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
 5
    # 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

10  # 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

15  # Track multicast traffic
    Multicast

    # Keep top N lists
    Scoreboard 10 /var/CUFlow/scoreboard /var/www/topten.html
20
    # 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
25  # total, and each of these.
    Router 192.168.0.34 expgrande
    Router 192.168.0.35 exppequeno

    # Services we are interested in
30  Service 20-21/tcp ftp
    Service 22/tcp ssh
    Service 53/udp,53/tcp dns
    Service 80/tcp http
    Service 110/tcp pop3
35  Service 143/tcp imap
    Service 412/tcp,412/udp dc
    Service 443/tcp https
    Service 4661-4662/tcp,4665/udp edonkey

40  # protocols we are interested in
    Protocol 1 icmp
    Protocol 4 ipinip
    Protocol 6 tcp
    Protocol 17 udp
45  Protocol 47 gre
    Protocol 51 ah
    Protocol 57 skip
    Protocol 88 eigrp
    # Protocol 169
50  # 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.

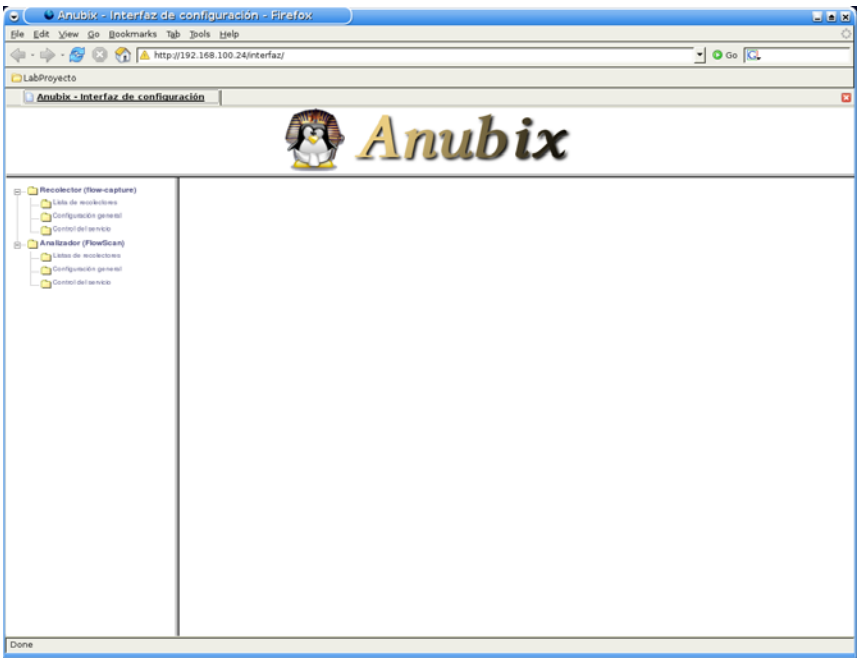

Figura E.1: Pantalla principal de la interfaz de configuración.

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.

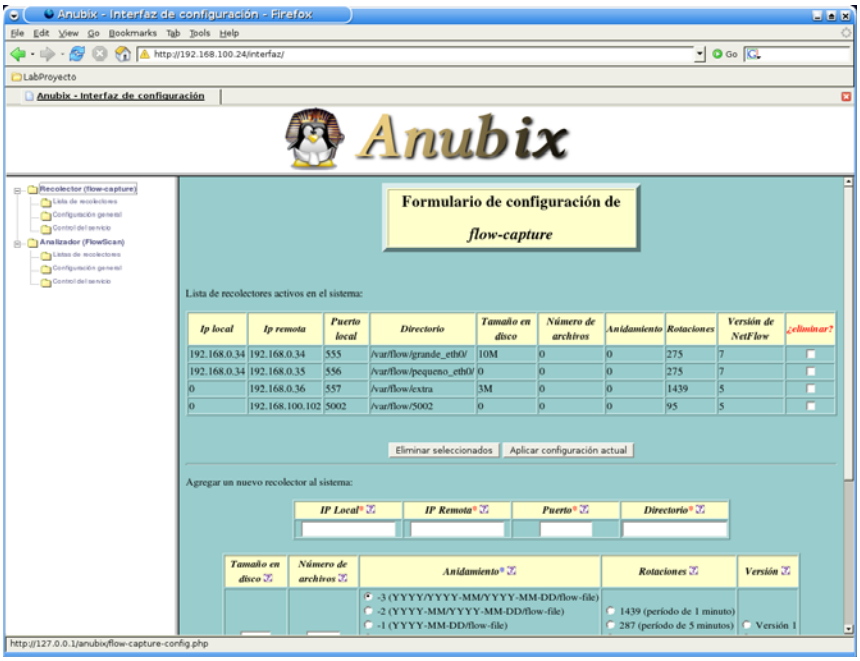

Figura E.2: Interfaz de configuración de *flow-capture*

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.

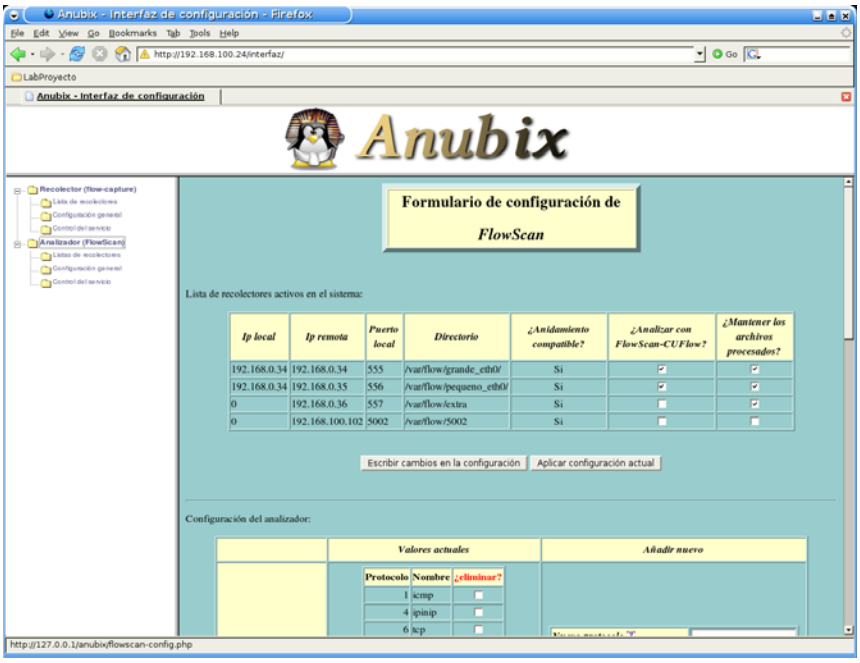

Figura E.3: Interfaz de configuración de *FlowScan*

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.