todo, las series 1.2 (y v1.3.0, que podría estar disponible para cuando salga publicado este artículo) incluyen buena parte de las nuevas funcionalidades, como soporte de protocolo, soluciones de errores e integración de geolocalización (de la que hablaremos más adelante).

Para compilar Wireshark desde los fuentes tendremos que descargarlo obteniendo la última versión estable [2] o, los más valientes, descargando mediante SVN una de las últimas compilaciones automatizadas [3]. Una vez descomprimido, la instalación es muy sencilla:

```
./configure
make
make install
```

Para compilar Wireshark necesitaremos satisfacer antes algunas dependencias. Como mínimo necesitaremos *libpcap*, pero es recomendable también disponer de GnuTLS, PCRE (*Perl-Compatible Regular Expressions*) y GeoIP

para disfrutar de las funcionalidades avanzadas.

## Ejecutar Wireshark

Para ejecutar Wireshark de modo que pueda capturar datos en vivo, no hay más remedio que hacerlo como root. Wireshark ha tenido varios problemas de seguridad en el pasado que permitían a potenciales atacantes remotos ejecutar código enviando datos manipulados para que Wireshark los procesara. Hay dos formas relativamente simples de evitar la ejecución de Wireshark con permisos de root. La primera de ellas es usando un programa para la captura de paquetes, como *dumpcap* o *tcpdump*, para luego examinar los archivos de captura con Wireshark con permisos de usuario no privilegiado.

El segundo método consiste en ejecutar Wireshark como root, pero dentro de un entorno contenido y controlado. Al ejecutar Wireshark en el interior de una máquina virtual, como VMware, en un primer momento se podría pensar que sólo sería posible ver los datos de red enviados desde y hacia esa máquina virtual en particular. En realidad, *libpcap* accede directamente al kernel, y por tanto a la tarjeta, a un nivel lo suficientemente bajo como para ver todo el tráfico que pasa por la interfaz de red física. Es decir, se puede ejecutar Wireshark desde una máquina virtual y capturar fácilmente el tráfico de red. La parte negativa es que cualquiera que posea acceso como root a la máquina virtual (al menos bajo VMware Server) podrá ver todo el tráfico de red de la máquina física subyacente, así como el de cualquier máquina virtualizada que haga uso de la misma interfaz.

## Filtros de Captura Libpcap

Capturar el tráfico de cualquier red con algo de actividad es como tratar de beber de una boca de incendios. La buena noticia es que casi todos los programas de captura que usan *libpcap* permiten el uso de los comandos de filtrado disponibles con esta librería (e incluso a veces, como es el caso de Wireshark, también de sus propios filtros). Lo malo es que el juego de filtros de *libpcap* es relativamente limitado, soportando principalmente la especificación de direcciones y puertos (ya que es consciente del protocolo usado).

Como mínimo, se puede usar para minimizar la cantidad de datos que se escribirán a disco o que tendrá que procesar Wireshark. Por ejemplo, podemos usar la siguiente sintaxis para capturar tráfico web con *libpcap*:

```
dst port 80 or 443
```

La página de manual de *tcpdump* cubre en detalle la sintaxis para los filtros de *libpcap*.

## Diseccionadores de Protocolos

El factor diferencial de Wireshark frente al resto de programas capturadores de paquetes es su gran número de diseccionadores de protocolos. En la práctica, Wireshark es capaz de comprender a fondo los protocolos principales (SSH, Telnet, NTP, etc.) y no sólo mostrar información en un formato más legible, sino que se puede utilizar un mayor número de opciones en el filtrado. Con Wireshark, no sólo podemos filtrar tráfico web,

```
tcp.dstport == 80 or 443
tcp.dstport == 443
```

sino que también podemos escoger componentes especiales específicos de una conexión HTTP, como por ejemplo códigos de respuesta. Si sólo quisiéramos mostrar los códigos 404:

```
http.response.code == 404
```

O también podríamos buscar todas las cookies que contengan la cadena *sessionid*. HTTP no es el único protocolo que comprende Wireshark; según el último cálculo, Wireshark contaba con alrededor de 1000 diseccionadores de protocolos (aunque no todos son tan completos como el de HTML). Por desgracia, estos diseccionadores de protocolos pueden suponer también un problema significativo, con unas 85 vulnerabilidades conocidas [4], que van desde simples denegaciones de servicio hasta desbordamientos de búfer con ejecución de código.

## GeoIP

Wireshark soporta ahora GeoIP, un juego de librerías que se pueden usar para realizar consultas a las bases de datos de MaxMind Geographic IP (básicamente se trata de un listado de direcciones IP y redes, así como de países a los que pertenecen). De manera gratuita, podemos acceder a datos de países, lo que significa que podemos crear filtros en Wireshark para mostrar el tráfico de un país específico.

```
ip an ip.geoip.country >
== "China"
```

Como pueda ser China.

## Creando un Registro de Red

A menos que ejecutemos Wireshark las veinticuatro horas del día, para poder analizar un problema habremos de recrearlo mientras examinamos el tráfico de red con Wireshark. ¿O no?

Las buenas noticias son que los discos duros son ya suficientemente baratos, así, un disco de un terabyte de capacidad permite guardar treinta gigabytes diarios durante un mes, o incluso más si la red no tiene mucha actividad durante los fines de semana. El programa *tshark* hace asequible el almacenamiento cíclico de cantidades de datos determinadas. La opción *-b*

```
tshark -i eth0 >
-b filesize:10240 -b >
files:1000 -w if-eth0
```

monitoriza la interfaz *eth0* creando hasta 1000 archivos de unos 10MB cada uno (un total de 1GB), con marcas de tiempo como nombres de archivo (YYYYMMDDHHMMSS):

```
if-eth0_00001_20090920041232
if-eth0_00002_20090920041252
if-eth0_00003_20090920041258
```

Estos archivos se pueden unir posteriormente con *mergcap* en caso de que el tráfico que nos interesa esté repartido entre varios de ellos. La ventaja de hacer capturas cíclicas es que no es necesario rotar los archivos, pudiendo especificar una cantidad máxima de datos sin tener que preocuparnos de que se agote el espacio disponible en disco.

Otra forma de capturar los datos de red para su posterior análisis es mediante la herramienta *tcpdump*. Las últimas versiones de *tcpdump* suelen soportar la opción *-C*, que comienza escribiendo a un nuevo archivo de captura al alcanzar un número de bytes determinado; mientras que la opción *-W* limita el número de archivos creados, sobrescribiendo los más antiguos al alcanzar el número máximo permitido, conformando de este modo una captura cíclica.

Nótese que no es necesario capturar todo el tráfico de red para disponer de un registro útil en el rastreo de problemas; simplemente registrando el tráfico DNS (puerto 53 TCP y UDP) cubrimos muchos problemas de seguridad,

como puedan ser ciertas descargas ilícitas que suelen generar consultas DNS para dominios extraños.

Por otro lado, se podría registrar sólo el tráfico entrante por el puerto 80, con lo que se tendría un registro de todas las peticiones web recibidas; no así de las respuestas, que ocuparían mucho más espacio. Como si de una cámara de vídeo se tratase, no evitaríamos potenciales incidentes de seguridad, pero tendríamos una idea muy aproximada de lo que ocurrió y, con suerte, de cuántos sistemas se han visto afectados por el ataque.

## Actualización a Upgrade 2.0

Nuestro agradecimiento a Jan Andrejkovic por añadir una nueva herramienta a la columna "Upgrade 2.0" [5]. Fedora incluye un programa llamado Presto [6] que hace uso de *DeltaRPMs* para proporcionar actualizaciones de menor tamaño. En un primer test, una actualización normal hubiera necesitado 972MB de datos descargados, pero con Presto no eran más que 224MB (un ahorro impresionante). Fedora 11 incluye ahora un paquete *yum-presto* (que no hay que confundir con el paquete *presto*, que es un motor orientado a gráficos), un plugin para el gestor de paquetes *yum*. La instalación es muy sencilla:

```
yum install yum-presto
```

Primero se debe actualizar manualmente */etc/yum.repos.d/fedora-updates.repo* de modo que incluya o bien una *baseurl* o un *mirrorurl* apuntando al sitio que contiene los RPMs para presto (están firmados con la clave GnuPG de Jonathan Dieter). Alternativamente, podemos usar nuestro propio repositorio y crear los RPMs mediante *presto-utils*. En los casos con más de un sistema, ésta puede ser la mejor opción. ■

## RECURSOS

- [1] Wireshark: <http://www.wireshark.org/>
- [2] Código fuente para la versión estable de Wireshark: <http://www.wireshark.org/download/src/>
- [3] Código fuente para la versión de desarrollo de Wireshark: <http://www.wireshark.org/download/automated/src/>
- [4] Vulnerabilidades de seguridad de Wireshark: <http://www.wireshark.org/security/>
- [5] "Upgrade 2.0" por Kurt Seifried, Linux Magazine, Número 58, pág. 8.
- [6] Presto: <https://fedorahosted.org/presto/>