Parte IV

Anexos

A. Mapa conceptual del software libre

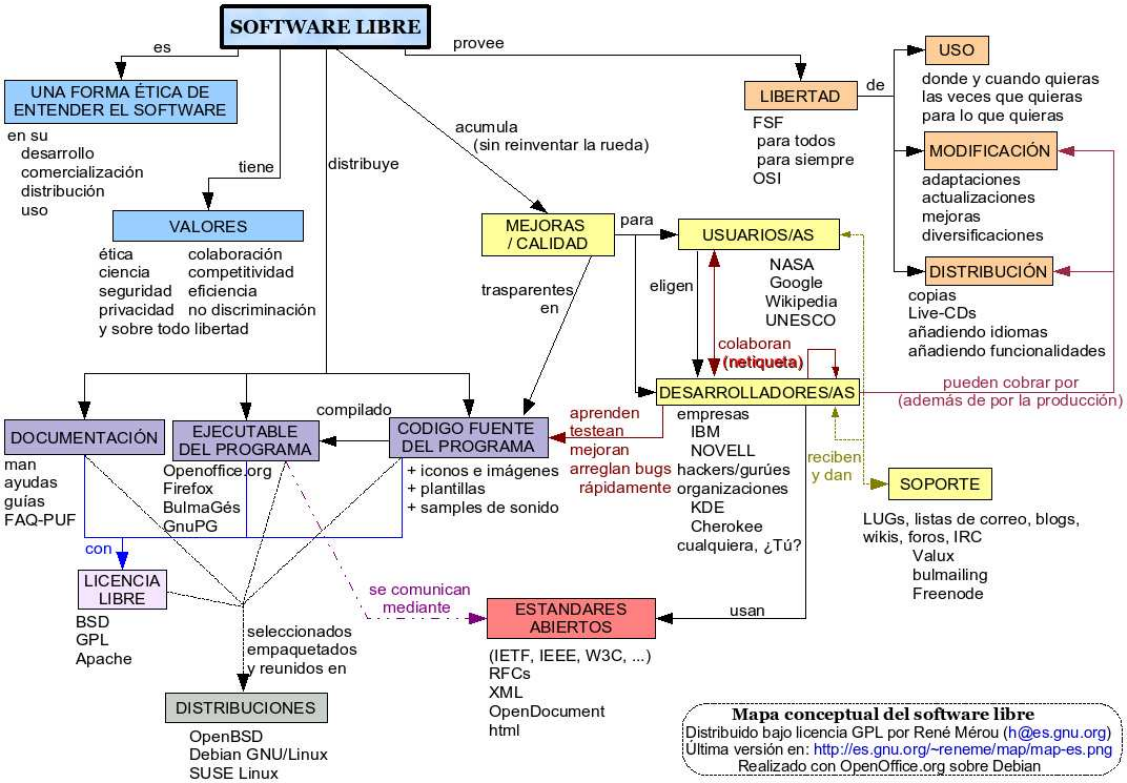

Figura 15: Mapa conceptual del software libre

B. Estructura del directorio principal de la aplicación

La aplicación está estructurada en un único directorio del sistema, cuyo contenido está organizado como se indica a continuación:

```
/var/www/anubix          – directorio de la aplicación
|-- configExtensions          – extensiones a la librería Config
|   `-- Container
|       `-- dglists.php
|       `-- firehol.php
|       `-- snort.php
|-- definicionForms        – definición de formularios en XML
|   |-- clamav-config-formdef.xml
|   |-- dg-config-formdef.xml
|   |-- fh-config-formdef.xml
|   |-- snrt-config-formdef.xml
|   `-- squid-config-formdef.xml
|-- ficheros-resumen       – directorio de ficheros resumen
|   |-- antivirus-resumen.xml
|   |-- cortafuegos-resumen.xml
|   |-- filtrocont-resumen.xml
|   |-- ids-resumen.xml
|   `-- proxy-resumen.xml
|-- includes               – directorio de ficheros de inclusión
|   |-- clase_Configurador.php
|   |-- clase_Configurador_Clamav.php
|   |-- clase_Configurador_DG.php
|   |-- clase_Configurador_FH.php
|   |-- clase_Configurador_Snrt.php
|   |-- clase_Configurador_Squid.php
|   |-- clase_GestorConfig.php
|   |-- errores.php
|   |-- funcionescomunes.php
|   |-- funcionessquid.php
|   `-- globales.php
|-- patFormsExtensiones    – extensiones a la librería patForms
|   |-- Definition.php
|   |-- GroupConditionalRequired.php
|   |-- Ipaddress.php
|   |-- Ipport.php
|   `-- Number.php
|-- privilscripts          – scripts auxiliares para ejecución
|   `-- sq-limpiacache.sh
|-- templates              – plantillas HTML para interfaces
|   |-- basicos
|   |   |-- clamav-config-template.html
|   |   |-- dg-config-template.html
|   |   |-- fh-config-template.html
|   |   |-- fh-status-template.html
|   |   |-- snrt-config-template.html
|   |   |-- snrt-editregl-template.html
|   |   |-- squid-config-template.tmpl
|   |   |-- squid-editarregla-template.tmpl
```

```
|   |   |-- squid-listaacls-template.tmpl
|   |   `-- squid-moverregla-template.tmpl
|   `-- menu                - menú dinámico (JavaScript)
|       |-- TreeMenu.css
|       `-- TreeMenu.js
|-- index.html
|-- menu.html
|-- titulo.html
|-- titulo.png
|-- clamav-config.php
|-- dg-config.php
|-- fh-config.php
|-- fh-status.php
|-- squid-config.php
|-- squid-editarregla.php
|-- squid-listaacls.php
|-- squid-moverregla.php
|-- snort-config.php
|-- snort-editaregla.php
`-- popups.js
```

C. Detalles del proxy Squid

C.1. Instalación y configuración de Squid

La instalación en Debian con apt-get es simple:

apt-get install squid squid-common

Trás la instalación con apt-get, Squid quedará instalado y corriendo como demonio desde el inicio, además queda correctamente creada la estructura de directorios de la caché de Squid, los archivos de log, y el usuario y grupo "proxy".

Si quisiéramos que "init" no lo arranque automáticamente al iniciar el sistema y arrancar squid manualmente, podemos quitar el enlace al script ("*update-rc.d -f squid remove*").

La opción -z en el comando squid ("*squid -z*") creará la estructura de los directorios de caché de Squid. Si el directorio de más alto nivel, especificado en squid.conf no existe Squid intentará crearlo, como usuario y grupo especificado en *squid.conf* (cache_effective_user/cache_effective_group).

C.2. Parámetros de configuración de Squid

La configuración de squid se encuentra en el archivo *squid.conf*. Tras las instalación encontramos un fichero extenso con multitud de opciones comentadas. A continuación aclaramos las principales opciones agrupándolas según su funcionalidad. Podemos consultar un manual más extenso y detallado sobre la configuración de Squid en: Manual de configuracion de squid `http://squid.visolve.com/squid/squid24s1/contents.htm`.

Opciones de red

Squid necesita comunicarse con:

- Servidores Web locales o remotos
- Clientes (navegadores y otros programas)
- Otros proxys cachés

Squid necesitará definir una dirección y un puerto para estas comunicaciones, usando TCP para la conexión con servidores web y clientes y el protocolo ICP (Protocolo de Caché de Internet) para comunicar con otros servidores de caché.

Parámetros:

http_port: define el puerto en el que escucha las peticiones de clientes http, puede tomar el valor del puerto, del nombre de host y el puerto o de la dirección IP y el puerto.

icp_port: fija el puerto por el que squid recibirá y enviará consultas ICP a otros servidores de caché vecinos.

htcp_port: fija el puerto por el que squid recibirá y enviará consultas htcp (Hyper Text Caching Protocol) a otros servidores de caché vecinos.

mcast_groups: especifica una lista de grupos multicast a los que se unirá el servidor para recibir paquetes multicast ICP.

tcp_outgoing_address: normalmente no se especifica, es preferible que el S.O. la elija.

udp_incoming_address: usado por el socket ICP al recibir paquetes de otra caché.

udp_outgoing_address: usado por el socket ICP al enviar paquetes de otra caché.

Parámetros a tener en cuenta en esta sección:

- http_port

Servidores caché pares y arquitectura de cachés

Los parámetros de esta sección son útiles en caso de haber más de un servidor caché en la red.

cache_peer: especifica otra de las cachés de la red, especifica su nombre o IP, la relación con nuestro servidor, los puertos en que funciona y otras opciones.

cache_peer_domain: limita el dominio en el que el servidor será consultado.

neighbor_type_domain: modifica la relación de los servidores de determinados dominios.

icp_query_timeout (mseg): Normalmente Squid asigna un valor óptimo a este temporizador.

maximum_icp_query_timeout (mseg): normalmente el temporizador de consulta ICP se establece dinámicamente, si se quiere fijar un tope máximo se puede usar este parámetro.

mcast_icp_query_timeout (mseg).

dead_peer_timeout (segundos).

hierarchy_stoplist: lista de palabras las cuales al ser encontradas en una URL causan que el objeto en cuestión sea manejado por esta caché.

no_cache: lista de elementos ACL que no serán guardados en la caché nunca.

Parámetros a tener en cuenta en esta sección:

- cache_peer
- no_cache

Tamaño de caché

cache_mem (bytes): especifica la cantidad de memoria usada para almacenar objetos en la caché.

cache_swap_low (percent, 0-100).

cache_swap_high (percent, 0-100).

maximum_object_size (bytes): los objetos que superen este tamaño no serán guardados en disco.

minimum_object_size (bytes): los objetos que no superen este tamaño no serán guardados en disco.

maximum_object_size_in_memory (bytes): los objetos que superen este tamaño no serán retenidos en la memoria caché.

ipcache_size (number of entries): tamaño de la ipcache.

ipcache_low (percent).

ipcache_high (percent).

fqdncache_size (number of entries): tamaño de la caché de entradas FQDN.

cache_replacement_policy: define la política a seguir en la renovación de la caché. Decisiones sobre que objetos permanecen y cuales son eliminados de la caché. Las distintas políticas son: LRU, heap GDSF, heap LFUDA, heap LRU.

memory_replacement_policy: define la política de renovación de la memoria.

Parámetros a tener en cuenta:

- cache_mem
- maximum_object_size_in_memory

Directorios de caché y log

Los parámetros de esta sección definen los directorios y ficheros de caché así como los logs que serán guardados.

cache_dir: define el tipo de sistema de archivado de la caché (normalmente ufs), la ruta al directorio donde los ficheros swap de la caché serán guardados, el máximo tamaño de la caché y el número de directorios en los niveles 1 y 2.

cache_access_log: ruta al archivo access.log que registra todas las consultas http e ICP hechas a la caché.

cache_log: ruta al archivo cache.log que registra información general sobre el comportamiento de la caché.

cache_store_log: indica donde está el fichero store.log, que registra el almacenamiento y borrado de objetos de la caché.

cache_swap_log: indica la ubicación del archivo swap.log, que registra metadatos de objetos guardados en la caché.

emulate_httpd_log (on|off): permite emular el formato de los archivos de log usado por la mayoría de servidores httpd.

log_ip_on_direct.

mime_table: localización del archivo mime.conf, que contiene los tipos mime soportados por squid.

log_mime_hdrs (on|off).

useragent_log.

referer_log.

pid_filename.

debug_options.

log_fqdn (on|off).

client_netmask: permite enmascarar direcciones de clientes en los ficheros de log.

Parámetros a tener en cuenta:

- cache_dir

Soporte de funciones externas

Squid permite invocar funciones que no están dentro del ejecutable de squid, otros programas como por ejemplo redirectores y autenticadores.

ftp_user.

ftp_list_width.

ftp_passive.

cache_dns_program.

dns_children.

dns_retransmit_interval.

dns_timeout.

dns_defnames (on|off).

dns_nameservers.

unlinkd_program.

pinger_program.

redirect_program.

redirect_children.

redirect_rewrites_host_header.

redirector_access.

authenticate_program.

authenticate_children.

authenticate_ttl.

authenticate_ip_ttl.

authenticate_ip_ttl_is_strict.

Parámetros a tener en cuenta:

- redirect_program
- redirect_children
- redirect_rewrites_host_header
- redirector_access

Modificaciones de la caché de squid

wais_relay_host.

wais_relay_port.

request_header_max_size (KB).

request_body_max_size (KB).

reply_body_max_size (KB).

refresh_pattern.

reference_age.

quick_abort_min (KB).

quick_abort_max (KB).

quick_abort_pct (%).

negative_ttl (unidades de tiempo).

positive_dns_ttl (unidades de tiempo).

negative_dns_ttl (unidades de tiempo).

range_offset_limit (bytes).

Temporizadores

connect_timeout (unidades de tiempo).

peer_connect_timeout (unidades de tiempo).

site select_timeout (unidades de tiempo).

read_timeout (unidades de tiempo).

request_timeout.

client_lifetime (unidades de tiempo).

half_closed_clients.

pconn_timeout.

ident_timeout.

shutdown_lifetime (unidades de tiempo).

Control de acceso

El control de acceso es una de las características más importantes de squid.

acl: este parámetro define una lista de acceso.

http_access: permite o deniega el acceso http a una lista de acceso.

icp_access: permite o deniega el acceso icp a una lista de acceso.

miss_access.

cache_peer_access: es similar a cache_peer_domine pero proporciona más flexibilidad al aplicarse sobre listas acl.

proxy_auth_realm.

ident_lookup_access.

Parámetros a tener en cuenta:

- acl
- http_access

Parámetros administrativos

cache_mgr: permite definir la dirección del administrador de la caché que recibirá los emails de avisos.

cache_effective_user: si la caché es arrancada como root se cambiará el UID al indicado en este campo, sino se mantendrá el UID. Hay que resaltar que si Squid no es iniciado como root no se podrá fijar http_port a un valor inferior a 1024.

cache_effective_group: si la caché es arrancada como root se cambiará el GID al indicado en este campo, sino se mantendrá el GID.

visible_hostname: define el nombre del host, para mostrarlo en los mensajes de error, etc.

unique_hostname: si se quiere tener diversas máquinas con el mismo visible_hostname, debemos diferenciarlas mediante este parámetro.

hostname_aliases: lista otros nombre DNS que tenga la caché.

Aceleración de httpd

Squid puede balancear o reducir la carga de un servidor web determinado.

httpd_accel_host: determina el nombre de host del servidor que se acelerará.

httpd_accel_port: puerto al que Squid se conectará en el modo accelerado.

httpd_accel_single_host.

httpd_accel_with_proxy (on|off).

httpd_accel_uses_host_header (on|off).

Parámetros a tener en cuenta:

- httpd_accel_host
- httpd_accel_port
- httpd_accel_with_proxy
- httpd_accel_uses_host_header

Otras

A continuación listamos los parámetros genéricos que no pueden clasificarse en ninguna de las categorias anteriores:

> dns_test, logfile_rotate, append_domain, tcp_recv_bufsize (bytes), err_html_text, deny_info, memory_pools, memory_pools_limit (bytes), forwarded_for, log_icp_queries, icp_hit_stale, minimum_direct_hops, minimum_direct_rtt, cachemgr_passwd, store_avg_object_size (kbytes), store_objects_per_bucket, client_db, netdb_low, netdb_high, netdb_ping_period, query_icmp, test_reachability, buffered_logs, reload_into_ims, always_direct, never_direct, anonymize_headers, fake_user_agent, icon_directory, error_directory, minimum_retry_timeout (seconds), maximum_single_addr_tries, snmp_port, snmp_access, snmp_incoming_address, snmp_outgoing_address, as_whois_server, wccp_router, wccp_version, wccp_incoming_address, wccp_outgoing_address.

Parámetros a tener en cuenta:

- never_direct
- always_direct

C.3. Parámetros de configuración principales

Después de mostrar la multitud de parámetros de configuración de Squid (Anexo C.2), comentaremos aquellos que tendremos en cuenta en nuestra aplicación. Es decir aquellos que el usuario podrá modificar haciendo uso de la aplicación desarrollada para este proyecto o aquellos que se vean afectado directamente por alguno de los anteriores:

http_port: este parámetro se utiliza para determinar la dirección del socket donde Squid escuchará las peticiones de los clientes http. Se contemplan tres formas: solo el puerto; el puerto y el nombre del host; y finalmente el puerto y la dirección IP.

> Uso:
> *http_port port*
> *http_port hostname:port*
> *http_port 1.2.3.4:port*
> Valor por defecto:
> *http_port 3128*

cache_peer: esta opción está dividida en cinco campos: la IP o el nombre de host; el tipo de relación con esa caché, (parent, sibling, multicast); el puerto http de destino del servidor; el puerto ICP (UDP) de peticiones; el quinto campo puede ser una, varias o ninguna palabra clave.
Algunas palabras clave son: proxy-only, Weight=n, ttl=n, no-query, default, round-robin, multicast-responder, closest-only, no-digest, no-netdb-exchange, no-delay, login=user:password, connect-timeout=nn, digest-url=url.

Uso:
cache_peer hostname type http_port icp_port options
Ejemplos:
cache_peer proxy.visolve.com parent 3128 3130 default
cache_peer 172.16.1.100 sibling 3128 3130 proxy-only

no_cache: si una lista de elementos ACL coinciden con lo especificado causan que la respuesta sea eliminada de la caché inmediatamente, es decir se usa para forzar a no cachear determinados objetos.

Uso:
no_cache deny/allow aclname
Valor por defecto:
acl QUERY urlpath_regex cgi-bin \?
no_cache deny QUERY

cache_mem: especifica la cantidad ideal de memoria para cachear: Objetos en tránsito; objetos Hot; objetos negativamente almacenados en el caché. Los datos de estos objetos se almacenan en bloques de 4 Kb. El parámetro cache_mem especifica un límite máximo en el tamaño total de bloques acomodados, donde los objetos en tránsito tienen mayor prioridad. Sin embargo los objetos Hot y aquellos negativamente almacenados en el caché podrán usar la memoria no utilizada hasta que esta sea requerida. De ser necesario, si un objeto en tránsito es mayor a la cantidad de memoria especificada, Squid excederá lo que sea necesario para satisfacer la petición.

Uso:
cache_mem bytes
Valor por defecto:
cache_mem 8 MB

maximum_object_size_in_memory: los objetos mayores del tamaño especificado en este parámetro no serán retenidos en la memoria caché. Deberá ser fijado un tamaño lo suficientemente alto como para que mantenga los objetos a los que se accede frecuentemente en la memoria caché, pero no tan alto como para que la caché se sature.

Uso:
maximum_object_size_in_memory (bytes)
Valor por defecto:
maximum_object_size_in_memory 8 KB

cache_dir: fija el directorio donde reside la caché de squid, además especifica el sistema de almacenamiento que se usará, por defecto *ufs,* si se quiere usar un sistema asíncrono *aufs*, este último es más rápido pero menos estable.
A continuación se especifica el espacio en Mbytes que se destinara al directorio de la caché, y finalmente el número de subdirectorios en el primer y segundo nivel de la caché.

Valor por defecto
cache_dir ufs /usr/local/squid/cache 100 16 256

redirect_program: indica la ubicación del ejecutable del programa redirector a usar, por defecto no se usa ninguno. En nuestro caso nos será útil si queremos que el antivirus filtre los ficheros descargados.

Uso:
redirect_program path/to/redirector

redirect_children: número de procesos del programa redirector a abrir.

Uso:
redirect_children number

redirect_rewrites_host_header: por omisión Squid reescribe en las cabeceras de las peticiones redirigidas el valor de cualquier host. Si se está ejecutando en modo acelerador puede no ser el efecto deseado.

redirector_access: si se define esta lista de acceso especifica que peticiones se mandan al redirecctor y cuales no. Por omisión se mandan todas las peticiones.

acl: con esta opción definimos una lista de acceso. También podemos pasarle la ruta a un fichero, en el que debe haber un elemento por linea. Por defecto las expresiones son sensibles a mayúsculas y minúsculas.
Además del nombre hay que pasarle el tipo de acl (*acltype*), podemos elegir entre: *src, dst, srcdomain, dstdomain, srcdom_regex, dstdom_regex, time, url_regex, urpath_regex, port, proto, method, browser, ident, ident_regex, src_as, dst_as, proxy_auth, proxy_auth_regex, snmp_community, maxconn, req_mime_type, arp.*

Uso:
acl aclname acltype string1 ... / "file"

http_access: permite o impide el acceso http a una acl dada. Es importante el orden en que aparecen listados los *http_access.* Si no concuerda con ninguno de los *http_access* de la lista por defecto se toma lo opuesto a la última linea. Por lo que es buena idea tener como última linea "deny all" o "allow all".

Uso:
http_access allow/deny [!]aclname ...
Valores por defecto:
http_access allow manager localhost
http_access deny manager
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access deny all

httpd_accel_host: esta opción se usa para establecer el nombre del servidor acelerado (hostname). Si se quiere acelerar más de un servidor o cachear tráfico transparentemente, como es nuestro caso, habrá que especificar la palabra *virtual* en lugar del nombre del equipo.
Es importante saber que habilitar *httpd_accel_host* deshabilita el cacheo proxy y el ICP. Si se quiere que estas funciones también estén habilitadas, es necesario habilitar la opción *httpd_accel_with_proxy.*

Uso:
httpd_accel_host hostname(IP)/virtual
Ejemplo, para proxy transparente:
httpd_accel_host virtual

httpd_accel_port: las peticiones aceleradas solo pueden ser reenviadas a un puerto: no existe una tabla que asocie los equipos acelerados a los puertos de destino. Squid se conectará al puerto que se especifique en *httpd_accel_port*. Para dar soporte virtual es necesario especificar el puerto como "0".

httpd_accel_with_proxy: si se usa la opción *httpd_accel_host*, Squid dejará de reconocer las peticiones de cacheo. Por lo tanto si se quiere tener las dos funciones activas, cacheo y aceleración, se debe activar *httpd_accel_with_proxy*.

Uso:
httpd_accel_with_proxy on/off

httpd_accel_uses_host_header: las peticiones HTTP/1.1 incluyen un Host: cabecera, que es básicamente el nombre de host de la URL. Squid puede acelerar diferentes servidores HTTP mirando esta cabecera. Sin embargo, Squid no comprueba la validez de esta cabecera de Host, por lo que abre un importante agujero de seguridad. Por lo que se recomienda que esta opción esté deshabilitada a no ser que sea necesaria.
Es importante comentar que si se usa Squid como proxy transparente esta opción debe ser habilitada.

Uso:
httpd_accel_uses_host_header on/off

always_direct: es posible especificar listas acl, las cuales serán siempre pedidas al servidor origen directamente. Se usa fundamentalmente si estamos usando la opción cache_peer.

Uso:
always_direct allow/deny [!]aclname ...
Valor por defecto:
(*always_direct* por defecto es *deny*)
Ejemplo, para reenviar directamente las peticiones FTP:
acl FTP proto FTP
always_direct allow FTP

never_direct: es la opción opuesta a always_direct, especifica que listas de control de acceso, acl, nunca serán enviadas directamente al servidor de origen.

D. Detalles del filtro de contenidos DansGuardian

D.1. Instalación y configuración de DansGuardian

La instalación es simple usando apt:

apt-get install dansguardian

Los archivos de configuración y listas de expresiones, IPs, sitios, etc, de DansGuardian están en */etc/dansguardian/*. A continuación listamos el contenido de este directorio:

bannedextensionlist
bannediplist
bannedmimetypelist
bannedphraselist
bannedregexpurllist
bannedsitelist
bannedurllist
banneduserlist
contentregexplist
dansguardian.conf
dansguardianf1.conf
exceptioniplist
exceptionphraselist
exceptionsitelist
exceptionurllist
exceptionuserlist
filtergroupslist
greysitelist
greyurllist
languages
phraselists
pics
transparent1x1.gif
weightedphraselist

El archivo de configuración principal es *dansguardian.conf*, sin embargo hay otros que también nos resultarán de interés, se detallan estos archivos y su contenido a continuación.

D.2. Parámetros de configuración de DansGuardian

Enumeración de parámetros Nos ocuparemos en primer lugar del archivo principal de configuración dansguardian.conf.

- **dansguardian.conf**

Estos son todos los parámetros posibles del archivo de configuración (*/etc/dansguardian/dansguardia.conf*), con sus valores por defecto:

```
reportinglevel = 3
languagedir = '/etc/dansguardian/languages'
language = 'ukenglish'
loglevel = 2
logexceptionhits = on
logfileformat = 1

# Network Settings
loglocation = '/var/log/dansguardian/access.log'
filterip =
filterport = 8080
proxyip = 127.0.0.1
proxyport = 3128
accessdeniedaddress = 'http:localhost/cgi-bin/dansguardian.pl'

nonstandarddelimiter = on
usecustombannedimage = 1
custombannedimagefile = '/etc/dansguardian/transparent1x1.gif'
filtergroups = 1
filtergroupslist = '/etc/dansguardian/filtergroupslist'
bannediplist = '/etc/dansguardian/bannediplist'
exceptioniplist = '/etc/dansguardian/exceptioniplist'
banneduserlist = '/etc/dansguardian/banneduserlist'
exceptionuserlist = '/etc/dansguardian/exceptionuserlist'
showweightedfound = on
weightedphrasemode = 2
urlcachenumber = 1000
urlcacheage = 900
phrasefiltermode = 2
preservecase = 0
hexdecodecontent = 0
forcequicksearch = 0
reverseaddresslookups = off
reverseclientiplookups = off

# Build bannedsitelist and bannedurllist cache files.
createlistcachefiles = on

maxuploadsize = -1
maxcontentfiltersize = 256
usernameidmethodproxyauth = on
usernameidmethodntlm = off # **NOT IMPLEMENTED**
usernameidmethodident = off
preemptivebanning = on
forwardedfor = off
usexforwardedfor = off
logconnectionhandlingerrors = on
maxchildren = 120
minchildren = 8
minsparechildren = 4
preforkchildren = 6
maxsparechildren = 32
maxagechildren = 500
ipcfilename = '/tmp/.dguardianipc'
```

urlipcfilename = '/tmp/.dguardianurlipc'
pidfilename = '/var/run/dansguardian.pid'
nodaemon = off
nologger = off
daemonuser = 'dansguardian'
daemongroup = 'dansguardian'
softrestart = off

Dicho archivo de configuración contiene comentarios explicativos de cada parámetro y sus posibles opciones, por lo tanto solo comentaremos los más relevantes, remitimos al lector a dicho archivo para más información.

accessdeniedaddress: dirección URL (no la ubicación física del archivo) del servidor Apache con el script perl de reporte de acceso denegado. Por lo general, Apache está instalado en el mismo servidor Squid y DansGuardian. De así desearlo, esta dirección puede ser cualquier pagina en cualquier servidor web.

reportinglevel: El nivel de información cuando una pagina es denegada es configurable. El mismo puede decir simplemente 'Acceso Denegado', o mostrar detalles de porque ha sido denegado y que es lo que se ha denegado. Esto último puede ser útil para testear, pero puede que no necesite tanto detalle en una aplicación de producción como en una escuela por ejemplo. El modo Stealth agrega los detalles al histórico pero no bloquea nada.
-1 = log, but do not block - Stealth mode
0 = just say 'Access Denied'
1 = report why but not what denied phrase
2 = report fully
3 = use HTML template file (accessdeniedaddress ignored) - recommended

loglevel: Esto permite configurar el nivel de reporte histórico (log). Puede no reportar nada, solo las páginas denegadas, texto o todos los requerimientos de páginas. Los requerimientos de páginas seguras (HTTPS) solo se registran en el histórico cuando el valor de logging equivale a 3 (todas las peticiones).
0 = none
1 = just denied
2 = all text based
3 = all requests

Language Settings: define el idioma (***language*** = *'ukenglish'*) en que se mostrarán los avisos a los clientes, debe ser un idioma de los que estén en ***languagedir*** = *'/etc/dansguardian/-languages'*

Network Settings: Aquí puede modificar la dirección de IP en la que DansGuardian va a aceptar conexiones (**filterip**), si se deja en blanco aceptará peticiones en todas las IP's, todas las tarjetas de red, modems...; el puerto en el que va a escuchar (**filterport**); la dirección de IP (**proxyip**) y puerto (**proxyport**) del servidor Squid.

Content Filtering Settings: Aquí se especifica la ubicación de los archivos que contienen las listas de filtrado. Por lo tanto no es recomendable modificar estos valores.

custombannedimagefile = '/etc/dansguardian/transparent1x1.gif'
filtergroupslist = '/etc/dansguardian/filtergroupslist'
bannediplist = '/etc/dansguardian/bannediplist'
exceptioniplist = '/etc/dansguardian/exceptioniplist'
banneduserlist = '/etc/dansguardian/banneduserlist'
exceptionuserlist = '/etc/dansguardian/exceptionuserlist'

Group Settings: filtergroups fija el numero de grupos de filtrado (1 por defecto). Un grupo de filtrado es un conjunto de opciones de filtrado que pueden aplicarse a un grupo de usuarios; debe haber como mínimo uno. DansGuardian buscará automáticamente el archivo dansguardianfN.conf donde N es el grupo de filtrado. Para asignar usuarios a un grupo de filtrado se usan las opciones de */etc/dansguardian/filtergroupslist*. Todos los usuarios pertenecen por defecto al grupo 1. Debe haber algún tipo de medio de autenticación para ser capaz de asignar usuarios a un grupo. Cuantos más grupos más copias de listas de palabras, sitios. URL's, etc, en RAM, por lo que se recomienda usar los mínimos.

reverseaddresslookups: Si habilita esta opción, DansGuardian hará la resolución inversa DNS (Servidor de Nombre de Dominio) y lo buscará en las listas correspondientes de sitio y URL. Esto evita que un usuario por ejemplo entre una dirección IP para acceder a dominios prohibidos. Esta opción afecta a la velocidad de búsqueda, por lo que a no ser que tenga un servidor DNS local, es recomendable dejar esta opción deshabilitada y usar la opción Blanket IP Block en el archivo bannedsitelist.

Build bannedsitelist and bannedurllist Cache Files (createlistcachefiles): Esta opción compara las fechas de actualización de los archivos de lista y caché y los regenera de ser necesario. Si un archivo bsl o bul .processed existe, ese será el contenido a utilizar. Esto incrementa el proceso de inicio en un 300 %. En computadores lentos la diferencia es significativa y no es necesario en computadores rápidos.

Username identification methods (usernameidmethod...): La opción proxyauth es para cuando se usa autenticación básica en el proxy (obviamente esta opción no sirve para aplicaciones de proxy transparente). La opción ntlm se usa cuando el proxy soporta autenticación MS NTLM. Esto solo funciona con IE5.5 SP1 o posterior, y no ha sido implementada todavía. La opción ident hace que DansGuardian se conecte a un servidor identd en el computador que origino el requerimiento.

forwardedfor: Esta opción agrega el campo X-Forwarded-For: <clientIP>al encabezado HTTP. Esto puede ayudar a resolver problemas con sitios que necesitan saber la dirección IP de origen, o puede ser útil para que las ACL's de Squid sigan funcionando.

maxchildren: Esta opción especifica el numero de procesos a iniciar para aceptar conexiones. Esto puede evitar ataques de DoS afectando a un numero ilimitado de procesos. En sitios con mucho tráfico, este valor debería ser duplicado o triplicado.

logconnectionhandlingerrors: Esta opción registra en el archivo histórico (log) información de 'debug' acerca de estados fork() y accept(), lo cual puede por lo general ser ignorado. Estos estados son registran por syslog y es seguro configurarlo tanto en si o no ('on' u 'off').

- **Otros archivos de configuración y listas**

exceptionsitelist: Este archivo contiene una lista de fines de dominios, al encontrarse en la URL requerida alguno de ellos, DansGuardian no filtrará la página. Observese que no se debe incluir http:// o www. al comienzo de los dominios.

exceptionurllist: Contiene una lista de URL's que no serán filtradas. Se requiere poner el dominio y el path, es decir 'foo.bar' no estaría bien, lo correcto seria 'foo.bar/porn/'. Como en el caso anterior no debe incluirse http:// o www. al comienzo de los dominios.

exceptioniplist: Este archivo contiene una lista de direcciones IP de clientes que saltarán el filtro. Por ejemplo, la dirección IP del administrador.

exceptionuserlist: Lista de nombre de usuario que no serán filtrados (para esto se necesita autenticación básica o "ident").

bannedphraselist: Contiene la lista de frases prohibidas. Las frases deben encerrarse entre los símbolos "<"y ">".

< test>cualquier palabra que empiece por 'test'

<test >cualquier palabra que acabe en 'test'

<test>cualquier palaba que contenga 'test'

< test >solo la palabra 'test'

<this is a test phrase>la frase exacta

<test>,<secondtest>siempre que se encuentren las dos palabras en la pagina.

DansGuardian incluye listas de ejemplo (en inglés) que se pueden incluir en este archivo, por ejemplo: *.Include</etc/dansguardian/phraselists/pornography/banned>*.

Debe evitar usar palabras como <sex>ya que de esta manera bloqueará páginas que contengan por ejemplo "Universidad Middlesex". Las frases pueden contener espacios. Use los espacios para obtener mayor beneficio del filtrado. Esta es la parte más útil de DansGuardian y atrapará más páginas que combinando los filtros de imagen y URL juntos.

También puede usar combinaciones de frases, que de ser encontradas en una página, serán bloqueadas.

bannedmimetypelist: Contiene una lista de tipos MIME prohibidos. Si una URL retorna un tipo MIME incluido en esta lista, DansGuardian lo bloqueará. DansGuardian incluye algunos ejemplo de tipos MIME a ser bloqueados. Esta es una forma interesante de bloquear películas no deseadas, por ejemplo. Es obvio que no tendría sentido filtrar los tipos MIME text/html o image/*.

bannedextensionlist: Contiene una lista de extensiones de archivos no permitidas. Si una URL termina con alguna extensión contenida en esta lista, DansGuardian la bloqueará. DansGuardian incluye un archivo de ejemplo que muestra como denegar extensiones. Esta es una buena forma de evitar que se bajen protectores de pantalla y otras herramientas. De más está decir que sería una tontería bloquear las extensiones .html o .jpg, etc.

bannedregexpurllist: Este archivo contiene una lista de expresiones regulares (regexp) a ser bloqueadas. Para más información sobre expresiones regulares, visite `http://www.opengroup.org/onlinepubs/7908799/xbd/re.html` (en ingles).

bannedsitelist: Este archivo contiene una lista de sitios prohibidos. Si ingresa un nombre de dominio aquí, todo el sitio correspondiente al dominio será bloqueado. Para bloquear partes especificas de un sitio, vea "bannedurllist". Además, si así lo quisiera, puede bloquear todos los sitios con excepción de los especificados en exceptionsitelist. También puede bloquear sitios especificados con direcciones IP e incluir (.Include) una colección de listas negras squidGuard (no lo usaremos). Para habilitar dichas listas, simplemente se ubican las listas donde sea apropiado y se quita el comentario de las lineas correspondientes a los Includes de las listas negras squidGuard al final del archivo bannedsitelist, verificando que las ubicaciones de los archivos son correctas. Para listas negras de URLs, modifique el archivo bannedurllist de manera similar.

bannedurllist: Este archivo permite bloquear partes especificas de un sitio en lugar del sito entero. Para bloquear un sitio entero, vea el archivo bannedsitelist. Para habilitar el bloqueo de listas negras squidGuard, necesita bajar las listas y modificar la sección de listas negras squidGuard al final del archivo, (de la misma forma en que bannedsitelist se ha explicado anteriormente).

weightedphraselist: En este archivo se asocia a cada cadena o palabra un valor, positivo o negativo. De esta manera trata de ponderarse las apariciones de ciertas palabras, por ejemplo: <slut><10>suma 10 al total y <breast>,<medical><-30>resta 30 de la cuenta.

Cuando se supera un valor, establecido en el parámetro ***naughtynesslimit*** del archivo dansguardianfN.conf de cada grupo de filtrado, se bloquea la página.

greysitelist: Similar a *exceptionsitelist* pero en este caso no se detiene el filtrado por contenido cuando se encuentra concordancia con un sitio Web listado en este fichero. No lo utilizaremos en nuestro caso.

greyurllist: Similar a *exceptioniplist* pero no se detiene el filtrado por contenido cuando se encuentra concordancia con una URL listada en este fichero. Tampoco lo utilizaremos en nuestro caso.

pics: Este archivo le permite hacer un ajuste fino del filtro de PICS (Plataforma para la Selección de Contenido de Internet). Cada sección PICS contiene una descripción de las configuraciones permitidas y de lo que las mismas representan. Las configuraciones predeterminadas de DansGuardian están pensadas para jóvenes donde, por ejemplo, los desnudos artísticos están permitidos. Para más detalles e información sobre PICS, visite (en ingles) este sitio `http://www.w3.org/PICS/`.

ICRA: La sección ICRA se explica por si misma. Un valor 0 significa que nada en esta categoría está permitido, mientras que un valor 1 lo permite. Por ejemplo:
ICRAnudityartistic = 1
permite los desnudos artísticos. Para información más detallada visite (en ingles) este sitio.

RSAC: RSAC es una versión antigua de ICRA. Estos valores varían de 0 (nada es permitido) pasando por 2 (valor predeterminado) hasta 4, que permite todo en la categoría. Para información más detallada visite (en inglés) este sitio `http://www.rsac.org/`.

evaluWEB: evaluWEB utiliza un sistema de calificación similar a la sistema usado en Filmes Británicos:
0 = U (Universal, para todas las edades)
1 = PG (Se recomienda la presencia de los padres)
2 = 18 (Solo para mayores de 18)

SafeSurf: Similar a RSAC, pero contiene un rango más amplio de categorías con valores desde 0 = filtrar todo hasta 9 = permitir todo. Para información más detallada visite (en ingles) este sitio `http://www.safesurf.com/`.

Weburbia: Vea evaluWEB. Para información más detallada visite (en ingles) este sitio `http://www.weburbia.com/safe/index.shtml`.

Vancouver Webpages: Otro sistema más para calificación. Vea (en ingles) este sitio `http://vancouver-webpages.com/VWP1.0/` para más información.

E. Detalles del antivirus ClamAV

E.1. Instalación y configuración

- **ClamAV y freshclam**

Instalación Instalamos usando la herramienta apt:

apt-get install -s clamav

Se instalarán los siguientes paquetes extras:

clamav-base clamav-freshclam libclamav1 libgmp3

Paquetes sugeridos:

unrar lha

Paquetes recomendados

arj unzoo

apt se encarga de instalar todos los paquetes necesarios. Crea el grupo y usuario clamav. Tras la instalación se configura el paquete clamav-freshclam con un simple interfaz en consola.

Configuración Lo único configurable son las actulizaciones de virus de la base de datos. De esto se encarga freshclam, que se conecta por el puerto 80 al repositorio que le indiquemos en el archivo de configuración. Para reconfigurar freshclam hacemos: *dpkg-reconfigure clamav-freshclam.* El archivo de configuración de freshclam podemos encontrarlo en */etc/clamav/freshclam.conf,* en posteriores secciones se comentará su contenido.

- **Squid ClamAV Redirector**

Instalación Los requerimientos para hacerlo funcionar son:

- Python 2.3.3
- pyclamav 0.2 `http://xael.org/norman/python/pyclamav/index.html`
- ClamAV 0.80
- Squid 2.5
- SCAVR 1.6.1 `http://www.jackal-net.at/tiki-list_file_gallery.php?galleryId=3`

Primero instalamos python2.3.3:

apt-get install python python2.3-dev

A continuación instalamos clamAV: (en nuestro caso ya está instalado, también libclamav que es requisito para pyclamav)

apt-get install clamav

apt-get install libclamav1-dev

Ahora debemos instalar pyclamav (GPL): pyClamAV es un binding python para libclamav escrito en C. Usando pyClamAV, podemos incluir capacidades de detección de virus a software escrito en python de forma fácil y eficiente. A continuación se muestra el proceso de instalación:

sfa4:~/pycav# tar xvfz pyclamav-0.2.1.tar.gz

pyclamav-0.2.1/CHANGELOG
pyclamav-0.2.1/README.txt
pyclamav-0.2.1/example.py
pyclamav-0.2.1/gpl.txt
pyclamav-0.2.1/pyclamav.c
pyclamav-0.2.1/setup.py

sfa4:~/pycav# cd pyclamav-0.2.1

sfa4:~/pycav/pyclamav-0.2.1# python setup.py build

running build
running build_ext
building 'pyclamav' extension
gcc -pthread -fno-strict-aliasing -DNDEBUG -g -O3 -Wall -Wstrict-prototypes -fPIC -I/usr/local/include -I/usr/include/python2.3 -c pyclamav.c -o build/temp.linux-i686-2.3/pyclamav.o
creating build/lib.linux-i686-2.3
gcc -pthread -shared build/temp.linux-i686-2.3/pyclamav.o -lclamav -o build/lib.linux-i686-2.3/pyclamav.so

sfa4:~/pycav/pyclamav-0.2.1# python setup.py install

running install
running build
running build_ext
running install_lib
copying build/lib.linux-i686-2.3/pyclamav.so -> /usr/lib/python2.3/site-packages

Y finalmente "instalamos" SCAVR:

sfa4:~/pycav# tar xvfz SCAVR.tar.gz

information.php
SCAVR.conf
SquidClamAV_Redirector.py

Copiamos el script a donde queramos, y le damos permisos de ejecucion:

sfa4:~/pycav# cp SquidClamAV_Redirector.py /usr/local/bin/

sfa4:~/pycav# chmod +x /usr/local/bin/SquidClamAV_Redirector.py

Copiamos el archivo de configuración del script al directorio donde están los archivos de configuración de Squid, cambiándole de grupo y propietario al de Squid, (en nuestro caso proxy):

sfa4:~/pycav# cp SCAVR.conf /etc/squid/

sfa4:~/pycav# chown proxy:proxy /etc/squid/SCAVR.conf

Configuración El fichero de configuración de SCAVR (*SCAVR.conf*), podemos situarlo donde queramos, teniendo en cuenta que deberemos pasarle la ruta al parámetro de configuración adecuado en el archivo de configuración del proxy Squid. En nuestro caso lo encontramos en */etc/squid/*.

También es necesario incluir algunas directivas en el archivo de configuración de Squid, con el fin de que reconozca el redirector, estas lineas están documentadas en E.2 en la página 111.

Por otro lado, debemos indicar la URL de la página de alerta que mostrará Squid, en caso de encontrar virus. Un ejemplo de página de información a la que se redirigirán las peticiones bloqueadas por haber encontrado algún virus es este:

```
<html>
<head>
    <title>Virus Information Page</title>
</head>
<body>
    <h1>Virus Information</h1>
    <p>The URL you requested <?php
    if ($_GET) { print $_GET['url']; } ?>
    is infected by a Virus ( <?php
    if ($_GET) { print $_GET['virus']; } ?>)</br>
    The Request is blocked</p>
</body>
</html>
```

E.2. Parámetros de configuración de ClamAV, freshclam y SCAVR

ClamAV y freshclam Las opciones de configuración para el usuario son, como ya hemos comentado, las referentes a las actualizaciones de la base de datos:

- Método de actualización de la base de datos de virus:

  demonio: freshclam se ejecutará como demonio todo el tiempo. Se debería escoger esta opción si se tiene una conexión permanente a Internet. En este caso se debe arrancar el demonio "*/etc/init.d/clamav-freshclam start*".

  ifup.d: freshclam se ejecutará como demonio mientras dure la conexión a Internet. Se recomienda escoger esta opción si se tiene conexión telefónica a Internet y no se desea que freshclam inicie nuevas conexiones.

  cron: freshclam se inicia por el planificador de órdenes (cron). Si se desea un control total sobre cuándo se actualiza la base de datos. Creamos el archivo */etc/cron.d/clamav-freshclam*, en el que insertamos la linea: "*27 */1 * * * clamav [-x /usr/bin/freshclam] && /usr/bin/freshclam –quiet*".

  manual: Sin invocación automática de freshclam. Ésto no se recomienda ya que la base de datos de clamav se actualiza constantemente. No hacemos nada, cada vez que se quiera actualizar el usuario deberá llamar al comando freshclam.

El archivo de configuración de freshclam podemos encontrarlo en */etc/clamav/freshclam.conf*, a continuación vemos un ejemplo del anterior fichero:

DatabaseOwner clamav
UpdateLogFile /var/log/clamav/freshclam.log
LogFileMaxSize 0
MaxAttempts 5
Check for new database 24 times a day
Checks 24
DatabaseMirror db.local.clamav.net
DatabaseMirror database.clamav.net
DatabaseDirectory /var/lib/clamav/
Proxy: http://proxy
HTTPProxyServer proxyhost
HTTPProxyPort proxyport
Proxy authentication: user:passw
HTTPProxyUsername user
HTTPProxyPassword passw

Numero de actualizaciones diarias: se fijan mediante el parámetro *Checks* seguido del numero de actualizaciones diarias.

URL de la base de datos: se fija con *DatabaseMirror*, puede haber varias una en cada linea. Por ejemplo es aconsejable dejar siempre database.clamav.net e incluir alguna otra. Deberíamos ofrecer una lista de posibles opciones, estas podemos sacarlas de */var/lib/dpkg/info/clamav-freshclam.templates*.

Opciones de proxy: si se quiere que freshclam se conecte a través de un proxy se deben incluir varios parámetros: *HTTPProxyServer* con el nombre de host del proxy, *HTTPProxyPort* con el puerto, *HTTPProxyUsername y HTTPProxyPassword* si es que es necesario autenticarse en el proxy.

Squid ClamAV Redirector El archivo de configuración del redirector lo hemos situado en */etc/squid/SCAVR.conf*. Podemos ver las opciones de configuración que tenemos editándolo:

[SquidClamAV]
virusurl = http://localhost/clamAV/infovir.php
cleancache = 300
ForceProtocol = http
MaxRequestsize = 2Mb
log_priority = LOG_INFO
log_facility = LOG_LOCAL6
acceptredirects = 300 301 302 303
[Debug]
Infected = true
Clean = true
Error = true
Ignored = true
[Extensions]
pattern = .scr .bat .pif .vbs .wsh .chm .hlp .reg .shs .vbe .wsf .xla .ini .diz .cpp .cpl .vxd .sys .lnk .hta .exe .zip .rar .ar .com .bzip .gz .eml .dll .ade .adp .adt .app .bas .bin .btm .cla .class .cmd .crt .csc .doc .dot .drv .email .fon .inf .ins .isp .jse .lib .mdb .mde .msi .msp .mst .ocx .pcd .pgm .ppt .rtf .sct .shb .vb .wsc .wss .dat .cab .svr
[Proxy]
http = http://localhost:3128/
#http2 = http://localhost:3128/

El archivo de configuración está dividido en 4 secciones; una primera general, despues una con parámetros de depuración, a continuación de extensiones de archivos y finalmente la sección de proxys.

virusurl indica donde está la página que se mostrará en caso de encontrar virus.

pattern una tras otra se indican las extensiones de archivos que serán escaneados. En archivos con otras extensiones no se buscaran virus.

http Una manera de mejorar el rendimiento es usar el proxy para traer las páginas, y una vez en la cache de squid se servirán al cliente siempre que no contengan virus. En caso de querer usar esta opción debemos incluir un acl para el redirector en el archivo squid.conf.

En el archivo *squid.conf* debemos incluir algunos parámetros:

redirect_program ruta_al_script -c ruta_al_archivo_de_configuración

redirect_children numero_de_procesos

Si queremos incrementar la eficiencia usando un proxy debemos incluir la siguiente acl:

redirector_access deny localhost

F. Detalles del cortafuegos fireHOL

F.1. Instalación y configuración de fireHOL

Instalación
Instalamos el paquete precompilado .deb: *apt-get -s install firehol*

Esto deja instalado fireHOL, pero no podrá correr hasta que editemos el archivo */etc/default/firehol* y cambiemos el valor del parámetro START_FIREHOL a YES.

```
sfa4:/# cat /etc/default/firehol
START_FIREHOL=NO
#If you want to have firehol wait for an iface to be up add it here
WAIT_FOR_IFACE=""
```

Una vez hecho esto podemos ejecutar fireHOL. El archivo de configuración de fireHOL podemos encontrarlo en */etc/firehol/firehol.conf*.

Invocación
fireHOL ha sido diseñado para ejecutarse como un servicio al inicio del sistema. A continuación mostramos algunos de los argumentos en la linea de comandos soportados por fireHOL:

start Activa la configuración del firewall, que debe encontrarse en /etc/firehol/firehol.conf.

try Activa el firewall, pero espera a que el usuario escriba la palabra "commit". Si no introduce esta palabra en 30 segundos se restablece la configuración anterior del firewall.

stop Para el filtrado del firewall, todo el trafico pasará sin ser comprobado.

restart es un alias de **start** para mantener la compatibilidad con /etc/init.d/iptables.

condrestart arranca el firewall FireHOL solo si no está aun activado. No detecta modificaciones en el fichero de configuración solo si ha sido arrancado o no.

status Muestra el firewall que está corriendo, similar a */sbin/iptables -nxvL | less*

panic Elimina todas las reglas del firewall que está corriendo y bloquea (DROP) todo el tráfico de todas las tablas de iptables (mangle, nat, filter) y de todas las cadenas predefinidas (PREROUTING, INPUT, FORWARD, OUTPUT, POSTROUTING), por lo tanto bloquea todas las comunicaciones IP. Esto no lo consigue cambiando la política por defecto a DROP, sino añadiendo una sola regla por tabla/cadena que bloquee todo el tráfico.
Cuando se activa el modo panic, FireHOL comprueba la existencia de la variable de entorno shell SSH_CLIENT (establecida por SSH). Si la encuentra se permitirán las conexiones SSH ya establecidas. (Para que lo anterior funcione, debemos hacer *su* sin el argumento (-), ya que su - sobreescribe las variables shell y por lo tanto la variable SSH_CLIENT se pierde.
También puede especificarse una dirección IP tras el parámetro panic, esto hace que se permitan todas las conexiones con dicha dirección de red.

save Arranca el firewall y lo guarda usando */sbin/iptables-save* en */etc/sysconfig/iptables*.

debug Parsea el fichero de configuración pero en lugar de activarlo muestra las sentencias de iptable generadas.

explain Entra en un modo interactivo que acepta comandos de configuración normales y muestra los comandos de iptables generados para cada uno de ellos. Además genera automáticamente un script basado en los comandos correctos recibidos.

helpme Intenta adivinar la configuración adecuada de fireHOL para la máquina en que corre. No se para el firewall y el fichero de configuración generado se muestra por la salida estándar. Una recomendación es ejecutarlo de la siguiente forma: */etc/init.d/firehol helpme >/tmp/-firehol.conf* .
El fichero de configuración de FireHOL generado debe ser editado antes de usarlo ya que se deberán tomar algunas decisiones, los comentarios generados sirven de ayuda en esta tarea.

<**a filename**> un fichero de configuración diferente. Si no se da otro argumento, se intenta hacer **try**. Sino debe aparecer uno de los siguientes argumentos: start, debug, try.

<**nothing**> Muestra ayuda sobre el uso de FireHOL.

F.2. Parámetros de configuración de fireHOL

A continuación mostramos un fichero de configuración completo a modo de ejemplo:

```
# Requir release 5 of FireHOL configuration directives
version 5

# A space separated list of all the IPs on the internet, I trust
office="my-office-pc.example.com"

# The IP address of this Linux and LAN for the rest of the world
public_ip="1.2.3.4"

# My LAN. Everything is allowed here.
interface eth0 lan
    policy accept    # The default is 'drop'.

# Make sure the traffic coming in, comes from valid Internet IPs,
# and that is targeting my public IP
interface ppp+ internet src not "$UNROUTABLE_IPS" dst "$public_ip"
    # Protect me from various kinds of attacks.
    protection strong

    # Public servers.
    server smtp accept
    server http accept
    server ftp accept
    server ssh accept src "$office"

    # Make sure idents do not timeout.
    server ident reject with tcp-reset

    # This is also a workstation.
    client all accept

# Route the LAN requests to the internet.
router lan2internet inface eth0 outface ppp+
    # Masquerading on outface.
    masquerade

    # Route all requests from inface to outface
```

```
# and their replies back.
route all accept
```

Comandos

- **Comandos principales:**

interface, controla lo que puede hacer el host donde está corriendo el firewall.

router, controla el trafico que pasa a través del host del firewall.

Subcomandos:

policy, especifica la acción por defecto a llevar a cabo para los paquetes que no se ajusten a ninguna regla.

protection, añade protección extra a *interfaces* y *routers.*

server, acepta peticiones para el servicio o servicios especificados.

client, acepta respuestas para el servicio o servicios especificados.

route, enruta trafico.

group, agrupa un numero de servicios con los mismos parámetros opcionales de reglas.

- **Comandos de ayuda**

blacklist, establece una lista negra unidireccional o bidireccional.

dnat, establece una regla *Destination NAT* para el tráfico enrutado.

dscp, establece el campo DSCP en la cabecera de los paquetes, a un valor en crudo o a una clase DiffServ.

ecn_shame, deshabilita ECN para todos los hosts en la lista *ECN Shame.*

iptables, añade algunos comandos de iptables personalizados al firewall.

mac, establece una dirección mac fuente con una IP asociada.

mark, marca el trafico para tratarlo con herramientas de formateado de tráfico.

masquerade, activa el *masquerading (NAT)* a la salida de un interfaz de red.

nat, establece una regla *NAT* para el tráfico enrutado.

redirect, establece una regla de redirección de puerto.

snat, establece una regla *Source NAT* para el tráfico enrutado.

tcpmss, fija el *MSS* de paquetes *TCP SYN* para routers.

tos, fija el campo de *Type of Service (TOS)* en las cabeceras de paquetes.

transparent_proxy, establece un proxy TCP transparente corriendo en el host del firewall.

transparent_squid, establece un proxy web transparente corriendo en el host del firewall.

version, se requiere una version específica de FireHOL

Acciones:

accept, permite al tráfico alcanzar su destino.

reject, no permite al tráfico pasar, pero manda un mensaje de rechazo al remitente.

drop, descarta el tráfico sin mandar nada de vuelta al remitente.

deny, es un alias de drop.

return, regresa al procesado del flujo padre de reglas.

mirror, manda el tráfico de vuelta al remitente, al puerto de destino.

redirect, redirecciona el tráfico a otro puerto de localhost.

- **Parámetros opcionales de las reglas:**

src, define la fuente del tráfico.

dst, define el destino del tráfico.

inface, define la interfaz de red por la que el tráfico es recibido.

outface, define la interfaz de red por la que el tráfico es enviado.

physin, define la interfaz de red física (para bridges) por la que el tráfico es recibido.

physout, define la interfaz de red física (para bridges) por la que el tráfico es recibido.

custom, pasa algunos parámetros específicos a las sentencias iptables generadas.

log, escribe algo en el *syslog* cuando el tráfico coincide.

loglimit, escribe algo (limitado) en el *syslog* cuando el tráfico coincide.

proto, especifica un protocolo especifico.

limit, limita la frecuencia con que el tráfico es comparado con las reglas.

sport, especifica el puerto fuente.

dport, especifica el puerto de destino.

uid, usuario, especifica el usuario que manda tráfico.

gid, grupo, especifica el grupo de usuario que manda tráfico.

pid, proceso, especifica el ID del proceso que manda tráfico.

sid, sesión, especifica el ID de sesión que manda tráfico.

cmd, comando, define el nombre del comando que manda tráfico.

mac, si coincide la dirección *MAC* de los paquetes.

mark, si coincide el identificador *MARK ID* de los paquetes.

tos, si coincide el tipo de servicio *(TOS)* de los paquetes.

dscp, si coincide el valor en crudo de *DSCP* o el valor de la clase *DiffServ* de los paquetes.

Servicios

Mostramos una lista de servicios definidos mediante nombres en fireHOL:

AH, all, amanda, any, anystateless, apcupsd, apcupsdnis, aptproxy, asterisk, cups, custom, cvspserver, darkstat, daytime, dcc, dcpp, dhcp, dhcprelay, dict, distcc, dns, echo, emule, eserver, ESP, finger, ftp, gift, giftui, gkrellmd, GRE, h323, heartbeat, http, https, hylafax, iax, iax2, icmp, ICMP, icp, ident, imap, imaps, irc, isakmp, jabber, jabberd, ldap, ldaps, lpd, microsoft_ds, mms, msn, multicast, mysql, netbackup, netbios_dgm, netbios_ns, netbios_ssn, nfs, nis, nntp, nntps, ntp, nut, nxserver, oracle, p2p, ping, pop3, pop3s, portmap, postgres, pptp, privoxy, radius, radiusold, radiusoldproxy, radiusproxy, rdp, rndc, rsync, rtp, samba, sip, smtp, smtps, snmp, snmptrap, socks, squid, ssh, stun, submission, sunrpc, swat, syslog, telnet, tftp, time, timestamp, upnp, uucp, vmware, vmwareauth, vmwareweb, vnc, webcache, webmin, whois, xdmcp.

G. Detalles del detector de intrusiones Snort

G.1. Opciones de ejecución de Snort

USAGE: snort [-options] <filter options>
Options:
-A Set alert mode: fast, full, console, or none (alert file alerts only)
"unsock" enables UNIX socket logging (experimental).
-b Log packets in tcpdump format (much faster!)
-c <rules>Use Rules File <rules>
-C Print out payloads with character data only (no hex)
-d Dump the Application Layer
-D Run Snort in background (daemon) mode
-e Display the second layer header info
-f Turn off fflush() calls after binary log writes
-F <bpf>Read BPF filters from file <bpf>
-g <gname>Run snort gid as <gname>group (or gid) after initialization
-h <hn>Home network = <hn>
-i <if>Listen on interface <if>
-I Add Interface name to alert output
-k <mode>Checksum mode (all,noip,notcp,noudp,noicmp,none)
-l <ld>Log to directory <ld>
-L <file>Log to this tcpdump file
-m <umask>Set umask = <umask>
-n <cnt>Exit after receiving <cnt>packets
-N Turn off logging (alerts still work)
-o Change the rule testing order to Pass|Alert|Log
-O Obfuscate the logged IP addresses
-p Disable promiscuous mode sniffing
-P <snap>Set explicit snaplen of packet (default: 1514)
-q Quiet. Don't show banner and status report
-r <tf>Read and process tcpdump file <tf>
-R <id>Include 'id' in snort_intf<id>.pid file name
-s Log alert messages to syslog
-S <n=v>Set rules file variable n equal to value v
-t <dir>Chroots process to <dir>after initialization
-T Test and report on the current Snort configuration
-u <uname>Run snort uid as <uname>user (or uid) after initialization
-U Use UTC for timestamps
-v Be verbose
-V Show version number
-w Dump 802.11 management and control frames
-X Dump the raw packet data starting at the link layer
-y Include year in timestamp in the alert and log files
-z Set assurance mode, match on established sesions (for TCP)
-? Show this information
<*Filter Options*> are standard BPF options, as seen in TCPDump

G.2. Instalación y configuración de Snort

Instalación
Instalaremos los paquetes precompilados de Snort con la herramienta apt-get:

apt-get -s install snort

Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Se instalarán los siguientes paquetes extras:
libnet0 libpcap0.8 snort-common snort-rules-default
Paquetes recomendados
snort-doc oinkmaster
Se instalarán los siguientes paquetes NUEVOS:
libnet0 libpcap0.8 snort snort-common snort-rules-default

Junto con los paquetes snort-common y snort se instalan los paquetes de la biblioteca libpcap y libnet.

Vemos que se instalará también un paquete de reglas desarrollado por la comunidad Snort, totalmente funcionales. Estas reglas pueden servir de base para desarrollar reglas propias más complejas o ajustarlas a las necesidades de cada escenario.

Lista de archivos de reglas de Snort

/etc/snort/rules/attack-responses.rules
/etc/snort/rules/backdoor.rules
/etc/snort/rules/bad-traffic.rules
/etc/snort/rules/chat.rules
/etc/snort/rules/ddos.rules
/etc/snort/rules/deleted.rules
/etc/snort/rules/dns.rules
/etc/snort/rules/dos.rules
/etc/snort/rules/experimental.rules
/etc/snort/rules/exploit.rules
/etc/snort/rules/finger.rules
/etc/snort/rules/ftp.rules
/etc/snort/rules/icmp-info.rules
/etc/snort/rules/icmp.rules
/etc/snort/rules/imap.rules
/etc/snort/rules/info.rules
/etc/snort/rules/local.rules
/etc/snort/rules/misc.rules
/etc/snort/rules/multimedia.rules
/etc/snort/rules/mysql.rules
/etc/snort/rules/netbios.rules
/etc/snort/rules/nntp.rules
/etc/snort/rules/oracle.rules
/etc/snort/rules/other-ids.rules
/etc/snort/rules/p2p.rules
/etc/snort/rules/policy.rules
/etc/snort/rules/pop2.rules
/etc/snort/rules/pop3.rules
/etc/snort/rules/porn.rules
/etc/snort/rules/rpc.rules
/etc/snort/rules/rservices.rules
/etc/snort/rules/scan.rules
/etc/snort/rules/shellcode.rules

/etc/snort/rules/smtp.rules
/etc/snort/rules/snmp.rules
/etc/snort/rules/sql.rules
/etc/snort/rules/telnet.rules
/etc/snort/rules/tftp.rules
/etc/snort/rules/virus.rules
/etc/snort/rules/web-attacks.rules
/etc/snort/rules/web-cgi.rules
/etc/snort/rules/web-client.rules
/etc/snort/rules/web-coldfusion.rules
/etc/snort/rules/web-frontpage.rules
/etc/snort/rules/web-iis.rules
/etc/snort/rules/web-misc.rules
/etc/snort/rules/web-php.rules
/etc/snort/rules/x11.rules

G.3. Parámetros de configuración de Snort

Como ya se comentó anteriormente en este documento, el archivo de configuración de Snort, */etc/snort/snort.conf* tiene cuatro partes bien diferenciadas:

- Variables de red.
- Configuración de preprocesadores.
- Configuración de plugins de salida.
- Personalización de reglas del motor de detección.

En la primera sección, **variables de red**, se da valor a ciertas variables, por ejemplo:

```
var HOME_NET any
var EXTERNAL_NET !$HOME_NET
var HTTP_SERVERS $HOME_NET
var HTTP_PORTS 80
```

y una variable importante es la que establece la ruta al directorio donde deben estar los archivos que contienen las reglas de detección:

```
var RULE_PATH /etc/snort/rules
```

En la siguiente sección, la de **configuración de los preprocesadores**, se dan las opciones apropiadas a cada preprocesador con sentencias de la forma:

```
preprocessor <nombre_del_preprocesador>: <opciones_de_configuración>
```

por ejemplo:

```
preprocessor flow: stats_interval 0 hash 2
```

Enumeramos a continuación algunos de los preprocesadores configurables en esta sección: flow, frag2, stream4, http_inspect, rpc_decode, bo, telnet_decode, flow-portscan, arpspoof, perfmonitor.

Para obtener información sobre los parámetros que puede tomar cada preprocesador remitimos al lector al archivo de configuración snort.conf, donde aparecen comentada tanto la función de cada preprocesador como los parámetros posibles.

La tercera sección aborda la **configuración de los plugins de salida**, al igual que en la sección anterior cada plugin se configura en una linea, especificando los parámetros oportunos, el formato es el siguiente:

output <nombre_del_plugin>: <opciones_de_configuración>

Los módulos de salida corren cuando los subsistemas de alertas y logging son llamados, después de los preprocesadores y el motor de detección. Se pueden especificar múltiples plugins de salida, se apilaran y se llamarán secuencialmente cuando ocurra un evento, como el sistema estandar de logging y alerta, los plugins de salida mandarán por defecto los resultados a */var/log/snort.*

A continuación comentamos los posibles plugins de salida:

alert_syslog: este modulo manda las alertas al syslog, también permite al usuario especificar la facilidad de logging y la prioridad, dando mayor flexibilidad a los usuarios.

alert_fast: este mostrará la salida en un formato simple de una linea en el archivo de salida especificado. Es más rápido que el método *full* ya que no imprime las cabeceras de los paquetes en el archivo de salida.

alert_full: mostrará los mensajes de alerta de Snort con la cabecera completa de los paquetes, las alertas se guardan en el directorio por defecto */var/log/snort/* y se creará un directorio por cada IP, estos ficheros contendrán los volcados de los paquetes que disparan las alertas. Este método ralentiza el funcionamiento de Snort considerablemente y está desaconsejado para sistemas con mucho tráfico.

alert_unixsock: se crea un socket unix al que se mandan las alertas y cualquier programa o procesador podrá recibir las alertas en tiempo real.

log_tcpdump: este módulo guardara los paquetes capturados en el formato tcpdump, es útil si se quiere procesar posteriormente el tráfico capturado, al haber gran variedad de programas que trabajan con este formato.

database: manda los datos a una base de datos sql, los argumentos de este plugin son el nombre de la base de datos y una lista de parámetros con el formato: parámetro = argumento, tales como: nombre de host, port, dbname, user, password, sensor_name, encoding, etc.

csv: permite escribir los datos de alerta en un formato fácil de exportar a una base de datos,

unified: pretende ser el plugin más rápido de Snort, guarda tanto las alertas como los paquetes en un formato binario descrito en *spo_unified.h.*

log null: en determinadas circunstancias es útil crear ciertas reglas que alerten de cierto tipo de tráfico pero que no almacenen las cabeceras de los paquetes que dispararon las alertas.

En la última sección del archivo de configuración se agregan las **reglas de detección**, se incluyen los ficheros de reglas con el parámetro *include*, seguido del nombre del fichero de reglas en cuestión, por ejemplo:

include $RULE_PATH/scan.rules

H. Descripción detallada de diseño de interfaces

H.1. Formularios de configuración del módulo Proxy (Squid)

H.1.1. Formulario "Control del Módulo"

Presentamos en primer lugar el formulario que aglutina las acciones de *control* del módulo.

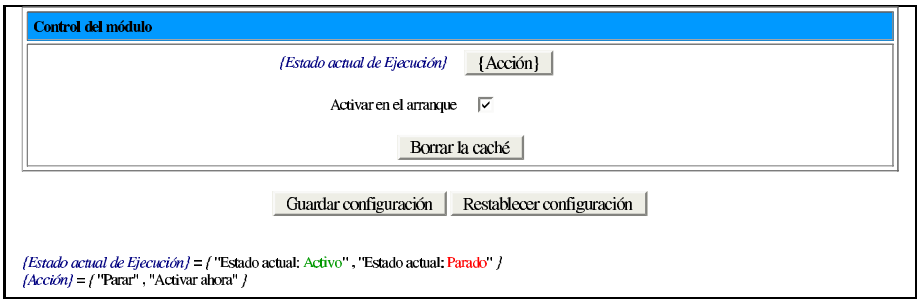

Figura 16: Módulo 'Proxy (Squid)' – Control del Módulo

Aquí se realizan las acciones de configuración que afectan directa o indirectamente a la ejecución del módulo, tales como: la activación del módulo, la desactivación, o la realización de una acción inmediata.

En este caso, con el Proxy Squid, nos encontramos con las opciones de:

{Acción} *(botón)* Esta etiqueta está sobre el botón que realiza la acción *"principal"* sobre el módulo, consistente en el control inmediato sobre su ejecución. Puede adquirir cualquiera de los valores que indica la etiqueta en su leyenda (ver fig. 16), mostrándolo sobre el botón, que vendrá dado dependiendo del estado actual de ejecución del servicio `squid` (es decir, si está activado, o parado). Al pulsar sobre él, se realiza la acción que indica la etiqueta en cada momento: ***Parar*** detiene el servicio `squid`, ***Activar ahora*** pone en marcha el servicio. Ambas acciones son inmediatas.

En el caso de la etiqueta que se muestra a la izquierda del botón, *{Estado actual de ejecución}*, muestra al usuario cuál es el estado en el que se encuentra el servicio `squid` en el momento de cargar este formulario. De esta forma, puede conocer, de un rápido vistazo, si el servicio se está ejecutando o no.

Activar en el arranque *(casilla de verificación)* Al marcar *"Activar en el arranque"*, y una vez guardada la configuración, automáticamente se reconfigura el sistema para que el servicio `squid` se active en el arranque de la máquina. Esto no provocará la ejecución inmediata del servicio, esto es, el usuario aún deberá pulsar el botón *{Acción}* para arrancar el servicio, o pararlo, de forma efectiva en este instante.

En el caso de no querer que `squid` se active en el arranque, se debe dejar desmarcada esta casilla. Esto provocará que el sistema se configure de manera que el servicio no se ejecute de forma automática. Sin embargo, se puede activar o parar en cualquier instante.

Borrar la caché *(botón)* Al pulsar sobre este botón, provocará el limpiado de la caché en disco de `squid`, de forma inmediata. Se trata de una acción de *control* y por tanto, está ubicada también en este formulario.

H.1.2. Formulario "Configuración básica"

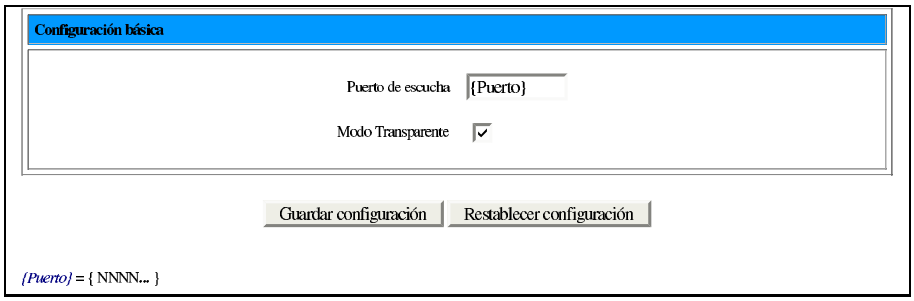

Figura 17: Módulo 'Proxy (Squid)' – Configuración básica

Mediante este formulario, se proporciona al usuario el conjunto básico (minimizado) de opciones de configuración de `squid`. En este caso, la configuración básica del servicio (es decir aquello que es indispensable configurar para asegurar su funcionalidad mínima) consiste en: la definición del puerto TCP de escucha del servicio, y si va a actuar en modo *Transparente*.

El significado de este formulario es que el usuario debe configurar estos parámetros para que el módulo actúe de la manera que él quiere; configurando únicamente este formulario, se garantiza que el servicio funcionará, con un conjunto básico de funcionalidades, que son las que se espera de un *Proxy* del tipo más *"simple"* posible.

Puerto de escucha *(campo de texto)* Se introduce aquí el puerto TCP en el cual el *demonio* `squid` escuchará para aceptar conexiones de los clientes. Como tal, puede tener valores comprendidos entre *1* y *65535*. Sin embargo, se recomienda un valor como `8080` ó `3128`, que son los habituales para este tipo de servicio de *Proxy*.

Modo Transparente *(casilla de verificación)* Si se marca esta casilla, la forma en que `squid` interpreta las solicitudes de los clientes cambia ligeramente. Ahora, se activa lo que se llama el modo de *aceleración*. Consiste en que las peticiones HTTP de los clientes (aparentemente directas hacia el servidor de destino) se interpretan de manera que provoquen una petición de `squid` hacia el servidor (aunque las peticiones de los clientes no tengan el formato propio de las peticiones de Proxy). Mediante este modo, los clientes no tienen forma de saber que están pasando a través de un Proxy para llegar al servidor (por eso se llama *Transparente*). Es necesaria la colaboración de *iptables*[7] para lograr interceptar el tráfico y redirigir las peticiones de los clientes a *squid*, sin que sea advertido por los mismos.

[7] En nuestro caso, no se llama directamente a *iptables*, sino que se emplea la interfaz de abstracción que nos proporciona FireHOL. Por tanto, la acción de introducir estas *reglas de iptables* en la pila de TCP/IP del sistema, se delega en el módulo *"Cortafuegos (FireHOL)"*, mediante el mecanismo de comunicación entre módulos implementado.

H.1.3. Formulario "Proxy Padre"

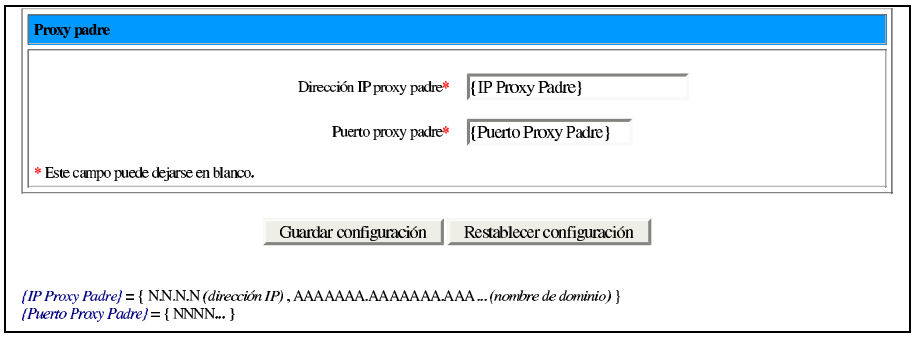

Figura 18: Módulo 'Proxy (Squid)' – Proxy Padre

Se configuran aquí los parámetros que permiten el reenvío de las peticiones de Proxy a un servidor de acceso perimetral a nivel de aplicación (es decir, otro Proxy) que se encuentra por "encima" de nuestro Cortafuegos (en tanto que nuestro Cortafuegos depende de este Proxy para acceder a la red exterior). Es por ello que se le llama *"Proxy Padre"*.

Estos parámetros son opcionales, por tanto, el usuario puede dejarlos en blanco. En ese caso, no se configurará Proxy Padre alguno (se entenderá entonces que Squid tiene acceso directo a través de la red a cualquier recurso externo). En caso de que el usuario rellene sólo uno de los parámetros, quedando el otro en blanco, no se considerará válido, indicándose al usuario mediante un mensaje de error.

Dirección IP proxy padre *(campo de texto)* Se configura aquí la dirección IP (también se admite un nombre de *host*, aunque no es lo habitual) del servidor de seguridad perimetral que actúa de Proxy Padre, al que se dirigirán todas las peticiones de Squid sobre recursos Web solicitados por los clientes, cada vez que sea necesario.

Puerto proxy padre *(campo de texto)* El puerto TCP del Proxy Padre hacia el cual Squid intentará conectar para enviar las solicitudes de Proxy.

H.1.4. Formulario "Control de acceso"

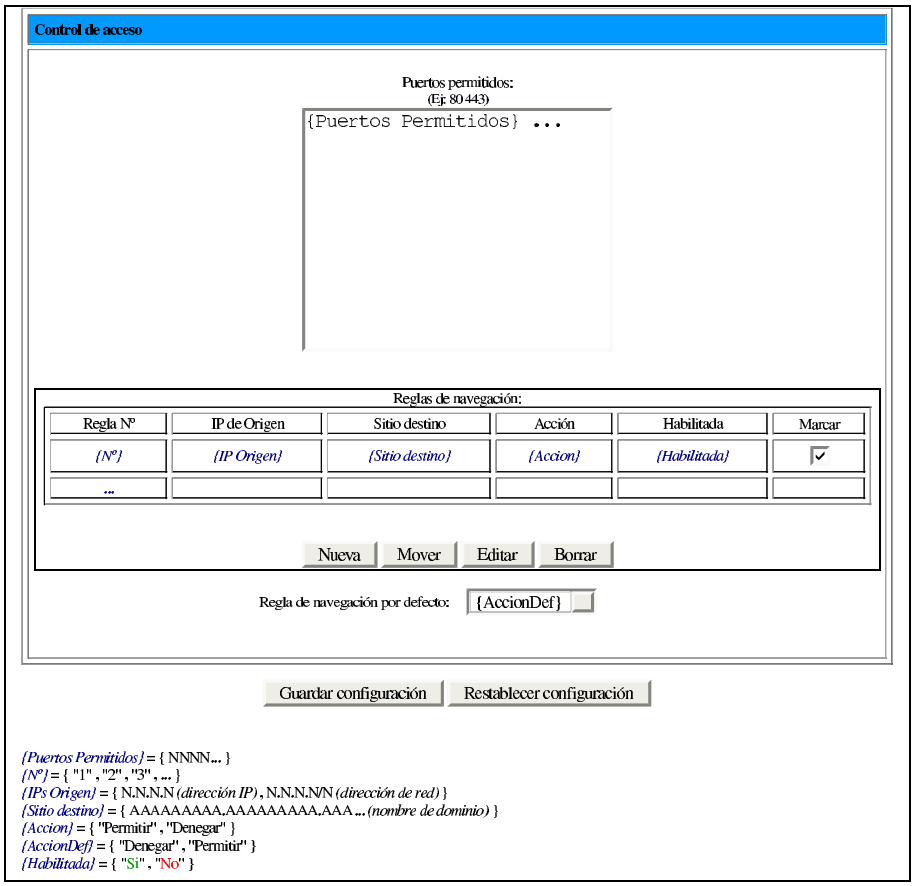

Figura 19: Módulo 'Proxy (Squid)' – Control de acceso

Se agrupan en este formulario aquellas configuraciones que afectan a los permisos de navegación. Esto es, mediante la configuración aquí indicada, se decide si un usuario dado tiene permiso o no para realizar una solicitud al Proxy.

Se ha dividido en 2 'secciones': *Puertos Permitidos*, y *Reglas de Navegación*, según se muestra a continuación:

Puertos permitidos *(cuadro de texto)* En este cuadro de texto el usuario puede introducir cuantos puertos desee a los cuales se permite la navegación Web. Es decir, se trata de una lista de puertos TCP, hacia los cuales se permite a los usuarios de la red realizar solicitudes Web. Esto determinará los puertos de destino hacia los que el Proxy podrá establecer una conexión (se trata de una medida de seguridad, que evita que se emplee el Proxy para realizar conexiones a puertos de destino no deseados). Se especifican mediante una serie de valores separados por blancos o retorno de carro. Por ejemplo, una lista de valores válida y quizás habitual sería:

```
80
443
8080
8888
```

Esto permitirá la navegación Web hacia los puertos *(estándares)* 80 y 443, así como a los puertos *(alternativos)* 8080 y 8888.

Aquí es necesario advertir que, como regla general, no se permitirá la navegación hacia un puerto que no se indique aquí expresamente. Por tanto, ésto no se ve afectado por el valor indicado en "*Regla de navegación por defecto*", ya que conceptualmente su función es diferente.

Reglas de navegación Se muestra aquí una lista tabulada de las reglas de navegación configuradas. Estas reglas son una manera más *fina* de indicar permisos de navegación, y en este caso, recogen *quién* puede acceder a *dónde*. Se tiene por tanto granularidad a nivel de *host*.

Cada regla tiene un nº de órden, ya que éste es importante a la hora de establecer prioridades o excepciones. Se pueden indicar direcciones IP de origen (o direcciones de red con máscara de bits), y sitios de destino (es decir, nombres de dominio). Así mismo, puede tratarse de una regla que permita o deniega el acceso, según se configure la misma (en el campo *"Acción"*). De esta forma, se tiene un gran control sobre qué usuarios de la red (individualmente, o agrupados en subredes lógicas) pueden acceder a qué servidores Web (indicados mediante el nombre de dominio). Como opción, también se puede indicar como sitio destino ***"Cualquier sitio"***, de manera que englobamos en una sóla regla, un permiso que afecte a todo un grupo de usuarios, o usuario en particular, independientemente del destino al que intente acceder (es decir, permitimos o denegamos el acceso de navegación en general, a un grupo de usuarios). Por último, cada regla se puede *habilitar* o no, esto es: una regla no habilitada es una regla que, aún permaneciendo definida, no resulta efectiva.

Mediante los botones *"Nueva"*, *"Mover"*, *"Editar"* o *"Borrar"* se puede operar sobre las reglas, mostrándose al pulsarlos un formulario auxiliar que permite realizar la operación indicada sobre la regla *marcada*[8].

Regla de navegación por defecto *(lista de selección)* Permite elegir entre *"Denegar"* o *"Permitir"*. Es en efecto la *última regla de navegación*, que se aplica una vez que se ha recorrido la lista al completo y no se ha encontrado una concordancia. Si el usuario prefiere que se deniegue por defecto la navegación excepto para aquellos puestos o destinos indicados expresamente, entonces pondrá aquí *"Denegar"*. Si por el contrario, prefiere permitir la navegación como regla general excepto en aquellos casos que se deniegue expresamente, entonces pondrá aquí *"Permitir"*.

[8] Casilla "Marcada". En el caso de estar marcadas más de una regla, se operará sobre la primera de ellas, esto es, la casilla marcada que tiene un nº de órden inferior.

Formulario auxiliar "Control de acceso - Mover Regla"

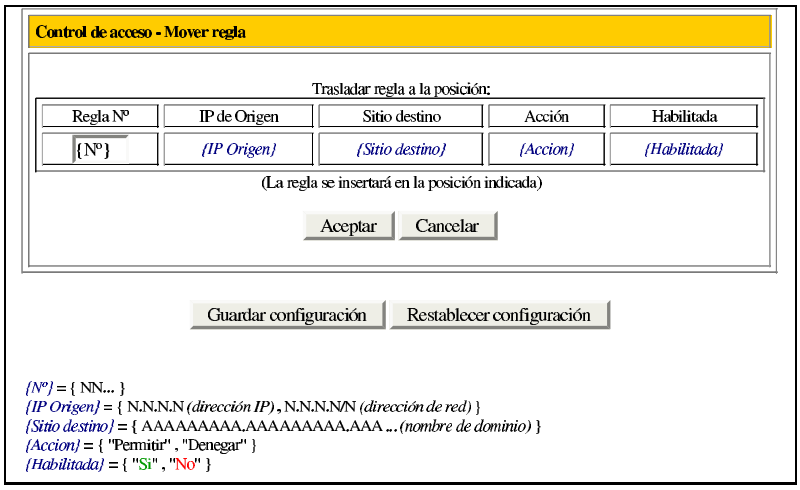

Figura 20: Módulo 'Proxy (Squid)' – Control de acceso - Mover Regla

Se accede a este formulario una vez que el usuario ha pulsado sobre el botón *"Mover"*, en la lista de Reglas de navegación del formulario *Control de acceso*. Su función es la de permitir al usuario modificar el nº de órden en el que aparece una regla de navegación dada, para cambiar de esta forma su prioridad, a la hora de buscar una concordancia. Suele ser el caso cuando existen reglas cuyos ámbitos de actuación no son conjuntos disjuntos (interesa decidir la prioridad que tendrá un grupo de actuación sobre otro), como por ejemplo: cuando se definen reglas con subredes superpuestas, o dominios y subdominios de estos.

El formulario muestra todos los detalles de la regla (nº, IP de origen, Sitio destino, Acción...) pero el único campo editable es el nº de órden. Los demás aparecen meramente como información adicional (principalmente, para evitar confusiones).

Regla Nº *(campo de texto)* Se introduce el nº de órden en el que el usuario desea que aparezca la regla de navegación.[9] Nunca se *machacará* una regla aunque se sitúe en una posición ya *ocupada*. Simplemente, se insertará en dicha posición, incrementándose la regla posterior (o posteriores) en una posición.

Aceptar *(botón)* Se valida el formulario realizando la acción de *Mover* deseada, volviendo al formulario principal *"Control de acceso"*.

Cancelar *(botón)* Se cierra el formulario (no se realiza la operación), volviendo al formulario principal *"Control de acceso"*.

[9] En el caso de mover una regla a una posición posterior a la actual, ninguna regla ocupará el nº de órden que quedará vacío (es decir no se desplazarán), de forma que quedará vacante esta posición para introducir otra regla. La razón de hacerlo así, es para asegurar que el nº de órden que introduce el usuario será *efectivamente* el nº de órden en el que se situará la regla.

Formulario auxiliar "Control de acceso - Nueva / Editar Regla"

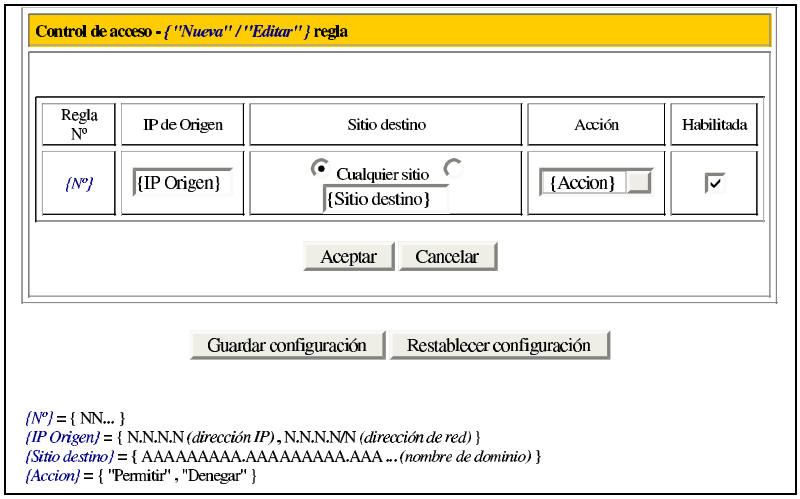

Figura 21: Módulo 'Proxy (Squid)' – Control de acceso - Nueva / Editar Regla

Este formulario aparece cuando el usuario ha pulsado sobre uno de los botones *"Nueva"* o *"Editar"*, de la lista de Reglas de navegación del formulario *Control de acceso*. Se trata de un formulario similar al anterior *("Mover Regla")* pero esta vez todos los campos son editables (a excepción del nº de órden).

Este formulario será reutilizable tanto si se desea crear una nueva regla, como si se desea editar las propiedades de una existente. La diferencia será que en el primer caso, los campos editables aparerán vacíos, mientras que en segundo, tendrán los correspondientes valores actuales para la regla en cuestión.

En el caso del nº de órden de la regla, que se muestra a modo de etiqueta en *{Nº}*, en la 1ª columna, será proporcionado por el sistema, y por tanto no es modificable por el usuario en este formulario. En el caso de tratarse de una **regla nueva**, este valor se establecerá a *n+1*, siendo *n* el nº de órden de la última regla que existiese definida.

El resto de los campos se definen como sigue:

IP de Origen *(campo de texto)* La dirección IP o rango de red del usuario que realiza la petición. Se puede especificar en cualquiera de los formatos que admite la directiva **'acl src'** de `squid`, por ejemplo: 10.0.0.0/8 (dirección de red/máscara de bits), 192.168.100.1-192.168.100.99 (rango de direcciones IP).

Sitio destino *(botón radial / campo de texto)* Se selecciona el *host* de destino que se solicite al Proxy. Se puede indicar *"Cualquier destino"* para que esta regla concuerde con cualquier destino posible (siendo por tanto este dato irrelevante), o bien introduciendo (en el campo de texto) el nombre de dominio al que se pretende acceder. El formato permitido para este último, es el mismo que admite la directiva **'acl dstdomain'** de `squid`, por ejemplo: `.dominio.net` (parte de un dominio), `www.debian.org` (un dominio completamente cualificado).

Acción *(lista de selección)* La acción de navegación que se aplicará si se obtiene una concordancia con la regla: *Permitir* (para permitir la navegación) o *Denegar* (para denegarla).

Habilitada *(casilla de verificación)* Estado en el que la regla se almacenará inicialmente en la lista de Reglas de Navegación, una vez que se pulse el botón *Aceptar*.

Aceptar *(botón)* Se valida el formulario realizando la inserción de la regla en la posición indicada (o su sustitución, en el caso de estar editando una regla ya existente), volviendo al formulario principal *"Control de acceso"*.

Cancelar *(botón)* Se cierra el formulario (no se realiza la operación), volviendo al formulario principal *"Control de acceso"*.

H.1.5. Formulario "Otros parámetros"

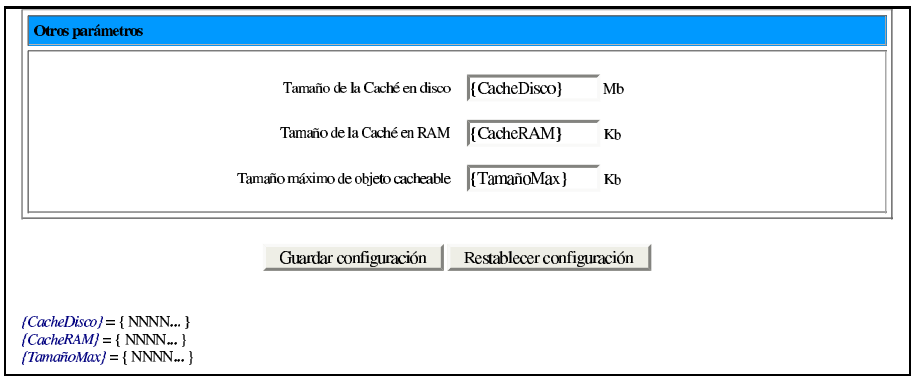

Figura 22: Módulo 'Proxy (Squid)' – Otros parámetros

En este formulario se dá la opción de configurar algunos otros parámetros de configuración, de carácter más *"avanzado"* que los que aparecen en los formularios anteriores. Son parámetros que normalmente no será necesario variar, ya que el funcionamiento básico del módulo se regula con el resto de los parámetros, pero que están aquí para realizar un ajuste fino de la configuración, variando parámetros a un nivel más bajo.

Tamaño de la Caché en disco Se especifica, en unidades de Megabyte, el espacio máximo de disco que `squid` utilizará para almacenar la caché de disco, esto es, el almacén de recursos Web que se han solicitado previamente y permanecen cacheados en disco, con objeto de acelerar la navegación.

Tamaño de la Caché en RAM Similar al anterior, pero esta vez regula el espacio de RAM, especificado en Kbytes, que se utilizará para almacenar dichos recursos Web solicitados previamente, con el mismo propósito (acelerar la navegación). La RAM es un medio más rápido que el disco, pero se trata de un recurso más escaso, con lo que el tamaño de esta caché se indicará normalmente con un valor inferior al de la anterior (la caché de disco).

Tamaño máximo de objeto cacheable Se define con este parámetro el tamaño máximo que tendrá un recurso descargado de la Web para que pueda introducirse en la caché.

H.1.6. Reflexión acerca de los formularios de configuración del módulo Proxy

Se ha intentado que los parámetros configurables aquí sean lo más genéricos posibles, y constituyan una configuración básica de la funcionalidad esperada para un software de Proxy, sin llegar a ahondar demasiado en las características especiales del software Squid en concreto, ni en sus peculiaridades. No obstante, por motivos de simplicidad, se transmite toda la potencia a la hora de especificar ciertos valores que permite dicho software, como por ejemplo en el caso de las direcciones

IP (donde se permite especificar IP's individuales, rangos de IP's consecutivas, máscaras de bits de formato numérico como /28, o máscaras de red de formato IP como /255.255.255.0).

Muchos de los parámetros de configuración constituyen una correspondencia biunívoca con las opciones de configuración del fichero `squid.conf`, sin embargo en algunos casos, se trata de una abstracción más elaborada, llevando la especificación de la configuración de dichos parámetros a un nivel más alto. Es el caso del *"Modo Transparente"*, y de la lista de *Reglas de Navegación.*

Acerca del ***"Modo Transparente"***, tiene su justificación en que se trata de un modo de navegación comúnmente utilizado en múltiples redes locales, metropolitanas y de área extensa, así como incluso por proveedores de acceso a Internet. Consiste en establecer un servidor Proxy en el camino de enrutado de las peticiones Web, redirigiendo dichas peticiones (con ayuda de mecanismo de traducciones de direcciones y redirecciones de puertos) para que sean *recibidas* por el servidor Proxy, en lugar de ser enrutadas como cualquier otro tráfico IP hacia el servidor de destino. El servidor Proxy interpreta la petición originalmente dirigida al servidor como si fuese una petición de Proxy, con lo que es éste (y no el cliente) el que finalmente contacta con el servidor de destino. Una vez descargado el contenido Web solicitado, y almacenado en la caché del Proxy, es devuelta hacia el cliente en respuesta a su solicitud (de nuevo empleando el mecanismo de traducción de direcciones, suplantando la identidad del servidor destino).

En nuestro caso, la implementación del *Modo Transparente* con `squid` e `iptables` pasa, por un lado, por instruir a `squid` para que interprete las peticiones dirigidas originalmente de forma directa a un servidor (peticiones HTTP del tipo '`GET` /url...') como si fuesen peticiones válidas de Proxy (que tienen otro formato: '`GET` http://host/url...'), y por otro, generar las correspondientes reglas de NAT de iptables que intercepten el tráfico Web y lo redirijan al puerto donde `squid` está escuchando en la máquina local.

Squid posee opciones de configuración que hacen la interpretación de peticiones HTTP como peticiones de Proxy, mediante lo que se llama el *modo acelerado* (ya que esta funcionalidad se ideó, en un principio, para implementar un proxy inverso a modo de *'caching front-end'* de un servidor o servidores Web (que constituirían el *back-end*), evitando mediante la técnica de caché que tuviesen que servir todas las peticiones, y acelerando por tanto la capacidad de servicio de páginas estáticas o dinámicas del conjunto). En cuanto a las reglas de NAT, no será necesario llegar al bajo nivel de `iptables` para esta tarea: gracias a *FireHOL*, la interfaz de abstracción que empleamos en el Módulo de configuración del Cortafuegos, bastará con una sóla órden de configuración (`transparent_squid`) para generar las reglas correspondientes. Puede verse un ejemplo en la sección 8.3.2 en la página 33.

Por tanto, como vemos, la activación del *Modo Transparente* introduce tanto cambios en la configuración de *Squid*, como de otros módulos. Para lograr que el módulo de *Squid* comunique a los demás estos cambios, se emplea el mecanismo de comunicación entre módulos implementado.

La activación del Modo Transparente influye además en la manera en que se interpretan algunas opciones de configuración: si está activo el Módulo *Filtro de Contenidos (DansGuardian)*, la activación del *Modo Transparente* tendrá un efecto ligeramente diferente al indicado: se tomará el puerto de escucha de *DansGuardian*, en lugar del de *Squid*, como destino para el reenvío de las peticiones interceptadas. La razón es simple: *DansGuardian* debe recibir antes que *Squid* dichas peticiones, por estar situado en este punto en el flujo de tráfico hacia el cliente. El número de puerto de escucha de `dansguardian` se obtendrá mediante el citado mecanismo de comunicación entre módulos.

El puerto de escucha de `squid` seguirá siendo el mismo (a él se redirigirán finalmente las peticiones), y se permite la utilización de Squid como un Proxy normal y corriente (configurándolo en los clientes Web) además de la navegación transparente. Los *puertos de navegación permitidos* (*"Puertos permitidos"* en el formulario de *"Control de acceso"*) seguirán teniendo el efecto deseado (sólo se podrá navegar *hacia* estos puertos, debido a las `acl`'s generadas[10]), sin embargo, el único

[10] Véase el apartado H.1.7

puerto de navegación realmente *transparente* seguirá siendo el 80 (el estándar de HTTP). Esto es debido a, por un lado, una limitación de esta versión de FireHOL (1.214), que no permite especificar otro puerto en la sentencia `transparent_squid`, y por otro, a que no se puede saber cuáles de los *puertos de navegación permitidos* se emplearán para conexiones HTTPS (HTTP sobre SSL), los cuales no deben ser redirigidos al Proxy automáticamente ya que la navegación HTTPS no es compatible con el mecanismo de Proxy Transparente.

En cuanto a la lista de ***Reglas de Navegación***, constituye un compromiso entre la potencia de especificación de *'acl's* de Squid, y la simplicidad conceptual a la hora de definir reglas de acceso con una interfaz gráfica desde el punto de vista del usuario. Se ha pretendido resolver la necesidad básica de especificación de permisos que puede tener un usuario administrador de red común (al que le interesa especificar permisos tanto por usuario de origen como por sitios Web de destino) mediante la introducción de reglas que recogen estos dos parámetros. El número de órden constituye una solución sencilla a la necesidad de especificar prioridades entre reglas (cuando dichos parámetros pueden englobar conjuntos no totalmente disjuntos) y por supuesto este valor de prioridad debe ser también totalmente controlable (mediante el mecanismo de *"Mover regla"*). El hecho de poder definir reglas que en un momento dado puedan estar *habilitadas* o no, responde también a una necesidad básica en la gestión habitual de un conjunto extenso de reglas, y a un entorno de red con necesidades cambiantes, que puede requerir reconfiguración de permisos bastante a menudo... Por último, la ***Regla de navegación por defecto*** evita la necesidad de incluir manualmente una última regla *'catch-all'* que englobe todos aquellos casos no explícitamente especificados.

H.1.7. Relación con los parámetros de configuración del software `squid`

Formulario "Configuración básica":

Campo	Parámetros	Comentarios
Puerto de escucha	`http_port`	Sólo se establecerá el puerto, no la dirección IP.
Modo Transparente	`httpd_accel_host` `httpd_accel_port` `httpd_accel_with_proxy` `httpd_accel_uses_host_header`	Se introducirán en estos parámetros los puertos de navegación especificados en *Control de acceso – Puertos permitidos*. Será necesario comunicar al Módulo Cortafuegos también este dato (ver la reflexión anteriormente comentada).

Formulario "Proxy padre":

Campo	Parámetros	Comentarios
Dirección IP proxy padre	`cache_peer`	Se utilizará para el valor de *"hostname"*.
Puerto proxy padre	`cache_peer`	Se utilizará para el valor de *"http_port"* (y *0 para "icp_port"*).

Formulario "Control de acceso":

Campo	Parámetros	Comentarios
Puertos permitidos	`acl` `http_access`	Se genera una **'acl'** para permitir la navegación sólo hacia los puertos de destino indicados. Si se activa el *Modo* Transparente, los valores aquí introducidos se reutilizan también para otros parámetros (ver *"Modo Transparente"*).
Regla de navegación por defecto	`acl` `http_access`	Se introducirá como una última regla *'catch-all'* automática.

Formulario auxiliar "Control de acceso – Mover regla":

Campo	Parámetros	Comentarios
Regla N°	`acl` `http_access`	Se reordena la lista de **'acl'**s así como las directivas **'http_access'** para que tengan efecto según el órden de prioridad indicado.

Formulario auxiliar "Control de acceso – Nueva / Editar regla":

Campo	Parámetros	Comentarios
IP de Origen	`acl` `http_access`	Se genera una **'acl'** para permitir la navegación sólo hacia los puertos de destino indicados. Si se activa el *Modo Transparente*, los valores aquí introducidos se reutilizan también para otros parámetros (ver *"Modo Transparente"*).
Sitio destino	`acl` `http_access`	Se genera una **'acl'** para permitir la navegación sólo hacia los nombres de host o direcciones IP de destino indicadas.
Acción	`http_access`	Se genera la directiva con el modificador **'allow'** o **'deny'** según la acción indicada por el usuario.
Habilitada	`acl` `http_access`	En el caso de no estar *habilitada*, se generan las directivas **'acl'** y **'http_access'** correspondientes para esta regla, pero se deshabilitan temporalmente.

Formulario "Otros parámetros":

Campo	Parámetros	Comentarios
Tamaño de la caché en disco	`cache_dir`	Se utiliza para el parámetro *'Mbytes'* de esta directiva.
Tamaño de la caché en RAM	`cache_mem`	Se introduce este valor tal cual (con el sujijo 'KB').
Tamaño máximo de objeto cacheable	`maximum_object_size`	Se introduce este valor tal cual (con el sujijo 'KB').

H.1.8. Importación/Exportación de parámetros

Se indican a continuación los parámetros *importables* (parámetros que puede necesitar *Squid* de otros módulos para ajustar su funcionamiento, o para hacerlo compatible con la funcionalidad común que se persigue) o *exportables* (parámetros que ofrecerá *Squid* a otros módulos que necesiten comunicarse con él, o simplemente para evitar al usuario la introducción de un mismo dato dos veces, en distintos módulos).

Parámetros importables:

Módulo	Parámetro	Descripción
Antivirus (ClamAV)	`activado`	En el caso de que el módulo Antivirus esté activado, se utilizará para *incorporar* esta funcionalidad a Squid; en caso contrario, se ignorará este módulo.
Antivirus (ClamAV)	-	Se añadirá a squid.conf la sentencia `redirect_program` que permite enviar a ClamAV las peticiones de Proxy para que éste escanee los ficheros descargados contra virus, antes de servirlos al cliente. Los parámetros de `redirect_program` serán fijos y por tanto no será necesario obtener este dato del módulo Antivirus. Así mismo, se añadirá la sentencia `redirect_children` en caso de no existir.

Parámetros exportables:

Parámetro	Descripción
`activado`	Se establecerá a `1` en el caso de que el módulo se haya configurado para activarse en el arranque (formulario **"Control del módulo"**), o al valor `0` en el caso contrario.
`puerto_escucha`	Puerto en el que `squid` recibirá las peticiones HTTP. Será el valor introducido en el campo ***Puerto de escucha*** del formulario **"Configuración básica"**.
`modo_transparente`	Tendrá el valor `1` en el caso de que se haya marcado la casilla ***Modo Transparente*** del formulario **"Configuración básica"**, o el valor `0` en caso contrario.
`direccion_proxy_padre`	Dirección IP (o nombre de host) del proxy padre configurado por el usuario. Se tomará del campo ***Dirección IP proxy padre*** del formulario **"Proxy Padre"**.
`puerto_proxy_padre`	Puerto configurado para el proxy padre (se toma del campo ***Puerto proxy padre*** del formulario **"Proxy padre"**).

H.2. Formularios de configuración del módulo Antivirus (ClamAV)

El módulo Antivirus se presenta, desde el punto de vista de la configuración por parte del usuario, como un módulo independiente, si bien en realidad, y como uno se puede imaginar, afecta en gran medida (y se ve afectado) por los demás módulos.

El módulo Antivirus filtrará el tráfico Web entrante *antes* de ser entregado a Squid, por tanto la configuración de este módulo afectará también a la configuración de Squid. Así mismo, un cambio en la configuración de Squid podrá afectar a este módulo. Todo ello se gestiona, de manera transparente al usuario, mediante el mecanismo de comunicación entre módulos, sin que tenga que preocuparse por ello; sino simplemente de establecer las opciones de configuración deseadas para este módulo.

A nivel de *paquetes de software*, como se ha visto, este módulo se compone de varios elementos claramente diferenciados (ClamAV, FreshClam, SCAVR), lo cual implica también varios ficheros de configuración separados. Sin embargo, la interfaz de configuración no diferenciará entre ellos, desde el punto de vista del usuario, actuando así de *wrapper* entre el software y el usuario. De nuevo, constituye una *capa de abstracción*, ocultando al usuario las complejidades de la configuración de más *bajo nivel*.

H.2.1. Formulario "Control del Módulo"

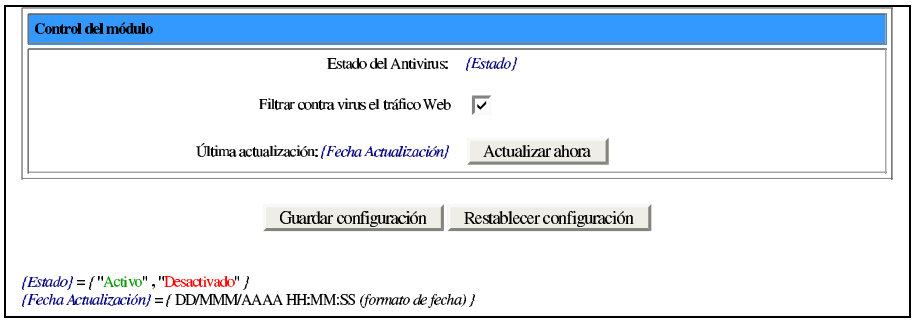

Figura 23: Módulo 'Antivirus (ClamAV)' – Control del módulo

En este formulario, como en los demás módulos, se aglutinan las acciones que afectan a la ejecución del módulo, controlando su ejecución.

A diferencia de los demás módulos, en este caso, puesto que el Antivirus *(ClamAV)* está "enganchado" al Proxy (*Squid* en nuestro caso), no tiene sentido "arrancar" o "parar" el servicio esta vez, de ahí la inexistencia de un botón para realizar esta acción. En lugar de esto, sí aparece la casilla de verificación *"Filtrar contra virus el tráfico Web"*, que efectivamente *engancha* el antivirus al Proxy (es decir, configura el redirector de Squid para que llame al antivirus) para permitir el filtrado contra virus de los ficheros descargados a través del proxy.

{Estado} *(etiqueta)* Muestra el estado actual del antivirus: *"Activo"* significa que el proxy Squid está configurado para filtrar contra virus el tráfico Web; *"Desactivado"* que no lo está.

Filtrar contra virus el tráfico Web *(casilla de verificación)* Activa o desactiva el servicio de Antivirus (tal y como se ha explicado anteriormente) para permitir que el proxy filtre contra virus el tráfico Web que se descarga a través de él. Su acción no es inmediata: será necesario *Guardar la configuración* (pulsando el botón correspondiente) para que se reconfigure el sistema al efecto.

{Fecha Actualización} *(etiqueta)* Muestra la última actualización que se ha realizado del patrón de virus. Este proceso de actualización es realizado periódicamente (según se haya configurado en el módulo) por el demonio `freshclam`.

Actualizar ahora *(botón)* Lanza la actualización inmediata del patrón de virus (mediante el mecanismo que se haya configurado en el módulo). Al recargarse este formulario, si todo ha ido bien, el valor que se muestra en *{Fecha Actualización}* debería haber cambiado.

H.2.2. Formulario "Configuración"

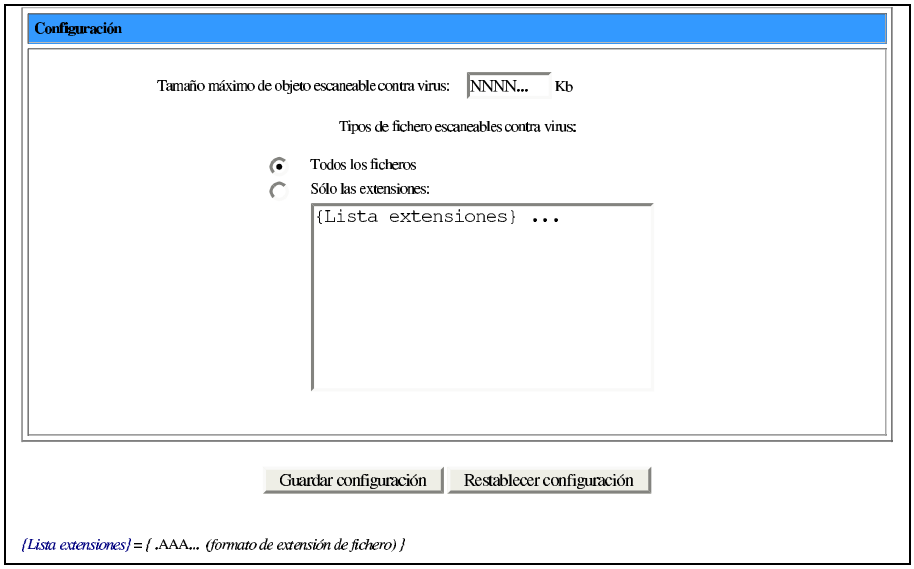

Figura 24: Módulo 'Antivirus (ClamAV)' – Configuración

Mediante este formulario se configura el funcionamiento básico del antivirus, en concreto: cuál es el tamaño máximo de fichero que se escaneará contra virus, y qué tipos de fichero serán los que se examinen.

Tamaño máximo de objeto escaneable contra virus *(campo de texto)* Se indica el tamaño máximo que podrá tener un fichero descargado de la Web para que se examine por el antivirus. Se recomienda especificar aquí un valor bastante alto (>4096 Kb) ya que, este parámetro, existe principalmente a efectos de "protección", para no degradar demasiado el rendimiento del servidor.

Tipos de fichero escaneables contra virus *(botones radiales)* Se indica si se desea que se examinen contra virus *todos los ficheros* descargados, o bien limitarse *sólo a las extensiones* especificadas en el cuadro de texto a continuación.

En el caso de elegir la 2ª opción (*"Sólo las extensiones"*), éstas se indicarán separadas por espacios, tabuladores o retornos de carro, por ejemplo:

```
.exe .bat .pif .scr
.vb  .vbs .vbe .wsh
.doc .rtf .ppt
.zip .rar .bzip .gz
```

Esto provocará que se escaneen sólo los ficheros cuyas extensiones aparezcan en esta lista; los demás, se dejarán pasar sin examinarlos contra virus (es una opción a tener en cuenta

para limitar la carga de CPU, y acelerar la descarga de ficheros que a priori se conoce que no contendrán virus).

H.2.3. Formulario "Actualizaciones"

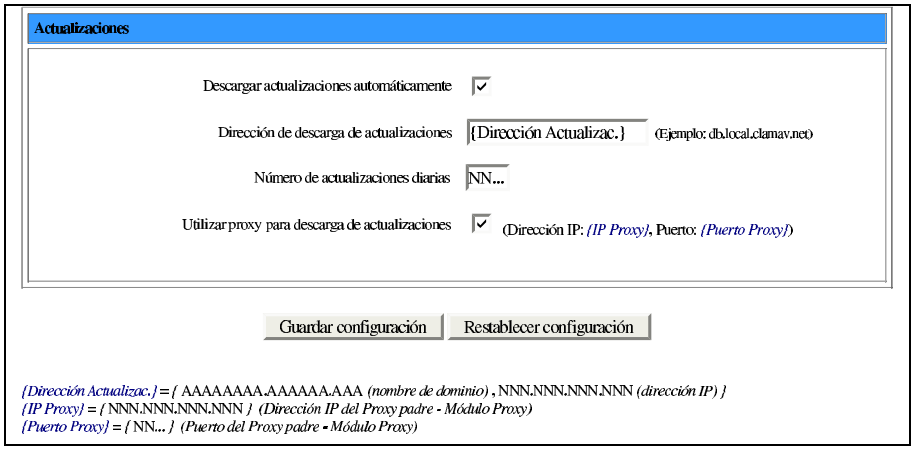

Figura 25: Módulo 'Antivirus (ClamAV)' – Actualizaciones

Este último formulario permite configurar las actualizaciones automáticas de la base de datos o *patrón* de virus del antivirus, que en nuestro caso se realiza mediante el programa `freshclam`. Se puede incluso desactivar este mecanismo de actualizaciones automáticas, de forma que la única vía de actualizar el antivirus sería realizándolo expresamente mediante el botón *"Actualizar ahora"* del formulario *Control del módulo* (ver figura 23).

Descargar actualizaciones automáticamente *(casilla de verificación)* Controla si se desea que la base de datos de virus se actualice de forma automática periódicamente (lo cual activaría el demonio `freshclam`).

Dirección de descarga de actualizaciones *(campo de texto)* La dirección IP o el nombre de host que se empleará para descargar las actualizaciones. Por ejemplo: `db.local.clamav.net`

Número de actualizaciones diarias *(campo de texto)* Se indica aquí el nº de veces por día que se desea que se compruebe (mediante conexión al *host* indicado en el campo anterior) si existe una nueva versión del patrón de virus, para proceder a su descarga en caso afirmativo. Por ejemplo se puede indicar `24`, lo cual comprobaría si existe un nuevo patrón de virus cada hora.

Utilizar proxy para descarga de actualizaciones *(casilla de verificación)* Se utilizará el *proxy padre* configurado en el módulo de Proxy, para la conexión al *host* especificado en *"Dirección de descarga de actualizaciones"*.

A modo informativo, se muestra a la derecha de esta casilla, cuál es el proxy padre (y el puerto) configurado en el módulo que Proxy, que se utilizará. (Estos valores se obtienen gracias al mecanismo de comunicación entre módulos).

H.2.4. Reflexión acerca de los formularios de configuración del módulo Antivirus

Como se puede apreciar, el módulo Antivirus presenta una gran simplicidad a la hora de realizar su configuración a través de la interfaz. Por contra, es un módulo cuya complejidad puede considerarse

ligeramente mayor a la de los demás, precisamente debido a su estrecha relación con los otros módulos.

Esto es debido a que el antivirus de red, para poder filtrar el tráfico de descarga contra virus, es necesario que se sitúe *en medio* del flujo de descarga de datos (al igual que ocurre con el filtro de contenidos *Dansguardian*, por ejemplo). Para ello, nos apoyamos en la utilidad de software descrita con anterioridad, *"Squid ClamAV Redirector" (SCAVR)*, que se relaciona estrechamente con el software Squid para lograr esta tarea con la mayor eficiencia.

Por otro lado, al ser en este caso 3 componentes de software diferentes (`clamav`, `SCAVR` y `freshclam`) resulta más complicado que la media gestionar este módulo, pues cada componente se ejecuta y se configura de manera diferente.

Por suerte para el usuario, la interfaz de configuración logra con eficacia el objetivo que persigue: actuar de *wrapper* ocultando al usuario las complejidades internas. El usuario por tanto únicamente tendrá que preocuparse de elegir las opciones que más le convengan para que la funcionalidad sea la esperada (cosa que, por otra parte, apenas será necesario, ya que la funcionalidad tal y como se entrega configurada por defecto, es casi completa y seguramente suficiente para la mayoría de las necesidades y de los usuarios actuales).

Puede saltar a la vista que no se dé la opción en este módulo de utilizar otros antivirus aparte de ClamAV. Es bien sabido que existen multitud de fabricantes que ofertan su software de antivirus en red, en muchos casos a modo de *appliances* que, situados en un punto estratégico de la red (suele ser *antes* del firewall), examinen todo el tráfico de red en busca de virus, y lo bloqueen en caso de que concuerde con uno de los patrones de virus. La manera de configurar estos antivirus, es utilizándolos como Proxy (*'proxy padre'*). Entonces, ¿por qué no se ha reservado una casilla de configuración para especificar la dirección del antivirus externo? La respuesta es que ya existe un lugar donde se puede configurar ésto: precisamente en el formulario *"Proxy padre"* del módulo Proxy. Incluir en el módulo *Antivirus* la misma casilla (pero con un significado o utilidad diferente) habría inducido a confusión.

También quizás merezca la pena detenerse en comentar la decisión de dejar la *posibilidad* de utilizar el proxy padre configurado en el módulo de Proxy, para la descarga de actualizaciones del antivirus, en lugar de hacerlo directamente. La razón es simple: se supone que el Proxy Padre es necesario para acceder al exterior (Internet), de hecho esa suele ser su principal función. Por tanto, se ha estimado que sea conveniente emplearlo también para realizar la conexión al sitio de actualizaciones del antivirus.

En todo caso, si se desea que esta conexión se realice directamente (sin pasar por Proxy alguno) se puede desactivar esta casilla (si bien sería conveniente verificar antes, por supuesto, que existe conectividad directa hacia dicho *host*).

H.2.5. Relación con los parámetros de configuración del software `ClamAV`

Formulario "Configuración":

Campo	Parámetros	Comentarios
Tamaño máximo de objeto escaneable contra virus	`MaxRequestsize`	Se establecerá al valor indicado, con el sufijo '`Kb`'.
Tipos de fichero escaneables contra virus	`pattern`	Se introducirán las extensiones (separadas por espacio) indicadas en *{Lista extensiones}*, o bien la palabra clave '`all`' en caso de estar seleccionado el botón de radio *"Todos los ficheros"*.

Formulario "Actualizaciones":

Campo	Parámetros	Comentarios
Dirección de descarga de actualizaciones	`DatabaseMirror`	Se establecerá al valor indicado en dicho campo (ya sea un nombre de *host*, o una dirección IP).
Número de actualizaciones diarias	`Checks`	Se establecerá este valor tal cual se indica.
Utilizar proxy para descarga de actualizaciones	`HttpProxyServer` `HttpProxyPort`	Se indican, respectivamente, los valores de los parámetros exportables `direccion_proxy_padre` y `puerto_proxy_padre` del módulo Proxy (en caso de estar definidos).

H.2.6. Importación/Exportación de parámetros

Se indican a continuación los parámetros *importables* o *exportables* para el módulo Antivirus.

Parámetros importables:

Módulo	Parámetro	Descripción
Proxy (Squid)	`direccion_proxy_padre`	Se utilizará, como se ha indicado en el apartado anterior, para configurar el Proxy para descarga de actualizaciones de `freshclam`. En este caso, configura el `HttpProxyServer`.
Proxy (Squid)	`puerto_proxy_padre`	Idem, para indicar el puerto (`HttpProxyPort`).

Parámetros exportables:

Parámetro	Descripción
`activado`	Se establecerá a `1` en el caso de que el módulo se haya configurado para activarse en el arranque (formulario **"Control del módulo"**), o al valor `0` en el caso contrario.
`lista_extensiones`	Se almacenarán en este parámetros la lista de extensiones separadas por espacio introducidas en *{Lista extensiones}*, del formulario **"Configuración"**, con objeto de tener este valor almacenado y poder recuperarlo más tarde (si el usuario eligió ***Todos los ficheros*** en este formulario).

H.3. Formularios de configuración del módulo Filtro de Contenidos (DansGuardian)

H.3.1. Formulario "Control del Módulo"

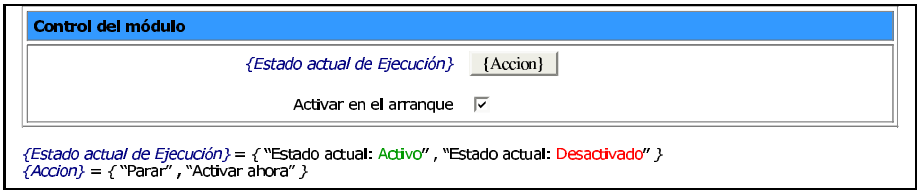

Figura 26: Módulo 'Filtro de Contenidos (DansGuardian)' – Control del módulo

Como en la mayoría de los módulos, DansGuardian es también un servicio que puede ser arrancado o parado, bien de forma inmediata (mediante el botón etiquetado como ***{Acción}***) o bien en el arranque del sistema, mediante la casilla de verificación. Como se puede ver, esto coincide casi al 100 % con lo visto en el módulo Proxy (Squid), y será un formulario común en muchos otros Módulos.

{Acción} *(botón)* Esta etiqueta puede adquirir cualquiera de los valores que indica su leyenda, mostrándolo sobre el botón, que vendrá dado dependiendo del estado actual de ejecución del servicio `dansguardian` (es decir, si está activado, o parado). Al pulsar sobre él, se realiza la acción que indica la etiqueta en cada momento: ***Parar*** detiene el servicio, ***Activar ahora*** pone en marcha el servicio. Ambas acciones son inmediatas.

En el caso de la etiqueta que se muestra a la izquierda del botón, *{Estado actual de ejecución}*, muestra al usuario cuál es el estado en el que se encuentra el servicio `dansguardian` en el momento de cargar este formulario.

Activar en el arranque *(casilla de verificación)* Al marcar *"Activar en el arranque"*, y una vez guardada la configuración, automáticamente se reconfigura el sistema para que el servicio `dansguardian` se active en el arranque de la máquina. Esto no provocará la ejecución inmediata del servicio, esto es, el usuario aún deberá pulsar el botón *{Acción}* para arrancar el servicio, o pararlo, de forma efectiva en este instante.

En el caso de no querer que `dansguardian` se active en el arranque, se debe dejar desmarcada esta casilla. Esto provocará que el sistema se configure de manera que el servicio no se ejecute de forma automática. Sin embargo, se puede activar o parar en cualquier instante.

H.3.2. Formulario "Configuración básica"

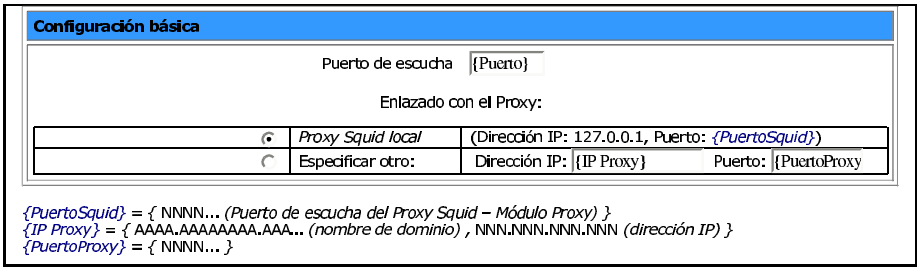

Figura 27: Módulo 'Filtro de Contenidos (DansGuardian)' – Configuración básica

Puerto de escucha *(campo de texto)* Aquí se indica el puerto en el cual escuchará DansGuardian las peticiones de los usuarios. DansGuardian actuará de proxy, recibiendo estas peticiones y obteniendo las páginas de Internet (o a través del Proxy configurado). Este puerto de escucha deberá ser diferente al que se emplee para el Squid local, ya que en caso contrario, habría un conflicto de enlace a los puertos *(binding)* por parte de los servicios.

Enlazado con el Proxy *(botones radiales)* Se ofrecen dos opciones a la hora de obtener el contenido de Internet:

Proxy Squid local: si se elige esta opción, se configurará DansGuardian automáticamente para enlazarse al proxy Squid local del cortafuegos. La configuración que en estos momentos tiene el Proxy Squid se mostrará en la etiqueta *{PuertoSquid}* (que es el puerto TCP en el que está escuchando el servicio Squid), y se usará este valor para configurar el proxy en DansGuardian. (Nota: En caso de que el servicio Squid se encuentre desactivado, no se enlazará a éste sino que se accederá directamente a Internet).

Especificar otro: se da al usuario la opción de no utilizar el proxy Squid local sino que se empleará otro Proxy de la red al que redirigir las peticiones. En los campos "Dirección IP" y "Puerto" se indicará cómo acceder a dicho Proxy.

H.3.3. Formulario "Grupos especiales de usuarios"

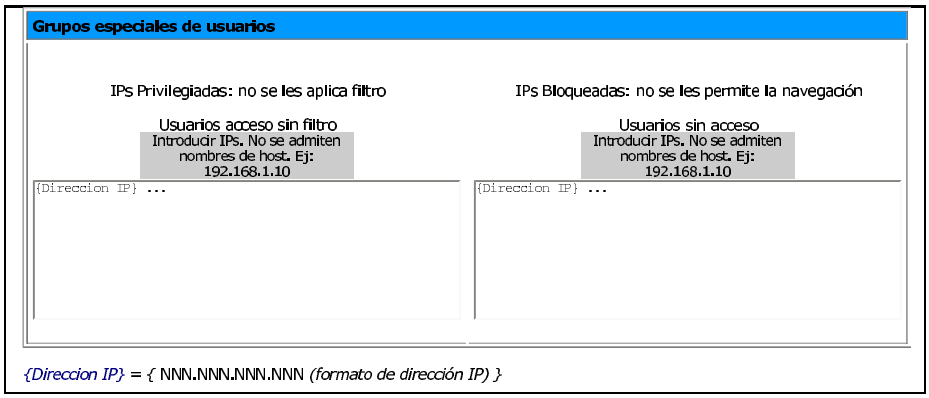

Figura 28: Módulo 'Filtro de Contenidos (DansGuardian)' – Grupos especiales de usuarios

Mediante este formulario se configuran los **usuarios de la red local** que tendrán un nivel de acceso diferente al de los usuarios normales. Estos usuarios de la red se especifican mediante sus correspondientes direcciones IP (por tanto para poder identificarlos, el administrador de la red deberá asignar IPs fijas a dichos usuarios, si es que se utiliza un mecanismo de asignación de IPs dinámicas como suele ser el caso común).

Usuarios acceso sin filtro *(cuadro de texto)* Se indicarán las direcciones IP, separadas por retorno de carro, a las que se quiera conceder un acceso ilimitado a Internet, esto es sin que se les apliquen las reglas de filtrado de contenidos. Normalmente estas direcciones IP corresponderán a las del administrador de la red, y puede que algunos servidores locales que necesiten acceder a Internet sin restricción alguna.

Usuarios sin acceso *(cuadro de texto)* Las direcciones IP aquí indicadas (igualmente separadas por retorno de carro), no tendrán acceso a Internet, ya que se filtrará su acceso por IP de origen, independientemente de la URL a la que intenten acceder. Se puede utilizar ésto

para “desactivar” el acceso a Internet a un usuario o grupos de usuarios, mostrándoseles esta circunstancia cuando intenten acceder, mediante la página de notificación de acceso bloqueado.

El resto de los filtros no serán tenidos en cuenta para los usuarios aquí indicados, es por ello que aparecen en pimer lugar.

H.3.4. Formulario “Filtro de URL’s”

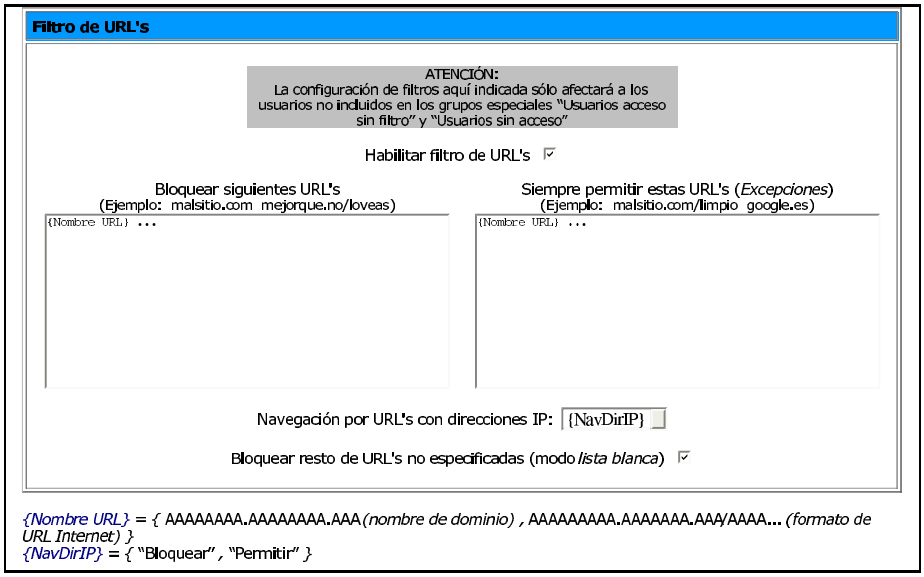

Figura 29: Módulo ‘Filtro de Contenidos (DansGuardian)’ – Filtro de URL’s

Mediante este formulario se configurará el filtrado por URLs, esto es, lo que se conoce como “lista negra” o “lista blanca”.

El filtrado de URLs así como, especialmente, el de contenidos, consume una potencia de proceso elevada, por estar basado en un sistema de reconocimiento de patrones, y puede que alguno de ellos no sea necesario en algunos casos concretos (dependiendo de las necesidades del administrador de red). Es por ello que se da la opción de desactivarlos por separado, mediante la casilla de verificación que aparece en la parte superior del formulario.

Habilitar filtro de URL’s *(casilla de verificación)* Se habilita o inhabilita la acción de filtrado que realiza este formulario.

Bloquear siguientes URL’s *(cuadro de texto)* El usuario puede indicar aquí las URL’s que directamente serán bloqueadas por el filtro de contenidos. Se indicará una URL en cada línea. El formato es el siguiente: se indica una parte de la URL que concordará con la que escriba el usuario en su navegador. Es decir, que la URL que aquí se escriba será parcial, no tiene por qué consistir en una URL completa, y aún así concordará con cualquier prefijo o sufijo de la misma.
Se indica de la forma *nombre.de.host/ruta_ opcional/dentro/del/servidor*, siendo esta última parte opcional (es decir que puede indicarse sólo el nombre de host, sin las carpetas virtuales).

Siempre permitir estas URL's (*Excepciones*) *(cuadro de texto)* Como su nombre indica, son excepciones a la lista de URLs bloqueadas. Normalmente serán nombres de host considerados inicialmente como "peligrosos" pero indicando esta vez carpetas virtuales que se conoce que su contenido está limpio.
Por ejemplo, si se consideró que *www.microsoft.com* es un sitio malicioso, se pudo poner esta URL (*microsoft.com*) en el recuadro de la izquierda, pero si queremos permitir el acceso a la carpeta virtual *www.microsoft.com/office* en concreto, sólo tenemos que indicarlo así en el recuadro de la derecha.

Navegación por URL's con direcciones IP *(lista desplegable)* Puede tomar dos valores, "Bloquear" o "Permitir". Especifica si se permitirá al usuario acceder a URLs que consisten en direcciones IP, en lugar de nombres de host, por ejemplo: http://212.31.87.5/ (con lo cual podría saltarse las reglas de filtrado de URLs).

Bloquear resto de URL's no especificadas (modo *lista blanca*) *(casilla de verificación)* Normalmente el filtro de contenidos funciona en modo *lista negra*. Esto significa que se permite todo, excepto las URLs que se indiquen expresamente como 'malignas'. Si se desea habilitar el modo inverso de funcionamiento, en el que sólo se pueda navegar por las URLs que se indiquen expresamente como válidas (en el recuadro de Excepciones) se marcará esta casilla. Así, el recuadro de Excepciones se convierte pues en una *lista blanca*.

H.3.5. Formulario "Filtro de contenidos"

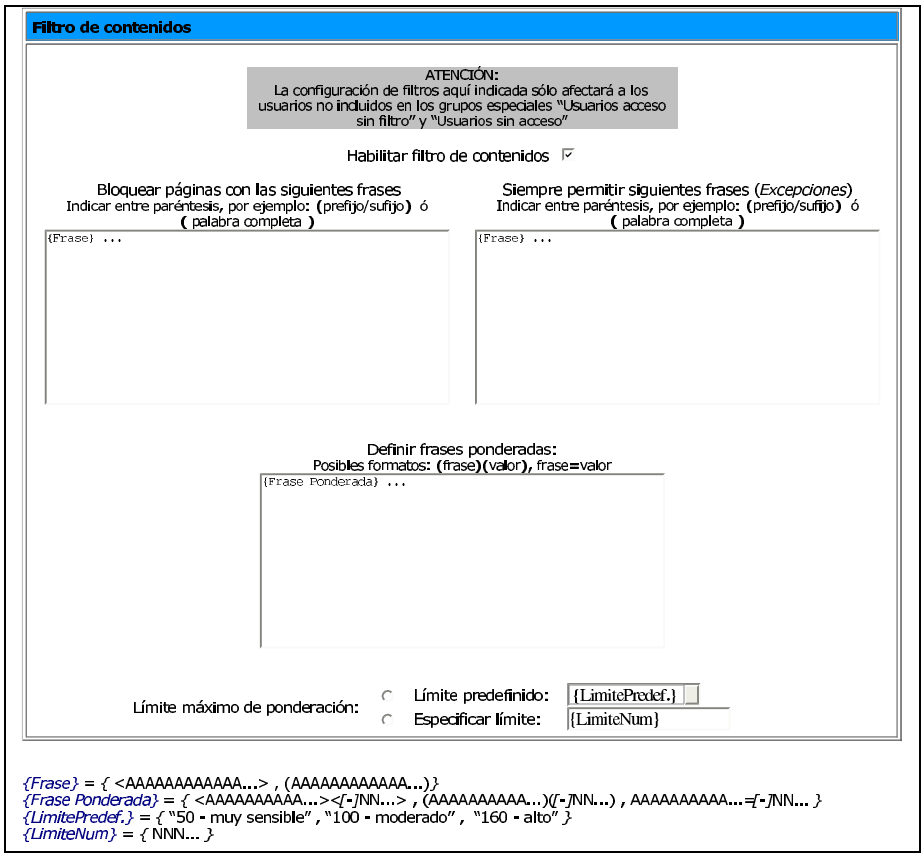

Figura 30: Módulo 'Filtro de Contenidos (DansGuardian)' – Filtro de contenidos

Este es el *filtro de contenidos* propiamente dicho, ya que se examina el *contenido* de la página Web para ver si se permite o no su visualización. Al igual que el anterior formulario, se puede desactivar independientemente para ahorrar CPU.

Bloquear páginas con las siguientes frases *(cuadro de texto)* Se indican aquí, en cada línea, las palabras, frases o prefijos y sufijos que se considerarán altamente dañinos y que por tanto no podrán pasar nunca el filtro. Se pueden indicar entre paréntesis en lugar de la acostumbrada sintaxis <> original de DansGuardian, con objeto de hacerlo menos complicado para el usuario nóvel.
Si se indica una palabra sin espacios al principio o al final (dentro de los paréntesis) se considerará como un sufijo o un prefijo, respectivamente. Hay que tener pues cuidado con ésto. También se pueden indicar frases enteras.
Si se indican en una sóla línea dos patrones (entre paréntesis) separados por una coma, se considerarán válidos si aparecen ambos en la misma página, en caso contrario no tendrán efecto. Por ejemplo: `( vibrador ),( placer )` bloquearán la página si aparecen ambas palabras, pero no si aparece sólo una de ellas.

Siempre permitir siguientes frases (*Excepciones*) *(cuadro de texto)* Como su nombre indica, son excepciones al recuadro de la izquierda, actuando de la misma forma que en el formulario anterior: aunque se encuentre una concordancia con alguna palabra de la izquierda, si además se encuentra concordancia con alguna de la derecha se dejará pasar. Por ejemplo, si en el recuadro de la izquierda apareciese `( viagra )` y en el de la derecha `( prospecto )`, entonces la aparición de esta segunda palabra en la página (indicando quizás que se trata de una página con información farmacéutica) inhabilitaría el filtro de la primera. Dicho de otra forma, es una forma de indicar palabras con un "peso negativo absoluto".

Definir frases ponderadas *(cuadro de texto)* Esta es una de las características más potentes del filtro de contenidos. En lugar de bloquear (o permitir) por completo una página Web por la aparición de una palabra, se pueden asignar pesos a cada palabra de forma que si aparece un nº suficiente de ellas (mediante un umbral configurable), finalmente se decidiría si bloquear o no la página. Así por ejemplo, si a cada palabra le asignamos un peso de 20, y tenemos un límite de 50, con la aparición de 3 de estas palabras (20+20+20=60) se conseguiría bloquear la página; en caso contrario no se consideraría suficientemente "dañina", y por tanto se dejaría pasar.
El formato puede ser similar al original de DansGuardian: `(frase)(valor)`, o bien una sintaxis en ocasiones quizás más clara indicada como `fase=valor`. Las reglas de escritura de la frase son las mismas que para los recuadros anteriores. Los valores pueden ser positivos o negativos.

Límite máximo de ponderación *(botón radial)* Se puede indicar el límite máximo (umbral) el cual si se supera, provocará el bloqueo del acceso a la página. Este límite se podrá especificar mediante una serie de valores predefinidos (en *"Límite predefinido"*) o bien como un valor numérico libre (en *"Especificar límite"*). Se recomiendan los valores que aparecen predefinidos que son:

- 50 - muy sensible
- 100 - moderado
- 160 - alto

H.3.6. Reflexión acerca de los formularios del módulo Filtro de Contenidos

Se ha intentado que la aparente complejidad de un sistema de filtro de contenidos quede oculta al usuario y aparezcan las opciones más simples y directas para realizar esta función. A pesar

del gran número de parámetros y ficheros que normalmente es necesario conocer para configurar adecuadamente DansGuardian, mediante los formularios descritos anteriormente se puede ver cómo en realidad la complejidad en la configuración no resulta en absoluto elevada (para consegui ésto se ha realizado un trabajo de abstracción digno de mencionar, pues no es DansGuardian el mejor ejemplo de software fácil o intuitivo de configurar a primera vista a nivel de sistema de ficheros).

Por otro lado, el procesamiento de los valores que se gestionan a través de los formularios no revisten de excesiva complejidad. En la mayor parte de los casos, se trata de listas de valores (URLs, frases) en las que se respeta prácticamente la sintaxis original, que tiene un significado concreto para DansGuardian, y que por tanto se entregan tal cual en los correspondientes ficheros (después de realizar mínimas adaptaciones). Lo que hay que saber es en qué ficheros introducir dichas listas, algo que queda oculto al usuario al verlo todo como un conjunto de formularios con campos directamente editables.

En la sección `H.3.7` se enumeran los ficheros a los que van los valores introducidos en cada uno de los campos con detalle.

Mención aparte merece la transformación que se realiza con la sintaxis de las "frases" que se utilizan en el formulario "Filtro de Contenidos" (Figura 30). Como se puede ver la sintaxis que se acepta aquí es:

```
(frase)
( frase)
(frase )
( frase )
```

Si bien la que permite DansGuardian por defecto es:

```
<frase>
< frase>
<frase >
< frase >
```

Esto se consigue mediante una sencilla traducción (entre los caracteres '`()`' y '`<>`') a la hora de mostrarlos al usuario. Se ha decidido sustituir este tipo de caracteres originales de DansGuardian por los paréntesis ya que se ha estimado que al usuario le resultará menos complicado de asumir que la palabra o frase a escribir debe estar entre paréntesis, y más fácil de escribir, que utilizando los caracteres originales.

En el caso del formulario "Filtro de URLs" (Figura 29), el contenido va directamente a los ficheros correspondientes (banned*{xxx}* y exception*{xxx}*) si bien haciendo una diferenciación: las URLs que no contengan el carácter '/', es decir que únicamente están compuestas por un nombre de host, se introducirán en el fichero *bannedsitelist*, mientras que las que sí contengan este carácter, se redirigirán al *bannedurllist*. De esta forma se consigue que DansGuardian funcione de una manera más eficiente, con el pequeño inconveniente (pensamos que asumible) de no mantener el órden estricto de estas líneas (al tener que dividirse en ficheros diferentes).

Por último, como se puede apreciar, en el formulario "Configuración básica" (Figura 27) aparece un valor dinámico que consiste en el puerto del Squid local que está corriendo en el cortafuegos (la dirección IP siempre será 127.0.0.1 por tratarse de la máquina local). Este dato, el nº de puerto de Squid, se obtiene de su módulo correspondiente (el de Proxy) mediante el mecanismo de comunicación entre módulos que ya se ha mencionado en otras ocasiones con anterioridad. En caso de que en algún momento cambie este dato de configuración en Squid, también tendrá que cambiar dicho valor en DansGuardian; no hay problema pues ésto también está previsto en dicho mecanismo de comunicación entre módulos.

H.3.7. Relación con los parámetros de configuración del software `DansGuardian`

A menos que se indique lo contrario, los parámetros aquí indicados serán los del fichero *dansguardian.conf*.

Formulario "Configuración básica":

Campo	Parámetros	Comentarios
Puerto de escucha	`filterport`	Sólo se establecerá el puerto, no la dirección IP (el parámetro `filterip` se dejará vacío).
Enlazado con el Proxy	`proxyip` `proxyport`	En caso de seleccionar *'Proxy Squid local'* se establecerá *proxyip = 127.0.0.1* y *proxyport* al puerto de Squid. En caso de seleccionar '*Especificar otro*', se darán a estos parámetros los valores introducidos en '*Dirección IP*' y '*Puerto*' (ambos serán obligatorios en este caso)

Formulario "Grupos especiales de usuarios":

Campo	Parámetros	Comentarios
Usuarios acceso sin filtro	-	Fichero **exceptioniplist**: Se introducirá cada línea tal cual.
Usuarios sin acceso	-	Fichero **bannediplist**: Se introducirá cada línea tal cual.

Formulario "Filtro de URL's":

Campo	Parámetros	Comentarios
Habilitar filtro de URL's	`bannedsitelist` `bannedurllist` `exceptionsitelist` `exceptionurllist`	Fichero **dansguardianf1.conf**: Se establecerá el valor de estas directivas a las rutas por defecto de los ficheros correspondientes a los que hacen referencia. De esta forma, DansGuardian los procesará.
Bloquear siguientes URL's	-	Fichero **bannedurllist**: Se introducirá cada línea que contenga una '/', tal cual. Fichero **bannedsitelist**: Se introducirá cada línea que no contenga una '/', tal cual.
Siempre permitir estas URL's (Excepciones)	-	Fichero **exceptionurllist**: Se introducirá cada línea que contenga una '/', tal cual. Fichero **exceptionsitelist**: Se introducirá cada línea que no contenga una '/', tal cual.
Navegación por URL's con direcciones IP	-	Fichero **bannedsitelist**: En caso de '*Bloquear*', se introducirá una línea con la cadena: '`*ip`'
Bloquear resto de URL's no especificadas (modo *lista blanca*)	-	Fichero **bannedsitelist**: En caso de marcar esta casilla, se introducirá una línea con la cadena: '`**`'

Formulario "Filtro de contenidos":

Campo	Parámetros	Comentarios
Habilitar filtro de contenidos	`bannedphraselist` `exceptionphraselist` `weightedphrasellist`	Fichero **dansguardianf1.conf**: Se establecerá el valor de estas directivas a las rutas por defecto de los ficheros correspondientes a los que hacen referencia. De esta forma, DansGuardian los procesará.
Bloquear páginas con las siguientes frases	-	Fichero **bannedphraselist**: Se introducirá cada línea, traduciendo los caracteres '()' por '<>'.
Siempre permitir siguientes frases (Excepciones)	-	Fichero **exceptionphraselist**: Se introducirá cada línea, traduciendo los caracteres '()' por '<>'.
Definir frases ponderadas	-	Fichero **weightedphraselist**: Se introducirá cada línea, traduciendo el formato alternativo *frase=valor* por el original *(frase)(valor)*, en caso de que aquél se emplee.
Límite máximo de ponderación	`naughtynesslimit`	(En el fichero **dansguardianf1.conf**): Se le dará el valor que ha seleccionado el usuario, ya sea un límite predefinido, o elegido al azar.

H.3.8. Importación/Exportación de parámetros

Se indican a continuación los parámetros *importables* o *exportables* para el módulo Filtro de Contenidos.

Parámetros importables:

Módulo	Parámetro	Descripción
Proxy (Squid)	`puerto_escucha`	Se empleará este valor del Proxy Squid para que, en caso de que el usuario seleccione '*Enlazado con el Proxy – Proxy Squid local*' se sepa automáticamente a qué puerto conectarse (para el parámetro `proxyport`).
Proxy (Squid)	`activado`	Si el Proxy Squid local está desactivado, en caso de que el usuario seleccione '*Enlazado con el Proxy – Proxy Squid local*' no se configurará valor alguno en el parámetro `proxyport`.

Parámetros exportables:

Parámetro	Descripción
`activado`	Se establecerá a `1` en el caso de que el módulo se haya configurado para activarse en el arranque (formulario **"Control del módulo"**), o al valor `0` en el caso contrario.
`puerto_escucha`	Puerto en el que `dansguardian` recibirá las peticiones HTTP. Será el valor introducido en el campo ***Puerto de escucha*** del formulario **"Configuración básica"**.
`proxy_externo`	Tendrá el valor `1` en el caso de que se haya configurado un Proxy externo al que conectarse (***Enlazado con el Proxy – Especificar otro*** del formulario **"Configuración básica"**), ó el valor `0` en caso contrario.
`direccion_proxy_externo`	Dirección IP (o nombre de host) del proxy externo configurado por el usuario. Se tomará del campo ***Enlazado con el Proxy – Dirección IP*** del formulario **"Configuración básica"**.
`puerto_proxy_externo`	Puerto configurado para el proxy externo (se toma del campo ***Enlazado con el Proxy – Puerto*** del formulario **"Configuración básica"**).

H.4. Formularios de configuración del módulo Firewall (FireHOL)

El módulo FireHOL, como se ha mencionado en anteriores ocasiones, tiene una estrecha relación con el resto de los módulos. Ello es debido a que la mayoría de éstos, y los componentes software subyacentes, necesitan de la colaboración del Cortafuegos para funcionar correctamente. Así, por ejemplo, si el software `squid` está escuchando en el puerto TCP 3128, el Cortafuegos deberá habilitar las conexiones hacia este puerto, en caso contrario los usuarios no podrían acceder (aunque siempre dependiendo de la política de Cortafuegos que el usuario haya decidido instaurar, claro está).

La idea es que el usuario no tenga que preocuparse por las interioridades del sistema para que éste funcione, sino que el mismo se *pre-configure* automáticamente para que su funcionamiento sea posible, atendiendo a aquellos parámetros de configuración que el usuario haya elegido para cada módulo.

El módulo Firewall por tanto deberá tener en cuenta los parámetros de configuración (exportables) de los demás módulos para establecer una configuración básica de filtrado que, sin ninguna acción más por parte del usuario (o incluso aunque haya instaurado una política restrictiva por defecto), permita el funcionamiento de los mismos. Pero no realiza esta acción de forma totalmente oculta al Usuario (pues éste puede querer hacer otra cosa, si sabe lo que está haciendo). En el formulario "Reglas automáticas" (en la sección H.4.3) se puede comprobar cómo se ha dado solución a esta cuestión.

A continuación se describen los formularios de los que consta este módulo.

H.4.1. Formulario "Control del Módulo"

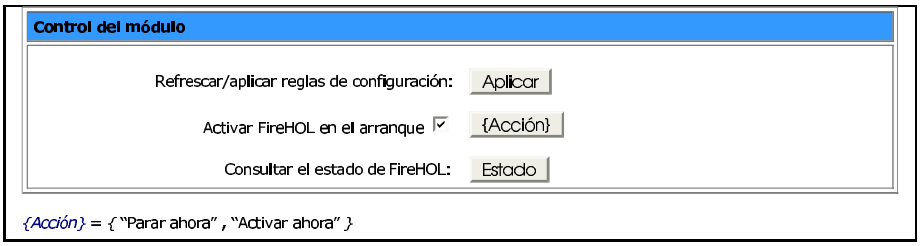

Figura 31: Módulo 'Firewall (FireHOL)' – Control del módulo

El formulario de "Control del Módulo", en el caso de FireHOL, tiene una apariencia un tanto distinta al resto de los módulos. Ello es debido a que el cortafuegos no es en realidad un servicio del sistema, sino que se configura de una u otra forma dependiendo de las necesidades del usuario (si desea activar el conjunto de reglas definidas, o desactivarlo, lo cual implica activar también otras). Así mismo, si el usuario realiza modificaciones en el conjunto de reglas, puede requerir que éstas se activen en un momento dado. Con el resto de los módulos, los cambios en la configuración por lo general no se ven de forma inmediata, pero dada la escasa probabilidad de que el usuario tenga que realizar cambios de configuración en un software que "funciona", normalmente se entiende que sea necesario reiniciar el servicio cuando se realizan éstos. Sin embargo, el módulo Firewall suele sufrir de una revisión constante de sus reglas; ello es debido a la imposibilidad de que la actividad de una red local sea estática en el tiempo, contínuamente se requiere la utilización de nuevos protocolos y acceso a servidores y redes diferentes a las definidas hasta el momento. Este número incrementado de los cambios en la configuración es lo que dá su razón de ser al botón "Aplicar" de este formulario.

Otra opción no muy utilizada, es la de parar el servicio Cortafuegos. Normalmente no será necesario, a menos que se detecte un error grave de configuración que requiera de más tiempo de

lo normal en ser subsanado, o bien que se decida abrir por completo el acceso hacia y desde el exterior en un momento dado, por necesidades indeterminadas o tiempo de resolución mínimo, asumiendo el riesgo en cuanto a la seguridad perimetral que ello supone.

Por último, se dá la oportunidad al usuario de observar el *estado* de funcionamiento del cortafuegos, en el que se muestra entre otras cosas, el conjunto de reglas de *bajo nivel* que han producido las configuraciones por él realizadas.

Refrescar/aplicar reglas de configuración *(botón)* Refresca la configuración del Cortafuegos, aplicando los cambios que ha realizado (y guardado) el usuario pero que todavía puedan no estar activos.
Normalmente, al guardar la configuración no se refresca automáticamente el conjunto de reglas, por motivo de seguridad.

Activar FireHOL en el arranque *(botón radial)* Permite que el cortafuegos se inicie en el arranque del sistema, en caso contrario las reglas definidas en este módulo no tendrían efecto, hasta que no se activase expresamente el cortafuegos.

{Acción} *(botón)* Activará o desactivará de manera inmediata el cortafuegos.
La *activación* implicará la aplicación de todas las reglas definidas en la configuración, y aquí visibles.
La *desactivación* implicará la aplicación de una configuración por defecto, que permite el paso de todo tráfico a través del enrutador integrado en el sistema.[11]

Consultar el estado de FireHOL *(botón)* Permite acceder a un formulario auxiliar que muestra el estado de ejecución del cortafuegos, con información sobre su funcionamiento y configuración interna.[12]

Formulario auxiliar "Control del Módulo – Estado de FireHOL"

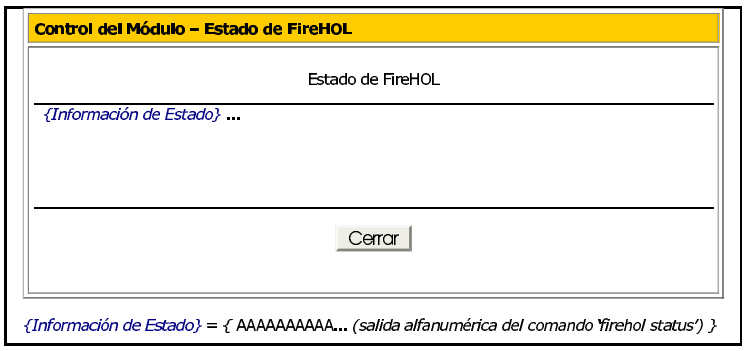

Figura 32: Módulo 'Firewall (FireHOL)' – Estado de FireHOL

Este es el formulario que se muestra cuando el usuario pulsa sobre el botón "Estado", del formulario principal "Control del Módulo". Consiste en una simple ventana emergente de informacion cuya única misión es la de mostrar dicha información al usuario, y permitir el cierre de la ventana por parte de éste.

Esta información es obtenida del sistema como se ha mencionado anteriormente y presentada donde aparece la etiqueta *{Información de Estado}*, a modo de espacio de presentación de texto multinínea de longitud indeterminada.

[11] Esto es lo que realiza la ejecución del comando '`/etc/init.d/firehol stop`', que a su vez llama a '`/etc/init.d/iptables stop`', y a nosotros nos parece adecuado...
[12] Ejecución del comando '`/sbin/firehol status`'.

Pulsando sobre el botón “Cerrar” desaparece el formulario sin realizar más acciones por su parte.

H.4.2. Formulario “Configuración general”

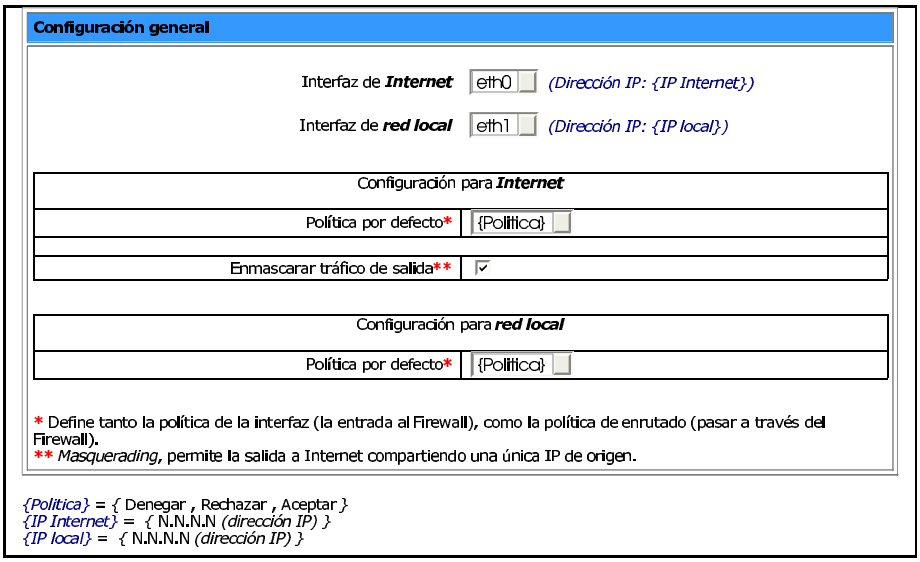

Figura 33: Módulo ‘Firewall (FireHOL)’ – Configuración general

Esta pantalla configura una serie de parámetros de carácter general que tendrán efecto globalmente en el módulo. Se trata de aquellos parámetros de carácter indispensable que es necesario establecer en un cortauegos típico, esto es: nomenclatura de interfaces (en nuestro caso habrá dos interfaces, las cuales se llamarán *Internet* y *red local*), política por defecto (si dejar pasar o bloquear todo el tráfico por defecto), y si activar el *enmascaramiento* (’Masquerading’) del tráfico saliente.

Intefaz de *Internet* *(lista desplegable)* El usuario puede elegir, de las interfaces de red presentes en el sistema, cuál se destinará para el tráfico de salida de Internet.

Interfaz de *red local* *(lista desplegable)* Igualmente, elegirá cuál de las interfaces se utilizará para conectarse a la red local. Las dos interfaces no podrán ser la misma.

Configuración para Internet: Se ofrecen los siguientes parámetros que afectarán a la interfaz de *Internet*.

Política por defecto *(lista desplegable)* Define la política por defecto para la interfaz de Internet, esto es, qué acción se realizará en presencia de tráfico en dicha interfaz, si no hay definida una regla de cortafuegos que diga lo contrario. Se permiten 3 posibles valores:

- **Denegar:** el tráfico entrante por la interfaz se descarta silenciosamente. Es lo que suele llamarte *“política DROP”*.
- **Rechazar:** el tráfico entrante se rechaza expresamente, esto es, se responde negativamente a las solicitudes de conexión. Suele denominarse *“política REJECT”*.
- **Aceptar:** el tráfico se deja pasar. (*“Política ACCEPT”*).

Enmascarar tráfico de salida *(casilla de verificación)* Si se activa, el tráfico saliente por la interfaz de Internet adquiere como IP de origen la propia de la interfaz. De esta forma, se ocultan las IP's de la red local hacia el exterior. En el sentido inverso de la comunicación, se realiza la traducción contraria, de forma que el tráfico vuelve a la IP de origen. Esto es lo que se conoce como *'Masquerading'*.

Configuración para red local: Al igual que con la interfaz de Internet, también se pueden configurar parámetros de la interfaz de la red local:

Política por defecto *(lista desplegable)* Define la política por defecto para la interfaz de la red local. Se permiten los mismos valores que para la interfaz Internet.

Hay que tener en cuenta que la política por defecto no tendrá efecto si el cortafuegos está *desactivado* (en el formulario "Control del Módulo"). En el momento en que el cortafuegos se desactiva, siempre se deja pasar todo el tráfico, independientemente de la política que se haya elegido. Este no es el motivo de definir las políticas, sino de servir de "comodín" cuando no hay una regla de cortafuegos explícitamente definida.

H.4.3. Formulario "Reglas automáticas"

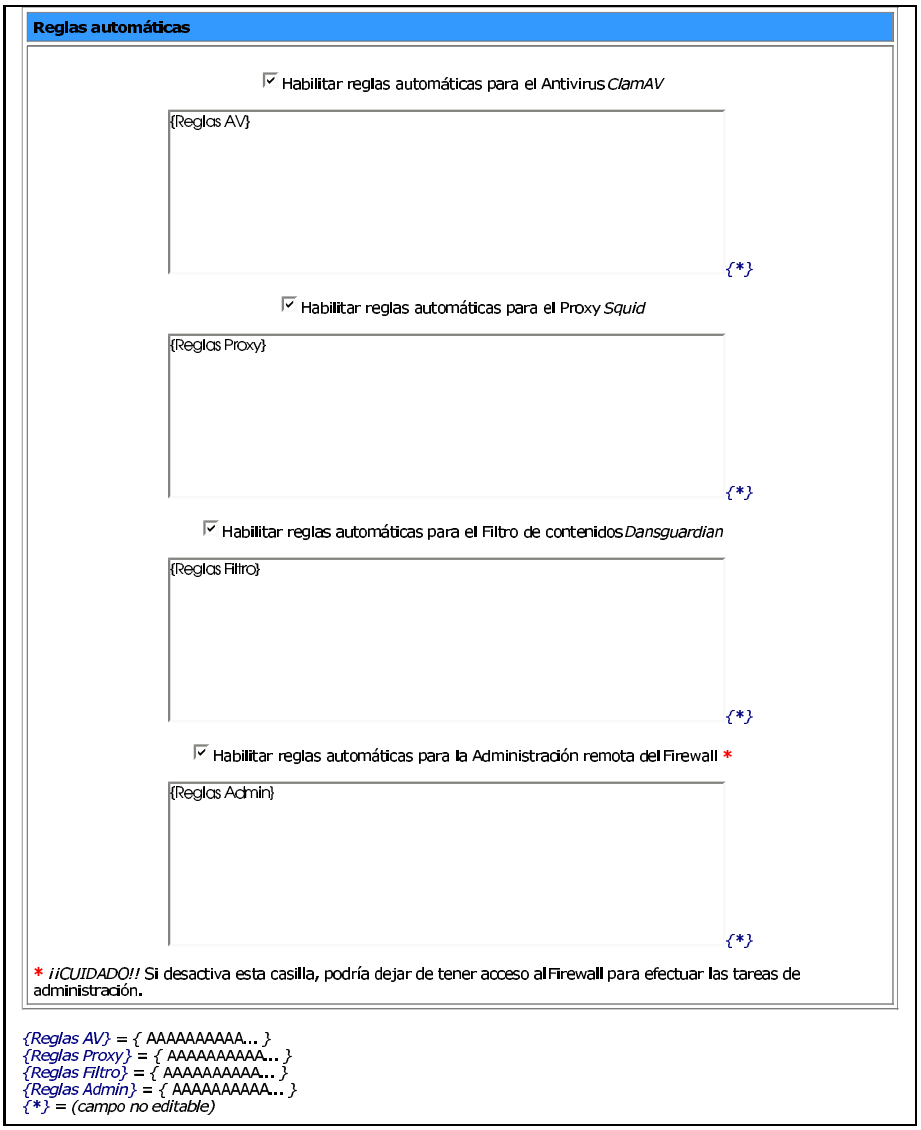

Figura 34: Módulo 'Firewall (FireHOL)' – Reglas automáticas

He aquí el resultado de permitir que los módulos del cortafuegos se hablen entre sí, y ofrezcan parámetros de su configuración a los demás módulos, para permitir una configuración automática allá donde sea posible. Este formulario permite que según las necesidades de conectividad de los módulos, el sistema se "configure" automáticamente para habilitar el paso explícito del tráfico que tenga que ver con dichas necesidades de conectividad.

Así, cada módulo publicará entre otras cosas, el puerto de escucha, y si está o no activo; mediante estos parámetros, el módulo cortafuegos puede generar las reglas de cortafuegos necesarias para permitir conexiones hacia dichos puertos por defecto.

El usuario puede revisar, mediante este formulario, dichas reglas automáticas, y decidir si deben estar activas, o por el contrario desactivarlas selectivamente para cada módulo.

El último recuadro contiene las reglas de Administración básicas para que sea posible la configuración del sistema desde el exterior (esto es, a través de la red) y no sólo desde la consola local. Se habilitarán los protocolos necesarios de administración (por ejemplo, SSH y Webmin) para que el usuario pueda conectarse al sistema y administrarlo.

Habilitar reglas automáticas para el Antivirus *ClamAV* *(casilla de verificación)* Se activan las reglas que el sistema propone para el módulo Antivirus.

Habilitar reglas automáticas para el Proxy *Squid* *(casilla de verificación)* Idem, para el módulo Proxy.

Habilitar reglas automáticas para el Filtro de contenidos *DansGuardian* *(casilla de verificación)* Igualmente, para el módulo Filtro de Contenidos.

Habilitar reglas automáticas para la administración remota del Firewall *(casilla de verificación)* Es aqui donde están definidas una serie de reglas que "deben estar" presentes para permitir la administración del firewall desde una localización remota (es decir, entrando en el Firewall a través de la red). Normalmente es necesario que estas reglas estén activadas, a menos que el usuario sólo desee administrar el sistema desde la consola local, o bien que haya definido sus propias reglas de administración, quizás más restrictivas que las aquí propuestas por el sistema.

H.4.4. Formulario "Reglas definidas por el usuario"

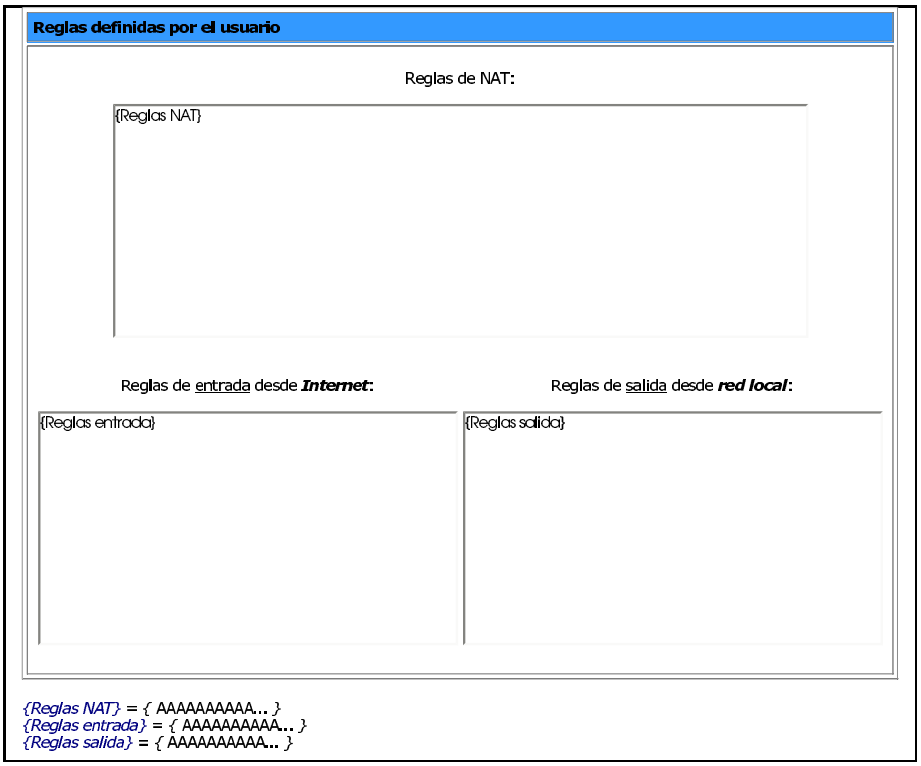

Figura 35: Módulo 'Firewall (FireHOL)' – Reglas definidas por el usuario

Mediante este formulario el usuario puede definir libremente reglas de cortafuegos, para establecer una política de seguridad hasta el nivel de profundidad que desee.

La forma de especificar dichas reglas es mediante el lenguaje FireHOL, un lenguaje que abstrae la complejidad habitual que supone la creación de reglas de cortafuegos en el sistema, con una sintaxis bastante simple. Véase la sección 8.6.3 para una descripción de dicho lenguaje y algunos ejemplos sobre su sintaxis.

Reglas de NAT *(cuadro de texto)* Permite especificar reglas de traducción de direcciones (NAT). Por ejemplo, reglas que capturen cierto tipo de tráfico y lo redirijan a un puerto distinto, o a una IP distinta. Ejemplos:

```
nat to-source 1.1.1.1 outface eth0 src 2.2.2.2 dst 3.3.3.3
nat to-destination 4.4.4.4 inface eth0 src 5.5.5.5 dst 6.6.6.6
nat redirect-to 8080 inface eth0 src 2.2.2.0/24 proto tcp dport 80
nat to-destination 1.1.1.1
nat to-destination 1.1.1.1 dst 2.2.2.2
nat to-destination 1.1.1.1 proto tcp dst 2.2.2.2
nat to-destination 1.1.1.1 proto tcp dport 25 dst 2.2.2.2
nat to-destination 1.1.1.1 proto tcp dport 25 dst 2.2.2.2 src 3.3.3.3
```

Reglas de entrada desde *Internet* *(cuadro de texto)* Se especifican aquí las reglas para el tráfico, tanto de conexión como de enrutado, que se introduce por la interfaz de **Internet**, y tienen como destino el propio Firewall (las de conexión a servicios) o la red local (las de enrutado). Ejemplos:

```
server ssh deny     # evita la administración remota vía SSH desde Internet
route  http accept  # permite acceder a servidores Web internos
route  all deny     # prohibe expresamente el resto del trafico de entrada a la red
```

No es necesario incluir las sentencias "`interface`" y/o "`router`" al principio de este conjunto de reglas, dichas sentencias se pueden asumir como presentes (pues se generarán automáticamente).

Reglas de salida desde *red local* *(cuadro de texto)* Idem, para el tráfico de conexión o enrutado que se introduce por la interfaz de la **red local**. Ejemplos:

```
server ssh accept   # se permite la conexión al cortafuegos mediante SSH
route  ftp accept   # se permite la salida mediante el protocolo FTP a Internet
route  http accept  # permite salida directa vía HTTP a Internet, sin pasar por el Proxy
```

H.4.5. Reflexión acerca de los formularios de configuración del módulo Firewall

Como se puede comprobar, este módulo es una mera interfaz gráfica que dá acceso a la configuración de FireHOL (como interfaz de abstracción sencilla y manejable, sobre todo en comparación con *iptables*) con las mismas posibilidades que éste ofrece en cuanto a poder descriptivo por medio de la sintaxis. Esperemos que esto no suponga una 'traba' para el usuario, que tiene que lidiar con cadenas de texto en lugar de con botones y controles, a la hora de definir la política de seguridad. La sintaxis que ofrece FireHOL pensamos que es lo suficientemente clara y sencilla como para que el tiempo de aprendizaje sea mínimo (además, la interfaz está pensada para ayudar a ésto, ya que se indica dónde se debe poner cada tipo de regla).

En posteriores versiones de la interfaz, se podría pensar en intentar crear un sistema gráfico que facilite aún más la tarea de especificar las reglas. Esto puede implicar también complicar el *parsing*

necesario de las reglas, así como reducir la sintaxis válida a emplear. Hemos dejado esta posibilidad como una mejora futura a la aplicación.

Mención aparte merece la parte de la interfaz que sí consigue abstraer totalmente los parámetros de configuración, que son los formularios de "Control del Módulo" y "Configuración general". Mediante el primero (exhaustivamente explicado en su correspondiente sección, vista anteriormente) se controla la ejecución del módulo de una manera similar a lo habitualmente visto, así como se consulta el estado de funcionamiento del mismo. En su caso, el formulario de "Configuración general" va más allá.

Se ha pensado que una de las tareas típicas a la hora de configurar un cortafuegos, es la de establecer políticas de seguridad dependiendo del sentido de la comunicación (normalmente, en un cortafuegos de dos "patas", será sentido exterior →interior, o bien a la inversa). Para ello, hay que tener convenientemente identificados los interfaces, cosa que no siempre resulta del todo sencillo.[13] Además, muchas veces ocurre que cambiamos una interfaz por otra, o por otro tipo de tecnología (por ejemplo pasar de Ethernet de cobre a Wireless) con lo cual enlazar las reglas a una interfaz concreta puede no ser una muy buena idea.

Pues bien, una solución que simplifica este problema en gran medida es la de poder tratar con interfaces "virtuales", o "alias" de interfaces (esto realizado a nivel de software y siempre a un alto nivel, nada que implique ajustar los parámetros de red de bajo nivel o crear nuevas interfaces en el sistema). Por tanto asignaremos a ambos tipos de interfaces (interfaz *exterior*, e *interior*) un rol diferente (lo hemos llamado "**Internet**" y "**red local**") y siempre enlazar un conjunto de reglas a un rol dado, en lugar de a una interfaz física. Así, si en el futuro el usuario desea cambiar la interfaz física por otra, no tendría que tocar la política de seguridad, simplemente asignar el rol correspondiente a la nueva interaz. Esto es lo que permite este formulario (mediante sus controles de "Interfaz de Internet" e "Interfaz de red local").

Por otro lado, en cuanto a la "política por defecto" también se debe realizar una aclaración. Mediante este campo, se controla el parámetro "`policy`" de FireHOL para dicha interfaz. Hay que tener en cuenta que FireHOL tiene en el momento de escribir este documento, un fallo de seguridad que afecta a este parámetro, cuando se establece al valor "`accept`". Este fallo de seguriad no significa que no se puedan crear políticas de seguridad de tipo "dejo pasar todo, excepto ciertos protocolos" sino que simplemente se ha de saber lo que se está haciendo (pues FireHOL dejará pasar el tráfico prohibido si no concuerdan los puertos de origen que se definen para los protocolos predefinidos). Una forma de evitar ésto, es redefiniendo los nombres de los protocolos que queramos prohibir, para no hacerlos tan restrictivos en cuanto a los posibles puertos que admiten. Esto se podrá hacer, o bien editando el fichero directamente y añadiendo las correspondientes definiciones de protocolos, o bien (en nuestro caso puede ser la más conveniente) utilizando la sintaxis "`server custom`" o "`router custom`" de FireHOL.

[13] Una técnica común puede ser conectar un cable Ethernet y comprobar el estado del enlace "link" de la interfaz en el sistema operativo, ésto se puede hacer con las herramientas "`mii-tool`" / "`ethtool`" de Linux. Otra posibilidad, es la de asignar una dirección IP concreta a la interfaz, y al conectarla a una red con el mismo direccionamiento, intentar hacer *ping* hacia dicha IP.

H.4.6. Relación con los parámetros de configuración del software FireHOL

Formulario "Configuración general":

Campo	Parámetros	Comentarios
Interfaz de Internet Interfaz de red local	`interface` `router`	Se utilizarán para el valor *<real interface>* de la sentencia `interface`, y para los parámetros *inface* y *outface* de la sentencia `router`, asignándoselos a la sentencia correspondiente según sea 'Internet' o 'red local'.
Política por defecto	`policy`	Se asignará (o creará) la directiva `policy` para la sentencia `interface` correspondiente, con uno de los valores: `drop`, `reject` ó `accept`.
Enmascarar tráfico de salida	`masquerade`	Se asignará (o creará) la directiva `masquerade` para la sentencia `router` correspondiente, de salida a Internet.

Formulario "Reglas automáticas":

Campo	Parámetros	Comentarios

Habilitar reglas automáticas para ...	`interface`
	`server`
	`transparent_proxy`
		Dependiendo de las reglas que se generen de acuerdo a la configuración, se modificarán las diferentes sentencias `interface` para incluir:
		■ directivas `server` para habilitar el acceso a los servicios
		■ directiva `transparent_proxy` para configurar el proxy transparente

Formulario "Reglas definidas por el usuario":

Campo	Parámetros	Comentarios
Reglas de NAT	`nat`	Se crearán las directivas globales `nat` tal y como las ha especificado el usuario.

Reglas de enrutado...	`interface`
	`router`
	*
		Se añadirán las directivas indicadas por el usuario a las sentencias `interface` y `router` que correspondan.

H.4.7. Importación/Exportación de parámetros

Se indican a continuación los parámetros *importables* o *exportables* para el módulo Firewall.

Parámetros importables:

Módulo	Parámetro	Descripción
Proxy (Squid)	`activado`	Si el Proxy Squid local está desactivado, no se configurará ninguna regla automática para este módulo.
Proxy (Squid)	`puerto_escucha`	Se empleará este valor del Proxy Squid para que se genere la correspondiente regla automática que permite a los clientes conectarse a este puerto (a menos que esté DansGuardian activado, en ese caso no se permitirá a los usuarios conectarse directamente a Squid).
Proxy (Squid)	`modo_transparente`	Si está activado (valor `1`) hay que tenerlo en cuenta para la generación de la directiva correspondiente de redirección de tráfico que realizará el Firewall (bajo el contexto de las reglas automáticas).
Filtro de Contenidos (DansGuardian)	`activado`	Se tendrá en cuenta este valor ya que, si DansGuardian está *desactivado*, además de no configurar ninguna regla automática para este módulo se deberá permitir la conexión directa de los usuarios al Proxy (en la sección de reglas automáticas para el Proxy).
Filtro de Contenidos (DansGuardian)	`puerto_escucha`	Se empleará este valor para generar la correspondiente regla automática que permite a los clientes conectarse a este puerto de DansGuardian.

Nota: en la versión actual de la aplicación, no va a ser necesario generar reglas automáticas para el Antivirus *ClamAV*, es por lo que no importamos ningún parámetro en concreto para este módulo. Por tanto actualmente la sección de reglas automáticas del Antivirus siempre aparecerá vacía. No obstante, queda este espacio reservado para futuras aplicaciones.

Parámetros exportables:

Parámetro	Descripción
`activado`	Se establecerá a `1` en el caso de que el módulo se haya configurado para activarse en el arranque (formulario **"Control del módulo"**), o al valor `0` en el caso contrario.

H.5. Formularios de configuración del módulo IDS (Snort)

El último de los interfaces que veremos es el del módulo Detector de intrusiones (IDS) `Snort`.

En este caso, se trata de un software del que hemos elegido un número reducido de parámetros configurables por el usuario, como vimos en la sección 8.7.3. La razón es que se trata de configurar la funcionalidad en un nivel mínimo de complejidad, pero que a la vez resulte útil al usuario. Por tanto, y puesto que la función principal de Snort es la de detectar patrones de intrusión, dejaremos que el usuario realice la configuración en dos niveles:

- Selección de conjuntos de reglas para detección (patrones predefinidos)
- Edición de dichos conjuntos de reglas (edición de patrones)

Además de las funciones típicas de control del módulo, que presentan todos los demás módulos.

Como se comentó anteriormente, `Snort` almacenará las alertas en un fichero de log, y la visualización de las mismas correrá por cuenta de una aplicación externa.

H.5.1. Formulario "Control del Módulo"

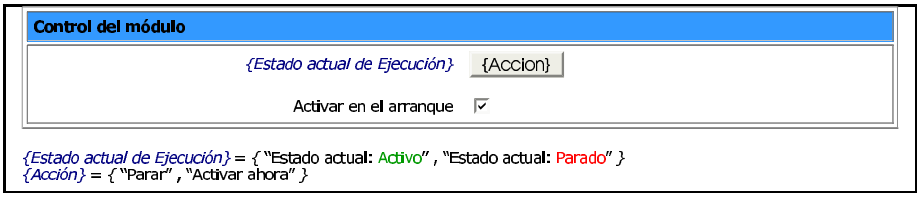

Figura 36: Módulo 'IDS (Snort)' – Control del Módulo

{Acción} *(botón)* Se da la opción de arrancar o parar el servicio (dependiendo de su estado actual). Así mismo, el sistema mostrará el estado actual de ejecución del software `Snort` mediante la etiqueta que hay a la izquierda de dicho botón.

Activar en el arranque *(casilla de verificación)* Mediante esta casilla, se puede configurar Snort para que se active automáticamente en el inicio del sistema.

H.5.2. Formulario "Habilitar reglas"

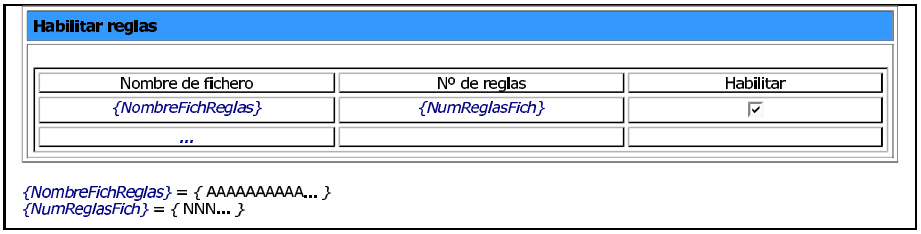

Figura 37: Módulo 'IDS (Snort)' – Habilitar reglas

Este formulario representa un nº variable de campos que el usuario utilizará y gestionará, dependiendo de los disponibles. Se muestran todos los ficheros de reglas predefinidas que hay disponibles, y se dá la opción de habilitarlos o no. La gestión de qué ficheros de reglas habrá en el sistema (añadirlos, eliminarlos) así como su edición, se realiza en el formulario siguiente.

{NombreFichReglas} *(etiqueta)* Muestra el nombre del fichero de reglas que se pretende habilitar/deshabilitar.

{NumReglasFich} *(etiqueta)* Muestra, para cada fichero, el nº de reglas de detección de patrones de las que consta (a modo informativo)

Habilitar *(casilla de verificación)* El usuario, mediante esta casilla, puede seleccionar si se utilizará, o se deshabilitará temporalmente, un fichero de reglas en concreto. Se pueden seleccionar/deseleccionar varios de una sóla vez. Al guardar la configuración, se aplican estos cambios en su conjunto.

Deshabilitar un fichero de reglas no significa que desaparezca del sistema, simplemente que deja de utilizarse temporalmente (el usuario podrá decidir si más tarde activa dicho fichero). Esta configuración "temporal", sin embargo, permanece entre reinicios del sistema. En realidad, es una forma cómoda de seleccionar qué queremos detectar, y qué no.

H.5.3. Formulario "Gestión de reglas"

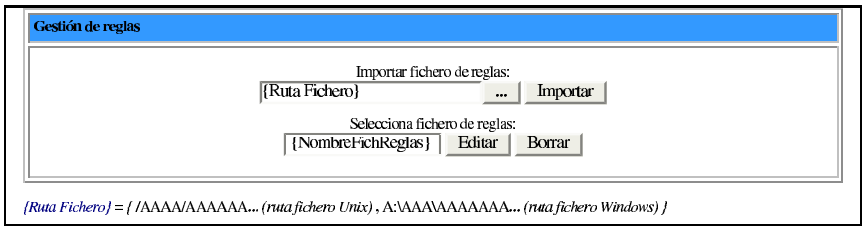

Figura 38: Módulo 'IDS (Snort)' – Gestión de reglas

Como dijimos anteriormente, este formulario es el que nos permite realmente gestionar los ficheros de reglas. Podemos añadir y eliminar ficheros de reglas del sistema, así como editar un fichero en concreto.

Importar *(botón)* Carga el nuevo fichero de reglas en el módulo, y lo incluye entre los demás. Por defecto estará *habilitado* (si bien hasta el siguiente reinicio del servicio no se tendrá en cuenta por `Snort`).

{NombreFichReglas} *(lista desplegable)* Nos permite seleccionar uno de los ficheros de reglas de los existentes en el módulo actualmente, para proceder a su gestión.

Editar *(botón)* Permite la edición del fichero de reglas seleccionado. Para ello, abre el formulario auxiliar "Gestión de reglas – Editar reglas" que veremos a continuación, mostrando el contenido de dicho fichero.

Borrar *(botón)* Elimina del sistema el fichero de reglas seleccionado. Esto es, la eliminación del fichero de reglas es permanente, al contrario que la acción de *deshabilitarlo* que vimos en el anterior módulo, cuya acción era temporal. Se borra del sistema, y desaparece de la lista de ficheros de reglas disponibles.

Formulario auxiliar "Gestión de reglas – Editar reglas"

Este formulario se abre al pulsar sobre el botón *"Editar"* del formulario *"Gestión de reglas"*. Muestra el contenido de un fichero de reglas de `Snort` concreto, y permite su edición.

Figura 39: Módulo 'IDS (Snort)' – Gestión de reglas – Editar reglas

En el cuadro de texto central se muestra el contenido del fichero de reglas al completo. La edición es libre por parte del usuario, pudiendo emplear toda la potencia descriptiva de la sintaxis de detección de patrones de Snort. En la sección 8.7.2 vimos un ejemplo de la sintaxis de estos ficheros.

Aceptar *(botón)* Se acepta la edición por parte del usuario y se actualiza el contenido del fichero en el sistema.

Cancelar *(botón)* Se cierra este formulario sin tener en cuenta las modificaciones realizadas.

H.5.4. Reflexión acerca de los formularios de configuración del módulo IDS

La potencia de Snort es enorme, sim embargo la funcionalidad básica que hemos buscado para nuestro caso es bastante sencilla. No se trata de hacer acciones automáticas en caso de detectar un intento de intrusión en el sistema (lo cual complicaría mucho la configuración e implementación) puesto que no entraba en los planes iniciales. No obstante puede dejarse como una línea futura a seguir.

En nuestro caso, simplemente se realizará la selección de los patrones que se desean utilizar para *detectar* estas intrusiones. La tarea de seleccionarlos, así como de afinarlos convenientemente para minimizar el nº de "falsos positivos", ya constituye de por sí un reto para el usuario con el fin de tener este módulo bien afinado y que resulte útil.

Dicha selección y edición, como hemos visto, es de lo más sencilla que puede ser posible. Se trata de tener un conjunto de ficheros, que se pueden seleccionar y deseleccionar libremente. A la hora de la implementación, esto únicamente significará tener una serie de directivas "`include`" que se comentarán o descomentarán en el correspondiente fichero de configuración (`snort.conf`).

Por otro lado, la edición del fichero es también una tarea muy simple de implementar. Se aceptan los cambios que el usuario realice sin mayor procesamiento por parte del sistema (ya que detectar una correcta sintaxis no ayudaría en demasía, así como podría limitar las posibilidades del software en cuanto a su capacidad expresiva). Es el propio software quien detectará que el fichero pueda tener una sintaxis mal definida, situación que se mostrará convenientemente al usuario (en forma de mensaje de error) en el momento en que intente recargar el 'demonio' (con el botón *"Activar ahora"*).

H.5.5. Relación con los parámetros de configuración del software `Snort`

Formulario "Habilitar reglas":

Campo	Parámetros	Comentarios

Nombre de fichero
Nº de reglas
	`RULE_PATH`
	`include`
	`alert`
		Se leerá del fichero de configuración la variable RULE_PATH así como las directivas `include` (tanto si están comentadas como si no), que referencian a todos los ficheros de reglas configurados en snort, presentándose en esta lista.
		De cada uno de los ficheros existentes, se realizará un conteo del nº de sentencias '`alert`' que contiene. Este dato indicará el nº de reglas que contiene, para mostrar este valor en la lista.
		En caso de que alguno de los ficheros referenciados en la configuración no se encuentren en el directorio RULE_PATH, se generará un mensaje de aviso para informar al usuario de este error en la configuración.

Habilitar	`include`	Se selecciona la directiva `include` seleccionada por el usuario, y se comenta o descomenta en el fichero de configuración, de acuerdo a si se deshabilita o habilita, respectivamente. Las directivas `include` que estén comentadas se mostrarán como deshabilitadas (casilla desmarcada).

Formulario "Gestión de reglas":

Campo	Parámetros	Comentarios
Importar fichero de reglas	`RULE_PATH` `include`	Este campo nomalmente estará vacío, a menos que el usuario decida seleccionar un fichero para importarlo en el sistema. En ese caso, al guardar la configuración, se copia dicho fichero al directorio referenciado por la directiva RULE_PATH y se genera la correspondiente directiva `include`.
Selecciona fichero de reglas:	`include`	Esta lista se forma de igual manera que la del formulario anterior: mediante las directivas `include` que aparecen en el fichero de configuración.
Editar	-	Se lanza el formulario auxiliar de edición para el fichero de reglas seleccionado, pasándole como parámetro el identificador de dicho fichero. Este parámetro sólo permite la selección de uno de los ficheros clasificados en la anterior lista, por motivos de seguridad.
Borrar	`RULE_PATH` `include`	Se obtiene el identificador del fichero seleccionado, y se procede a su borrado. Para ello, se construye la ruta del mismo mediante la variable RULE_PATH y la directiva `include` correspondiente al fichero seleccionado, se localiza el fichero en concreto y se intenta su eliminación. Cualquier error en esta tarea será notificada al usuario. Después de realizar esta tarea, se procederá a eliminar la directiva `include` que hacía referencia a dicho fichero. En caso de que dicho fichero no exista, sólo se borrará la directiva `include`. Este botón sólo permite la eliminación de uno de los ficheros clasificados en la anterior lista, y en la ruta RULE_PATH, por motivos de seguridad.

Formulario auxiliar "Gestión de reglas – Editar reglas":

Campo	Parámetros	Comentarios
Editando el fichero:	`RULE_PATH` `include`	El formulario obtiene el identificador del fichero del formulario principal, y procede a su localización. Para ello, lo seleccionará de entre todos los ficheros del directorio indicado por la variable RULE_PATH con el nombre indicado en las directivas `include` (tanto si están comentadas como si no). El fichero cuya identificación coincida con la pasada por el anterior formulario, será cargado en este campo editable, para permitir su edición. En caso de que dicho fichero no pueda ser localizado, se generará el correspondiente mensaje de error.
Aceptar	`RULE_PATH` `include`	Se guarda el fichero en el directorio RULE_PATH con el nombre que le corresponde (identificado mediante el mismo procedimiento que el anterior paso), y se cierra el formulario. Si se produce un error en esta operación, no se cerrará el formulario, mostrándose este mensaje al usuario. Esta acción no modifica en ninguna medida las directivas `include` del fichero de configuración.
Cancelar	-	Se cierra el formulario, sin realizar ninguna acción.

H.5.6. Importación/Exportación de parámetros

Se indican a continuación los parámetros *importables* o *exportables* para el módulo Detector de intrusiones (IDS).

Parámetros importables:

Este módulo no importará parámetros de ningún otro para realizar su función. Tampoco depende de ningún cambio en la configuración de los demás módulos, para ajustar su funcionamiento.

Parámetros exportables:

Parámetro	Descripción
`activado`	Se establecerá a `1` en el caso de que el módulo se haya configurado para activarse en el arranque (formulario **"Control del módulo"**), o al valor `0` en el caso contrario.

I. Re-implementación de patForms_Definition

patFormsExtensiones/Definition.php

```
<?php
/**
 * Base class for xml based Definitions of a patForms instance
 *
 * A patForms_Definition can be written to and populated from an xml file.
 * It can be used to populate a patForms instance with form elements, rules
 * and other properties.
 *
 * This requires the PEAR XML_Serializer package to be installed and
 * available in the include_path
 *
 * ---
 * Modificada para que funcione con PHP4 --
 * Corregido fallo en read() en el tratamiento de los Enum --
 * ---
 *
 * $Id: Definition.php 272 2005-07-31 14:15:46Z sfuchs $
 *
 * @author              Sven Fuchs <svenfuchs@artweb-design.de>
 * @package             patForms
 * @subpackage  Definition
 * @license             LGPL
 * @copyright   PHP Application Tools <http://www.php-tools.net>
 * @see         patForms_Creator_Definition
 */

/**
 * Base class for xml based Definitions of a patForms instance
 *
 * A patForms_Definition can be written to and populated from an xml file.
 * It can be used to populate a patForms instance with form elements, rules
 * and other properties.
 *
 * @author              Sven Fuchs <svenfuchs@artweb-design.de>
 * @package             patForms
 * @subpackage  Definition
 * @license             LGPL
 * @copyright   PHP Application Tools <http://www.php-tools.net>
 * @see         patForms_Creator_Definition
 */
class patForms_Definition {

    /**
        * Stores the definitions properties
        *
        * @var      array
        * @access       private
        */
        var $data = array();

    /**
        * The constructor
        *
        * Adds a timestamp as mtime to the definition's data, that might be
```

```
     * overwritten by loading a definition file
     *
     * @access        public
     * @param   string    $name          The name of the form
     * @param   string    $autoValidate  The autoValidate property of the form
     */
    function __construct($name, $autoValidate = '')
    {
            $this->data['name'] = $name;
            //$this->data['mtime'] = time();
            if ($autoValidate) {
                    $this->data['autoValidate'] = $autoValidate;
            }
    }

/**
     * Constructor para PHP4
     *
     * @access        public
     * @param   string    $name
     * @param   string    $autoValidate
     */
    function patForms_Definition($name, $autoValidate = '')
    {
            $this->__construct($name, $autoValidate);
    }

/**
     * Factory method to create a new patForms_Definition instance
     *
     * not implemented
     *
     * @static
     * @access        public
     * @param       array     $conf   Data needed by the factory method
     */
    function create($conf)
    {
    }

/**
     * Returns definition properties
     *
     * @access        public
     * @param       array     $name  Name of the property to recieve
     * @return  mixed   The requested property
     * @todo    Remove this and replace by classical get/set methods?
     */
    function __get($name)
    {
            if ( isset ($this->data[$name])) {
                    return $this->data[$name];
            }
    }

/**
     * Adds the definition for a form element
     *
```

```
     * @access        public
     * @param         string    $name   The name of the element
     * @param         string    $type    The type of the element
     * @param         array     $attributes     Attributes for the element
     * @param         array     $rules   Definitions for patForm_Rules
     * @todo      Change protocol to addElement(array $data) to be more flexible
     */
    function addElement($name, $type, $attributes = array(), $rules = array())
    {
            if (is_array($type)) {
                    extract($type);
            }

            $this->data['elements'][$name]['name'] = $name;
            $this->data['elements'][$name]['type'] = $type;

            foreach ($attributes as $key => $value) {
                    $value = $this->cast($value);
                    $this->data['elements'][$name]['attributes'][$key] = $value;
            }
            foreach ($rules as $key => $rule) {
                    $this->data['elements'][$name]['rules'][$key] = $rule;
            }
    }

/**
     * Loads the definitions properties from an xml file
     *
     * @access        public
     * @param         string    $filename    The filename of the xml file
     */
    function load($filename)
    {
            $data = $this->read($filename);

            foreach ($data as $key => $value) {
                    if ($key == 'elements') {
                            foreach ($value as $name => $element) {
                                    $this->addElement($name, $element);
                            }
                    } else {
                            $this->data[$key] = $this->cast($value);
                    }
            }
    }

/**
     * Writes the definitions properties to an xml file
     *
     * @access        public
     * @param         string    $filename    The filename of the xml file
     */
    function save($filename)
    {
            $this->write($filename, $this->data);
    }

/**
```

```
     * Reads an xml file
     *
     * @access        protected
     * @param         string    $filename    The filename of the xml file
     * @return    array     The data read from the file
     */
     function read($filename)
     {
             require_once "XML/Unserializer.php";

             $xml = file_get_contents($filename);
              $unserializer  = new XML_Unserializer(array (
                     'ignoreKeys' => array('tag')
              ));
              $unserializer ->unserialize($xml);
             return  $unserializer ->getUnserializedData();
     }

/**
     * Writes data to an xml file
     *
     * @access        protected
     * @param         string    $filename    The filename of the xml file
     * @param         array     $data        The data to write
     */
     function write($filename, $data)
     {
             require_once "XML/Serializer.php";

              $serializer  = new XML_Serializer(array (
                     'addDecl' => true,
                     'encoding' => 'ISO-8859-1',
                     'indent' => '  ',
                     'rootName' => 'form',
                     'defaultTagName' => 'tag'
              ));
              $serializer ->serialize($data);
             $xml = $serializer->getSerializedData();

             $fp = fopen($filename, 'w+');
             fputs($fp, $xml);
              fclose ($fp );
     }

/**
     * Tries to guess the correct datatype for a variable and casts it if
     * necessary
     *
     * Please note: this doesn't seem to work right now with patForms_Elements
     * and will therefor return the value parameter without changing anything
     * for now. This may change in future.
     *
     * @access        protected
     * @param         string    $value      The value to check
     * @return    array     The value (probably cast to another type)
     */
     function cast($value)
     {
```

```
                return $value;

                // seems as if patForms_Element(s) are broken here
                // e.g. in patForms_Element_Text::serializeHtmlDefault()
                // at line 245 if( $this->attributes['display'] == 'no' )
                // will result to true if the display attribute is set
                // to (php boolean) true
                // so casting the 'true'/'false' and 'yes'/'no' values
                // would break intended behaviour here

                if (is_array($value) OR is_bool($value)) {
                        return $value;
                }
                if ($value === 'true') {
                        return true;
                }
                if ($value === 'false') {
                        return false;
                }
                if (preg_match('/^[+-]?[0-9]+$/', $value)) {
                        settype($value, 'int');
                        return $value;
                }
                if (preg_match('/^[+-]?[0-9]*\.[0-9]+$/', $value)) {
                        settype($value, 'double');
                        return $value;
                }
                return $value;
        }
}


?>
```

J. Interfaz de la clase estática ManejoErrores

Lista de Métodos:

object &**almacena_error** (*object* **$error**)
boolean **esFatal** (*object* **$error**)
void **establecer_modo** (*integer* **$modo**)
array &**formatear_errores** ([*boolean* **$vaciar** = false])
string &**formatea_error** (*object* &**$error**)
object &**maneja_error** (*object* &**$error**)
mixed &**presenta_error** (*object* &**$error**)
string **traduce_nivel_error** (*integer* **$nivel**)
boolean **validar_formulario** (*object* &**$form**)
array &**_errores** ()

Descripción de los Métodos:

---

almacena_error: Almacenar un error.

Posteriormente se pueden recuperar (formateados) con el método formatear_errores()

- return: mismo objeto patError que recibió
- access: protected

object &**almacena_error** (*object* **$error**)

- *object* ***$error****: El objeto patError que se recibe como parámetro de la función callback*

---

esFatal: Determina si el parámetro es de tipo patError, y si se trata de un error fatal (es decir, si debe detener el flujo normal del programa)

Si la clase está en modo ALMACENAR, también examina todos los errores que se han ido produciendo, y si alguno es fatal también devuelve true.

- return: si el error es fatal
- access: public

boolean **esFatal** (*object* **$error**)

- *object* ***$error****: El error a examinar*

---

establecer_modo: Establece el modo de gestión de los errores.

Puede ser:
```
- FW_MANEJOERRORES_MODO_ALMACENAR (0) =>Almacenar los errores
- FW_MANEJOERRORES_MODO_PRESENTAR (1) =>Presentar los errores
```

- access: public

void **establecer_modo** (*integer* **$modo**)

- *integer* ***$modo****: El modo de operación, una de las constantes anteriores*

---

formatea_error: Formatear un error.

- return: El error formateado
- access: protected

string &**formatea_error** (*object* &**$error**)

- *object* &***$error****: El objeto patError que se desea formatear con código HTML*

---

maneja_error: Método callback para el manejo de un error.

Según el modo de operación (establecido con *ManejoErrores::establecer_modo()*) se almacenará este error en la lista global de errores, o bien se presentará directamente.
Este método es apto para utilizarse como método 'callback' en *patErrorManager::setErrorHandling()*

- return: mismo objeto patError que recibió
- access: public

object &**maneja_error** (*object* &**$error**)

- *object* &***$error****: El objeto patError que se recibe como parámetro de la función callback*

---

presenta_error: Presentar un error.

Formatea y presenta un error

- return: El mismo objeto patError que recibió
- access: protected

object &**presenta_error** (*object* &**$error**)

- *object &**$error**: El objeto patError que se recibe como parámetro de la función callback*

---

traduce_nivel_error: Método auxiliar que traduce el tipo (nivel) del error en algo más legible a nivel humano.

- return: La cadena descriptiva que lo representa
- access: protected

string **traduce_nivel_error** (*integer* **$nivel**)

- *integer **$nivel**: El tipo de error (constante E_NOTICE, E_WARNING...)*

---

validar_formulario: Comprueba si un formulario tiene errores de validación, y genera el correspondiente mensaje de error (mediante patErrorManager).

Se utilizará si se desea proseguir con el procesamiento de los valores del formulario (con un uso similar al resultado de *patForms::validateForm()*) – Nota: esta función también llama a validateForm().

- return: Si hubo o no errores de validación
- access: public

boolean **validar_formulario** (*object* &**$form**)

- *object &**$form**: El formulario (patForms) para examinar errores*

_errores: Obtiene la lista de errores almacenados.

Por si se quiere hacer algo más "creativo" con ellos...

- return: Array de objetos patError (por referencia)
- access: protected

array &**_errores** ()

K. Arranque de un sistema GNU/LINUX

El comienzo del proceso de arranque de un PC, comienza por la ejecución de código en ROM, BIOS (Basic Input/Output System), que carga la información del sector 0 y cilindro 0 del disco de arranque. El disco de arranque suele estar configurado normalmente como la primera disquetera o lector de CD-Rom. La BIOS intenta ejecutar entonces este sector. En la mayoría de discos arrancables, el sector 0, cilindro 0 contiene, o bien código de un gestor de arranque como LILO o GRUB, que localiza el núcleo, lo carga y lo ejecuta para comenzar el proceso de arranque propiamente dicho. El comienzo del núcleo de un sistema operativo, como Linux.

Si un núcleo de Linux se ha copiado a un disquete (mediante una copia raw), el primer sector del disquete será el primer sector del núcleo de Linux. Este primer sector continuará el proceso de arranque cargando el resto del núcleo desde el disco de arranque.

Cuando el núcleo se ha cargado completamente, inicializa los drivers de los dispositivos y sus estructuras de datos internas. Una vez que ha sido inicializado completamente, consulta una posición especial en su imagen llamada "*ramdisk word*". Esta palabra contiene la información de dónde y cómo encontrar el sistema de ficheros raíz. Un sistema de ficheros raíz es simplemente un sistema de ficheros que será montado en el directorio raíz "/". Hay que decirle al núcleo dónde debe buscar el sistema de ficheros raíz; si no puede encontrar una imagen que se pueda cargar, se parará el proceso de arranque.

En algunas situaciones, normalmente cuando se arranca desde un disquete, el sistema de ficheros raíz se carga en memoria RAM a la que se accede como si fuera un disco. La RAM es varios órdenes de magnitud más rápida que un disquete, de manera que se gana en velocidad. Además, el núcleo puede cargar un sistema de ficheros comprimido desde el disquete y descomprimirlo en el disco de memoria, permitiendo así introducir muchos más ficheros en un solo disquete de arranque.

Una vez que el sistema ha cargado adecuadamente el sistema de ficheros raíz, se procede a ejecutar el programa "init" (en /bin o /sbin). El proceso "init" lee su fichero de configuración (/etc/inittab), busca una línea que contiene "sysinit", y ejecuta el script asociado. El script que se ejecuta puede variar de una distribución de GNU/Linux a otra. En sistemas Debian, esta línea es similar a:

```
si::sysinit:/etc/init.d/rcS
```

El script que pasa a ejecutarse (/etc/init.d/rcS) procede a ejecutar, de manera modular, los diferentes scripts que aparecen en /etc/rcS.d/. Estos se encargan de configurar el sistema de manera básica, iniciando los servicios indispensables del sistema, como la ejecución de "fsck" para comprobar el sistema de ficheros raíz, la carga de los módulos del núcleo necesarios, la inicialización del "swap", de la red, montando los discos especificados en /etc/fstab, etc.

Cuando el script de sysinit finaliza su ejecución, se devuelve el control al proceso "init", que entonces procede a entrar en el nivel de ejecución (o runlevel) por defecto, especificado en /etc/inittab por la palabra "initdefault". Será algo parecido a la siguiente línea:

```
id:2:initdefault:
```

El nivel de ejecución elegido incluirá un conjunto de scripts, ubicados en /etc/rcN.d (donde N es el número correspondiente al nivel de ejecución), que procederán a ejecutarse.

Por último, en el fichero "/etc/inittab" se especifica un programa como "getty", que se encarga de manejar las comunicaciones desde la consola y los ttys (terminales remotos). Es este programa el responsable de mostrar el conocido mensaje de "login:". Se le pasará posteriormente el control al programa "login", que se encargará de validar la entrada del usuario al sistema.

Arranque con *initrd*

El fichero especial /dev/initrd es un dispositivo de bloques de sólo lectura. Este dispositivo es un disco RAM que es inicializado por el gestor de arranque antes de que se arranque el núcleo. Entonces, el núcleo puede utilizar el contenido de /dev/initrd para realizar un arranque en dos fases.

En la primera fase, el núcleo comienza su ejecución y monta un sistema de ficheros raíz a partir de los contenidos de /dev/initrd.

Durante la segunda fase, se cargan drivers y módulos adicionales a partir del contenido de sistema de ficheros inicial. Después de cargar los módulos adicionales, un nuevo sistema de ficheros (el sistema de ficheros raíz habitual) se monta desde un dispositivo diferente.

A continuación vemos los pasos del proceso de arranque cuando se utiliza "initrd":

1. El gestor de arranque carga el núcleo y los contenidos de /dev/initrd en memoria.
2. Cuando comienza la ejecución del núcleo, esté se descomprime y copia el contenido de /dev/initrd en el dispositivo /dev/ram0, liberando posteriormente la memoria usada por /dev/initrd.
3. El núcleo monta el fichero /dev/ram0 como el sistema de ficheros raíz inicial, en modo lectura/escritura.
4. Si el sistema de ficheros raíz "normal" coincide con el sistema de ficheros raíz inicial (/dev/ram0), entonces el núcleo salta hasta el último paso de la secuencia de arranque habitual.
5. Si el ejecutable /linuxrc existe en el sistema de ficheros inicial, se ejecuta con uid 0. Este fichero debe tener permisos de ejecución. /linuxrc puede ser cualquier ejecutable, incluyendo un script de shell.
6. Si /linuxrc no se ejecuta o bien ya ha terminado su ejecución, se monta el sistema de ficheros normal. (Si /linuxrc termina con algún sistema de ficheros montado en el sistema de ficheros raíz inicial, entonces el comportamiento del núcleo no está especificado).
7. Si el sistema de ficheros normal contiene un directorio llamado /initrd, entonces el dispositivo '/dev/ram0' se mueve de "/" a "/initrd". En caso contrario, /dev/ram0 es desmontado. (Cuando se mueve de "/" a "/initrd", /dev/ram0 no se desmonta y, por tanto, los procesos pueden continuar ejecutándose desde /dev/ram0. Si el directorio /initrd no existe en el sistema de ficheros normal y cualquier proceso continúa en ejecución desde /dev/ram0 cuando /linuxrc termina, el comportamiento del núcleo no está especificado).
8. El proceso de arranque habitual (ejecución de "init") tiene lugar en el sistema de ficheros normal.

L. Arranque del sistema LiveCD-Metadistros

En el caso de Metadistros, el proceso de arranque tiene lugar desde un CD-ROM. Para crear este CD-ROM de arranque se hace uso de la herramienta ISOLINUX, que es capaz de arrancar el sistema desde un CD-ROM y pasar posteriormente el control al proceso "init". Podemos encontrar más información sobre Isolinux en: `http://syslinux.zytor.com/iso.php`.

Lo que vamos a analizar en este apartado es el proceso que tiene lugar desde que la BIOS pasa el control al CD-ROM hasta que pasa a ejecutarse el proceso "init". Todo este proceso se lleva a cabo a partir de un conjunto de herramientas denominadas "calzador".

El calzador incluye, entre otras, la herramienta isolinux y todos sus ficheros de configuración. Es por ello por lo que el directorio en el que se encuentra el calzador en el CD-ROM se llama /isolinux.

A continuación se citan los ficheros de configuración del programa ISOLINUX. Se puede encontrar información más detallada acerca de los mismos en el enlace anteriormente citado. Los ficheros de configuración son los siguientes:

- isolinux.bin
- isolinux.cat
- isolinux.cfg
- isolinux.msg
- bootlogo
- f2.msg y f3.msg
- memtest
- spanish.kbd
- vmlinuz.usuario
- initrd.gz

Todos estos ficheros se encontrarán en el directorio /isolinux del CD final.

El fichero más relevante de los anteriores es initrd. Ya se ha descrito anteriormente la forma de arrancar un sistema cuando se utiliza "initrd". En el caso de Metadistros, initrd debe ser capaz de preparar el sistema para arrancar la imagen de la distribución. Esto incluye cargar los módulos que sean necesarios, localizar la imagen de la distribución, etc. A lo largo de esta apartado, veremos cómo es posible conseguir todo esto.

Estructura del contenido de initrd para Meta-Distros

El directorio '/' El directorio raíz contiene una estructura de directorios similar a la de cualquier sistema GNU/Linux, pero con algunas modificaciones. La estructura de directorios es la siguiente:

```
.
|–META
|–bin ->/META/bin
|–boot ->/META/boot
|–cdrom
```

```
|–dev
|–etc
|–lib ->/META/lib
|–linuxrc
|–mnt
|–modules
|–opt ->/META/opt
|–proc sbin ->/META/sbin
|–static
|–tmp ->/var/tmp
\–usr ->/META/usr
```

Si lo comparamos con la estructura habitual de una distribución cualquiera vemos que hay directorios que faltan (como por ejemplo /home, /root, /floppy, /var...), hay alguno nuevo (/META) y hay muchos que son enlaces a otros dentro del directorio /META (salvo /tmp que apunta a /var/tmp; de esta manera se tienen unificados todos los directorios temporales, que necesitan permisos de escritura). Es importante señalar la existencia del fichero /linuxrc, el cual es responsable del arranque del sistema una vez se ha montado initrd.

El Directorio META El contenido del mismo es el siguiente:

```
.
|– bin ->/cdrom/META/bin
|– boot ->/cdrom/META/boot
|– etc ->/cdrom/META/etc
|– lib ->/cdrom/META/lib
|– opt ->/cdrom/META/opt
|– sbin ->/cdrom/META/sbin
|– usr ->/cdrom/META/usr
\– var ->/cdrom/META/var
```

El contenido de este directorio no es relevante puesto que, una vez localizada la imagen, se montará en este directorio, sobreescribiendo su contenido.

El directorio 'modules' En este directorio, se encuentran aquellos módulos imprescindibles para habilitar los diferentes métodos de arranque del sistema. Por tanto, deberá incluir módulos para dar soporte a aquellos dispositivos que pueden ser potencialmente utilizados para arrancar el sistema, como pueden ser dispositivos IDE, SCSI, USB o PCMCIA.

Por otro lado, debe existir el módulo "cloop.o". Éste permite montar en el loopback imágenes comprimidas de un sistema GNU/Linux. Este módulo es necesario para poder arrancar sistemas que hayan sido previamente comprimidos para optimizar el espacio y poder añadir más paquetes y programas a la meta-distribución. De esta manera, se pueden crear metadistros que ocupen hasta 2 GB. en un solo CD.

El directorio 'keymaps' En la documentación de Isolinux se comenta que no hay problema si no existe este fichero, porque una vez esté cargado el sistema de la imagen, ya se incluyen estos mapas de teclado.

El directorio 'dev' Contiene los ficheros de dispositivos. No hay cambios significativos con respecto a un sistema normal, a excepción de que el número de dispositivos se ha reducido considerablemente, puesto que sólo se han incluido aquellos dispositivos que son imprescindibles para initrd.

El directorio 'etc' Dentro de este directorio existe un reducido grupo de ficheros de configuración:

.

|– auto.mnt

|– exports

|– filesystems

|– fstab

|– group

|– ld.so.conf ->/META/etc/ld.so.conf

|– mtab

|– passwd

|– resolv.conf

\– shadow

El directorio 'static': BusyBox BusyBox combina versiones pequeñas de muchas utilidades comunes UNIX en un único ejecutable pequeño. Provee un conjunto de reemplazos para la mayoría de utilidades básicas, como grep, gzip, tar, etc.

Las utilidades de BusyBox suelen tener un menor número de opciones que las aplicaciones "reales", pero mantienen las funcionalidades básicas. BusyBox se ha diseñado con la idea de minimizar el tamaño y los recursos que ocupan estas aplicaciones al máximo.

Para el caso de Meta-Distros, todas las utilidades compiladas en BusyBox se han ubicado en /static. Estas utilidades son necesarias para los scripts de arranque puesto que, entre otras, ahí se encuentran utilidades como la shell. Para acceder a las utilidades compiladas dentro de BusyBox directamente, es necesario crear enlaces "duros" desde cada una de las utilidades al ejecutable "busybox", que se encargará de ejecutar estas aplicaciones adecuadamente. El resultado de utilizar BusyBox es que se dispone de un conjunto mínimo de ejecutables básicos para cualquier sistema UNIX que permiten la ejecución de los diversos scripts de arranque en initrd.

El fichero '/linuxrc': el control del arranque en initrd.

Este fichero es el responsable de la ejecución del arranque hasta que se le pasa el control al proceso "init". Es, con mucho, el fichero más importante de todo el sistema initrd, y debe ser analizado en detalle.

La ejecución de linuxrc tiene como objetivo principal preparar el sistema para arrancar la imagen de la meta-distribución que hayamos creado. En realidad, éste es el objetivo del calzador, dentro del cual se encuentra toda la parte relacionada con initrd y linuxrc. Entonces, el resultado de la ejecución de linuxrc debe ser un sistema preparado para trabajar con una meta-distribución cualquiera que, en principio, no debe reunir ningún tipo de requisito ni condición especial para funcionar.

Una de las particularidades de todo esto es que se debe contemplar la posibilidad de ejecutar el sistema en modo live, lo que quiere decir que no se instala nada en el disco duro; toda la ejecución del sistema operativo tiene lugar desde una unidad de CD-ROM.

La enorme cantidad de hardware disponible en el mercado, así como las dificultades que conlleva el disponer de un sistema de ficheros de solo lectura (como es el caso de un CD-ROM) convierten al calzador en un elemento delicado: será necesario llevar a cabo todo un conjunto de acciones

concretas para hacer posible todo esto. Y esto es precisamente lo que vamos a analizar en este apartado.

El proceso completo lo podemos estructurar, a grandes rasgos, en los siguientes pasos:

1. Preparación del sistema para arrancar desde la mayor cantidad posible de dispositivos de CD-ROM. Esto incluye una gran variedad de dispositivos y buses a los que se conectan, incluyendo buses IDE, SCSI o incluso dispositivos que se conectan al puerto paralelo.
   Para hacer hacer posible lo anterior, es necesario disponer de los módulos del núcleo necesarios para dar soporte a toda esta variedad de hardware. En particular, serán necesarios los módulos SCSI. Estos se encuentran en el directorio */modules* del sistema de ficheros de *initrd*.

2. Detección de los dispositivos SCSI. En primer lugar, se analiza si se pasó por la línea de comandos la palabra "noscsi", que deshabilitaría la detección de este tipo de dispositivos. Si no es el caso, se procede a cargar cada uno de los módulos de que se dispone. Si la carga de alguno tiene éxito quiere decir que existe el dispositivo al que da soporte.

3. Preparación de la imagen de la distribución. El siguiente paso consiste en localizar la imagen de la distribución con la que se va a trabajar (que es la meta-distribución propiamente dicha que hemos creado).
   Para ello, se introduce en la variable $DEVICES el conjunto de dispositivos que potencialmente pueden tener la imagen. Estos incluyen a dispositivos IDE (/dev/hd?), SCSI (/dev/scd?) y CDROM por puerto paralelo (/dev/pcd?). Entonces, se procede a intentar montar todos estos dispositivos.
   Si hay éxito, se comprueba la existencia del directorio META dentro del mismo (es decir, el directorio /cdrom/META). Si existe este directorio, la imagen ha sido localizada. Lo que falta es determinar si la imagen está comprimida (con cloop) o no. Para ello, se hace una búsqueda del fichero "META" o "META.cloop" dentro del directorio /cdrom/META. Si no se encontrara ninguno de los dos ficheros, se abortaría el proceso de arranque y se saldría a una consola muy limitada. En caso contrario, se montaría la imagen en */META*.

4. Algunos detalles más. Se realizan algunas tareas como copiar el caché de las bibliotecas de funciones para el enlazador.
   Además, se establece la nueva variable del PATH con los directorios habituales de cualquier distribución (/bin, /sbin,...) más el directorio de los binarios del calzador (/cdrom/isolinux/cdroot/sbin). Inicialmente, esta variable se inicializó únicamente con el valor "/static", que era la ruta donde se encontraban los binarios necesarios en la primera fase.
   Por último, se borran los directorios "/modules" y "/static", que ya no son necesarios. De esta manera, a partir de ahora se utilizarán los módulos y ejecutables de la imagen montada.

5. Creación de los directorios con permiso de escritura. En cualquier sistema GNU/Linux es necesario disponer de algunas partes del sistema de ficheros con permisos de escritura. Esto es imprescindible para que determinados programas puedan escribir cierta información. Pero también es necesario para que un usuario pueda acceder en escritura al disco.
   En el caso habitual de una distribución instalada en el disco duro, dada la naturaleza innata de este dispositivo, es posible acceder en escritura al sistema de ficheros. Sin embargo, en el caso que nos ocupa, debido a que la imagen de la distribución se encuentra en un dispositivo de sólo lectura (como es el caso de un CD-ROM) y a que uno de los objetivos de Meta-Distros es no tener que modificar absolutamente ni un bit del disco duro del usuario, hay que buscar algún método para habilitar la escritura en el sistema de ficheros.
   La solución pasa por utilizar parte de la memoria RAM como una unidad donde montar una parte del sistema de ficheros. El primer paso consiste en determinar la cantidad de memoria RAM con la que cuenta el sistema. Conocido este dato, es necesario decidir qué parte de la misma se va a destinar para la partición. Esta partición se montará en el directorio /ramdisk,

dentro del cual se crearán los directorios "home" y "var".
El siguiente paso es crear sendos enlaces de /home y de /var a los correspondientes en "/ramdisk". Es importante señalar que serán necesarios más directorios con permisos de escritura fuera de estos dos: /etc/sysconfig, /etc/cups/,... Si recordamos, initrd se montó precisamente en RAM y, por tanto, es posible escribir en el mismo, así que lo único que hemos de hacer es crear dichos directorio en el sistema de ficheros. Una vez creados todos los directorios con permiso de escritura, lo único que falta es copiar todos los ficheros necesarios a los mismos.

6. Preparación de los scripts de arranque y de otros ficheros de configuración. El siguiente paso durante la ejecución de linuxrc es preparar el arranque para cuando el proceso INIT tome el control. Esto involucra fundamentalmente a los directorios /etc/init.d y /etc/rc*.d. Lo que se hace es, en primer lugar, eliminar todos los ficheros que afectan al arranque para, posteriormente, "reconstruir" el sistema de arranque como interese.
   En este sentido, lo que se hace es regenerar el fichero /etc/init.d/rcS, que se hace apuntar al script "init.sh" que se encuentra en el calzador. De esta manera, queda modificado la secuencia habitual de arranque, para pasar a una en la que se tiene en cuenta todas las condiciones especiales del sistema del que estamos hablando (este proceso se comentará posteriormente).
   Aparte, se crean otros enlaces relacionados con el arranque. Así mismo, se copia del calzador una plantilla del fichero de configuración de X-Window al directorio /etc/X11.

7. Por último, se habilita la capacidad del núcleo para cargar módulos, se cambia la ubicación del sistema de ficheros raíz y se devuelve el control a INIT, que pasa a ejecutar el fichero /etc/init.d/rcS.

Continuación del arranque: el proceso INIT

Una vez que termina de ejecutarse el script linuxrc, el control del arranque pasa a INIT. Como ya se ha comentado, este programa analiza el fichero /etc/inittab en busca de la línea sysinit, que indica el script que debe ejecutar.

En el caso que nos ocupa, se procederá a ejecutar /etc/init.d/rcS. Habitualmente este script lanza la ejecución de todos los scripts ubicados en /etc/rcS.d/. Sin embargo, en el caso de metadistros, el fichero /etc/init.d/rcS es un enlace al script "init.sh" que se encuentra en el directorio de los binarios del calzador.

Por tanto, cuando el proceso INIT toma el control del arranque, lo que sucede realmente es que se produce la ejecución de /cdrom/isolinux/cdroot/sbin/init.sh.

El proceso que tiene lugar se puede esquematizar en los siguientes pasos:

1. El primer paso que tiene lugar es la carga de los ficheros de configuración de Metadistros. Estos se encuentran en el directorio /cdrom/isolinux/conf. Cada uno de ellos contiene diferentes variables que controlan diversos aspectos del funcionamiento de la meta-distribución, una vez se está ejecutando. Estos ficheros de configuración son los siguientes:

   q.conf: Aquí se define el conjunto de variables que aparece en el diagrama de flujo. El significado de las mismas es el siguiente:

   - QSPLASH: Indica si se muestra o no la pantalla de "Splash" durante la carga del sistema, pantalla que "oculta" los mensajes del núcleo y muestra algún tipo de indicador de carga, con la idea de convertir el arranque en algo más divertido.
   - QLANGUAGE: Controla si se le pregunta o no al usuario por el lenguaje que debe utilizar para la distribución. QPASS: Indica si se pregunta al usuario por la contraseña de root.

- QUSER: Controla si se pregunta o no por el nombre del usuario del sistema y su contraseña (usuario que existirá además de root).
- QHOST: Indica si se pregunta por el nombre del equipo o no.
- QINSTALL: Controla si se pregunta al usuario si quiere instalar la distribución al disco duro.
- QX
- QEXPERT
- QNET: Indica si se pregunta o no por la configuración manual de los parámetros de red.

var.conf: En este caso, las variables que se definen son las siguientes:

- DISTRO: Controla el nombre de la distribución.
- USERNAME: Nombre del usuario alternativo a root.
- UPASSWORD: Contraseña del usuario anterior.
- RPASSWORD: Contraseña de root.
- HOSTNAME: Nombre del equipo.
- LANGUAGE: Lenguaje que se utilizará.
- INSTALL: Indica si se debe instalar la distribución o se debe hacer un arranque live.
- ROOT: Controla si se crea o no un usuario alternativo a root.
- EXPERT: Si se activa, se pueden seleccionar opciones más complejas.
- XCONF: Esta opción está pensada para monitores y tarjetas gráficas antiguas.
- DHCP: Activada, indica una configuración de la red mediante DHCP.
- STARTUSER: Nombre del usuario con el que arrancará la distribución.
- STARTX: Indica si se arranca o no el servidor X cuando arranque la distribución.
- LTSP: Controla si se activa o no el servicio LTSP6.

lang.conf: Este último fichero de configuración contiene información acerca de las "locales" específicas para cada idioma. Éstas son las variables que contienen la información relativa al lenguaje que se va a utilizar, el tipo de moneda, y demás información regional propia de cada país.

2. El siguiente paso consiste en analizar lo que se ha introducido por la línea de comandos. Es posible pasar comandos al núcleo antes de arrancar el sistema. También es posible pasar a través de la línea de comandos las variables anteriores con un valor diferente del predeterminado. Este valor introducido prevalece frente al que se estableció por defecto. Así, se modifica el comportamiento de la meta-distribución en relación con el habitual.

3. Ahora se procede a configurar adecuadamente el teclado. Para ello, se analiza la variable KEYTABLE del fichero "lang.conf" y, en función del valor de la misma, se carga la tabla de teclas adecuada.

4. El siguiente paso es establecer el PATH. Se introducen las rutas habituales en cualquier sistema GNU/Linux, sin incluir las relacionadas con X-WINDOW, y añadiendo el directorio de los binarios del calzador.

5. Llegados a este punto, se produce el inicio de la detección de hardware (a partir de la ejecución del script "hw-detect.sh"). Este proceso tendrá lugar en segundo plano, mientras continúa la ejecución del script principal.
   Analizaremos en detalle el proceso de detección de hardware en los apartados siguientes.

6. Lo siguiente que tiene lugar es el análisis de las diferentes variables que se encontraron en los ficheros de configuración para realizar las acciones pertinentes.

Así pues, por poner un ejemplo, si la variable QSPLASH tiene un valor verdadero (o "Y"), se mostrará una pantalla de presentación durante el arranque. Es importante hacer hincapié en dos variables particulares:
QNET: Si esta variable contiene un valor verdadero, se procederá a configurar la red manualmente a partir de la ejecución del script "netconfig.sh".
QX: En este caso, si la variable es verdadera se ejecutará "x-detect.sh". En caso contrario, se ejecutará "mkxf86config.sh".

7. Una vez analizadas todas las variables de configuración, se procede a crear un usuario normal y el root, a partir de la ejecución de "make-user.sh".

8. Se continúa el arranque estableciendo el nombre del sistema, generando el fichero /etc/hosts a partir de la información anterior, y se arranca el demonio CUPS y otros servicios.

9. Por último, si la variable STARTX es verdadera, se arrancará X-WINDOW. En este caso, se ampliará el contenido de la variable PATH, introduciendo las rutas de los binarios de X, y se ejecutará el comando xinit con los parámetros adecuados, para lanzar la sesión de X. Si STARTX es falsa, el arranque termina en una consola de usuario.

10. En cualquiera de los dos casos anteriores, una vez que la sesión termina, se produce la ejecución del comando "halt", que cierra el sistema.

Proceso de detección de hardware

Durante el proceso de arranque del sistema (mientras se ejecuta "init"), tiene lugar la ejecución en segundo plano del script "hw-detect.sh". Este script es el verdadero responsable de la detección de hardware en la meta-distribución.

Vamos a analizar en este apartado los diferentes pasos que tienen lugar durante la ejecución del mismo:

1. En primer lugar, se establece el reloj del sistema a partir del reloj hardware, mediante el comando "hwclock". Se le pasan las opciones "-s" (para establecer la hora del sistema a partir de la del reloj hardware).

2. Se monta el sistema de ficheros raíz como lectura y escritura.

3. El siguiente paso es dar soporte para dispositivos PCMCIA. Para ello, se carga el módulo principal: "pcmacia_core.o". Y después se intenta cargar los módulos siguientes:
1. yenta_socket.o: soporte para CardBus (la nueva versión de 32 bits de las tradicionales tarjetas PCMCIA de 16 bits).
2. i82365.o: soporte para "bridges" para bus ISA, de equipos antiguos.
3. tcic.o: soporte para unos "bridges" especiales de equipos antiguos.
Así se consigue habilitar el soporte tanto para tarjetas CardBus como para hardware antiguo especial. Si la carga de ninguno de estos módulos tiene éxito, se descarga el módulo "pcmcia_core" y se establece PCMCIA=0. En caso contrario, se carga el módulo "ds" (que también forma parte del soporte básico de PCMCIA junto con el módulo "pcmcia_core") y se ejecuta el comando "cardmgr"14. Así queda habilitado el soporte para dispositivos PCMCIA.

4. Ahora se pasa al soporte de dispositivos USB. El procedimiento es similar al anterior. En primer lugar, se carga el módulo "usbcore.o", que proporciona el soporte básico para dispositivos USB. Después, es necesario dar soporte al hardware USB específico que existe en el sistema. Para ello, hay que cargar uno de los módulos siguientes: "usb-uhci.o" o "usb-ohci.o" En este punto, y suponiendo que los módulos se hayan cargado sin problemas, se monta el sistema de ficheros usbdevfs, para permitir, mediante /proc/, el acceso a la información de

los dispositivos USB conectados. Sin embargo, aún no existe soporte para ningún dispositivo USB concreto. Sólo está cargada la base común para todos los dispositivos USB. La carga de los módulos concretos según el hardware que haya conectado tendrá lugar más adelante. Si la carga de los módulos ha tenido problemas, se descarga el módulo "usbcore.o".

5. Ahora se procede a proporcionar soporte para dispositivos FIREWIRE. Lo único que se hace es cargar el módulo "ohci1394.o", que da soporte para un conjunto de chipsets muy utilizados.

6. El siguiente paso es dar soporte para HOTPLUG, en función de lo que se haya detectado hasta el momento. Hay una variable, HOTPLUG, que inicialmente está vacía. Esta variable pasará a contener "yes" siempre y cuando se detecte algún dispositivo USB, FIREWIRE o PCMCIA. En el caso de que esta variable contenga "yes", se activa el manejador "HOTPLUG" (/sbin/hotplug).

7. Detección del resto de dispositivos hardware. Para ello, se utiliza la herramienta "hwsetup". Este programa, escrito por Klaus Knopper, es básicamente una modificación de Kudzu, que utiliza las mismas librerías de detección de hardware, pero que funciona de manera levemente diferente.
   En este caso, la detección y configuración de hardware se produce de manera no interactiva. Además, se cargan los módulos necesarios y se generan los ficheros en /dev.
   El funcionamiento de hwsetup es muy similar al de Kudzu: hace una búsqueda del hardware instalado en el sistema y compara los resultados con una base de datos de dispositivos que se almacena en /cdrom/isolinux/cdroot/hwdata (en el calzador). A partir de la misma se cargan los módulos necesarios para dar soporte al hardware detectado.

8. El último paso consiste en detectar y montar todas las particiones existentes. Para ello, se hace una búsqueda en /proc/partitions. A partir del resultado y con la ayuda de fdisk se dispone de toda la información acerca de las particiones y del tipo de sistema de ficheros que hay en cada una de ellas. Entonces, se crean los puntos de montaje en /mnt, se genera el fichero /etc/fstab y se montan todas las particiones encontradas.

9. Se termina la ejecución de hw-detect.sh.

Proceso de instalación

Existe un instalador integrado en el calzador, que permite la instalación de la distribución base al disco duro. Este instalador incluye las siguientes características: es gráfico, basado en los programas "Xdialog" y "Zenity"; permite el manejo de particiones, mediante programas como "QTParted" y "cfdisk", seleccionables por el usuario; configura de manera adecuada el gestor de arranque (GRUB) para que, una vez producida la instalación en el disco duro y se reinicie el ordenador, se pueda elegir arrancar el/los sistema(s) operativo(s) que existía(n) anteriormente o se pueda arrancar el nuevo sistema operativo instalado.

Básicamente, el proceso que tiene lugar durante la instalación se resume en los siguientes pasos:

1. Se realiza el particionado necesario para crear una partición de espacio suficiente en el disco duro para albergar la distribución que se va a instalar. Esto se hace a partir de los programas de particionado "QTParted" y/o "cfdisk". El primero, además de ser gráfico, permite el redimensionamiento de particiones, por el que se puede cambiar el tamaño de una partición sin perder el contenido. Esto permite reducir el tamaño de alguna de las particiones existentes en el disco duro sin perder información. El segundo, todo un clásico en sistemas GNU/Linux, más limitado en funcionalidad, está orientado al usuario experto, puesto que su manejo es más complicado.

2. Posteriormente se procede a elegir la partición donde tendrá lugar la posterior instalación.

3. Después, se le pregunta al usuario algunos detalles sobre la distribución final, como la configuración de red, la contraseña de "root", etc.

4. Entonces, se procede a la copia de todos los ficheros necesarios. Mientras tiene lugar este proceso, se muestra una barra de progreso que indica el estado actual de la copia.

5. Hecho lo anterior, termina el proceso de instalación.

M. Script para la creación de Metadistro

```
#!/bin/bash

# Directorios que se van a usar:
# directorio donde esta descoprimida la distro ->unos 2 Gb
SOURCES=/mnt/md/sources
# directorio con el que se va a crear la iso final. Contiene el isolinux y META
MASTER=/mnt/md/master
# directorio en donde se va a guardar la iso final, el cd para tostar ->entre 600 y 650 Mb
ISODIR=/mnt/md/iso
# nombre de la metadistro
META=Anubix

# Se crea la imagen comprimida con Squashfs
OPTS=""
VER=`mksquashfs -version | grep "mksquashfs version" | cut -d " " -f 3`
if [ $VER = "2.1" ]; then
OPTS="-2.0"
fi
mksquashfs $SOURCES $MASTER/META/META.squashfs $OPTS

# Ahora crea el la imagen ISO final con lo que haya en el directorio $MASTER.
#Y crea el sector de arranque con el directorio "isolinux"
mkisofs -l -r -J -V "${META}" -hide-rr-moved -v -b isolinux/isolinux.bin -c isolinux/boot.cat
-no-emul-boot -boot-load-size
4 -boot-info-table -o $ISODIR/${META}.iso $MASTER

# Borra la imagen comprimida del sources
rm -fr $MASTER/META/META.squashfs

# Ahora se tuesta el CD
# Si es un regrabable (muy recomendable para hacer pruebas) se borra.
cdrecord -v dev=0,0 blank=fast

# Y ahora se tuesta. Se puede cambiar la velocidad de la grabación.
cdrecord -v speed=16 dev=0,0 $ISODIR/${META}.iso

#Si se quiere emular con qemu
#qemu -boot c -cdrom iso/Anubix.iso

# Se borra la imagen ISO final
rm -fr $ISODIR/${META}_cd.iso
```

N. Resultados de validaciones. Listas de comprobación.

N.1. Pruebas de interfaces y contenidos

Pruebas	Resultados
Verificación del contenido	Se ha verificado manualmente de forma exhaustiva
Sitio en construcción	No existen páginas en blanco o semi-acabadas
Validación HTML	No se ha producido ningún problema de validación
Plug-ins necesarios	No se han empleado plug-ins
Consistencia diagramación	Cada una de las secciones de la aplicación tiene elementos consistentes. Menús en el mismo lugar, uniformidad de colores y tipos de letra, etc...
Ancho de la diagramación	Al reducir la resolución a 800x600 algunas pantallas de la aplicación desbordan las dimensiones visuales de la pantalla y se requiere el uso del scroll.
Diagramación vs. S.O.	Se han realizado pruebas de navegación de la aplicación desde clientes en sistemas operativos Windows y Unix.

N.2. Pruebas funcionales y de operación

Pruebas	Resultados
Campos Obligatorios	Se validan todos los campos, si no se introduce un campo considerado obligatorio se muestra un error por pantalla para que el usuario introduzca el valor del campo.
Validaciones	Se realizan validaciones de los campos introducidos.
Ingreso de datos	Se comprueba la corrección de los datos y su adecuada sintaxis antes de introducirlos en los archivos de configuración de destino.
Reingreso y corrección de datos	Ante un error en la validación remota, se vuelve a la pantalla anterior y se muestra un error explicando al usuario el motivo de error. Se da la oportunidad al usuario de corregir el error.
Elementos de interfaz	Todos los botones tienen el estilo estándar.
Multiplataforma	Los problemas surgidos con diversos navegadores son practicamente inexistentes.
Botones de interacción	Los botones interactivos que abren pop-ups realizan validación de datos tanto en origen como en destino.

N.3. Pruebas de rapidez de acceso

Nº	Conceptos de Rapidez de Acceso	Respuesta
1	¿El usuario puede encontrar en no mas de 3 clicks la información que busca?	SI
2	¿Aparece el menú de navegación en un lugar destacado? ¿se ve fácilmente?	SI
3	¿El sitio cuenta con un mapa y/o buscador que dé acceso alternativo a los contenidos?	NO
4	¿El sitio mantiene una navegación consistente y coherente en todas sus páginas?	SI
5	¿El diseño usa jerarquías visuales para determinar lo importante de un solo vistazo?	SI
6	¿Los formularios ofrecen opciones que permiten al usuario evitar, cancelar o rehacer una acción?	SI
7	¿El tamaño de las letras de los textos es adecuado?	SI
8	¿El nombre de la URL está vinculado con el nombre o función del sistema? ¿Se ofrecen en la barra del navegador nombres descriptivos de la sección en que se encuentra el usuario?	SI

N.4. Pruebas de usabilidad

Nº	Conceptos de Usabilidad	Respuesta
1	¿Existen elementos de la imagen propia de la aplicación (e.j. logotipos)?	SI
2	¿El logotipo es visible desde cualquier parte de la aplicación web?	SI
3	¿El diseño del sitio es eficiente, rápido e intuitivo?	SI
4	¿Se verificó la consistencia de todos los enlaces?	SI
5	¿La aplicación mantiene una navegación coherente en todas las pantallas?	SI
6	¿Se informa al usuario claramente del área del sitio que está visitando?	SI
7	¿Usa elementos para indicar los campos obligatorios en los formularios?	SI
8	¿Después de que ocurra un error es fácil volver al punto donde se encontraba y se entregan recomendaciones de los pasos a seguir?	SI
9	¿Usa jerarquias visuales para distinguir importancia de elementos?	SI