Proyecto de fin de carrera:

Monitorización remota en redes de área local

Autor: Rafael Micó Miranda

Tutor: D. Antonio J. Estepa Alonso

Ingeniería de Telecomunicación

Departamento de Ingeniería de Sistemas y Automática

Área de Ingeniería Telemática

Escuela Superior de Ingenieros Universidad de Sevilla

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción y objetivos

En la actualidad, cada día un mayor número de redes de pequeñas, medianas o grandes empresas que prestan servicios a través de Internet o que necesitan tomar servicios desde Internet establecen sistemas cortafuegos o *firewalls* para proteger sus sistemas de posibles ataques e intrusiones desde el exterior. Desde la simple configuración adecuada de los *routers* que les conectan a la *World Wide Web* hasta la instalación de complejas zonas desmilitarizadas o *DMZs*, dichos usuarios confían en el éxito que tendrán sus políticas de seguridad y en su funcionamiento autónomo y automatizado para despreocuparse, hasta cierto punto, del estado de su red.

Sin embargo esta despreocupación puede ser falsamente provocada: según el Instituto Nacional de Estadística durante el año 2003/2004 un 33,8 % de las empresas españolas declararon haber tenido algún problema de seguridad informática, mientras que la cifra durante el año 2004/2005 se redujo a un 27,3 %. Si sumamos a esta información que más del 90 % de las empresas españolas de más de diez empleados dispone de un acceso a Internet, cabe pensar que realmente el problema no da lugar a la citada despreocupación.

En los *firewalls* se implementan muchas funciones de control y sus configuraciones pueden ser de una cierta madurez y complejidad. Existe, por tanto, una necesidad creciente de *monitorizar* el estado de los sistemas *firewall* así como el uso de los recursos de la red de área local, y esta monitorización no debe servir únicamente para realizar una gestión de la contabilidad sino que debe usarse también como un agente de prevención de ataques externos o internos. Por lo tanto, de las cinco áreas funcionales de gestión definidas en el modelo OSI, este Proyecto se enmarca dentro de las áreas de Gestión de la Seguridad y de la Gestión de Contabilidad.

Como podemos intuir, las topologías de las redes o de los sistemas de cortafuegos que establezcan en las empresas son muy variables y pueden tener estructuras muy distintas. Las preguntas a las que trataremos de responder a lo largo de este Proyecto intentan no centrarse sobre ninguna estructura específica, sino que abordan distintos aspectos que ilustraremos con un ejemplo genérico:

> Si administramos una red grande de una empresa corporativa con distintas subredes para distintos departamentos cada una de ellos con sus propios *firewalls*, distintas *DMZs* para distintos accesos a Internet, ¿cómo monitorizar las distintas subredes? ¿Cómo analizar las necesidades de tráfico y ancho de banda de los departamentos? ¿Cómo comprobar el estado o las alarmas de los distintos *firewalls* y sistemas alojados en distintos segmentos de la red? ¿Cómo centralizar esa información? ¿Cómo hacerlo de forma eficiente?

Estas son las distintas preguntas a las que vamos a intentar dar respuesta en este Proyecto de Fin de Carrera.

La falta de herramientas de *software* libre que solucionen los problemas anteriomente mencionados motiva la realización de este Proyecto de Fin de Carrera. Por lo tanto, el objetivo de este Proyecto es la creación de un sistema de monitorización remota que cumpla las siguientes necesidades:

- Será un sistema de monitorización remota, en el que la información de monitorización será generada en los sistemas remotos y enviada a otros sistemas para su almacenamiento y procesado.
- Será un sistema centralizado.
- Será un sistema escalable y adaptable al tamaño de la red que se desee monitorizar.
- Será un sistema que en *software* se basará en **GNU/Linux**.

El resto del presente documento de este Proyecto de Fin de Carrera se estructura en los siguientes capítulos autocontenidos:

Primero, en el capítulo 2, "*Análisis de requisitos y descripción funcional del sistema*", haremos una reflexión acerca de este Proyecto, realizando un análisis más detallado de la problemática a solventar y haciendo una aproximación al sistema que se pretende desarrollar.

Tras ello, en el capítulo 3, "*Planificación y costes*", planteamos la planificación y evaluación de costes estimadas para el desarrollo de este Proyecto.

En el capítulo 4, "*Monitorización estadística de tráfico*", hablaremos sobre la recogida de información estadística del tráfico de la red, apoyándonos para ello en el protocolo *NetFlow*.

En el capítulo 5, "*Monitorización de tráfico*", veremos las posibilidades de recogida de información completa de tráfico desde un sistema remoto, haciendo uso de configuraciones específicas de *iptables*.

En el capítulo 6, "*Recogida de registros*", hablaremos sobre la recogida de registros, de informes de incidencias y de estado de las máquinas que dan soporte a nuestra red, basándonos para ello en *syslog-ng*.

Tras ello, en el capítulo 7, "*Tratamiento de la información*", veremos algunas posibilidades para la visualización y manejo de la información de gestión recogida con las utilidades anteriores.

En el capítulo 8, "*Interfaz de configuración*", trataremos algunos aspectos acerca de la interfaz de configuración de las utilidades seleccionadas desarrollada en este Proyecto.

Finalmente, en los apéndices del documento se tratarán, entre otros, aspectos de instalaciones y configuraciones de las utilidades empleadas, así como aspectos del desarrollo de la interfaz.

Capítulo 2

Análisis de requisitos y descripción funcional del sistema

2.1. Introducción

Presentamos en este capítulo un estudio más detallado de la problemática que pretende solucionar este Proyecto de Fin de Carrera planteando:

- Un catálogo de requisitos a cumplir por el sistema a desarrollar.
- Un esquema con las funcionalidades de la solución adoptada.

2.2. Catálogo de requisitos

Mediante reuniones del equipo de trabajo y el estudio de las necesidades a las que se pretende dar solución con este proyecto, se identifican los requisitos listados a continuación:

REQ1: Será un sistema de monitorización remota en el que la información de monitorización será generada en los sistemas remotos y enviada a otros sistemas para su almacenamiento y procesado. Con esto se busca la alteración mímina, en términos de instalación y configuración, de los equipos de una red ya existente en estado «productivo» así como afectar lo menos posible a su rendimiento, dejando las tareas más necesitadas de recursos a otros sistemas diferentes.

REQ2: Será un sistema centralizado, lo que acarreará sus ventajas e inconvenientes clásicos. Por una parte dispondremos de una centralización de la recogida y procesado de la información, mayor facilidad de configuración y uso y menor gasto de instalación, mientras que por otro lado careceremos de redundancia y exponemos nuestro sistema centralizado a mayores problemas de seguridad (como, por ejemplo, frente a ataques de denegación de servicio).

REQ3: Será un sistema escalable y adaptable a las necesidades y al tamaño de la red a la que se desee monitorizar así como a los cambios futuros que en la misma se deseen realizar, de manera que se permita una rápida instalación y configuración en los nuevos sistemas incorporados a la red y en el sistema centralizado de monitorización. En lo que sigue en el desarrollo del Proyecto no nos ceñiremos a ninguna estructura de red concreta sino que plantearemos topologías específicas en cada caso según la necesidad y a modo de ejemplo.

REQ4: La información de monitorización será, en caso de necesidad, capaz de atravesar distintas redes sin alterar su contenido para llegar hasta el sistema centralizado de monitorización.

REQ5: Será un sistema que en *software* se basará en **GNU/Linux**. Esto nos permitirá poder escoger dentro del abanico de utilidades y herramientas ya existentes para estas plataformas buscando además la gratuidad del *software* escogido.

REQ6: Se buscará la mayor la seguridad posible tanto de las utilidades seleccionadas como de la información de monitorización.

A modo de ejemplo se muestra en la Figura 2.1 en la que se ilustra la hipotética red de una empresa en la que a través de una *DMZ* intermedia se permite el acceso a sus servidores desde Internet a la vez que se protegen los equipos de los empleados o servidores internos de la propia empresa.

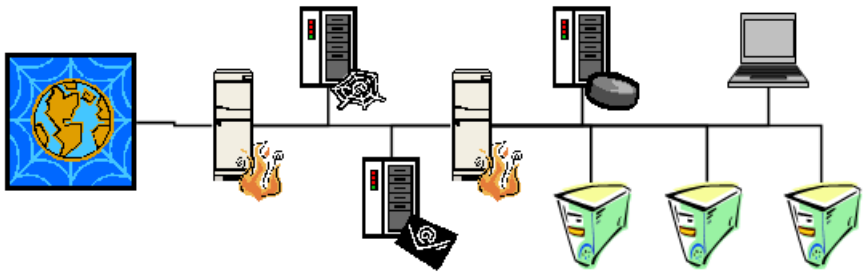


Figura 2.1: Ejemplo de red con *DMZ* intermedia

En dicha Figura 2.1 se pueden identificar, de izquierda a derecha, los siguientes elementos:

- Un acceso a Internet.
- El *firewall* exterior de la *DMZ*.
- Dentro de la *DMZ* se pueden localizar:
  - Un servidor *Web*
  - Un servidor de correo.

- Tras ello encontramos el *firewall* más interno de la *DMZ*.
- Ya en la red interna de la empresa, podemos encontrar:
  - Un servidor de archivos.
  - Distintos equipos de usuario.

Sobre este ejemplo ubicaremos en el siguiente punto 2.3 las funcionalidades propuestas en la solución planteada en este Proyecto, de manera que podamos abstraernos de esta red de ejemplo para implementar en cualquier otra topología de red nuestra solución.

2.3. Funcionalidades de la solución planteada

Tras ver los requisitos que deberá cumplir la solución de este Proyecto presentamos esquemáticamente las funcionalidades y principios de la solución planteada:

FUNC1: Se utilizará el sistema operativo basado en GNU/Linux **Debian Sarge**. La elección de este sistema operativo busca la utilización de *software* libre a la vez que persigue la seguridad de las aplicaciones que se seleccionen. Esto se debe a la rigurosa política de seguridad perseguida por el proyecto Debian que es impuesta en sus aplicaciones, haciendo que la versión *stable* (o versión reconocida como oficial) ofrezca unas altas cotas de seguridad y estabilidad en todos sus aspectos.

FUNC2: Se realizará una monitorización remota del uso de la red: el sistema realizará una recogida estadística de tráfico de la red. Esta recogida de información del estado de la red se efectuará de forma continua obteniendo información constantemente del estado de la red, de la carga que la atraviesa y de los servicios que están siendo utilizados. Este sistema deberá ser poco exigente en recursos (tanto de la red como de los sistemas que intervengan en la monitorización) dado su continuo funcionamiento, y se buscará también su automatización. Se realizará un procesado y análisis de la información ya almacenada en nuestro sistema centralizado de monitorización. El procesado buscará mejorar la legibilidad y la calidad de la información de monitorización generada por la red, realizándose en el sistema centralizado a partir de la información ya almacenada para no ocupar los recursos de los sistemas de la red.

FUNC3: Se hará una monitoritorización del tráfico de la red: por las limitaciones implícitas que poseerá la monitorización estadística del punto anterior, será necesario establecer en el sistema un proceso complementario de monitorización completa del tráfico de la red. Este sistema no actuará de forma continua sino que será selectivo tanto temporalmente (se hará funcionar en los intervalos de tiempo necesarios) como con la información a recoger (ya que en estados de alta carga de la red someter a la misma a un sobreesfuerzo por la extensa información de monitorización no sería adecuado).

FUNC4: Se efectuará una monitorización remota de *logs*: el sistema realizará la recogida de informes, alarmas y registros de los sistemas de la red. Estos registros serán los generados por los sistemas *firewall* de la red así como por los distintos servidores que pudiesen existir en la misma. Esta recogida también actuará de forma continua por lo que se buscará su poca exigencia de recursos. Igualmente, se realizará un procesado y análisis de la información ya almacenada en nuestro sistema centralizado de monitorización.

FUNC5: Se desarrollará una interfaz del sistema (GUI): se confeccionará una interfaz de configuración para las utilidades escogidas que facilite su uso y plantee un manejo cómodo a un administrador humano. En el desarrollo de dicha interfaz se buscará realizarlo adaptándonos a una estructura de tres capas (capa de presentación, capa de negocio y capa de datos) y dividiéndola en módulos (un módulo para cada utilidad escogida) para facilitar su desarrollo y su integración con otros proyectos.

Así, planteamos en la Figura 2.2, sobre la red ya mostrada y a modo de ejemplo, el esquema de nuestra propuesta de un sistema de monitorización remota que pueda solventar la problemática planteada. En dicha figura se muestra la existencia del sistema centralizado de monitorización que recogerá la información adecuada desde otros sistemas de la red.

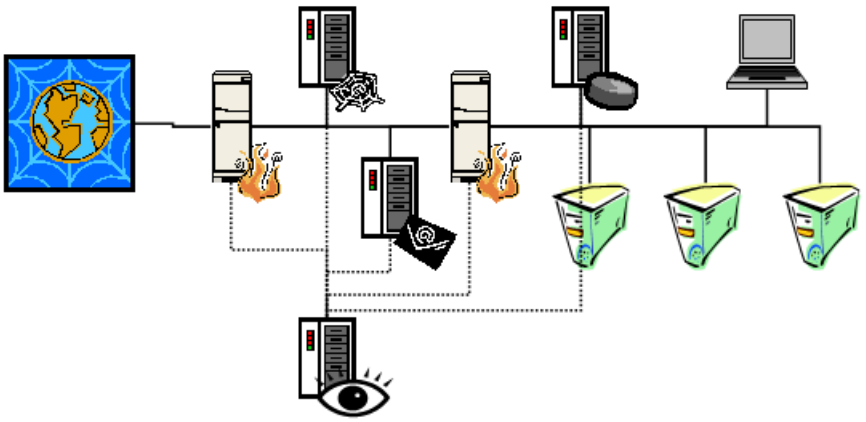


Figura 2.2: *DMZ* intermedia con sistema de monitorización

Las distintas funcionalidades que desarrollará nuestra solución podrían identificarse sobre esta red como se muestra en la Figura 2.3.

De esta forma:

1. El sistema centralizado de monitorización ejecutará el sistema **Debian Sarge**, presentando al Administrador de Red una interfaz del sistema (GUI) que le permitirá interactuar con el sistema de monitorización. Este interfaz permitirá a su usuario cambiar elementos de configuración ya existentes o introducir elementos nuevos en la configuración.

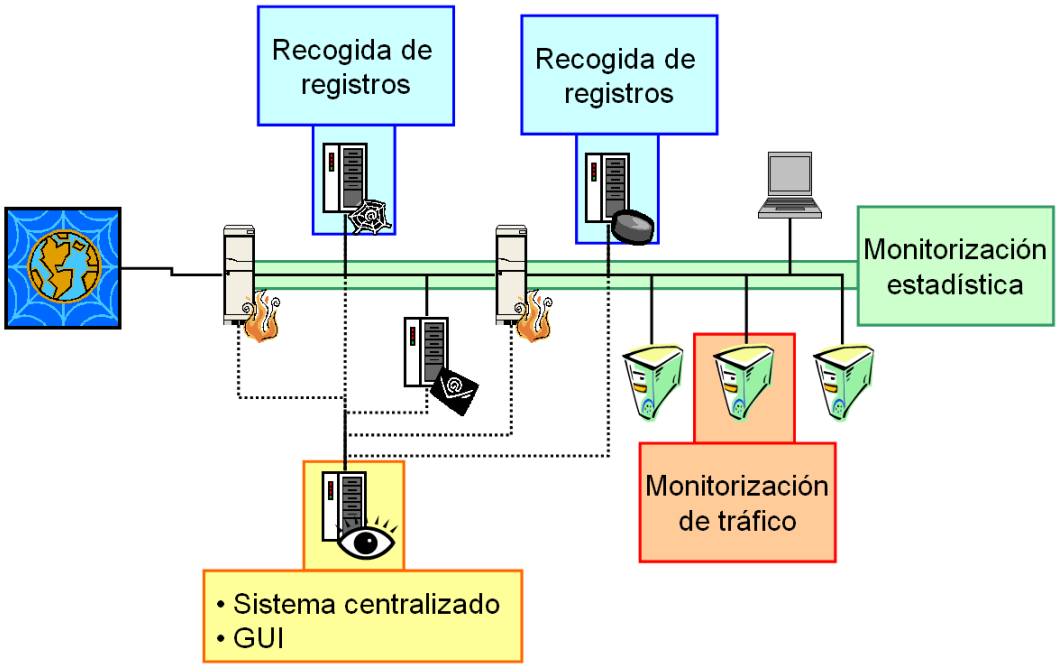


Figura 2.3: Funcionalidades y su identificación con la red de ejemplo

2. Se recogerá la información estadística del uso de la red a partir de la información que circule por la propia red. Esta información estadistica será generada de forma remota en sistemas de la red, minimizando la alteración de la red para su instalación y puesta en funcionamiento.

3. Se podrá efectuar una monitorización de tráfico de una forma selectiva, tanto en el tiempo como sobre la información en sí. Esta monitorización de tráfico también se efectuará de una forma remota en sistemas de la red, por los mismos motivos que se han indicado en el punto anterior.

4. Se recogerán los informes y registros de los sistemas de la red, tales como servidores o *firewalls*. Se buscará minimizar la complejidad de la instalación y puesta en funcionamiento de esta capacidad en los sistemas de la red.

5. El transporte de la información de monitorización se realizará atravesando las redes que sean necesarias, ya sea la propia red de datos que se monitoriza o redes independientes o exclusivas para la monitorización.

6. Se realizará un procesamiento y análisis de la información de monitorización (cualquiera de las tres citadas anteriormente) en el sistema centralizado, evitando el consumo de recursos de los sistemas de la red dedicados ya a otras labores.

A partir de este ejemplo simple puede abstraerse la implantación de las funcionalidades deseadas de nuestro sistema de monitorización en una red de

topología arbitraria.

Durante el resto de este documento iremos estudiando en mayor detalle cada una de las funcionalidades presentadas para el desarrollo de la solución de este Proyecto en una serie de capítulos autocontenidos. Primero, en el capítulo 4, hablaremos sobre la recogida de información estadística del uso de la red. En segundo lugar veremos en el capítulo 5 las posibilidades de recogida de información completa de tráfico desde un sistema remoto. En el capítulo 6 aspectos sobre la recogida de registros y de informes de los sistemas de nuestra red. Tras ello, en el capítulo 7 veremos algunas formas de procesar la información obtenida en nuestro sistema centralizado de monitorización. En el capítulo 8 trataremos algunos aspectos acerca del desarrollo de la interfaz de configuración.

2.3.1. Justificación

Pasamos en este punto a hacer una justificación más extensa de las funcionalidades que se desarrollarán en este Proyecto de Fin de Carrera.

1. Selección del sistema operativo **Debian Sarge**: para solventar los requisitos de utilizar un sistema basado en *software* libre se podía haber escogido otra distribución del sistema operativo GNU/Linux. Distribuciones como **Red Hat** o **Fedora** son comunes en sistemas servidores y bien podría haberse escogido una de ellas para el desarrollo de este Proyecto. En cambio, en nuestra opinión, la distribución **Debian Sarge** ofrece las siguientes ventajas:

   - El proyecto Debian persigue unos objetivos de seguridad y estabilidad en sus aplicaciones superiores a las del resto de las distribuciones, haciéndola una distribución más robusta que las demás.
   - Debian organiza su *software* en un sistema de paquetes muy adecuado para la instalación cómoda y limpia. Además ofrece una cantidad de programas y utilidades bajo licencias de *software* libre enorme que facilitará encontrar el *software* adecuado para nuestras necesidades.
   - El gran impulso que la distribución **Ubuntu** está teniendo dentro de la comunidad de GNU/Linux impulsa indirectamente a la distribución Debian, al basarse Ubuntu en esta última. La grán penetración del proyecto Ubuntu facilita la introducción de los usuarios al mundo Debian, y está planteando a muchos administradores el cambio a este sistema operativo cada vez más manejable y a la vez más seguro que el utilizado en sus servidores.

2. Selección del enfoque de sistema de monitorización remoto: hasta el momento hemos justificado el uso de sistemas de monitorización remota para:

   - Reducir gasto de recursos en sistemas de la red.
   - Minimizar tanto el tiempo como la complejidad de la instalación del sistema en un entorno «en producción».
   - Permitir la escalabilidad del sistema, dotándolo de mayor flexibilidad.

Si bien esto es cierto, queremos ahora hacer la siguiente reflexión acerca de esta *monitorización remota*:

A día de hoy, el sistema tradicional de monitorización empleado consiste en un sistema instalado en la red (en el lugar adecuado para ello) que captura y forma la información de monitorización. Ese sistema realizará las labores tanto de captura como de procesado de la información y se accederá a esta información de monitorización accediendo al sistema en sí, y este acceso podrá ser un acceso remoto[1].

En cambio, la monitorización remota dispone de un funcionamiento diferente desde la base: la información se capturará en la red (igualmente en el lugar adecuado) pero es enviada a un sistema que será el que realice el procesado. Hay una separación de las labores de captura y procesado. Por decirlo de algún modo, el sistema de monitorización *accede de forma remota* a la información que captura, e igualmente a este sistema de monitorización se podrá acceder de forma remota.

Esto, a nuestro parecer, ofrece las ventajas ya citadas de reducción de gasto de recursos, complejidad de instalación y flexibilidad del sistema. Es fácil entender esto si nos replanteamos en este punto la pregunta que introdujimos en la introducción de este documento:

> Si administramos una red grande de una empresa corporativa con distintas subredes para distintos departamentos cada una de ellos con sus propios *firewalls*, distintas *DMZs* para distintos accesos a Internet, ¿cómo monitorizar las distintas subredes? ¿Cómo analizar las necesidades de tráfico y ancho de banda de los departamentos? ¿Cómo comprobar el estado o las alarmas de los distintos *firewalls* y sistemas alojados en distintos segmentos de la red? ¿Cómo centralizar esa información? ¿Cómo hacerlo de forma eficiente?

Traduciendo esta pregunta en términos de número de sistemas necesarios para realizar la labor de monitorización, recursos económicos, coste de instalación y configuración, etc., del sistema clásico no remoto de monitorización están claras las ventajas del sistema de monitorización remota que se propone en este Proyecto frente a ese enfoque más clásico, y por ello hemos elegido el enfoque de la monitorización remota para este Proyecto.

3. Monitorización estadística del uso de la red: tener una buena contabilidad del uso de una red es una gran herramienta para la auditoría de la misma. Mediante sistemas de supervisión del tráfico de la red, podemos hacer un estudio detallado de los servicios más utilizados así como detectar los malos usos que se hagan de la misma.

   Esto obliga a mantener un sistema permanente de recogida de la información que atraviese nuestra red, y como ya hemos dicho nuestro sistema se basará en la monitorización remota. Se dispondrá de algún mecanismo que capturará la información de la red y la hará llegar al sistema centralizado de monitorización.

[1] Con este acceso remoto queremos referirnos a que no necesitará hacerse dicho acceso *in situ*, sino que podría hacerse a traves de una red.

Un aspecto a tener en cuenta es el lugar donde deberá ubicarse ese mecanismo capturador de la información. Se ha dicho que se monitoriza la red pero, obviamente, la red en sí no podrá realizar la captura sino que deberá hacerse en alguno de los sistemas de la red. Esto obliga a buscar los mejores sitios posibles para establecer dichos mecanismos, y estos sitios por norma general serán los interfaces de las distintas redes (como el interfaz entre nuestra red de área local e Internet) ya que son lugares donde el traspaso de información será notable. Esta norma general siempre podrá verse alterada en función de las necesidades concretas de la red que se desee monitorizar si se da el caso de tener unas necesidades más específicas.

4. Monitorización del tráfico de la red: el sistema de monitorización estadística anterior estará limitado por la tecnología utilizada, como se verá en el capítulo 4, "*Monitorización estadística de tráfico*". De esta problemática surgirá la necesidad de suplir las carencias que acusará la monitorización estadística.

   La monitorización *total* del tráfico o *retransmisión* del mismo no será un sistema capaz de sustituir a la monitorización estadística a pesar de suplir sus carencias. El hecho de retransmitir totalmente el tráfico que circula por la red hacia el sistema de monitorización podría, según la topología de la red, someterla a un sobreesfuerzo que no pueda soportar, causando más problemas por la congestión producida que los solucionados. Por ello, la monitorización estadística y la retransmisión deberán coexistir en el sistema de monitorización remota, y esta segunda solo funcionará de manera selectiva tanto en el tiempo como con la información que se desea retransmitir (por el ya mencionado problema de la congestión).

   A esto hay que añadir que, al igual que ocurre con la monitorización estadística, hará falta algún tipo de mecanismo que nos permita realizar la retransmisión. Dicho mecanismo debe de estar emplazado en el lugar adecuado para funcionar correctamente, y esas ubicaciones serán por norma general o los interfaces entre distintas redes o algún servidor concreto del que dispongamos en nuestra red.

5. Monitorización remota de *logs*: mientras que los mecanismos anteriores de monitorización estadística y retransmisión del tráfico se centraban en la propia red, la monitorización remota de *logs* o recogida de registros se centra en los sistemas de la red.

   El objetivo de este mecanismo es obtener los registros de incidencias, informes de estado, alarmas, etc., de los distintos sistemas que compongan nuestra red: los distintos servidores, los *firewalls*... De esta forma podremos tener en detalle cuál es el estado de la red, y de esta forma se podrán detectar de forma anticipada los posibles malfuncionamientos erráticos de los sistemas.

6. Interfaz del sistema (GUI): para facilitar el proceso de configuración de las aplicaciones seleccionadas que solventarán las funcionalidades anteriores se propone el desarrollo de una interfaz del sistema. Dicha interfaz estará ideada para la configuración y control de las aplicaciones que se ejecuten del lado del sistema de monitorización. Si bien podría haberse

planteado el control sobre alguna de las aplicaciones que se manejarán en los sistemas remotos esto añadiría una complejidad alta al desarrollo de la interfaz, y tal desarrollo ha sido descartado en principio para el desarrollo de este Proyecto. Se podrá consultar con mucha más profundidad el concepto de esta interfaz de configuración en el capítulo 8, "*Interfaz de configuración*".

Capítulo 3

Planificación y costes

3.1. Planificación del Proyecto

Mediante la planificación pretendemos estimar, en medida de lo posible, los recursos de tiempo y esfuerzo necesarios para el desarrollo de este proyecto.

Para realizar dicha estimación, dividiremos el Proyecto en una serie de fases y estimaremos el tiempo necesario para la realización de cada una de esas fases. Tras ello se compararán los datos estimados (que llamaremos *Estimación Inicial*) con los datos reales obtenidos una vez finalizado el desarrollo de este Proyecto (datos que llamaremos *Estimación Final*). Así compararemos ambos valores y obtendremos una medida de la desviación de la estimación realizada, que llamaremos *Error Relativo a la estimación* o *RE*.

Presentamos estos tres valores (*Estimación Inicial*, *Estimación Final* y *RE*) en el Cuadro 3.1 en el que la unidad de medida será el *día*, donde este *día* se corresponderá con la jornada laboral de tres horas desarrollada por una única persona. Esta unidad de medida pretende ser una ponderación media del esfuerzo semanal desarrollado para este Proyecto.

3.2. Estimación de costes

Una vez realizado el proyecto conocemos el número aproximado de líneas de código (*LDC*) sin contar el código HTML generado con aplicaciones auxiliares, el código XML generado a partir de aplicaciones intermedias ni el código reutilizado a partir del PFC *Anubix: Servidor de seguridad perimetral*. La cifra en cuestión es de, aproximadamente, 6800 *líneas de código*. Conocida esta cifra, podemos aplicar algún método de medición de costes, y hemos utilizado el modelo COCOMO (*Constructive Cost Model*) para la estimación de los mismos así como se hizo en el PFC *Anubix: Servidor de seguridad perimetral*.

El método COCOMO nos permitirá hacer una estimación del coste del Proyecto en base del tamaño del mismo expresado en *miles de líneas de código* o *KLDC*. Este método es ampliamente utilizado en la actualidad en la industria

Fase	**Estimación Inicial**	**Estimación Final**	**RE**
Búsqueda de documentación	20 días	30 días	33.3 %
Análisis de precedentes	10 días	12 días	16.6 %
Estudio de viabilidad	2 días	5 días	60 %
Análisis de requisitos	10 días	15 días	33.3 %
Estudio de estado del arte	20 días	32 días	37.5 %
Selección y análisis de tecnologías	20 días	18 días	-11.1 %
Estudio e implementación de la interfaz	90 días	140 días	35.7 %
Pruebas	7 días	7 días	0 %
Presentación	7 días	7 días	0 %
Total	186 días	266 días	30.0 %

Cuadro 3.1: Planificación temporal del Proyecto

del *software*.

Utilizando el modelo COCOMO básico para un proyecto de tipo orgánico (clasificación adecuada para este Proyecto dada su extensión), la ecuación del Esfuerzo E en COCOMO es la siguiente:

$$E = a \cdot KLDC^b \,[\text{personas} \times \text{meses}] \tag{3.1}$$

La estimación del Tiempo de desarrollo T sigue la siguiente expresión:

$$T = c \cdot E^d \,[\text{meses}] \tag{3.2}$$

En estas expresiones, los parámetros a, b, c y d son unos coeficientes dados por el modelo COCOMO:

Tipo de proyecto	a	b	c	d
Orgánico	2.4	1.05	2.5	0.38
Semiacoplado	3	1.12	2.5	0.35
Empotrado	3.6	1.2	2.5	0.32

Cuadro 3.2: Coeficientes del modelo

Para nuestro caso de 6800 *LDC*, o 6.8 *KLDC*, aplicamos la siguiente expresión para obtener el Esfuerzo:

$$E = a \cdot KLDC^b = 2{,}4 \times 6{,}8^{1{,}05} = 17{,}96 \,[\text{personas} \times \text{meses}] \tag{3.3}$$

Y para el Tiempo de desarrollo:

$$T = c \cdot E^d = 2{,}5 \times 17{,}96^{0{,}38} = 7{,}49 \,[\text{meses}] \tag{3.4}$$

El número de personas ideal, N, para desarrollar el proyecto en el tiempo estimado se calcularía como sigue:

$$N = \frac{E}{T} = \frac{17{,}96}{7{,}49} = 2{,}39\,[\text{personas}] \tag{3.5}$$

Como se indica en el resultado de la expresión anterior, según el método COCOMO harían falta dos personas para realizar el desarrollo del Proyecto durante un período de siete meses y medio, con dedicación exclusiva.

Capítulo 4

Monitorización estadística de tráfico

Índice del capítulo

4.1. Introducción a *NetFlow*

NetFlow es un protocolo de comunicaciones ideado por *Cisco Systems*. En primer lugar fue concebido como un protocolo para que distintos encaminadores y conmutadores de la red pudiesen intercambiar rápidamente información de su estado, notificando a los demás conmutadores si estaban en estado de congestión y, teniendo la información del estado de los demás conmutadores, poder decidir realizar el encaminamiento del tráfico por otro segmento de la red.

NetFlow evoluciona y comienza a ser utilizado no sólo como una herramienta para la comunicación entre los encaminadores sino que también se comienza a plantear su uso como un sistema de recogida de estadísticas sobre tráfico [2]. Con *NetFlow*, es posible la recolección de información del estado de los encaminadores y máquinas de la red, sin necesidad de pasar luego a ejercer una acción de control sobre los dispositivos.

En este capítulo del documento nos vamos a centrar sobre el uso de *NetFlow* para la recogida de la información del estado de la red en sistemas remotos y

su envío al sistema donde se desea procesar la información. En primer lugar haremos una breve visión sobre algunas sondas disponibles y sus principios de funcionamiento. Tras ello hablaremos sobre la *sonda* que hemos utilizado, que será la utilidad que nos genera la información. Estudiaremos las posibilidades de generación de información que ofrece la sonda, su configuración y haremos un comentario referente a sus posibilidades de seguridad. En tercer lugar trataremos sobre el *recolector* que hemos elegido, que es la utilidad que recoge o recopila la información de las sondas. El cuarto punto versará sobre algunas cuestiones a tener en cuenta acerca de la seguridad de la información de *NetFlow*. Por último, en el apéndice A, "*Monitorización estadística de tráfico*", veremos, entre otras, cuestiones referentes a instalaciones y configuraciones de las distintas utilidades.

4.2. Sonda de *NetFlow*: *fprobe-ng*

4.2.1. Sondas de *NetFlow*

En el punto 4.1 hemos mencionado ya el concepto de *sonda*. Lo que aquí denominamos *sonda* es un programa que se ejecuta en un sistema remoto (sistema que tiene acceso físico al tráfico que deseamos monitorizar), elabora la información estadística y la envía a un segundo sistema que denominaremos *recolector* (sistema que será nuestro sistema de monitorización). Para identificar con claridad la ubicación que tendrían las distintas sondas y el recolector en la ilustración de red que hemos estado siguiendo mostramos la Figura 4.1.

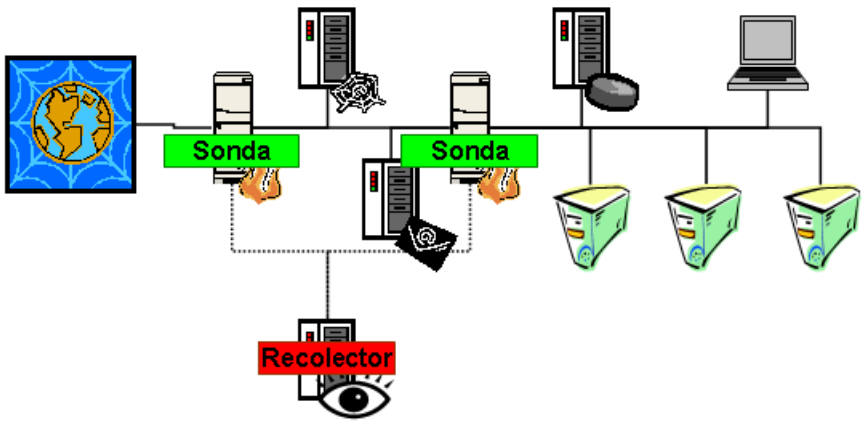


Figura 4.1: Ubicación de las sondas y el recolector

Para los sistemas GNU/Linux existen distintas sondas que permiten la recogida de la información en los sistemas remotos. Algunas de ellas son:

- *softflowd*: una sonda que escuchará en modo promiscuo en un interfaz del sistema en el que se encuentre instalada. Soporta IPv6 y es capaz de leer archivos *pcap* grabados por *tcpdump* y sistemas compatibles. No se

proporciona como *paquete Debian*. Para más información acerca de esta sonda véase [6].

- *pcNetFlow*: sonda que actualmente se encuentra sin desarrollo. Utiliza el principio de funcionamiento de la anterior: capturar los datos al funcionar en modo promiscuo en un interfaz, elaborando la información y transmitiéndola vía *NetFlow* hacia un recolector de destino. Para más información acerca de esta sonda véase [7].

- *fprobe*: una sonda de funcionamiento similar a la anterior. Exportará la información capturada en formato *NetFlow* v5. Actualmente no se encuentra muy actualizada a pesar de que se proporciona como *paquete Debian*. Para más información acerca de esta sonda véase [8].

- *fprobe-ng*: se presenta como una sonda alternativa para reemplazar a la anterior. Basada en *libpcap* también actúa como un *sniffer* en un interfaz para recopilar la información del tráfico que lo atraviesa. Soporta *NetFlow* versiones 1, 5 y 7, tiene más opciones de configuración que su predecesora. Se encuentra disponible como *paquete Debian*. Para más información acerca de esta sonda véase [9].

- *fprobe-ulog*: un proyecto paralelo al anterior que utiliza un método distinto para la captura de la información. En vez de basarse en *libpcap* utilizará el núcleo de *netfilter*, el cortafuegos del sistema, para identificar con cadenas de *iptables* el tráfico que queremos analizar por lo que requiere de una configuración adicional y compatibilidad del sistema con el módulo *ULOG* del *kernel*, resultando en una utilización bastante más compleja. También se ofrece como *paquete Debian*. Para más información acerca de esta sonda véanse [9, 11, 12].

De la lista anterior de sondas nosotros hemos optado por *fprobe-ng* dado la facilidad de su uso al basarse en *libpcap*, sus posibilidades de configuración y al ofrecerse como *paquete Debian*.

4.2.2. Introducción al paquete *fprobe-ng*

Nosotros hemos trabajado sobre una distribución **Debian Sarge** que actualmente se encuentra en estado estable, y nos hemos centrado la sonda ***fprobe-ng***, que está disponible en el momento de desarrollo de este Proyecto como *paquete Debian* en su versión 1.1-2.

fprobe-ng funcionará en un interfaz capturando en modo promiscuo los datos que atraviesen dicho interfaz y formando con los datos capturados la información que luego será enviada mediante el protocolo *NetFlow* hacia el recolector. Mostramos la ficha sobre el paquete ***fprobe-ng*** en el Cuadro 4.1.

En el apéndice A.2, "*Instalación y configuración de la sonda*", se verán en detalle los aspectos de instalación y configuración de esta aplicación.

Paquete	***fprobe-ng***
Autor	Slava Astashonok < sla@0n.ru >
Versión	1.1
Mantenedor	Radu Spineanu < radu@timisoara.roedu.net >
Versión del *paquete Debian*	1.1-2
Lenguaje	C
Web	http://fprobe.sourceforge.net

Cuadro 4.1: Ficha de *fprobe-ng*

4.2.3. La sonda y el protocolo *NetFlow*

Es importante hacer una reflexión sobre el comportamiento de la sonda con respecto al protocolo *NetFlow*.

La sonda *fprobe-ng* es, funcionalmente, un *sniffer* que captura en modo promiscuo la información que atraviese la interfaz donde la propia sonda esté escuchando. La sonda, en vez de representar la información capturada de una manera gráfica o almacenarla en un archivo, mandará la información a un recolector a través del protocolo *NetFlow* usando las capacidades que este ofrece.

NetFlow, al ser ideado en principio como un protocolo de intercambio de información de encaminadores que posteriormente pasó a ser utilizado para la recogida de información estadística, está limitado en la cantidad de información que porta en su interior. *NetFlow* está orientado a conocer los volúmenes de tráfico y las cargas de los interfaces o segmentos de red, y esa es la información que *fprobe-ng* mandará hacia el recolector al usarlo. Podremos ver el formato de los mensajes de *NetFlow* en el apéndice A.1, "*NetFlow versión 7*", de este documento.

Por tanto, dado lo limitado de la información estadística que se transporta en el protocolo no es posible que en el recolector se haga un análisis a nivel de aplicación de los datos capturados por la sonda en el sistema remoto, y esta es una acción que tampoco realizará la sonda. Como se verá en el apéndice A.1 del presente documento, la información que recoge la sonda y que puede enviar sobre *NetFlow* es información propia de los niveles de Red y Transporte, no del nivel de Aplicación, por eso hablamos de una *monitorización estadística* del tráfico.

Por último, el protocolo *NetFlow* en su definición esta pensado para transportar informaciones de encaminamiento que la sonda no utilizará en ningún caso. Como hemos dicho la sonda sólo es un *sniffer* y por tanto no atenderá los campos de *NetFlow* relativos a `nexthop`, `src_as`, etc, como se explicará en el apéndice A.1 de este documento.

4.3. Recolector de *NetFlow*: *flow-tools*

4.3.1. Recolectores de *NetFlow*

Como hemos adelantado en la sección 4.2.1, "*Sondas de NetFlow*", el *recolector* de *NetFlow* es el programa que recibirá la información generada en un sistema remoto por la *sonda* y procederá a su tratamiento.

Al igual que existen diversas sondas (véase el punto 4.2.1 de este documento) existen también distintos recolectores de *NetFlow*. Algunos de ellos son:

- *cflowd*: un recolector desarrollado para recoger y analizar información de *NetFlow*. Permite al usuario almacenar la información capturada y ofrece distintas formas de visualizarla. Actualmente no se prosigue su desarrollo y sus creadores recomiendan utilizar *flow-tools* como sustituto. Para más detalle acerca de este recolector véase [13].
- *flow-tools*: es un conjunto de herramientas para el tratamiento de la información de *NetFlow*. Consta de un recolector que almacena la información comprimida en el disco duro del sistema, algunas herramientas para el tratamiento de la información, retransmisores de la información, exportadores a bases de datos y alguna otra utilidad más. Es un conjunto de utilidades configurable y con bastantes opciones, y se encuentra disponible como *paquete Debian*. Para más detalle acerca de este recolector véase [14].
- *pmacct*: un paquete de utilidades reciente y actualmente en versión no definitiva. Dispone de unas capacidades similares a las de la herramienta anterior, además de herramientas para facilitar la exportación a bases de datos *RRD*. Para más detalle acerca de este recolector véase [23].

De la lista de herramientas anteriores nosotros hemos elegido *flow-tools*, por su combinación de utilidades, sencillez, estado estable y amplia utilización, capacidades de configuración y encontrarse disponible como *paquete Debian*.

4.3.2. Introducción al paquete *flow-tools*

Nosotros, al trabajar sobre **Debian Sarge** nos hemos centrado en las utilidades ofrecidas dentro del paquete ***flow-tools*** en su versión 0.67-8, entre las que se encuentran una serie de herramientas para el estudio de la información estadística, su almacenamiento, y por supuesto su captura con la ayuda del recolector ***flow-capture***.

Mostramos la ficha del paquete ***flow-tools*** en el Cuadro 4.2.

Dentro de las muchas utilidades que contiene el paquete ***flow-tools*** trataremos ahora sólo la utilidad *flow-capture*. *flow-capture* atenderá la llegada de paquetes del protocolo *NetFlow* en nuestro sistema de monitorización y procederá a su almacenaje de manera estática y en un formato propio comprimido en el disco duro del sistema de monitorización.

Paquete	***flow-tools***
Autor	Mark Fullmer < maf@splintered.net >
Versión	0.67
Mantenedor	Anibal Monsalve Salazar < anibal@debian.org >
Versión del *paquete Debian*	0.67-8
Lenguaje	C
Web	http://www.splintered.net/sw/flow-tools

Cuadro 4.2: Ficha de *flow-tools*

En el apéndice A.3, "*Instalación y configuración del recolector*", se verán en detalle los aspectos de instalación y configuración de esta utilidad.

4.3.3. El recolector y el protocolo *NetFlow*

Al contrario que la sonda comentada en la sección 4.2.3, "*La sonda y el protocolo NetFlow*", el recolector de *NetFlow* aquí usado, *flow-capture*, sí es capaz de hacer un uso pleno de la información transportada por el protocolo.

Con esto queremos decir que aunque la sonda no introduzca determinados tipos de información en el protocolo, el recolector sí va a leer dicha información del protocolo, por lo que tendremos que filtrar con posterioridad a la captura la información que queremos que luego se procese.

El recolector recogerá la información de las sondas de la red y las almacenará de forma estática en el disco duro del sistema donde se ejecute, y podrá crear estructura de directorios que posteriormente permitirán el acceso a la información de forma cómoda y ordenada. Además, el almacenamiento de la información de *NetFlow* de forma comprimida permitirá a este recolector almacenar mayor cantidad de información que los otros recolectores vistos en el punto 4.3.1.

4.4. Consideraciones de seguridad para *NetFlow*

Hemos tratado de encontrar durante el estudio de las utilidades antes mecionadas una solución en lo que a la seguridad del transporte de la información de *NetFlow* se refiere.

Es importante garantizar la seguridad de la información de monitorización en su viaje por la red desde el sistema remoto en el que la sonda captura la información hasta el sistema de monitorización en el que el recolector recibe dicha información. *NetFlow* por sí mismo no da una seguridad a los datos, no están cifrados, y estos son capturables e interpretables por cualquier *sniffer* que esté capturando información en la red como por ejemplo *ethereal*.

Por ello hemos tratado de encontrar una solución a este transporte inseguro de la información de monitorización a través de la red, donde debemos hacer

las siguientes consideraciones:

1. *NetFlow* se transporta sobre UDP, por lo que las soluciones basadas en túneles SSL como la aplicación *stunnel* no son utilizables.
2. No existen *paquetes Debian* de utilidades para el transporte seguro de UDP, por lo que su instalación en un elevado número de sistemas puede resultar una complicación además de un gran gasto de tiempo.
3. Las herramientas existentes para el transporte seguro de UDP sobre una conexión SSL no se encuentran todavía en un estado de desarrollo finalizado. Una herramienta de estas características es *Zebedee* [24].

Por tanto, se recomienda que el transporte de la información de monitorización se produzca en una red distinta, independiente a la red de datos e inaccesible, como se muestra en la Figura 4.2. De esta manera, las máquinas en las que escuchen las sondas *fprobe-ng* deberán estar conectadas con los recolectores de información de la red mediante interfaces de red separados y sobre una estructura de red independiente a la cual no se deberá poder acceder desde la red de datos.

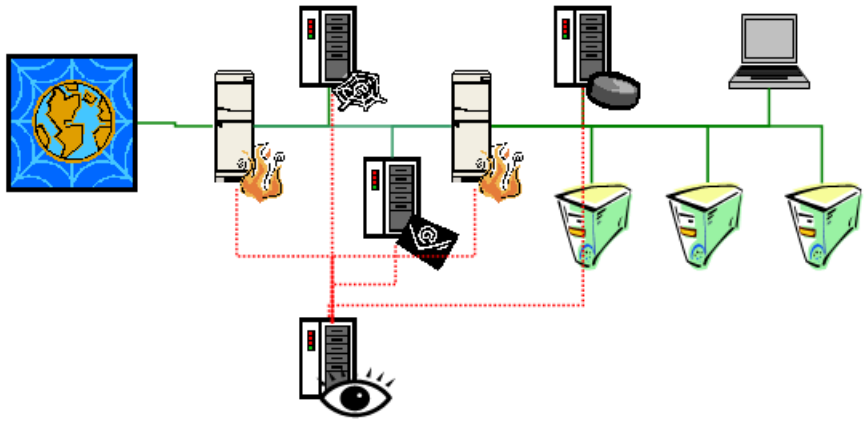


Figura 4.2: Separación de la red de datos de la red de datos de gestión

Capítulo 5

Monitorización de tráfico

Índice del capítulo

5.1. La retransmisión del tráfico

Como se ha indicado en la sección 4.2.3, "*La sonda y el protocolo NetFlow*", el uso de sistemas de monitorización remota estadística de tráfico, como el uso del protocolo *NetFlow*, llevan al problema de no disponer al completo de la información que está siendo transportada en otro segmento de la red. Como se ha explicado a lo largo de la sección 4.2 de este documento, con *NetFlow* podremos configurar una sonda para que intercepte el tráfico comportándose como un *sniffer* en modo promiscuo y se enviará información de los niveles de Red y Transporte hacia un recolector debidamente configurado. Ahora bien, ¿y si necesitamos poder monitorizar al 100 % el contenido del tráfico que circula por nuestro sistema remoto?

Si queremos poder estudiar con todo nivel de detalle, sin suprimir los campos de datos de los paquetes, el tráfico que circula por un sistema remoto y así poder acceder a la información del nivel de Aplicación tenemos que hacer de alguna forma que dicho tráfico llegue completamente a nuestro sistema remoto. Una vez que disponemos de manera local de dicho tráfico podemos pasar a su estudio utilizando las distintas herramientas que existen, como por ejemplo *ethereal*, tal como se muestra en la Figura 5.1.

Podríamos pensar en soluciones *hardware*, como *switchs* o concentradores comerciales que disponen de un puerto especial por el que replican o retransmiten una copia de todo el tráfico que atraviesa sus puertos. Dicho puerto se podría conectar a nuestro sistema de monitorización para que analizase el tráfico, pero esta forma de solventar el problema tiene sus inconvenientes:

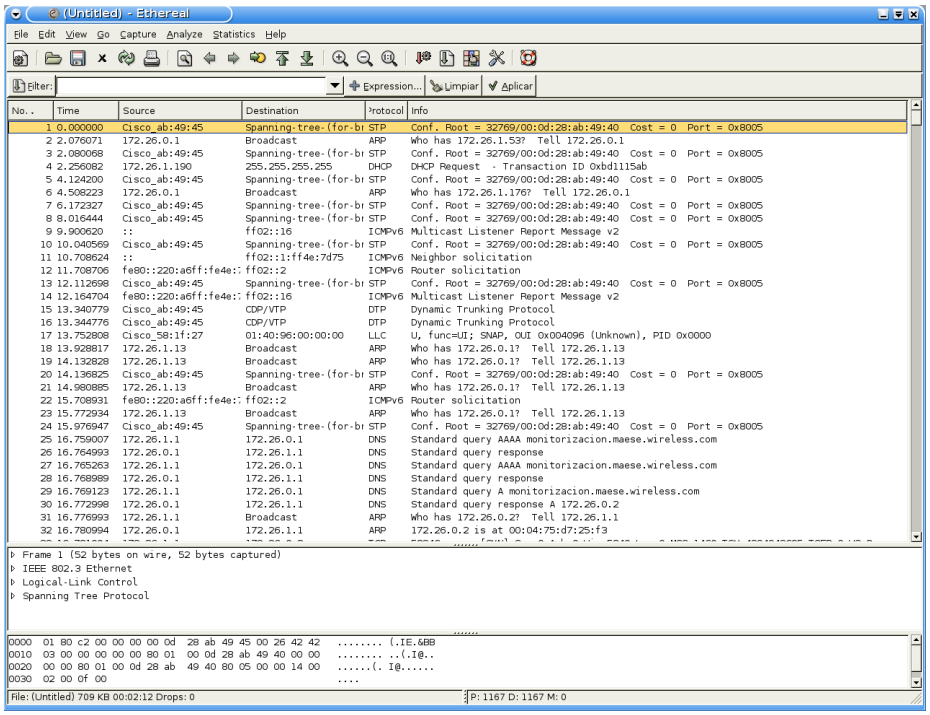


Figura 5.1: Tráfico retransmitido capturado con *ethereal*

- Impone la necesidad de comprar equipos *hardware* adicionales, con el consecuente gasto económico.

- En redes muy cargadas, un ordenador normal que podría bastarnos para nuestro sistema de monitorización podría no ser suficientemente potente para realizar adecuadamente la captura de paquetes, pudiendo producirse pérdidas indeseables. Desear evitar estar pérdidas podría llevar a cambiar dicho equipo, aumentando el coste de la operación.

Por tanto surge la necesidad de buscar formas más elegantes y selectivas, y sobre todo que sean soluciones *software*, para provocar la retransmisión del tráfico.

En este capítulo vamos a ver una posibilidad que existe para obtener una copia del tráfico que circula por un sistema remoto. Primero hablaremos de las capacidades que *iptables* ofrece como retransmisor de paquetes, capacidad que puede ser usada para realizar una copia del tráfico y enviarla hacia otro sistema. En segundo lugar haremos una breve reseña a cómo se pueden efectuar operaciones sencillas y de manera cómoda sobre una máquina remota. Por último, en el apéndice B, "*Monitorización de tráfico*", veremos los detalles de configuraciones e instalaciones de las distintas herramientas comentadas en este capítulo.

5.2. Retransmisión del tráfico: *iptables*

5.2.1. Introducción a *iptables*

iptables es el conocido cortafuegos utilizado en los sistemas Linux a partir de su versión de núcleo 2.4 [25, 26].

iptables consiste en una serie de tablas por las que deben transcurrir los paquetes que queremos enviar, recibir, o que han de atravesar nuestro sistema (paquetes de los cuales no somos ni origen ni destino). En dichas tablas podemos introducir reglas particulares que permiten configurar el cortafuegos, como impedir las comunicaciones con un *host* concreto o prohibir el tráfico de un servicio específico que atraviesa el sistema [27].

iptables dispone de unas capacidades y posibilidades sorprendentes si se profundiza en su funcionamiento. *iptables* puede marcar paquetes de forma interna al núcleo de la máquina para que a los paquetes marcados se les aplique un procesado posterior, puede modificar campos de los paquetes como el campo *ToS* o incluso las direcciones IP de origen o destino de una comunicación, o es capaz de realizar funciones de supervisión de ciertos protocolos. Estas funcionalidades son posibles gracias no sólo al núcleo del propio *iptables* sino a su interacción con el núcleo del sistema (o ***kernel*** de aquí en adelante) y a la modularidad del propio *iptables* que ha permitido a múltiples desarrolladores añadir funcionalidades a este cortafuegos.

iptables suele necesitar de su correspondiente módulo en el *kernel* del sistema para poder funcionar correctamente, por lo que la tarea de utilizar funcionalidades algo especiales de *iptables* suele estar asociada a la tarea de parchear el *kernel* del sistema para que la nueva funcionalidad sea reconocida. Para facilitar esta labor existe la herramienta ***patch-o-matic-ng*** [31] que es un *script* que realiza de forma cómoda la labor de introducir en el código fuente del *kernel* y de *iptables* el código de los módulos de los que queremos hacer uso. Detalles sobre esta herramienta y su uso se verán en el apéndice B, "*Monitorización de tráfico*".

5.2.2. *iptables* y el módulo *ROUTE*

La capacidad de retransmisión de paquetes de *iptables* la hemos conseguido gracias al módulo ***ROUTE*** de *iptables* [29, 30]. Dicho módulo, en la versión que nosotros hemos necesitado usar, está disponible en el *paquete Debian* de la versión actualmente en estado *testing*, concretamente la versión 1.3.3, de *iptables* mientras que la versión en estado *stable* se encuentra en la 1.2.2.

Además de tener que utilizar una versión *testing* de *iptables* es necesario un *kernel* compilado adecuadamente con soporte para el módulo *ROUTE*, y ninguna de las dos versiones del *kernel* que se distribuyen para la versión *stable* de **Debian Sarge** en el momento de realización de este Proyecto, las versiones 2.4.27 y 2.6.8, lo soportan.

Por ello, tras probar a compilar tanto el *kernel* como *iptables* de las versiones *stable* o *testing* para hacer funcionar el módulo *ROUTE* y no haber sido posible, hemos utilizado las versiones de *kernel*, *iptables* y *patch-o-matic-ng* que se detallan en los Cuadros 5.1, 5.2 y 5.3, respectivamente.

Linux Kernel	
Versión	2.6.15.1
Lenguaje	C
Web	http://www.kernel.org

Cuadro 5.1: Ficha de *Linux Kernel*

Utilidad	***iptables***
Autor	Netfilter Team
Versión	1.3.5
Lenguaje	C
Web	http://www.netfilter.org

Cuadro 5.2: Ficha de *iptables*

Utilidad	***patch-o-matic-ng***
Autor	Netfilter Team
Versión	20060206 (snapshot)
Lenguaje	C
Web	http://www.netfilter.org

Cuadro 5.3: Ficha de *patch-o-matic-ng*

Con el módulo *ROUTE*, *iptables* es capaz de realizar funciones de encaminamiento poco habituales, como por ejemplo desviar el tráfico proveniente de un primer sistema destinado a un segundo sistema dirigiéndolo hacia una tercera máquina distinta. En versiones más modernas de este módulo este desvío de tráfico se puede realizar no sólo desviando el tráfico sino dejando el tráfico original sin modificaciones y permitiendo que siga su flujo habitual dentro de las tablas de *iptables*, y realizando una copia de los paquetes se puede encaminar hacia un tercer sistema, que será nuestro sistema de monitorización. A esto es a lo que denominamos *retransmisión* del tráfico.

Gracias a esta funcionalidad, la retransmisión del tráfico es posible con *iptables*. Para ello será necesario la instalación de un núcleo compatible con el módulo *ROUTE* y la instalación de un paquete de *iptables* configurado a tal efecto en cada una de las máquinas donde queramos proceder a la retransmisión de los paquetes. Este proceso de instalación se detalla en el apéndice B.1, "*Instalación del módulo ROUTE*", aunque sería más útil y conveniente disponer de *paquetes Debian* de estas versiones compatibles con *ROUTE* [37, 38, 39]. En los

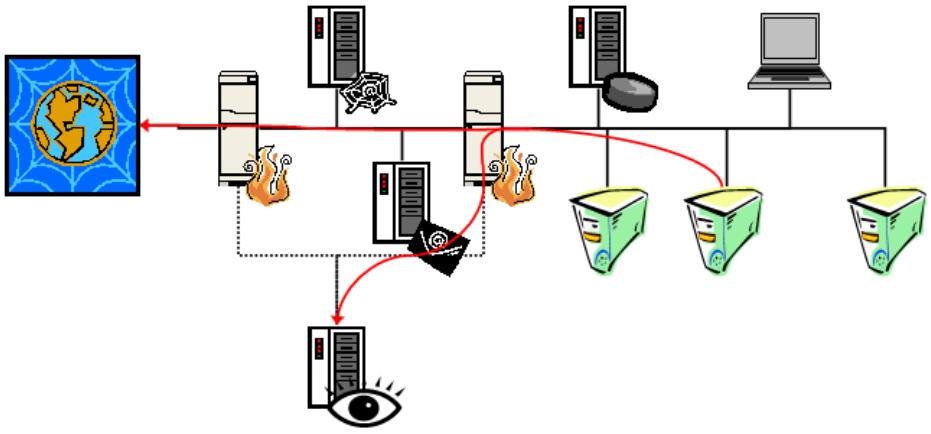


Figura 5.2: Esquema de funcionamiento de la retransmisión de tráfico

apéndices B.1.3.1, "*Creando un paquete Debian del kernel*", y B.1.4.1, "*Creando un paquete Debian de iptables*", se detalla cómo fabricar esos *paquetes Debian*. Tras la instalación bastará proceder a la activación de la retransmisión que consistirá en activar la regla de *iptables* adecuada, algo que veremos con más detalle en el apéndice B.1.5, "*Activación del módulo ROUTE*". Cabe aquí destacar que dado que se basa en el uso de *iptables* cada regla de retransmisión es altamente configurable, de manera que podemos ser muy selectivos con el tráfico que deseemos retransmitir: exclusivamente un puerto, exclusivamente un *host* concreto, exclusivamente un interfaz, etc.

5.3. Ejecución remota de operaciones

Como se ha comentado en la sección 5.2.2, para efectuar la retransmisión de paquetes en las máquinas remotas hacia la máquina que alberga el sistema de monitorización es necesario introducir reglas adicionales en las tablas de *iptables* de los sistemas remotos. Esto implica, evidentemente, una acción de control de forma remota desde la máquina de monitorización sobre las máquinas que retransmitirán el tráfico.

Una manera simple y segura de efectuar esta acción de control es basarse en el uso del paquete ***ssh*** [40, 41] que viene incluido por defecto dentro de nuestro sistema **Debian Sarge**. *ssh* es un reemplazo seguro para los viejos sistemas de ejecución remota de operaciones y de consola remota como *rlogin*, *rcp* o *telnet*. *ssh* se utilizará desde el sistema de monitorización conectándose a servidores activos en las máquinas que habrán de retransmitir el tráfico donde se estará ejecutando el demonio del servidor *ssh*, ***sshd***, también incluido por defecto en la distribución **Debian Sarge**. Tras ello, sobre una conexión encriptada y segura, podremos pasar a la ejecución de comandos en el sistema remoto para activar la retransmisión de paquetes.

Paquete	***ssh***
Autor	Tatu Ylonen < ylo@cs.hut.fi >
Versión	3.8.1
Mantenedor	Matthew Vernon < matthew@debian.org >
Versión del *paquete Debian*	3.8.1p1-8.sarge.4
Lenguaje	C
Web	http://www.openssh.net

Cuadro 5.4: Ficha de *ssh*

En el apéndice B.2, "*Instalación y configuración del paquete ssh*", veremos los detalles de instalación y configuración de ***ssh*** para la creación de pares de llaves cliente-servidor [42] para realizar de forma cómoda y segura la autenticación entre el sistema de monitorización y los sistemas remotos.

Capítulo 6

Recogida de registros

Índice del capítulo

6.1. Introducción a los registros del sistema

Tradicionalmente, *syslog* ha permitido a los administradores obtener información de registros en sus sistemas de manera uniforme para toda la red, realizando la tarea de guardar, analizar y procesar los archivos de registro fácilmente.

Los registros habitualmente se usan para comprobar la salud del sistema. Muchos administradores ni siquiera se molestan en mirar los registros a menos que se encuentren con un problema en el sistema. Hoy día es también una cuestión de mejorar la fiabilidad, es decir, usar el sistema como una alerta temprana antes de que los problemas vayan a peor. También es ahora la integridad de los mensajes del sistema más importante que nunca, ya que permiten a los administradores levantar defensas basadas en datos reales.

Estas y otras necesidades han cambiado en los últimos años, y el servicio *syslogd* [43, 44] tradicional de BSD no puede cubrir todos estos aspectos, por lo que han aparecido una serie de nuevas utilidades que pretenden rellenar esas grietas y ofrecer funcionalidades completas que sustituyan al viejo *syslogd*.

Durante este capítulo vamos a estudiar algunas de las posibilidades que existen de cara a la recogida de registros en una red con múltiples generadores de registros. En primer lugar haremos una breve mención a las soluciones propuestas para el tratamiento de los eventos del sistema y del manejo de los mensajes. En segundo lugar haremos un breve acercamiento al paquete de *syslog-ng*, la utilidad que hemos seleccionado para el manejo de los mensajes de los registros.

Tras ello, en el tercer punto haremos unas menciones acerca de los aspectos de seguridad que nos permite utilizar *syslog-ng*. Por último, en el apéndice C, "*Recogida de registros*", veremos en detalle los aspectos relativos a la instalación y configuración de *syslog-ng* para un mejor manejo de los mensajes del sistema.

6.2. Estado del arte

En la actualidad la mayoría de las distribuciones de Linux basan su sistema de recogida de eventos del sistema y de manejo de los registros en la utilidad *syslogd* de Berkeley. Este programa, ideado en un principio para unas necesidades menores que las actuales, tiene una serie de carencias que nos llevan a querer sustituirlo por gestores de eventos más eficaces. Algunas de esas carencias son:

- Una clara falta de seguridad, al manejar mensajes en texto claro y sin efectuar ningún tipo de autenticación entre los sistemas.
- Transferencia de los mensajes sobre UDP, protocolo no orientado a conexión, lo que conlleva a los habituales problemas referentes a la pérdida de mensajes.
- Pérdida de la autoría del mensaje, al no propagar entre los distintos servidores el nombre del autor sino del retransmisor del mensaje.
- Una capacidad de configuración algo escasa que en la actualidad no es capaz de satisfacer esquemas complejos de múltiples máquinas y utilidades.

Existen una variedad de herramientas que pretenden subsanar algunas o todas estas carencias, de las que destacamos las siguientes:

- *syslog-sign* es una *RFC* propuesta por el *IETF* [46] para solucionar alguna carencia de *syslogd*. Busca ser compatible con *syslogd* y apenas añade una autenticación con llaves cliente-servidor, pero sigue basándose en UDP y no realiza encriptación de los mensajes.
- *syslog-reliable* es también una *RFC* [47] más compleja y que pretende solucionar el transporte (pasando a realizarse sobre TCP), la autenticación y la encriptación. Existe un intento de plasmar *syslog-reliable* en una herramienta por parte del *San Diego Supercomputer Center*, llamada *SDSC Secure Syslog* [48].
- *syslog-ng*, o *syslog de nueva generación* [49], es una aplicación bastante extendida como sustituta de *syslogd*. Ofrece una configuración mucho más potente y capaz que su precursor añadiendo poca dificultad. Ofrece transporte sobre TCP y sobre UDP para asegurar la compatibilidad con la versión anterior, y aunque no ofrece autenticación o encriptación (algo en lo que trabajan los autores) su configuración con otras aplicaciones externas para solucionar esta carencia es muy sencilla, como se verá en la sección 6.4, "*Consideraciones de seguridad: stunnel y openssl*", de este documento.

Nosotros hemos elegido *syslog-ng* por su combinación de sencillez y versatilidad, además de suponer una solución a las mayores carencias del *syslogd* habitualmente incluido en las distribuciones Linux. Además, *syslog-ng* se proporciona como *paquete Debian*, por lo que facilita aun más su uso.

6.3. Recogida de registros: *syslog-ng*

6.3.1. Introducción al paquete *syslog-ng*

syslog-ng es una utilidad que sustituye al tradicional *syslogd* de los sistemas Linux, el demonio de recogida de eventos del sistema ideado por Berkeley. Con esta sustitución, *syslog-ng* se encargará del manejo de los eventos del sistema de una manera más ordenada y efectiva, ya que actualmente estos eventos del sistema son de muy distinto origen y causan que los registros del sistema se llenen de mensajes poco importantes o *ruido* que puede llegar a ocultarnos los mensajes importantes.

syslog-ng incluye un sistema de filtrado de los mensajes no solo por la *facilidad* que los crea sino por su contenido e importancia y pretende mantener esa información sin alteraciones cuando los mensajes de eventos del sistema viajan por la red, algo que es especialmente importante cuando una máquina almacena en sus registros las notificaciones tanto propias como de otra variedad de máquinas remotas y es necesario mantener la identidad del propietario de cada uno de esos mensajes.

Hemos decidido usar *syslog-ng* por sus posibilidades de clasificación y tratamiento de los mensajes de eventos del sistema de varias máquinas a la vez, permitiéndonos recoger todos los registros en estructuras ordenadas que luego puedan ser cómodamente consultadas. Además de esto, como se verá en la sección 6.4, "*Consideraciones de seguridad: stunnel y openssl*", *syslog-ng* nos permite establecer mayores cotas que *syslogd* de seguridad en la entrega de los mensajes.

syslog-ng se proporciona como *paquete Debian* en su versión 1.6.5-2.2. En su instalación, se eliminarán las utilidades *klogd* y *sysklogd* que por defecto vienen en la distribución **Debian Sarge**. Mostramos la ficha del paquete ***syslog-ng*** en el Cuadro 6.1.

syslog-ng deberá ser instalado en las distintas máquinas, pudiendo actuar el mismo programa como emisor y receptor de los mensajes. Así, en la Figura 6.1 queda reflejada cómo se instalaría en la red.

6.4. Consideraciones de seguridad: *stunnel* y *openssl*

Se vio en la sección 4.4, "*Consideraciones de seguridad para NetFlow*", de este documento la carencia que presentaba el protocolo *NetFlow* al realizar sus comunicaciones debido a su transporte sobre UDP. *syslogd* también basaba sus

Paquete	***syslog-ng***
Autor	Balázs Scheidler < bazsi@balabit.hu >
Versión	1.6.5
Mantenedor	Attila Szalay < sasa@debian.org >
Versión del *paquete Debian*	1.6.5-2.2
Lenguaje	C
Web	http://www.balabit.hu/products/syslog-ng/

Cuadro 6.1: Ficha de *syslog-ng*

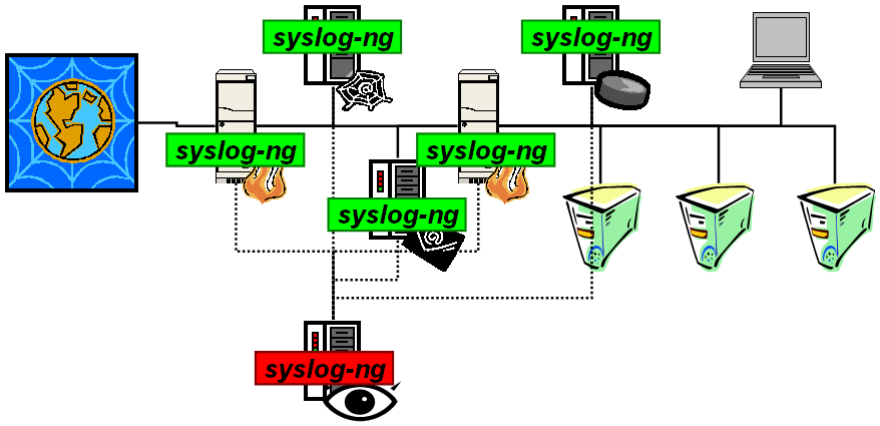


Figura 6.1: Ubicación de *syslog-ng*

capacidades de transferencia de mensajes a máquinas remotas sobre UDP, por lo que también aparece esta vulnerabilidad.

syslog-ng en cambio nos permite la posibilidad de transmitir su información por la red tanto sobre UDP como sobre TCP, y como se verá en el apéndice C.2, "*Consideraciones de seguridad: transporte mediante stunnel*", hemos elegido esta segunda opción precisamente pensando en poder asegurar sus comunicaciones.

stunnel es una utilidad que permite establecer túneles cifrados SSL para las comunicaciones entre máquinas remotas de una red [53]. Es una utilidad extendida que permite asegurar las comunicaciones de forma sencilla, siendo un uso frecuente del mismo utilizarlo para construir servidores de correo seguros sobre versiones no seguras de una manera rápida y simple.

Detallamos en el Cuadro 6.2 la ficha del paquete ***stunnel***.

Para utilizar las capacidades que *stunnel* nos ofrece deberemos configurar adecuadamente a *syslog-ng* para que lo utilice, con lo que nos quedará un esquema de funcionamiento como el indicado en la Figura 6.2.

Paquete	***stunnel***
Autor	Michal Trojnara < Michal.Trojnara@mirt.net >
Versión	3.26
Mantenedor	Julien Lemoine < speedblue@debian.org >
Versión del *paquete Debian*	3.26-3
Lenguaje	C
Web	http://www.stunnel.org/

Cuadro 6.2: Ficha de *stunnel*

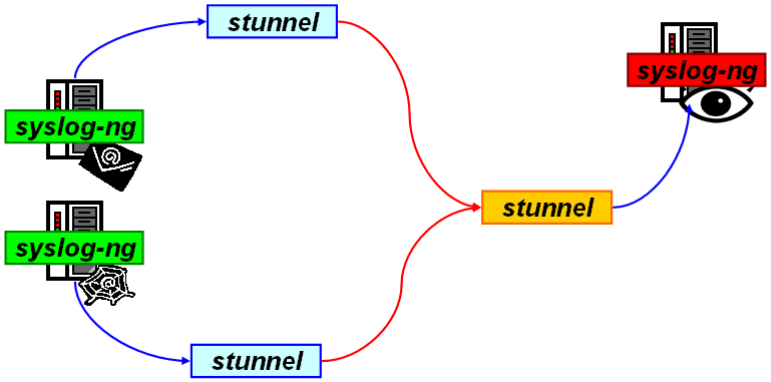


Figura 6.2: Uso de *stunnel*

De esta forma, cada utilidad *syslog-ng* se comunicará de forma no cifrada con una utilidad *stunnel* de forma local ya que ambas utilidades estarán en el mismo sistema, mientras que la comunicación entre las dos utilidades *stunnel* sí se realizará de manera cifrada atravesando nuestra red.

Por último, utilizaremos también el paquete ***openssl*** junto a *stunnel* para crear certificados [54, 55] que nos permitan realizar una autenticación entre las distintas máquinas remotas y el servidor que recoge los mensajes, con lo que añadiremos autenticación y acreditación a los mensajes transmitidos por las aplicaciones *syslog-ng*.

Mostramos la ficha del paquete ***openssl*** en el Cuadro 6.3.

Gracias a los certificados creados con *openssl*, haremos que *stunnel* utilice pares de certificados cliente-servidor de manera que el servidor de *stunnel* exija a los clientes el uso de un certificado para poder establecer el túnel con el servidor y se asegurará de que el certificado es válido, realizando así la autenticación de los clientes.

Paquete	***openssl***
Versión	0.9.7e
Mantenedor	Christoph Martin < christoph.martin@uni-mainz.de >
Versión del *paquete Debian*	0.9.7e-3
Lenguaje	C
Web	http://www.openssl.org

Cuadro 6.3: Ficha de *openssl*

Capítulo 7

Tratamiento de la información

Índice del capítulo

7.1. Tratamiento de la información de *NetFlow*

La información que *NetFlow* ofrece no es demasiado alta, tan sólo es información de los niveles de Red y Transporte, pero lo que sí ocurre es que podemos disponer de un gran volumen de esa información si disponemos de un elevado número de sondas y recolectores en nuestra red, o bien si se trata una red con una carga alta, con una alto volumen de tráfico. En cualquiera de estos casos las sondas de *NetFlow* que estén capturando información de manera continuada estarán mandando continuos flujos de información hacia nuestros recolectores, y se hace necesario realizar una presentación de la información ya capturada de una forma eficiente y cómoda.

Dentro de esta sección nos acercaremos a tres posibles maneras de visualizar la información recogida por nuestros recolectores. El primer punto versa sobre una visualización simple en una consola, con todas las limitaciones que ello conlleva. En el segundo punto de la sección veremos cómo es posible reenviar la información ya capturada a otros recolectores con mejores capacidades de representación gráfica de la información, como por ejemplo *ntop*. Como tercera posibilidad veremos las capacidades de exportación de la información recogida a bases de datos *SQL* y *RRD*. Finalmente, en el apéndice D.1, "*Tratamiento de la información de NetFlow*", veremos los detalles más técnicos de estos aspectos.

7.1.1. Presentación mediante consola

El primero de los métodos de presentación que detalladamente se verán en el apéndice D.1, "*Tratamiento de la información de NetFlow*", será la presentación en modo consola o terminal, en función de donde apliquemos el método.

La presentación en modo consola está limitada en las posibilidades que ofrece, y citamos las siguientes ventajas e inconvenientes:

- La primera limitación es la caducidad de la información, puesto que en los sistemas de consola más comunes la información más reciente desplaza a la anterior, quedando esta imposible de acceder en períodos cortos de tiempo. Si esto se añade a la posibilidad de usar el sistema sobre una red cargada de tráfico, las sondas nos inundarán con tanta información que poco podremos ver sobre la consola que estemos usando.
- La segunda limitación, que más bien es una molestia, es que en las máquinas donde estemos almacenando la información podrá no haber modo gráfico alguno en buena parte de las ocasiones y tendremos como única opción de consola a los terminales *tty* tradicionales, muy limitados en resolución y en número de filas y de columnas. En estos terminales las líneas de información que querremos comprobar, si son extensas, estarán truncadas y resultarán en un galimatías de direcciones IP y números difíciles de entender.
- Por otro lado, la presentación en modo consola tiene la ventaja de tener una latencia nula entre la recogida de la información y su presentación si se configura adecuadamente su funcionamiento, haciendo que cada paquete de *NetFlow* que se reciba en el recolector sea enviado al terminal.

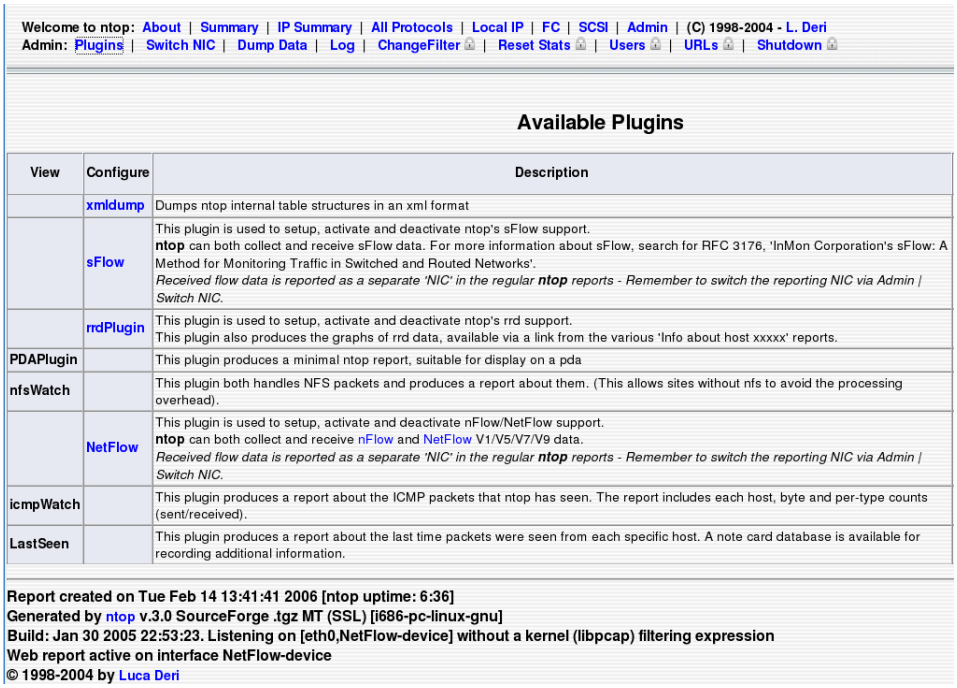
Por tanto, la presentación mediante consola es poco recomendable y está dirigida más a los administradores que necesiten hacer alguna comprobación en el sistema cuando los demás sistemas fallan o verificar que las sondas funcionan durante un proceso de instalación.

En el apéndice D.1.1, "*Presentación mediante consola*", veremos con detalle como utilizar las utilidades incluidas dentro del paquete ***flow-tools*** con estos propósitos [59].

7.1.2. Exportando a *ntop*

ntop es una conocida utilidad de monitorización de redes. Se trata de un *sniffer* en modo promiscuo que captura la información que pasa por la interfaz donde se encuentra instalado. Presenta la información recogida mediante formato *Web*, en una serie de páginas donde podemos consultar los datos en función de los *hosts*, de los protocolos, etc.

ntop dispone de una serie de *plug-ins* que le permiten funcionalidades adicionales más especializadas. Entre ellas se encuentra una *plug-in* llamada *NetFlow* que activa un recolector de *NetFlow* dentro del propio *ntop* [60], como se muestra en la Figura 7.1.



Welcome to ntop: About | Summary | IP Summary | All Protocols | Local IP | FC | SCSI | Admin | (C) 1998-2004 - L. Deri
Admin: Plugins | Switch NIC | Dump Data | Log | ChangeFilter | Reset Stats | Users | URLs | Shutdown

Available Plugins

View	Configure	Description
	xmldump	Dumps ntop internal table structures in an xml format
	sFlow	This plugin is used to setup, activate and deactivate ntop's sFlow support. **ntop** can both collect and receive sFlow data. For more information about sFlow, search for RFC 3176, 'InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks'. *Received flow data is reported as a separate 'NIC' in the regular **ntop** reports - Remember to switch the reporting NIC via Admin | Switch NIC.*
	rrdPlugin	This plugin is used to setup, activate and deactivate ntop's rrd support. This plugin also produces the graphs of rrd data, available via a link from the various 'Info about host xxxxx' reports.
PDAPlugin		This plugin produces a minimal ntop report, suitable for display on a pda
nfsWatch		This plugin both handles NFS packets and produces a report about them. (This allows sites without nfs to avoid the processing overhead).
	NetFlow	This plugin is used to setup, activate and deactivate nFlow/NetFlow support. **ntop** can both collect and receive nFlow and NetFlow V1/V5/V7/V9 data. *Received flow data is reported as a separate 'NIC' in the regular **ntop** reports - Remember to switch the reporting NIC via Admin | Switch NIC.*
icmpWatch		This plugin produces a report about the ICMP packets that ntop has seen. The report includes each host, byte and per-type counts (sent/received).
LastSeen		This plugin produces a report about the last time packets were seen from each specific host. A note card database is available for recording additional information.

Report created on Tue Feb 14 13:41:41 2006 [ntop uptime: 6:36]
Generated by ntop v.3.0 SourceForge .tgz MT (SSL) [i686-pc-linux-gnu]
Build: Jan 30 2005 22:53:23. Listening on [eth0,NetFlow-device] without a kernel (libpcap) filtering expression
Web report active on interface NetFlow-device
© 1998-2004 by Luca Deri

Figura 7.1: Plugin de *ntop* para activar el soporte de *NetFlow*

De esta forma, con la configuración adecuada, *ntop* puede estar funcionando en su manera habitual (como *sniffer*) y a la vez puede recoger información de una serie de sondas remotas instaladas por nuestra red. *ntop* permite diferenciar qué información proviene de la captura local y qué otra información proviene de las capturas de las sondas, por lo que se evitan los problemas de interpretaciones incorrectas.

Veremos con detalle la configuración que debemos dar al sistema para producir este efecto en el apéndice D.1.2, "*Exportando a ntop*".

7.1.3. Exportando a una base de datos

Como hemos dicho, la información de *NetFlow* puede ser no muy variada pero sí tendremos gran cantidad de registros sobre la misma. Por ello, es lógico pensar en la posibilidad o necesidad de exportar la información capturada a una base de datos.

Una posibilidad es la exportación a bases de datos *MySQL* que están ampliamente extendidas en la actualidad. Podremos elegir qué información de *NetFlow* queremos almacenar en la base de datos, y posteriormente podremos manejar la información ya almacenada en la base de datos de las múltiples maneras que se pueden hacer en la actualidad: con consultas a través de una interfaz *Web*, a través de otros programas, etc. Estudiaremos este aspecto en más detalle en el apéndice D.1.3, "*Exportando a una base de datos MySQL*", a través de la utilidad ***flow-export*** [22] incluída en el paquete de herramientas ***flow-tools***.

El problema sin embargo radica en la cantidad de información que se almacena en la base de datos, que en algunos momentos puede ser desbordante y dificultará su manejo por parte de cualquier servidor de bases de datos. Debemos pensar entonces cómo utilizar adecuadamente la base de datos. Un uso lógico es utilizarla para almacenamiento temporal de la información de gestión durante un período de seguridad, para su estudio en caso de necesidad. Esto nos permitirá acceder a la información estadística de tráfico ya almacenada si la situación actual nos impone la necesidad de estudiar el estado de la red durante un período pasado de tiempo. Un período habitual de salvaguarda de los datos es de tres días, lo que es un punto de compromiso entre el tamaño de la información almacenada y el período de tiempo capturado.

Una solución a este problema del almacenamiento y que hemos tratado en este documento es la exportación a bases de datos *Round Robin* o *RRD*, bases de datos circulares y de tamaño fijo que permiten abarcar cómodamente períodos largos de tiempo a costa de bajar la resolución de la información paulatinamente al incremento del período temporal abarcado, pero ofrecen buenas capacidades para el almacenamiento de extensos períodos de tiempo. Veremos estos aspectos con más detenimiento en el apéndice D.1.4, "*Exportando a bases de datos Round Robin*". Concretamente, veremos las capacidades de la utilidad ***FlowScan*** [58] y de algunas extensiones de la misma, ***CUFlow***, ***CUGrapher*** [66] y ***FlowMonitor*** [67] para el almacenamiento de la información de *NetFlow* y su manejo en dichas bases de datos *RRD*. No todas estas herramientas se presentan en *paquetes Debian*, y las que lo hacen ni siquiera están disponibles en el momento de realización de este Proyecto en la versión *stable*. Mostramos las fichas de los paquetes disponibles en versión *texting* en los Cuadros 7.1 y 7.2, y la información equivalente de *FlowMonitor* en el Cuadro 7.3.

Paquete	***flowscan***
Autor	Dave Plonka < plonka@doit.wisc.edu >
Versión	1.006
Mantenedor	Anibal Monsalve Salazar < anibal@debian.org >
Versión del *paquete Debian*	1.006-8
Lenguaje	Perl
Web	http://net.doit.wisc.edu/ plonka/FlowScan/

Cuadro 7.1: Ficha de *flowscan*

Paquete	***flowscan-cuflow*** y ***flowscan-cugrapher***
Autor	Johan Andersen < johan@columbia.edu > y Matt Selsky < selsky@columbia.edu >
Versión	1.7
Mantenedor	Russell Stuart < russell-debian@stuart.id.au >
Versión del *paquete Debian*	1.7-1
Lenguaje	Perl
Web	http://www.columbia.edu/acis/ networks/advanced/CUFlow/

Cuadro 7.2: Ficha de *flowscan-cuflow* y *flowscan-cugrapher*

Utilidad	***flowscan-flowmonitor***
Autor	Johan Andersen < johan@columbia.edu >
Versión	1.2
Lenguaje	Perl
Web	http://www.columbia.edu/acis/networks/ advanced/FlowMonitor/FlowMonitor.html

Cuadro 7.3: Ficha de *flowscan-flowmonitor*

7.2. Tratamiento de la información de los registros

7.2.1. Visualización en consola

Similar a lo que ocurre con la presentación de la información de *NetFlow* mediante consola, como se indicó en el punto 7.1.1, la presentación en modo consola es un sistema algo tosco y primitivo. Presentará unas ventajas e inconvenientes similares a las vistas con anterioridad:

- Con respecto a la limitación de la caducidad de la información, la diferencia con la presentación de la información de *NetFlow* radica en la velocidad con la que esta se genera, que será muy inferior a la de *NetFlow*. Esto hará que la visualización en modo consola pueda tener mejores resultados que la visualización que se obtuvo con *NetFlow*.

- En cuanto a la segunda limitación mencionada, acerca de la limitación en resolución y en número de filas y columnas de los terminales sin modo gráfico, el problema también se ve menos acusado dada la naturaleza del contenido de la información. Las líneas más legibles y cortas de los mensajes del sistema tendrán una mejor cabida en los terminales que la información de *NetFlow*.

- Igual que en el caso anterior, la presentación en modo consola tiene la ventaja de tener una latencia nula entre la recogida de los mensajes y su

presentación si se configura adecuadamente su funcionamiento, obteniendo en el terminal y «en tiempo real» los mensajes del sistema.

La representación de los registros en modo consola está ideada para que quede de manera permanente en un terminal, reflejando los informes de los registros para que un administrador los pueda ver al momento de su generación. Esto permite la más rápida actuación por parte de los administradores para solventar los problemas que se generen especialmente en las fases de instalación y configuración de las aplicaciones *syslog-ng* o cualquier otra utilidad que vaya a afectar a los registros del sistema.

7.2.2. Visualización mediante página *Web*

La visualización de los registros del sistema mediante página *Web* es muy similar a la presentación en modo consola, si bien ofrece una serie de ventajas y beneficios que la hacen más interesante:

- En primer lugar, si la información se ofrece mediante una página *Web* esta puede ser accedida desde múltiples lugares, desde todos aquellos desde los que se pueda acceder al servidor que la aloje.
- En segundo lugar, soluciona los problemas de visualización caduca y limitada en resolución de la consola, al ser dependiente ahora de la configuración del cliente.

Por otra parte plantea también unos nuevos problemas:

- Como contrapartida a la primera ventaja, será necesario establecer una política de seguridad en nuestro servidor que impida a usuarios no adecuados acceder a la información, dado que ahora será accesible desde todos los puntos desde los cuales se pueda acceder al servidor.
- Referente a la segunda, implica la necesidad de máquinas con entorno gráfico, lo cual hace pensar en la necesidad de más máquinas para proceder a la monitorización si no queremos mezclar máquinas con entorno gráfico con máquinas sin el mismo.

Para resolver este punto y el anterior (7.2.1) hemos utilizado la utilidad ***ccze*** [68]. *ccze* es un *colorizador* de registros como algunas opciones adicionales como son su generación de código *HTML*, por lo que además de permitir una visualización cómoda en consola es capaz de hacer una exportación idéntica a una página *Web*. ***ccze*** se distribuye también como *paquete Debian*, y mostramos su ficha en el Cuadro 7.4.

Paquete	***ccze***
Autor	Gergely Nagy < algernon@bonehunter.rulez.org >
Versión	0.2.1
Mantenedor	Gergely Nagy < algernon@bonehunter.rulez.org >
Versión del *paquete Debian*	0.2.1-1
Lenguaje	C
Web	http://bonehunter.rulez.org/CCZE.html

Cuadro 7.4: Ficha de *ccze*

Capítulo 8

Interfaz de configuración

Índice del capítulo

8.1. Introducción

Hasta este momento, a lo largo de este documento, hemos presentado una serie de tecnologías y utilidades que intentan subsanar las carencias motivadoras de este Proyecto ya planteadas en su introducción. Dentro de este Proyecto de Fin de Carrera hemos querido buscar una solución que intente, en medida de lo posible, facilitar al máximo la implantanción en entornos «en estado productivo» de las distintas tecnologías que se han estudiado. Gracias al uso de los *paquetes Debian* y la automatización de su instalación se facilita en gran parte la labor de la primera implantación de las utilidades seleccionadas en nuestro sistema.

Sin embargo, la complejidad de uso y sobre todo de configuración de algunas de las aplicaciones que se han ido presentando en este documento impiden un uso simple y automatizado y a la vez flexible y potente de dichas aplicaciones. Esto anula en cierto modo el interés que estas utilidades pueden tener para su implantación tanto en entornos que ya se encuentran en funcionamiento como en sistemas que comiencen a desarrollarse.

Para solucionar parte de esta complejidad se ha desarrollado dentro de este Proyecto una interfaz de configuración para el manejo de algunas de las utilidades estudiadas. De esta forma se añade al Proyecto una fase radicalmente

distinta de desarrollo de una aplicación a la vez que se persigue lograr un producto más sólido que facilite el manejo de las utilidades seleccionadas.

En este capítulo abordaremos algunos asuntos de importancia de cara al desarrollo de dicha interfaz. En primer lugar realizaremos un estudio de viabilidad de la solución propuesta. A continuación se planteará el análisis de los requisitos de la aplicación que se pretende desarrollar. En el tercer punto comentaremos aspectos de diseño de la interfaz. Finalmente, en el apéndice E, "*Interfaz de configuración*", se verán, entre otros, aspectos relativos a la instalación y configuración de dicha interfaz así como detalles de la implementación de la misma.

8.2. Estudio de viabilidad

En esta fase, puesto que se trata de la realización de un Proyecto de Fin de Carrera, es obvio que se pretende realizar desarrollos allá donde sea necesario, así como reutilizar productos estándares (tales como librerías y componentes de *software*) para facilitar la construcción y propiciar la finalización del mismo en un tiempo razonable. Se tendrá en cuenta por tanto la inclusión de productos estándares y reutilizables allá donde sea posible y cuya licencia de uso lo permitan.

En una vista general del Proyecto se observa que el objeto de trabajo está dividido en dos partes bien diferenciadas, las cuales son:

1. Estudio de problemática y planteamiento de posibles soluciones
2. Interfaz de configuración para un usuario humano

La primera parte se centra más en el estudio de componentes *software* estándares y en la configuración de los mismos sobre una plataforma de sistema operativo (en nuestro caso **Debian Sarge**), buscando su personalización para incluir únicamente lo necesario para cumplir los requisitos de este Proyecto (por motivos de seguridad, rendimiento y espacio) y automatización en la medida de lo posible tanto en el proceso de instalación (con objeto de facilitar su empleo como producto final) como en el funcionamiento autónomo. Mientras tanto, la segunda parte del Proyecto que ahora tratamos tiene un componente importante de desarrollo.

El objetivo de este Proyecto, como ya se ha comentado, es el estudio de una serie de herramientas y sistemas de monitorización que proporcionen al Administrador de Red información adecuada sobre el estado de la misma con la finalidad de solventar la problemática planteada en la introducción de este documento. Dentro del Proyecto, la interfaz de configuración tiene, por una parte, la misión de permitir a este Administrador la revisión del estado de funcionamiento y, por otra, la realización de tareas de configuración y ajuste de parámetros.

Con la interfaz se trata de generar un producto a medida (ya que se ha visto que no hay disponible en la actualidad una aplicación que realice la tarea requerida con el nivel de detalle y comodidad deseados, y más aún ajustándose a los componentes *software* elegidos). La interfaz no cubrirá todos los aspectos

tratados en este Proyecto, sino que se han escogido tan sólo algunas de las utilidades estudiadas. Los motivos que han llevado a la selección son:

1. Escoger utilidades de fácil integración en un entorno «en estado productivo». Con esto se busca evitar aplicaciones cuya instalación y configuración sean poco automatizables y muy particularizadas, lo que sería poco conveniente en sistemas y redes ya operativos y de cierta envergadura.

2. Seleccionar aplicaciones fácilmente escalables, de manera que un aumento en el tamaño de la red o en el sistema no oblige a un pesado proceso de reconfiguración tanto en el sistema que albergue la interfaz, en el nuevo sistema que se integra en la red o en cualquiera de los sistemas que ya lo integran.

3. Combinar utilidades que de manera conjunta ofrezcan una funcionalidad completa e interesante.

4. Evitar utilidades y funcionalidades de menor interés y que no propicien la finalización del Proyecto en un tiempo razonable.

Por estos motivos, la interfaz de configuración contempla a las siguientes utilidades:

1. Recolector de *NetFlow flow-capture*.

2. Analizador *FlowScan*, con su módulo *CUFlow*.

Para la implementación se contará con el apoyo de librerías de clases estándares existentes (extraídas de repositorios de código reutilizable publicados en Internet, puestos a disposición de la comunidad *Open-Source* por grupos de desarrollo voluntarios y sin ánimo de lucro (como suele ser común el los proyectos GPL)). Estas librerías son:

- *PEAR::Config*: facilita el procesamiento de ficheros de configuración de texto en formatos conocidos y estándares (tales como XML, INI, o el estándar genérico «tipo Apache»). Véase [72].

- *patForms*: librería de generación de formularios *Web*, y de ayuda al procesamiento de los datos introducidos a través de los mismos. Véase [73].

- *patTemplates*: librería para el manejo de plantillas *Web*, fácilmente acoplable con patForms. Véase [74].

- *patError*: gestión de errores en la aplicación, utilizada exhaustivamente por patForms. Véase [75].

Estas librerías (desarrolladas en el lenguaje de *scripts* PHP) facilitan enormemente la tarea de, por un lado leer la información de configuración a partir de los ficheros de texto de los programas antes mencionados (en este caso mediante *PEAR::Config*), y por otro generar el formulario *Web* (con *patForms*, y las librerías auxiliares) que presentará una representación gráfica del contenido de dichos ficheros ante el usuario, ofreciéndole la posibilidad de modificar su contenido.

En el sentido inverso de la comunicación, el usuario puede introducir los valores que desee y enviarlos de vuelta a la aplicación (a la capa de lógica de negocio), pudiendo ser leídos con facilidad (de nuevo con la ayuda de *patForms*) y transformarlos para introducirlos posteriormente en los ficheros de configuración necesarios, con la ayuda de *PEAR::Config*.

Por tanto, vemos que con el modelo de componentes descrito, que se adapta perfectamente a la arquitectura de sistema propuesta (tanto en cuestion de sistema operativo Linux como de arquitectura de ejecución de tres capas), se consigue realizar la implementación de un sistema funcional en un tiempo de desarrollo adecuado, y con la funcionalidad requerida al completo.

8.3. Análisis

En esta fase y a partir de la elaboración de un catálogo de los requisitos del sistema (tanto funcionales como no funcionales) se persigue obtener un modelo de la interfaz del sistema que describa su funcionamiento y estructura.

8.3.1. Catálogo de requisitos de la interfaz

8.3.1.1. Requisitos funcionales

Mediante reuniones del equipo de trabajo y el estudio de las necesidades a las que se pretende dar solución con este proyecto, se identifican (referentes únicamente a la interfaz de configuración) los siguientes requisitos funcionales:

RF1: El sistema debe presentar el estado actual de funcionamiento de cada uno de los componentes *software* que pretende controlar.

RF2: El sistema debe presentar al usuario la configuración actual con la que se encuentran funcionando cada uno de los Componentes *software*.

RF3: El sistema debe permitir al usuario la modificación de los parámetros de configuración que se le ofrezca de cada uno de los componentes *software*.

RF4: El sistema debe detectar errores en la introducción de los datos de configuración (valores incorrectos, datos incoherentes, etc.) y evitar introducirlos en la configuración de los componentes *software* (a fin de evitar un mal funcionamiento de los mismos). Así mismo, si se da esta situación, debe presentar un mensaje de error al usuario advirtiendo de este hecho.

RF5: El sistema permitirá al usuario el control sobre la ejecución de un componente *software* íntimamente relacionado con el módulo objeto de configuración. En concreto, debe permitir al usuario el inicio y parada inmediatos de un componente *software*, así como la configuración en el sistema operativo de su inicio o parada automáticos (es decir, del arranque del servicio).

8.3.1.2. Requisitos no funcionales

Mediante la revisión del entorno tecnológico sobre el que operará la aplicación, y las necesidades y restricciones técnicas con las que se cuenta, se ha

elaborado el siguiente catálogo de requisitos no funcionales:

Requisitos sobre la arquitectura y usabilidad:

RNF1: El sistema debe funcionar sobre una arquitectura *Web* de tres capas, y estar desarrollado al completo de manera compatible con el lenguaje PHP y el servidor *Web* sobre el que éste se ejecuta.

RNF2: El sistema presentará al usuario un menú o índice que le llevará a las «pantallas» de configuración de cada uno de los módulos independientes (que controlan a su vez los distintos componentes *software* específicos).

RNF3: Los módulos emplearán un mecanismo de comunicación entre sí que les permita compartir cierta información (principalmente datos de configuración) con el objetivo de que el usuario no tenga que introducir un mismo parámetro en múltiples ocasiones (en varias pantallas diferentes). De esta forma se evitan posibles errores por inconsistencia en las configuraciones de distintos componentes *software*.

Este último requisito, tras estudiar lo que supondrá para el desarrollo de la aplicación, nos lleva a determinar un nuevo requisito no funcional (que catalogaremos también como requisito sobre la arquitectura), que resulta de importancia en cuanto a cómo afectará a dicho método de desarrollo:

RNF4: Se empleará Orientación a Objetos en la implementación de los módulos, con objeto de facilitar la reutilización de código y el paso de mensajes entre módulos.

Ello se debe a que la orientación a objetos permitirá independizar con mayor facilidad unos módulos de otros (mediante la propiedad del encapsulamiento), unificará interfaces, y permitirá el paso de mensajes: así, cuando un módulo requiera que otro adquiera y procese cierta información, se podrá realizar mediante este mecanismo con cierta facilidad.

RNF5: Se empleará XML preferentemente como lenguaje de descripción de estructuras de datos estáticas configurables y/o parametrizables (por ejemplo: para la descripción de los interfaces de usuario, y en los ficheros propios de configuración).

8.3.2. Modelo del sistema

Se busca en este punto crear un modelo que represente el sistema desde un punto de vista no detallado, como corresponde al proceso de Análisis, si bien los requisitos, y en especial el RNF4, dan ya de por sí cierta información acerca de cómo se deberá construir el sistema.

Para representar el sistema se han identificado los elementos de los que constaría y, modelándolos mediante Clases, se muestra en la Figura 8.1 un diagrama de clases del sistema (o, más formalmente, «Modelo de objetos de Análisis»), que muestra a alto nivel las diferentes clases y las relaciones entre ellas. En este modelo se ha conservado la estructura del PFC de *Óliver López Yela* y *Jose Carlos Ramírez Pérez Anubix: Servidor de seguridad perimetral.*

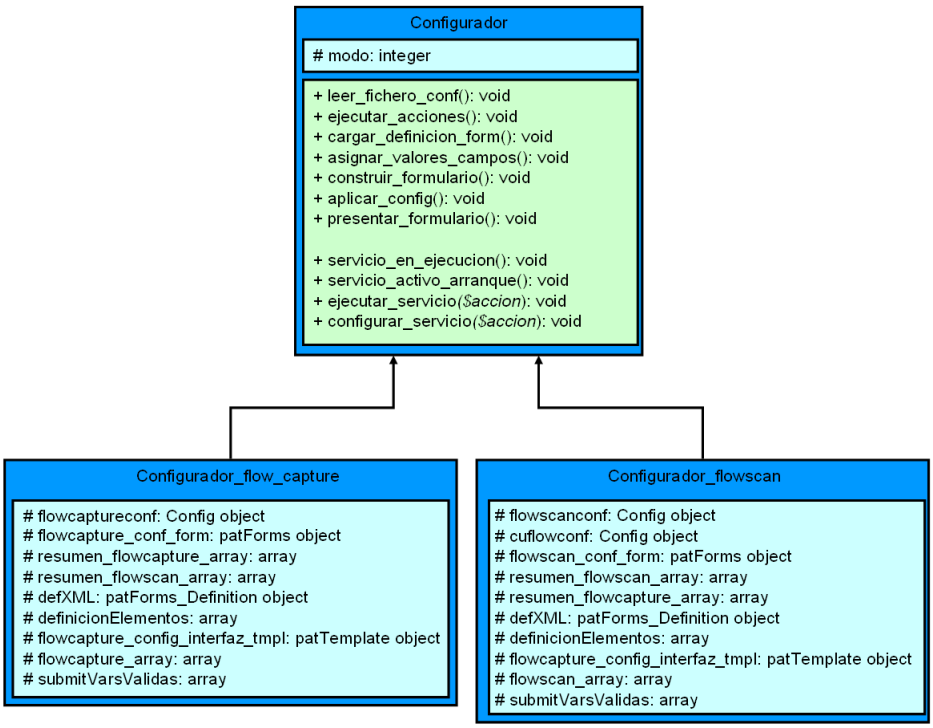


Figura 8.1: Diagrama de clases de la interfaz

8.3.2.1. Descripción del modelo

Como se puede ver en este diagrama, se tiene una clase principal **Configurador**, de la cual heredan una serie de clases hijas. Estas clases hijas tienen como misión particularizar la implementación para cada caso en concreto.

La clase **Configurador** es una abstracción del proceso de diálogo entre el usuario y el módulo que desea configurar (si bien, también podría entenderse como el módulo en sí mismo).

Las restantes clases hijas, que heredan de **Configurador**, tienen como misión implementar el funcionamiento de la clase para cada módulo en concreto, esto es: **Configurador_flow_capture** para el módulo del recolector *flow-capture*, **Configurador_flowscan** para el módulo del analizador *FlowScan*, etc... Esto permite, por ejemplo, en el futuro sustituir cualquiera de estas clases hijas por una implementación específica para otro componente de Software diferente (por ejemplo, otro recolector que no sea *flow-capture*), sin tener que modificar ni la implementación de los demás módulos, ni la interfaz que se emplea para dialogar con el módulo.

8.3.2.2. Justificación

Por los requisitos del sistema, como se ha visto anteriormente, podrá ser necesario que los diferentes módulos se comuniquen entre sí para proporcionar-

se mútuamente información de configuración. Esto implica que, una vez que se realicen cambios en la configuración de un módulo, los restantes deban leer estos cambios para «refrescar» su configuración. Esto se implementa mediante el paso de mensajes: el módulo con el que dialoga el usuario, y que realiza cambios en su configuración, comunica a los demás módulos que deben refrescar su configuración (para introducir cambios en los diferentes ficheros de configuración que se ven afectados).

Esta es la razón de ser del atributo *modo*, de la clase padre **Configurador**: cuando un módulo necesite refrescar su configuración, se instanciará en «modo Refrescar» (`MO_MODO_REFRESCAR`), en caso contrario, se instanciará en «modo Normal» (`MO_MODO_NORMAL`). Esto indicará al objeto internamente en qué «modo de funcionamiento» se encuentra, lo cual será de especial importancia en cuanto al procesamiento de la interfaz de usuario, como veremos posteriormente. Ello es debido a que un módulo que está en «modo Refrescar», no debe procesar información alguna referente a la interfaz de usuario (ya que en ese momento el usuario está dialogando con otro módulo). Este mecanismo evitará errores, y el desarrollador sabrá a «qué atenerse» en cada situación.

8.3.2.3. Descripción de atributos y métodos

Como se puede ver en el diagrama de la Figura 8.1, las clases hijas tienen en común que poseen dos tipos de atributos importantes: *config* y *form*. El significado de los mismos es el siguiente:

- *config* es la representación en memoria de la estructura y contenido del fichero de configuración.
- *form* es la representación de la estructura y contenido de los campos visibles en la interfaz de usuario.

Realmente se tratan de objetos (clases), que más tarde se verá cómo se han implementado (en este caso, utilizando librerías estándares disponibles). La idea es que un módulo tiene como principal función la transformación de los parámetros de los ficheros de configuración en campos visibles en el formulario y configurables por el usuario (y también el proceso inverso). Por tanto debe almacenar internamente tanto el conjunto de parámetros de configuración tal y como se almacenan en el correspondiente fichero, como el conjunto de campos que contiene la interfaz de usuario (con sus tipos, restricciones y formato de visualización concreto). De ahí la existencia de estos dos atributos.

La instanciación de cada uno de estos objetos, con la estructura de datos concreta que necesite cada módulo, será responsabilidad de dicha subclase.

En cuanto a los métodos, podemos agruparlos en:

1. Métodos de configuración
   - *leer_fichero_conf()* : lee el fichero (o ficheros) de configuración correspondientes al módulo, valida su sintaxis y semántica, e introduce su contenido en el atributo `config` del objeto.
   - *ejecutar_acciones()*: ejecuta las acciones inmediatas que precise el módulo durante su configuración (como arrancar o parar servicios) o las que el usuario solicite a través de la interfaz (acciones como: activar el servicio, establecerlo en el arranque, etc...)
   - *cargar_definicion_form()*: carga la definición del formulario (campos, tipos, apariencia) y genera la correspondiente estructura en memoria.
   - *asignar_valores_campos()*: rellena esta estructura en memoria con los valores leídos del fichero de configuración, realizando las transformaciones que sean necesarias.
   - *construir_formulario()*: crea el formulario gráfico que el usuario verá en pantalla, conteniendo toda la información anteriormente recopilada.
   - *aplicar_config()*: una vez que el usuario ha introducido los valores deseados, y envía estos al sistema, se capturan los mismos, se validan y se transforman para introducirlos de nuevo en los ficheros de configuración que sea necesario.
   - *presentar_formulario()*: se presenta finalmente el formulario al usuario, con los posibles mensajes de error fruto de las validaciones anteriores anexos al mismo.

   Como se puede ver, estos módulos se podrían ejecutar de una forma prácticamente secuencial en el flujo normal de ejecución de un módulo. Esto se ha diseñado así expresamente, con el objetivo de hacer el código más sencillo e inteligible, a la vez que fácilmente modificable y ampliable. Teniendo ésto en cuenta, la ejecución de dichos métodos siguen el flujo de datos normal de:

   Fichero de configuración → Validar y transformar → Cargar formulario → Rellenar formulario → Presentar formulario.

   Pero también es válido para este otro flujo «inverso»:

   Datos introducidos por el usuario → Validar y transformar → Cargar formulario → Rellenar formulario → Escribir configuración → Presentar resultados.

   Como se puede ver, lo único que varía es el origen (y el destino) de los datos, siendo gran parte del resto del proceso común a ambos flujos de datos. Ello es debido a que para poder escribir los datos, es necesario leerlos antes. Dicho de otro modo, el usuario introduce modificaciones en los datos que se leen de la configuración. Por tanto el flujo lectura-proceso-escritura se deberá seguir realizando en cada transacción de forma casi idéntica, ya sea sólo para mostrar datos, como para realizar modificaciones en los mismos. También es común en ambos flujos la presentación del

formulario al final. Es este proceso global lo que se implementa en los métodos enumerados anteriormente.

2. Métodos de control de servicios

   - *servicio_en_ejecucion()*: consulta si un servicio del sistema se encuentra actualmente en ejecución.
   - *servicio_activo_arranque()*: consulta si un servicio del sistema está configurado para arrancarse en el inicio.
   - *ejecutar_servicio(accion)*: realiza las acciones inmediatas necesarias para poner en marcha, o detener la ejecución, de un servicio del sistema.
   - *configurar_servicio(accion)*: realiza las acciones diferidas necesarias para configurar, en el arranque del sistema, la ejecución (o no) de un servicio.

   En cuanto a estos métodos, como es lógico, en cada módulo se implementarán para controlar el servicio del sistema que realiza la ejecución del componente *software* íntimamente relacionado con el módulo. Por ejemplo, en el caso del módulo del recolector *flow-capture*, estos métodos controlarán la ejecución del servicio *flow-capture* en el sistema.

8.4. Diseño

8.4.1. Consideraciones sobre la arquitectura del sistema

8.4.1.1. Definición de formularios en XML

Como se comentó en el Análisis de Requisitos, y en concreto en el requisito no funcional RNF5, se ha decidido emplear preferentemente el lenguaje XML para describir las estructuras de datos referentes a los interfaces de usuario (formularios de configuración).

En un compromiso entre mantener este principio y la flexibilidad que proporciona, así como no aumentar demasiado la carga del sistema en cuanto al procesamiento de los formularios y la transformación de los mismos se refiere, sin menospreciar la conveniencia de poder personalizar la apariencia de la aplicación de una forma simple, se ha optado por un esquema mixto que utiliza tanto XML como HTML para la descripción de los formularios (pantallas) de configuración del sistema, estando repartida la responsabilidad de cada lenguaje como a continuación se indica:

- HTML para las plantillas (es decir, la apariencia)
- XML para los elementos estructurados (independientes de la apariencia)

Es decir, que por un lado se emplearán estas plantillas HTML (esto es, código HTML en el que aparecen ciertas «marcas» o etiquetas donde se insertará el contenido dinámico posteriormente) que representan enteramente la apariencia de los formularios (y al tener una estructura interna muy similar a la que tendrán finalmente, una vez renderizados, resultan fácilmente personalizables), y por otro lado ficheros XML donde se describen los campos que aparecerán en los

formularios y los metadatos asociados a ellos (es decir, datos que describen estos campos, como son por ejemplo: el tipo, longitud, nombre, descripción, atributos...).

1. Plantillas HTML

   La idea es que el código HTML sea fácilmente procesable por *patForms* (el núcleo de procesamiento de formularios de la aplicación) de forma que éste pueda incluir los elementos dinámicos que necesite.

   Tras estudiar las posibilidades existentes, se decide el uso de *patTemplates*, que entre otras cosas es capaz de realizar sustitución de etiquetas dentro del código HTML, así como la repetición de secciones de código una o múltiples veces (lo cual es útil cuando se tiene un número indeterminado de controles).

   Su total integración con *patForms* facilita en gran medida la implementación, ya que actúa como una especie de filtro de salida o renderizador a la hora de generar la salida HTML, sustituyendo automáticamente las etiquetas que aparecen en la plantilla en el momento de generar el formulario.

2. Definiciones XML

   Se utilizará *patForms_Definition*, una clase auxiliar de *patForms* que realiza, con ayuda de *PEAR::XML_Serializer*, una serialización y deserialización de la estructura en memoria de la definición de los formularios. Esta estructura serializada está descrita en XML y por tanto nos servirá como base para cargar una definición inicial de los mismos a partir de un fichero.

Veremos más detalles sobre la implementación concreta de estas clases en la sección de E.2, "*Construcción de la interfaz*", del apéndice E, "*Interfaz de configuración*".

Capítulo 9

Conclusiones y líneas de avance

9.1. Conclusiones

En esta sección de la memoria de este Proyecto de Fin de Carrera exponemos las conclusiones finales del trabajo desarrollado.

Las principales conclusiones son:

- Este proyecto ofrece al «Estado del arte» actual un sistema que integra módulos y funcionalidades de distinta naturaleza, estableciendo una relación entre los mismos.
- La permisividad y flexibilidad del sistema al adaptarse a los cambios que manualmente pueda realizar el usuario en los ficheros de configuración (mediante editores de texto u otras utilidades gráficas). Se ha tenido en especial consideración que los distintos módulos respeten el formato y contenidos originales de los ficheros de configuración, haciendo que sean adaptables a los cambios provocados en los mismos.
- La ventaja en la utilización de un sistema operativo estándar y abierto.
- La enorme potencia y versatilidad del uso de librerías de código abiertas de PHP, además de la conveniencia a la hora de la reducción de costes en el desarrollo de las interfaces.
- Al basarnos en una estructura modular y de tres capas, se hace posible con facilidad la integración de nuevos módulos en el sistema así como la sustitución de los ya existentes por otros.

Así mismo, el desarrollo de este Proyecto nos ha permitido ampliar nuestros conocimientos sobre distintas materias y campos. Algunas de las aportaciones personales con las que este Proyecto nos ha enriquecido son:

- Conocimientos de herramientas (*software*) sobre seguridad y monitorización.

- Conocimientos (profundos) del lenguaje de programación PHP.
- Utilización de PHP como lenguaje orientado a objetos.
- Profundización en la estructura interna de los sistemas GNU/Linux.
- Desarrollo de paquetes Debian.
- Personalización y creación de *kernels* para GNU/Linux.

9.2. Futuras líneas de avance

Este PFC admite gran cantidad de adiciones e innovaciones en un futuro inmediato. La arquitectura modular (tanto del sistema operativo escogido como de las interfaces desarrolladas) hacen simple la incorporación de nuevas funcionalidades al sistema desarrollado.

Algunas de las posibilidades de mejora y de continuación para este Proyecto son:

1. Extender las interfaces de configuración a más parámetros además que los seleccionados, como las rutas de almacenamiento interno de los archivos, opciones adicionales en los recolectores, etc.

2. Separar por completo el código PHP del HTML, lo que implicaría la separación total de la presentación de la interfaz de su implementación. Esto se debe a que durante el desarrollo de los módulos se ha hecho uso en algún momento de pequelas cadenas de código HTML incrustado dentro del PHP.

3. Realizar mejoras en el código de los módulos desarrollados que mejoren el comportamiento de los mismos y permitan, en medida de lo posible, la «autorecuperación» de situaciones erróneas. Algunos ejemplos de esto son: permitir que *flow-capture* no se configure en interfaces inexsistentes, hacer que *FlowScan* se detenga en en caso de iniciarse el sistema y estar el servicio configurado en el arranque, etc.

4. Ampliar los aspectos del tratamiento de la información y su procesado de cada a un usuario humano, permitiendo más opciones para su estudio (especialmente para el caso de *syslog-ng*).

5. Integrar la ejecución de operaciones remotas de control en la interfaz de usuario, de manera que se facilite pasar de realizar la monitorización estadística a la retransmisión de tráfico en el sistema remoto mediante la activación del módulo *ROUTE*.

6. Desarrollar los módulos correspondientes a las demás utilidades y tecnologías estudiadas en este Proyecto, como *syslog-ng* o *flow-export* y sus complementos *flow-fanout* o *flow-send*, para permitir al usuario su configuración y manipulación.

7. Integrar finalmente la interfaz y las utilidades seleccionadas dentro de la distribución en CD autoarrancable «Anubix», desarrollada en el PFC *Anubix: Servidor de seguridad perimetral*. Si bien el diseño y la realización completas de este Proyecto se han dirigido a la compatibilidad absoluta con dicho PFC, no se ha realizado la integración dentro de la distribución «Anubix» por la extensión y complejidad que alcanzaría este Proyecto.

Apéndice A

Monitorización estadística de tráfico

Índice del capítulo

A.1. *NetFlow* versión 7

Como hemos comentado a lo largo del capítulo 4, "*Monitorización estadística de tráfico*", tanto la sonda que genera la información como el recolector que la recibe deben ser compatibles y utilizar la misma versión del *NetFlow*. En nuestro caso, la sonda y el recolector seleccionados son compatibles en usar las versiones 1, 5 y 7 del protocolo *NetFlow*.

Vamos a hacer en esta sección un acercamiento a la versión 7 del protocolo que es la más actual de las que pueden soportar ambas aplicaciones de sonda y recolector utilizados mostrando los campos de la cabecera y del registro en los Cuadros A.1 y A.2 respectivamente [5].

Bytes	***Contents***	***Description***
0-1	version	*NetFlow export format version number*
2-3	count	*Number of flows exported in this flow frame (protocol data unit, or PDU)*
4-7	SysUptime	*Current time in milliseconds since the export device booted*
8-11	unix_secs	*Current seconds since 0000 UTC 1970*
12-15	unix_nsecs	*Residual nanoseconds since 0000 UTC 1970*
16-19	flow_sequence	*Sequence counter of total flows seen*
20-23	reserved	*Unused (zero) bytes*

Cuadro A.1: Campos de la cabecera de *NetFlow* V7

Puede verse que la cabecera de *NetFlow* del Cuadro A.1 transporta información general, como `SysUptime` (tiempo que lleva operativo el sistema que genera el mensaje) o `unix_secs` (marca de tiempo formada por los segundos transcurridos desde el 1 de enero de 1970). El campo `count`, segundo de la cabecera, indica el número de registros que se entregan dentro de este mensaje. Esto se entenderá mejor con la siguiente explicación: una cabecera de *NetFlow* va acompañada de varios campos de datos, también llamados registros o en la literatura inglesa *flows*, como el que se muestra en el Cuadro A.2. Así, gracias al campo `count`, el receptor del mensaje sabe cuantos de estos *flows* debe buscar en el mensaje recibido.

Es gracias a los contenidos del registro de *NetFlow* del Cuadro A.2 cuando se puede entender lo referido en el punto 4.2.3, "*La sonda y el protocolo NetFlow*", donde comentábamos las peculiaridades de la sonda al manejar los contenidos de *NetFlow*. Campos como `nexthop` (siguiente encaminador o *router* de la ruta a seguir) o `src_as` (número del *sistema autónomo*[1] de origen) carecerían de sentido si la información es proveniente de un *sniffer* como es el caso de nuestra sonda. Además, merece resaltar que a pesar de estar definidos algunos de los campos no portan información al ser sus contenidos siempre «cero» y casualmente son muchos de los campos que nuestra sonda no podría utilizar adecuadamente.

[1] En la literatura consultada, un *sistema autónomo* o *AS* es una red o conjunto de redes que comparten una política común en cuanto a las reglas de encaminamiento. Habitualmente pertenece a una única entidad administrativa (como una universidad o división de una empresa) y a cada *AS* se le asigna de forma global un identificador numérico. Las redes dentro del *AS* comparten la información de encaminamiento con protocolos IGP y el *AS* la comparte con otros *AS* mediante BGP.

Bytes	***Contents***	***Description***
0-3	srcaddr	*Source IP address; in case of destination-only flows, set to zero.*
4-7	dstaddr	*Destination IP address.*
8-11	nexthop	*Next hop router; always set to zero.*
12-13	input	*SNMP index of input interface; always set to zero.*
14-15	output	*SNMP index of output interface.*
16-19	dPkts	*Packets in the flow.*
20-23	dOctets	*Total number of Layer 3 bytes in the packets of the flow.*
24-27	First	*SysUptime, in milliseconds, at start of flow.*
28-31	Last	*SysUptime, in milliseconds, at the time the last packet of the flow was received.*
32-33	srcport	*TCP/UDP source port number; set to zero if flow mask is destination-only or source-destination.*
34-35	dstport	*TCP/UDP destination port number; set to zero if flow mask is destination-only or source-destination.*
36	flags †	*Flags indicating, among other things, what flow fields are invalid.*
37	tcp_flags	*TCP flags; always set to zero.*
38	prot	*IP protocol type (for example, TCP = 6; UDP = 17); set to zero if flow mask is destination-only or source-destination.*
39	tos	*IP type of service; switch sets it to the ToS of the first packet of the flow.*
40-41	src_as	*Source autonomous system number, either origin or peer; always set to zero.*
42-43	dst_as	*Destination autonomous system number, either origin or peer; always set to zero.*
44	src_mask	*Source address prefix mask; always set to zero.*
45	dst_mask	*Destination address prefix mask; always set to zero.*
46-47	flags †	*Flags indicating, among other things, what flows are invalid.*
48-51	router_sc ⋆	*IP address of the router that is bypassed by the Catalyst 5000 series switch. This is the same address the router uses when it sends NetFlow export packets. This IP address is propagated to all switches bypassing the router through the FCP protocol.*

† Cambio sobre *NetFlow* version 5.

⋆ Adición sobre *NetFlow* version 5.

Cuadro A.2: Campos del registro de *NetFlow* V7

A.2. Instalación y configuración de la sonda

A.2.1. Instalación de *fprobe-ng*

Procedemos en este punto y en los siguientes al proceso de instalación y configuración de la sonda seleccionada, *fprobe-ng*.

Al estar disponible como *paquete Debian* la instalación es extremadamente sencilla:

```
#> apt-get install fprobe-ng
```

Con el siguiente comando se podrán consultar los contenidos del paquete:

```
#> dpkg -L fprobe-ng
```

A.2.2. Configuración de *fprobe-ng*

La sonda se puede configurar cómodamente atendiendo a su fichero de configuración en `/etc/default/fprobe-ng`. Así mismo, listamos el contenido por defecto del archivo de configuración para estudiar su contenido:

`/etc/default/fprobe-ng` (original)

```
#fprobe-ng default configuration file

INTERFACE="eth0"
FLOW_COLLECTOR="localhost:555"

#fprobe can't distinguish IP packet from other (e.g. ARP)
OTHER_ARGS="-fip"
```

El archivo de configuración es leído por *fprobe-ng* en su arranque, y son simplemente parámetros de la línea de comandos para la ejecución del programa. Podríamos ejecutar *fprobe-ng* igualmente llamando a su binario y dándole una idéntica lista de parámetros a los que podemos encontrar en el archivo de configuración.

También hay que comentar que ***fprobe-ng*** emplaza una *script* de arranque en `/etc/init.d` para que su arranque se produzca junto al arranque del sistema.

Para entender mejor el archivo de configuración vamos a presentar los parámetros principales de configuración de *fprobe-ng* como se indica en el Cuadro A.3. Para más información acerca de estos y otros parámetros puede consultarse el manual del programa [10].

Hacemos ahora las siguientes consideraciones referentes a la estructura de nuestra red que nos servirán de ejemplo para plasmar una configuración concreta en la utilidad:

- La sonda va a escuchar en modo promiscuo en la interfaz eth0 de una máquina de nuestra red.
- La sonda enviará la información a un recolector que escuchará en la máquina 192.168.100.24 de nuestra red, en el puerto 555.

Parámetro	Descripción
-i `interface`	Interfaz en la que *fprobe-ng* escuchará en modo promiscuo.
-n `version`	Versión del protocolo *NetFlow* a utilizar. Soporta las versiones 1,5 y 7. Por defecto utilizará la 5.
-f `expression`	Sentencia en formato *tcpdump* que se utilizará para filtrar la captura de datos.
`remote:port`	Indican la dirección IP y el puerto donde el recolector espera el flujo de *NetFlow*. *fprobe-ng* puede mandar la misma información a más de un recolector.

Cuadro A.3: Parámetros de *fprobe-ng*

- La sonda enviará los datos sobre *NetFlow* versión 7, compatible con el recolector.

Tras esta descripción, listamos finalmente el contenido de nuestro archivo de configuración:

`/etc/default/fprobe-ng` (modificado)

```
#fprobe-ng default configuration file

INTERFACE="eth0"
FLOW_COLLECTOR="192.168.100.24:555"

#fprobe can't distinguish IP packet from other (e.g. ARP)
OTHER_ARGS="-fip -n 7"
```

De esta forma, la sonda escuchará en modo promiscuo en el interfaz seleccionado y mandará hacia el recolector ubicado en el puerto 555 del sistema con dirección IP 192.168.100.24 la información generada a través de *NetFlow* versión 7.

A.3. Instalación y configuración del recolector

A.3.1. Instalación de *flow-tools*

Procedemos en este punto y en los siguientes al proceso de instalación y configuración del paquete de herramientas para el manejo de *NetFlow* ***flow-tools***.

Como también se provee en un *paquete Debian*, su instalación es simple y rápida:

```
#> apt-get install flow-tools
```

El contenido del paquete se podrá listar con la siguiente orden:

```
#> dpkg -L flow-tools
```

A.3.2. Configuración de *flow-capture*

Nos centraremos en este punto únicamente en el recolector *flow-capture* comprendido dentro del paquete ***flow-tools***. Su configuración se establece en el archivo `/etc/flow-tools/flow-capture.conf`. De dicho fichero, listamos su contenido por defecto:

`/etc/flow-tools/flow-capture.conf` (original)

```
# Configuration for flow-capture
#
# Robin Elfrink <robin@a1.nl>
#
# Every line is basically just the options to flow-capture, see
# flow-capture(1) for explanation.


# Example 1:
# Capture flows from router at 10.1.1.10, listening at port 3000.
# Store flows in /var/flow/myrouter.
-w /var/flow/myrouter 0/10.1.1.10/3000

# Example 2:
# Capture flows from router at 10.3.2.6, listening at port 3002.
# Store flows in /var/flow/mysecondrouter. Rotate files every
# 5 minutes.
-w /var/flow/mysecondrouter -n 275 0/10.3.2.6/3002

# Example 3:
# Same as above, but only listen at address 10.3.2.5, and store
# files under 'YYYY/YYYY-MM/YYYY-MM-DD' directories.
-w /var/flow/mysecondrouter -n 275 -N 3 10.3.2.5/10.3.2.6/3002
```

Como puede verse, el fichero está claramente comentado y facilita su comprensión, pero aun así comentamos algunos de sus parámetros de configuración principales en el Cuadro A.4. Para más información, véase su manual [15].

Tengamos en cuenta los siguientes puntos sobre la estructura de la red:

- La máquina que va a alojar al recolector es la misma que la que aloja a la sonda, es decir, se aloja en la dirección IP 192.168.100.24.
- La sonda usada y el recolector son compatibles en utilizar las versiones de *NetFlow* 1, 5 y 7, y se seleccionará esta última versión dado que fue la elegida en la configuración de la sonda.

Parámetro	**Descripción**
-w `workdir`	Especifica el directorio principal donde el recolector almacenará la información proveniente de la sonda.
-n `rotations`	Indica el número de ficheros que almacenará *flow-capture* durante un día, o lo que es lo mismo el intervalo temporal para guardar la información en el disco. El valor por defecto es 95, es decir, crea un archivo cada 15 minutos.
-e `expire_count`	Número de archivos máximo a almacenar por el recolector. En caso de sobrepasarse se procede a la eliminación de los más antiguos. Se comprobarán todos los subdirectorios del directorio principal del recolector.
-E `expire_size`	Tamaño máximo ocupado por los archivos capturados por el recolector. En caso de sobrepasarse se procede a la eliminación de los más antiguos. Se comprobarán todos los subdirectorios del directorio principal del recolector.
-v `pdu_version`	Especifica la versión de *NetFlow* a usar.
-N `nesting_level`	Indica el formato de anidamiento de los archivos: -3 YYYY/YYYY-MM/YYYY-MM-DD/flow-file -2 YYYY-MM/YYYY-MM-DD/flow-file -1 YYYY-MM-DD/flow-file 0 flow-file 1 YYYY/flow-file 2 YYYY/YYYY-MM/flow-file 3 YYYY/YYYY-MM/YYYY-MM-DD/flow-file Por defecto el valor es 3.
`localip/remoteip/port`	Establece la dirección local donde se escuchará, la dirección remota a la que se aceptarán los flujos y el puerto donde se atienden las llegadas. Un valor 0 en los dos primeros hará válida cualquier dirección.

Cuadro A.4: Parámetros de *flow-capture*

Con todo esto, mostramos finalmente nuestro fichero de configuración:

`/etc/flow-tools/flow-capture.conf` (modificado)

```
# Configuration for flow-capture
#
# Robin Elfrink <robin@a1.nl>
#
# Every line is basically just the options to flow-capture, see
# flow-capture(1) for explanation.

# Capturamos flujos provenientes de 192.168.100.24 escuchando en
    192.168.100.24:555
# Esperamos NetFlow Version 7
# Almacenamos en /var/flow/eth0
-V 7 -w /var/flow/eth0 192.168.100.24/192.168.100.24/555
```

Por tanto, nos quedaría funcionando según el esquema mostrado en la Figura A.1.

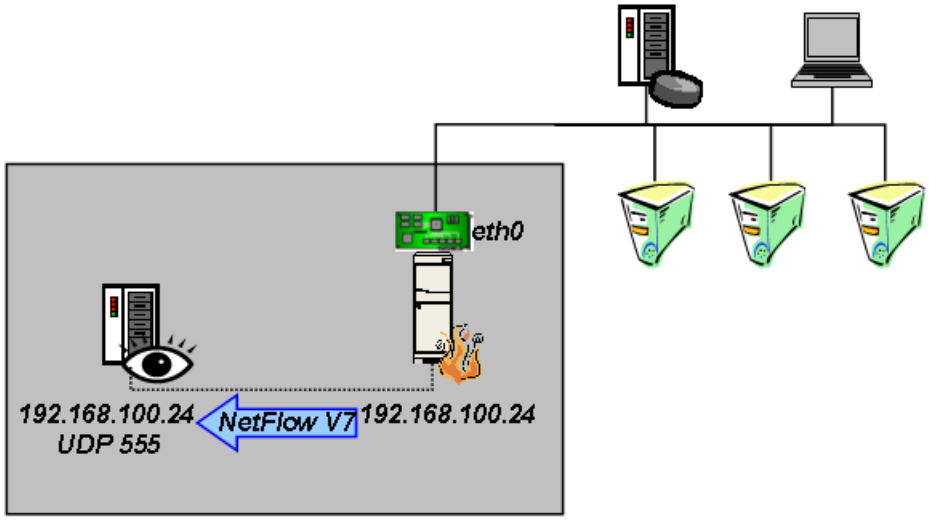


Figura A.1: Esquema de funcionamiento de la sonda y el recolector

Al igual que ocurría con ***fprobe-ng***, el archivo de configuración es leído en el arranque y equivale a introducir la misma línea de parámetros en la ejecución del binario. Así mismo se emplaza una *script* en `/etc/init.d` para su arranque con el sistema.

Por último, merece la pena resaltar que *flow-capture* solo aceptará la información de sondas que estén listadas en su archivo de configuración, lo que ofrece una seguridad nimia. Por ello recomendamos seguir las consideraciones realizadas en la sección 4.4, "*Consideraciones de seguridad para NetFlow*".

Apéndice B

Monitorización de tráfico

Índice del capítulo

B.1. Instalación del módulo *ROUTE*

B.1.1. Obtención de los paquetes

Como se adelantó en el punto 5.2.2, "*iptables y el módulo ROUTE*", de este documento en esta ocasión no hemos podido utilizar *paquetes Debian* para recoger las utilidades que hemos necesitado para efectuar la monitorización de tráfico mediante la retransmisión de los paquetes con *iptables*. A continuación vamos a seguir el proceso completo de obtención, configuración e instalación del los paquetes del ***kernel*** de Linux y de ***iptables*** para conseguir esta funcionalidad.

Comenzaremos obteniendo los distintos paquetes que necesitamos, descargándolos de Internet. Primero obtengamos el ***kernel***:

```
#> cd /usr/src

#> wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.15.1.tar.bz2
--11:56:33--  http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.15.1.tar.bz2
           => 'linux-2.6.15.1.tar.bz2'
Resolving www.kernel.org... 204.152.191.5, 204.152.191.37
Connecting to www.kernel.org[204.152.191.5]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 39,844,313 [application/x-bzip2]
```

```
100%[==============================>] 39,844,313    66.11K/s    ETA 00:00

12:03:15 (96.92 KB/s) - 'linux-2.6.15.1.tar.bz2' saved [39844313/39844313]

#> wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.15.1.tar.bz2.sign
--12:06:26--  http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.15.1.tar.bz2.
    sign
           => 'linux-2.6.15.1.tar.bz2.sign'
Resolving www.kernel.org... 204.152.191.37, 204.152.191.5
Connecting to www.kernel.org[204.152.191.37]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 248 [application/pgp-signature]

100%[==============================>] 248           --.--K/s

12:06:27 (2.37 MB/s) - 'linux-2.6.15.1.tar.bz2.sign' saved [248/248]
```

A continuación obtendremos las fuentes de ***iptables***:

```
#> wget ftp://ftp.netfilter.org/pub/iptables/iptables-1.3.5.tar.bz2
--12:10:11--  ftp://ftp.netfilter.org/pub/iptables/iptables-1.3.5.tar.bz2
           => 'iptables-1.3.5.tar.bz2'
Resolving ftp.netfilter.org... 213.95.27.115
Connecting to ftp.netfilter.org[213.95.27.115]:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD /pub/iptables ... done.
==> PASV ... done.    ==> RETR iptables-1.3.5.tar.bz2 ... done.
Length: 191,820 (unauthoritative)

100%[==============================>] 191,820       76.84K/s

12:10:27 (76.63 KB/s) - 'iptables-1.3.5.tar.bz2' saved [191820]

#> wget ftp://ftp.netfilter.org/pub/iptables/iptables-1.3.5.tar.bz2.sig
--12:10:48--  ftp://ftp.netfilter.org/pub/iptables/iptables-1.3.5.tar.bz2.sig
           => 'iptables-1.3.5.tar.bz2.sig'
Resolving ftp.netfilter.org... 213.95.27.115
Connecting to ftp.netfilter.org[213.95.27.115]:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD /pub/iptables ... done.
==> PASV ... done.    ==> RETR iptables-1.3.5.tar.bz2.sig ... done.
Length: 65 (unauthoritative)

100%[==============================>] 65            --.--K/s

12:11:03 (33.71 KB/s) - 'iptables-1.3.5.tar.bz2.sig' saved [65]
```

Y en tercer lugar, obtengamos las de ***patch-o-matic***:

```
#> wget ftp://ftp.netfilter.org/pub/patch-o-matic-ng/snapshot/patch-o-matic-ng
    -20060206.tar.bz2
--11:58:51--  ftp://ftp.netfilter.org/pub/patch-o-matic-ng/snapshot/patch-o-matic
    -ng-20060206.tar.bz2
           => 'patch-o-matic-ng-20060206.tar.bz2'
Resolving ftp.netfilter.org... 213.95.27.115
Connecting to ftp.netfilter.org[213.95.27.115]:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD /pub/patch-o-matic-ng/snapshot ... done.
==> PASV ... done.    ==> RETR patch-o-matic-ng-20060206.tar.bz2 ... done.
Length: 364,284 (unauthoritative)

100%[==============================>] 364,284       31.86K/s    ETA 00:00

11:59:27 (20.04 KB/s) - 'patch-o-matic-ng-20060206.tar.bz2' saved [364284]

#> wget ftp://ftp.netfilter.org/pub/patch-o-matic-ng/snapshot/patch-o-matic-ng
    -20060206.tar.bz2.md5sum
--12:01:10--  ftp://ftp.netfilter.org/pub/patch-o-matic-ng/snapshot/patch-o-matic
    -ng-20060206.tar.bz2.md5sum
```

```
          => 'patch-o-matic-ng-20060206.tar.bz2.md5sum'
Resolving ftp.netfilter.org... 213.95.27.115
Connecting to ftp.netfilter.org[213.95.27.115]:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD /pub/patch-o-matic-ng/snapshot ... done.
==> PASV ... done.    ==> RETR patch-o-matic-ng-20060206.tar.bz2.md5sum ... done.
Length: 68 (unauthoritative)

100%[==============================>] 68            --.--K/s

12:01:28 (29.97 KB/s) - 'patch-o-matic-ng-20060206.tar.bz2.md5sum' saved [68]
```

Comprobaremos a continuación la integridad de los archivos descargados. Comencemos igualmente por el *kernel*. Necesitaremos obtener la firma de *kernel.org* para poder comprobar la integridad del archivo [32, 33]:

```
#> gpg --keyserver wwwkeys.pgp.net --recv-keys 0x517D0F0E
gpg: keyring '/root/.gnupg/secring.gpg' created
gpg: requesting key 517D0F0E from hkp server wwwkeys.pgp.net
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: key 517D0F0E: public key "Linux Kernel Archives Verification Key <
    ftpadmin@kernel.org>" imported
gpg: no ultimately trusted keys found
gpg: Total number processed: 1
gpg:               imported: 1

#> gpg --verify linux-2.6.15.1.tar.bz2.sign linux-2.6.15.1.tar.bz2
gpg: Signature made Sun Jan 15 07:58:31 2006 CET using DSA key ID 517D0F0E
gpg: Good signature from "Linux Kernel Archives Verification Key <ftpadmin@kernel
    .org>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: C75D C40A 11D7 AF88 9981  ED5B C86B A06A 517D 0F0E
```

El proceso es idéntico para verificar la integridad de *iptables* [34]:

```
#> gpg --keyserver wwwkeys.pgp.net --recv-keys 0xCA9A8D5B
gpg: requesting key CA9A8D5B from hkp server wwwkeys.pgp.net
gpg: key CA9A8D5B: public key "Netfilter Core Team <coreteam@netfilter.org>"
    imported
gpg: no ultimately trusted keys found
gpg: Total number processed: 1
gpg:               imported: 1

gpg --verify iptables-1.3.5.tar.bz2.sig iptables-1.3.5.tar.bz2
gpg: Signature made Wed Feb  1 14:03:57 2006 CET using DSA key ID CA9A8D5B
gpg: Good signature from "Netfilter Core Team <coreteam@netfilter.org>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 02AC E2A4 74DD 09D7 FD45  2E2E 35FA 89CC CA9A 8D5B
```

Por último comprobaremos la integridad de *patch-o-matic-ng* [35]:

```
#> md5sum -c -v patch-o-matic-ng-20060206.tar.bz2.md5sum
patch-o-matic-ng-20060206.tar.bz2 OK
```

Ya solo falta descomprimir las fuentes de los tres archivos:

```
#> tar -xvjf linux-2.6.15.1.tar.bz2
...

#> tar -xvjf iptables-1.3.5.tar.bz2
...

#> tar -xvjf patch-o-matic-ng-20060206.tar.bz2
...
```

B.1.2. Aplicando *patch-o-matic-ng* para compatibilizar con el módulo *ROUTE*

En esta sección veremos el uso de *patch-o-matic-ng* para parchear tanto al *kernel* como a *iptables* y hacerlos compatibles con el módulo *ROUTE* [29, 30].

patch-o-matic-ng se maneja de una forma muy sencilla [31]. Se basa en una *script* que nos va preguntando sucesivamente por cada uno de los módulos que contiene si deseamos instalarlos o no. *patch-o-matic-ng* comprueba qué módulos están ya instalados y cuales no son compatibles para instalarlos (bien porque han quedado obsoletos o bien porque las fuentes que queremos parchear han quedado anticuadas) de manera que nos evita que cometamos errores de manera cómoda. *patch-o-matic-ng* tiene además divididos los módulos que contiene en varios paquetes distintos, como *updates*, *submitted*, *pending*, *base*, *extra* u *obsolete*. Como es evidente, el nombre del paquete hace referencia al estado en que se encuentran los módulos que contiene. En nuestro caso el módulo *ROUTE* está en el paquete *extra* [29].

Veamos su uso[1]:

```
#> cd /usr/src/patch-o-matic-ng-20060205/

#> ./runme extra --kernel-path=/usr/src/linux-2.6.15.1 --iptables-path=/usr/src/
     iptables-1.3.5
Loading patchlet definitions..... done
```

Podremos responder a cada pregunta de la *script* con *N* para no aplicar el parche, *Y* para aplicar el parche, *T* para comprobar si el parche podría ser aplicado sin problemas o con *Q* para finalizar la *script*, entre otras opciones. Nosotros no aplicaremos ningún parche hasta llegar a la pregunta sombre el módulo *ROUTE*:

```
Welcome to Patch-o-matic ($Revision: 4088 $)!

Kernel:   2.6.15, /usr/src/linux-2.6.15.1
Iptables: 1.3.5, /usr/src/iptables-1.3.5
Each patch is a new feature: many have minimal impact, some do not.
Almost every one has bugs, so don't apply what you don't need!
-----------------------------------------------------
Already applied: NETMAP iprange

Testing ROUTE... not applied
The ROUTE patch:
   Author: éCdric de Launois <delaunois@info.ucl.ac.be>
   Status: Experimental


  This option adds a 'ROUTE' target, which enables you to setup unusual
  routes. For example, the ROUTE lets you route a received packet through
  an interface or towards a host, even if the regular destination of the
  packet is the router itself. The ROUTE target is also able to change the
  incoming interface of a packet.

  The target can be or not a final target. It has to be used inside the
  mangle table.

  ROUTE target options:
  --oif   ifname    Send the packet out using 'ifname' network interface.
  --iif   ifname    Change the packet's incoming interface to 'ifname'.
  --gw    ip        Route the packet via this gateway.
```

[1] Nótese que al parecer hubo un error en el empaquetado de esta versión de *patch-o-matic-ng* ya que el nombre del paquete y del directorio que se crea no son coincidentes.

```
  --continue        Route the packet and continue traversing the rules.
  --tee             Route a copy of the packet, but continue traversing
                    the rules with the original packet, undisturbed.

  Note that --iif, --continue, and --tee, are mutually exclusive.

  Examples :

  # To force all outgoing icmp packet to go through the eth1 interface
  # (final target) :
  iptables -A POSTROUTING -t mangle -p icmp -j ROUTE --oif eth1

  # To tunnel outgoing http packets and continue traversing the rules :
  iptables -A POSTROUTING -t mangle -p tcp --dport 80 -j ROUTE --oif tunl1 --
       continue

  # To forward all ssh packets to gateway w.x.y.z, and continue traversing
  # the rules :
  iptables -A POSTROUTING -t mangle -p tcp --dport 22 -j ROUTE --gw w.x.y.z --
       continue

  # To change the incoming network interface from eth0 to eth1 for all icmp
  # packets (final target) :
  iptables -A PREROUTING -t mangle -p icmp -i eth0 -j ROUTE --iif eth1

  # To copy (duplicate) all traffic from and to a local ECHO server
  # to a second box (nonfinal target)
  iptables -A PREROUTING -t mangle -p tcp --dport 7 -j ROUTE --gw 1.2.3.4 --tee
  iptables -A POSTROUTING -t mangle -p tcp --sport 7 -j ROUTE --gw 1.2.3.4 --tee

----------------------------------------------------------
Do you want to apply this patch [N/y/t/f/a/r/b/w/q/?]
```

A esta pregunta responderemos *Y*, y en el siguiente módulo ya podremos responder *Q* para finalizar la *script*.

```
Excellent! Source trees are ready for compilation.
```

Podemos ver como para la versión de *iptables* seleccionada algunos de los módulos incluidos en el paquete *extra* ya están incluidos como indica la sentencia «*Already applied: NETMAP iprange*». Esto es una muestra del rápido desarrollo y evolución a la que están sometidas tanto *iptables* como *patch-o-matic-ng*, y que hacen difícil a veces conseguir versiones compatibles de varios módulos entre sí y con un mismo *iptables* y *kernel*. Así mismo, la evolución de *patch-o-matic-ng* es todavía más violenta y caótica, desapareciendo y reapareciendo módulos nuevos y viejos de una versión a la siguiente.

A pesar de estos problemas, como ve se este proceso de parcheado de las fuentes es muy simplificado por *patch-o-matic-ng*, y ahora podemos pasar a compilar tanto el *kernel* como *iptables*.

B.1.3. Compilación e instalación del *kernel*

Pasaremos ahora a compilar el *kernel* [36] compatible con el módulo *ROUTE*. Primero será necesario obtener un par de paquetes necesarios para poder configurar y compilar el *kernel*:

```
#> apt-get install build-essential libncurses5-dev
```

No es nuestro objetivo centrarnos en el estudio de dichos paquetes. Instalandos ambos, usaremos un menú presentado en consola para activar como módulo (en lugar de establecerlo como *built-in*) el módulo *ROUTE* en el *kernel*.

```
#> cd /usr/src/linux-2.6.15.1

#> make menuconfig
```

Con la última orden, se nos mostrará un menú como el mostrado en la Figura B.1.

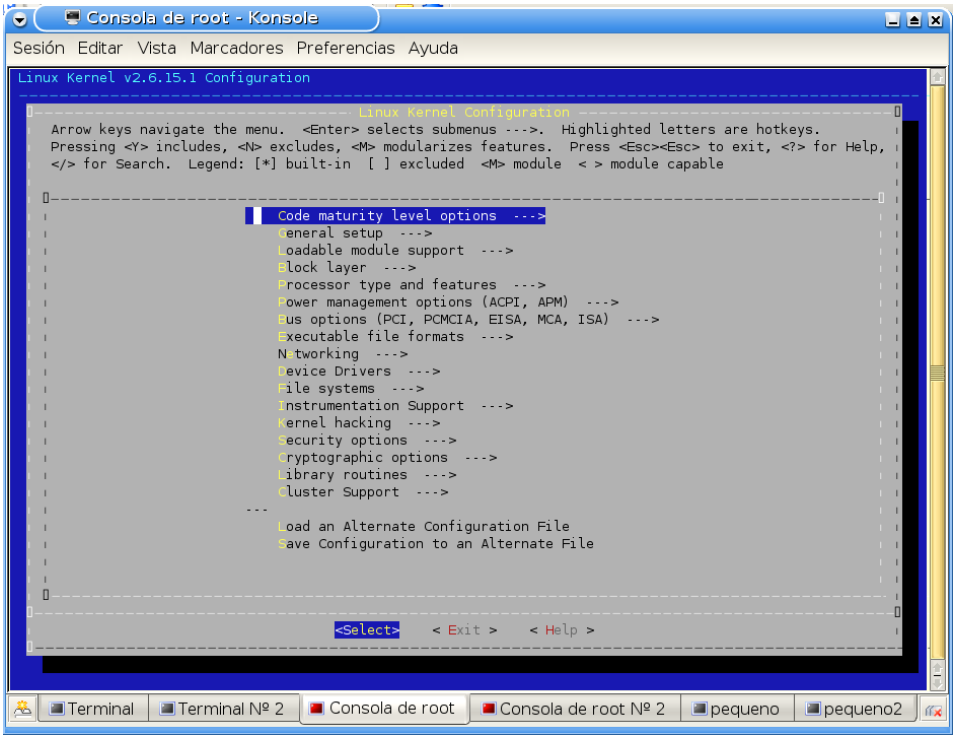


Figura B.1: Menú de configuración del *kernel* en modo consola

Gracias a este menú se pueden configurar las distintas opciones que componen nuestro *kernel*. Concretamente, el módulo *ROUTE* que deseamos configurar se encuentra en:

```
Networking
        Networking Options
                Network packet filtering (replaces ipchains)
                        IP: Netfilter Configuration
                                ROUTE target support (NEW)
```

Tras activarlo como módulo (M), podremos salir del menú grabando la configuración del nuevo *kernel*. Tras esto solo nos resta compilar e instalar:

```
#> cd /usr/src/linux-2.6.15.1

#> make all
...

#> make modules_install
...

#> cp /usr/src/linux-2.6.15.1/System.map /boot/System.map-2.6.15.1-ROUTE
```

```
#> cp /usr/src/linux-2.6.15.1/arch/i386/boot/bzImage /boot/vmlinuz-2.6.15.1-ROUTE

#> mkinitrd -o /boot/initrd.img-2.6.15.1-ROUTE 2.6.15.1
```

Tras esto, solo nos queda añadir este *kernel* al archivo `/boot/grub/menu.lst` las siguientes líneas para habilitar este núcleo en el arranque del sistema:

`/boot/grub/menu.lst` (líneas añadidas)

```
    title           Debian GNU/Linux, kernel 2.6.15.1-ROUTE
    root            (hd0,1)
    kernel          /boot/vmlinuz-2.6.15.1-ROUTE root=/dev/hda2 ro
    initrd          /boot/initrd.img-2.6.15.1-ROUTE
5   savedefault
    boot
```

B.1.3.1. Creando un *paquete Debian* del *kernel*

Como alternativa a lo anterior, se ofrece la posibilidad de crear un *paquete Debian* del *kernel* [37]. El motivo de crear este paquete es facilitar el transporte de este *kernel* a más sistemas, sin tener que pasar por los pesados pasos de compilación de la sección anterior. Este sistema es recomendado encarecidamente, ya que no solo tiene las ventajas anteriores sino que nos permite almacenar el *kernel* y nos ahorra tener que volver a repetir el proceso (muy costoso en términos de tiempo).

Para empezar será necesario crear un enlace simbólico al directorio donde reside el *kernel* ya configurado. Para ello podemos ejecutar lo siguiente:

```
#> ln -s /usr/src/linux /usr/src/linux-2.6.15.1
```

Con esto tendremos el directorio `/usr/src/linux` que será un enlace al directorio real donde se ubica nuestro *kernel*.

También necesitaremos descargar e instalar el siguiente paquete:

```
#> apt-get install kernel-package
```

El paquete ***kernel-package*** dispone de una serie de utilidades que facilitan la creación de *paquetes Debian* de *kernels* para los sistemas. Nosotros sólo vamos a mencionar los comandos y parámetros necesarios para obtener el *paquete Debian* del *kernel* deseado, sin centrarnos especialmente en todas sus capacidades.

Lo primero será tener el *kernel* adecuadamente configurado, como se hizo en la sección B.1.3, "*Compilación e instalación del kernel*". También limpiaremos los resultados de la compilación de la sección anterior para enfatizar el hecho de que la construcción del paquete del *kernel* es independiente a la compilación anterior. Para ello se pueden utilizar los comandos:

```
#> cd /usr/src/linux

#> make menuconfig
...

#> make clean
...
```

Ya estamos preparados para crear el paquete del *kernel*. Con el programa *make-kpkg* incluído dentro del paquete ***kernel-package*** primero nos aseguraremos de limpiar la imagen del *kernel* que vamos a empaquetar de posibles intentos anteriores (fallidos o no) de empaquetado. Para ello ejecutaremos:

```
#> cd /usr/src/linux

#> make-kpkg clean
...
```

Finalmente crearemos el paquete del *kernel*. Para poder diferenciar la versión que estamos compilando de otras versiones compiladas y, sobre todo, para evitar dañar la versión del *kernel* que esté actualmente siendo ejecutada en el sistema (en el caso de ser la misma versión que la que vamos a empaquetar) utilizaremos el parámetro `--append-to-version` de *make-kpkg*. Esto nos permitirá evitar los problemas anteriormente citados además de permitirnos diferenciar con comodidad distintas versiones de *kernels* que hayamos empaquetado. El comando para empaquetar el *kernel* sería el siguiente:

```
#> make-kpkg --initrd --bzimage --append-to-version=-route kernel_image
...
```

La ejecución de este comando nos creará en el directorio padre del directorio donde es ejecutado, es decir en `/usr/src`, el *paquete Debian* correspondiente:

```
linux-image-2.6.15.1-route_2.6.15.1-route-10.00.Custom_i386.deb
```

La forma de utilizar estos paquetes es muy sencilla: se instalarán con el gestor de paquetes *dpkg* de los sistemas **Debian Sarge**. En su instalación el paquete ubicará en los lugares adecuados los módulos del *kernel* (típicamente en `/lib/modules`)y los demás archivos, y realizará las modificaciones adecuadas al archivo `/boot/grub/menu.lst` que permitirán arrancar el nuevo *kernel*, todo ello de forma automática y transparente.

B.1.4. Compilación en instalación de *iptables*

El siguiente paso será actualizar la versión de *iptables* del sistema sustituyéndola por una versión compatible con el módulo *ROUTE*. Dado que en el sistema **Debian Sarge** viene ya una versión de *iptables* instalada, lo primero que haremos será eliminarla del sistema al completo para evitar posibles problemas de cruce de versiones:

```
#> apt-get remove --purge iptables
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  iptables*
0 upgraded, 0 newly installed, 1 to remove and 1 not upgraded.
Need to get 0B of archives.
After unpacking 1270kB disk space will be freed.
Do you want to continue? [Y/n] Y
(Reading database ... 103701 files and directories currently installed.)
Removing iptables ...
```

Tras esto, podemos pasar a compilar e instalar *iptables*, algo que se hace rápidamente:

```
#> cd /usr/src/iptables-1.3.5

#> make KERNEL_DIR=/usr/src/linux-2.6.15.1
...

#> make install KERNEL_DIR=/usr/src/linux-2.6.15.1
...
```

Con esto ya tenemos el nuevo *iptables* compatible con el módulo *ROUTE* instalado en nuestro sistema. Tras esto podremos reiniciar el sistema con el nuevo *kernel* para activar la compatibilidad con *ROUTE*.

B.1.4.1. Creando un *paquete Debian* de *iptables*

El proceso de creado de un *paquete Debian* para *iptables* [38, 39] difiere bastante del sistema para hacer un paquete del *kernel* que se estudió en la sección B.1.3.1, "*Creando un paquete Debian del kernel*". Los motivos de querer hacer un *paquete Debian* son los mismos: aumentar la facilidad en el transporte e instalación de los paquetes necesarios en todas las máquinas de nuestro sistema de monitorización.

Empezaremos obteniendo los paquetes necesarios con la siguiente orden:

```
#> apt-get install dh-make fakeroot
```

No es nuestro objetivo centrarnos en el uso y funcionamiento de las utilidades comprendidas en dichos paquetes. Tras ello, nos moveremos al directorio de las fuentes de *iptables* ya descomprimidas, donde limpiaremos los objetos ya compilados. También modificaremos el nombre del directorio de las fuentes para diferenciar el paquete resultante del paquete original:

```
#> mv /usr/src/iptables-1.3.5 /usr/src/iptables-1.3.5-ROUTE

#> cd /usr/src/iptables-1.3.5-ROUTE

#> make clean
```

Ahora ejecutaremos el comando ***dh_make*** que acabamos de obtener en el paquete ***dh-make*** que nos preparará una serie de archivos para poder realizar posteriormente nuestro paquete.

```
#> dh_make -e rmicmir@gmail.com -c gpl -p iptables-1.3.5-route -f ../iptables
    -1.3.5.tar.bz2

Type of package: single binary, multiple binary, library, kernel module or cdbs?
 [s/m/l/k/b] s

Maintainer name : Rafael Mico Miranda
Email-Address   : rmicmir@gmail.com
Date            : Wed,  7 Jun 2006 12:35:14 +0200
Package Name    : iptables-1.3.5-ROUTE
Version         : 1.3.5-ROUTE
License         : gpl
Type of Package : Single
Hit <enter> to confirm:
Done. Please edit the files in the debian/ subdirectory now. You should also
check that the iptables-1.3.5-ROUTE Makefiles install into $DESTDIR and not in /
    .
```

Esto nos creará un directorio `debian` dentro de nuestra estructura de las fuentes de *iptables*, es decir, nos creará el directorio `/debian` dentro de la ruta de `/usr/src/iptables-1.3.5-ROUTE`. En él encontraremos una serie de archivos que habrá que editar para configurar correctamente el paquete. Ha sido necesario realizar modificaciones en los siguientes ficheros:

```
/usr/src/iptables-1.3.5-ROUTE/Makefile
/usr/src/iptables-1.3.5-ROUTE/Rules.make
/usr/src/iptables-1.3.5-ROUTE/debian/changelog
/usr/src/iptables-1.3.5-ROUTE/debian/control
/usr/src/iptables-1.3.5-ROUTE/debian/copyright
/usr/src/iptables-1.3.5-ROUTE/debian/rules
```

Los archivos `Makefile` y `Rules.make` han sido editados para arreglar problemas en la creación del paquete. Las modificaciones realizadas han sido, respectivamente, las siguientes:

`/usr/src/iptables-1.3.5-ROUTE/Makefile` (modificado)

```
    ifndef KERNEL_DIR
15  KERNEL_DIR=/usr/src/linux
    endif
    IPTABLES_VERSION:=1.3.5
    OLD_IPTABLES_VERSION:=1.3.4

20

    LIBDIR:=/lib
    BINDIR:=/sbin
    MANDIR:=/usr/share/man
    INCDIR:=/include
25
    # directory for new iptables releases
    RELEASE_DIR:=/tmp
```

`/usr/src/iptables-1.3.5-ROUTE/Rules.make` (modificado)

```
    # Have to handle extensions which no longer exist.
    clean: $(EXTRA_CLEANS)
            rm -f $(SHARED_LIBS) $(EXTRAS) $(EXTRAS_EXP) $(SHARED_LIBS:%.so=%_sh.o)
10          rm -f extensions/initext.c extensions/initext6.c
            @find . -name '*.[ao]' -o -name '*.so' | xargs rm -f

    install: all $(EXTRA_INSTALLS)
            @if [ -f /usr/local/bin/iptables ];\
15          then echo 'Erasing iptables from old location (now "$(BINDIR)").';\
```

```
        rm -f /usr/local/bin/iptables;\
        fi

install-experimental: $(EXTRA_INSTALLS_EXP)
```

Las modificaciones realizadas sobre los archivos citados emplazados en `/debian` (dentro de `/usr/src/iptables-1.3.5-ROUTE`) son mucho más simples:

`/usr/src/iptables-1.3.5-ROUTE/debian/changelog` (modificado)

```
iptables-1.3.5-route (1.3.5-ROUTE-1) experimental; urgency=low

  * Initial release
  * This is my first Debian package.
  * Adjusted the Makefile to fix $DESTDIR problems.


 -- Rafael Mico Miranda <rmicmir@gmail.com>  Wed,  7 Jun 2006 12:35:14 +0200
```

`/usr/src/iptables-1.3.5-ROUTE/debian/control` (modificado)

```
Source: iptables-1.3.5-route
Section: net
Priority: optional
Maintainer: Rafael Mico Miranda <rmicmir@gmail.com>
Build-Depends: debhelper (>= 4.0.0)
Standards-Version: 3.6.2

Package: iptables-1.3.5-route
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}
Replaces: iptables
Description: iptables with patch-o-matic ROUTE target
 iptables-1.3.5 with patch-o-matic ROUTE target
```

`/usr/src/iptables-1.3.5-ROUTE/debian/copyright` (modificado)

```
This package was debianized by Rafael Mico Miranda <rmicmir@gmail.com> on
Wed,  7 Jun 2006 12:35:14 +0200.

It was downloaded from ftp://ftp.netfilter.org/pub/iptables/iptables-1.3.5.tar.
    bz2
  and sources were modified with
  ftp://ftp.netfilter.org/pub/patch-o-matic-ng/snapshot/patch-o-matic-ng
      -20060206.tar.bz2

Copyright Holder: netfilter.org

License:

   This package is free software; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation; either version 2 of the License, or
   (at your option) any later version.

   This package is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this package; if not, write to the Free Software
   Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301 USA

On Debian systems, the complete text of the GNU General
Public License can be found in `/usr/share/common-licenses/GPL'.
```

Hecho esto, podemos pasar a compilar y empaquetar nuestro paquete, lo que haremos con la siguiente orden:

```
#> cd /usr/src/iptables-1.3.5-route

#> dpkg-buildpackage -rfakeroot
```

Esto nos creará en el directorio padre `/usr/src` los siguientes archivos:

```
iptables-1.3.5-route_1.3.5-ROUTE-1.dsc
iptables-1.3.5-route_1.3.5-ROUTE-1.tar.gz
iptables-1.3.5-route_1.3.5-ROUTE-1_i386.changes
iptables-1.3.5-route_1.3.5-ROUTE-1_i386.deb
```

El que nos interesa en cuestión es el archivo `.deb` que es el paquete instalable. Tras instalarlo con *dpkg* podemos ver qué archivos ha emplazado, y se puede comprobar que la ubicación es idéntica a la de un *iptables* normal.

```
#> /usr/src# dpkg -L iptables-1.3.5-route
/.
/usr
/usr/bin
/usr/sbin
/usr/share
/usr/share/man
/usr/share/man/man8
/usr/share/man/man8/iptables-save.8.gz
/usr/share/man/man8/ip6tables.8.gz
/usr/share/man/man8/iptables.8.gz
/usr/share/man/man8/iptables-restore.8.gz
/usr/share/doc
/usr/share/doc/iptables-1.3.5-route
/usr/share/doc/iptables-1.3.5-route/README.Debian
/usr/share/doc/iptables-1.3.5-route/copyright
/usr/share/doc/iptables-1.3.5-route/changelog.Debian.gz
/sbin
/sbin/iptables
/sbin/iptables-save
/sbin/iptables-restore
/sbin/ip6tables
/lib
/lib/iptables
/lib/iptables/libipt_ah.so
/lib/iptables/libipt_addrtype.so
/lib/iptables/libipt_comment.so
/lib/iptables/libipt_connlimit.so
/lib/iptables/libipt_connmark.so
/lib/iptables/libipt_conntrack.so
/lib/iptables/libipt_dscp.so
/lib/iptables/libipt_ecn.so
/lib/iptables/libipt_esp.so
/lib/iptables/libipt_hashlimit.so
/lib/iptables/libipt_helper.so
/lib/iptables/libipt_icmp.so
/lib/iptables/libipt_iprange.so
/lib/iptables/libipt_length.so
/lib/iptables/libipt_limit.so
/lib/iptables/libipt_mac.so
/lib/iptables/libipt_mark.so
/lib/iptables/libipt_multiport.so
/lib/iptables/libipt_owner.so
/lib/iptables/libipt_physdev.so
/lib/iptables/libipt_pkttype.so
/lib/iptables/libipt_policy.so
/lib/iptables/libipt_realm.so
/lib/iptables/libipt_rpc.so
/lib/iptables/libipt_sctp.so
/lib/iptables/libipt_standard.so
/lib/iptables/libipt_state.so
/lib/iptables/libipt_tcp.so
/lib/iptables/libipt_tcpmss.so
/lib/iptables/libipt_tos.so
/lib/iptables/libipt_ttl.so
/lib/iptables/libipt_udp.so
```

```
/lib/iptables/libipt_unclean.so
/lib/iptables/libipt_CLASSIFY.so
/lib/iptables/libipt_CONNMARK.so
/lib/iptables/libipt_DNAT.so
/lib/iptables/libipt_DSCP.so
/lib/iptables/libipt_ECN.so
/lib/iptables/libipt_LOG.so
/lib/iptables/libipt_MARK.so
/lib/iptables/libipt_MASQUERADE.so
/lib/iptables/libipt_MIRROR.so
/lib/iptables/libipt_NETMAP.so
/lib/iptables/libipt_NFQUEUE.so
/lib/iptables/libipt_NOTRACK.so
/lib/iptables/libipt_REDIRECT.so
/lib/iptables/libipt_REJECT.so
/lib/iptables/libipt_SAME.so
/lib/iptables/libipt_SNAT.so
/lib/iptables/libipt_TARPIT.so
/lib/iptables/libipt_TCPMSS.so
/lib/iptables/libipt_TOS.so
/lib/iptables/libipt_TRACE.so
/lib/iptables/libipt_TTL.so
/lib/iptables/libipt_ULOG.so
/lib/iptables/libipt_CLUSTERIP.so
/lib/iptables/libipt_ROUTE.so
/lib/iptables/libipt_connbytes.so
/lib/iptables/libipt_dccp.so
/lib/iptables/libipt_recent.so
/lib/iptables/libipt_string.so
/lib/iptables/libip6t_connmark.so
/lib/iptables/libip6t_eui64.so
/lib/iptables/libip6t_hl.so
/lib/iptables/libip6t_icmpv6.so
/lib/iptables/libip6t_length.so
/lib/iptables/libip6t_limit.so
/lib/iptables/libip6t_mac.so
/lib/iptables/libip6t_mark.so
/lib/iptables/libip6t_multiport.so
/lib/iptables/libip6t_owner.so
/lib/iptables/libip6t_physdev.so
/lib/iptables/libip6t_policy.so
/lib/iptables/libip6t_standard.so
/lib/iptables/libip6t_state.so
/lib/iptables/libip6t_tcp.so
/lib/iptables/libip6t_udp.so
/lib/iptables/libip6t_CONNMARK.so
/lib/iptables/libip6t_HL.so
/lib/iptables/libip6t_LOG.so
/lib/iptables/libip6t_NFQUEUE.so
/lib/iptables/libip6t_MARK.so
/lib/iptables/libip6t_TRACE.so
/lib/iptables/libip6t_REJECT.so
/lib/iptables/libip6t_ROUTE.so
/lib/iptables/libip6t_ah.so
/lib/iptables/libip6t_esp.so
/lib/iptables/libip6t_frag.so
/lib/iptables/libip6t_ipv6header.so
/lib/iptables/libip6t_hbh.so
/lib/iptables/libip6t_dst.so
/lib/iptables/libip6t_rt.so
```

B.1.5. Activación del módulo *ROUTE*

En este momento, tras seguir las secciones B.1.1, B.1.2, B.1.3 y B.1.4, ya disponemos de un sistema compatible con el módulo *ROUTE* de *iptables*. Ahora veremos el uso que daremos del módulo para retransmitir los paquetes hacia la máquina que alberga el sistema de monitorización.

La mayor parte de la documentación sobre el módulo *ROUTE* ya ha sido mostrada durante la aplicación de *patch-o-matic-ng*, como se puede ver en la sección B.1.2, "*Aplicando patch-o-matic-ng para compatibilizar con el módulo ROUTE*", y podemos consultar una ayuda más resumida de la siguiente forma:

```
#> iptables -j ROUTE --help
...
ROUTE target v1.11 options:
    --oif       ifname   Route packet through 'ifname' network interface
    --iif       ifname   Change packet's incoming interface to 'ifname'
    --gw        ip       Route packet via this gateway 'ip'
    --continue           Route packet and continue traversing the
                         rules. Not valid with --iif or --tee.
    --tee                Duplicate packet, route the duplicate,
                         continue traversing with original packet.
                         Not valid with --iif or --continue.
```

Nosotros queremos aprovecharnos de la opción `--tee` [29, 30] que es la que permite duplicar un paquete mientras que el paquete original sigue siendo procesado por el resto de tablas de *iptables* de forma transparente a la duplicación. Como puede verse, *ROUTE* es un objetivo, un destino al que pueden enviarse los paquetes en *iptables* al igual que se mandarían al objetivo `ACCEPT` o a otra cadena definida por el usuario. Además de esto *ROUTE* es un objetivo que solo podrá ser utilizado dentro de la tabla `mangle` de *iptables*, la tabla utilizada para la alteración específica de paquetes.

Suponiendo que el sistema remoto es un sistema que actúa como cortafuegos de una red o *DMZ*, para analizar el tráfico que atraviesa el sistema será conveniente establecer la orden de retransmisión en la subtabla `FORWARD` de la tabla `mangle` del sistema remoto.

Ahora hacemos la suposición siguiente acerca de la estructura de la red:

- El sistema de monitorización se aloja en el sistema con dirección IP 192.168.100.21.

Con esto ya basta ejecutar el siguiente comando en el sistema remoto para para retransmitir con *iptables* [28] hacia nuestro sistema de monitorización:

```
#> iptables -t mangle -I FORWARD -j ROUTE --gw 192.168.100.21 --tee
```

Sin embargo, si deseamos ver todo el tráfico que se intenta enviar un *host* concreto, en este caso 192.168.100.22, deberíamos hacer la retransmisión antes que en la subtabla `FORWARD` por si las reglas de filtrado actuasen antes de dicha subtabla eliminando paquetes que el *host* sí está enviando:

```
#> iptables -t mangle -I PREROUTING -s 192.168.100.22 -j ROUTE --gw
     192.168.100.21 --tee

#> iptables -t mangle -I POSTROUTING -d 192.168.100.22 -j ROUTE --gw
     192.168.100.21 --tee
```

El que la retransmisión de paquetes se realice con *iptables* nos abre un abanico de posibilidades adicionales, permitiéndonos realizar retransmisiones mucho más específicas haciendo uso de sus capacidades, pero por norma general desearemos saber exactamente qué hace un *host* concreto, por lo que habitualmente se retransmitirá todo su tráfico.

A modo de detalle final, se adjunta la Figura B.2 donde se matiza la ubicación que deberán tener en la red tanto el módulo *ROUTE*, con su correspondiente *kernel* e *iptables* compatibles, y el *sniffer* o sistema que analizará el tráfico.

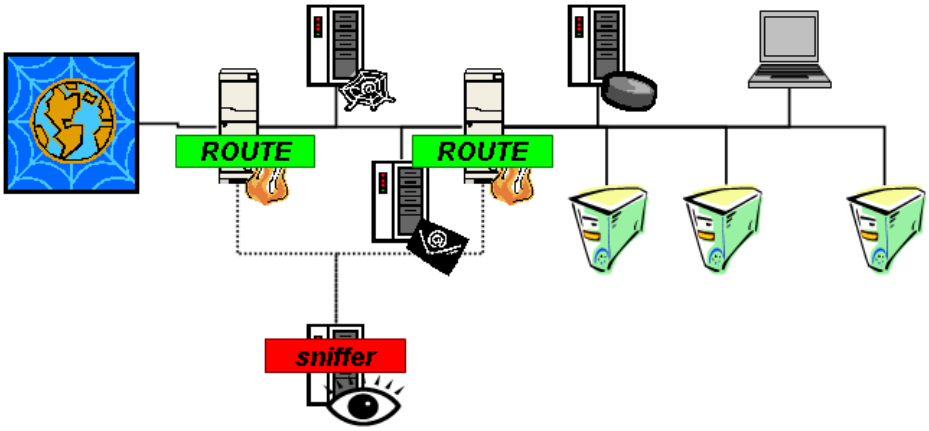


Figura B.2: Ubicación del módulo *ROUTE*

B.2. Instalación y configuración del paquete *ssh*

B.2.1. Instalación de *ssh*

El paquete ***ssh*** se instala por defecto en la instalación base del sistema **Debian Sarge**. Para comprobar que se encuentra instalado podemos hacer lo siguiente:

```
#> dpkg -l ssh
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name     Version              Description
+++-========-====================-====================
ii  ssh      3.8.1p1-8.sarge.4    Secure rlogin/rsh/rcp replacement (OpenSSH)
```

En caso de que no apareciese como instalado, siempre podemos instalarlo con su correspondiente orden de *apt*:

```
#> apt-get install ssh
```

Los contenidos del paquete se podrán consultar con la siguiente orden:

```
#> dpkg -L ssh
```

B.2.2. Configuración de *ssh*

Su archivo de configuración se encuentra en `/etc/ssh/ssh_config`, y no vamos a realizar ninguna modificación en ella.

Como se vio en la sección B.1.5, "*Activación del módulo ROUTE*", será necesario realizar una acción sobre el sistema remoto que contiene las versiones modificadas del *kernel* y de *iptables* para activar el módulo *ROUTE*. El funcionamiento normal de *ssh* se basa en una autenticación de usuario y clave para acceder a la sesión en el sistema remoto. En nuestro caso queremos eliminar la necesidad de la contraseña para que las acciones de control puedan realizarse de una forma más transparente de manera que no se necesitase la interacción de una persona para introducir la clave, pero manteniendo unos niveles de seguridad mínimos.

Por ello, vamos a proceder a crear un par de llaves cliente-servidor [40, 41] para poder autenticar de forma cómoda al sistema que alberga el sistema de monitorización (que de cara a la aplicación *ssh* hará las veces de cliente) contra el sistema que retransmitirá el tráfico (que hará las veces de servidor). Estas llaves identificarán al cliente en el servidor y permitirán la acción de control sin tener que mediar una clave en el proceso, lo que es ideal para ser utilizado luego en sistemas más automatizados.

Sí hay que resaltar que para poder manejar *iptables* en un sistema es necesario tener los privilegios de `root`, así que en lo que sigue tendremos que tomar el papel de ese usuario y las conexiones del cliente a los servidores deberán hacerse basándose en la cuenta de `root`.

Comenzaremos creando en el cliente, la máquina que retransmitirá el tráfico, el par de llaves cliente servidor:

(Sistema de monitorización)

```
#> cd /root/.ssh

#> ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
fe:19:7b:86:ac:32:84:6e:4f:f9:33:ce:e5:1d:c7:29 root@debian

#> ls -lash
total 20K
4.0K drwx------   2 root root 4.0K Feb 12 16:18 .
4.0K drwxr-xr-x  14 root root 4.0K Feb 12 16:16 ..
4.0K -rw-------   1 root root  668 Feb 12 16:18 id_rsa
4.0K -rw-r--r--   1 root root  601 Feb 12 16:18 id_rsa.pub
4.0K -rw-r--r--   1 root root  239 Feb 12 16:16 known_hosts
```

Para que no nos solicite claves en un futuro será necesario establecer el *passphrase* en blanco. La ejecución del comando anterior nos ha generado los archivos `id_rsa` e `id_rsa.pub`, donde el segundo archivo es el que contiene la llave pública. Ahora tenemos que distribuir la llave del cliente a los sistemas remotos que albergan los *kernels* e *iptables* modificados, de manera que todos

tengan la misma llave y podamos conectarnos a todos ellos desde el sistema de monitorización. Para ello distribuiremos el archivo `id_rsa.pub` a los sistemas remotos y lo añadiremos a su archivo `authorized_keys` en cada sistema remoto:

(Sistema remoto)

```
#> cd /root/.ssh

#> cat id_rsa.pub >> autorized_keys
```

Una vez hecho esto podemos efectuar fácilmente los comandos remotos, ya que el servidor que contiene el archivo `authorized_keys` actualizado recibirá del cliente la llave adecuada que realizará la autenticación. En el ejemplo siguiente estaríamos ordenando al sistema remoto 192.168.100.24 que retransmita el tráfico que la atraviesa a 192.168.100.21 que sería nuestro sistema de monitorización en el que se ejecutaría un *sniffer*, como se detalla en la Figura B.3:

(Sistema remoto)

```
#> ssh root@192.168.100.24 iptables -t mangle -I FORWARD -j ROUTE --gw
     192.168.100.21 --tee
```

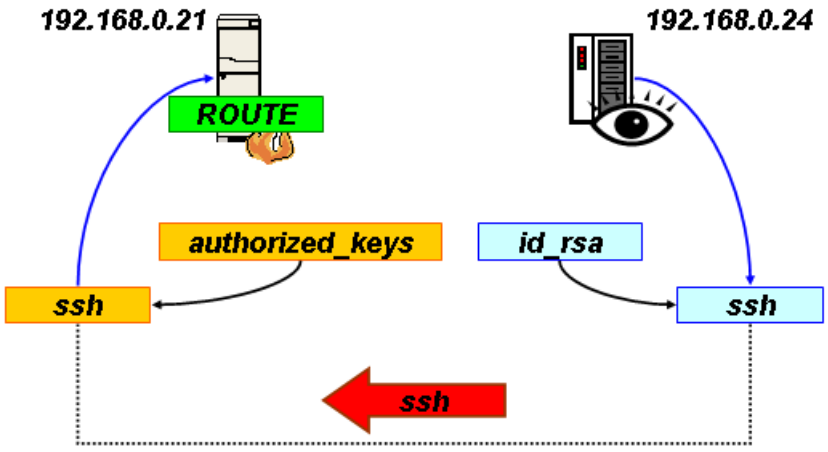


Figura B.3: Uso de *ssh* y sus llaves para la ejecución de comandos

Merece la pena recordar que la autenticación con el par de llaves cliente-servidor autentifica a un usuario concreto, en este caso `root`, entre los dos sistemas, por ello hay que emplazar los archivos en los directorios de `/root/.ssh` tanto en los servidores (sistemas remotos) como en el cliente (sistema de monitorización).

Apéndice C

Recogida de registros

Índice del capítulo

C.1. Instalación y configuración de *syslog-ng*

C.1.1. Instalación de *syslog-ng*

Procedemos en este punto y en los siguientes al proceso de instalación y configuración del manejador de eventos y mensajes del sistema *syslog-ng*.

La instalación se puede realizar cómodamente con el siguiente comando gracias a que se provee la utilidad en formato *paquete Debian*:

```
#> apt-get install syslog-ng
```

Si así lo deseamos, los contenidos del paquete se podrán consultar como sigue:

```
#> dpkg -L syslog-ng
```

C.1.2. Configuración de *syslog-ng*

La configuración de *syslog-ng* se realiza mediante su archivo de configuración en `/etc/syslog-ng/syslog-ng.conf` [51, 52]. El archivo de configuración es extenso pero tiene una estructura clara además de densamente comentada, y explica la filosofía de funcionamiento de *syslog-ng* a la vez que nos introduce sus

parámetros de configuración. Lo presentamos desglosando su estructura en sus secciones principales:

1. En primer lugar aparecen una serie de opciones comunes al funcionamiento del programa.

`/etc/syslog-ng/syslog-ng.conf` (I - opciones)

```
#
# Configuration file for syslog-ng under Debian
#
# attempts at reproducing default syslog behavior

# the standard syslog levels are (in descending order of priority):
# emerg alert crit err warning notice info debug
# the aliases "error", "panic", and "warn" are deprecated
# the "none" priority found in the original syslogd configuration is
# only used in internal messages created by syslogd


######
# options

options {
        # disable the chained hostname format in logs
        # (default is enabled)
        chain_hostnames(0);

        # the time to wait before a died connection is re-established
        # (default is 60)
        time_reopen(10);

        # the time to wait before an idle destination file is closed
        # (default is 60)
        time_reap(360);

        # the number of lines buffered before written to file
        # you might want to increase this if your disk isn't catching with
        # all the log messages you get or if you want less disk activity
        # (say on a laptop)
        # (default is 0)
        #sync(0);

        # the number of lines fitting in the output queue
        log_fifo_size(2048);

        # enable or disable directory creation for destination files
        create_dirs(yes);

        # default owner, group, and permissions for log files
        # (defaults are 0, 0, 0600)
        #owner(root);
        group(adm);
        perm(0640);

        # default owner, group, and permissions for created directories
        # (defaults are 0, 0, 0700)
        #dir_owner(root);
        #dir_group(root);
        dir_perm(0755);

        # enable or disable DNS usage
        # syslog-ng blocks on DNS queries, so enabling DNS may lead to
        # a Denial of Service attack
        # (default is yes)
        use_dns(no);

        # maximum length of message in bytes
        # this is only limited by the program listening on the /dev/log
            Unix
        # socket, glibc can handle arbitrary length log messages, but --
            for
        # example -- syslogd accepts only 1024 bytes
```

```
        # (default is 2048)
        #log_msg_size(2048);
};
```

2. A continuación se definen una secuencia de fuentes de los datos que *syslog-ng* manejará. Estos orígenes serán llamadas al sistema de programas locales o remotos, las cuales se recibirán mediante una conexión TCP o UDP.

`/etc/syslog-ng/syslog-ng.conf` (II - fuentes)

```
######
# sources

# all known message sources
source s_all {
        # message generated by Syslog-NG
        internal();
        # standard Linux log source (this is the default place for the
            syslog()
        # function to send logs to)
        unix-stream("/dev/log");
        # messages from the kernel
        file("/proc/kmsg" log_prefix("kernel: "));
        # use the above line if you want to receive remote UDP logging
            messages
        # (this is equivalent to the "-r" syslogd flag)
        # udp();
};
```

En este caso en vez de establecerse distintas fuentes hay definida una única fuente, `s_all`, que recibe mensajes desde una serie de puntos distintos. Se podrían haber especificado una serie de fuentes distintas, como `s_internal`, `s_unix_stream` o `s_file`, cada una de ellas con un punto de entrada distinto igualmente.

3. En tercer lugar se enumeran una serie de destinos o sumideros de los datos que el programa recoja durante su ejecución. Los destinos podrán ser ficheros de texto, programas o incluso salidas hacia otras máquinas mediante flujos TCP o UDP.

`/etc/syslog-ng/syslog-ng.conf` (III - destinos)

```
######
# destinations

# some standard log files
destination df_auth { file("/var/log/auth.log"); };
destination df_syslog { file("/var/log/syslog"); };
destination df_cron { file("/var/log/cron.log"); };
destination df_daemon { file("/var/log/daemon.log"); };
destination df_kern { file("/var/log/kern.log"); };
destination df_lpr { file("/var/log/lpr.log"); };
destination df_mail { file("/var/log/mail.log"); };
destination df_user { file("/var/log/user.log"); };
destination df_uucp { file("/var/log/uucp.log"); };

# these files are meant for the mail system log files
# and provide re-usable destinations for {mail,cron,...}.info,
# {mail,cron,...}.notice, etc.
destination df_facility_dot_info { file("/var/log/$FACILITY.info"); };
destination df_facility_dot_notice { file("/var/log/$FACILITY.notice"); };
destination df_facility_dot_warn { file("/var/log/$FACILITY.warn"); };
destination df_facility_dot_err { file("/var/log/$FACILITY.err"); };
destination df_facility_dot_crit { file("/var/log/$FACILITY.crit"); };

# these files are meant for the news system, and are kept separated
```

```
# because they should be owned by "news" instead of "root"
destination df_news_dot_notice { file("/var/log/news/news.notice" owner("
    news")); };
destination df_news_dot_err { file("/var/log/news/news.err" owner("news"))
    ; };
destination df_news_dot_crit { file("/var/log/news/news.crit" owner("news
    ")); };

# some more classical and useful files found in standard syslog
    configurations
destination df_debug { file("/var/log/debug"); };
destination df_messages { file("/var/log/messages"); };

# pipes
# a console to view log messages under X
destination dp_xconsole { pipe("/dev/xconsole"); };

# consoles
# this will send messages to everyone logged in
destination du_all { usertty("*"); };
```

Aquí, por ejemplo, se define el destino `df_syslog` que será el que nos creará el archivo de *syslog-ng* en `/var/log/syslog`.

4. Lo siguiente que encontramos en el archivo de configuración son una lista de filtros que permiten identificar con mayor exactitud el origen del mensaje y nos permiten realizar una separación de los mensajes.

`/etc/syslog-ng/syslog-ng.conf` (IV - filtros)

```
######
# filters

# all messages from the auth and authpriv facilities
filter f_auth { facility(auth, authpriv); };

# all messages except from the auth and authpriv facilities
filter f_syslog { not facility(auth, authpriv); };

# respectively: messages from the cron, daemon, kern, lpr, mail, news,
    user,
# and uucp facilities
filter f_cron { facility(cron); };
filter f_daemon { facility(daemon); };
filter f_kern { facility(kern); };
filter f_lpr { facility(lpr); };
filter f_mail { facility(mail); };
filter f_news { facility(news); };
filter f_user { facility(user); };
filter f_uucp { facility(uucp); };

# some filters to select messages of priority greater or equal to info,
    warn,
# and err
# (equivalents of syslogd's *.info, *.warn, and *.err)
filter f_at_least_info { level(info..emerg); };
filter f_at_least_notice { level(notice..emerg); };
filter f_at_least_warn { level(warn..emerg); };
filter f_at_least_err { level(err..emerg); };
filter f_at_least_crit { level(crit..emerg); };

# all messages of priority debug not coming from the auth, authpriv, news,
    and
# mail facilities
filter f_debug { level(debug) and not facility(auth, authpriv, news, mail)
    ; };

# all messages of info, notice, or warn priority not coming form the auth,
# authpriv, cron, daemon, mail, and news facilities
filter f_messages {
        level(info,notice,warn)
```

```
                and not facility(auth,authpriv,cron,daemon,mail,news);
};

# messages with priority emerg
filter f_emerg { level(emerg); };

# complex filter for messages usually sent to the xconsole
filter f_xconsole {
    facility(daemon,mail)
        or level(debug,info,notice,warn)
        or (facility(news)
                and level(crit,err,notice));
};
```

En este caso, fijándonos de nuevo en el *syslog-ng*, se define un filtro `f_syslog`.

5. Por último, unas definiciones de registro que asociarán las fuentes, filtros y destinos en unidades de registro.

`/etc/syslog-ng/syslog-ng.conf` (y V - registros)

```
######
# logs
# order matters if you use "flags(final);" to mark the end of processing
    in a
# "log" statement

# these rules provide the same behavior as the commented original syslogd
    rules

# auth,authpriv.*                 /var/log/auth.log
log {
        source(s_all);
        filter(f_auth);
        destination(df_auth);
};

# *.*;auth,authpriv.none          -/var/log/syslog
log {
        source(s_all);
        filter(f_syslog);
        destination(df_syslog);
};

# this is commented out in the default syslog.conf
# cron.*                          /var/log/cron.log
#log {
#        source(s_all);
#        filter(f_cron);
#        destination(df_cron);
#};

# daemon.*                        -/var/log/daemon.log
log {
        source(s_all);
        filter(f_daemon);
        destination(df_daemon);
};

# kern.*                          -/var/log/kern.log
log {
        source(s_all);
        filter(f_kern);
        destination(df_kern);
};

# lpr.*                           -/var/log/lpr.log
log {
        source(s_all);
        filter(f_lpr);
        destination(df_lpr);
};
```

```
# mail.*                          -/var/log/mail.log
log {
        source(s_all);
        filter(f_mail);
        destination(df_mail);
};

# user.*                          -/var/log/user.log
log {
        source(s_all);
        filter(f_user);
        destination(df_user);
};

# uucp.*                          /var/log/uucp.log
log {
        source(s_all);
        filter(f_uucp);
        destination(df_uucp);
};

# mail.info                       -/var/log/mail.info
log {
        source(s_all);
        filter(f_mail);
        filter(f_at_least_info);
        destination(df_facility_dot_info);
};

# mail.warn                       -/var/log/mail.warn
log {
        source(s_all);
        filter(f_mail);
        filter(f_at_least_warn);
        destination(df_facility_dot_warn);
};

# mail.err                        /var/log/mail.err
log {
        source(s_all);
        filter(f_mail);
        filter(f_at_least_err);
        destination(df_facility_dot_err);
};

# news.crit                       /var/log/news/news.crit
log {
        source(s_all);
        filter(f_news);
        filter(f_at_least_crit);
        destination(df_news_dot_crit);
};

# news.err                        /var/log/news/news.err
log {
        source(s_all);
        filter(f_news);
        filter(f_at_least_err);
        destination(df_news_dot_err);
};

# news.notice                     /var/log/news/news.notice
log {
        source(s_all);
        filter(f_news);
        filter(f_at_least_notice);
        destination(df_news_dot_notice);
};


# *.=debug;\
#         auth,authpriv.none;\
#         news.none;mail.none     -/var/log/debug
```

```
log {
        source(s_all);
        filter(f_debug);
        destination(df_debug);
};


# *.=info;*.=notice;*.=warn;\
#         auth,authpriv.none;\
#         cron,daemon.none;\
#         mail,news.none          -/var/log/messages
log {
        source(s_all);
        filter(f_messages);
        destination(df_messages);
};

# *.emerg                         *
log {
        source(s_all);
        filter(f_emerg);
        destination(du_all);
};


# daemon.*;mail.*;\
#         news.crit;news.err;news.notice;\
#         *.=debug;*.=info;\
#         *.=notice;*.=warn       |/dev/xconsole
log {
        source(s_all);
        filter(f_xconsole);
        destination(dp_xconsole);
};
```

Aquí es donde, finalmente, se asocian la fuente `s_all`, el filtro `f_syslog` y el destino `df_syslog` en una misma regla para crear el archivo `/var/log/syslog` del sistema.

Si hacemos ahora las siguientes suposiciones sobre la estructura de la red:

- El sistema de monitorización se aloja en el sistema con dirección IP 192.168.100.24.
- El servicio *syslog-ng* del sistema de monitorización atenderá los mensajes en el puerto 514
- El único sistema generador de registros se aloja en la dirección IP 192.168.100.24, es decir, está en el mismo sistema que el receptor de los mensajes.

Visto cuál es el funcionamiento de *syslog-ng* y la estructura, será fácil entender el uso que le daremos:

- En los sistemas remotos, desde donde queremos recibir los registros, estableceremos los siguientes parámetros adicionales al resto de su configuración para habilitar el envío sobre TCP:

  `/etc/syslog-ng/syslog-ng.conf` (líneas a añadir en generadores de registros)

```
destination dest_tcp {
        tcp("192.168.100.24" port(514));
};
```

```
log {
        source(s_all);
        destination(dest_tcp);
};
```

Con esta configuración haremos que todos los registros se manden por TCP hacia el puerto 514 de la máquina 192.168.100.24.

- En el sistema de monitorización que recibe los registros de los demás será necesario establecer lo siguiente:

`/etc/syslog-ng/syslog-ng.conf` (líneas a añadir en receptores de registros)

```
options {
        ...
        check_hostname(yes);
        keep_hostname(yes);
};

source s_tcp {
        tcp(ip(192.168.100.24) port(514) keep-alive(yes));
};

destination hosts {
        file("/var/log/HOSTS/$HOST/$YEAR/$MONTH/$DAY/
             $FACILITY$YEAR$MONTH$DAY"
        owner(root) group(root) perm(0600) dir_perm(0700) create_dirs(yes)
             );
};
destination df_syslog_common {
        file("/var/log/syslog_common");
};

log {
        source(s_all);
        filter(f_syslog);
        destination(df_syslog_common);
};

log {
        source(s_tcp);
        filter(f_syslog);
        destination(df_syslog_common);
};

log {
        source(s_all);
        destination(hosts);
};

log {
        source(s_tcp);
        destination(hosts);
};
```

Con esta configuración haremos lo siguiente:

- Las opciones `check_hostname` y `keep_hostname` evitarán que se pierda el nombre del sistema que generó el registro, es decir, estas opciones serán las que nos mantendrán la autoría del mensaje.
- Establecemos una fuente, `s_tcp`, que escucha las llegadas de los mensajes de los sistemas remotos.
- Especificamos un destino, `hosts`, que cree una estructura de directorios a partir de `/var/log/HOSTS` en la que podremos diferencias los mensajes por fecha, máquina y facilidad de la máquina que lo originó.

- Especificamos un destino, `df_syslog_common` para la aglutinación conjunta de todos los `syslog`, en el archivo `/var/log/syslog_common`.
- Se hacen unas especificaciones de registros para que tanto los mensajes locales como remotos se almacenen tanto en la estructura de directorios de `/var/log/HOSTS` como en el archivo `/var/log/syslog_common`.

Por tanto, nos quedaría funcionando el sistema según se indica en la Figura C.1.

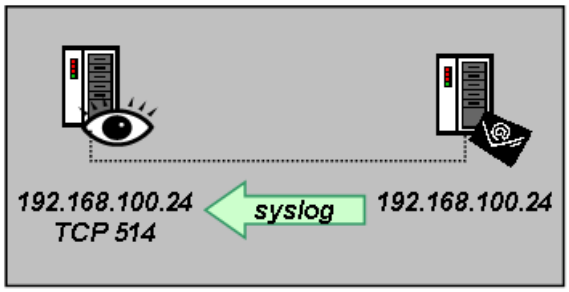


Figura C.1: Esquema de funcionamiento de *syslog-ng*

Si aceptamos esta configuración, será importante reconfigurar *logrotate* para que el archivo `syslog_common` sea también rotado junto al resto de registros del sistema. Para ellos atendemos a su configuración en `/etc/logrotate.d/syslog-ng`. En dicho archivo, podremos añadir al final las siguientes líneas para que nuestro nuevo `syslog_common` sea tratado igual que el viejo `syslog`:

`/etc/logrotate.d/syslog-ng` (líneas a añadir)

```
/var/log/syslog_common {
   rotate 7
   daily
   compress
   postrotate
      /etc/init.d/syslog-ng reload >/dev/null
   endscript
}
```

C.2. Consideraciones de seguridad: transporte mediante *stunnel*

C.2.1. Instalación de *stunnel*

Como se adelantó en la sección 6.4, "*Consideraciones de seguridad: stunnel y openssl*", de este documento, *stunnel* nos va a permitir realizar un transporte seguro de los mensajes de los registros del sistema entre las distintas aplicaciones *syslog-ng* de los sistemas de nuestra red. Para ello utilizaremos un transporte sobre TCP de los mensajes de *syslog-ng*, algo imposible de hacer con *syslogd*

ya que tan sólo admitía su transporte sobre UDP, además de una configuración específica que se verá a lo largo de esta sección.

Al igual que la mayoría de las utilidades estudiadas en este documento, ***stunnel*** se proporciona en un *paquete Debian* y su instalación se realiza cómodamente:

```
#> apt-get install stunnel
```

Los contenidos del paquete se podrán consultar como sigue:

```
#> dpkg -L stunnel
```

C.2.2. Configuración de *stunnel*

Para realizar el transporte seguro de los mensajes de *syslog-ng* sobre *stunnel*, este último tendrá que ser configurado en dos configuraciones distintas, una como sistema cliente y otra como sistema servidor (nuestro sistema de monitorización) al que se conectarán los distintos clientes emisores de los mensajes. Para ello es necesario conocer algunos de los parámetros de configuración de esta utilidad, que comentamos en el Cuadro C.1. Para más detalle sobre estos y otros parámetros puede consultarse el manual de la aplicación [56].

Parámetro	**Descripción**
-d `[host:]port`	Modo demonio. *stunnel* escuchará peticiones en la dirección (opcional) y puerto indicados. Si no se especifica dirección escuchará en INADDR_ANY.
-r `[host:]port`	Destino. Indican a *stunnel* la dirección (opcional) y puerto a los que mandar la información. Si no se especifica dirección los mandará a INADDR_LOOPBACK.
-c	Modo cliente. Usar este parámetro hará que la comunicación hacia el destino (especificada con -r) sea la comunicación SSL.
-A `certfile`	Fichero de la Autoridad de Certificación.
-p `pemfile`	Fichero del certificado del servidor.
-v `level`	Nivel de la certificación: 1 Comprobar la validez del certificado si se presenta un certificado. 2 Comprueba la validez del certificado del cliente (exigiendo uno). 3 Comprueba la validez del certificado del cliente contra un certificado local. Por defecto no hay verificación.

Cuadro C.1: Parámetros de *stunnel*

Si queremos hacer ahora que *syslog-ng* utilice esta comunicación segura [53], será necesario lo siguiente:

1. Establecer la siguiente configuración en la sistema servidor, que será el mismo sistema donde *syslog-ng* esperará recibir los mensajes de los clientes (es decir, nuestro sistema de monitorización):

`/etc/syslog-ng/syslog-ng.conf` (líneas a añadir en recolector de registros)

```
source s_tcp_ssl {
        tcp(ip(127.0.0.1) port(514) keep-alive(yes));
};

destination df_tcp_ssl_file { file("/var/log/syslog_tcp_ssl"); };

log {
        source(s_tcp_ssl);
        filter(f_syslog);
        destination(df_tcp_ssl_file);
};
```

   Con esto haremos que los mensajes recibidos por la fuente `df_tcp_ssl_file` sean enviados hacia un nuevo fichero en `/var/log/syslog_tcp_ssl`. En lugar de hacer esto podríamos realizar una salida hacia una estructura de directorios o hacia alguna de las otras que se mostraron en la sección C.1.1, "*Instalación de syslog-ng*".

2. En los sistemas remotos, que albergan los *syslog-ng* generadores de mensajes, estableceremos la siguiente configuración:

`/etc/syslog-ng/syslog-ng.conf` (líneas a añadir en emisores de registros)

```
destination df_tcp_ssl_server { tcp("127.0.0.1" port(51400)); };

log {
        source(s_all);
        destination(df_tcp_ssl_server);
};
```

   Con esta nueva definición de registro hacemos que todos los mensajes que se generen a partir de la fuente `s_all` (en principio la única fuente existente por lo que todos los mensajes provienen de ella) se envíen hacia el nuevo destino `df_tcp_ssl_server`. Nótese que si esta configuración y la anterior fuesen establecidas sobre un mismo *syslog-ng* los mensajes se entregarían a través del bucle local.

3. Por último, tenemos que arrancar los demonios de *stunnel*. En primer lugar en la máquina servidora donde está configurado *syslog-ng* para atender los mensajes de los clientes:

   ```
   #> stunnel -d 192.168.100.24:5140 -r 127.0.0.1:514
   ```

   Y en la máquina cliente:

   ```
   #> stunnel -c -d 127.0.0.1:51400 -r 192.168.100.24:5140
   ```

Con esto tenemos ya una comunicación segura entre nuestro cliente y nuestro servidor. Para clarificar todo esto, veamos la Figura C.2.

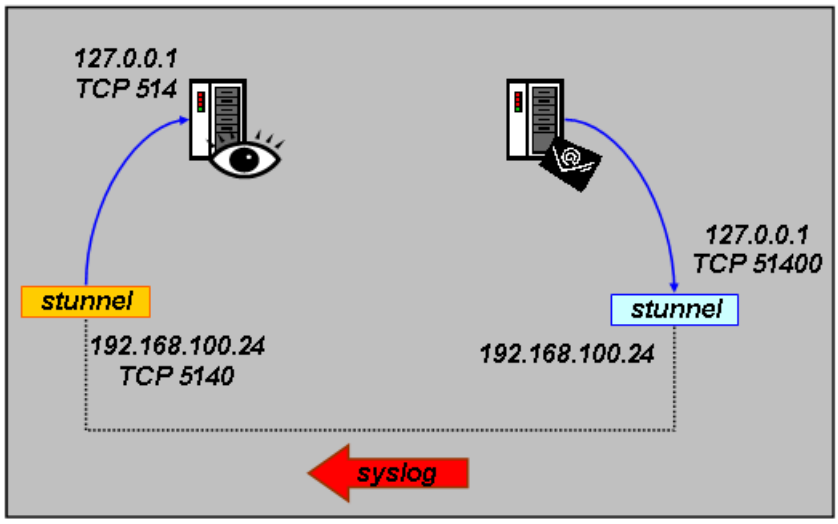


Figura C.2: Esquema de funcionamiento de *syslog-ng* con *stunnel*

C.2.3. Autenticación mediante *stunnel*: el paquete *openssl*

Una vez establecidos los túneles de comunicación segura, ahora lo ideal sería establecer una autenticación entre el cliente y el servidor para asegurarnos que sólo nuestros clientes van a mandarnos sus registros y que clientes no deseados nos llenen nuestros *logs* con falsos registros. Esto lo podemos realizar gracias a los certificados y firmas digitales, y vamos a ver a continuación cómo usarlos.

Nos basaremos en el paquete ***openssl*** que se suministra para **Debian Sarge** en su versión 0.9.7e-3. Su instalación es la siguiente:

```
#> apt-get install openssl
```

Nos centraremos sólo en los pasos necesarios para hacer uso de los certificados que es capaz de crear, sin estudiar el paquete en detalle sino sirviéndonos únicamente de lo necesario del mismo. Para trabajar con él, nos trasladaremos al siguiente directorio:

```
#> cd /usr/lib/ssl/misc
```

En el se encuentran las herramientas que utilizaremos para fabricar los certificados [54, 55]. Los pasos que realizamos son los siguientes:

1. Crear una llave de Autoridad de Certificación. Durante el proceso se nos pedirá distinta información:

```
#> ./CA.sh -newca

CA certificate filename (or enter to create)

Making CA certificate ...
Generating a 1024 bit RSA private key
.....................................................++++++
.....................................++++++
writing new private key to './demoCA/private/./cakey.pem'
```

```
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Sevilla
Locality Name (eg, city) []:Sevilla
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Escuela Superior
      de Ingenieros
Organizational Unit Name (eg, section) []:Autoridad Certificadora de la
     Escuela Superior de Ingenieros
Common Name (eg, YOUR name) []:Certificador
Email Address []:certificador@esi.us.es
```

Esto nos crea una estructura de directorios en `/usr/lib/ssl/misc/demoCA/`. Debemos tomar nota del *PEM pass phrase* que nos hará falta con posterioridad.

2. Crear la llave privada del servidor y un certificado:

```
#> ./CA.pl -newreq-nodes

Generating a 1024 bit RSA private key
.++++++
......++++++
writing new private key to 'newreq.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Sevilla
Locality Name (eg, city) []:Sevilla
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Escuela Superior
      de Ingenieros
Organizational Unit Name (eg, section) []:Laboratorio de Software de
     Fuentes Abiertas
Common Name (eg, YOUR name) []:rmicmir
Email Address []:rmicmir@sfa.es

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:password
An optional company name []:company
```

Este comando nos creará el archivo `/usr/lib/ssl/misc/newreq.pem`.

3. Firmar, contra nuestra propia Autoridad de Certificación, el certificado `newreq.pem` recién creado . Para ello necesitaremos el *PEM pass phrase* que usamos anteriormente al crear la llave de la Autoridad de Certificación.

```
#> ./CA.sh -sign

Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for ./demoCA/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 1 (0x1)
```

```
        Validity
            Not Before: Dec 13 11:28:21 2005 GMT
            Not After : Dec 13 11:28:21 2006 GMT
        Subject:
            countryName               = ES
            stateOrProvinceName       = Sevilla
            localityName              = Sevilla
            organizationName          = Escuela Superior de Ingenieros\
            organizationalUnitName    = Laboratorio de Software de Fuentes
                 Abiertas
            commonName                = rmicmir
            emailAddress              = rmicmir@sfa.es
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                6E:05:79:E1:40:DE:14:FF:72:21:CB:A3:5B:FF:DC:B5:96:F3:04:CD
            X509v3 Authority Key Identifier:
                keyid:7F:45:5C:B4:91:B4:E9:66:7D:4B:2E:D2:7A:CD:48:77:17:D5
                     :C2:92
                DirName:/C=ES/ST=Sevilla/L=Sevilla/O=Escuela Superior de
                     Ingenieros/OU=Autoridad Certificadora de la Escuela
                     Superior de Ingenieros/CN=Certificador/emailAddress=
                     certificador@esi.us.es
                serial:86:E6:23:2B:AF:64:E0:A9

Certificate is to be certified until Dec 13 11:28:21 2006 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=ES, ST=Sevilla, L=Sevilla, O=Escuela Superior de
             Ingenieros, OU=Autoridad Certificadora de la Escuela Superior
             de Ingenieros, CN=Certificador/emailAddress=certificador@esi.us
             .es
        Validity
            Not Before: Dec 13 11:28:21 2005 GMT
            Not After : Dec 13 11:28:21 2006 GMT
        Subject: C=ES, ST=Sevilla, L=Sevilla, O=Escuela Superior de
             Ingenieros , OU=Laboratorio de Software de Fuentes Abiertas, CN
             =rmicmir/emailAddress=rmicmir@sfa.es
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:a2:3c:4b:78:b5:6b:d3:e7:01:7b:a4:90:6d:93:
                    45:51:22:94:1d:22:8f:11:c3:d7:f7:2f:c2:ae:4d:
                    46:8f:28:c1:af:96:98:27:83:40:ae:70:cb:73:38:
                    2c:f3:fb:7b:1a:83:f6:3c:22:44:07:ca:93:d7:dd:
                    bc:67:af:7e:db:7b:64:72:da:43:3b:a4:4e:47:54:
                    7e:0f:ac:6d:32:52:dc:7a:72:0d:cc:4c:f7:8c:9a:
                    f5:9a:0c:f6:b6:e4:6b:db:ed:0e:2f:01:b3:29:6d:
                    e6:21:ed:c3:16:5c:fd:f7:a3:46:5b:c8:45:3b:50:
                    f9:69:77:f7:61:ad:d6:27:d1
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                6E:05:79:E1:40:DE:14:FF:72:21:CB:A3:5B:FF:DC:B5:96:F3:04:CD
            X509v3 Authority Key Identifier:
                keyid:7F:45:5C:B4:91:B4:E9:66:7D:4B:2E:D2:7A:CD:48:77:17:D5
                     :C2:92
```

```
                DirName:/C=ES/ST=Sevilla/L=Sevilla/O=Escuela Superior de
                    Ingenieros/OU=Autoridad Certificadora de la Escuela
                    Superior de Ingenieros/CN=Certificador/emailAddress=
                    certificador@esi.us.es
                serial:86:E6:23:2B:AF:64:E0:A9

    Signature Algorithm: md5WithRSAEncryption
        30:09:37:a8:c7:05:a6:b9:fb:eb:b1:58:0b:bb:be:c1:2c:a0:
        97:ee:d5:47:2b:34:6a:69:90:22:8f:cc:68:5c:41:32:a7:98:
        7b:0a:f8:73:da:e7:b1:21:53:00:56:fa:ff:47:9b:b9:d6:0f:
        6b:21:14:13:20:ff:0b:11:45:6f:41:d4:2a:90:e5:33:bb:8c:
        94:aa:01:62:f7:a4:d3:ed:8f:6a:d4:ef:93:55:e4:64:df:44:
        c8:56:b7:92:b9:89:1d:da:b5:31:8e:56:aa:ef:15:3b:dc:bd:
        8b:5b:84:ca:f9:36:15:80:48:a9:da:2f:3e:02:7b:0a:66:17:
        14:ef
-----BEGIN CERTIFICATE-----
MIIEkTCCA/qgAwIBAgIBATANBgkqhkiG9w0BAQQFADCB3zELMAkGA1UEBhMCRVMx
EDAOBgNVBAgTB1NldmlsbGExEDAOBgNVBAcTB1NldmlsbGExJzAlBgNVBAoTHkVz
Y3VlbGEgU3VwZXJpb3IgZGUgSW5nZW5pZXJvczFFMEMGA1UECxM8QXV0b3JpZGFk
IENlcnRpZmljYWRvcmEgZGUgbGEgRXNjdWVsYSBTdXBlcmlvciBkZSBJbmdlbml
cm9zMRUwEwYDVQQDEwxDZXJ0aWZpY2Fkb3IxJTAjBgkqhkiG9w0BCQEWFmNlcnRp
ZmljYWRvckBlc2kudXMuZXMwHhcNMDUxMjEzMTEyODIxWhcNMDYxMjEzMTEyODIx
WjCBwjELMAkGA1UEBhMCRVMxEDAOBgNVBAgTB1NldmlsbGExEDAOBgNVBAcTB1Nl
dmlsbGExKDAmBgNVBAoTH0VzY3VlbGEgU3VwZXJpb3IgZGUgSW5nZW5pZXJvcyAx
NDAyBgNVBAsTK0xhYm9yYXRvcmlvIGRlIFNvZnR3YXJlIGRlIEZ1ZW50ZXMgQWJp
ZXJ0YXMxEDAOBgNVBAMTB3JtaWNtaXIxHTAbBgkqhkiG9w0BCQEWDnJtaWNtaXJA
c2ZhLmVzMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCiPEt4tWvT5wF7pJBt
k0VRIpQdIo8Rw9f3L8KuTUaPKMGvlpgng0CucMtzOCzz+3sag/Y8IkQHypPX3bxn
r37be2Ry2kM7pE5HVH4PrG0yUtx6cg3MTPeMmvWaDPa25Gvb7Q4vAbMpbeYh7cMW
XP33o0ZbyEU7UPlpd/dhrdYn0QIDAQABo4IBdjCCAXIwCQYDVR0TBAIwADAsBglg
hkgBhvhCAQ0EHxYdT3BlblNTTCBHZW5lcmF0ZWQgQ2VydGlmaWNhdGUwHQYDVR0O
BBYEFG4FeeFA3hT/ciHLo1v/3LWW8wTNMIIBFgYDVR0jBIIBDTCCAQmAFH9FXLSR
tOlmfUsu0nrNSHcX1cKSoYHlpIHiMIHfMQswCQYDVQQGEwJFUzEQMA4GA1UECBMH
U2V2aWxsYTEQMA4GA1UEBxMHU2V2aWxsYTEnMCUGA1UEChMeRXNjdWVsYSBTdXBl
cmlvciBkZSBJbmdlbmllcm9zMUUwQwYDVQQLEzxBdXRvcmlkYWQgQ2VydGlmaWNh
ZG9yYSBkZSBsYSBFc2N1ZWxhIFN1cGVyaW9yIGRlIElluZ2VuaWVyb3MxFTATBgNV
BAMTDENlcnRpZmljYWRvcjElMCMGCSqGSIb3DQEJARYWY2VydGlmaWNhZG9yQGVz
aS51cy5lc4IJAIbmIyuvZOCpMA0GCSqGSIb3DQEBBAUAA4GBADAJN6jHBaa5++ux
WAu7vsEsoJfu1UcrNGppkCKPzGhcQTKnmHsK+HPa57EhUwBW+v9Hm7nWD2shFBMg
/wsRRW9B1CqQ5TO7jJSqAWL3pNPtj2rU
```

4. Unir el certificado firmado y el certificado del servidor:

```
#> cat newreq.pem newcert.pem > server_stunnel.pem
```

   Para mayor comodidad, copiaremos el archivo creado en `/etc/stunnel/server_stunnel.pem`.

5. Distribuir el archivo `/usr/lib/ssl/misc/demoCA/cacert.pem` a todos los clientes que se vayan a conectar mediante *stunnel*. Suponemos que se guardará en todos los clientes en `/etc/stunnel/cacert.pem`

6. En el servidor, habrá que arrancar *stunnel* de la siguiente forma:

```
#> stunnel -d 192.168.100.24:5140 -r 127.0.0.1:514 -p /etc/stunnel/
     server_stunnel.pem
```

7. En los clientes, se arrancará *stunnel* como sigue:

```
#> stunnel -c -d 127.0.0.1:514 -r 192.168.100.24:5140 -A /etc/stunnel/
     cacert.pem -v2
```

Tras esto, nuestro servidor de *stunnel* que espera las conexiones de los clientes sólo aceptará a los clientes que tienen un certificado válido asociado a su propia certificación. Su funcionamiento se indica en la Figura C.3.

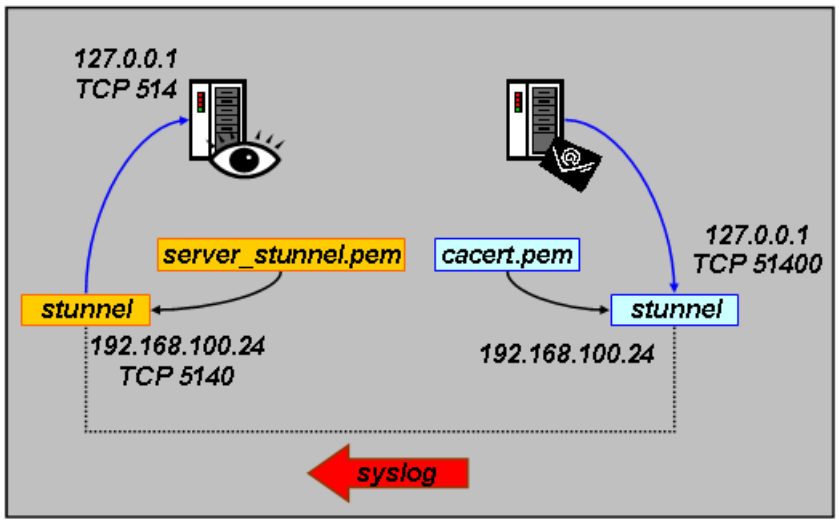


Figura C.3: Esquema de funcionamiento de *syslog-ng* con *stunnel* y certificados de *openssl*

Apéndice D

Tratamiento de la información

Índice del capítulo

D.1. Tratamiento de la información de *NetFlow*

D.1.1. Presentación mediante consola

Una vez que tenemos la información recogida por el recolector *flow-capture* está almacenada en nuestro sistema de monitorización, podemos pasar a presentarla. En este primer apartado trataremos la simple presentación mediante consola.

Para ello vamos a utilizar algunas de las utilidades incluidas en el paquete ***flow-tools***, ya instalado en el apéndice A.3.1, "*Instalación de flow-tools*", concretamente las utilidades *flow-cat* y *flow-print*.

flow-cat es la herramienta que permite leer de manera recursiva y ordenada los archivos emplazados dentro de la estructura de directorios que ha creado el recolector. Leerá los datos comprimidos del disco almacenados por *flow-capture* y su salida es de un formato intermedio que ha de ser redirigido a las otras utilidades del paquete ***flow-tools*** para que operen con los datos. La segunda de las utilidades, *flow-print*, es la que tomará esa información en formato intermedio y la mostrará en formato ASCII por la consola.

Veamos algunas de los parámetros más interesantes que presentan ambas utilidades. Los de *flow-cat* los mostramos en el Cuadro D.1 y los de *flow-print*, que dispone de muchas menos opciones, en el Cuadro D.2. Para más información sobre estas aplicaciones pueden consultarse sus páginas del manual [16, 17].

Parámetro	**Descripción**
-t `start_time`	Hora de comienzo del intervalo que se quiere estudiar.
-T `end_time`	Hora de final del intervalo.
-a	No ignorará los archivos temporales emplazados en la estructura de directorios.
`file|directory`	Archivo o ruta del directorio a analizar. En el caso del directorio es de comportamiento recursivo.

Cuadro D.1: Parámetros de *flow-cat*

Parámetro	**Descripción**
-f `format`	Formato de la salida. Algunos de ellos son: 0 Una línea: muestra interfaces, IPs, protocolos, cantidad de paquetes y octetos, y los puertos en hexadecimal. Muy abreviada. 3 Una línea: como la anterior, pero suprime los interfaces para muestrar los puertos en decimal. Más legible. 6 Una línea: muestra solo IPs y cantidad de paquetes y octetos. Útil para controlar consumos. 24 Salida completa: IPs, interfaces, puertos, protocolos, cantidad de paquetes y octetos, exportador. Muy detallada. Por defecto *flow-print* se adaptará a la información que deba visualizar.
-n	Usar nombres simbólicos donde sea adecuado si el formato de salida es `3`. Configurable en `/etc/flow-tools/sym/`.

Cuadro D.2: Parámetros de *flow-print*

En este momento es necesario recordar que la información que se muestra es información generada por la sonda, y que las capacidades de las herramientas del paquete ***flow-tools*** son directamente dependientes de la información de la sonda. Por lo tanto, la información que la sonda no inserte o inserte indebidamente en el flujo *NetFlow* será interpretada de forma directa por el recolector. Veamos pues cual es la información y el formato que es mostrado por *flow-print*:

```
#> flow-cat /var/flow/eth0/ | flow-print

srcIP               dstIP               router_sc        prot    srcPort
     dstPort          octets      packets
```

```
193.189.124.15/0    192.168.100.24/0    0.0.0.0          6      80
    32780           7756          10
192.168.100.24/0    193.189.124.15/0    0.0.0.0          6      32780
    80              1526          10
192.168.100.150/0   192.168.100.255/0   0.0.0.0          17     138
    138             237           1
192.168.100.150/0   192.168.100.255/0   0.0.0.0          17     138
    138             244           1
150.214.186.69/0    192.168.100.24/0    0.0.0.0          17     53
    32769           393           3
192.168.100.24/0    150.214.186.69/0    0.0.0.0          17     32769
    53              176           3
```

En cambio, podemos pedir a *flow-print* que nos muestre la misma información más detallada:

```
#> flow-cat /var/flow/eth0/ | flow-print -f 24

#Sif SrcIPaddress     DIf  DstIPaddress     Pr SrcP  DstP  Pkts       Octets
          StartDate  StartTime     EndDate    EndTime       ExporterAddr
    RouterSrc          Active B/Pk  Ts Fl  SrcAS DstAS

0000 193.189.124.15   0000 192.168.100.24   06 80    32780 10         7756
           2005-11-25 17:45:57.500  2005-11-25 17:46:26.040  192.168.100.24
    0.0.0.0             28.540 775   00 1b  0     0
0000 192.168.100.24   0000 193.189.124.15   06 32780 80    10         1526
           2005-11-25 17:45:57.455  2005-11-25 17:46:25.997  192.168.100.24
    0.0.0.0             28.542 152   00 1b  0     0
0000 192.168.100.150  0000 192.168.100.255  11 138   138   1          237
           2005-11-25 17:51:44.989  2005-11-25 17:51:44.989  192.168.100.24
    0.0.0.0             0.000 237    00 00  0     0
0000 192.168.100.150  0000 192.168.100.255  11 138   138   1          244
           2005-11-25 17:57:54.504  2005-11-25 17:57:54.504  192.168.100.24
    0.0.0.0             0.000 244    00 00  0     0
0000 150.214.186.69   0000 192.168.100.24   11 53    32769 3          393
           2005-11-25 17:57:40.875  2005-11-25 17:58:14.303  192.168.100.24
    0.0.0.0             33.428 131   00 00  0     0
0000 192.168.100.24   0000 150.214.186.69   11 32769 53    3          176
           2005-11-25 17:57:40.868  2005-11-25 17:58:14.296  192.168.100.24
    0.0.0.0             33.428 58    00 00  0     0
```

Por último, podemos pedir a *flow-cat* que nos muestre sólo un intervalo de tiempo concreto a modo de filtrado temporal:

```
#> flow-cat -t "11/27/05 11:00" -T "11/27/05 11:15" /var/flow/eth0/ | flow-print
    -f 24

#Sif SrcIPaddress     DIf  DstIPaddress     Pr SrcP  DstP  Pkts       Octets
          StartDate  StartTime     EndDate    EndTime       ExporterAddr
    RouterSrc          Active B/Pk  Ts Fl  SrcAS DstAS

0000 192.168.100.150  0000 192.168.100.255  11 138   138   1          237
           2005-11-27 11:03:45.345  2005-11-27 11:03:45.345  192.168.100.24
    0.0.0.0             0.000 237    00 00  0     0
0000 192.168.100.22   0000 192.168.100.24   06 42350 514   1          112
           2005-11-27 11:06:58.913  2005-11-27 11:06:58.913  192.168.100.24
    0.0.0.0             0.000 112    00 18  0     0
0000 192.168.100.24   0000 192.168.100.22   06 514   42350 1          52
           2005-11-27 11:06:58.914  2005-11-27 11:06:58.914  192.168.100.24
    0.0.0.0             0.000 52     00 10  0     0
0000 192.168.100.150  0000 192.168.100.255  11 138   138   1          244
           2005-11-27 11:12:53.562  2005-11-27 11:12:53.562  192.168.100.24
    0.0.0.0             0.000 244    00 00  0     0
```

D.1.1.1. Filtrando la información de *NetFlow*

Existen otras herramientas incluidas en el paquete ***flow-tools*** que permiten hacer estudios más afinados que los anteriormente presentados. *flow-nfilter* es una utilidad que permite añadir filtros para seleccionar con más detalle la

información que analizamos. La definición de los filtros se basa en la creación de unos ficheros de configuración que posteriormente leerá *flow-nfilter* . En el Cuadro D.3 se muestran algunos de los parámetros principales de *flow-nfilter*. Para más detalle puede consultarse el manual de la utilidad [18].

Parámetro	**Descripción**
-f `filter_fname`	Fichero que contiene las definiciones de los filtros. Por defecto usará `/etc/flow-tools/cfg/filter`.
-F `filter_definition`	Filtro a utilizar. Por defecto utilizará el filtro `default`.

Cuadro D.3: Parámetros de *flow-nfilter*

La forma de usarlo es sencilla: se filtrará la salida de *flow-cat* hacia la siguiente utilidad que estemos utilizando para analizar la información. Los parámetros para la configuración de *flow-nfilter* y la formación de sus filtros son muchos y referimos al lector a su manual para conocerlos con detalle.

En el siguiente ejemplo hemos definido un archivo llamado `filtro.cfg` con el siguiente contenido:

`/etc/flow-tools/cfg/filtro.cfg`

```
filter-primitive port80
         type ip-port
         permit 80

filter-primitive port3000
         type ip-port
         permit 3000

filter-definition FILTRADO
         match ip-source-port port80
         or
         match ip-destination-port port3000
```

Tras ello, basta solo utilizar el comando adecuado. En este caso, a modo de ejemplo, utilizamos también las capacidades de filtrado temporal de *flow-cat* para mostrar la plena compatibilidad entre aplicaciones:

```
#> flow-cat -t "11/08/05 19:00" -T "11/08/05 19:15" /var/flow/eth0/ | flow-
     nfilter -f /etc/flow-tools/cfg/filtro.cfg -F FILTRADO | flow-print

192.168.100.24/0    192.168.100.22/0    0.0.0.0          6       32886
     3000           1071         11
192.168.100.24/0    192.168.100.22/0    0.0.0.0          6       32885
     3000           860          6
192.168.100.24/0    192.168.100.22/0    0.0.0.0          6       32887
     3000           1049         10
192.168.100.24/0    192.168.100.22/0    0.0.0.0          6       32888
     3000           1124         11
192.168.100.24/0    192.168.100.22/0    0.0.0.0          6       32889
     3000           822          7
204.152.191.5/0     192.168.100.24/0    0.0.0.0          6       80
     32890          3184381      2129
192.168.100.24/0    192.168.100.22/0    0.0.0.0          6       32906
     3000           801          6
192.168.100.24/0    192.168.100.22/0    0.0.0.0          6       32903
     3000           762          5
192.168.100.24/0    192.168.100.22/0    0.0.0.0          6       32904
     3000           761          5
```

```
192.168.100.24/0    192.168.100.22/0    0.0.0.0          6       32902
    3000            761          5
192.168.100.24/0    192.168.100.22/0    0.0.0.0          6       32895
    3000            752          5
192.168.100.24/0    192.168.100.22/0    0.0.0.0          6       32898
    3000            761          5
192.168.100.24/0    192.168.100.22/0    0.0.0.0          6       32899
    3000            761          5
192.168.100.24/0    192.168.100.22/0    0.0.0.0          6       32894
    3000            802          6
192.168.100.24/0    192.168.100.22/0    0.0.0.0          6       32900
    3000            761
```

D.1.1.2. Presentanto informes en la consola

Por último, la utilidad *flow-stat* contenida en el paquete ***flow-tools*** permite realizar pequeños informes muy legibles de forma cómoda. Algunos de sus parámetros más utilizados son mostrados en el Cuadro D.4. Más detalle sobre estos y otros parámetros puede consultarse en su manual [19].

Parámetro	**Descripción**
-f `format`	Formato del informe: 7 Puerto UDP/TCP. 5 Puerto UDP/TCP de destino. 6 Puerto UDP/TCP de origen. 10 IP origen/destino. 8 IP destino. 9 IP origen.

Cuadro D.4: Parámetros de *flow-stat*

Su uso es sencillo: será el destino de la salida de *flow-cat*, por lo que también podrían utilizarse filtros intermedios definidos con *flow-nfilter*:

```
#> flow-cat  /var/flow/eth0/ | flow-stat -f 10

#  --- ---- ---- Report Information --- --- ---
#
# Fields:     Total
# Symbols:    Disabled
# Sorting:    None
# Name:       Source/Destination IP
#
# Args:       flow-stat -f 10
#
#
# src IPaddr      dst IPaddr        flows                 octets
    packets
#
192.168.100.21    192.168.100.255   13                    6478                  26
192.168.100.22    192.168.100.24    678                   9953371
    26508
192.168.100.24    192.168.100.22    739                   33897308
    38038
192.168.100.23    192.168.100.24    6                     11640                 94
192.168.100.24    192.168.100.23    6                     12224                 168
150.214.186.69    192.168.100.24    15                    7326                  39
192.168.100.24    150.214.186.69    15                    2435                  39
192.168.100.24    204.152.191.5     1                     75241
    1331
```

```
204.152.191.5     192.168.100.24    1                 3184381
    2129
192.168.100.24    216.239.59.99     6                 6945              38
216.239.59.103    192.168.100.24    2                 890               6
192.168.100.24    66.35.250.209     31                37095             277
66.35.250.209     192.168.100.24    31                237410            285
192.168.100.24    67.19.167.98      2                 1612              14
66.35.250.203     192.168.100.24    2                 11484             14
67.19.167.98      192.168.100.24    2                 6261              12
192.168.100.24    66.35.250.203     2                 2182              15
18.7.14.127       192.168.100.24    2                 6482              12
192.168.100.24    18.7.14.127       2                 1530              14
64.20.33.131      192.168.100.24    2                 2954              13
192.168.100.24    64.20.33.131      2                 3145              16
```

D.1.2. Exportando a *ntop*

Veremos en esta punto cómo exportar la información recogida en el recolector a la utilidad *ntop*, que permitirá el estudio de la información de forma gráfica y cómoda.

La exportación se puede realizar de dos formas distintas: volcando periódicamente la información almacenada por el recolector o haciendo un volcado «en tiempo real».

D.1.2.1. Volcado con emisión periódica

Con el primero de los sistemas se realizaría una emisión periódica de la información recogida por el recolector *flow-capture* emitiéndose dicha información usando el protocolo *NetFlow*, que también es capaz de interpretar *ntop*.

Haremos las siguientes suposiciones acerca de la estructura de la red para este apartado:

- El sistema que tiene *ntop* preparada para recibir los flujos de *NetFlow* se encuentra en la dirección 192.168.100.24.
- El puerto en el que atenderá los mensajes será el puerto 2055.

Para la emisión de forma periódica de la información del recolector hemos usado la utilidad *flow-send* del paquete ***flow-tools***. La herramienta permite la utilización de los parámetros mostrados en el Cuadro D.5. Para más detalle puede consultarse su manual [20].

Como puede verse esto es, a todos los efectos, convertir al recolector *flow-capture* a su vez en la sonda de otro recolector, sonda que enviará la información almacenada por el recolector. Con el siguiente comando podríamos exportar ya la información hacia el nuevo recolector:

```
#> flow-cat /var/flow | flow-send -V 7 192.168.100.24/192.168.100.24/2055
```

Mostramos la Figura D.1 para comprender mejor su idea de funcionamiento.

Es importante resaltar que *ntop* recogerá la información como si de datos capturados por él mismo se tratase, de manera que si enviamos dos veces la misma información hacia *ntop* esta se verá registrada por duplicado. Para evitar

Parámetro	Descripción
-V `pdu_version`	Especifica la versión del protocolo a utilizar.
-m `privacy_mask`	Permite poner una máscara para ocultar los hosts de la red. Alterará los campos de IP de destino e IP de origen del flujo de *NetFlow*.
`localip/remoteip/port`	Permite especificar la IP del origen del flujo de datos *NetFlow*, la IP del recolector y el puerto donde este espera el flujo.

Cuadro D.5: Parámetros de *flow-send*

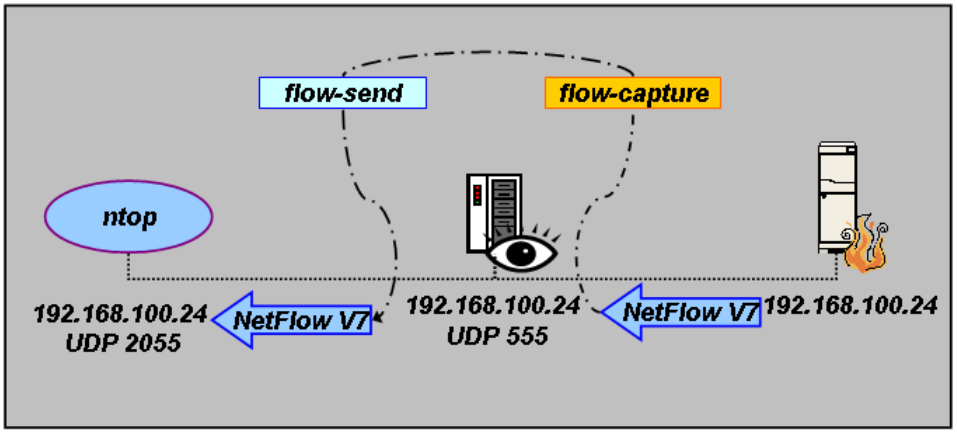


Figura D.1: Esquema de funcionamiento de *flow-send* para la exportación a *ntop*

este comportamiento podemos aprovechar el uso de la capacidad de *flow-cat* para reportar intervalos de tiempo. Presentamos a continuación un `crontab` para el demonio *cron* que emitiría la información cada hora hacia *ntop*, sin repeticiones:

`crontab`

```
    01 01 * * * flow-cat -t "00:00 Today" -T "01:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
    01 02 * * * flow-cat -t "01:00 Today" -T "02:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
    01 03 * * * flow-cat -t "02:00 Today" -T "03:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
    01 04 * * * flow-cat -t "03:00 Today" -T "04:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
 5  01 05 * * * flow-cat -t "04:00 Today" -T "05:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
    01 06 * * * flow-cat -t "05:00 Today" -T "06:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
    01 07 * * * flow-cat -t "06:00 Today" -T "07:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
    01 08 * * * flow-cat -t "07:00 Today" -T "08:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
    01 09 * * * flow-cat -t "08:00 Today" -T "09:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
10  01 10 * * * flow-cat -t "09:00 Today" -T "10:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
    01 11 * * * flow-cat -t "10:00 Today" -T "11:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
    01 12 * * * flow-cat -t "11:00 Today" -T "12:00 Today" /var/flows/eth0/ | flow-
```

```
        send -V 7 192.168.100.24/192.168.100.24/2055
    01 13 * * * flow-cat -t "12:00 Today" -T "13:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
    01 14 * * * flow-cat -t "13:00 Today" -T "14:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
15  01 15 * * * flow-cat -t "14:00 Today" -T "15:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
    01 16 * * * flow-cat -t "15:00 Today" -T "16:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
    01 17 * * * flow-cat -t "16:00 Today" -T "17:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
    01 18 * * * flow-cat -t "17:00 Today" -T "18:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
    01 19 * * * flow-cat -t "18:00 Today" -T "19:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
20  01 20 * * * flow-cat -t "19:00 Today" -T "20:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
    01 21 * * * flow-cat -t "20:00 Today" -T "21:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
    01 22 * * * flow-cat -t "21:00 Today" -T "22:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
    01 23 * * * flow-cat -t "22:00 Today" -T "23:00 Today" /var/flows/eth0/ | flow-
        send -V 7 192.168.100.24/192.168.100.24/2055
    01 00 * * * flow-cat -t "23:00 Yesterday" -T "00:00 Today" /var/flows/eth0/ |
        flow-send -V 7 192.168.100.24/192.168.100.24/2055
```

D.1.2.2. Volcado «en tiempo real»

Este segundo método hará que la información recogida por el recolector sea retransmitida al momento hacia *ntop*, por lo que la calificamos «en tiempo real».

En este apartado haremos también las siguientes suposiciones acerca de la estructura de la red :

- El sistema que tiene *ntop* preparado para recibir los flujos de *NetFlow* se encuentra en la dirección IP 192.168.100.24.
- Dicho sistema espera recibir los mensajes en el puerto 2055.

Para esta segunda forma de emisión usaremos la herramienta *flow-fanout* incluida también en el paquete ***flow-tools***. Se muestra la Figura D.2 para ilustrar el uso que se hará de *flow-fanout*.

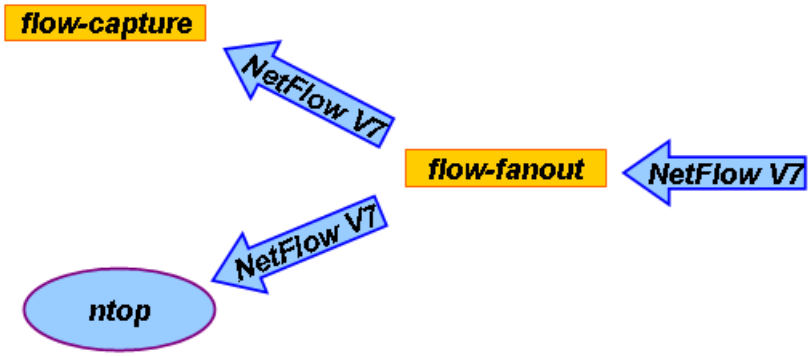


Figura D.2: Esquema de funcionamiento de *flow-fanout* para la exportación a *ntop*

El funcionamiento mostrado de esta herramienta hará que tengamos que modificar la configuración de *flow-capture* usada hasta ahora, dado que el flujo de los paquetes de *NetFlow* será el siguiente:

1. El flujo de la sonda del sistema remoto será enviado hacia *flow-fanout*.
2. *flow-fanout* retransmitirá el flujo hacia nuestro recolector, *flow-capture*.
3. *flow-fanout* retransmitirá igualmente el flujo hacia *ntop*.

Los parámetros de configuración que *flow-fanout* acepta son los indicados en el Cuadro D.6. Se pueden consultar más detalles de esta aplicación en su manual [21].

Dicho esto, y modificando la configuración de *flow-capture* para que atienda los flujos en el puerto 556 en vez de en 555, bastaría el siguiente comando para que *flow-fanout* enviase la información tanto a *flow-capture* como a *ntop* si este está escuchando en la máquina 192.168.100.24 en el puerto 2055:

```
#> flow-fanout 192.168.100.24/192.168.100.24/555
    192.168.100.24/192.168.100.24/556 192.168.100.24/192.168.100.24/2055
```

Sea cual sea el sistema escogido, una vez se dispone de la información exportada a *ntop* éste se puede manejar idénticamente a como se manejaría en el caso de captura normal, como se muestra a modo de ejemplo en la Figura D.3:

Nota: es necesario revisar la configuración de *ntop* si la sonda y él mismo están en subredes distintas y *ntop* solo atiende el tráfico de las máquinas locales. En tal caso, *ntop* no registraría la información recibida en los montajes anteriores.

Parámetro	**Descripción**
-V `pdu_version`	Versión de las PDUs que se retransmitirán.
-m `privacy_mask`	Permite poner una máscara para ocultar los hosts de la red.
-S `stat_interval`	Número de minutos del intervalo en el que *flow-fanout* imprimirá un mensaje en `stderr` indicando numero de flujos recibidos, paquetes procesados y flujos perdidos.

`localip/remoteip/port`
`localip/remoteip/port`
`...`
	En la primera tripleta: dirección IP local donde se esperan los flujos, IP de la que se esperan y puerto donde se esperan. En las demás tripletas: dirección IP del origen de los datos, IP del destino a donde se envían y su puerto. Esta segunda estructura se puede repetir tantas veces como recolectores haya a los que queramos enviar la información.

Cuadro D.6: Parámetros de *flow-fanout*

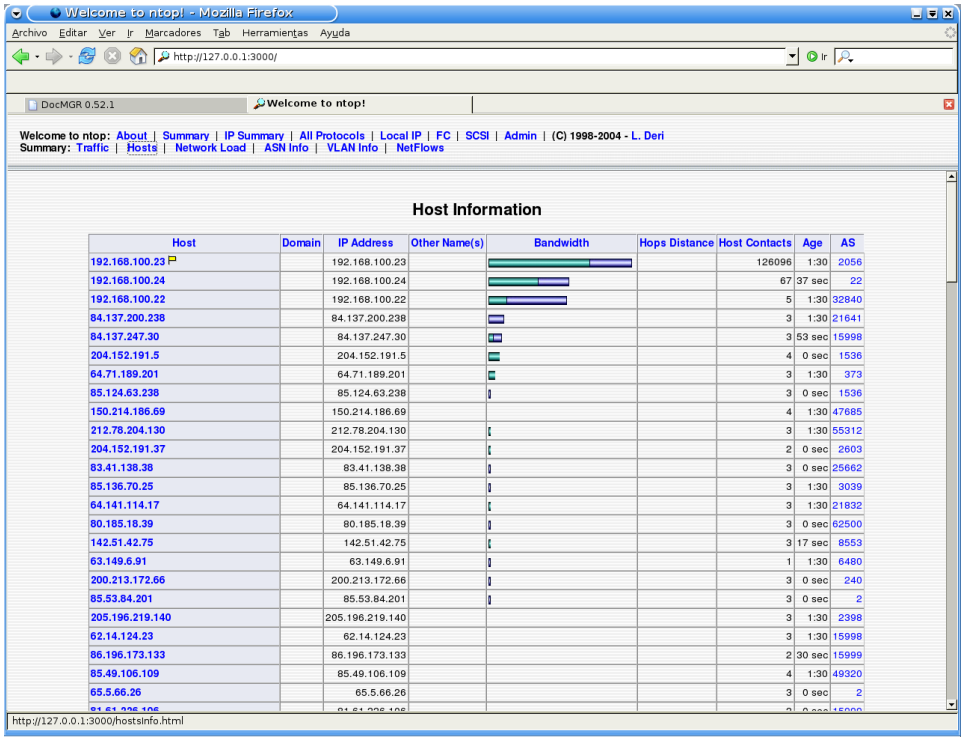


Figura D.3: *ntop* con la información ya exportada

D.1.3. Exportando a una base de datos *MySQL*

Estudiaremos en esta sección la posibilidad de exportar la información almacenada por el recolector a una base de datos *MySQL*. En caso de no disponer de un servidor de bases de datos *MySQL* ya instalado en el sistema se pueden consultar en la sección D.1.3.1, "*Instalando un servidor de base de datos MySQL*", los detalles para realizarlo.

El paquete ***flow-tools*** dispone de la utilidad *flow-export* para la exportación de la información almacenada por el recolector en diversos formatos, desde *ASCII* a bases de datos *MySQL* o *PostgreSQL*. Veamos cuáles son los parámetros de configuración de *flow-export*, indicados en el Cuadro D.7, para comprobar sus posibilidades. Más detalles sobre estos y otros parámetros pueden ser consultados en el manual de la aplicación [22].

Parámetro	**Descripción**
-f `format`	Especifica el formato de salida. Se dispone de: 0 *cflowd* 1 *pcap* 2 *ASCII CSV* 3 *MySQL* 4 *wire* 5 *PGSQ*
-m `mask`	Especifica qué campos serán exportados. Véase el Cuadro D.8.
-u `user:password:host:port:name:table`	Sólo para las exportaciones a *MySQL* o *PGSQ*, indican parámetros para el acceso al servidor de la base de datos. Especifica el usuario de la base de datos, su contraseña, la dirección del servidor, el puerto en el que atiende las solicitudes, el nombre de la base de datos y la tabla en la que se desea almacenar la información.

Cuadro D.7: Parámetros de *flow-export*

La máscara indicará que campos de la información almacenada en *NetFlow* queremos exportar hacia la salida. Para las exportaciones a bases de datos mostramos en el Cuadro D.8 la información de las máscaras, su relación con la información transportada en el protocolo *NetFlow* y el nombre del campo con el que *flow-export* intentaría exportarlos hacia la base de datos.

Máscara	**Valor numérico**	**Campo en *NetFlow* v7**	**Nombre de exportación**	**Notas**
UNIX_SECS	0x000000001	unix_secs	unix_secs	
UNIX_NSECS	0x000000002	unix_nsecs	unix_nsecs	
SYSUPTIME	0x000000004	SysUptime	sysuptime	
EXADDR	0x000000008		exaddr	IP del exportador del flujo.
DFLOWS	0x000000010		dflows	Número de flujos (agregaciones).
DPKTS	0x000000020	dPkts	dpkts	
DOCTETS	0x000000040	dOctets	doctets	
FIRST	0x000000080	First	first	
LAST	0x000000100	Last	last	
ENGINE_TYPE	0x000000200		engine_type	
ENGINE_ID	0x000000400		engine_id	
SRCADDR	0x000001000	srcaddr	srcaddr	
DSTADDR	0x000002000	dstaddr	dstaddr	
SRC_PREFIX	0x000004000			Aparece en el manual, pero no es exportable.
DST_PREFIX	0x000008000			Aparece en el manual, pero no es exportable.
NEXTHOP	0x000010000	nexthop	nexthop	Siempre vale 'cero'.
INPUT	0x000020000	input	input	Siempre vale 'cero'.
OUTPUT	0x000040000	output	output	
SRCPORT	0x000080000	srcport	srcport	
DSTPORT	0x000100000	dstport	dstport	
PROT	0x000200000	prot	prot	
TOS	0x000400000	tos	tos	
TCP_FLAGS	0x000800000	tcp_flags	tcp_flags	Siempre vale 'cero'.
SRC_MASK	0x001000000	src_mask	src_mask	Siempre vale 'cero'.
DST_MASK	0x002000000	dst_mask	dst_mask	Siempre vale 'cero'.
SRC_AS	0x004000000	src_as	src_as	Siempre vale 'cero'.
DST_AS	0x008000000	dst_as	dst_as	Siempre vale 'cero'.
IN_ENCAPS	0x010000000		in_encaps	Sólo *NetFlow* version 6. Tamaño de la encapsulación (entrada).
OUT_ENCAPS	0x020000000		out_encaps	Sólo *NetFlow* version 6. Tamaño de la encapsulación (salida).
PEER_NEXTHOP	0x040000000		peer_nexthop	
ROUTER_SC	0x080000000	router_sc	router_sc	
EXTRA_PKTS	0x100000000		extra_pkts	
MARKED_TOS	0x200000000		marked_tos	

Cuadro D.8: Relación de la exportación de *flow-export* con los campos de *NetFlow* V7

Puede verse que los campos del protocolo *NetFlow* que faltan en esta lista son: *version*, *count*, *flow-sequence*, *reserved*, *flags (x2)*. Estos campos coinciden en que no tienen significación fuera del envío de la información de la sonda al recolector, es decir, son información sobre el propio protocolo o sobre el envío de la información y ya no tienen sentido fuera de ese contexto. Así mismo, *flow-export* puede introducir información adicional a *NetFlow*, así como lo hacía *flow-print* (aunque no lo resaltamos en el capítulo D.1.1, "*Presentación mediante consola*") como la dirección de la sonda (*exaddr*).

Con esta información y podemos escribir la estructura de una base de datos *MySQL* [62, 63] que será a la cual enviaremos la información capturada. Para la siguiente estructura hemos omitido los campos que *NetFlow* siempre transportará con valor 'cero':

```
CREATE TABLE `tabla1` (
`sysuptime` int(15) NOT NULL default '0',
`srcaddr` varchar(15) NOT NULL default '',
`srcport` smallint(6) NOT NULL default '0',
`dstaddr` varchar(15) NOT NULL default '',
`dstport` smallint(6) NOT NULL default '0',
`prot` smallint(2) NOT NULL default '0',
`tos` smallint(6) NOT NULL default '0',
`unix_secs` int(15) NOT NULL default '0',
`unix_nsecs` int(15) NOT NULL default '0',
`dPkts` smallint(10) NOT NULL default '0',
`dOctets` int(15) NOT NULL default '0',
`first` int(15) NOT NULL default '0',
`last` int(15) NOT NULL default '0',
) TYPE=MyISAM;
```

Para mostrar como se exporta la información a una base de datos *MySQL* llamada *flows* con esta estructura se utilizarían indistintamente los siguientes comandos, donde el usuario de la base de datos sería *flow* y no tendría contraseña:

```
#> flow-cat /var/flow/eth0 | flow-export -f3 -mUNIX_SECS,UNIX_NSECS,SYSUPTIME,
     DPKTS,DOCTETS,FIRST,LAST,SRCADDR,DSTADDR,SRCPORT,DSTPORT,PROT,TOS -u flow::
     localhost:3306:flows:tabla1

#> flow-cat /var/flow/eth0 | flow-export -f3 -m0x7831EF -u flow::localhost:3306:
     flows:tabla1
```

Dicho usuario *flow* podría crearse con los siguientes comandos de *SQL*:

```
INSERT INTO user (host,user,password) VALUES('localhost','flow',PASSWORD(''));
GRANT SELECT , INSERT , UPDATE , DELETE ON `flows`.* TO 'flow'@'localhost';
FLUSH PRIVILEGES;
```

Por último, al basarse la exportación que *flow-export* realiza en una adquisición de datos a través de *flow-cat*, es posible realizar el mismo tratamiento de los datos que se hacía hacia *ntop* como se indicó en la sección D.1.2, "*Exportando a ntop*".

D.1.3.1. Instalando un servidor de base de datos *MySQL*

Cubrimos en este punto la instalación y configuración de un servidor de base de datos tipo *MySQL* para nuestra distribución **Debian Sarge**.

En primer lugar obtendremos los paquetes necesarios:

```
#> apt-get install mysql-server-4.1
```

Esto nos instalará el paquete ***mysql-server-4.1*** en el sistema junto a una extensa lista de paquetes que son dependencias del mismo.

El servidor *MySQL* se instala pro defecto sin contraseña para el usuario `root`. Para establecer una contraseña para dicho usuario podemos ejecutar la siguiente orden:

```
#> mysqladmin -u root password NUEVO_PASSWORD
```

donde «NUEVO_PASSWORD» será la nueva contraseña que queremos establecer. «password» sería en circunstancias normales la contraseña actual del usuario `root` de *MySQL*, pero que en este caso no está establecida y acepta este valor. Podemos comprobar que hemos establecido la contraseña deseada como sigue:

```
#> mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8 to server version: 4.0.24_Debian-10sarge2-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> exit
Bye

#>
```

La configuración del servidor reside en `/etc/mysql/my.cnf`. Dicho archivo está bien comentado y es autoexplicativo. Nosotros, en principio, no necesitamos realizar ninguna modificación más a dicha configuración.

D.1.4. Exportando a bases de datos *Round Robin*

Estudiaremos en esta sección la posibilidad de exportar la información almacenada por el recolector a una base de datos *Round Robin*. Las bases de datos *Round Robin* o *RRD* tienen la ventaja, como se comentó en la sección 7.1.3, "*Exportando a una base de datos*", de tener tamaño fijo por lo que tendremos menos problemas de almacenamiento en disco que con las bases de datos *MySQL*.

Existen una serie de herramientas que podemos configurar para obtener gráficas del tráfico que atraviesa nuestros sistemas remotos una vez que hemos recibido la información mediante *NetFlow* procedente desde nuestra sonda *fprobe-ng* y se ha procedido a su captura con *flow-capture*. Esta herramienta es *FlowScan* [58, 64, 65], a la que añadiremos dos complementos, *CUFlow* y *CUGrapher* [66], para poder fabricar las gráficas de uso de la red. Todas estas herramientas se presentan en paquetes para **Debian Sarge** en versión *testing*, y en los siguientes puntos veremos cómo proceder a la instalación y configuración de estas utilidades.

D.1.4.1. Obteniendo los paquetes

Para empezar necesitaremos cambiar temporalmente nuestra distribución **Debian Sarge** de *stable* a *testing* para lo que será necesario modificar el archivo `/etc/apt/sources.list`:

`/etc/apt/sources.list` para versión *testing*

```
# Repositorios para version "stable"
#deb http://ftp.es.debian.org/debian/ stable main
#deb-src http://ftp.es.debian.org/debian/ stable main

#deb http://ftp.funet.fi/pub/linux/mirrors/debian/ stable main
#deb-src http://ftp.funet.fi/pub/linux/mirrors/debian/ stable main

#deb http://security.debian.org/ stable/updates main

# Repositorios para version "testing"
deb http://ftp.es.debian.org/debian/ testing main
deb-src http://ftp.es.debian.org/debian/ testing main
```

Hecho esto, podemos pasar a usar *apt* para obtener los paquetes:

```
#> apt-get update
...

#> apt-get install flowscan flowscan-cuflow flowscan-cugrahper
```

Los contenidos de los paquetes se podrán consultar con las siguientes instrucciones:

```
#> dpkg -L flowscan
```

```
#> dpkg -L flowscan-cuflow
```

```
#> dpkg -L flowscan-cugrapher
```

D.1.4.2. Configurando *FlowScan*

Las *scripts* en Perl que forman *FlowScan* se almacenarán en `/usr/share/perl5` y las configuraciones de las herramientas en `/etc/flowscan`. Comenzaremos viendo la configuración de *FlowScan* ubicada en `/etc/flowscan/flowscan.cf`:

1. La directiva `FlowFileGlob` indica donde está el directorio que contiene los archivos a procesar. *FlowScan* no puede leer la estructura de directorios creada por *flow-capture* así que es necesario cambiar el parámetro `nesting_level` de la configuración de *flow-capture* al valor 0 para que no cree subdirectorios, como se indicó en el Cuadro A.4. Nosotros vamos a suponer que el directorio es `/var/flow/eth0`:

`/etc/flowscan/flowscan.cf` (I)

```
# flowscan Configuration Directives
    ##########################################

# FlowFileGlob (REQUIRED)
# use this glob (file pattern match) when looking for raw flow files to be
# processed, e.g.:
# FlowFileGlob /var/local/flows/flows.*:*[0-9]
# FlowFileGlob flows.*:*[0-9]
FlowFileGlob /var/flow/eth0/ft-v*[0-9]
```

Es importante decir que *FlowScan* una vez ha analizado un archivo lo elimina, a no ser que exista un directorio llamado `saved` en el directorio donde se encuentran los archivos (en este caso sería `/var/flow/eth0/saved`). Si

existe dicho directorio *FlowScan* moverá ahí los archivos ya procesados, y este directorio estará sujeto además al posible control espacio en disco (de número de archivos o de tamaño) establecido en la configuración del recolector (ver el Cuadro A.4 en la sección A.3.2, "*Configuración de flow-capture*" para más información acerca del control de espacio en disco). Esto es muy útil para la depuración de errores ya que evita que perdamos los flujos capturados.

Así mismo, si queremos que procese más de un directorio lo que debemos hacer es escribir la lista de directorios a utilizar dentro de la directiva `FlowFileGlob`.

`/etc/flowscan/flowscan.cf` (Ejemplo para varios directorios)

```
# flowscan Configuration Directives
    ##########################################

# FlowFileGlob (REQUIRED)
# use this glob (file pattern match) when looking for raw flow files to be
# processed, e.g.:
# FlowFileGlob /var/local/flows/flows.*:*[0-9]
# FlowFileGlob flows.*:*[0-9]
FlowFileGlob /var/flow/eth0/ft-v*[0-9] /var/flow/eth1/ft-v*[0-9]
```

2. Lo siguiente es indicar a *FlowScan* que módulo se usará para el estudio de los archivos con la directiva `ReportClasses`. Nosotros usaremos `CUFLow`, que es el módulo que hemos añadido con los paquetes ***flowscan-cuflow*** y ***flowscan-cugrahper***:

`/etc/flowscan/flowscan.cf` (II)

```
# ReportClasses (REQUIRED)
# a comma-seperated list of FlowScan report classes, e.g.:
# ReportClasses CampusIO
# ReportClasses SubNetIO
ReportClasses CUFlow
```

Al igual que ocurre con la directiva `FlowFileGlob`, si deseamos que se trate la información capturada por varios módulos bastará escribir la lista de módulos deseados en la directiva `ReportClasses`. No obstante, estos módulos deberán ser compatibles entre sí.

3. Lo siguiente es indicar a *FlowScan* cuanto tiempo deseamos que pase entre dos estudios consecutivos del directorio que contiene los archivos, mediante la directiva `WaitSeconds`:

`/etc/flowscan/flowscan.cf` (III)

```

# WaitSeconds (OPTIONAL)
# This should be <= the "-s" value passed on the command-line to cflowd, e
    .g.:
# WaitSeconds 300
WaitSeconds 30
```

En períodos de prueba, para agilizar el tráfico, es conveniente reconfigurar *flow-capture* para que grabe los archivos a un ritmo mayor de uno cada 15 minutos (su configuración por defecto) mediante el parámetro `rotations` que se vió en el Cuadro A.4. Un valor de `1439` hará que se creen cada minuto (el ritmo más rápido posible).

4. Por último, indicar si queremos que *FlowScan* nos muestre una salida detallada de su comportamiento:

`/etc/flowscan/flowscan.cf` (y IV)

```
20
    # Verbose (OPTIONAL, non-zero = true)
    Verbose 1
```

Darle valor «1» a `Verbose` mostrará la salida detallada. Es necesario advertir que este valor es muy práctico para la depuración y comprobación del buen funcionamiento de la utilidad, pero su uso, sobre todo si se redirige dicha salida a un fichero, generará archivos de gran tamaño en nuestro sistema.

Así mismo hemos querido desarrollar una *script* que permita el manejo de *FlowScan* como si de otro servicio o demonio del sistema se tratase. A continuación presentamos dicha *script*:

`/etc/init.d/flowscan` (*script* de arranque)

```
    #!/bin/sh
    # description: Start FlowScan


 5  start(){
        FLOWSCAN_PID=`ps axjf | grep -v axjf | grep -v grep | grep "/usr/bin/
            flowscan" | sort -k 9 | tail -n 1 | awk '{print $2}'`

        if [ "$FLOWSCAN_PID" == "" ]
        then
10          /usr/bin/flowscan >>/var/log/flowscan 2>&1 </dev/null & >/dev/
                null
            #touch /var/lock/flowscan.1
            sleep 1
            FLOWSCAN_PID=`ps axjf | grep -v axjf | grep -v grep | grep "/usr
                /bin/flowscan" | sort -k 9 | tail -n 1 | awk '{print $2}'`
            echo "Starting flowscan service... "
15
            if [ "$FLOWSCAN_PID" == "" ]
            then
                echo "ERROR: flowscan is not running."
            else
20              echo "Started with PID "$FLOWSCAN_PID
            fi

        else
            echo "ERROR: flowscan is already running."
25      fi
        echo
    }

    stop(){
30      FLOWSCAN_PID=`ps axjf | grep -v axjf | grep -v grep | grep "/usr/bin/
            flowscan" | sort -k 9 | tail -n 1 | awk '{print $2}'`

        if [ "$FLOWSCAN_PID" == "" ]
        then
            echo "ERROR: flowscan is not running."
35      else
            kill $FLOWSCAN_PID
            #rm -f /var/lock/flowscan.1
            echo "Stopping flowscan service... Found PID "$FLOWSCAN_PID",
                killing it"
        fi
40      echo
    }
```

```
    case "$1" in
    start)
45          start

    ;;
    stop)
            stop
50  ;;

    restart)
            stop
            start
55  ;;
    *)
            echo "Usage: $0 { start | stop | restart}"
    ;;
    esac
60  exit 0
```

La *script* es simple: dado que *FlowScan* no genera archivos PID durante su ejecución, hemos tenido que recurrir a comprobar si el servicio está activo mediante *ps*. La *script* permite las habituales llamadas *start*, *stop* y *restart* de los demonios del sistema. La emplazaremos en el archivo `/etc/init.d/flowscan`, y podremos hacer que se inicie en el arranque del sistema y se detenga en su apagado con la siguiente orden:

```
#> update-rc.d flowscan defaults 25
 Adding system startup for /etc/init.d/flowscan ...
   /etc/rc0.d/K25flowscan -> ../init.d/flowscan
   /etc/rc1.d/K25flowscan -> ../init.d/flowscan
   /etc/rc6.d/K25flowscan -> ../init.d/flowscan
   /etc/rc2.d/S25flowscan -> ../init.d/flowscan
   /etc/rc3.d/S25flowscan -> ../init.d/flowscan
   /etc/rc4.d/S25flowscan -> ../init.d/flowscan
   /etc/rc5.d/S25flowscan -> ../init.d/flowscan
```

Si deseamos eliminarlo del arranque y detención del sistema basta ejecutar:

```
#> update-rc.d -f flowscan remove
update-rc.d: /etc/init.d/flowscan exists during rc.d purge (continuing)
 Removing any system startup links for /etc/init.d/flowscan ...
   /etc/rc0.d/K25flowscan
   /etc/rc1.d/K25flowscan
   /etc/rc2.d/S25flowscan
   /etc/rc3.d/S25flowscan
   /etc/rc4.d/S25flowscan
   /etc/rc5.d/S25flowscan
   /etc/rc6.d/K25flowscan
```

Por último, en dicha *script* se indica al programa que dirija su salida hacia el archivo ubicado en `/var/log/flowscan` en caso de que estemos haciendo uso de la salida detallada.

D.1.4.3. Configurando *CUFlow*

Una vez configurado *FlowScan* configuraremos *CUFlow*, el módulo que usaremos para analizar los archivos. Su configuración reside en `/etc/flowscan/CUFlow.cf`:

1. Lo primero que *CUFlow* nos permite es indicar las subredes que componen nuestra red:

`/etc/flowscan/CUFlow.cf` (I)

```
    # These are the subnets in our network
```

```
    # These are used only to determine whether a packet is inbound our
    # outbound
    # Subnet 10.0.0.0/16
    Subnet 172.26.0.0/23
```

2. Con la directiva `Network` indicamos las redes de las que queremos tener luego información separada:

`/etc/flowscan/CUFlow.cf` (II)

```
    # These are networks we are particularly interested in, and want to
    # get separate rrd's for their aggregate traffic
    # Network 10.0.1.0/24 routers
    Network 172.26.0.0/24 backbone
    Network 172.26.1.0/24 routers
```

3. La siguiente directiva es la que indica a *CUFlow* el directorio donde se almacenarán los archivos *RRD* que contendrán sus resultados:

`/etc/flowscan/CUFlow.cf` (III)

```
    # Where to put the rrd's
    # Make sure this is the same as $rrddir in CUGrapher.pl
    # OutputDir /cflow/reports/rrds
    OutputDir /var/CUFlow/rrds
```

   Este directorio también hay que introducirlo dentro del archivo `/etc/flowscan/CUGrapher.cf`, como se indicará posteriormente. No lo indica la documentación, pero se ha encontrado que es necesario crear también un directorio que cuelgue del padre del especificado con `OutputDir` y que se llame `scoreboard` para el correcto funcionamiento del programa. En este ejemplo concreto habrá que crear entonces los directorios `/var/CUFlow/rrds` y `/var/CUFlow/scoreboard`.

4. Las directivas `Scoreboard` y `AggregateScore` permiten realizar una salida en formato HTML de un estudio de los usuarios más consumidores de ancho de banda. La primera creará el estudio de los más consumidores del último período estudiado y la segunda realizará un histórico. Las salidas deberán ser emplazadas dentro de un dirctorio válido para ser servido por el servidor *Web* instalado en el sistema:

`/etc/flowscan/CUFlow.cf` (IV)

```

    # Keep top N lists
    # Show the top ten talkers, storing reports in /cflow/flows/reports
    # and keeping the current report in /etc/httpd/data/reports/topten.html
    # Scoreboard 10 /cflow/reports/scoreboard /var/www/html/topten.html
    Scoreboard 10 /var/CUFlow/scoreboard /var/www/topten.html

    # Same, but build an over-time average top N list
    # AggregateScore 10 /cflow/reports/scoreboard/agg.dat /var/www/html/
        overall.html
    AggregateScore 10 /var/CUFlow/scoreboard/agg.dat /var/www/overall.html
```

5. El parámetro `Router` permite diferenciar a los exportadores, nuestras sondas, de los archivos que están siendo analizados:

`/etc/flowscan/CUFlow.cf` (y V)

```
30
    # Our two netflow exporters. Produce service and protocol reports for the
    # total, and each of these.
    #Router 10.0.1.1 router1
    #Router 10.0.1.2 router2
35  Router 172.26.0.1 routerSalida
```

6. Se incluyen más parámetros en la configuración que ya no necesitamos configurar, pero cuyo uso puede ser interesante para establecer definir los protocolos que se analizarán, los servicios, los *AS* que diferenciamos, etc...

Configurado ya *CUFlow*, *FlowScan* es ya capaz de analizar los archivos que hemos generado anteriormente con *flow-capture* y nos generaría las bases de datos *RRD* además de las salidas de `Scoreboard` y `AggregateScore`, como se muestra en la Figura D.4. Para ello tan sólo tenemos que ejecutar la herramienta *FlowScan*:

```
#> flowscan
...
```

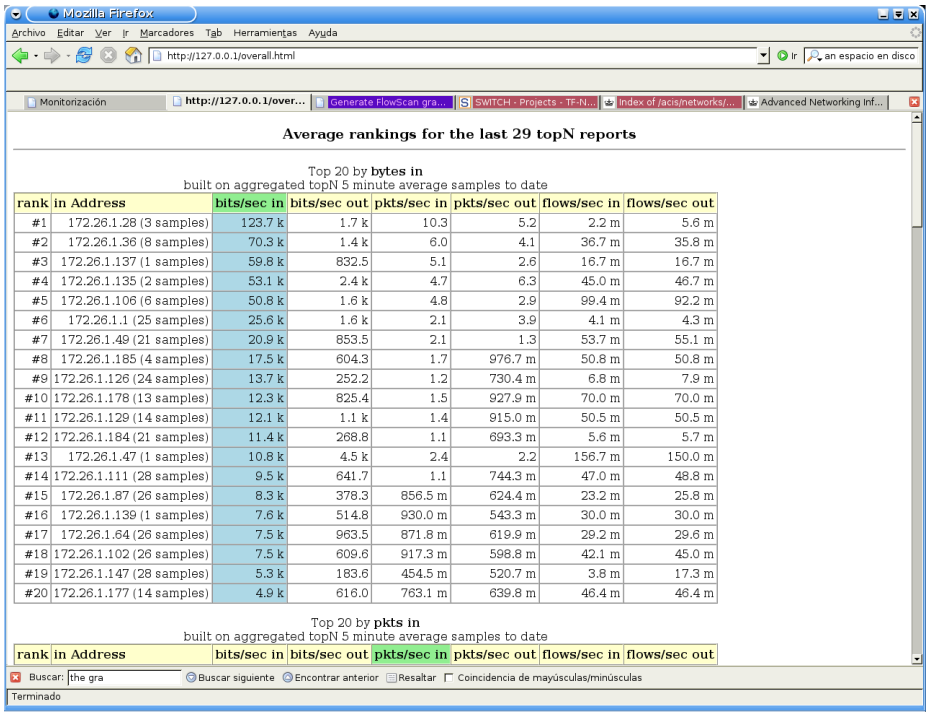


Figura D.4: `Scoreboard` de *CUFlow*

D.1.4.4. Configurando *CUGrapher*

Ahora tan solo nos falta configurar la parte gráfica de las herramientas, es decir, la *script* en Perl *CUGrapher* que viene incluída dentro del paquete ***flowscan-cugrapher***. Para ello sólo hay que hacer algunas modificaciones a su

archivo de configuracion, `/etc/flowscan/CUGrapher.cf`, para hacerla funcionar:

1. Es necesario indicarle a la *script* donde se encuentran los archivos *RRD*:

`/etc/flowscan/CUGrapher.cf` (I)

```
# Configuration file for CUGrapger.pl.
#

# Where to find the rrd's.  You must set this.
# Make sure it is the same as OutputDir in CUFlow.cf.
#
# Eg:       OutputDir /cflow/reports/rrds
# Default: not set
OutputDir /var/CUFlow/rrds
```

   Esta ruta será la misma que la que se especificó en la directiva `OutputDir` de *CUFlow*.

2. Los demás parámetros son autoexplicativos y fácilmente configurables. Pueden presentar un mayor interés las directivas `Organization` y `Title` para la representación de nuestras gráficas.

`/etc/flowscan/CUGrapher.cf` (y II)

```

# Organisation name - appears in the graph title.
#
# Eg:       Organization Columbia University Campus
# Default: (blank)
Organization Escuela Superior de Ingenieros - CUFlow

# Default number of hours to go back.
#
# Eg:       Hours 24
# Default: Hours 48

# Default width of graph in pixels.
#
# Eg:       Width 800
# Default: Width 640

# Default height of graph in pixels.
#
# Eg:       Height 480
# Default: Height 320

# Default image type.  Can be png or gif.
#
# Eg:       ImageType gif
# Default: ImageType png

# Default graph title.
#
# Eg:       Title My Graph
# Default: Title Well Known Protocols/Services

#
# Specify a graph to be displayed on startup, when no query
# has been entered.  To display multiple graphs supply
# multiple "DefaultGraph" lines.  To generate the string
# following the "DefaultGraph":
#  1.  Use the web page to generate the graph you want.
#  2.  Copy the query part of the URL displayed by your browser
#       (ie everything part the '?').
#  3.  Remove the ';showmenu=1' from the query string copied.
#
```

```
    # Eg:       DefaultGraph report=bits;hours=48;imageType=png;width=640;
        height=320;duration=;router=all;all_all_services=1;legend=1;title=My
        %20Graph
    # Default: not set
55
    # The path to the AggregateScore web page build by CUFlow.  If
    # non-blank a like to AggregateScore web page will be displayed.
    # If supplied this must be the same as the file name given to
    # the AggregateScore setting in CUFlow.cf.
60  #
    # Eg:       AggregateScore /var/local/netflow/cuflow/agg10.html
    # Default: (not set)

    # The path to the Scoreboard web page built by CUFlow.  If
65  # non-blank a link to the Scoreboard web page will be
    # displayed.  If supplied this must be the same as the file
    # name given to the Scoreboard setting in CUFlow.cf.
    #
    # Eg:       Scoreboard /var/local/netflow/cuflow/top10.html
70  # Default: (not set)
```

Tras ello, solo nos queda ejecutar la *script* `CUGrapher.cgi` que está emplazada dentro de los directorios de `cgi-bin` de nuestro servidor *Web*. Para ello tendríamos que introducir en nuestro navegador *Web* una dirección como la siguiente: http://127.0.0.1/cgi-bin/CUGrahper.cgi, y obtendríamos un resultado como el mostrado en la Figura D.5:

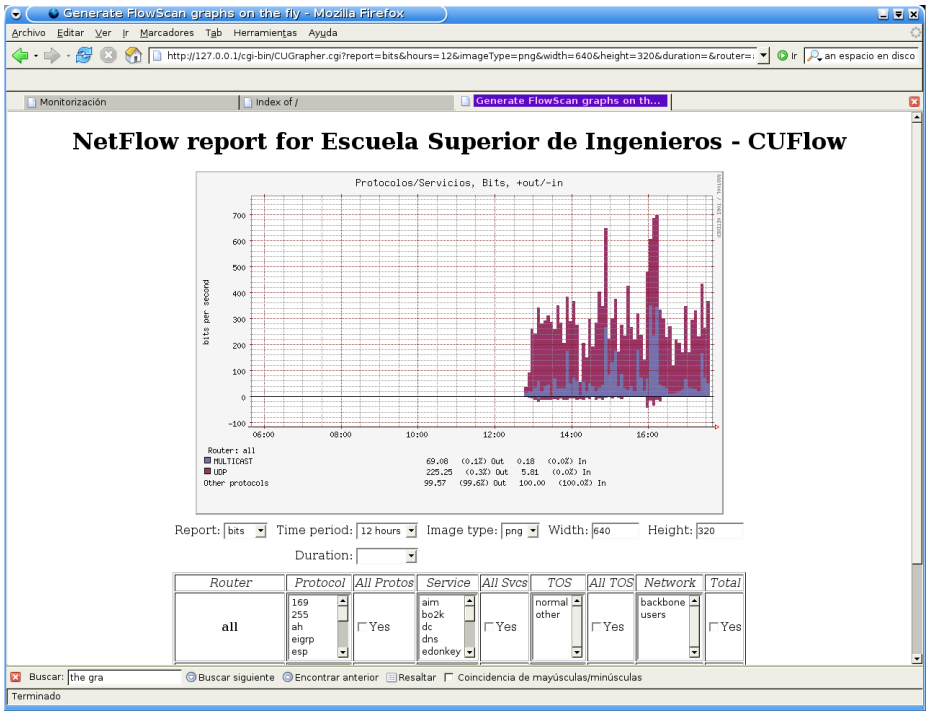


Figura D.5: Presentación de *CUGrapher*

D.1.4.5. Instalando y configurando *FlowMonitor*

Además de los módulos anteriores también se ha estudiado otro complemento más de *FlowScan* llamado *FlowMonitor* [67] que permite realizar otro tipo de informes de consumo identificando a los usuarios y presentándolos diferenciados. Este módulo nos permite establecer unos umbrales que al ser sobrepasados harán que los usuarios que los sobrepasen se nos presenten en una lista. Sin embargo, a pesar de ser un complemento de la utilidad *FlowScan*, *FlowMonitor* necesita de una base de datos *MySQL* para su funcionamiento pero hemos querido tratarla aquí dada su dependencia de la utilidad *FlowScan*. ***FlowMonitor*** no se encuentra disponible en *paquete Debian*, pero podemos obtenerlo de la siguiente forma:

```
#> cd /usr/src

#> wget http://www.columbia.edu/acis/networks/advanced/FlowMonitor/FlowMonitor
    -1.2.tar.gz
...

#> tar -xvjf FlowMonitor-1.2.tar.gz
```

Esto nos dejará los siguientes archivos:

```
/usr/src/FlowMonitor-1.2/COPYING
/usr/src/FlowMonitor-1.2/FlowMonitor.cf
/usr/src/FlowMonitor-1.2/FlowMonitor.pm
/usr/src/FlowMonitor-1.2/README.txt
```

Para hacer el comportamiento de *FlowMonitor* similar al de los otros módulos de *FlowScan*, emplazaremos su archivo de configuración y su *script* en las mismas ubicaciones:

```
#> cp /usr/src/FlowMonitor-1.2/FlowMonitor.cf /etc/flowscan/FlowMonitor.cf

#> cp /usr/src/FlowMonitor-1.2/FlowMonitor.pm /usr/share/perl5/FlowMonitor.pm
```

Para conseguir esto, necesitamos cambiar la ubicación del archivo de configuración dentro de `FlowMonitor.pm` para que la localice correctamente:

`/usr/share/perl5/FlowMonitor.pm` (modificado)

```
    $ENV{'ORACLE_HOME'} = "/opt/oracle-9.2" unless defined $ENV{'ORACLE_HOME'};
260 #&parseConfig("${FindBin::Bin}/FlowMonitor.cf"); # Read our config file
    &parseConfig("/etc/${FindBin::Script}/FlowMonitor.cf");      # Read our config
        file
```

Así mismo, deberemos modificar la configuración de *FlowScan*, emplazada en `/etc/flowscan/flowscan.cf`, para que haga uso de este módulo:

`/etc/flowscan/flowscan.cf` (modificado)

```
 10 # ReportClasses (REQUIRED)
    # a comma-seperated list of FlowScan report classes, e.g.:
    # ReportClasses CampusIO
    # ReportClasses SubNetIO
    # ReportClasses CUFlow
 15 ReportClasses FlowMonitor
```

FlowMonitor y *CUFlow* no son compatibles entre sí.

FlowMonitor necesita una base de datos para almacenar su información temporal, y por defecto querría usar una base de datos *Oracle* para ello. Nosotros

cambiaremos la configuración para reutilizar la base de datos *MySQL* que ya utilizamos anteriormente en el punto D.1.3, "*Exportando a una base de datos MySQL*". Primero necesitaremos modificar el archivo `/usr/share/perl5/FlowMonitor.pm` para indicar el nombre del usuario de la base de datos y su clave, que en este caso será el usuario *flow* y no tendrá clave:

`/usr/share/perl5/FlowMonitor.pm` (modificado)

```
     my($SUBNETS);                  # A trie of internal subnets. IPs we
                                    # should keep data on.
     my(%ROUTERS);                  # A hash of ip's we should
                                    # police records from
225  my($INUSAGE);                  # A trie for recording the usage of IPs
                                    # we find in flow records for inbound data
     my($OUTUSAGE);                 # A trie for recording the usage of IPs
                                    # we find in flow records for inbound data
     my($IGNORE);                   # A trie of ips we should not list
230  my($QUOTA);                    # Bytes allowed in a given INTERVAL
     my($INTERVAL);                 # Time between resets of an IPs counter, in
                                    # units of seconds
     my(%VIOLATORS);                # Hash keyed by %age of policy, whose value
                                    # is a filename to write IPs to
235  my($DBNAME) = 'dbi:';          # Which DBI module to use
     my($DBUSER) = 'flow';              # What username to connect as
     my($DBPASS) = '';              # What password to connect with

     my($SYSDATE) = '';             # What to select to get the current time
240  my($DATEUNITS) = '';           # Number of seconds in the units $SYSDATE
                                    # math is done in. Ie, if $SYSDATE - 1 is
                                    # 1 day before sysdate, then DATEUNITS will
                                    # be 86400 (seconds per day)
```

Para crear el usuario de *MySQL* adecuado y sin contraseña podremos usar los mismos comandos de *SQL* vistos en la sección D.1.3, "*Exportando a una base de datos MySQL*".

Únicamente falta adaptar la configuración de *FlowMonitor* a nuestro sistema. El archivo de configuración se encuentra en `/etc/flowscan/FlowMonitor.cf`:

1. *FlowMonitor* nos permite primero establecer una serie de subredes y exportadores de *NetFlow*:

`/etc/flowscan/FlowMonitor.cf` (I)

```
     # Example

     #Subnet 160.39.0.0/16
     #Subnet 192.5.43.0/24
  5  #Subnet 67.99.58.192/30
     #Subnet 128.59.0.0/16
     #Subnet 156.111.0.0/16
     #Subnet 129.236.0.0/16
     #Subnet 156.145.0.0/16
 10  #Subnet 207.10.136.0/21
     #Subnet 209.2.47.0/24
     #Subnet 209.2.48.0/22
     #Subnet 209.2.185.0/24
     #Subnet 209.2.208.0/20
 15  #Subnet 209.2.224.0/20
     Subnet 172.26.0.0/24
     Subnet 172.26.1.0/24

     #Router 128.59.1.4
 20  Router 172.26.0.1
```

2. La directiva `Carry-Forward` permite tener un mejor seguimiento del consumo de los usuarios. Si tenemos el limite establecido a 100 MegaBytes por hora y un usuario consume 120 MegaBytes, con esta directiva activada la siguiente hora comenzará con 20 MegaBytes en su cuenta en lugar de con 0:

`/etc/flowscan/FlowMonitor.cf` (II)

```
20  Router 172.26.0.1

    Carry-Forward
```

3. Lo siguiente es la definición de la política (`Policy`) del límite de consumo. Soporta tres métodos distintos de contabilidad según se especifique `Inbound` (donde sólo se contabiliza el tráfico que tiene como destino un elemento contenido dentro de una definición previa de `Subnet`), `Outbound` (donde se contabilizará el tráfico originado por un elemento definido en una `Subnet`) o `Both` (contabilizará el tráfico con origen o destino comprendido en las definiciones de `Subnet`):

`/etc/flowscan/FlowMonitor.cf` (III)

```

    # 100 MB / 1 hours
25  Policy Outbound 100000000 3600
    # 60 MB / 1 hours
    #Policy Outbound 60000000 3600
    # 200 MB / 1 hours
    #Policy 200000000 3600
30  # 4 GB / 24 hours
    #Policy 4320000000 86400
    # 8 GB / 24 hours
    #Policy 8640000000 86400
    # 2 MB / 10 minutes
35  #Policy 2000000 600
```

4. Con la directiva `DBName` especificamos la base de datos a utilizar. En lugar de usar una base de datos Oracle utilizaremos una *MySQL*, concretamente una llamada *flows*:

`/etc/flowscan/FlowMonitor.cf` (IV)

```
35  #Policy 2000000 600

    #DBName Oracle oradb
    DBName mysql flows
```

FlowMonitor no es especialmente flexible en estos aspectos. Dentro de la base de datos ***flows*** necesariamente debe existir una tabla llamada *iplogs* con el siguiente formato:

```
CREATE TABLE `iplogs` (
        `ipaddr` VARCHAR( 16 ) NOT NULL ,
        `bytes` INT( 16 ) NOT NULL ,
        `starttime` DATETIME NOT NULL ,
        UNIQUE (`ipaddr`)
) TYPE = MyISAM ;
```

5. La directiva `Violators` es utilizada para marcar a los usuarios que sobrepasan un cierto nivel dentro de la política establecida. El formato es fácil de entender con la siguiente configuración:

`/etc/flowscan/FlowMonitor.cf` (V)

```
DBName mysql flows

# Violators 0 /var/www/html/violators-0.txt

# Every user gets logged to 0percenters
Violators 0 /var/www/0percenters
# People at 50% over policy get logged to 50percenters
Violators 50 /var/www/50percenters
# People at 100% over policy get logged to 100percenters
Violators 100 /var/www/100percenters
# People at 200% or more over policy go to losers
Violators 200 /var/www/losers
```

Un usuario solo aparecerá en el límite más alto que haya sobrepasado, es decir, si un usuario sobrepasa el 100 % de la política solo aparecerá en `100percenters` en el ejemplo anterior.

6. Con la directiva `Logfile` podemos llevar un registro de qué usuario estuvo en qué archivo a qué hora:

`/etc/flowscan/FlowMonitor.cf` (VI)

```
Violators 200 /var/www/losers

#Logfile /var/www/html/violator-history.txt
Logfile /var/www/violator-history.txt
```

7. Por último, las directivas `Ignore` e `IgnoreList` permiten, respectivamente, establecer una serie de bloques de red (especificados en notación CIDR) a los que no se contemplará para la lista de `Violators` y exportar dicha lista a un archivo de texto.

`/etc/flowscan/FlowMonitor.cf` (VII)

```
IgnoreList /var/www/html/ignorelist.txt

# Test ignore
Ignore 128.59.1.1/32

# Ignore Machine room
Ignore 128.59.59.0/24
```

Ahora, al ejecutar la herramienta *FlowScan* se crearán los archivos de violaciones en los lugares que hemos especificado al proceder al análisis de los archivos que contienen las capturas de *NetFlow*.

D.2. Tratamiento de la información de los registros

D.2.1. Visualización en consola y página *Web*: *ccze*

ccze es un simple visualizador de registros. *ccze* está preconfigurado para mostrar una serie de registros de distintos programas, como *fetchmail*, *proftpd*, *squid*, o el que a nosotros nos interesa, *syslog-ng* [68].

ccze no hace ningún tipo de análisis o resumen de los contenidos del archivo del registro sino que lo muestra tal y como es pero aplicándole una *colorización* por defecto. Esto tiene la utilidad que, junto a su capacidad de volcar sus resultados a una consola o terminal o hacia una página *Web* que se actualizarán de forma automática, permitirán a un administrador del sistema estar al tanto de los contenidos de los registros sin tener que esperar al resumen que otros programas hagan de forma periódica.

ccze se proporciona como *paquete Debian* en su versión 0.2.1-1, y su instalación es rápida y sencilla:

```
#> apt-get install ccze
```

Se podrán consultar los contenidos del paquete de la siguiente forma:

```
#> dpkg -L ccze
```

Como hemos comentado, *ccze* está en su mayor parte preconfigurado y no permite realizar modificaciones importantes sobre su funcionamiento. El único archivo de configuración modificable es `/etc/cczerc`. Dicho archivo sólo nos permite modificar la colorización que cada uno de los campos del registro sufre una vez reconocido. No nos deja, por ejemplo, realizar una colorización en función del *host* que emite el mensaje, una característica que sería útil y que echamos en falta. Lo que sí nos permite *ccze* es alguna opción en su ejecución, como se muestra en el Cuadro D.9. Más información acerca de sus parámetros puede encontrarse en su manual [69].

Parámetro	**Descripción**
-A	Salida en formato *raw ANSI*
-h	Salida en formato HTML

Cuadro D.9: Parámetros de *ccze*

Su puesta en funcionamiento será la siguiente: configuraremos a *syslog-ng* para que una nueva salida de registro sea el propio *ccze*, volcando sus resultados a distintas salidas:

`/etc/syslog-ng/syslog-ng.cfg` (líneas añadidas)

```
destination df_ccze {
        program("ccze > /dev/tty8");
};

destination df_cczeweb {
```

```
        program("ccze -h > /var/www/ccze.html");
};

log {
        source(s_all);
        filter(f_syslog);
        destination(df_ccze);
};

log {
        source(s_all);
        filter(f_syslog);
        destination(df_cczeweb);
};
```

Con esta configuración, hacemos que *syslog-ng* filtre los mensajes y los mande a *ccze* de dos formas distintas: la primera los mandará al terminal número 8 y la segunda a un archivo de formato HTML. Suponemos que la ruta `/var/www` está dentro de la ruta habilitada para el servidor *apache*. Se adjuntan a continuación dos imágenes en las que se puede ver el funcionamiento de *ccze* tanto en modo consola como exportando a página *Web*:

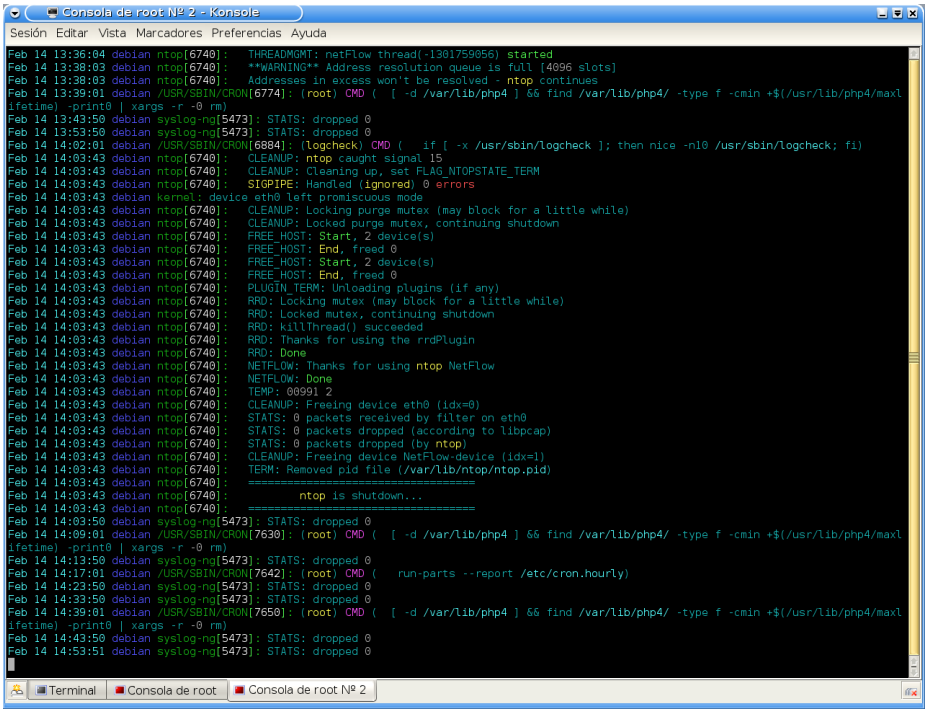


Figura D.6: Uso de *ccze* mediante consola

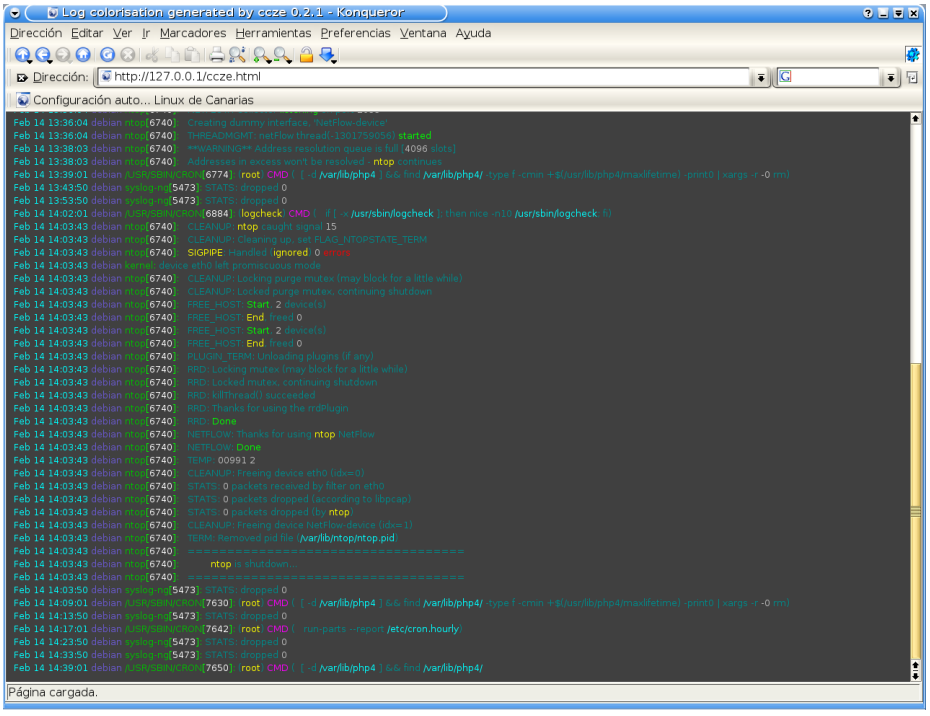


Figura D.7: Uso de *ccze* mediante página *Web*

Apéndice E

Interfaz de configuración

Índice del capítulo

E.1. Requisitos previos para la interfaz

E.1.1. Servidor *Web*: *lighttpd*

Vamos a cubrir es esta sección del documento la instalación y configuración del servidor *Web lighttpd* [70], elegido por *Óliver López Yela* y *Jose Carlos Ramírez Pérez* en el Proyecto de Fin de Carrera *Anubix: Servidor de seguridad perimetral*. El principal motivo de seleccionar este servidor *Web* en lugar del extendido y popular servidor *Web apache* fue la baja carga del sistema que produce *lighttpd* comparado con *apache*. Durante el desarrollo de este proyecto hemos tenido en cuenta en numerosas ocasiones temas referentes a carga de los sistemas o de las redes y por eso hemos querido

continuar conservando ese aspecto en esta parte del proyecto.

lighttpd se encuentra en el momento de la realización de este Proyecto como *paquete Debian* disponible en los repositorios de la versión *stable*, concretamente en la versión 1.4.1. La instalación será muy simple:

```
#> apt-get install lighttpd
```

Los contenidos del paquete se podrán consultar como sigue:

```
#> dpkg -L lighttpd
```

El archivo de configuración de *lighttpd* se ubica en `/etc/lighttpd/lighttpd.conf`. Necesitaremos realizar algunas modificaciones en el mismo [71]:

1. Primero necesitamos habilitar el soporte PHP y la ejecución de *scripts* CGI en el servidor:

`/etc/lighttpd/lighttpd.conf` (I)

```
    ## modules to load
10  # at least mod_access and mod_accesslog should be loaded
    # all other module should only be loaded if really neccesary
    # - saves some time
    # - saves memory
    server.modules              = (
15  #                               "mod_rewrite",
    #                               "mod_redirect",
                                    "mod_access",
    #                               "mod_auth",
    #                               "mod_status",
20                                  "mod_fastcgi",
    #                               "mod_simple_vhost",
    #                               "mod_evhost",
                                    "mod_cgi",
    #                               "mod_compress",
25  #                               "mod_ssi",
    #                               "mod_usertrack",
    #                               "mod_rrdtool",
                                    "mod_accesslog" )
```

2. Será conveniente cambiar la interfaz donde *lighttpd* atiende sus peticiones:

`/etc/lighttpd/lighttpd.conf` (II)

```
    ######### Options that are good to be but not neccesary to be changed
         #######

    ## bind to port (default: 80)
    #server.port                = 81
110
    ## bind to localhost (default: all interfaces)
    #server.bind                = "grisu.home.kneschke.de"
    server.bind  = 192.168.100.24
```

3. Necesitaremos activar el módulo `fastcgi` que habilita el soporte PHP:

`/etc/lighttpd/lighttpd.conf` (III)

```
    #### fastcgi module
    ## read fastcgi.txt for more info
165 fastcgi.server              = ( ".php" =>
                                    ( "localhost" =>
                                      (
```

```
                                        "socket" => "/tmp/php-fastcgi.socket",
    #                                    "bin-path" => "/usr/local/bin/php"
170                                      "bin-path" => "/usr/bin/php4-cgi"
                                    )
                                  )
                                )
```

Nótese que la ruta que hemos indicado para el parámetro `bin-path` aún no existe ya que no hemos instalado todavía el soporte PHP en el sistema.

4. También será necesario configurar el soporte para CGI:

`/etc/lighttpd/lighttpd.conf` (y IV)

```
    #### CGI module
175 cgi.assign                  = ( ".pl"  => "/usr/bin/perl",
                                    ".cgi" => "/usr/bin/perl" )
```

Además de esto, es necesario crear un enlace simbólico dentro de la ruta servida por *lighttpd* al directorio del sistema que alberga los *scripts* CGI:

```
#> ln -s /usr/lib/cgi-bin/ /var/www/cgi-bin
```

5. Existen más opciones de configuración de *lighttpd* en su archivo de configuración que pueden ser interesantes, pero no nos centraremos sobre ellas dado que el archivo de configuración está bien comentado y es de fácil comprensión.

E.1.2. Soporte PHP: *PEAR*

Para dotar a nuestro sistema de soporte PHP necesitaremos descargar una serie de paquetes con *apt*:

```
#> apt-get install php4 php4-cgi php4-cli php4-common php4-pear
```

Gracias al paquete ***php4-pear*** disponemos de acceso a una serie de repositorios de módulos y código PHP, de los que obtendremos los siguientes módulos necesarios para el funcionamiento de nuestra interfaz de configuración:

```
#> pear install Config patError patForms patTemplate
```

También necesitaremos algunos paquetes en estado *beta*, por lo que tenemos que cambiar la configuración de *PEAR* para que utilice esos paquetes de los repositorios. Eso se hará con la siguiente orden:

```
#> pear config-set preferred_state beta
```

Tras ello descargaremos los paquetes necesitados en estado *beta*:

```
#> pear upgrade XML_Parser

#> pear install XML_Serializer
```

Todo el código correspondiente a estos módulos se ubica en `/usr/share/php`. A medida que instalemos más módulos su código se ubicará dentro de esa ruta.

Hecho esto, *lighttpd* estará listo para ser reiniciado con su nuevo soporte PHP:

```
#> /etc/init.d/lighttpd restart
Restarting lighttpd: lighttpd.
```

E.1.3. Ejecución de instrucciones con privilegios: *sudo*

Como se ha indicado en el apartado 8.3.2.3, "*Descripción de atributos y métodos*", existen ciertos métodos dentro de los módulos de la interfaz que realizan algún tipo de interacción con los servicios del sistema. Para que dicha interacción sea adecuada, es necesario que la interfaz disponga de algún método para realizar las operaciones adecuadas con los privilegios suficientes, normalmente privilegios de `root`.

Una solución simple a este problema es la utilización de la aplicación *sudo* para realizar esas operaciones con los privilegios adecuados. Si bien una solución mucho más adecuada sería el desarrollo de una aplicación intermedia que recibiese las peticiones de la interfaz y realizase las operaciones pertinentes, de manera que la interfaz (o mejor dicho el usuario de la interfaz) nunca adquiriese tales privilegios dotando a la aplicación de mayor seguridad, esta solución es compleja y se escapa del ámbito de este Proyecto de Fin de Carrera. Por tanto nos quedamos con la solución de *sudo* que, configurada adecuadamente, no tiene grandes inconvenientes de seguridad.

sudo se incluye en la instalación base del sistema **Debian Sarge**. De no ser así, su instalación puede realizarse como sigue:

```
#> apt-get install sudo
```

El archivo de configuración de *sudo* reside en `/etc/sudoers`. Este archivo es de gran importancia ya que en él se indican los usuarios que van a poder realizar las operaciones privilegiadas y cuáles son esas operaciones privilegiadas para cada usuario. Además en la configuración haremos que al usuario que está ejecutando la interfaz, normalmente `www-data`[1], no se le solicite su contraseña para que así el funcionamiento de la interfaz sea el adecuado. No es nuestro objetivo centrarnos en detalle sobre los aspectos de la configuración de *sudo* ni en el manejo del editor de su configuración, la aplicación *visudo*.

El archivo de configuración que nosotros hemos utilizado es el siguiente:

/etc/sudoers

```
# /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the man page for details on how to write a sudoers file.
#

# Host alias specification

# User alias specification

# Cmnd alias specification
Cmnd_Alias INTERFAZFC = /etc/init.d/flow-capture, /usr/bin/flow-capture
Cmnd_Alias INTERFAZFS = /etc/init.d/flowscan, /usr/bin/flowscan
Cmnd_Alias COMANDOS = /bin/rmdir, /bin/mv, /bin/mkdir, /usr/sbin/update-rc.d

# User privilege specification
root    ALL=(ALL) ALL
www-data ALL= (root) NOPASSWD: INTERFAZFC, INTERFAZFS, COMANDOS
```

[1] Nos referimos al usuario del sistema sobre el que se está ejecutando la interfaz, es decir, el usuario sobre el que se ejecuta el servidor *Web* instalado en el sistema.

Con esta configuración, y siempre buscando la mayor seguridad posible, el usuario `www-data` tiene permisos para ejecutar las distintas operaciones requeridas por la interfaz sin darle permiso para efectuar ninguna otra operación no listada entre las anteriores.

E.1.4. Acceso a ficheros con privilegios restringidos

Durante el proceso de instalación y configuración de las distintas herramientas seleccionadas en este documento se han visto una serie de archivos de configuración que ahora pasarán a ser modificados por nuestra interfaz. Estos archivos, por defecto, son accesibles en modo escritura únicamente por el usuario `root` y se presenta entonces el problema de acceder a ellos desde el usuario de la interfaz, típicamente `www-data`.

No es una buena solución dar permisos de `root` al usuario `www-data` de forma permanente por motivos de seguridad, pero no es tampoco una buena solución realizar cambios sobre los permisos y propietarios de los archivos que han de ser modificados por la interfaz, así como en los directorios en los que la interfaz pueda necesitar crear algún fichero (como los ficheros de resumen).

De estas dos (malas) opciones nosotros hemos elegido la segunda ya que pensamos que ofrece menos fallos de seguridad. Por tanto se ha realizado lo siguiente con cada archivo o directorio afectado:

```
#> chgrp www-data FICHERO
```

```
#> chmod g+w FICHERO
```

Con el primer comando cambiamos el grupo que es propietario de `FICHERO` haciendo que sea el grupo `www-data`, grupo al que en principio pertenecerá únicamente el usuario `www-data` sobre el que se ejecuta el servidor *Web*. Con el segundo, cambiamos los permisos del grupo añadiéndole permisos de escritura en caso de que no lo tuviese.

Como decimos esta solución no es buena ya que cualquier modificación en los permisos del archivo llevará a molestos mensajes de error en la interfaz además de a su malfuncionamiento. En cambio, en el uso y funcionamiento normal de la interfaz y de las utilidades seleccionadas no hay nada que, en principio, vaya a realizar modificaciones sobre los permisos de esos archivos y directorios por lo que, a pesar de ser mala, la solución se muestra a priori válida.

Los archivos sobre los que la interfaz hará modificaciones serán:

```
/etc/flow-tools/flow-capture.conf
/etc/flowscan/flowscan.cf
/etc/flowscan/CUFlow.cf
/var/www/interfaz/ficheros-resumen/resumenFlow.xml
```

Los directorios a los que será necesaro cambiar los permisos debido a la posible creación de nuevos ficheros por parte de la interfaz serán:

```
/var/www/interfaz/ficheros-resumen/
```

E.1.5. Ejecutando *lighttpd* como superusuario

Una forma alternativa para solventar las dos problemáticas anteriores planteadas en los puntos E.1.3 y E.1.4 pasa por conseguir hacer que el servidor *lighttpd* se ejecute con permisos de superusuario. Si bien esto no es posible directamente (mediante la configuración del servidor) se puede lograr hacer que *light* se ejecute como `root` de la manera siguiente:

1. En primer lugar es necesario volver a la configuración de *lighttpd* y realizar las siguientes modificaciones:

   `/etc/lighttpd/lighttpd.conf` (modificaciones)

   ```
   #### fastcgi module
   ## read fastcgi.txt for more info
   fastcgi.server               = ( ".php" =>
                                    ( "localhost" =>
                                      (
                                        "socket" => "/tmp/php-fastcgi.socket",
   #                                     "bin-path" => "/usr/local/bin/php"
                                        "bin-path" => "sudo /usr/bin/php4-cgi"
                                      )
                                    )
                                  )
   ```

   Con este pequeño truco estamos haciendo que el intérprete PHP, `/usr/bin/php4-cgi`, siempre se esté ejecutando como parte de una orden *sudo*.

2. En segunfo lugar hay que configurar *sudo* de la manera correcta:

   `/etc/sudoers`

   ```
   # /etc/sudoers
   #
   # This file MUST be edited with the 'visudo' command as root.
   #
   # See the man page for details on how to write a sudoers file.
   #

   # Host alias specification

   # User alias specification

   # Cmnd alias specification

   # User privilege specification
   root    ALL=(ALL) ALL
   www-data ALL= (root) NOPASSWD: ALL
   ```

   Con esta pequeña configuración hacemos que el usuario `www-data` pueda ejecutar cualquier comando como superusuario sin que se le pida la clave. Si no queremos darle permisos de ejecutar todos los comandos del sistema podemos utilizar una configuración como la vista en el punto E.1.3.

Esta configuración combinada permitirá a *lighttpd* la ejecución de comandos privilegiados además de la escritura en directorios y modificación de archivos con privilegios, pero en cambio no estamos modificando ni los permisos ni los propietarios de los archivos y directorios afectados. Si escogemos un usuario adecuadamente configurado, como por ejemplo un usuario nuevo creado en el sistema al que privemos de *shell* y demás privilegios, esta opción apenas presenta problemas de seguridad. Esta será la opción usada finalmente por comodidad y para respetar los privilegios y propietarios de los archivos del sistema.

E.2. Construcción de la interfaz

E.2.1. Estructura de los ficheros fuente

La aplicación estará estructurada en un único directorio del sistema, cuyo contenido está organizado como se indica a continuación:

```
/var/www/interfaz/      #directorio de la aplicacion
|
|-- configExtensions    #extensiones a la ílibrera Config
|   `-- Container
|       |-- cuflow_Container.php
|       |-- flow_capture_Container.php
|       `-- flowscan_Container.php
|
|-- definicionForms     #definicion de formularios en XML
|   |-- definicionFormsFlow.php
|   |
|   |-- extract         #utilidades para conversion de formularios
|   |   |-- arrays2xml.php
|   |   |-- form2xml.php
|   |   `-- xml2form.php
|   |
|   |-- flow-capture-config-formdef.xml
|   `-- flowscan-config-formdef.xml
|
|-- ficheros-resumen    #directorio de ficheros resumen
|   |-- resumen-flowcapture.xml
|   `-- resumen-flowscan.xml
|
|-- flow-capture-config.php
|
|-- flowscan-config.php
|
|-- includes             #directorio de ficheros de inclusion
|   |-- clase_ConfiguradorFlow.php
|   |-- clase_Configurador_flow_capture.php
|   |-- clase_Configurador_flowscan.php
|   |-- clase_GestorConfigFlow.php
|   |-- erroresFlow.php
|   |-- funcionescomunesFlow.php
|   `-- globalesFlow.php
|
|-- index.html          #archivo de inicio de la interfaz
|
|-- patFormsExtensiones #extensiones a la libreria patForms
|   |-- CIDR.php
|   |-- Definition.php
|   |-- Directory.php
|   |-- GroupConditionalRequired.php
|   |-- HTMLFile.php
|   |-- Ipaddress.php
|   |-- Ipaddress2.php
|   |-- Ipport.php
|   |-- Number.php
|   |-- Service.php
|   |-- Size.php
|   |-- TOS.php
|   `-- UsedValues.php
|
`-- templates           #plantillas HTML
    |
    |-- basicos         #plantillas HTML para interfaces
    |   |-- flow-capture-config-template.html
    |   |-- flowscan-config-template.html
    |   `-- interrogacion.gif
    |
    `-- menu            #plantillas HTML para el menu
        |-- TMimages
        |   |-- branch.gif
        |   |-- branchbottom.gif
        |   |-- branchtop.gif
```

```
|   |-- folder.gif
|   |-- line.gif
|   |-- linebottom.gif
|   |-- minus.gif
|   |-- minusbottom.gif
|   |-- minustop.gif
|   |-- plus.gif
|   |-- plusbottom.gif
|   `-- plustop.gif
|-- TreeMenu.css
|-- TreeMenu.js
|-- menu.html
|-- titulo.html
`-- titulo.png
```

E.2.2. Notas sobre la implementación

E.2.2.1. Clase **Configurador**: Modos de operación

Este punto versa sobre el atributo `modo` de la clase **Configurador**.

Como se ha explicado anteriormente, el hecho de instanciar el objeto en `MODO_REFRESCAR` varía el funcionamiento interno de la clase, y concretamente implicará que no se debe procesar órden alguna procedente de la interfaz de usuario, sino únicamente leer los ficheros de configuración (y, especialmente, los ficheros de resúmen) para calcular los nuevos parámetros afectados y escribir los cambios. El `MODO_REFRESCAR` pues se trata pues de un modo de operación limitado o reducido para el módulo.

Para instanciar el objeto en uno u otro modo, y poder identificar dicho modo sin equivocación posible, se han definido dos constantes en el fichero `includes/clase_ConfiguradorFlow.php`, que son:

- `MO_MODO_NORMAL`: con valor 0
- `MO_MODO_REFRESCAR`: con valor 1

El constructor de la clase padre **Configurador** asigna el valor al atributo `modo` según se le haya pasado como parámetro (siendo por defecto `MO_MODO_NORMAL`).

En la implementación, hay que tener en cuenta que una clase en `MODO_REFRESCAR` no debe procesar ningún *submit*, ni obtener valores de los formularios, ni tampoco presentar interfaz gráfica alguna. Su única función será volver a cargar la configuración, con los datos auxiliares provenientes de los ficheros de resúmen (que serán los que más probablemente hayan cambiado, si se le ha invocado en este modo), y re-escribir dicha configuación para, de esta forma, hacerla efectiva.

E.2.2.2. Control de los Servicios del sistema

Este punto trata los métodos de control de servicios del sistema de la clase **Configurador** y sus subclases.

Para que un módulo pueda controlar la ejecución del servicio del sistema íntimamente relacionado con el mismo (por ejemplo: el módulo del recolector

flow-capture necesitará controlar la ejecución del servicio del sistema del recolector), se han creado una serie de métodos en la clase **Configurador** que son:

- *servicio_en_ejecucion()*
- *ejecutar_servicio($accion)*
- *servicio_activo_arranque()*
- *configurar_servicio($accion)*

Mientras que los dos primeros tratan de controlar la ejecución inmediata del servicio (es decir, el estado en que se encuentra el demonio en concreto en este mismo instante), los dos últimos se emplean para modificar la configuración de inicio del servicio (es decir, si el sistema operativo lanzará el servicio en el momento del arranque).

El parámetro `$accion` es la acción que se desea ejecutar o configurar, y puede tomar un valor de los siguientes (definidos en `includes/clase_ConfiguradorFlow.php`):

- `MO_SERVICIO_PARAR`: con valor 0
- `MO_SERVICIO_INICIAR`: con valor 1
- `MO_SERVICIO_RECARGAR`: con valor 2

Así, se indica si se desea iniciar o parar un servicio (o incluso si se desea recargar, esto es, lanzar una señal al demonio para que vuelva a leer sus ficheros de configuración y así adquiera los cambios de forma inmediata sin detener su ejecución). En el caso de emplear estas constantes como argumento del método *configurar_servicio()*, el significado será: `MO_SERVICIO_INICIAR` para configurar el inicio automático del servicio en el arranque; `MO_SERVICIO_PARAR` para no iniciar el servicio en el arranque; y `MO_SERVICIO_RECARGAR` no debe usarse pues no tiene sentido en este contexto.

E.2.2.3. Implementación de la definición de formularios en XML

En este punto se trata la clase **patforms_Definition** empleada a la hora de cargar los formularios de *patForms*. Esta clase fue definida por *Óliver López Yela* y *Jose Carlos Ramírez Pérez* en su PFC *Anubix: Servidor de seguridad perimetral*, y se utilizará igualmente en este Proyecto.

Esta clase fue re-escrita a partir de la original **patforms_Definition** encontrada en la distribución de *patForms* [73], ya que sólo era compatible con PHP versión 5. Para hacerla compatible con PHP 4 (la versión de PHP que utilizamos es nuestra distribución **Debian Sarge**) se ha modificado el código de inicialización de la clase (el constructor), la declaración de variables, y la especificación de la visibilidad de éstas así como de los métodos.

También se corrigió un fallo de la clase original acerca del manejo de los campos de tipo `Enum` (así es como se llaman en *patForms* a los *SELECT* de

HTML). Este fallo consistía en que, si bien al pasar de *Array* a la representación en XML los índices de tipo numérico generaban un nuevo tag (que por defecto se le ha dado el nombre <*tag*>), en la transformación inversa estos elementos se conservaban con dicho nombre ('*tag*') cuando lo correcto sería ignorar dichos elementos; haciendo ésto, los índices numéricos se regeneran automáticamente. Esta corrección se puede ver en la implementación del método *read()*.

Dicha clase modificada se podrá encontrar junto con los archivos de la interfaz, concretamente en `patFormsExtensions/Definition.php`.

E.2.2.4. Utilización de *Arrays* dinámicos para *patTemplate*

Uno de los principales problemas encontrados en el desarrollo de la interfaz se a debido a la naturaleza de las herramientas seleccionadas y la aparición de un número a priori indeterminado de veces de algunos de los elementos que forman parte de sus configuraciones. Para solventar estos problemas ha sido necesario hacer un uso extenso de las capacidades de *patTemplate* para representar dichos elementos en los formularios de la interfaz.

Gracias al método *addRows()* de *patTemplate* [74] podremos insertar de manera repetitiva en los formuarios los elementos de unos *Arrays* adecuadamente estructurados, de manera que cada repetición de la estructura con la información de los elementos se corresponderá con cada una de las repeticiones de los mismos. Estos *Arrays* son adecuadamente ubicados dentro de las plantillas de los módulos gracias a que *patTemplate* puede definir «subplantillas» dentro de sus plantillas y a que son reutilizadas un número a priori no definido de veces en función del tamaño del *Array* utilizado.

No obstante, el manejo de estos *Arrays* no se hará como cabría pensar dentro del método *leer_fichero_conf()*, sino que se realizará en:

- *asignar_valores_campos()*: donde se aprovechan las posibles transformaciónes que se hagan sobre los valores leídos de la configuración, como la asignación de valores por defecto en campos no obligatorios.
- *aplicar_config()*: donde el *Array* se actualizará con la adición o eliminación de elementos en función de las peticiones del usuario, para que se muestre actualizada la información de la interfaz tras el procesado del último *submit* y no haya una inconsistencia entre las últimas acciones realizadas y el estado real de la utilidad.

Finalmente en *presentar_formulario()* es donde se procederá al relleno de las «subplantillas» dinámicas dentro de la plantilla de *patTemplate* con los *Arrays* adecuados, junto con la sustitución de los demás elementos del formulario que no son propios de *patForms*.

E.2.2.5. Traspaso de información entre utilidades mediante ficheros de resumen

Para el traspaso de información entre las distintas utilidades se ha hecho uso de los ficheros de resumen en formato XML generados por cada una de las

mismas.

En el caso de la interfaz del recolector *flow-capture* este leerá el resumen del analizador *FlowScan* para determinar cuáles de los recolectores están siendo procesados, mientras que por otro lado *FlowScan* utilizará el resumen generado por el recolector *flow-capture* para obtener la lista de recolectores del sistema y ofrecer al usuario mediante la interfaz la posibilidad de procesar la información recabada por los mismos.

A la hora de hacer modificaciones desde un módulo *A* sobre la configuración de un módulo externo *B*, estas se harán creando en el fichero de resumen de formato XML del módulo *A* que se ejecuta un resumen del módulo externo *B* con las modificaciones pertinentes. Tras ello la clase **GestorConfig**, que se encuentra definida en `includes/clase_GestorConfigFlow.php`, realizará una instanciación de un objeto del módulo externo *B* en «modo Refrescar» que permitirá la actualización de la configuración del módulo alterado *B* de forma indirecta. Durante esta alteración se hará uso del fichero de resumen del módulo *B* creado por el módulo *A* para realizar las modificaciones pertinentes en la configuración, y tras ello se determinará si el servicio *B* está en ejecución y en caso afirmativo realizar la lectura de los parámetros modificados indirectamente.

E.2.2.6. Gestión de los *submit* relacionados con elementos dinámicos

Otros de los mayores problemas que se ha encontrado durante el desarrollo de la interfaz ha sido la gestión de los *submit* relacionados con los *Arrays* de elementos dinámicos presentes en los formularios. La indeterminación de la cantidad de apariciones de los mismos obliga a manejar dichos *submit* también en forma de *Arrays*, y a pesar de que PHP ofrece capacidades para su manejo no ha sido fácil hallar una solución si además se añade a esta mezcla el traspaso de información que existe entre los distintos módulos.

Como ya se ha indicado en el punto E.2.2.4, "*Utilización de Arrays dinámicos para patTemplate*" los *Arrays* correspondientes a los elementos dinámicos son manejados en *asignar_valores_campos()* y en *aplicar_config()*, y en dichas funciones se preparará también el *Array* de *submits* correspondiente para la interacción con el usuario.

Tras hacer un *submit* en la interfaz, se volverá a procesar el módulo de la aplicación al completo y por tanto se volverán a ejecutar los métodos *asignar_valores_campos()* y *aplicar_config()* en la nueva ejecución.

En esta segunda ejecución debemos encontrar la relación entre el *Array* de *submits* enviado desde el formulario y el *Array* dinámico generado en esta ejecución a través de los métodos *asignar_valores_campos()* y *aplicar_config()*, y esa relación será el órden en el que los elementos aparecen en los *Arrays*.

Así, para el *Array* de recolectores de *flow-capture* que es importado por *FlowScan* a través del fichero de resumen en formato XML, será tan sólo *flow-capture* quien modifique el órden de los elementos que en él aparecen así como los índices de dichos elementos. *FlowScan* siempre mantendrá su *Array* de recolectores indexado de la misma forma en que lo haga *flow-capture*.

Por otro lado, para los demás *arrays* que aparecen en *FlowScan* y que no tienen relación alguna con *flow-capture* será la primera utilidad la única que deba mantener el indexado de los elementos.

Desgraciadamente no ha sido posible hacer que este sistema sea capaz de superar que un usuario haga un uso desordenado de las interfaces de configuración: realizar cambios en la configuración de los recolectores en una u otra aplicación sin proceder a una carga adecuada de la interfaz de la segunda aplicación, donde por carga adecuada queremos decir una ejecución total y desde cero de la interfaz que permita la sincronización de los módulos, puede conllevar a malfuncionamientos y errores en la ejecución de ambos módulos. Para clarificar esto expresamos el siguiente ejemplo: supongamos que un usario dispone de las interfaces de configuracion de *flow-capture* y *FlowScan* a la vez en sendas ventanas de su navegador *Web* favorito. Si dicho usuario realiza cambios en la lista de recolectores de *FlowScan*, no aplica esos cambios, pasa a la interfaz de *flow-capture* y borra algunos de los recolectores y finalmente regresa a la interfaz de *FlowScan* para aplicar los cambios que había seleccionado antes, se podrá producir un error. Esto se debe a que se ha perdido el sincronismo de la información entre los dos módulos por el uso desordenado por parte del usuario y por la carancecia de un *carácter transaccional* del sistema de configuración mediante una interfaz *Web*.

E.2.2.7. Utilización de «expresiones regulares»

En el desarrollo de la interfaz de configuración se ha intentado utilizar siempre «expresiones regulares» [78] para:

1. Estudiar y validar los archivos de configuración de las distintas utilidades.
2. Realizar al validación de los parámetros introducidos por el usuario a través del formulario de la interfaz.

Si bien la utilización de expresiones regulares resulta muy complicada y errática en las fases tempranas de desarrollo de la interfaz, su amplio soporte en PHP con métodos como *preg_match_all()* o *preg_replace()* produce a la larga un importante ahorro de tiempo y sobre todo una considerable reducción del código necesario para realizar las validaciones antes citadas. Utilidades de *software* libre como *kregexpeditor* facilitan enormente el desarrollo y uso de las expresiones regulares para los usuarios poco acostumbrados.

E.2.2.8. Estructura de objetos Config generados a partir de los archivos de configuración

Presentamos en este punto la estructura creada para los objetos de tipo **Config** que se crearán en el estudio de los archivos de configuración de las distintas aplicaciones seleccionadas. Consideramos esto necesario ya que en su definición hubo que hacer una serie de consideraciones y tomar algunas decisiones en su diseño. Presentaremos un ejemplo de archivo de configuración específico y junto a él la estructura que presentará el objeto **Config** correspondiente.

1. Objeto de configuración obtenido a partir de `flow-capture.conf`

   El archivo de configuración de ejemplo a seguir será el siguiente:

   `flow-capture.conf`

```
-w /var/flow/grande_eth0/ -V 7 -n 275 -N 0 -e 0 -E 10M
    192.168.0.34/192.168.0.34/555
-w /var/flow/pequeno_eth0/ -V 7 -n 275 -N 0 -e 0 -E 0
    192.168.0.34/192.168.0.35/556
-w /var/flow/extra -e 0 -E 3M -N 0 -n 1439 -V 5 0/192.168.0.36/557
```

   a) La creación de secciones (*section* del objeto **Config**) se saca a partir de cada línea válida, siendo el nombre de la sección lo único que debe diferenciar unívocamente a cada recolector definido: la tripleta `localip/remoteip/port`.

   b) Cada parámetro de configuración de cada recolector será una directiva (*directive* del objeto **Config**) incluída dentro de la sección correspondiente.

   Por tanto, para el fichero de configuración mostrado la estructura del objeto **Config** sería la mostrada en el Cuadro E.1.

2. Objeto de configuración obtenido a partir de `flowscan.cf`

   El archivo de configuración de ejemplo a seguir será el siguiente:

   `flowscan.cf`

```
# flowscan Configuration Directives
    ##########################################

# FlowFileGlob (REQUIRED)
# use this glob (file pattern match) when looking for raw flow files to be
# processed, e.g.:
# FlowFileGlob /var/local/flows/flows.*:*[0-9]
# FlowFileGlob flows.*:*[0-9]
FlowFileGlob /var/flow/grande_eth0/ft-v07* /var/flow/pequeno_eth0/ft-v07*

ReportClasses CUFlow

# WaitSeconds (OPTIONAL)
# This should be <= the "-s" value passed on the command-line to cflowd, e
    .g.:
WaitSeconds 60

# Verbose (OPTIONAL, non-zero = true)
Verbose 1
```

Elemento		**Nombre**	**Contenido**
Sección		192.168.0.34/192.168.0.34/555	
	Directiva	w	/var/flow/grande_eth0/
	Directiva	V	7
	Directiva	n	275
	Directiva	N	0
	Directiva	e	0
	Directiva	E	10M
	Directiva	localip	192.168.0.34
	Directiva	remoteip	192.168.0.34
	Directiva	port	555
Sección		192.168.0.34/192.168.0.35/556	
	Directiva	w	/var/flow/pequeno_eth0/
	Directiva	V	7
	Directiva	n	275
	Directiva	N	0
	Directiva	e	0
	Directiva	E	0
	Directiva	localip	192.168.0.34
	Directiva	remoteip	192.168.0.35
	Directiva	port	556
Sección		0/192.168.0.36/557	
	Directiva	w	/var/flow/extra
	Directiva	e	0
	Directiva	E	3M
	Directiva	N	0
	Directiva	n	1439
	Directiva	V	5
	Directiva	localip	0
	Directiva	remoteip	192.168.0.36
	Directiva	port	557

Cuadro E.1: Objeto Config para `flow-capture.conf`

a) La creación de secciones (*section* del objeto **Config**) se saca a partir de cada «nueva sección» encontrada en el archivo. Así, la primera vez que encontramos la directiva `FlowFileglob` o `WaitSeconds` en el archivo se crearán dichas secciones.

b) Cada sección de configuración contendrá al menos una directiva (*directive* del objeto **Config**). En el caso de `FlowFileglob` o `ReportClasses`, que pueden tener más de un valor, podrán tener varias directivas dentro de la sección.

Por tanto, para el fichero de configuración mostrado la estructura del objeto **Config** sería la mostrada en el Cuadro E.2.

3. Objeto de configuración obtenido a partir de `CUFlow.cf`

   El archivo de configuración de ejemplo a seguir será el siguiente:

Elemento		**Nombre**	**Contenido**
Sección		FlowFileGlob	
	Directiva	FlowFileGlob	/var/flow/grande_eth0/ft-v07*
	Directiva	FlowFileGlob	/var/flow/pequeno_eth0/ft-v07*
Sección		ReportClasses	
	Directiva	ReportClasses	CUFlow
Sección		WaitSeconds	
	Directiva	WaitSeconds	60
Sección		Verbose	
	Directiva	Verbose	1

Cuadro E.2: Objeto Config para `flowscan.cf`

`CUFlow.cf`

```
# These are the subnets in our network
# These are used only to determine whether a packet is inbound our
# outbound
Subnet 192.168.0.0/24

# These are networks we are particularly interested in, and want to
# get separate rrd's for their aggregate traffic
Network 192.168.0.34/32 grande

# Where to put the rrd's
# Make sure this is the same as $rrddir in CUGrapher.pl
# OutputDir /cflow/reports/rrds
OutputDir /var/CUFlow/rrds

# Track multicast traffic
Multicast

# Keep top N lists
Scoreboard 10 /var/CUFlow/scoreboard /var/www/topten.html

# Same, but build an over-time average top N list
AggregateScore 10 /var/CUFlow/scoreboard/agg.dat /var/www/overall.html

# Our two netflow exporters. Produce service and protocol reports for the
# total, and each of these.
Router 192.168.0.34 expgrande
Router 192.168.0.35 exppequeno

# Services we are interested in
Service 20-21/tcp ftp
Service 22/tcp ssh
Service 53/udp,53/tcp dns
Service 80/tcp http
Service 110/tcp pop3
Service 143/tcp imap
Service 412/tcp,412/udp dc
Service 443/tcp https
Service 4661-4662/tcp,4665/udp edonkey

# protocols we are interested in
Protocol 1 icmp
Protocol 4 ipinip
Protocol 6 tcp
Protocol 17 udp
Protocol 47 gre
Protocol 51 ah
Protocol 57 skip
Protocol 88 eigrp
# Protocol 169
# Protocol 255
```

```
    # ToS bit percentages to graph
    TOS 0 normal
    TOS 1-255 other
55
    # Interested in traffic to/from AS 1
    ASNumber 3 vario3
```

a) La creación de secciones (*section* del objeto **Config**) se saca a partir de cada «nueva sección» encontrada en el archivo. Así, la primera vez que encontramos la directiva `Network` o `Service` en el archivo se crearán dichas secciones.

b) Cada sección de configuración contendrá al menos una directiva (*directive* del objeto **Config**).

Por tanto, para el fichero de configuración mostrado la estructura del objeto **Config** sería la mostrada en el Cuadro E.3.

Elemento		**Nombre**	**Contenido**
Sección		Subnet	
	Directiva	Subnet	192.168.0.0/24
Sección		Network	
	Directiva	grande	192.168.0.34/32
Sección		OutputDir	
	Directiva	OutputDir	/var/CUFlow/rrds
Sección		Multicast	
	Directiva	Multicast	Multicast
Sección		Scoreboard	
	Directiva	numero	10
	Directiva	directorio	/var/CUFlow/scoreboard
	Directiva	salida	/var/www/topten.html
Sección		AggregateScore	
	Directiva	numero	10
	Directiva	archivo	/var/CUFlow/scoreboard/agg.dat
	Directiva	salida	/var/www/overall.html
Sección		Router	
	Directiva	expgrande	192.168.0.34
	Directiva	exppequeno	192.168.0.35
Sección		Service	
	Directiva	ftp	20-21/tcp
	Directiva	ssh	22/tcp
	Directiva	dns	53/udp,53/tcp
	Directiva	http	80/tcp
	Directiva	pop3	110/tcp
	Directiva	imap	143/tcp
	Directiva	dc	412/tcp,412/udp
	Directiva	https	443/tcp
	Directiva	edonkey	4661-4662/tcp,4665/udp
Sección		Protocol	
	Directiva	icmp	1
	Directiva	ipinip	4
	Directiva	tcp	6
	Directiva	udp	17
	Directiva	gre	47
	Directiva	ah	51
	Directiva	skip	57
	Directiva	eigrp	88
Sección		TOS	
	Directiva	normal	0
	Directiva	other	1-255
Sección		ASNumber	
	Directiva	vario3	3

Cuadro E.3: Objeto Config para `CUFlow.cf`

E.3. Instalación de la interfaz

La interfaz desarrollada se provee dentro del CD adjunto que acompaña la documentación de este Proyecto. Puede localizarse en el directorio raíz del CD `interfaz` en una carpeta independiente, como se exige en los *Requerimientos para la recepción de los Proyectos Fin de Carrera en Secretaría*.

La instalación de la interfaz es muy simple y se ofrecen las siguientes opciones:

1. La copia de los archivos a una carpeta dentro de los directorios servidos por el servidor *Web*, típicamente `/var/www`.

2. La copia de los archivos a una carpeta no servida por el servidor, y la realización posterior de un enlace simbólico a dicha carpeta mediante un archivo emplazado dentro de los directorios servidos por el servidor *Web*.

En caso de, por ejemplo, usar el primer método y copiar los archivos a la carpeta `/var/www/interfaz/` la aplicación de la interfaz podrá ser accedida desde la siguiente dirección: http://192.168.100.24/interfaz/ (dirección IP asignada en la configuración de *lighttpd* como única dirección servida). Si la instalación ha sido correcta, debemos ver la pantalla principal de la aplicación como se muestra en la Figura E.1.

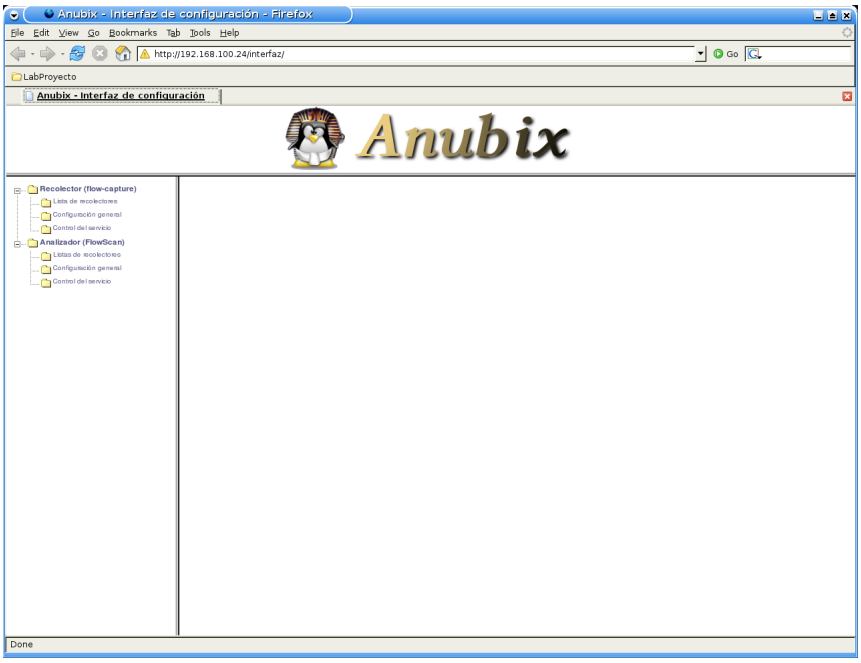


Figura E.1: Pantalla principal de la interfaz de configuración.

E.4. Interfaz para *flow-capture*

E.4.1. Formulario de configuración

Presentamos en la Figura E.2 una captura de la interfaz de configuración de *flow-capture* desarrollado en este Proyecto.

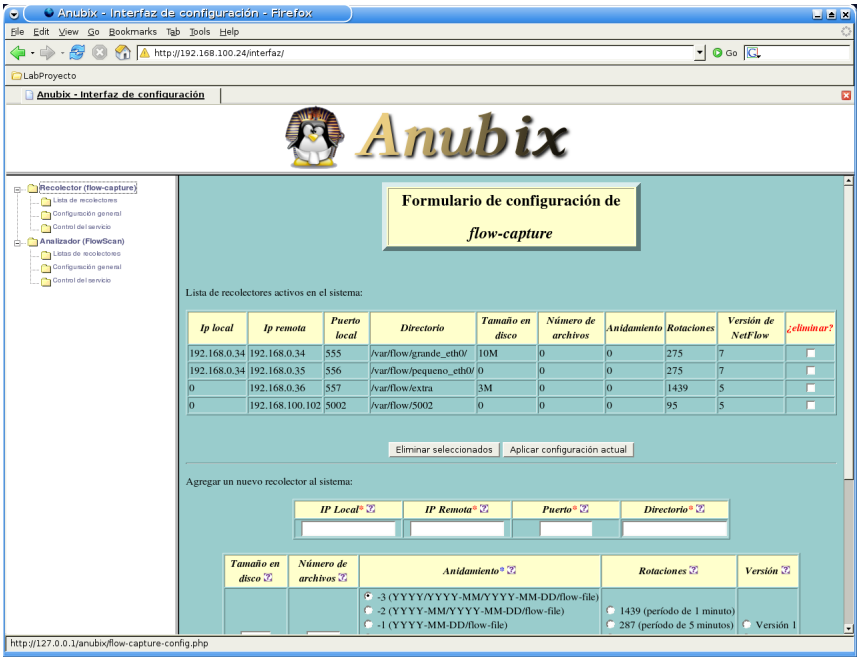


Figura E.2: Interfaz de configuración de *flow-capture*

Los elementos que presenta dicho formulario, definidos a través de *patForms*, para interactuar con el usuario son:

1. IP Local: Dirección IP local donde se espera el flujo proveniente de un recolector. La dirección 0 es válida para indicar cualquier dirección IP del sistema local.
2. IP Remota: Dirección IP remota de donde esperar el flujo de la sonda.
3. Puerto: Puerto local donde esperar el flujo de la sonda.
4. Directorio: Directorio raíz de almacenamiento de las capturas. En él se emplazarán los archivos generados por *flow-capture*.
5. Versión: Versión de NetFlow que se espera recibir de la sonda.
6. Tamaño en disco: Tamaño máximo a ocupar en disco por todos los archivos generados por el recolector.
7. Número de archivos: Número máximo de archivos a mantener en disco para el recolector.

8. Rotaciones: Número de archivos a crear por día

9. Anidamiento: Formato del anidamiento de la captura.

También se presentan los siguientes elementos no definidos a través del formulario de *patForms*:

1. *Checkboxes* «¿eliminar?» seleccionables de la lista de recolectores: permiten al usuario seleccionar uno o varios recolectores activos del sistema para proceder a su eliminación.

2. Botón de acción «Eliminar seleccionados»: permite la eliminación de los recolectores seleccionados de la lista de recolectores activos del sistema.

3. Botón de acción «Aplicar configuración actual»: permite aplicar la configuración establecida actualmente en el archivo de configuración de *flow-capture* y presentada en la interfaz de configuración mediante el reinicio del servicio.

4. Botón de acción «Añadir recolector»: permite añadir un nuevo recolector al sistema especificado mediante los valores de los campos citados en el listado anterior.

5. Botón de acción «Activar» o «Parar»: según proceda, permite activar o parar el servicio del recolector.

6. Botón de acción «Habilitar» o «Deshabilitar»: permite habilitar o no el servicio para que se inicie en el arranque del sistema.

E.4.1.1. Reflexion de los campos configurables

Los campos ofrecidos al usuario en la interfaz de configuración han sido escogidos por los siguientes motivos:

1. Permitir a un usuario introducir de forma cómoda dentro de la configuración de *flow-capture* nuevos recolectores.

2. Ofrecer en la introducción de los nuevos recolectores la posibilidad de alterar algunos aspectos de su configuración, seleccionando los más relevantes y evitando parámetros de funcionalidades muy específicas que compliquen al usuario el manejo de la interfaz.

3. Dar valores por defecto adecuados para los parámetros menos relevantes de la configuración, evitando la distracción del usuario con pormenores de la configuración.

4. Ofrecer, mediante el uso de *checkboxes* y seleccionables tipo *radio*, un interfaz robusto que no induzca al usuario la introducción de errores y valores incorrectos.

5. Permitir al usuario cambiar de forma simple la ejecución del servicio, así como su inicialización en el arranque del sistema.

E.4.2. Importación y exportación de parámetros desde otros módulos

Como parte de la interacción entre las distintas utilidades se producen los siguientes intercambios de información entre las mismas:

1. Desde el módulo del analizador *FlowScan* se obtendrá la lista de recolectores activos en el sistema que están siendo además analizados por dicha utilidad. Esta información se obtiene a partir del fichero de resumen de configuración de *FlowScan*.

2. En el caso del borrado de un recolector que está siendo analizado por *FlowScan*, se procederá a la eliminación del mismo en esta segunda utilidad. Para ello se modificará desde el módulo de *flow-capture* tanto el archivo de configuración de *FlowScan* como su respectivo fichero de resumen de configuración.

E.5. Interfaz para *FlowScan*

E.5.1. Formulario de configuración

Presentamos en la Figura E.3 una captura de la interfaz de configuración de *FlowScan* desarrollado en este Proyecto.

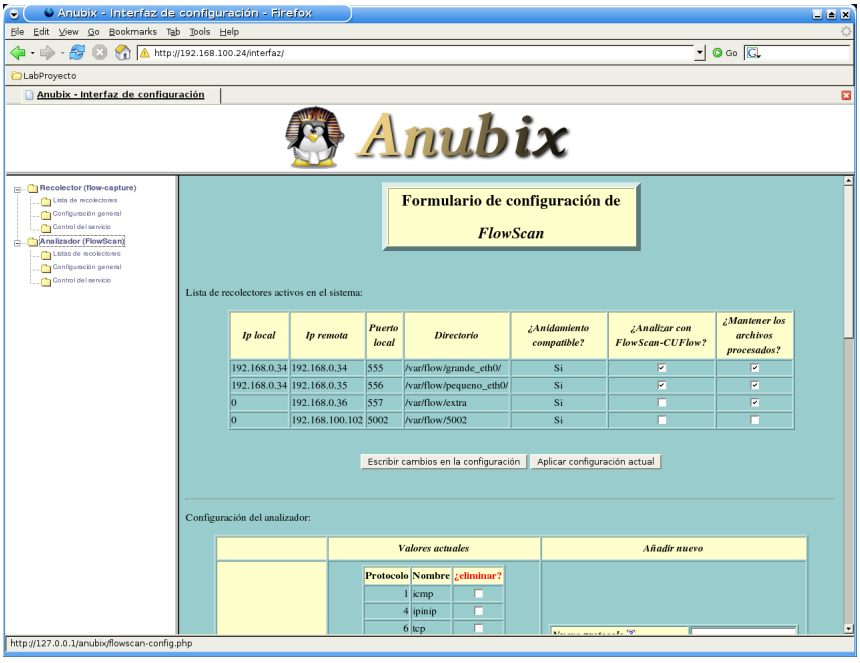


Figura E.3: Interfaz de configuración de *FlowScan*

Los elementos que presenta dicho formulario, definidos a través de *patForms*, para interactuar con el usuario son:

1. Nuevo protocolo: Numérico del nuevo protocolo a analizar. Serán los protocolos en los que estamos interesados en distinguir.
2. Nombre del nuevo protocolo: Nombre del protocolo a analizar.
3. Nuevo servicio: Numérico del nuevo servicio a analizar. Serán los servicios en los que estamos interesados en distinguir.
4. Nombre del nuevo servicio': Nombre del servicio a analizar.
5. Nuevo ToS: Numérico del nuevo ToS a analizar.
6. Nombre del nuevo ToS: Nombre del ToS a analizar.
7. Nueva subred: Nueva subred a analizar (en formato CIDR). Serán las subredes de nuestra red, y serán utilizadas para determinar qué tráfico es entrante y qué tráfico es saliente.
8. Nueva red: Nueva red a analizar (en formato CIDR). Son redes en las que estamos interesados especialmente, y queremos mantenerlas diferenciadas para su estudio.
9. Nombre de la nueva red: Nombre de la red a analizar.
10. Nuevo exportador: Dirección IP del exportador a analizar. El exportador es una sonda de un sistema remoto. Con este parámetro se puede distinguir el tráfico capturado por cada sonda.
11. Nombre del nuevo exportador: Nombre del exportador a analizar.
12. Nuevo AS: Numérico del nuevo AS a analizar. Se utilizará para diferenciar el tráfico desde/hacia el AS en concreto.
13. Nombre del nuevo AS: Nombre del AS a analizar.
14. Registros del Scoreboard: Número de registros del Scoreboard. Será el número máximo de elementos que se mostrarán en el resumen de Scoreboard.
15. Fichero de salida del Scoreboard: Fichero de salida del Scoreboard. Debería ser un fichero dentro de las rutas servidas por el servidor *Web*.
16. Registros del AggregateScore: Número de registros del AggregateScore. Será el número máximo de elementos que se mostrarán en el resumen de AggregateScore.
17. Fichero de salida del Aggregate Score: Fichero de salida del Aggregate Score. Debería ser un fichero dentro de las rutas servidas por el servidor *Web*.
18. Tiempo entre análisis: Tiempo entre análisis consecutivos del analizador (en segundos).

También se presentan los siguientes elementos no definidos a través del formulario de *patForms*:

1. *Checkboxes* seleccionables «¿Analizar con FlowScan-CUFlow?»: permiten al usuario seleccionar los recolectores a los que se desea analizar con *FlowScan*.

2. *Checkboxes* seleccionables «¿Mantener los archivos procesados?»: permiten al usuario seleccionar, para cada recolector, si desea mantener (no eliminar) o no mantener los archivos del recolector ya procesados por el analizador mediante su almacenamiento en un directorio `saved` (ver sección D.1.4.2, "*Configurando FlowScan*", para más detalle).

3. Botónes de acción «Compatibilizar»: permiten al usuario realizar cambios en la configuración del recolector seleccionado para cambiar su formato de anidamiento al formato compatible con *FlowScan* de manera rápida y cómoda.

4. Botón de acción «Escribir cambios en la configuración»: realiza las modificaciones adecuadas, tales como activación o desactivación del análisis de recolectores, adición o eliminación de parámetros de configuración o modificación en parámetros generales, escribiendo los cambios en los archivos de configuración adecuados.

5. Botón de acción «Aplicar configuración actual»: permite aplicar la configuración establecida actualmente en los archivos de configuración de *FlowScan* y presentada en la interfaz de configuración mediante el reinicio del servicio.

6. Botón de acción «Activar» o «Parar»: según proceda, permite activar o parar el servicio del analizador.

7. Botón de acción «Habilitar» o «Deshabilitar»: permite habilitar o no el servicio para que se inicie en el arranque del sistema.

E.5.1.1. Reflexion de los campos configurables

Los campos ofrecidos al usuario en la interfaz de configuración han sido escogidos por los siguientes motivos:

1. Permitir a un usuario seleccionar de forma cómoda qué recolectores del sistema van a ser procesados mediante *FlowScan*.

2. Dar una forma simple de modificar la configuración de los recolectores del sistema para adaptarlos a las necesidades de configuración de *FlowScan*, sin obligar al usuario interaccionar con otros módulos.

3. Ofrecer al usuario una forma robusta de añadir y eliminar elementos de la configuración del analizador.

4. Permitir al usuario cambiar de forma simple la ejecución del servicio, así como su inicialización en el arranque del sistema.

E.5.2. Importación y exportación de parámetros desde otros módulos

Como parte de la interacción entre las distintas utilidades se producen los siguientes intercambios de información entre las mismas:

1. Desde el módulo del recolector *flow-capture* se obtendrá la lista de recolectores activos en el sistema. Esta información se obtiene a partir del fichero de resumen de configuración de *flow-capture*.

2. En el caso de compatibilizar un recolector que no era compatible en su anidamiento con *FlowScan*, se procederá a realizar los cambios adecuados en la configuración de *flow-capture*. Para ello se modificará desde el módulo de *FlowScan* tanto el archivo de configuración de *flow-capture* como su respectivo fichero de resumen de configuración.

Apéndice F

Validación y pruebas realizadas

F.1. Escenario de pruebas

Para la comprobación del correcto funcionamiento de las utilidades seleccionadas así como de la interfaz que las maneja se dispuso de un entorno de pruebas. Durante esta sección se verán los distintos pasos necesarios para replicar dicho entorno:

1. Disposición de una instalación del sistema operativo **Debian Sarge** funcional y limpia. No se ha tratado en este documento la instalación de dicho sistema operativo pues no se considera que sea materia propia del Proyecto. Sin embargo, sí señalamos los siguientes puntos:

   a) La máquina sobre la que se instale deberá disponer al menos de una tarjeta de red. En nuestro caso se dispuso de una tarjeta de red *ethernet* a la que se dió la dirección IP 192.168.100.24.

   b) La instalación del sistema operativo **Debian Sarge** no necesitará en ningún caso de un entorno gráfico, si bien en el entorno de pruebas utilizado sí se disponia del mismo al ser utilizado el propio sistema de pruebas como sistema de escritorio.

2. Instalación en el sistema **Debian Sarge** de la sonda *fprobe-ng*. La sonda se configuró para su funcionamiento en modo promiscuo en la interfaz con dirección IP 192.168.100.24, y la información recogida por la misma se envió a la dirección IP 192.168.100.24 y concretamente al puerto UDP 555 a través de *NetFlow* versión 7 (aunque posteriormente se cambió, por motivos de testado, a versión 5).

   Con esta configuración ya adelantamos que la sonda y el recolector de *NetFlow* estarán instalados sobre un mismo sistema.

   Más detalles sobre la instalación y configuración de la sonda pueden consultarse en el apéndice A.2, "*Instalación y configuración de la sonda*".

3. Instalación en el sistema **Debian Sarge** del recolector *flow-capture*. No será necesario hacer ningún tipo de configuración adicional en el recolector

ya que es uno de los objetivos de las pruebas comprobar la configuración existente a través de la interfaz de configuración y darle la configuración adecuada para recibir la información de *NetFlow* generada por la sonda instalada en el punto anterior.

Más detalles sobre la instalación del recolector *flow-capture* pueden verse en el apéndice A.3.1, "*Instalación de flow-tools*".

4. Instalación en el sistema **Debian Sarge** del analizador *FlowScan* y sus módulos *CUFlow* y *CUGrapher*. Además será necesario realizar la configuración manual de algunos parámetros no manejados por la interfaz de configuración, parámetros no ligados directamente al uso de la aplicación sino a su funcionamiento interno y presentación. Esos parámetros serán:

   a) Modificar la directiva `ReportClasses` de *FlowScan* para que haga uso del módulo `CUFlow`.

   b) Crear y ubicar la *script* para el manejo del servicio de *FlowScan*, ubicándola en la ruta adecuada.

   c) Modificar las directivas `OutputDir`, `Scoreboard` y `AggregateScore` de *CUFlow* para que realicen el almacenamiento de sus archivos (no donde ubicarán sus salidas) en directorios válidos del sistema.

   d) Modificar la directiva `OutputDir` de la configuración de *CUGrapher* para que coincida con el valor que contiene la misma directiva de la configuración de *CUFlow*.

   e) Si se desea, modificar las directivas adicionales para una configuración más ajustada de *CUGrapher*.

   No se realizarán más cambios en las configuraciones de *FlowScan* o *CUFlow* dado que es objetivo de la interfaz la modificación de las mismas.

   Se puede consultar más información acerca de este proceso de instalación y las configuraciones pertinentes a lo largo del apéndice D.1.4, "*Exportando a bases de datos Round Robin*".

5. Instalación en el sistema **Debian Sarge** del servidor *Web lighttpd*. Dicho servidor será configurado para atender peticiones en la dirección IP 192.168.100.24, de manera que nuestro sistema centralizado de monitorización, y concretamente la interfaz de coniguración, será accesible en dicha dirección IP.

   Detalles sobre la instalación del servidor *lighttpd* y su adecuada configuración se muestran en el apéndice E.1.1, "*Servidor Web: lighttpd*".

6. Dotar al servidor *Web lighttpd* de soporte para PHP. Para ello será necesaria la instalación de *PEAR* en el sistema **Debian Sarge**.

   Puede leerse más acerca de la instalación y configuración del soporte PHP en el apéndice E.1.2, "*Soporte PHP: PEAR*".

7. Proceder a la instalación de la interfaz de configuración en el sistema **Debian Sarge**. La interfaz será tras ello accesible en la dirección IP 192.168.100.24.

En el apéndice E.3, "*Instalación de la interfaz*", se encuentran las instrucciones para la instalación de la interfaz de configuración. Por otro lado, será necesario contemplar los aspectos presentados en los apéndices E.1.3, "*Ejecución de instrucciones con privilegios: sudo*", y E.1.4, "*Acceso a ficheros con privilegios restringidos*", para que la interfaz quede en estado totalmente operativo.

8. Durante algunas fases de las pruebas se dispuso de un segundo sistema **Debian Sarge** en el que se ejecutaba una sonda *fprobe-ng* adicional, que dirigía sus flujos hacia la dirección 192.168.100.24 y al puerto UDP 556. Con este segundo exportador se pudo comprobar el comportamiento de las aplicaciones en entornos con más de un exportador de información.

F.2. Pruebas de depuración realizadas sobre la interfaz

En esta sección se presentan una serie de pruebas realizadas sobre la interfaz de configuración de las distintas utilidades. Las pruebas indicadas se corresponden a usos intencionadamente erróneos de la interfaz para someterla a casos excepcionales, si bien durante el desarrollo de la mismas se hizo una utilización completa de las capacidades de la interfaz y, aunque no detallado en esta sección, el funcionamiento de la interfaz y de las utilidades manejadas fue el adecuado.

Pasamos pues a citar las pruebas concretas realizadas sobre cada uno de los módulos.

F.2.1. Depuración de la interfaz para *flow-capture*

Las siguientes pruebas, además de las de un uso normal del programa, se realizaron sobre el módulo de *flow-capture* obteniendo resultados satisfactorios:

1. Lectura y escritura del fichero de configuración

   a) Disponer de líneas de sintaxis no válida en el archivo de configuración de *flow-capture*. En tal caso se muestra un mensaje de error por pantalla y no se permite la ejecución normal de la interfaz.

   b) No tener definido ningún recolector en el archivo de configuración de *flow-capture*. En tal caso el recolector se muestra como «Parado» en el «Estado del recolector».

   c) Existencia de directivas no reconocidas en la definición de algún recolector en el archivo de configuración. Dichas directivas se conservarán en la configuración y no se modificarán a pesar de no mostrarse en el formulario.

2. Inserción de un nuevo recolector desde el formulario

   a) Introducir valores incorrectos en los campos para la inserción de un nuevo recolector. En tal caso se dará un error de validación en el formulario.

   b) No especificar valores en todos los campos obligatorios del formulario. En tal caso se dará un error de validación.

   c) Intentar establecer directorios con espacios en el campo «Directorio» en la inserción de un nuevo recolector, debido a que *flow-capture* no soporta directorios con espacios en su configuración. En tal caso se dará un error de validación.

   d) Introducir un valor ya utilizado en los campos «Puerto» o «Directorio» en la inserción de un nuevo recolector. En tal caso se dará un error de validación.

3. Acciones provocadas desde módulos externos

   a) En caso de realizarse alguna modificación sobre la configuración de un recolector de *flow-capture* y estar *flow-capture* en ejecución, se procederá a la modificación de la configuración y a la aplicación de la nueva configuración a través del reinicio del servicio.

F.2.2. Depuración de la interfaz para *FlowScan*

Las siguientes pruebas, además de las de un uso normal del programa, se realizaron sobre el módulo de *FlowScan* obteniendo resultados satisfactorios:

1. Lectura y escritura del fichero de configuración

   a) Disponer de líneas de sintaxis no válida en el archivo de configuración de *FlowScan*. En tal caso se muestra un mensaje de error por pantalla y no se permite la ejecución normal de la interfaz.

   b) No estar analizando ningún recolector en el sistema. En tal caso se presenta un mensaje de aviso por pantalla y la interfaz no permite la ejecución del servicio. En caso de que además el servicio de *FlowScan* se encontrase en ejecución este se detendría.

2. Modificación de la configuración desde el formulario

   a) Introducir valores incorrectos en los campos para la inserción de nuevos elementos de la configuración del analizador. En tal caso se dará un error de validación en el formulario.

   b) No especificar valores en todos los campos obligatorios del formulario. En tal caso se dará un error de validación.

   c) Introducir un valor ya utilizado en los campos «Nombre del nuevo protocolo», «Nombre del nuevo servicio», «Nombre del nuevo ToS», «Nombre de la nueva red», «Nombre del nuevo exportador» o «Nombre del nuevo AS» en la inserción de nuevos elementos en la configuración. En tal caso se dará un error de validación.

3. Acciones provocadas desde módulos externos

   a) En caso de ser borrado un recolector que esté siendo analizado por *FlowScan* se procederá a la detención del procesado del mismo.

Bibliografía

[1] Óliver López Yela y Jose Carlos Ramírez Pérez, *"Proyecto fin de carrera: Anubix: Servidor de seguridad perimetral"*

[2] Peter J. Welcher, *"NetFlow"*
http://www.netcraftsmen.net/welcher/papers/netflow.html

[3] Wikipedia, *"Netflow"*
http://en.wikipedia.org/wiki/Netflow

[4] Thw Swiss Education & Research Network, *"FloMA: Pointers and Software"*
http://www.switch.ch/tf-tant/floma/software.html

[5] *"NetFlow Export Datagram Formats"*
http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/nfc/nfc_3_6/iug/format.htm

[6] Damien Miller, *"Softflowd"*
http://www.mindrot.org/softflowd.html

[7] Satoshi Fujii, *"pcNetFlow"*
http://inet-lab.naist.jp/measurement/pcNetFlow/pcNetFlow.html

[8] Bogdan Surdu, *"fprobe"*
http://psi.home.ro/flow/

[9] Slava Astashonok, *"NetFlow probes: fprobe and fprobe-ulog"*
http://sourceforge.net/projects/fprobe

[10] Slava Astashonok, *"fprobe - a NetFlow probe. Linux Man Page"*
http://www.penguin-soft.com/penguin/man/8/fprobe.html

[11] Harald Welte, *"Userspace Packet Logging for netfilter"*
http://gnumonks.org/projects/ulogd

[12] William Stearns, *"Iptables: Getting full packets out of the kernel"*
http://www.stearns.org/doc/iptables-ulog.current.html

[13] The Cooperative Association for Internet Data Analysis, *"cflowd: Traffic Flow Analysis Tool"*
http://www.caida.org/tools/measurement/cflowd/

[14] Mark Fullmer, *"flow-tools"*
http://www.splintered.net/sw/flow-tools/

[15] Mark Fullmer, *"flow-capture – Manage storage of flow file archives by expiring old data. Linux Man Page"*
http://www.penguin-soft.com/penguin/man/1/flow-capture.html

[16] Mark Fullmer, *"flow-cat – Concatenate flow files. Linux Man Page"*
http://www.penguin-soft.com/penguin/man/1/flow-cat.html

[17] Mark Fullmer, *"flow-print – Display flows in formatted ASCII. Linux Man Page"*
http://www.penguin-soft.com/penguin/man/1/flow-print.html

[18] Mark Fullmer, *"flow-nfilter – Filter flows. Linux Man Page"*
http://www.penguin-soft.com/penguin/man/1/flow-nfilter.html

[19] Mark Fullmer, *"flow-stat – Generate reports with flow data. Linux Man Page"*
http://www.penguin-soft.com/penguin/man/1/flow-stat.html

[20] Mark Fullmer, *"flow-send – Transmit flow data with the NetFlow protocol. Linux Man Page"*
http://www.penguin-soft.com/penguin/man/1/flow-send.html

[21] Mark Fullmer, *"flow-fanout – Fanout (replicate) flow exports to many destinations. Linux Man Page"*
http://www.penguin-soft.com/penguin/man/1/flow-fanout.html

[22] Mark Fullmer, *"flow-export – Export flow-tools files into other NetFlow packages. Linux Man Page"*
http://www.penguin-soft.com/penguin/man/1/flow-export.html

[23] Paolo Lucente, *"pmacct"*
http://www.pmacct.net/

[24] Neil Winton, *"Zebedee: Secure IP tunnel"*
http://www.winton.org.uk/zebedee/

[25] The Netfilter Core Team, *"The netfilter.org «iptables» project"*
http://www.netfilter.org/projects/iptables/index.html

[26] Rusty Russell, *"Linux 2.4 Packet Filtering HOWTO. Chapter 3. So What's A Packet Filter?"*
http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO-3.html

[27] Rusty Russell, *"Linux 2.4 Packet Filtering HOWTO. Chapter 6. How Packets Traverse The Filters"*
http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO-6.html

[28] Herve Eychenne, *"iptables - administration tool for IPv4 packet filtering and NAT. Linux Man Page"*
http://www.penguin-soft.com/penguin/man/8/iptables.html

[29] Cédric de Launois, *"ROUTE - iptables ROUTE target"*
http://www.netfilter.org/projects/patch-o-matic/pom-extra.html#pom-extra-ROUTE

[30] Patrick Schaaf, *"ROUTE –tee target extension"*
http://lists.netfilter.org/pipermail/netfilter-devel/2004-November/017515.html

[31] Marie Fabrice, *"Netfilter Extensions HOWTO. Chapter 2. Patch-O-Matic"*
http://www.netfilter.org/documentation/HOWTO/netfilter-extensions-HOWTO-2.html

[32] The Free Software Foundation, Inc., *"The GNU Privacy Guard"*
http://www.gnupg.org/

[33] The Linux Kernel Organization, *"The Linux Kernel Archives OpenPGP Signature"*
http://www.kernel.org/signature.html

[34] *"Public Key Server – Verbose Index"*
http://pgpkeys.mit.edu:11371/pks/lookup?op=vindex&search=0xCA9A8D5B

[35] Ulrich Drepper & Scott Miller, *"md5sum - compute and check MD5 message digest. Linux Man Page"*
http://www.penguin-soft.com/penguin/man/1/md5sum.html

[36] Brian Ward, *"Kernel-Como"*
http://www.insflug.org/COMOs/Kernel-Como/Kernel-Como.html

[37] Kevin McKinley, *"Creating custom kernels with Debian's kernel-package system"*,
http://newbiedoc.sourceforge.net/system/kernel-pkg.html

[38] Javier Fernández-Sanguino, *"Creación de paquetes de Debian"*
http://es.tldp.org/Presentaciones/200002hispalinux/conf-29/29-html/debian-paquetes.html

[39] Josip Rodin, *"Guía del nuevo desarrollador de Debian"*
http://www.debian.org/doc/manuals/maint-guide/index.es.html

[40] Machtelt Garrels, *"Introduction to Linux. A Hands on Guide. Section 10.3. Remote execution of applications"*
http://www.tldp.org/LDP/intro-linux/html/sect_10_03.html

[41] O'Reilly & Associates, *"Linux Network Administrators Guide. Section 12.5. Configuring Remote Loginand Execution"*
http://www.faqs.org/docs/linux_network/x-087-2-appl.remote.html

[42] Juan Ayup, *"SSH sin password"*
http://rio4.com.ar/blog/ssh-sin-password/

[43] Antonio Villalón Huerta, *"Seguridad en Unix y redes. El sistema de log en Unix"*
http://es.tldp.org/Manuales-LuCAS/doc-unixsec/unixsec-html/node85.html

[44] Kurt Seifried, *"Guía de Seguridad del Administrador de Linux. Ficheros de Log y otros métodos de monitorización"*
http://www.guug.org/LuCAS/Manuales-LuCAS/GSAL/gsal-19991128-htm/ficheroslog.htm

[45] Christian Schmitz, *"Registro de Sistema de Próxima Generación: Syslog-NG"*
http://www.linux-magazine.es/issue/01/Syslog.pdf

[46] syslog Working Group, *"Signed syslog Messages"*
http://www.ietf.org/internet-drafts/draft-ietf-syslog-sign-18.txt

[47] Network Working Group, *"Reliable Delivery for syslog"*
http://tools.ietf.org/wg/syslog/draft-ietf-syslog-reliable/rfc3195.txt

[48] Abe Singer, Devin Kowatch & Tom Perrine, *"SDSC Secure Syslog"*
http://security.sdsc.edu/software/sdsc-syslog/

[49] BalaBit IT Ltd., *"Syslog-ng"*
http://www.balabit.com/products/syslog_ng/

[50] BalaBit IT Ltd., *"syslog-ng - logs system messages. Linux Man Page"*
http://www.penguin-soft.com/penguin/man/8/syslog-ng.html

[51] BalaBit IT Ltd., *"/etc/syslog-ng/syslog-ng.conf - syslog-ng configuration file. Linux Man Page"*
http://www.penguin-soft.com/penguin/man/5/syslog-ng.conf.html

[52] Nate Campi, *"Expanded syslog-ng conf"*
http://www.campin.net/syslog-ng/expanded-syslog-ng.conf

[53] Nate Campi, *"Central Loghost Mini-HOWTO"*
http://www.campin.net/newlogcheck.html

[54] Anne Carasik & Steve Acheson, *"Using Certificates with Stunnel"*
http://www.stunnel.org/faq/certs.html

[55] *"Setting up client auth with Stunnel"*
http://www.stunnel.org/faq/openssl_stunnel_ServerClientAuth.txt

[56] Michal Trojnara, *"stunnel - universal SSL tunnel. Linux Man Page"*
http://www.penguin-soft.com/penguin/man/8/stunnel.html

[57] Michael W. Lucas, *"Monitoring Network Traffic with Netflow"*
http://www.onlamp.com/pub/a/bsd/2005/08/18/
Big_Scary_Daemons.html

[58] Michael W. Lucas, *"Visualizing Network Traffic with Netflow and FlowScan"*
http://www.onlamp.com/pub/a/bsd/2005/09/15/
Big_Scary_Daemons.html

[59] Michael W. Lucas, *"Building Detailed Network Reports with Netflow"*
http://www.onlamp.com/pub/a/bsd/2005/10/27/
Big_Scary_Daemons.html

[60] Paul Blankenbaker, *"Using the Network Security Toolkit. Chapter 9. Ntop NetFlow Collector Traffic Monitoring"*
http://nst.sourceforge.net/nst/docs/user/ch09.html

[61] Satoshi Fujii, *"NetFlow2MySQL version 0.24 manual"*
http://inet-lab.naist.jp/measurement/NetFlow2MySQL/
NetFlow2MySQL.html

[62] *"NetFlow"*
http://blog.jal.idv.tw/post/1/115

[63] Horatio B. Bogbindero, *"flow-tools"*
http://mailman.splintered.net/pipermail/flow-tools/2003-
February/001107.html

[64] The Cooperative Association for Internet Data Analysis, *"FlowScan - Network Traffic Flow Visualization and Reporting Tool"*
http://www.caida.org/tools/utilities/flowscan/

[65] Dave Plonka, *"FlowScan"*
http://net.doit.wisc.edu/ plonka/FlowScan/

[66] Johan Andersen, *"CUFlow"*
http://www.columbia.edu/acis/networks/advanced/CUFlow/CUFlow.html

[67] Johan Andersen, *"FlowMonitor"*
http://www.columbia.edu/acis/networks/advanced/FlowMonitor/FlowMonitor.html

[68] Gergeley Nagy, *"CCZE"*
http://bonehunter.rulez.org/software/ccze/

[69] Gergely Nagy, *"ccze - A robust log colorizer. Linux Man Page"*
http://www.penguin-soft.com/penguin/man/1/ccze.html

[70] Jan Kneschke, *"lighttpd"*
http://www.lighttpd.net/

[71] Jan Kneschke, *"lighttpd Reference Documentation. Configuration File Options"*
http://trac.lighttpd.net/trac/wiki/Docs%3AConfigurationOptions

[72] The PEAR Group, *"PEAR :: Package :: Config"*
http://pear.php.net/package/Config/docs/latest/

[73] Stephan Schmidt, Sebastian Mordziol & Gerd Schauffelberger, *"patForms"*
http://dogs.php-tools.net/docs/patForms/

[74] Stephan Schmidt, *"patTemplate"*
http://dogs.php-tools.net/docs/patTemplate/

[75] Stephan Schmidt, Sebastian Mordziol & Gerd Schauffelberger, *"patError"*
http://dogs.php-tools.net/docs/patError/

[76] Team Melonfire, *"Template-Based Web Development With patTemplate (part 1)"*
http://www.devshed.com/c/a/PHP/TemplateBased-Web-Development-With-patTemplate-part-1/

[77] Team Melonfire, *"Template-Based Web Development With patTemplate (part 2)"*
http://www.devshed.com/c/a/PHP/TemplateBased-Web-Development-With-patTemplate-part-2/

[78] Wikipedia, *"Expresión regular"*
http://es.wikipedia.org/wiki/Expresi%C3%B3n_regular

[79] Ministerio de Administraciones Públicas, *"Métrica. Versión 3"*
http://www.csi.map.es/csi/metrica3/