Sistemas Informáticos

Curso 2004-05

FreeWall-1
Un cortafuegos corporativo para plataformas Win32

Fermín Serna Fernández de la Torre
Javier Reza Bernández
Adolfo Vázquez Rodríguez

Dirigido por:
Prof. Luis Javier García Villalba
Dpto. Sistemas Informáticos y Programación

Facultad de Informática
Universidad Complutense de Madrid

Los abajo firmantes: Fermín J. Serna Fdez. de la Torre, Javier Reza Bernández y Adolfo Vázquez Rodríguez, autorizan a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales, y, mencionando expresamente a sus autores, tanto la presente memoria, como el código, la documentación, y/o el prototipo desarrollado.

Fermín J. Serna Fdez. de la Torre **Javier Reza Bernández** **Adolfo Vázquez Rodríguez**

Índice de contenidos

Índice de figuras

Propuesta proyecto: FreeWall-1

FreeWall-1 es el proyecto para construir un cortafuegos corporativo para plataformas Win32 similar a los cortafuegos comerciales existentes tipo FireWall-1 de CheckPoint o StoneGate de StoneSoft. FreeWall-1 pretende ser una alternativa robusta, fiable y gratuita.

FreeWall-1 no pretende ser una copia de un producto comercial, sino que incorporará tecnologías únicas como un IPS (Intrusion Prevention System) en el propio filtrado del cortafuegos. Este IPS en un principio estará basado en reglas de nuestra propia creación o compatibilidad con algunas reglas del IDS SNORT.

En un principio dado los recursos humanos (3 personas) y de tiempo (9 meses) que se dispondrán no se incorporará soporte VPN (Virtual Private network), servidores seguros, ni balanceo de carga. Pero se dejará una interfaz para su fácil inclusión posterior.

FreeWall-1 se compone principalmente de tres grandes componentes:

- **Módulo Kernel Win32**: Este módulo, en espacio de Kernel, será el encargado de defragmentación TCP/IP, aplicar la política de filtrado/NAT y la política del IPS para permitir o denegar el paso del datagrama TCP/IP.
- **Management Server**: Será un servicio en NT encargado de comunicarse con el módulo a través de IOCTL y con los Management Clients a través de TCP/IP bajo una comunicación cifrada con AES (Advanced Encryption Standard) de 256 bits. Sus funciones, entre otras, serán las de cargar las politicas de filtrado e IPS al iniciar el servidor, gestión de logs, actualización de firmas IPS y gestión de peticiones de los clientes (ver logs, actualizar configuración,...).
- **Management Client**: Será un frontend para la configuración remota del cortafuegos. Será posible configurar, entre otras, la política de filtrado y NAT (Network Address Translation). Se comunicará con el Management Server a través del canal cifrado vía TCP/IP previa autentificación.

Partes opcionales:

- Para mejorar la rapidez del cortafuegos y el ancho de banda que pueda soportar se podrían incluir reglas compiladas y no una tabla de reglas. El management server compila a codigo objeto las reglas y lo manda por IOCTL al módulo de Kernel el cual lo trata como una función.
- Red neuronal de detección de comportamientos anómalos de usuarios en el IPS.
- Alta disponibilidad y balanceo de carga mediante multicast y un algoritmo para el balanceo de carga.
- Soporte plugins para que el usuario final pueda ser capaz de programar y usar sus propias verificaciones en el IPS.
- Portar el módulo Kernel y el management server a los UNIX más comunes en entornos de cortafuegos (Solaris SPARC, Linux, HPUX, etc...).

Project proposal: FreeWall-1

FreeWall-1 is the project to build a corporative firewall for Win32 platform similar to the commercial existing ones such as FireWall-1 of Checkpoint or StoneGate of StoneSoft. FreeWall-1 aims to be a robust alternative, reliable and free.

FreeWall-1 objectives are not to be a copy of a commercial product, it will have unique technologies such as an IPS (Intrusion Prevention System) in the filtering stage of the firewall. This IPS in a first stage will be rule based compatible with IDS SNORT rules.

Due to limited human (3 people) and time resources (9 months) we will face, FreeWall-1 will not support technologies such as VPN (Virtual Private Network), secure servers nor load balancing. But there will be a clean interface for easy future development.

FreeWall-1 has three main components:

- **Win32 kernel module**: This module, at Kernel Space, will be encharged of the TCPIP defragmentation, filter/NAT based on current loaded policy and filter (IPS) in order to drop or pass the current TCPIP datagram.
- **Management Server**: It will be a NT service encharged to link with the kernel space driver through IOCTL and with the clients GUIS through a secure and ciphered channel, using AES (Advanced Encryption Standard) of 256 bits over TCP/IP. The management server will have to deal with the load of policies, logs management, IPS rules updates and client protocol dispatcher.
- **Management Client**: It will be the front-end for the remote configuration of the firewall. It will be possible to configure, among other things, the filtering and NAT (Network Address Translation) policies. It will link with the management server through a ciphered channel once authenticated.

Optional components:

- In order to increase the performance and bandwidth supported it can be developed the compiled rule component. The management server will compile the policy into IA32 op-codes and send it though IOCTL to the kernel driver.
- Neural net for user abnormal behaviour detection in the IPS.
- High availability and load balancing through multicast and algorithms.
- Plugin support for advanced IPS rules.
- Kernel driver and management server port to most common UNIXes (Solaris SPARC, Linux, HPUX, etc...).

1. Arquitectura del sistema

Visión general

Arquitectura
Flujo de Información
Plataforma

Driver de Red

Arquitectura interna
IOCTL
Descripción componentes

Management Server

Arquitectura interna
Protocolo TCP/IP
Descripción componentes

Management Client

Arquitectura interna
Políticas
Descripción componentes

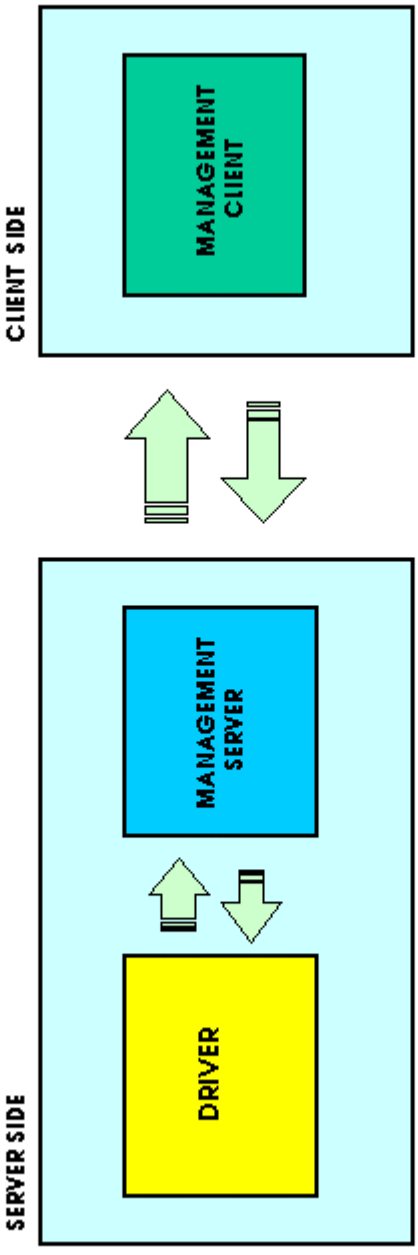

Figura 1. Arquitectura general de Freewall1

Visión general

Arquitectura.

Si bien a la hora de diseñar un firewall se ponen en juego muchas variables relacionadas con las necesidades concretas del sistema a proteger, existen una serie de diseños con sus topologías de red asociadas, que dan muestra de las posibles arquitecturas disponibles a la hora de desarrollar una aplicación de este tipo. Para ilustrar las posibilidades, hemos utilizado la clasificación proporcionada por [SEI99].

Según este organismo existen dos alternativas bien diferenciadas en el diseño de una aplicación de este tipo, en base al número de sistemas entre los que se divide las distintas capas del firewall. Los sistemas basados en una única capa, mantienen todos los servicios en un único sistema o host, que se encarga de gestionar todos los paquetes que llegan y salen de la red a proteger. Por otro lado, los sistemas multicapa, dividen entre varios hosts, algunos internos a la red a proteger y otros externos a la misma, los distintos servicios del firewall. Los siguientes ejemplos ilustran estas dos categorías.

Red perimetral.

La red perimetral se basa en un único host que se encarga de supervisar toda la entrada y salida de paquetes hacia y desde los hosts que forman parte de la red que protege. La principal desventaja suele ser el rendimiento en el análisis de paquetes, debido a que por lo general éste se hace a nivel usuario, con la consiguiente carga de trabajo, que puede repercutir en el rendimiento de la red.

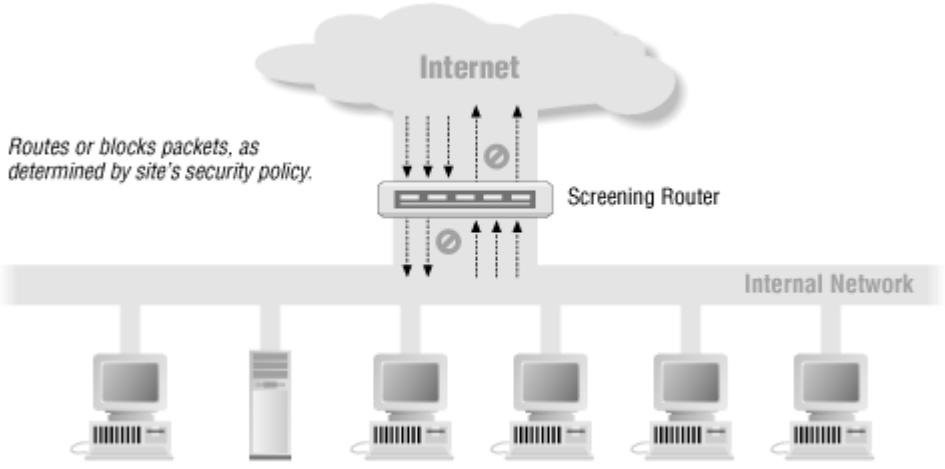

Figura 2. Topología típica de una red de tipo perimetral.[ORE95]

Red basada en Bastion Host.

Una de las arquitecturas multicapa más habituales es la que introduce máquinas adicionales, internas a la red protegida, que sirven para gestionar el trabajo solicitado por los hosts de la red, reduciendo así en la carga de trabajo al host principal encargado de esta tarea. Estas máquinas adicionales se conocen como Bastion Hosts, pudiéndose configurar según las necesidades concretas de cada red. La principal desventaja de esta configuración es que al estar el firewall repartido entre varios sistemas, la seguridad de la red se puede ver comprometida por una deficiente administración de estos, sobre todo en configuraciones donde entran en juego un gran número de Bastion Hosts.

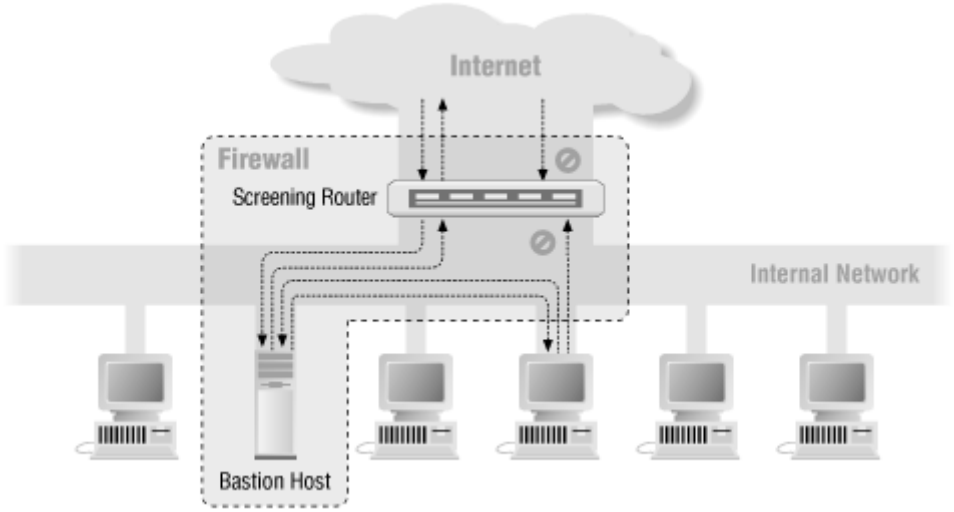

Figura 3. Topología típica de una red basada en un Bastión Host. [ORE95]

Los argumentos a favor de uno u otro diseño, recalan sobre todo en el coste de la solución, especialmente en el aspecto referente a su mantenimiento, y en la integridad y rendimiento de los sistemas que dan soporte a la solución firewall. FreeWall1 integra lo mejor de cada solución, proveyendo de un único punto de acceso a toda la red, con la repercusión en los costes de la misma, a la vez que incluye determinadas técnicas, como el filtrado a nivel kernel o su capacidad de stateful firewall, que minimiza al máximo la caída del rendimiento del sistema frente a escenarios de gran tráfico de datos a lo largo de un único host.

FreeWall1 sigue un patrón cliente/servidor, pudiendo gestionar múltiples clientes a la vez. Para ello incluye dos soluciones específicas de los sistemas para los que ha sido portado, en concreto hilos para plataformas Win32 y creación de procesos hijos mediante la llamada fork( ) en entornos Linux. Siguiendo este patrón nuestro Integrated Defense System (IDS)

se divide en dos sistemas, uno a nivel cliente, cuyo principal subsistema es el management client y otro a nivel servidor, que cuenta con dos subsistemas, comunicados entre si, el management server y el driver de red. Su principal labor es la siguiente.

Driver.
Este módulo, en espacio de Kernel, será el encargado de defragmentación TCP/IP, aplicar la política de filtrado y la política del IPS para permitir o denegar el paso del datagrama TCP/IP.

Management Server.
Es un servicio en NT y daemon en Linux, encargado de comunicarse con el driver a través de IOCTL y con los Management Clients a traves de TCP/IP bajo una comunicación cifrada con AES (Advanced Encryption Standard) de 256 bits. Sus funciones, entre otras, son las de cargar las políticas de filtrado e IPS al iniciar el servidor, gestión de logs, actualización de firmas IPS y gestión de peticiones de los clientes, como visualizar los logs o actualizar la configuración.

Management Client.
Es un frontend para la configuración remota del cortafuegos. Gracias al mismo es posible configurar, entre otras, la política de filtrado. Se comunicará con el management server a través del canal cifrado vía TCP/IP previa autentificación.

Integrated Defense Systems
Las soluciones aportadas por el equipo en la creación del firewall, configuran éste como una solución Integrated Defense System, en sus siglas IDS. Existen varias soluciones de este tipo en base a su área de acción.

Host-based IDS
Se encargan de monitorizar el trafico de información dentro de uno o varios sistemas. Así se encargan de analizar los datos que el sistema operativo y las aplicaciones que corren sobre éste generan.

Network-based IDS.
Están orientados al análisis de los paquetes que circulan por la red en la que está situado. Para ello analiza el contenido y estructura de los paquetes que circulan por la misma. Es en este marco donde mejor podemos encuadrar el objetivo de nuestro proyecto, FreeWall1.

Vulnerability-assessment IDS.
Monitorizan sistemas y redes en busca de vulnerabilidades que puedan afectar a la integridad y seguridad de los mismos frente a ataques.

Flujo de la información

En la siguiente figura analizamos el flujo que un paquete sigue a través del IDS desde que llega a través del driver MAC del sistema embebido que alberga FreeWall1, hasta que éste alcanza su host destino dentro de la red que éste protege.

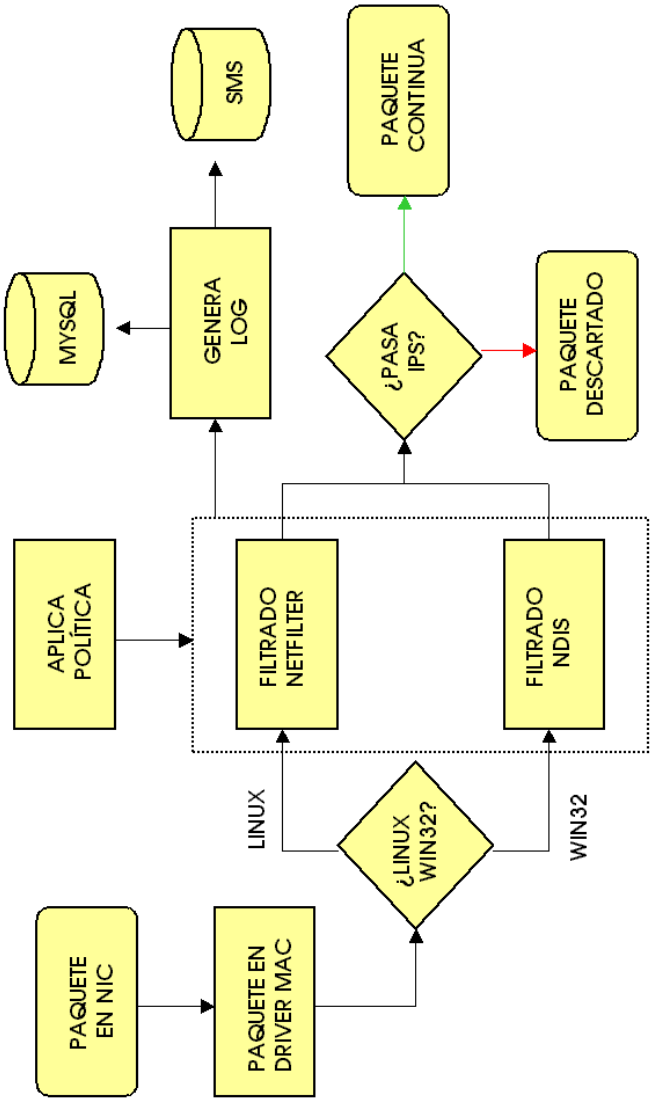

Figura 4. Flujo de información en FreeWall1

Plataforma

Con el fin de ofrecer una solución de enchufar-y-listo, hemos optado por integrar FreeWall1 en una plataforma hardware embebida, la Soekris Engineering net4801. Aunque FreeWall1 puede ser instalado en plataformas Win32, hemos optado por instalarlo sobre una distribución Linux, en concreto Pebble Linux, basada en Debian GNU/Linux.

Soekris Engineering net4801.
Como características más interesantes este sistema empotrado dispone de un lector de tarjetas tipo CompactFlash, las cuales hemos utilizado para instalar el sistema operativo, en este caso Pebble Linux, sobre el que corre FreeWall1. A su vez, la net4801 dispone de tres puertos Ethernet 10/100, los cuales se pueden utilizar indistintamente como puertos de entrada o como puertos de salida. Estos puertos irían conectados por un lado a la red o redes a proteger, sobre las que actuaría el firewall, y por otro a las redes públicas con las que quiere interaccionar el sistema.

Figura 5. Vista general del Soekris Engineering net4801

Pebble Linux.
Pebble Linux es una distribución Linux cuya configuración mínima ocupa 8 MB y en su configuración más extensa ocupa 64 MB. Está basada en Debian GNU/Linux. Para su instalación en el net4801, hemos utilizado una Sandisk CompactFlash de 128 MB. Una vez instalada esta en el habitáculo correspondiente del net4801, y habiendo arrancado Pebble Linux en dicho sistema, creamos un ramfs en el net4801 donde incluimos los binarios necesarios para ejecutar FreeWall1. De esta forma aumentamos la velocidad del sistema, ya que la velocidad de transmisión de la CompactFlash es considerablemente menor, y esto repercutiría en el rendimiento del filtrado, clave en el rendimiento global de FreeWall1.

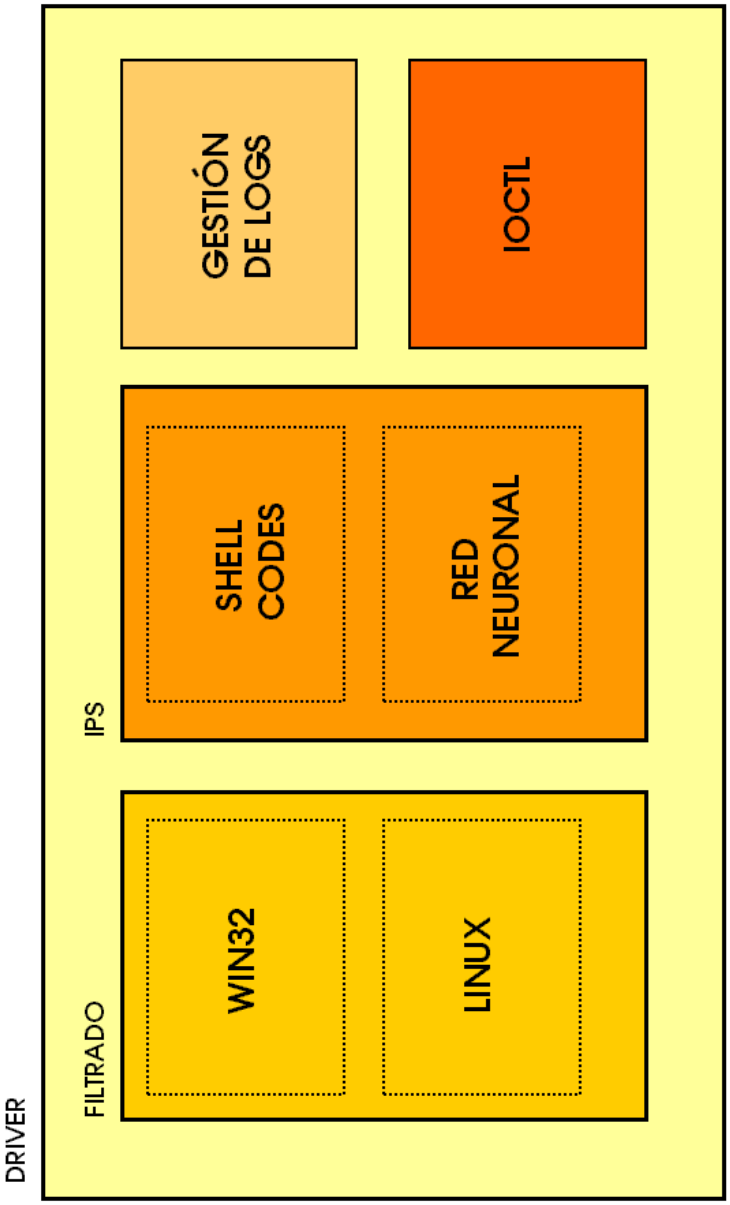

Figura 6. Arquitectura Interna del Driver

Driver

El driver es el encargado de llevar a cabo el filtrado efectivo de los paquetes que circulan a través de FreeWall1, ya sea hacia o desde la red que éste protege. El driver como subsistema de la parte servidor de FreeWall1 se comunica con el otro subsistema, el management server, a través de IOCTL. El filtrado en el driver se realiza a partir de las políticas indicadas por el management server. Este subsistema consta de cuatro partes principales.

Filtrado.
Es la primera capa de actuación de FreeWall1. Debido a las diferencias en la arquitectura de red de cada sistema operativo, el sistema de filtrado cuenta con funciones diferentes para las plataformas Win32 y Linux. En este nivel FreeWall1 recibe el paquete de la red, bien directamente del driver MAC en el caso de Win32, bien a través del framework netfilter en el caso de Linux. Ese paquete, en base a la política recibida, será eliminado o enviado a través del módulo IOCTL al management server. El filtrado en ambas plataformas se lleva a cabo a nivel kernel, para optimizar el rendimiento del sistema, que se vería mermado si por cada paquete que llegara tuviera que elevarlo hasta la capa de usuario y allí decidir su destino.

Instrusion Prevention System.
Se considera IPS a todos aquellos sistemas que ayudan a evitar ataques o actividades sospechosas de conducir a ellos. En FreeWall1 hemos introducido dos de estos sistemas. Por el lado un sistema de eliminación de ShellCodes, típico de la mayoría de los firewalls, pero con la posibilidad de detección de ShellCodes polimórficos, que veremos más en detalle en el próximo capítulo. Por otro lado disponemos de un sistema IPS basado en una red neuronal tipo Elman, que evita ataques que utilizan el protocolo HTTP.

Gestión de logs.
La gestión de logs a nivel driver se hace utilizando una lista de punto de interés, en la cual se van enlazando todos aquellos logs que se van generando. Sólo se generan logs por aquellos paquetes en los que el campo log de la regla correspondiente en la política indicada por el management server, tenga valor LOG. Este sistema de gestión de logs a su vez alimenta a través de IOCTL al SDK del management server. La conexión para intercambiar el log creado se lleva a cabo entre ambos cuando éste es creado, momento es el que se produce el evento FW1_EV_LOG, que activa la comunicación por medio de IOCTL.

IOCTL

El método seleccionado para la comunicación entre el driver y el demonio (espacio KERNEL y espacio USUARIO respectivamente) fue a través de IOCTLs. IOCTL (I/O

control) es un mecanismo por el cual dado un descriptor de fichero (en UNIX abriendo el fichero /dev/FreeWall-1) podemos mandar peticiones al driver con unos argumentos en forma de buffer de entrada. El driver procesara dichas peticiones y nos responderá con un código de control y resultados en forma de un buffer de salida.

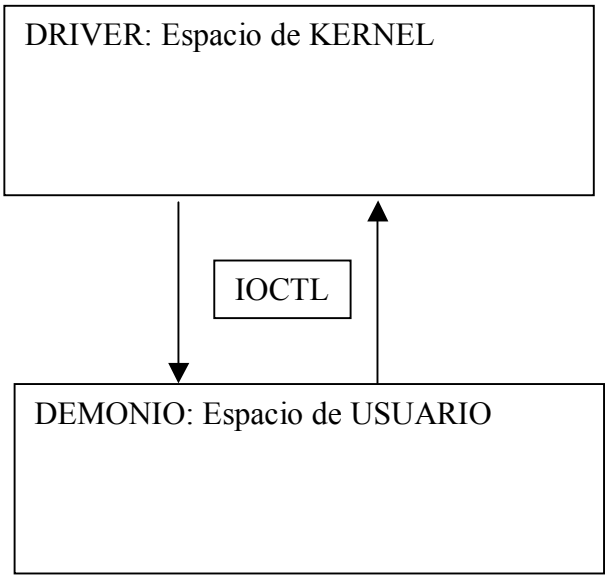

Figura 7. Arquitectura de comunicación basada en IOCTL

A continuación reflejamos las distintas peticiones IOCTL, con sus parámetros de entrada y salida, implementadas.

- **IOCTL_FW1_ENABLE:**

  Función: Habilita el cortafuegos en memoria.
  Parámetros de entrada: Ninguno.
  Parámetros de salida: Ninguno.

- **IOCTL_FW1_DISABLE:**

  Función: Deshabilita el cortafuegos en memoria.
  Parámetros de entrada: Ninguno.
  Parámetros de salida: Ninguno.

- **IOCTL_FW1_GET_LOG:**

  Función: Solicita el siguiente registro log del driver, que una vez entregado lo borra de la lista de registros.
  Parámetros de entrada: Ninguno.
  Parámetros de salida: Buffer de salida con el registro log en cuestión.

- **IOCTL_FW1_CLEAR_FILTER_RULES:**

  Función: Solicita que se borren todas las reglas de filtrado configuradas anteriormente .
  Parámetros de entrada: Ninguno.
  Parámetros de salida: Ninguno.

- **IOCTL_FW1_CLEAR_NAT_RULES:**

  Función: Solicita que se borren todas las reglas de NAT configuradas anteriormente .
  Parámetros de entrada: Ninguno.
  Parámetros de salida: Ninguno.

- **IOCTL_FW1_CLEAR_IPS_RULES:**

  Función: Solicita que se borren todas las reglas de IPS configuradas anteriormente.
  Parámetros de entrada: Ninguno.
  Parámetros de salida: Ninguno.

- **IOCTL_FW1_ADD_FILTER_RULE:**

  Función: Solicita que se añada una regla a la lista de reglas de filtrado.
  Parámetros de entrada: Buffer con la estructura de la regla en cuestión.
  Parámetros de salida: Ninguno.

- **IOCTL_FW1_ADD_NAT_RULE:**

  Función: Solicita que se añada una regla a la lista de reglas de NAT.
  Parámetros de entrada: Buffer con la estructura de la regla en cuestión.
  Parámetros de salida: Ninguno.

- **IOCTL_FW1_ADD_IPS_RULE:**

  Función: Solicita que se añada una regla a la lista de reglas de IPS.
  Parámetros de entrada: Buffer con la estructura de la regla en cuestión.
  Parámetros de salida: Ninguno.

Descripción componentes

driver/FW1_filter.c

Este módulo se encarga de aplicar las distintas reglas de filtrado configuradas a un paquete en busca de una concordancia. En el caso de encontrarla se devolvería la decisión de dicha regla, bien sea aceptar el paquete o rechazarlo. Del mismo modo, y bajo concordancia, se generará un registro log con todos los datos de la conexión en el caso de que la regla generadora de la concordancia lo tenga especificado.

driver/FW1_hashtable.c

En este módulo está la implementación del tipo abstracto de dato de una tabla hash abierta (tabla hash cuyos nodos son listas por posibles concordancias de varios nodos con una misma clave). Esta tabla hash se usará para el control de conexiones y el stateful firewalling. Se eligió una tabla hash debido a su gran rendimiento en accesos directos a nodos y, para así, seguir teniendo un compromiso con el rendimiento del sistema.

driver/FW1_list.c

Este módulo contiene una implementación de una lista de memoria dinámica con punto de interés. El punto de interés nos servirá para una mayor rapidez en el acceso directo a nodos de la lista. No se optó por implementación mediante arrays debido a la limitación de número de nodos posibles. Debido a que no siempre buscaremos el nodo inicial o el final, el punto de interés nos ayuda a tener un compromiso con el rendimiento óptimo.

driver/FW1_log.c

El presente módulo se encarga de gestionar la estrutura de logs (lista con punto de interés) cuando el driver necesita ingresar un registro log ante una regla que lo indique, o eliminar un registro (ante una petición IOCTL de bajada de logs a espacio de usuario).

driver/FW1_net_utils.c

El presente módulo implementa funciones de libc no disponibles en espacio de kernel. Las funciones implementadas sirven para convertir números a formato de red dado nuestro formato de host, convertir los flags de un paquete TCP a un registro log entendible por una persona, entre otras funciones

driver/linux/FreeWall-1.c

Este es el módulo raíz del driver de LINUX, implementa las funciones necesarias y requeridas por el cargador de módulos. Existen dos funciones:

FW1_init será llamada cuando el cargador de módulos quiera cargar el presente módulo dándole capacidad para efectuar las operaciones necesarias como p.e. registrarse a los eventos de NETFILTER o de IOCTL del fichero /dev/FreeWall-1.

FW1_exit será llamado cuando el cargador de módulos quiera eliminar dicho módulo del sistema, dándole capacidad para liberar memoria o registros a eventos asociados.

driver/linux/FW1_dispatchers.c

El presente módulo se encarga de gestionar la petición de IOCTLs en UNIX.

Una vez estamos registrados a los eventos open(), close() e ioctl() del fichero /dev/FreeWall-1, en este módulo se presentan la implementación de las funciones que gestionan dichos eventos.

La función importante es el gestor de IOCTL ya que es el punto de conexión con espacio de usuario.

driver/linux/FW1_hooks.c

El siguiente módulo implementa la función que se registró previamente en NETFILTER. Cada vez que haya un paquete con dirección IN, OUT o FORWARD será llamada esta función.

Esta función gestionará en conjunción con otros módulos el filtrado, ips y nat del paquete de red.

driver/win32/FreeWall-1.c

Este es el módulo raiz del driver de WINDOWS, implementa las funciones necesarias y requeridas por el cargador de módulos. Existen dos funciones:

DriverEntry será llamada cuando el cargador de módulos quiera cargar el presente módulo dándole capacidad para efectuar las operaciones necesarias como p.e. registrarse a los eventos de NDIS o de IOCTL.

FW1_unload será llamado cuando el cargador de módulos quiera eliminar dicho módulo del sistema, dándole capacidad para liberar memoria o registros a eventos asociados.

driver/win32/FW1_dispatchers.c

El presente módulo se encarga de gestionar la petición de IOCTLs en WINDOWS.

Una vez estamos registrados a los eventos open(), close() e ioctl(), en este módulo se presentan la implementación de las funciones que gestionan dichos eventos.

La función importante es el gestor de IOCTL ya que es el punto de conexión con espacio de usuario.

driver/win32/FW1_hook.c

El presente módulo es el encargado de modificar la tabla de exportaciones del binario NDIS.SYS en memoria KERNEL. Usando técnicas de virus informáticos redirigimos las funciones de NDIS.SYS a nuestro propio driver FREEWALL-1.SYS.

Una vez tratadas las peticiones, redirigimos el flujo a la función original.

driver/win32/FW1_ndis_funcs.c

El presente módulo implementa las distintas funciones para trabajar con paquetes de red NDIS.

driver/win32/FW1_ndis_hooks.c

En el siguiente módulo se encuentra la implementación de las funciones que hace de puente entre NDIS.SYS y FREEWALL-1.SYS.

La tabla de exportaciones del primero tendrá punteros a estas funciones para que cuando se llame a una función NDIS, realmente se está trabajando con nuestras funciones.

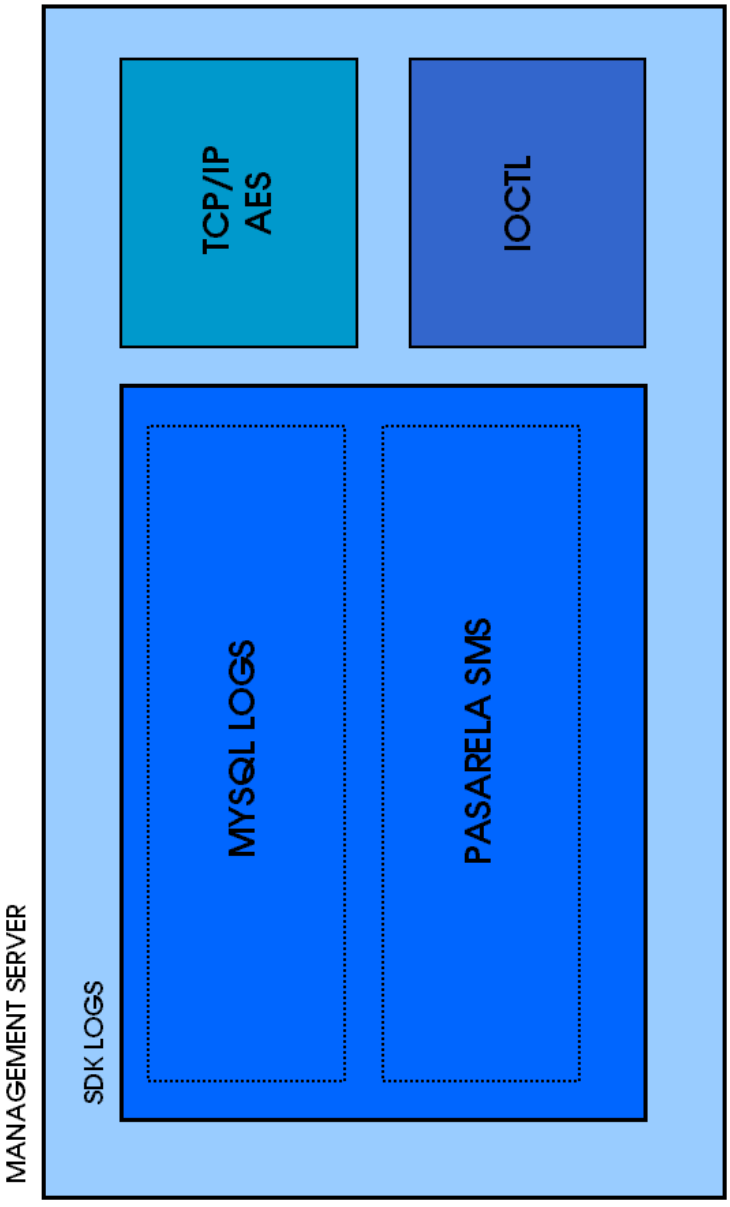

Figura 8. Arquitectura Interna del Management Server

Management Server

El management server es el encargado de gestionar las políticas indicadas por el administrador de FreeWall1 a través del management client, que veremos a continuación, y aplicarlas convenientemente a través del driver a los paquetes que pasan a través del IDS. Este subsistema se encarga de la gestión general de la información que FreeWall1 va generando, ofreciéndola posteriormente a través de uno de los muchos módulos de difusión de la información con los que cuenta. Los principales módulos de este subsistema son.

Software Development Kit para logs.
Como ya se ha comentado anteriormente al abordar el diseño interno del driver, éste envía un mensaje a través de IOCTL al management server para que pueda hacerse cargo de los logs que van generándose. Para gestionar estos logs, y ofrecer al desarrollador un framework para poder tratar esos datos según sus necesidades, hemos desarrollado un SDK específico. Para ilustrar el uso de dicho SDK, hemos implementado dos ejemplos de uso del mismo, un sistema de notificación vía SMS y un sistema de almacenamiento de logs en MYSQL. El primero utiliza la pasarela SMS de PeopleCall para enviar un mensaje al móvil indicado cada vez que el management server recibe un log del driver. Por otro lado el sistema de almacenamiento de logs basado en MYSQL, ofrece una herramienta de consulta de cara a realizar posibles análisis forenses del sistema.

Protocolo TCP/IP

El método seleccionado para la comunicación entre el demonio y los diferentes clientes GUI fue a través de SOCKETS. Para una mayor confidencialidad de la conexión, se decidió cifrarla mediante AES con una clave de 256 bits. Debido a la complejidad de las peticiones, también se decidió estandarizar dicha comunicación mediante un protocolo bien especificado. El protocolo especificado usará la siguiente estructura que se transmitirá en buffers de 1024 bytes para evitar ataques de padding:

```
struct mens
{
  struct cab
  {
    uint32_t type;
    uint32_t version;
  } header;
  union
  {
    struct ping_st
    {
      uint32_t ping_sequence;
      uint32_t timestamp;
```

```
    } ping;
    struct error_st
    {
      uint32_t num_error;
    } error;
    struct policy_st
    {
      uint32_t policy_number;
      uint32_t rc;
      uint32_t data_len;
      char data[FW1_PORTBUFFER - 124];
    } policy;
    struct status_st
    {
      uint32_t firewall_enabled;
      uint32_t ips_enabled;
      uint32_t current_policy;
      uint32_t rc;
    } status;
    struct policy_enum_st
    {
      uint32_t num;
      uint32_t policies[FW1_MAX_POLICIES];
    } policy_enum;
    struct logs_st
    {
      uint32_t data_len;
      char data[FW1_PORTBUFFER - 124];
    } logs;
  } data;
#ifdef WIN32
};
#else
} __attribute__ ((packed));
#endif
```

El protocolo consta de una cabecera común para todas las peticiones y un cuerpo del mensaje diferente para cada petición. Se introduce un MAGIC_VALUE para comprobar la integridad del paquete descifrado en el campo req→cab→version que debe ser FW1_VERSION definido en cada versión del cortafuegos para así evitar el uso de clientes antiguos con versiones más modernas del protocolo.

A continuación recogemos un resumen de las distintas operaciones (req→cab→type) del protocolo especificado.

Comunicación cliente – servidor.
Para esta comunicación se definieron los siguientes tipos de mensajes que implementan funciones como la petición de logs, borrar políticas, cargar políticas, obtener la lista de política o almacenar dicha política.

```
#define FW1_CTS_PONG           0
#define FW1_CTS_ERROR          1
#define FW1_CTS_GET_POLICY     2
#define FW1_CTS_PUT_POLICY     3
#define FW1_CTS_PUT_POLICY_END 4
#define FW1_CTS_GET_STATUS     5
#define FW1_CTS_PUT_STATUS     6
#define FW1_CTS_ENUM_POLICIES  7
#define FW1_CTS_DEL_POLICY     8
#define FW1_CTS_LOAD_POLICY    9
#define FW1_CTS_GET_LOGS       10
```

Comunicación servidor – cliente.
Para esta comunicación se definieron los siguientes tipos de mensajes que implementan los códigos de operación de la petición de logs, borrar políticas, cargar políticas, obtener la lista de políticas, guardar política, entre otros e implementan el envio de políticas.

```
#define FW1_STC_PING           0
#define FW1_STC_ERROR          1
#define FW1_STC_GET_POLICY     2
#define FW1_STC_GET_POLICY_END 3
#define FW1_STC_PUT_POLICY     4
#define FW1_STC_GET_STATUS     5
#define FW1_STC_PUT_STATUS     6
#define FW1_STC_ENUM_POLICIES  7
#define FW1_STC_DEL_POLICY     8
#define FW1_STC_LOAD_POLICY    9
#define FW1_STC_GET_LOGS       10
```

Mensajes de Error.
Durante la comunicación pueden existir distintos tipos de errores (error de autentificación, timeout, error de protocolo o versión, entre otros) que serán indicados mediante los siguientes códigos de error:

```
#define FW1_ERROR_AUTHERROR    0
#define FW1_ERROR_TIMEOUT      1
#define FW1_ERROR_PROTO        2
#define FW1_ERROR_VERSION      3
```

Descripción componentes

daemon/conf_parser.c

Este módulo se encarga de gestionar la configuración del proceso servicio (o demonio). En UNIX los ficheros de configuración estarán en /etc/FreeWall-1 y en Windows en el directorio de instalación.

Tenemos dos funciones importantes:

Una primera función se encarga de volcar al fichero de configuración del servidor la estructura en memoria. P.e. un cliente ha podido solicitar cargar una nueva politica y ha de ser reflejado en el fichero para sucesivos reinicios del sistema.

La segunda función se encarga de leer y rellenar la estructura de configuración del servidor. Entre otras cosas lee información de las políticas disponibles y de la cargada actualmente.

daemon/crypt_stuff.c

Este módulo se encarga de gestionar la parte de cifrado AES que será utilizado para enviar datos a través de un socket dado.

Tiene varias funciones importantes:

Función para inicializar el sistema de cifrado AES 256 bits con una clave que lee de fichero.

Función que lee datos de un socket y los descifra usando la clave y el sistema de cifrado previamente inicializado.

Función que cifra datos y los envía por un socket usando la clave y el sistema de cifrado previamente inicializado.

daemon/dispatch_port.c

Este módulo se encarga de gestionar el socket que estará escuchando en el puerto TCP. Una vez el cliente GUI conecte a este puerto, se creará un proceso hijo mediante fork() en UNIX o CreateThread en Windows para su gestión y el no bloqueo para futuros clientes.

Mediante está técnica de creación de nuevos procesos/threads soportamos múltiples conexiones simultáneas e independientes entre sí identificadas por descriptores de socket.

daemon/FW1_list.c

Este módulo contiene una implementación de una lista de memoria dinámica con punto de interés. El punto de interés nos servirá para una mayor rapidez en el acceso directo a nodos de la lista. No se optó por implementación mediante arrays debido a la limitación en el número de nodos posibles.

Debido a que no siempre buscaremos el nodo inicial o el final, el punto de interés nos ayuda a tener un compromiso con el rendimiento óptimo.

daemon/policy_parser.c

Este módulo contiene todas las funciones necesarias para leer de fichero una politíca y rellenar las estructuras correspondientes de reglas nat, filtrado, objetos y servicios.

Una de sus características principales es la separación de reglas múltiples (varios objetos fuente p.e.) en varias reglas simples. Esto es necesario debido a que el protocolo de subida de reglas al driver por medio de IOCTL actúa sobre reglas simples, y el filtrado funciona también en base a reglas simples.

daemon/protocol.c

Este módulo contiene la implementación del protocolo de comunicación entre el servidor y el cliente. Dicho protocolo implementa las suficientes funciones para un correcto y potente servicio.

Un cliente, a través de este protocolo, puede efectuar las siguientes operaciones:

Pedir lista de políticas
(Des)Activar cortafuegos
(Des)Activar Nat
(Des)Activar IPS
Cargar política
Borrar política
Pedir logs existentes

daemon/rijndael.c

Este módulo implementa el cifrado AES que será utilizado para cifrar y descifrar datos provenientes de conexiones. Como datos característicos de este módulo se puede indicar que usamos AES en modo CBC y con claves de 256 bits.

daemon/shared_utils.c

Módulo auxiliar con funciones útiles necesarias que comparten diferentes módulos de la aplicación.

Estas funciones compartidas gestionan logs, cargan una política dada o nos dice si existe una política o no en el servidor.

daemon/unix/freewall1_service.c

Módulo principal del servidor en UNIX donde reside la función main y todas las variables globales utilizadas.

La función main, inicializará el cifrado y cargará su clave, abrirá el flujo de comunicación con el driver a través de IOCTL e iniciará los trámites para escuchar en un cierto puerto TCP para las conexiones de los clientes.

Seguidamente se convierte en demonio para así liberar el terminal y poder trabajar en *background*.

daemon/unix/signals.c

Este módulo es el encargado de gestionar las distintas señales en UNIX. Tiene funciones para registrar una función a una señal (en nuestro caso SIGTERM y SIGCHILD).

Cuando un cliente termina la comunicación con el servidor, al morir el hijo se genera la señal SIGCHILD y el proceso padre es notificado.

Cuando el administrador del sistema quiere reiniciar el sistema, se genera una señal SIGTERM para que se liberen los recursos del sistema y finalice el programa de una manera limpia.

daemon/unix/utils.c

Módulo auxiliar específico de UNIX con funciones útiles necesarias que comparten diferentes módulos de la aplicación.

Estas funciones compartidas gestionan p.e. convertir la aplicación en un proceso demonio residente.

daemon/win32/freewall1_service.c

Módulo principal del servidor en Windows donde reside la función main y todas las variables globales utilizadas.

La función main, inicializará el cifrado y cargará su clave, abrirá el flujo de comunicación con el driver a través de IOCTL e iniciará los trámites para escuchar en un cierto puerto TCP para las conexiones de los clientes.

Seguidamente se convierte en servicio para así liberar el terminal y poder trabajar en *background*.

daemon/win32/service_stuff.c

Este módulo se encarga de interactuar con los eventos generados desde el panel de servicios de Windows. P.e. si un usuario requiere parar el servicio, éste es el módulo notificado y dónde se efectúan las operaciones necesarias en base a la petición.

daemon/win32/utils.c

Módulo auxiliar específico de WINDOWS con funciones útiles necesarias que comparten diferentes módulos de la aplicación.

Estas funciones compartidas gestionan p.e. averiguar si el sistema Windows es al menos mayor o igual que un Windows 2000.

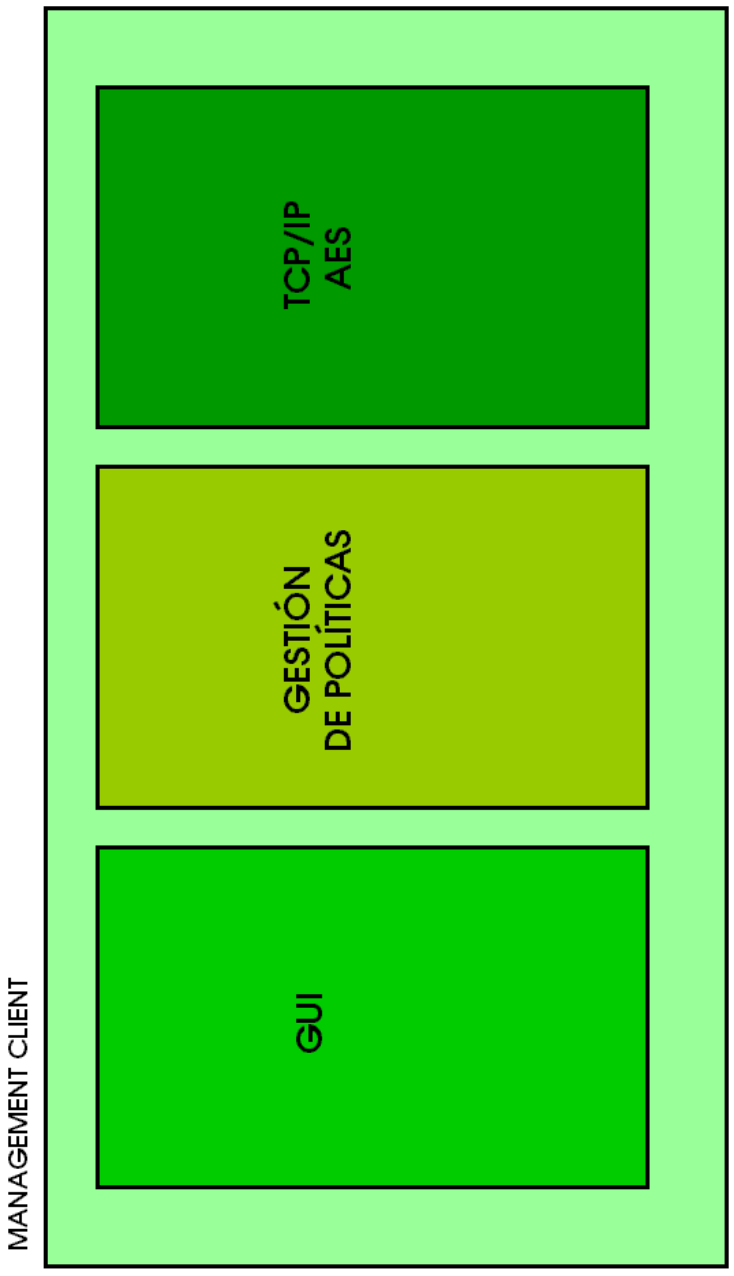

Figura 9. Arquitectura Interna del Management Client

Management Client

El management client es la interfaz de FreeWall1 con el administrador de la red. A través de este subsistema el administrador, previa autentificación cuya integridad viene protegida por una protección basada en clave MD5 y encriptación AES, puede gestionar todas las políticas que se encuentran instaladas en FreeWall1. De esta forma el administrador, puede modificar, crear y eliminar las políticas instaladas en la parte del servidor y gestionadas por el management server. Dispone de los siguientes subsistemas.

Graphical User Interface.
Desarrollada pensando en una gestión rápida de los elementos de la política, este módulo permite gestionar cada elemento de la política sólo con hacer clic y soltar. Ofrece una interfaz basada en árboles que facilita la búsqueda de información dentro del conjunto de reglas que forman la política, clasificándolas en base a características, como el protocolo implicado.

Política

El conjunto de las políticas se encuentran residentes en la parte servidor de FreeWall1, de esta forma el administrador podrá gestionarlas desde cualquier host, ya sea interno o no a la red que se está protegiendo. Una política de FreeWall1 constará de las siguientes partes.

- Conjunto de reglas de filtrado
- Conjunto de reglas NAT
- Editor de objetos y servicios.

Comenzará por una cabecera de comienzo de política y terminará por otra cabecera que indicará el fin de la política.

<politica id=1>\n – Comienzo de una política
</política>\n – Fin de una política.

El id de la política nos servirá para tener diferentes creadas y poder cargar una u otra.

Entre ambas cabeceras se incluirán un conjunto de reglas de filtrado, un conjunto de reglas NAT y un conjunto de servicios y objetos de red que estarán delimitados a su vez por cabeceras propias.

El formato de una regla de filtrado será el siguiente:

<filter_rule> id=1 obj_src=0 obj_dst=13 srv=255 log=nolog action=allow <filter_rule>\n

Como vemos estará delimitado por dos cabeceras de comienzo y fin de regla que nos permitirán distinguir que se trata de una regla de filtrado.

Tendrá los campos siguientes:

- **<filter_rule>**: Indica el inicio de una regla de filtrado
- **</filter_rule>**: Indica el fin de una regla de filtrado
- **id**: Identifica a la regla dentro de la política
- **obj_src**: Objeto fuente del que proviene el paquete
- **obj_dst**: Objeto al que va dirigido el paquete
- **srv**: Servicio al que pertenece el paquete/conexión
- **log**: Indicará si se debe crear un registro de log para el paquete (LOG o NOLOG)
- **action**: Indica la acción a realizar con el paquete (ALLOW, DROP o DENY)

Es importante observar que todos los campos están separados por un espacio. (Se debe respetar este formato para el que el parser funcione).

Valores reservados:

- FW1_MATCH_ALL: 0

El formato de una regla de NAT será el siguiente:

<nat_rule> id=1 obj_src=23 objd_st=14 srv=45 log=nolog accion=allow obj_src_new=56 objt_dst_new=34 srv_new=13 </nat_rule>\n

Como vemos estará delimitado por dos cabeceras de comienzo y fin de regla que nos permitirán distinguir que se trata de una regla de filtrado.

Tendrá los campos siguientes:

- **<nat_rule>:** Indica el inicio de la regla NAT
- **</nat_rule>:** Marca el fin de una regla NAT
- **id**: Identifica a la regla dentro de la política
- **obj_src**: Objeto del que proviene el paquete
- **obj_dst**: Objeto al que va dirigido el paquete
- **srv**: Servicio al que pertenece el paquete
- **log**: Si el paquete debe producir logs
- **action**: Acción a realizar con el paquete
- **obj_src_new**: Dirección origen del paquete transformada
- **obj_dst_new**: Dirección destino del paquete transformada
- **srv_new**: Servicio transformado

Valores reservados:

- FW1_MATCH_ALL: 0

Una vez definidas las reglas NAT se definirán lo objetos y servicios que las componen. En primer lugar se definirán los objetos de red que al igual que las reglas estarán delimitados por un comienzo y fin de objeto:

<obj> id=13 type=ip data=123.123.45.24</obj>\n
<obj> id=14 type=range_ip data=123.45.60.12-123.45.60.45</obj>\n
<obj> id=15 type=net data=123.45.60.0/32</obj>\n

Aquí puede observarse que la dirección del objeto puede ser una ip, una red, un conjunto de ips o un nombre.

- **<obj>:** Indica el comienzo de un servicio
- **</obj>:** Indica el fin del servicio
- **id:** Identificará al objeto dentro de la política
- **type:** Indicará el tipo de objeto del cual se trata(ip,net …)
- **data:** Indicará la dirección el objeto si es un ip o una red

Y por ultimo se especificarán los servicios a los cuales pertenecen los paquetes:

<srv> id=12 proto=udp port=23</srv>\n
<srv> id=17 proto=icmp type=8 code=0</srv>\n
<srv> id=18 proto=tcp port=80</srv>\n
<srv> id=18 proto=34</srv>\n

Al igual que sus predecesores estará delimitado por una cabecera de inicio y otra de fin de servicio y constará de:

- **<srv>:** Indica el comienzo de un servicio
- **</srv>:** Indica el fin del servicio
- **id:** Identifica al servicio dentro de la política
- **proto:** Indica el tipo de servicio.
- **type:** tipo de código icmp, en caso de ser un srv icmp
- **code:** code de código icmp, en caso de ser un srv icmp
- **port:** puerto en caso de ser un servicio tcp o udp

Valores reservados:

- FW1_MATCH_ALL: 0

Descripción componentes

gui/Autentificacion.cpp

Implementa todo el sistema de gestión necesario para que al arrancar, el administrador de FreeWall1 pueda autentificarse sea cual sea su ubicación. Es en este punto donde se crea el protocolo que irá asociado a la acción del administrador durante toda su sesión en el management client.

gui/Conexion.cpp

Incluye las funciones de más bajo nivel para establecer la comunicación a través de TCP/IP. Mediante el uso de la API Winsock2, se implementan todas las estructuras pertenecientes a esta API necesarias para poner en marcha la comunicación. También se han creado los métodos de envío y recepción de datos, ligados a las funciones send y recv, pertenecientes a Winsock2.

gui/Conexión_cifrada.cpp

Contiene toda la interfaz que el management client ofrece para establecer una comunicación TCP/IP cifrada mediante el AES. Este sistema de cifrado necesita de una llave o key para poder funcionar, que en nuestro caso es el MD5 del password que el administrador ha introducido utilizando la clase Autentificacion.

gui/FormObjeto.cpp

Se encarga de añadir un nuevo objeto a la política que en esos momentos se está manejando en el management client. Entre otras comprobaciones se comprueba que la dirección o direcciones IP, según el caso, sean correctas.

gui/FormServicio.cpp

Se encarga de añadir un nuevo servicio a la política que en esos momentos se está manejando en el management client. Realiza las comprobaciones pertinentes, como si el puerto se encuentra dentro del rango de puertos disponibles.

gui/FrameXXXX.cpp

Las clases con esta notación sirven de soporte a los frames que soportan las distintas opciones tanto en la creación de objetos como en la creación de servicios. Mediante una ComboBox y en base a la selección fijada en esta, se mostrará uno u otro objeto de tipo frame, de forma que los demás se quedarán en un segundo plano.

gui/gui.cpp

Contiene los métodos necesarios para crear el formulario principal del management client. Implementa todos aquellos métodos que la GUI necesita para arrastrar objetos y servicios para ser añadidos a las reglas que forman la política, abrir menús y demás posibilidades que ofrecemos en la misma.

gui/ListaPoliticas.cpp

Detalla el número de políticas que hay en el servidor, así como algunos detalles relevantes de las mismas, con el uso de un formulario.

gui/MD5.cpp

Clase que dado un buffer te devuelve la firma digital o firma hash de dicho buffer.

gui/ObjetoRed.cpp

Se encarga de crear el objeto de red en base a los datos que o bien recibe de la política enviada por el management server, o bien a través del formulario de creación de objetos correspondiente.

gui/Politica.cpp

Implementa una política, pieza fundamental en el funcionamiento de FreeWall1, la cual puede contar con una lista de reglas NAT y una lista de Reglas de filtrado, así como una lista de objetos y servicios asociados a la misma. Nótese que aunque el management server tiene el NAT como una de sus posibles futuras ampliaciones, desde el management client ya se ha tenido en cuenta esa futura posibilidad, integrando este tipo de reglas dentro de las políticas que se manejan entre el management server y el management client.

gui/Protocolo.cpp

Contiene todos las posibles formas de comunicación entre el management server y el management client. Todas las funciones que incluye se asientan sobre las de Conexion_cifrada, siendo esta clase la que define el nivel inmediatamente inferior al fijado por Protocolo. El nivel más bajo lo ocuparían las funciones de la clase Conexión.

gui/ReglaFiltrado.cpp

Implementa una regla de filtrado, que incluye entre otros componentes una lista de servicios, una lista de objetos origen y una lista de objetos destino.

gui/ReglaNAT.cpp

Implementa una regla de tipo NAT que indica el tipo de servicio modificado a utilizar, el objeto destino y el objeto origen.

gui/Servicio.cpp

Clase que heredan tanto ServicioTCPUDP como ServicioICMP, incluye campos comunes a todos los tipos de servicios, como identificador o comentario.

gui/ServicioICMP.cpp

Implementa el servicio de tipo ICMP, de forma que incluye el código para identificar el tipo de mensaje ICMP que es. Más información sobre los tipos de mensajes ICMP en el apéndice E.

2. Técnicas Desarrolladas

Filtrado en plataformas Win32

Arquitectura de red de Win32
Network Driver Interface Specification
Técnica NDIS Hooking
Técnica de infección del ndis.sys

Filtrado en plataformas Linux

ShellCodes

Mapa de un proceso en memoria
Buffer overflows
Shellcodes
Shellcodes polimórficos
FreeWall1 vs shellcodes

Redes neuronales aplicadas a IDS

Stateful Firewall

SDK para la gestión de logs

Pasarela SMS
MySQL logs

Filtrado en plataformas Win32

Arquitectura de red de Win32

Como se puede observar en la figura 10, las plataformas Win32 ofrecen una arquitectura de red, dividida en varias capas, que siguen la distribución propuesta en [OSI94]. A su vez estas capas se subdividen en aquellas que trabajan en modo usuario y en modo kernel. Las capas que Microsoft define para su plataforma, son las siguientes.

Basic Network Services.
Proporciona servicios de red avanzados a las aplicaciones de usuario. Dentro de esta capa se encuadran funcionalidades como la gestión de las direcciones de red, ficheros y de otros servicios de red avanzados, tales como IPSec o QoS.

Interprocess Communications.
Se encarga de proporcionar soporte a los sistemas cliente/servidor y a los sistemas distribuidos. Así ofrece herramientas como los Remote Procedure Calls (RPC) y los Distributed Component Object Model (DCOM) entre otros.

Network Application Programming Interface.
Ofrece versiones de las Application Programming Interface (APIs) estándar más utilizadas en el entorno de red, como pueden ser Winsock, NetBIOS, Telephony API (TAPI) y Messaging API (MAPI) entre otros.

Transport Driver Interface.
Sirve de enlace entre las capas contenidas en el modo usuario y las contenidas en el modo kernel. A su vez ofrece una interfaz estándar entre los protocolos de red y los usuarios de esos protocolos.

Network Protocol.
Los protocolos de red ofrecen servicios a los usuarios y aplicaciones de estos, de forma que les permiten enviar datos a través de una red. Algunos de los protocolos que soporta la plataforma Win32 son TCP/IP, IPX/SPX, AppleTalk y NetBEUI.

Network Driver Interface Specification.
Sirve de capa de comunicación entre los protocolos de red y los dispositivos físicos que proveen de datos a estos. Como veremos a continuación, NDIS ofrece múltiples posibilidades a la hora de establecer esa relación protocolo-dispositivo, a través de los drivers de red. Esta capa soporta tanto protocolos orientados a conexión, como puede ser RDSI o ATM, como a no orientados a conexión, como los tradicionales Ethernet, Token Ring o Fiber Distributed Data Interface (FDDI).

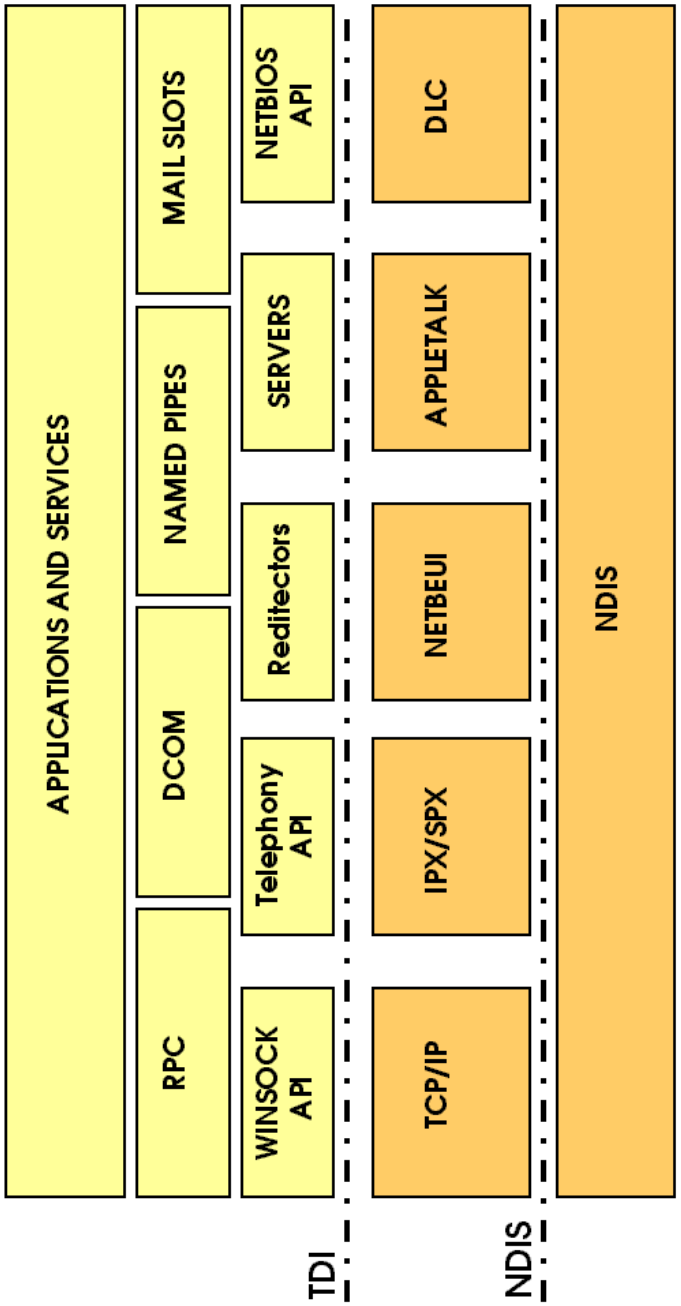

Figura 10. Arquitectura de Red de la plataforma Win32

Para establecer la comunicación el modelo por capas va transfiriendo la información de capa en capa hasta llegar al nivel kernel en donde el paquete a enviar comienza a formarse.

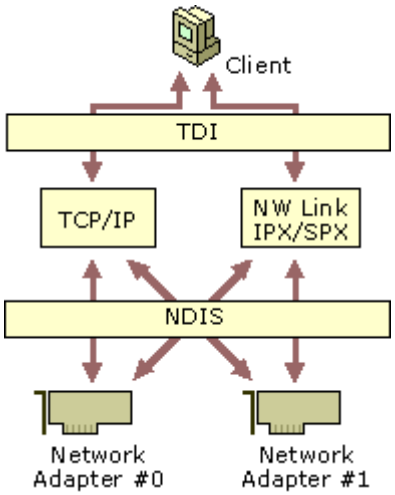

Figura 11. Comunicación entre capas en el network stack de Win32

En FreeWall1 hemos situado el filtrado a nivel kernel de forma que podamos discriminar de forma más precisa aquellos paquetes no deseados dentro de nuestra red. Para ello hemos hecho un uso extensivo de los servicios de la capa de Network Driver Interface Specification, estándar desarrollado inicialmente por 3Com y Microsoft, cuya última versión, la 5.1, da soporte a los nuevos sistemas Windows, como XP. Vamos a ver más a fondo como funciona esa capa, correspondiente a la subcapa Logical Link Control de la capa Data Link, la segunda de las capas OSI, y cual es su interacción con el resto de las capas que la rodean, en especial con las de menor nivel que ellas, relación que será clave para llevar a cabo nuestro filtrado.

Network Driver Interface Specification

NDIS es una especificación que permite a los protocolos de nivel superior, o protocolos de red, como TCP/IP y similares, acceder a los drivers de los dispositivos de red, también conocidos como MAC drivers, de una manera uniforme, ofreciendo una interfaz sencilla. NDIS se articula en torno al NDIS Wrapper, un framework que define como los protocolos de red deben comunicarse con los MAC drivers. Este framework sirve para desarrollar los NDIS drivers, que son los encargados de filtrar los paquetes hacia uno u otro protocolo, según los requisitos indicados por estos.

La base de todo el sistema NDIS, son los paquetes NDIS, o NDIS_PACKET, estos son los módulos básicos que gestiona la capa NDIS y los drivers asociados a ella. El NDIS_PACKET se encarga de encapsular los datos provenientes de los paquetes que circulan por la red, una vez han sido tratados en la capa MAC. Los paquetes de datos pueden llegar completos o fragmentados, para estos últimos habrá que esperar que el evento CompleteHandler ocurra, momento en el cual podremos proceder al tratamiento de dicho paquete. La estructura habitual de un elemento NDIS_PACKET se resume en la siguiente figura.

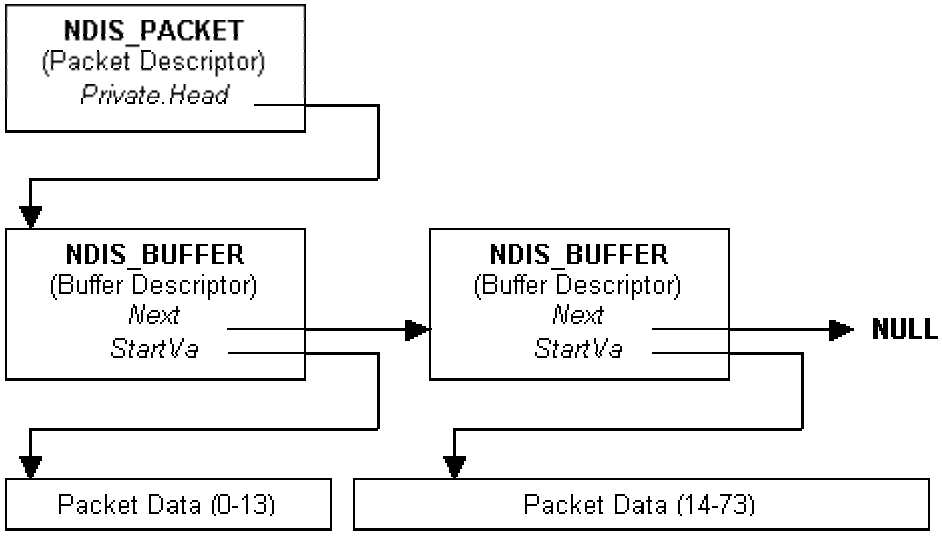

Figura 12. Estructura general de un NDIS_PACKET

Como se puede apreciar en la anterior figura, la estructura NDIS_PACKET consiste en un puntero a otro tipo de estructura, la NDIS_BUFFER. Esta estructura, a su vez puede estar enlazada con otro elemento de su mismo tipo, formando así, a modo de lista enlazada, una cadena en la cual se van almacenando los datos correspondientes a un paquete MAC. Aunque existe la posibilidad de que el NDIS_PACKET disponga únicamente de un solo NDIS_BUFFER, por lo general el tamaño de los paquetes a empaquetar requiere de más de un NDIS_BUFFER, de este modo, como puede verse en la figura ya indicada, cada NDIS_BUFFER guarda una parte del paquete original.

Las funciones proporcionadas por el NDIS Wrapper para el tratamiento de los NDIS_PACKET abarcan prácticamente todas las opciones disponibles para el desarrollador de drivers a nivel kernel. Pero la característica más interesante en la estructura NDIS_PACKET son una serie de campos disponibles para los drivers MAC y los drivers de protocolos de red, que permiten el intercambio de información entre estos. Para llevar a cabo el filtrado para un protocolo de red en concreto, éste debe haberse suscrito

previamente a los servicios de NDIS, mediante la llamada NdisOpenAdapter, fijando el tipo de filtrado y consecuentemente el tipo de paquetes que desea recibir. Varios protocolos de red pueden estar suscritos a un mismo tipo de datos, con lo que el filtrado de esos datos para un protocolo determinado no implica que dicho filtrado se realice para el resto de los protocolos de red suscritos al mismo.

Técnica NDIS Hooking

Como ya se ha apuntado anteriormente, en la plataforma Win32, existe la posibilidad de filtrar tanto a nivel usuario como a nivel kernel. La principal desventaja de filtrar a nivel usuario es que todas las posibles técnicas y servicios que existen a este respecto, se sirven de un driver a nivel kernel, con lo que el tiempo de respuesta se incrementa considerablemente, haciendo inviable su uso dado el perfil de prácticamente la mayoría de las aplicaciones de red. Estas soluciones a nivel usuario son principalmente dos. Una de ellas consiste en la modificación del servicio Winsock ligado a la capa de Network Application Programming Interface, que en cualquier caso no permite el acceso directo a los paquetes. La otra solución, aportada por Microsoft, es la librería Windows 2000 Packet Filtering Interface que permite el filtrado solamente en base a tres campos, dirección IP de origen, dirección IP de destino y puertos TCP a los que accede. Este tipo de drivers suele utilizarse en aplicaciones de sniffing, de forma que los paquetes se leen pero no se detienen esperando una decisión sobre si deben ser o no filtrados.

Frente a las opciones a nivel de usuario, hay tres posibilidades a nivel kernel, las cuales ofrecen la posibilidad de realizar un filtrado más profundo de los paquetes recibidos. De estas tres opciones disponibles una es un servicio que ofrece Microsoft a nivel de filtrado a nivel kernel, se trata del Windows 2000 Filter Hook Driver. Este driver en realidad es una ampliación en modo kernel del IP Filter Driver, ya incluído en versiones anteriores de Windows 2000. En segundo lugar tendríamos la opción de desarrollar un NDIS Transport Data Interface Driver, que no permite analizar el paquete, pero por otro lado ofrece paquetes defragmentados y con datos validados por la pila TCP/IP. Por último el driver que ofrece mayores posibilidades de filtrado, son los NDIS Inmediate Driver. Estos se sitúan entre el driver MAC y el driver correspondiente al protocolo suscrito al servicio NDIS, rompiendo anteriormente el enlace directo que existía entre ellos.

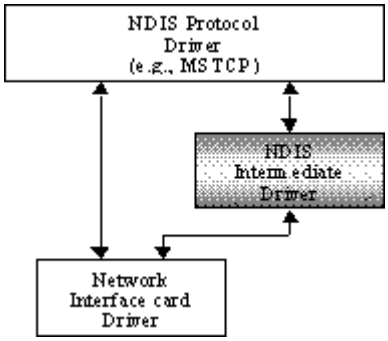

Figura 13. Configuración de un NDIS Intermediate Driver

De esta forma a través suyo pasan todos los paquetes que ambos drivers envían, pudiendo así realizar cualquier tipo de función sobre estos, como compresión o encriptación de los mismos. En nuestro caso nos centraremos en las posibilidades de filtrado que ofrece esta configuración.

Para obtener las máximas posibilidades de filtrado, en vez de crear un NDIS IM driver, hemos modificado directamente el que provee el sistema, mediante una técnica conocida como NDIS Hooking. El NDIS Hooking permite modificar las funciones del NDIS Wrapper adaptándolas a las necesidades particulares del firewall, así se crean versiones alternativas para ciertas funciones del NDIS Wrapper que no se adapten a nuestras necesidades.

La base de nuestra solución de filtrado para plataformas Win32 se basa en esta técnica. Para ello hemos desarrollado una serie de hooks, que modifican la acción del NDIS Wrapper adaptándola a nuestras necesidades. Por un lado tenemos una serie de hooks que redireccionan la suscripción y la eliminación de la misma, de protocolos hacia nuestras funciones, en concreto las funciones NdisRegisterProtocol y NdisDeregisterProtocol. Por otro lado creamos otros dos hooks redirigiendo en este caso la apertura y cierre de adaptadores de red, esto es de drivers MAC, en concreto redirigimos las siguientes funciones del NDIS Wrapper, NdisOpenAdapter y NdisCloseAdapter. Las funciones que hemos desarrollado y que sustituyen a las originales del Wrapper, una vez realizada la acción que requerimos vuelven a llamar a las originales del Wrapper, de forma que nos situamos en medio de la comunicación entre los adaptadores de red, los protocolos y el NDIS Wrapper.

Los anteriores hooks nos permiten conocer cuales son los dispositivos y protocolos conectados al sistema y su relación, pero para filtrar la información que intercambian en forma de paquetes necesitamos situarnos en medio de su comunicación, por lo que cada vez que se crea una nueva relación entre un protocolo y un adaptador de red, esto es cuando el protocolo se suscribe al NDIS Wrapper a través del hook que hemos introducido anteriormente, se crean un par de hooks, colocándolos en las funciones de envío y

recepción de datos del protocolo. De esta forma tenemos acceso a todos los paquetes que pasan por la network stack. En concreto hemos limitado nuestra monitorización a los paquetes que provienen o se dirigen a los protocolos TCP y WANARP (TCPIP para módems), por ser estos los de más interés.

Todos los anteriores hooks, tanto los referidos a capturar los protocolos y adaptadores conectados a la red, como los correspondientes a las funciones de envío y recepción de datos de los protocolos suscritos al NDIS Wrapper, se incluyen en un fichero llamado FreeWall1.sys. Nuestro objetivo ahora es integrar dicho fichero en el kernel de Windows, de cara a hacer efectiva la sustitución que éste hace del NDIS Wrapper.

Técnica de infección del ndis.sys

Para lograr nuestro objetivo tendremos que actuar sobre el ndis.sys en tiempo de ejecución, esto es cuando éste se encuentre en memoria. Para ello tendremos que operar sobre los campos adecuados de dicho fichero. Al igual que la práctica totalidad de los ficheros que se ejecutan en Windows, ndis.sys es un fichero que ha sido creado siguiendo el Portable Executable Format de Windows, el cual es común a todas las versiones de este sistema operativo, desde 32 a 64 bits, incluyendo cierto soporte con versiones de MSDOS.

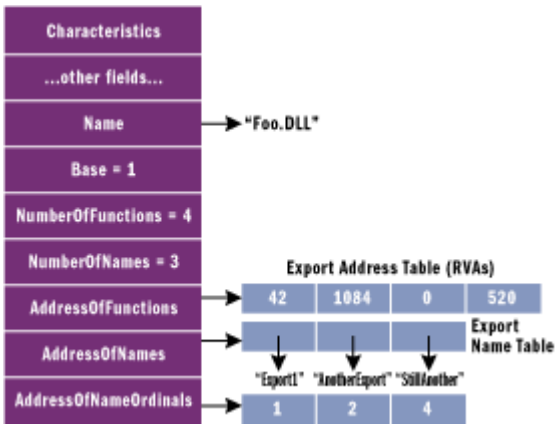

Figura 14. Formato del Portable Executable Format, poniendo de relieve la tabla de exportaciones

Nuestro objetivo es cambiar las funciones que ndis.sys para que apunten a Freewall1.sys, de forma que todo lo que se cargue después sobre la network stack reconozca la modificación que hemos introducido en ndis.sys. Al igual que cualquier otro Portable Executable, ndis.sys se carga en memoria, mapeándose con el offset correspondiente cada una de las secciones del formato. Una vez cargado en memoria ndis.sys, antes de que el sistema operativo cargue cualquier otra utilidad, tendremos que cargar el FreeWall1.sys, el cual se encargará de ejecutar las rutinas necesarias para redirigir en la tabla de exportaciones del ndis.sys las funciones que así lo requieran para llevar a cabo NDIS hooking. Una vez modificadas estas, el resto de utilidades que se carguen en el sistema funcionarán de acuerdo a las modificaciones que hemos introducido. Al estar residentes en memoria, cada vez que se reinicie el sistema habrá que volver a repetir todo el proceso de infección.

Filtrado en plataformas Linux

Hasta la versión 2.3 del Kernel de Linux el filtrado de paquetes a nivel kernel era en exceso complejo, de forma que para ligar un protocolo a los paquetes recibidos a través de un único adaptador de red suponía tener un conocimiento bastante completo de dicho Kernel. En las últimas versiones del Kernel 2.3 y ya a partir de la versión 2.4 del mismo se incluyó de serie un framework surgido de la reconstrucción de toda la parte de filtrado de paquetes en Linux. Netfilter supuso un cambio en la forma de llevar a cabo el filtrado en Linux, de forma que las antiguas herramientas de filtrado de paquetes como ipfw o iptables se han reescrito utilizando este nuevo framework.

Netfilter ofrece una solución global de filtrado, que funciona tanto a nivel kernel como a nivel usuario, de esta forma una aplicación de nivel usuario puede recibir, según sus necesidades, los paquetes filtrados en base a estas por Netfilter. Del mismo modo los módulos que forman parte del Kernel de Linux pueden suscribirse a los servicios de Netfilter y recibir aquellos paquetes que consideren adecuados a sus necesidades.

La arquitectura de Netfilter se muestra en la figura 15. En ella se puede comprobar una serie de puntos de control, o hooks, los cuales aceptan suscripción de módulos tanto a nivel kernel como a nivel usuario. Antes de iniciar su procesamiento a través de Netfilter, los paquetes atraviesan una serie de test de integridad básicos, que comprueban entre otras cosas la corrección del checksum del paquete en cuestión. Una vez el paquete inicia su camino a través de la arquitectura de Netfilter, en base a un routing inicial deberá decidir cual será su camino, bien continuar su camino hacia otro host, a través de FORWARD, o bien pasar a ser analizado por el actual proceso en marcha, a través de INPUT. Asímismo existe una tercera opción, OUTPUT, para los paquetes que se generan en el proceso en marcha. En realidad estos tres caminos, son tablas en las que se indica el camino que debe de seguir cada paquete, de forma que si no figura ninguna regla que indique que hacer con éste, el paquete será por defecto descartado.

Junto a estas tablas, existen seis hooks o puntos de control para los paquetes que atraviesan la arquitectura de Netfilter, NF_IP_PRE_ROUTING, NF_IP_LOCAL_IN, NF_IP_FORWARD, NF_IP_POST_ROUTING y NF_IP_LOCAL_OUT. A través de las funciones nf_register_hook() y nf_unregister_hook(), se puede asociar una función a dicho hook, de forma que cada vez que un paquete llega al hook ejecuta la función, y en base a la salida de esta llevará a cabo una de las siguientes acciones.

NF_ACCEPT. El paquete continua a través de la arquitectura Netfilter
NF_DROP. El paquete se filtra, es eliminado.
NF_STOLEN. El paquete es capturado de la arquitectura, luego habrá que reintroducirlo
NF_QUEUE. Se añade a la cola ip_queue, generalmente para su uso a nivel usuario
NF_REPEAT. Vuelve a llamar de nuevo al hook

Utilizando este framework hemos asociado una serie de funciones en base a nuestras políticas, detalladas en el capítulo anterior, a los hooks NF_IP_LOCAL_IN, NF_IP_FORWARD y NF_IP_LOCAL_OUT, de forma que el paquete puede o bien seguir adelante, en ese caso nuestras funciones asociadas a estos hooks devuelven NF_ACCEPT, o bien ser filtrados, caso en el que devolverían NF_DROP.

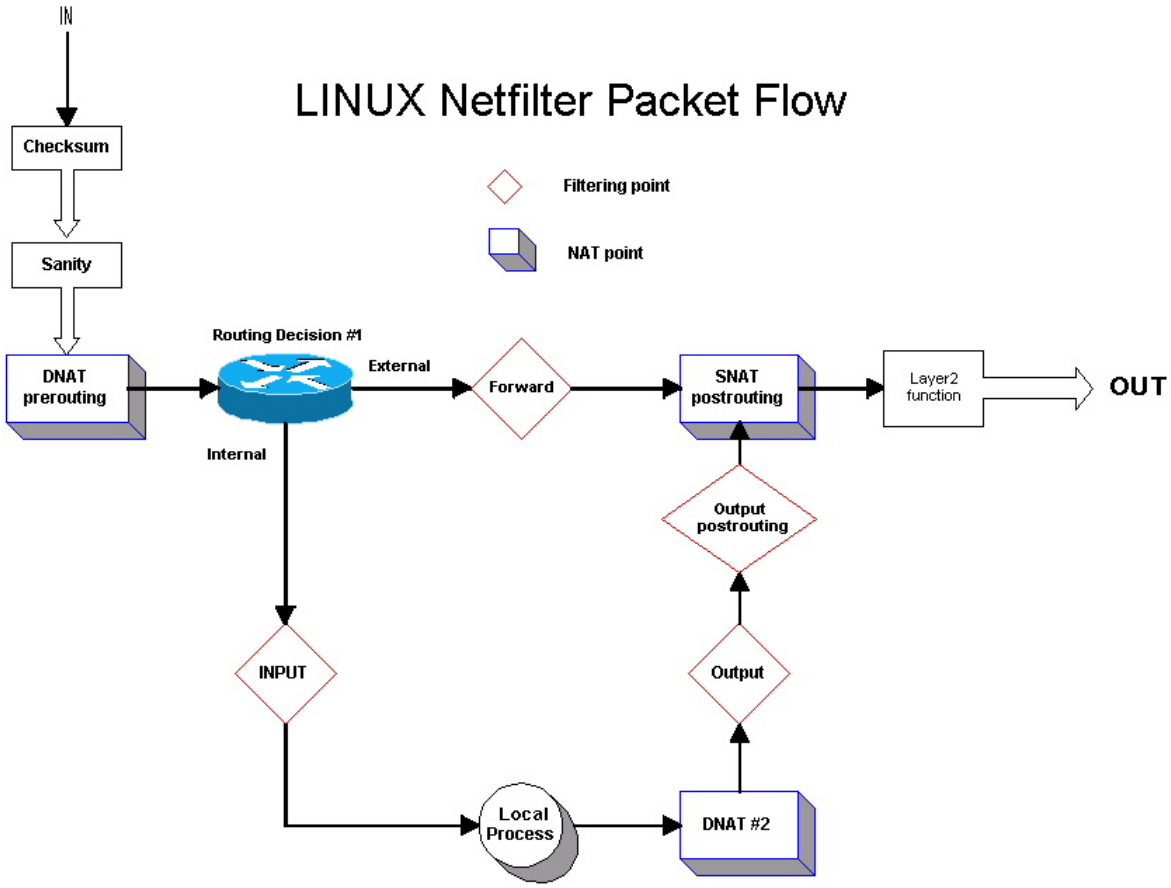

Figura 15. Arquitectura del framework Netfilter. [NET01]

ShellCodes

Mapa de un proceso en memoria

Cuando un binario se ejecuta, una imagen de ese binario se mapea en memoria, constituyendo un proceso, que el sistema operativo se encargará de ejecutar. Esa imagen en memoria está dividida en tres partes bien definidas.

Zona de código.
En ella se incluye el código del programa asociado al binario que ha sido ejecutado. Está región es de sólo lectura, de forma que cualquier intento de modificarla producirá una violación de segmento.

Zona de datos.
Esta zona junto con la pila, son variables en tamaño, adaptándose a las necesidades del usuario en cada momento. Esta zona en concreto contiene principalmente datos estáticos, ya hayan sido estos inicializados o no. Para modificar el tamaño de esta región existen varias llamadas al sistema disponibles, como brk( ), de forma que si en un momento dado no se puede ofrecer más memoria, ya que la memoria disponible entre la zona de datos y la pila se ha agotado, el proceso asociado a esta imagen de memoria se paralizaría, y se volvería a planificar de forma que tuviera una mayor cantidad de memoria.

Pila.
Esta zona contiene las funciones y variables asociadas a estas que se van creando en tiempo de ejecución. Su funcionamiento se basa en la creación de una serie de marcos que se cargan en lo alto de la pila, siguiendo el modelo TAD de pila con LIFO. En esos marcos, en el caso de las funciones, se guarda información como sus variables locales o la información de retorno a la dirección en la que la función comenzó a ejecutarse. Esta dirección es clave en la técnica que abordamos, ya que será la que nos permita jugar con el sistema. La pila contiene un puntero a lo algo de la misma, y en la mayoría de las arquitecturas incluye un segundo puntero que apunta a un punto fijo del frame, de forma que el cálculo de las direcciones donde se encuentran los parámetros de la función y las variables locales a estas resulta menos tedioso. En la arquitectura que hemos utilizado, compatible con la arquitectura x86 y basada en un procesador NSC SC1100, estos punteros pasan a recibir el nombre de esp para el puntero que apunta a la cima de pila, y ebp para el puntero de referencia dentro del frame.

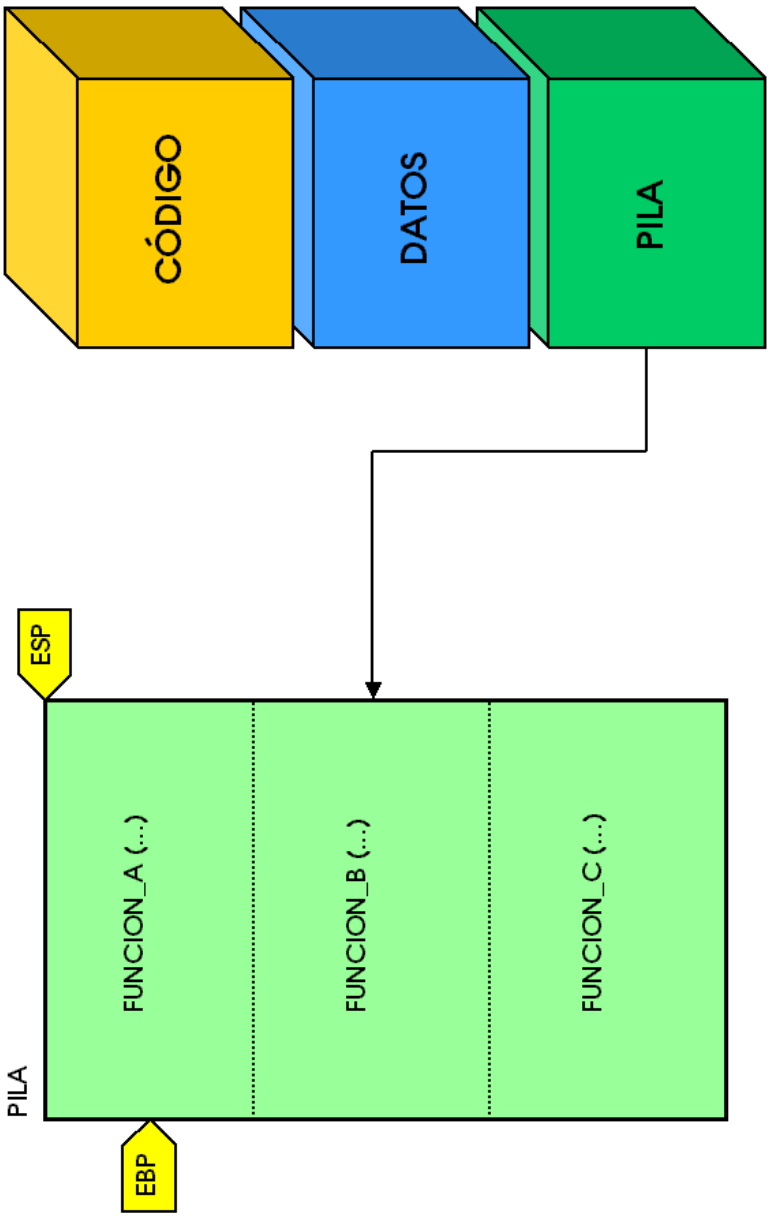

Figura 16. Imagen de un proceso en memoria para plataformas x86

Buffer overflows

La parte del mapa en memoria de un proceso que más nos interesará para ilustrar una de las formas más usuales de ataques a las que un firewall se enfrenta, será la pila. Como hemos indicado anteriormente allí se guarda toda la información de las funciones que se van ejecutando en el proceso, entre otras cosas las variables locales asociadas a estas. Será jugando con la memoria asignada en la pila a estas variables como se logrará producir el ataque. Un elemento clave en C son los buffers arrays, más conocidos como arrays. Estos en realidad son zonas de memoria de un tamaño determinado, o buffers, que contiene datos del mismo tipo.

Son estos buffers los que dan nombre a la técnica del buffer overflow. Esta técnica se basa en la posibilidad que tiene un usuario de acceder por una mala programación de una aplicación a posiciones de memoria que deberían en cierta forma estar ocultas al mismo. En concreto la zona más interesante a la cual se tendrá acceso será al campo return address de la función que produce el buffer overflow, de forma que podremos modificar esa dirección de retorno, a la cual saltará la función una vez terminada su tarea, para que apunte a nuestro shellcode, el cual ejecutará aquellas acciones que consideremos adecuadas.

ShellCodes

Una vez se ha modificado la dirección de retorno de la función, lo habitual será intentar abrir una shell, si el programa en cuestión que ha generado el buffer overflow, contiene su código. Por lo general esto no será así, o el atacante querrá ejecutar una serie de instrucciones distinta. Para poder llevar a cabo la ejecución de estas instrucciones habrá que incluirlas en el espacio que proporciona el buffer overflow, haciendo que la dirección de retorno de la función que produce el buffer overflow apunte al comienzo de estas. El problema de acertar con la dirección de retorno en la dirección exacta donde comienza el shellcode no es trivial, y por lo general requiere de un número elevado de intentos para llevarse a cabo. Es por esto que junto con el shellcode se incluyen un elevado número de operaciones NOP, disponible en la mayoría de las arquitecturas, o similares a esta, como las que se incluyen más adelante, antes de la dirección de comienzo del shellcode para que haya más posibilidades de acertar.

De esta forma prácticamente siempre se acierta con la dirección del shellcode, gracias a que las NOP llevarán el flujo de ejecución hasta el comienzo del shellcode. El formato general de un shellcode consta de dos partes principales, como se puede deducir de la técnica de instalación del shellcode que hemos detallado anteriormente.

Sección NOP.
Forma la cabecera del shellcode, y consiste en un gran número de instrucciones NOP o instrucciones equivalentes a estas seguidas. Al final de esa secuencia de NOP se encuentra la segunda parte del shellcode, el código propiamente dicho.

Sección de código.
Aquí se incluye el código del shellcode, que por lo general contendrá una cadena similar a /bin/x en plataforma Linux, donde x puede ser cualquiera de los shell que ofrece dicho sistema, o c:/cmd.exe, en el caso de que el shellcode vaya orientado hacia una plataforma Win32.

ShellCodes polimórficos

Como hemos visto en el anterior capítulo los IDS se pueden basar en dos tipos de búsqueda para localizar Shellcodes dentro de los paquetes que pasan a través suyo. Un primer tipo de búsqueda sería tratar de encontrar una cadena larga de instrucciones NOP, fácilmente localizables buscando dentro del paquete por la cadena 0x90, o similares a esta. Un segundo tipo de búsqueda correspondería a la búsqueda de patrones similares a los indicados anteriormente, teniendo en cuenta la plataforma que se va a atacar. Es este tipo de búsqueda el que implementan la mayor parte de los IDS comerciales, mediante la inclusión de reglas, en donde se especifican los patrones a buscar.

El problema con la búsqueda de patrones similares a los indicados surge con la última generación de shellcodes, que incluyen métodos de encriptación para que el código del shellcode no sea accesible. Por lo general los métodos de encriptación utilizados en esta nueva generación de shellcodes no suelen ser muy complejos, debido a que los IDS no cuentan con mucho tiempo para tratar los paquetes que entran, ya que sino sería impracticable su función. El formato de esta nueva generación de shellcodes cambia ligeramente respecto de la generación anterior.

Los shellcodes polimórficos incluyen una nueva área, en la cual se incluye la máquina de encriptación. Se trata de una máquina polimórfica de forma que el código del shellcode dispone cada vez que es encriptado de una rutina de desencriptación distinta. Actualmente no existe un gran número de este tipo de máquinas, de entre las cuales la más conocida es ADMmutate, aunque ya hay una gran experiencia y un buen número de trabajos sobre máquinas de este tipo, realizados en la creación de virus polimórficos, que utiliza los mismos principios, con lo que en un futuro este tipo de máquinas se irán popularizando, trabajos como [PHR01] ya apuntan en esa dirección, y con ellas, los shellcodes polimórficos.

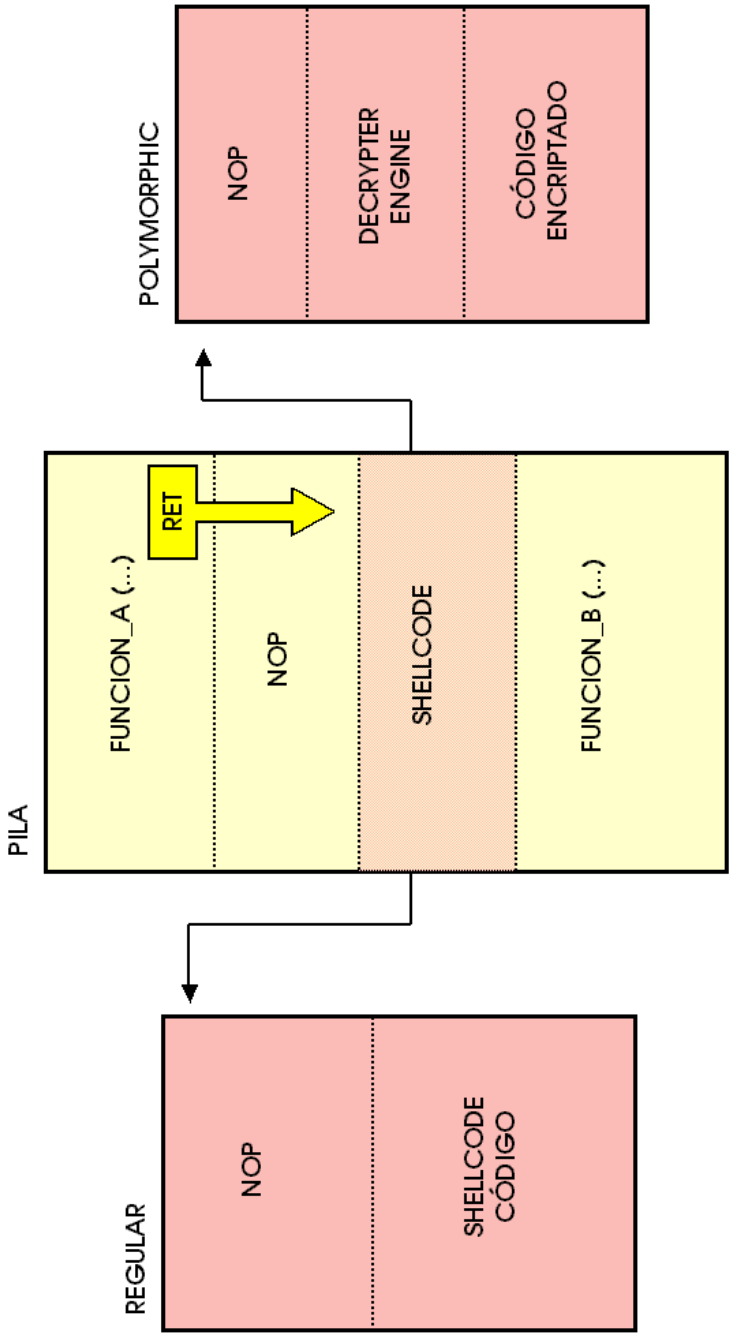

Figura 17. Formatos de shellcode y su distribución en la pila bajo buffer overflow

FreeWall1 vs shellcodes

Como hemos visto en los anteriores apartados, las técnicas de escritura de shellcodes están en constante evolución, y en breve el uso de algoritmos más sofisticados a la hora de codificar el código embebido en los shellcodes hará, como en el caso de los virus, muy difícil su detección, más teniendo en cuenta el tipo de aplicación en la que nos movemos, donde el tiempo es clave. Existen actualmente dos formas de abordar la detección de shellcodes en IDS, mediante reglas que a modo de sistema experto detectan que patrones de cadena embebidos en los paquetes pueden ser susceptibles de formar parte de un shellcode. Esta solución implica tener actualizada una base de datos con todos los nuevos patrones que puedan surgir, lo que ya de por si compromete la seguridad de los sistemas que utilizan ese tipo de IDS como protección, ya que desde que surge una nueva técnica hasta que esta se hace pública suele pasar un tiempo no despreciable. Junto a este tiempo no asumible en una aplicación de este tipo, si suponemos un sistema experto ideal donde estuvieran almacenados todos los patrones de ataques disponibles, aún habría un segundo inconveniente, la codificación de la nueva generación de shellcodes. Aunque actualmente las máquinas polimórficas son sencillas, ya están apareciendo nuevas máquinas con algoritmos que ofrecen una seguridad mayor sobre su código al atacante. La situación requerirá que los IDS incluyan métodos de desencriptación cada vez más complejos, lo cual llevará a un estado en el que estos necesiten un tiempo no asumible para analizar cada paquete que llegue al sistema.

Por tanto, toda la generación de IDS que utilizan estas técnicas verá reducido su radio de acción a los shellcodes de primera generación, que no disponen de encriptación alguna. Frente a esta solución nuestra propuesta se basa en el análisis de la cadena de instrucciones NOP situadas en la cabecera del shellcode. Este análisis es sencillo y reduce al mínimo la carga que el IDS pueda tener al respecto en su rendimiento. El análisis de estas instrucciones NOP por parte de los IDS ha propiciado la aparición de nuevas ideas por parte de los desarrolladores de shellcodes. En concreto la más utilizada ha sido la sustitución de esta instrucción, por instrucciones que equivalen al NOP en cada uno de los repertorios de instrucciones de cada arquitectura. Frente a esa situación hemos ampliado el rango de búsqueda de instrucciones NOP a todos los tipos de instrucciones equivalentes en cada una de las arquitecturas disponibles, así FreeWall1 detecta shellcodes en arquitecturas IA32 (x86), SPARC y PA-RISC (HPUX). El apéndice D incluye una muestra para distintas arquitecturas de las instrucciones similares a NOP que FreeWall-1 es capaz de detectar.

Redes neuronales aplicadas a IDS

La búsqueda de sistemas cada vez más eficientes en la búsqueda de posibles patrones de ataque, ha traído consigo la reutilización de modelos ya usados en otros campos, como el control, es el caso de las redes neuronales artificiales. Existen un gran número de trabajos al

respecto como [CUN99] o [DRA01], que muestran como las redes neuronales pueden ser un eficiente sistema IPS.

Las redes neuronales artificiales se basan en el mismo principio de las redes neuronales biológicas, son un conjunto de neuronas que interaccionan entre sí en la búsqueda del reconocimiento de una salida, en nuestro caso un patrón de ataque, en base a unas entradas, en nuestro caso los datos que llegan al IDS. Una red neuronal artificial cuenta con tres zonas bien diferenciadas, la zona de entrada, la zona oculta y la zona de salida, todas ellas constituidas como conjuntos de neuronas, de forma que existen diferentes formas de interaccionar entre las neuronas de uno y otro nivel. Para analizar un problema dado se distribuyen los datos de entrada a las neuronas que forman la zona de entrada, obteniéndose las soluciones en una o varias de las neuronas que forman parte de la zona de salida.

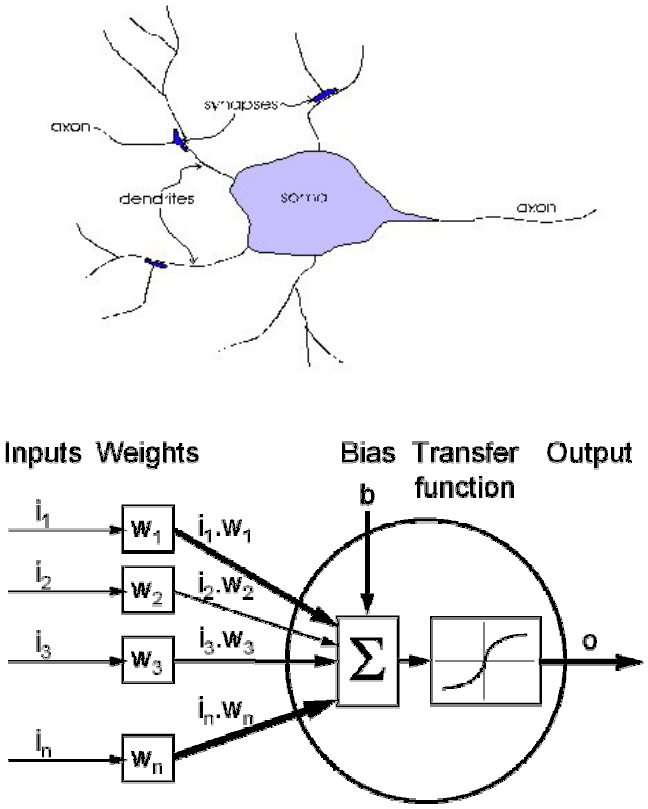

Figura 18. Modelo de neurona biológica (arriba) y artificial (abajo)

Dada una entrada de datos, en nuestro caso paquetes con posibles patrones, la red neuronal es capaz de analizar dichos paquetes en la búsqueda de patrones y dar una respuesta afirmativa si dichos patrones son encontrados. Para ello hay que entrenar a la red con una conjuntos de datos de entrada y sus correspondientes salidas. De esta forma la red neuronal

irá modificando los pesos asociados a cada una de las entradas que recibe una neurona procedentes de otras neuronas adyacentes a la misma, hasta que dadas las entradas obtenga las salidas indicadas en dicho conjunto de datos. Para llevar a cabo este aprendizaje la red neuronal utiliza uno de los muchos algoritmos de aprendizaje disponibles en la actualidad. Aunque esta es la forma más usual de utilizar una red neuronal, existen determinados modelos de redes neuronales que son capaces de aprender por si mismas en base a un conjunto de entradas sin conjunto de salidas asociado.

De entre el gran número de redes neuronales disponibles, en nuestro caso hemos utilizado una red neuronal de Elman descrita en [ELM90], un tipo de red neuronal que simula el uso de una memoria, gracias a que el tiempo es un aspecto relevante dentro del diseño de la red, además de las entradas de la misma. Gracias a esta memoria este tipo de red es capaz de reconocer patrones de comportamiento y por lo tanto susceptible de detectar posibles ataques.

Para diseñar y simular nuestra red neuronal de Elman, hemos utilizado el Stuttgart Neural Network Simulator (SNNS) [SNN01], desarrollado en el Institute for Parallel and Distributed High Performance Systems de la Universidad de Stuttgart.

Nuestro objetivo ha sido entrenar a la red neuronal para que detectase ataques de tipo SQL injection, Command injection y Cross Site Scripting (XSS). Para lo cual hemos utilizado el conjunto de datos de entrada y salida descrito en el apéndice B para entrenarla. En este conjunto las S puede ser cualquiera de las siguientes palabras "update", "xp_", "sp_", "insert", "select", "union", "drop", "having", "group", "or", "and", "into" y "file", la X cualquiera de las siguientes palabras "script", "alert(", "window." y "document.", y la C cualquiera de las siguientes palabras "bin", "sbin", "cat", "xterm", "cmd.exe", "%00", "etc", "win", "http" y "ftp", gracias al uso de estas categorías léxicas optimizamos el rendimiento. Una vez obtenemos la salida de la red neuronal utilizamos la utilidad net2c, parte del proyecto SNNS, que porta la salida de la red neuronal a código C, de forma que si cualquiera de las neuronas de salida de la red ha reconocido un patrón de ataque de los indicados en el siguiente entrenamiento, descartamos el paquete. Al final de este conjunto de entrenamiento se incluye una sección NORMAL que incluye aquellas aquellos patrones de salida que deben ser reconocidos como salidas legítimas de la red neuronal, que no constituyen riesgo de ataque, con lo que el paquete en este caso continuaría su camino.

Stateful Firewall

A lo largo de este capítulo hemos visto distintos niveles de filtrado para el tráfico de datos que pasa a través de FreeWall1. Siguiendo esa filosofía, tras el filtrado y el análisis de herramientas IPS, hemos utilizado una última capa de protección, convirtiendo a FreeWall1 en un stateful firewall. Mientras que tanto el filtrado como el análisis IPS que funciona sobre él actúan como un proxy frente a los paquetes que pasan a través suyo, analizando

todo lo cual les hace ser sistemas especialmente sensibles a los cambios de tráficos llegando incluso a poder ser saturados si el tráfico es excesivamente elevado, el stateful firewall añade un nivel a nuestro IDS que soporta cualquiera intensidad de flujo de datos.

El stateful firewall se basa en el concepto de estado de la conexión. El estado viene determinado por una serie de parámetros que varían según el protocolo sobre el que esté actuando. Por lo general todos los datos serán a nivel de la capa de transporte, aunque debido a la naturaleza de algunos servicios como FTP, donde es necesaria más de una conexión utilizando el protocolo de transporte TCP, también podrán existir datos a nivel de la capa de aplicación, que vendrán determinados por cada tipo de servicio en concreto. Tal es el caso de SMTP que incluye los comandos propios de ese protocolo, que permiten analizar si cabe de forma más certera, la consistencia del paquete en ese momento de la comunicación entre dos sistemas utilizando dicho protocolo.

El stafeful firewall, cada vez que uno de los hosts que protege hace una petición a través suyo de uso de un servicio en otro host externo a la red protegida, crea un estado en donde guarda la información de esa conexión en particular. Ese estado se guarda en una tabla de estados, donde se incluyen todas las conexiones que se están manteniendo en ese momento a través de dicho stateful firewall. De esta forma se guarda debida cuenta de todas las conexiones que han salido de la red protegida, conexiones que pueden ser consideradas por el stateful firewall como válidas. De esta forma se crea un potente filtro para todos los paquetes que lleguen desde fuera hacia la red protegida. Así cualquier paquete exterior que pretenda pasar el filtro del stateful firewall tendrá que corresponder a alguna de las conexiones que están identificadas mediante un estado en la ya citada tabla de estados.

Protocolos de transporte como TCP, perteneciente a los protocolos orientados a conexión, guardan una estrecha relación con los stateful firewall, ya que su método de conexión viene dado por un autómata de un número finito de estados, lo cual está en estrecha relación con los estados que maneja el stateful firewall. El protocolo TCP tiene asociado un autómata de once estados, seis de los cuales permiten la conexión entre dos host utilizando este protocolo, mientras que otros cinco permiten la desconexión entre los mismos. De esta forma en el estado correspondiente a la comunicación mediante TCP se sabrá en cada momento en que momento de la comunicación TCP se encuentra cada uno de los dos hosts, tanto el interno a la red como el externo a la misma.

El protocolo TCP se resumiría en los siguientes pasos, que irían representados convenientemente en los estados asociados. El host cliente, que forma parte de la red protegida por el stateful firewall, envía un paquete con su bit SYN a uno, para establecer la comunicación con el host destino externo a la red. Al atravesar el stateful firewall, éste reconoce el bit SYN como una señal de que una nueva comunicación se está poniendo en marcha, con lo que crea un estado con la información integrada en ese paquete, como la IP origen, IP destino y demás campos de interés. Cuanto mayor sea la información

almacenada en el estado menor será la posibilidad de que un paquete con no muy buenas intenciones pase indebidamente el filtro fijado por el stateful firewall.

Si el host destino externo a la red protegida que actúa como servidor dispone del servicio requerido por el host cliente, entonces envía de vuelta un paquete en el que tanto el bit SYN como el bit ACK han sido fijados a uno. Al llegar este paquete al stateful firewall, éste reconoce la conexión o estado al que pertenece gracias al análisis de la información que lleva dentro, y lo reenvía hasta su destinatario, el host cliente. Éste para terminar de establecer la conexión envía un paquete en el que sólo el bit ACK está a uno, de forma que la comunicación queda establecida, y el stateful firewall fija el estado asociado a esa conexión como ESTABLISHED, a la espera de próximos eventos. El cierre como se puede apreciar en la figura que se muestra a continuación sigue una serie de pasos similares hasta que la conexión se da por cerrada. En dicha figura se muestran los estados por los que va pasando la conexión TCP que serán los distintas fases por las que pase el estado del stateful firewall asociado a dicha conexión.

En el caso del protocolo TCP al ser un protocolo orientado a conexión la adopción del modelo de estados que propugna el stateful firewall es inmediato, sin embargo esto no sucede para protocolos de transporte no orientados a conexión como pueden ser UDP o ICMP. En ambos casos no existe un autómata asociado que defina una serie de estados, ni siquiera existe una forma de establecer cierta comunicación entre los dos hosts extremos de la misma, de forma que un estado será eliminado de la tabla de estados cuando se haya superado un cierto tiempo límite, provocando un timeout. Este último punto es de vital importancia en el rendimiento del stateful firewall, ya que una tabla llena frenaría en seco la entrada de paquetes, base de los ataques Denial of Service (DoS) que envían un gran número de mensajes con el bit SYN activado de cara a mantener siempre llena la tabla de estados del stateful firewall a atacar, y evitando que éste pueda establecer otras comunicaciones.

FreeWall1 es un stateful firewall pensado para reducir al máximo la complejidad y homogeneizar el desarrollo de los autómatas ligados a cada protocolo. Es por ello que se ha optado por un autómata de un solo estado, de forma que abarcara tanto orientados a conexión como los no orientados. Además de la información necesaria para crear un estado dada una conexión, IP destino, IP origen y protocolo utilizado, también se ha incluido como información adicional el puerto desde el que opera la aplicación. De esta forma la tabla de estados de FreeWall1 dispone de una colección de estados, que contienen la siguiente información.

Identificador. Número único que identifica al estado.
Syncronise. Bit SYN.
Ack. Bit ACK
Protocol. Protocolo que se utiliza en la conexión.
Port. Puertos implicados en la conexión, ya sean TCP o UDP.

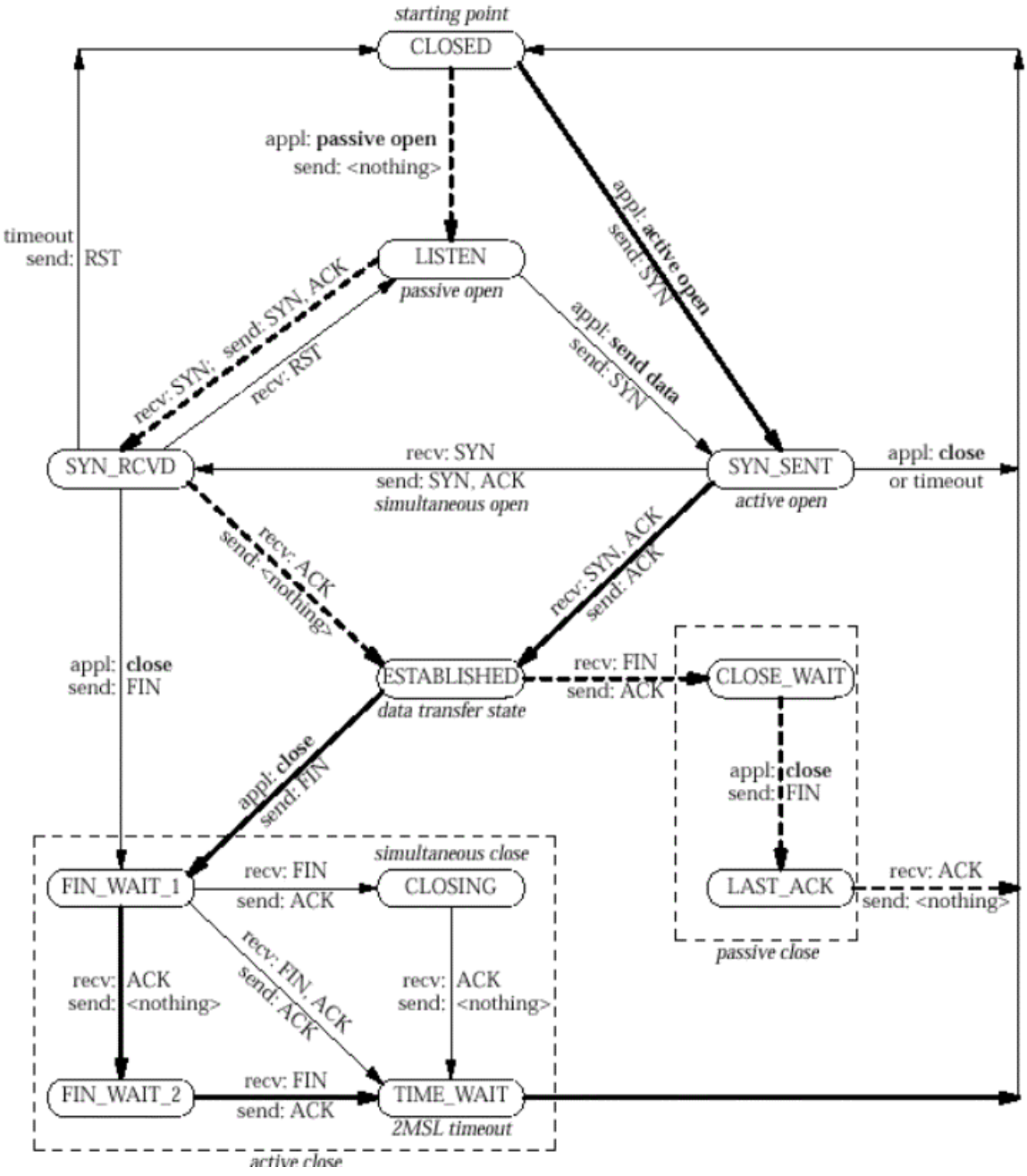

Figura 19. Autómata que define el protocolo de transporte TCP. [ABD01]

SDK para gestión avanzada de logs

Hemos desarrollado un SDK para la creación de plugins para FreeWall1. La documentación de dicho SDK se encuentra disponible en el apéndice A. Utilizando dicho SDK hemos desarrollado dos plugins para la gestión de los logs procedentes del driver.

Plugin de almacenamiento de registros en una BBDD MYSQL

Se desarrolló un plugin de gestión avanzada de registros logs en C que se utilizaba como puente a una aplicación en Java para el almacenamiento de dichos registros logs en una base de datos MYSQL.

El plugin en C no es más que un puente que llama a la función WinExec para ejecutar en el entorno Java nuestra clase de gestión de logs.

Para un correcto funcionamiento debemos tener acceso a un servidor MYSQL correctamente configurado donde se conocen los siguientes datos:

- Usuario y contraseña.
- Dirección ip del servidor.
- Nombre de la base de datos sobre la que trabajar
- Nombre de la tabla en la que guardar los registros logs.

Para la creación de la base de datos se usará el siguiente comando una vez autentificado y dentro del programa cliente MYSQL en el servidor donde esté alojada la BBDD:

create database FW1;

Para la creación de la tabla que guardará los logs se usarán los siguientes comandos:

Use FW1;
Create table logs (id int(6) autoincrement, log varchar(256));

La inserción de registros logs en la base de datos se hará mediante la siguiente query SQL (nótese que el gestor de base de datos al ver el atributo autoincrement cambiará el valor id por un valor adecuado):

Use FW1;
Insert into logs values(‘’,’registro_log’);

Para una posterior consulta de registros en la base de datos se utilizará la siguiente sentencia SQL:

Use FW1;
Select * from logs;

Desde Java se usará el driver JDBC correspondiente a MYSQL para ejecutar dicha inserción como se puede apreciar en el código Java.

```
import java.lang.*;
import java.sql.*;
import java.util.Vector;
import java.util.*;

public class Basedatoslogs {
        private Connection dbcon;
        private Statement statement;

        private void init_db() {
                String loginUser = "root";
                String loginPasswd = "admin";
                String loginUrl = "jdbc:mysql://127.0.0.1:3306/FW1";
                try {

                        Class.forName("org.gjt.mm.mysql.Driver");
                        dbcon = DriverManager.getConnection(loginUrl,
                        loginUser, loginPasswd);
                        statement = dbcon.createStatement();

                }
                catch (SQLException ex) {
                        ex.printStackTrace();
                        return;
                }
                catch (java.lang.Exception ex) {
                        ex.printStackTrace();
                        return;
                }

        }


        private void end_db() {
                try {

                        statement.close();
                        dbcon.close();
                }
                catch (SQLException ex) {
```

```
                        ex.printStackTrace();
                        return;
                }
                catch (java.lang.Exception ex) {
                        ex.printStackTrace();
                        return;
                }

        }

       public boolean registraLog(String logs) {

          boolean correcto=false;
          init_db();

          try {

               String inserta = "INSERT INTO logs VALUES
                                    ('','"+log+"')";
               int res = statement.executeUpdate(inserta);
               if (res > 0) {
                   correcto=true;

               }

           } catch (SQLException ex) {
                        ex.printStackTrace();
                        //return(correcto);
           } catch (java.lang.Exception ex) {
                        ex.printStackTrace();
                        //return(correcto);
           }
           end_db();
           return (correcto);

       }


}
```

Plugin de envío de registros por SMS

Se desarrolló un plugin de gestión avanzada de registros logs en C que se utilizaba como puente a una aplicación en Java para el envío de registros a traves de la pasarela de SMS de la empresa PeopleCall.

El plugin en C no es más que un puente que llama a la función WinExec para ejecutar en el entorno java nuestra clase de gestión de logs.

Para un correcto funcionamiento debemos tener acceso a Internet correctamente configurado donde se conocen los siguientes datos:

- Usuario y contraseña de la pasarela de PEOPLECALL.

Al no existir una pasarela de uso directo a traves de sockets o con un protocolo especifico, desarrollamos un cliente en Java que emula al navegador conectando con el servidor WEB de PEOPLECALL donde se autentificará y posteriormente enviará el mensaje SMS emulando la petición HTTP como si de una persona física se tratará y no un proceso robot.

En una primera fase el cliente envía una petición de LOGIN con su nombre de usuario y contraseña. El servidor verificará dicha petición y en caso de ser correcta nos enviará una COOKIE de sesión que utilizaremos en la segunda fase para el envío del mensaje SMS.

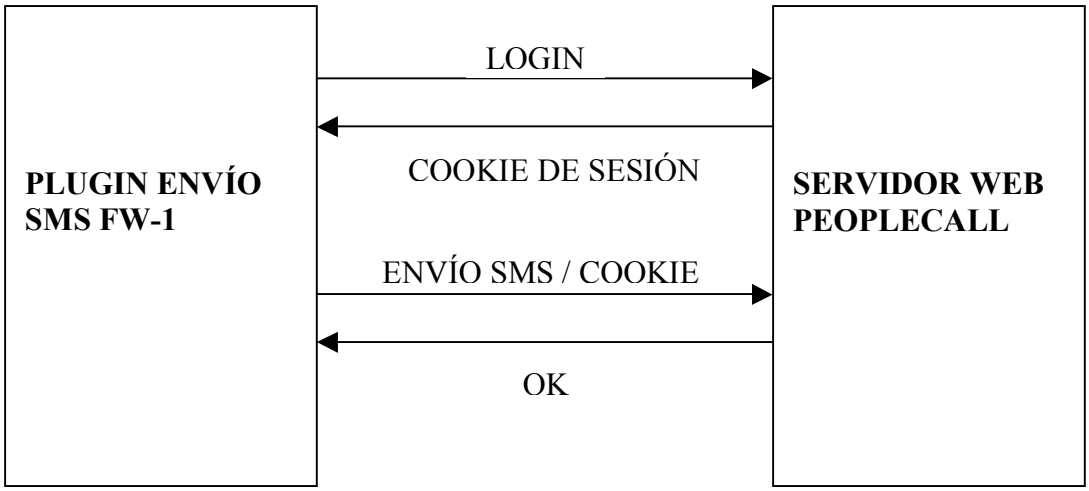

Figura 20. Diagrama de conexiones con la pasarela de SMS PeopleCall

Desde Java se usará el entorno de SOCKETS para enviar las dos peticiones HTTP como se puede apreciar en el código Java.

```
import java.io.*;
import java.net.*;

public class SMS {

 private String socket_envia(String host, int puerto, String output) {

        Socket smtpSocket = null;
        DataOutputStream os = null;
        BufferedReader is= null;
        String input = new String();

        try {
            smtpSocket = new Socket(host, puerto);
            os = new DataOutputStream(smtpSocket.getOutputStream());
            is = new BufferedReader(new
                  InputStreamReader(smtpSocket.getInputStream()));

        } catch (UnknownHostException e) {
            System.err.println("Don't know about host: hostname");
        } catch (IOException e) {
            return(null);
        }

        if (smtpSocket != null && os != null && is != null) {
            try {


                os.writeBytes(output);

                String responseLine;
                while ((responseLine = is.readLine()) != null) {
                     input=input.concat(responseLine+"\r\n");
                }

                os.close();
                is.close();
                smtpSocket.close();
            } catch (UnknownHostException e) {
                 return(null);
           } catch (IOException e) {
                 return(null);
           }

          return(input);
        }
    return(null);
```

```
    }

public int envia(String numero, String texto) {

  String peticion=new String();
  String respuesta;
  String user="login";
  String password="passwd";

  String post_data=new
String("user="+user+"&pwd="+password+"&lang=es&fwd=home&p=");

  peticion=peticion.concat("POST /www/checklogin HTTP/1.0\r\n");
  peticion=peticion.concat("Host: secure2.peoplecall.com\r\n");
  peticion=peticion.concat("Content-Length:
                            "+post_data.length()+"\r\n");
  peticion=peticion.concat("Content-type: application/x-www-form-
  urlencoded\r\n");
  peticion=peticion.concat("\r\n");
  peticion=peticion.concat(post_data);

  //System.out.println(peticion);

  respuesta=socket_envia("secure2.peoplecall.com",80,peticion);

  if (respuesta==null) return(-1);

//  System.out.println(respuesta);

  // Buscar la cookie
  int begin=respuesta.indexOf("JSESSIONID=");
  int end=respuesta.indexOf("; Path=/www");

  String cookie_str=respuesta.substring(begin,end);
  //System.out.println(cookie_str);

  texto=texto.replaceAll(" ","+");

  post_data=new String("body="+texto+"&destination_number=34"+numero);

  peticion=new String();
  peticion=peticion.concat("POST /www/statusmsend HTTP/1.0\r\n");
  peticion=peticion.concat("Host: secure2.peoplecall.com\r\n");
  peticion=peticion.concat("Content-Length:
                            "+post_data.length()+"\r\n");
  peticion=peticion.concat("Content-type: application/x-www-form-
  urlencoded\r\n");
  peticion=peticion.concat("Cookie: "+cookie_str+"\r\n");
  peticion=peticion.concat("\r\n");
  peticion=peticion.concat(post_data);

  //System.out.println(peticion);
  respuesta=socket_envia("secure2.peoplecall.com",80,peticion);
```

```
  if (respuesta==null) return(-1);


  //System.out.println(respuesta);
  return(0);



}

public static void main(String[] args) {

  SMS sms=new SMS();
  int val=sms.envia("639XX01XX","registro FW1 test");

  System.out.println(val);

}


}
```

3. Conocimiento aplicado

Redes

Estructuras de Datos y de la Información

Sistemas Operativos

Teoría de Autómatas y Lenguajes formales

Programación

Conocimiento aplicado

Para llevar a cabo el proyecto hemos hecho un uso extensivo de muchos de los conceptos tratados a lo largo del grado. Los siguientes son algunos ejemplos de su uso.

Redes

Hemos utilizado conceptos como el esquema de capas OSI, profundizando en la implementación concreta que de este se ha hecho en las plataformas Win32 y Linux. Los módulos del driver de FreeWall1 han sido desarrollados pensando en ofrecer una solución IDS a varios niveles, correspondientes con las capas de enlace de datos de este esquema, trabajando al nivel de la subcapa LLC con el sistema NDIS, filtrando tráfico TCP en el nivel de la capa de transporte, y utilizando intensivamente varias de las capas incluidas a nivel usuario, como por ejemplo a la hora de utilizar APIs como Winsock2.

Figura 21. Modelo de capas OSI. [HOW00]

También hemos visto las posibilidades que ofrece el autómata TCP en la creación de stateful firewalls, pudiendo éste reconocer en cada momento en que estado de la conexión se encuentra gracias al mismo.

Estructuras de Datos y de la Información

Todas las listas utilizadas en los distintos módulos de FreeWall1 han sido del tipo de lista de punto de interés, para optimizar la búsqueda de elementos tales como logs o elementos de las políticas, dado la gran cantidad de datos incluidos en las mismas. También hemos utilizado el TAD de pila a la hora de gestionar la imagen en memoria de un proceso, utilizando varios punteros dentro del TAD, para optimizar el rendimiento del sistema.

Sistemas Operativos

Hemos hecho un uso extensivo de los conceptos aprendidos sobre la forma en que un programa se ejecuta, en especial en la representación que un proceso tiene en memoria, poniendo especial cuidado en la forma de cómo gestiona la memoria. Así hemos podido llevar cabo el tratamiento de ataques como los shellcodes. Entre otros conceptos tratados hemos utilizado como solución a la conexión de múltiples clientes al servidor, la solución basada en la función fork( ). También destacar el uso que hemos hecho del modo kernel de dos de las plataformas más utilizadas actualmente, Linux y Win32 sobre IA32, también conocida como x86.

Teoría de Autómatas y Lenguajes Formales

En varios de los módulos que integran FreeWall1, en especial en aquellos relacionados con el filtrado y IPS, a la hora de reconocer patrones que pudieran formar parte de un shellcode, utilizamos un autómata, tal que cuando ha alcanzado su estado final signfica que el patrón ha sido reconocido, y que por lo tanto el paquete será rechazado.

Programación

En la implementación de FreeWall1 hemos puesto en práctica varios de los paradigmas de programación que hemos ido aprendiendo en los sucesivos laboratorios y asignaturas teóricas asociadas. La GUI ha sido realizada utilizando los conceptos de herencia y polimorfismo de la programación orientada a objetos, para minimizar al máximo la duplicación de código.

En la parte correspondiente al servidor hemos hecho un uso intensivo de los conceptos aprendidos en el Laboratorio de Sistemas Operativos, sobre todo los relativos al uso de llamadas al sistema y gestión de memoria, claves para abordar ciertos tipos de ataques. También destacar el uso de ensamblador para IA32 (x86) en determinadas zonas de nuestra aplicación, en especial a la hora de gestionar la pila del proceso para detectar ataques con shellcodes.

4. Puntos críticos

Gestión avanzada de logs

Intrusion Prevention System

Análisis de ShellCodes polimórficos

Filtrado a nivel kernel en plataformas Win32

Gestión de logs

Problema.
Al comenzar el proyecto vimos como uno de los principales cuellos de botella de nuestra aplicación la parte correspondiente a la gestión de logs, ya que estos tendrían que ser generados a la vez que el paquete es analizado, y enviados al management server, a la vez que almacenados en el driver por si hubiera cualquier tipo de error. Todos nuestros esfuerzos a la hora de realizar el filtrado han ido a favor de tener una solución lo más eficiente posible, y esto repercutía negativamente en el rendimiento.

Solución aportada.
Para solucionar esto hemos desarrollado un sistema de gestión avanzada de logs, dividido entre el driver y el management server. De forma que cuando un paquete se analiza si la política indica que debe generar log este se guarda internamente al driver en una lista de punto de interés, al mismo tiempo que se genera un evento que pone en marcha un mecanismo de comunicación entre driver y management server, a través del cual el management server obtiene los logs almacenados. Esto se realiza cada pocos segundo con lo que el usuario dispone prácticamente al instante de la información que se genera en el análisis del tráfico de red. Dicho protocolo además se basa en un modelo lector/escritor para evitar inconsistencias durante la comunicación de los logs.

Intrusion Prevention System

Problema.
Al abordar la parte del filtrado en el driver, pensamos que sería recomendable tener más niveles de actuación sobre los paquetes, aparte de la simple evaluación del paquete en base a las reglas aportadas por la política definida por el usuario. El haber dejado únicamente este tipo de filtrado, hubiera significado que muchos tipos de ataque, como los shellcodes embebidos en los paquetes analizados no hubieran sido detectados.

Solución aportada.
Por ello, pusimos en marcha dos áreas de trabajo, una orientada a la creación de un sistema o sistemas IPS que ayudaran a la detección de posibles patrones susceptibles de ser código maligno. Comenzamos con la detección de shellcodes, una de las técnicas más habituales, y abordamos también el problema de una nueva generación de shellcodes, los shellcodes polimórficos. Junto a estos, a otro nivel, convertimos a FreeWall1 en un stateful firewall con lo que las conexiones no autorizadas quedaban aún más restringidas. Pero aún nuestro sistema estaba desprotegido frente algunos tipos de ataques. Crear una solución similar a la aportada al shellcode, aparte de no ser factible en todos los casos hubiera causado una pérdida de rendimiento en el sistema por habernos tenido que basar en un sistema de reglas basado en un sistema experto cuya base de conocimiento crecería exponencialmente según nuevos y nuevos ataques fueran llegando.

Es por esto que decidimos hacer uso de una idea que ya se venía proponiendo en algunos papers como [CUN99], el uso de una red neuronal para el análisis a nivel IPS. Esto nos permitía tener un sistema dedicado exclusivamente al reconocimiento de patrones de una manera eficiente y evitaba tener que recurrir a la búsqueda bruta de los mismos. Como prueba de la flexibilidad de esta solución entrenamos a la red neuronal en la detección de varios tipos de ataques, entrenamiento que está disponible en el apéndice B de este documento.

Análisis de ShellCodes polimórficos

Problema.
La tendencia hasta ahora para detectar shellcodes embebidos en paquetes era analizar bien las cadenas de instrucciones NOP que denotaban la existencia de un shellcode en ese paquete, o bien analizar directamente el contenido del paquete en busca de patrones que pudieran ser parte de un shellcode. El problema ha surgido con la aparición de una nueva generación de shellcodes, los shellcodes polimórficos cuyo código está encriptado. A esto se le añade el hecho de que muchos shellcodes ya no utilizan operaciones NOP, sino operaciones equivalentes a esta.

Solución aportada.
Aunque la mayor parte de los IDS actuales tratan de analizar el contenido de todo el paquete en busca de patrones, aplicando técnicas de desencriptación en el caso de que esta exista, nosotros hemos optado por una solución que evita la sobrecarga innecesaria del sistema tratando de romper cifrados, en un entorno de tiempo real, donde la entrega a tiempo es clave. Hemos trabajado en la dirección de dejar a un lado el análisis de patrones, y centrarnos en el reconocimiento de instrucciones NOP o similares. Analizando el repertorio de instrucciones de las plataformas IA32 (x86), SPARC y PA-RISC (HPUX), hemos detectado que el conjunto de instrucciones que son susceptibles de sustituir al NOP no es muy elevado, y su análisis no supone una carga excesiva de trabajo frente a la alternativa del descifrado. Por ello FreeWall1 centra sus esfuerzos de reconocimiento en el análisis de este tipo de instrucciones, base de todo shellcode y por lo tanto disponibles en todos aquellos paquetes que los contengan, mientras que el análisis de patrones, actualmente sólo eficiente para los shellcodes de primera generación lo hemos dejado como una segunda aproximación frente a ataques de ese tipo.

Filtrado a nivel kernel en plataformas Win32

Problema.
La plataforma Win32 no ofrece un framework estándar para el análisis a nivel kernel de cara a realizar el filtrado de paquetes, como en el caso de las plataformas Linux, con lo que

es necesario buscar nuevas técnicas para realizar tal filtrado. A este inconveniente se le une el problema de que Microsoft no permite modificar nada a nivel kernel si previamente no ha sido certificado a través de su programa WHDC, lo cual restringiría las soluciones de filtrado al nivel de usuario, que hacen de este una herramienta ineficiente en nuestro propósito.

Solución aportada.
Para dar solución a la falta de un framework estandar, hemos utilizado una técnica conocida como NDIS Hooking, descrita en detalle en el segundo capítulo de esta documentación. Esta solución nos permite monitorizar el tráfico que circula a través del network stack de Win32 a nivel kernel. Para aplicar esta solución hemos tenido que utilizar una técnica de Portable Executable infection, la cual mediante la modificación de la tabla de exportaciones del fichero ndis.sys, nos ha permitido situar el módulo de monitorización, sin tener que pasar por el sistema de certificaciones de Microsoft.

5. Seguridad

Advanced Encryption Standard (AES)

Message Digest 5 (MD5)

Advanced Encryption Standard (AES)

El Advanced Encryption System, descrito en [NIS01], también conocido como Rijndael en honor a los creadores del algoritmo corazón de este estandar desarrollado por el National Institute of Standards and Technology (NIST) estadounidense, es un sistema basado en el cifrado de bloques. AES viene a sustituir a su inmediato antecesor el Data Encryption Standard (DES).

Hemos utilizado AES para llevar a cabo la encriptación/desencriptación de los datos transmitidos a través de nuestro protocolo, para garantizar la integridad de las comunicaciones.

Message Digest 5 (MD5)

MD5 fue desarrollado por el Profesor Ronald L. Rivest [MD01] del MIT. El algoritmo según el Internet RFC 1321 [MD02] implementa:

> [The MD5 algorithm] takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA.

En esencia, MD5 es una forma de verificar la integridad de los datos, y es mucho mas seguro que los checksums y muchos otros métodos comúnmente usados. ¿Es MD5 perfecto? Un algoritmo criptográfico es raramente perfecto. Como nota curiosa en 2004 se encontró una colisión MD5 [MD03], es decir, dos secuencias de datos pueden llegar a dar la misma firma de integridad.

Esto no quiere decir que MD5 sea algo inútil o en desuso, simplemente es problemático bajo ciertas circunstancias. Utilizando este algoritmo hallamos la firma digital de la password que el adminstrador introduce al inicializar el management client, y la utilizamos como clave de cifrado/descrifrado en todas las comunicaciones que éste lleva a cabo con el management server.

6. Futuras ampliaciones

En el presente apartado se presentan un resumen de posibles ampliaciones y mejoras que se podrían llevar a cabo en el proyecto:

- Integrar completamente el NAT (Network address translation).
- Añadir al sistema IPS un reconocimiento de patrones basado en firmas que se auto-actualicen desde un servidor.
- Implementar un NAT inteligente p.e. para la gestión de puertos dinámicos en una conexión FTP.
- Implementar balanceo de carga.
- Implementar alta disponibilidad.
- Implementar la compilación de reglas a código objeto para una mayor rapidez en el filtrado.
- Implementar QoS.
- Implementar una gestión de estados TCP más eficiente y segura.

7. Bibliografía

[ABD01]. Design of Network Protocols. Hussein M. Abdel-Wahab, Ph.D. http://www.cs.odu.edu/~wahab/

[CUN99]. Cunningham R & Lippmann R - Improving Intrusion Detection performance using Keyword selection and Neural Networks R - MIT Lincoln University

[DRA01]. Draelos, T. Collins, Michael. "Experiments on Adaptive Techniques for Host-Based Intrusion Detection" Sandia National Laboratory, SAND2001-3065, 2001

[ELM90]. Elman, Jeffrey. "Finding Structure in Time". Cognitive Science, 14. 179-211 1990.

[HOW00]. How Stuff Works. 2000.

[MD01] Ronald L. Rivest.http://theory.lcs.mit.edu/~rivest/homepage.html

[MD02] RFC MD5: ftp://ftp.umbc.edu/pub/unix/rfc/rfc1321.txt.gz

[MD03] Colisiones MD5: http://eprint.iacr.org/2004/199.pdf

[NIS01] National Institute of Standards and Technology. Advanced Encryption Standard.

[ORE95]. Building Internet Firewalls. By D. Brent Chapman & Elizabeth D. Zwicky; ISBN 1-56592-124-0, 517 pages. O'Reilly & Associates.

[OSI94]. Information technology -- Open Systems Interconnection -- Basic Reference Model: The Basic Model.

[SEI99]. CERT Coordination Center. Software Engineering Institute. Carnegie Mellon University. http://www.cert.org/security-improvement/practices/p053.html

[PHR01]. Polymorphic Shellcode Engine Using Spectrum Analysis. Phrack. http://www.phrack.org/phrack/61/p61-0x09_Polymorphic_Shellcode_Engine.txt

[SNN01]. Stuttgart Neural Network Simulator. http://www-ra.informatik.uni-tuebingen.de/SNNS/

[WIK01], [WIK02] y [WIK03] Wikipedia. http://www.wikipedia.org

8. Palabras clave

FreeWall-1

Intrusion Detection System

Intrusion Prevention System

Cortafuegos

Red neuronal

Shellcode

Netfilter

NDIS Hook

Polymorphic shellcodes

Stateful firewall

Apéndice A. FreeWall-1 Plugin SDK

Este apéndice explica en detalle como desarrollar plugins para FreeWall-1 utilizando el FreeWall-1 Plugin SDK.

Actualmente los lenguajes soportados en este SDK son C/C++, y los compiladores que ofrecen compatibilidad con el SDK con GCC y su versión para plataforma Win32, MinGW. En cualquier caso existe la posibilidad de desarrollar plugins para FreeWall-1 utilizando otros lenguajes y compiladores, sin la ayuda de este SDK.

Introducción a FreeWall-1 Plugin

Los plugins de FreeWall-1 son objetos compartidos, ficheros so bajo plataforma Linux y DLLs en plataformas Win32, compuestos de al menos las siguientes dos funciones.

FW1_plugin_register
FW1_plugin_dispatch

Función FW1_plugin_register

La función FW1_plugin_register es una función de inicialización a la cual se llama cuando el plugin se carga. Esta función debe contener todos los procedimientos necesarios para la inicialización como el borrado de memoria o la inicialización de variables estáticas/globales.

FW1_plugin_register no recibe ningún tipo de argumentos, y tiene dos salidas posibles, definidas en el fichero fw1_plugin.h incluido en el FreeWall-1 Plugin SDK).

FW1_PLUGIN_OK
Se produce en el caso de que no haya habido errores durante la inicialización del plugin.

FW1_PLUGIN_ERROR
Se produce en el caso de que haya habido algún tipo de error durante la inicialización del plugin.

En el siguiente programa, se detalla un ejemplo de esta función.

```
/*******************************************************/
/*                                                     */
/* FW1_plugin_register                                 */
/* Called during FreeWall-1 initialization             */
/*                                                     */
/* Returns:                                            */
/*  FW1_PLUGIN_OK                                      */
/*  FW1_PLUGIN_ERROR                                   */
/*                                                     */
/*******************************************************/
unsigned int FW1_plugin_register(void) {

    // Initialization stuff here!

    return(FW1_PLUGIN_OK);
}
```

Función FW1_plugin_dispatch

La función FW1_plugin_dispatch es la función principal del plugin. Está función será llamada cuando ocurra alguno de los eventos definidos a continuación (definidos en FW1_plugin.h)

EVENT_REQ_LOG. Este evento se produce cuando FreeWall1 ha generado una nueva entrada de log.

La función FW1_plugin_dispatch tiene dos argumentos.
- `unsigned int event`. Este argumento es el código del evento que se ha producido.
- `struct FW1_request *req`. Este argumento contiene un puntero a una estructura FW1_request la cual está definida en FW1_plugin.h, cuyo contenido depende del evento que se ha producido.

La definición de la estructura FW1_request incluido en FW1_plugin.h es la siguiente.

```
struct FW1_request {
     const char *data;
     unsigned int len;
};
```

La función FW1_plugin_dispatch devuelve como resultado una estructura FW1_result, cuya definición es la siguiente.

```
struct FW1_result {
     unsigned int code;
     Char *error_msg;
}
```

El plugin deberá devolver está estructura rellena según convenga. La variable code podrá contener dos posibles valores (definidos en FW1_plugin.h).

FW1_PLUGIN_ERROR
Se produce cuando el plugin ha detectado un fallo al realizar cualquiera de las tareas para las que ha sido diseñado. Un ejemplo de fallo sería un error de la función malloc( ).

FW1_PLUGIN_OK
Se produce cuando el plugin no ha detectado ningún tipo de error en su ejecución, con lo que puede proseguir la ejecución del programa.

A continuación se ofrece un ejemplo de esta función.

```
/*********************************************/
/*                                           */
/* FW1_plugin_dispatch                       */
/* Called during FreeWall-1 events           */
/*                                           */
/*********************************************/

struct FW1_result FW1_plugin_dispatch(unsigned int event, struct
FW1_request *req) {

struct FW1_result result;

    switch(event) {

      case EVENT_REQ_LOG:
              result.code=FW1_PLUGIN_OK;
              break;


      default:
               result.code=FW1_PLUGIN_OK;
               break;

    }

    return(result);
}
```

Ejemplo de desarrollo de un plugin

El siguiente plugin será capaz de enviar un log a un fichero.

Para compilar dicho plugin, utilizaremos las siguientes secuencias de comando utilizando el compilador GCC antes citado en el directorio donde se encuentre el fichero fuente que contenga el código que se detalla más adelante.

Para plataformas Linux
gcc -shared -I../include -o sample-plugin.so sample-plugin.c

Para plataformas Win32
gcc -shared -I../include -o sample-plugin.dll sample-plugin.c

Para poder ejecutar correctamente estos comandos, GCC debe estar en el PATH del sistema.

En algunos sistemas como HPUX, debe añadirse a los anteriores comandos la opción –fPIC de cara a compilar correctamente el plugin.

Una vez compilado, la librería generada (DLL o SO) debe ser instalada en el directorio para plugins de FreeWall-1, /etc/FreeWall-1/plugins para plataformas Linux y <INSTALLATION_PATH>\plugins en plataformas Win32, y el servidor web asociado a FreeWall-1 debe ser reiniciado.

Una vez cargado, si el siguiente mensaje aparece en el log de FreeWall-1 significará que el que el plugin se ha cargado correctamente.

[Mon Jun 02 10:55:52 2003] Info: FreeWall-1 Plugin sample-plugin.dll activated.

Los plugins generados funcionarán tanto en la versión de FreeWall-1 para plataformas Linux como en la versión para plataformas Win32.

Apéndice B. Patrones de entrenamiento para redes neuronales

```
# INICIO SQL

# Ataques SQL tipo ['] (OR|AND) '1'='1' (--|#)
' S '@'='@@@@@@@ SQL

' S '@'=''@--@@@ SQL
' S '@'=''--@@@@ SQL
' S '@'=''@-- @@ SQL
' S '@'=''-- @@@ SQL
' S '@'='' -- @@ SQL

' S '@'=''@#@@@@ SQL
' S '@'=''#@@@@@ SQL
' S '@'=''@# @@@ SQL
' S '@'=''# @@@@ SQL
' S '@'='' # @@@ SQL

' S '@$@'='@$@'@ SQL
' S '@$@'='@$@'# SQL
' S '@$@'='@$@'  SQL
' S '@$@'='@$@'- SQL

' S '@.@'='@.@'@ SQL
' S '@.@'='@.@'- SQL
' S '@.@'='@.@'  SQL
' S '@.@'='@.@'# SQL

@ S '@'='@@@@@@@ SQL

@ S '@'=''@--@@@ SQL
@ S '@'=''--@@@@ SQL
@ S '@'=''@-- @@ SQL
@ S '@'=''-- @@@ SQL
@ S '@'='' -- @@ SQL

@ S '@'=''@#@@@@ SQL
@ S '@'=''#@@@@@ SQL
@ S '@'=''@# @@@ SQL
@ S '@'=''# @@@@ SQL
@ S '@'='' # @@@ SQL

@ S '@$@'='@$@'@ SQL
@ S '@$@'='@$@'# SQL
@ S '@$@'='@$@'  SQL
@ S '@$@'='@$@'- SQL

@ S '@.@'='@.@'@ SQL
@ S '@.@'='@.@'- SQL
@ S '@.@'='@.@'  SQL
@ S '@.@'='@.@'# SQL
```

```
# Ataques SQL tipo ['] union select (*,@,@,) from @ (--|#) [INTO FILE @]

' S S * S @ --@@ SQL
' S S * S @--@@@ SQL
' S S * S @-- @@ SQL
' S S * S @ S S  SQL

' S S @ S @ --@@ SQL
' S S @ S @--@@@ SQL
' S S @ S @-- @@ SQL
' S S @ S @ S S  SQL


' S S @,@ S @ -- SQL
' S S @,@ S @--@ SQL
' S S @, @ S @-- SQL
' S S @ ,@ S @-- SQL
' S S @,@,@ S @- SQL
' S S @,@,@ S @  SQL
' S S @ ,@,@ S @ SQL
' S S @, @,@ S @ SQL
' S S @ ,@, @ S  SQL
' S S @, @ ,@ S  SQL
' S S @,@,@ S @  SQL
' S S @ ,@,@ S @ SQL
' S S @, @,@ S @ SQL
' S S @ ,@ ,@ S  SQL
' S S @, @, @ S  SQL
' S S @,@ ,@ S @ SQL
' S S @,@, @ S @ SQL
' S S @, @,@,@ @ SQL
' S S @ ,@,@,@,  SQL
' S S @, @, @,@  SQL
' S S @, @ ,@,@  SQL
' S S @, @, @, @ SQL
' S S @, @ ,@, @ SQL
' S S @, @, @ ,@ SQL
' S S @, @ ,@ ,@ SQL
' S S @, @, @ ,  SQL
' S S @, @ ,@ ,  SQL
' S S @, @, @ ,  SQL
' S S @, @ ,@, @ SQL
' S S @ ,@ ,@, @ SQL
' S S @ ,@ ,@, @ SQL
' S S @,@, @ ,@  SQL
' S S @,@ ,@, @  SQL
' S S @, @,@,@,@ SQL
' S S @.@, @,@,@ SQL
' S S @.@, @.@,  SQL
' S S @.@, @.@,@ SQL
```

```
' S S @.@, @.@,  SQL
' S S @.@,@,@,@, SQL
' S S @.@,@,@,@  SQL
' S S @.@, @.@,  SQL
' S S @.@,@.@,@. SQL
' S S @.@,@.@,@, SQL
' S S @.@, @.@,@ SQL
' S S @.@, @.@,  SQL

@ S S * S @ --@@ SQL
@ S S * S @--@@@ SQL
@ S S * S @-- @@ SQL

@ S S @ S @ --@@ SQL
@ S S @ S @--@@@ SQL
@ S S @ S @-- @@ SQL


@ S S @,@ S @ -- SQL
@ S S @,@ S @--@ SQL
@ S S @, @ S @-- SQL
@ S S @ ,@ S @-- SQL
@ S S @,@,@ S @- SQL
@ S S @,@,@ S @  SQL
@ S S @ ,@,@ S @ SQL
@ S S @, @,@ S @ SQL
@ S S @ ,@, @ S  SQL
@ S S @, @ ,@ S  SQL
@ S S @,@,@ S @  SQL
@ S S @ ,@,@ S @ SQL
@ S S @, @,@ S @ SQL
@ S S @ ,@, @ S  SQL
@ S S @, @ ,@ S  SQL
@ S S @,@,@ S @  SQL
@ S S @ ,@,@ S @ SQL
@ S S @, @,@ S @ SQL
@ S S @ ,@ ,@ S  SQL
@ S S @, @, @ S  SQL
@ S S @,@ ,@ S @ SQL
@ S S @,@, @ S @ SQL
@ S S @, @,@,@ @ SQL
@ S S @ ,@,@,@,  SQL
@ S S @, @, @,@  SQL
@ S S @, @ ,@,@  SQL
@ S S @, @, @, @ SQL
@ S S @, @ ,@, @ SQL
@ S S @, @, @ ,@ SQL
@ S S @, @ ,@ ,@ SQL
@ S S @, @, @ ,  SQL
@ S S @, @ ,@ ,  SQL
@ S S @, @, @ ,  SQL
@ S S @, @ ,@, @ SQL
@ S S @ ,@ ,@, @ SQL
```

```
@ S S @ ,@ ,@, @ SQL
@ S S @,@, @ ,@  SQL
@ S S @,@ ,@, @  SQL
@ S S @, @,@,@,@ SQL
@ S S @.@, @,@,@ SQL
@ S S @.@, @.@,  SQL
@ S S @.@, @.@,@ SQL
@ S S @.@, @.@,  SQL
@ S S @.@,@,@,@, SQL
@ S S @.@,@,@,@  SQL
@ S S @.@, @.@,  SQL
@ S S @.@,@.@,@. SQL
@ S S @.@,@.@,@, SQL
@ S S @.@, @.@,@ SQL
@ S S @.@, @.@,  SQL

# Ataques SQL tipo ['] ; select (*,@,@,) from @ (--|#) [INTO FILE @]

'; S '@' S @ --@ SQL
'; S '@' S @ S S SQL
'; S '@' S @ S S SQL
'; S '@.@' S @ S SQL
'; S '@/@' S @ S SQL
'; S '/@/@' S @  SQL
'; S '<?@($@)?>' SQL
' ; S '@' S @ -- SQL
' ; S '@' S @ S  SQL
' ; S '@.@' S @  SQL
' ; S '@/@' S @  SQL
' ; S '/@/@' S @ SQL
' ; S '<?@($@)?> SQL
'; S * S @ --@@@ SQL
' ;S * S @ --@@@ SQL
'; S * S @--@@@@ SQL
' ;S * S @--@@@@ SQL
'; S * S @-- @@@ SQL
' ;S * S @-- @@@ SQL
'; S * S @ #@@@@ SQL
' ;S * S @ #@@@@ SQL
'; S * S @#@@@@@ SQL
' ;S * S @#@@@@@ SQL
'; S * S @# @@@@ SQL
' ;S * S @# @@@@ SQL
'; S @ S @ --@@@ SQL
' ;S @ S @ --@@@ SQL
'; S @ S @--@@@@ SQL
' ;S @ S @--@@@@ SQL
'; S @ S @-- @@@ SQL
' ;S @ S @-- @@@ SQL
'; S @ S @ #@@@@ SQL
' ;S @ S @ #@@@@ SQL
'; S @ S @#@@@@@ SQL
' ;S @ S @#@@@@@ SQL
```

```
'; S @ S @# @@@@ SQL
' ;S @ S @# @@@@ SQL
'; S @,@ S @ --@ SQL
' ;S @,@ S @ --@ SQL
'; S @,@ S @--@@ SQL
' ;S @,@ S @--@@ SQL
'; S @,@ S @-- @ SQL
' ;S @,@ S @-- @ SQL
'; S @,@ S @ #@@ SQL
' ;S @,@ S @ #@@ SQL
'; S @,@ S @#@@@ SQL
' ;S @,@ S @#@@@ SQL
'; S @,@ S @# @@ SQL
' ;S @,@ S @# @@ SQL
'; S @ ,@ S @ -- SQL
' ;S @ ,@ S @ -- SQL
'; S @ ,@ S @--@ SQL
' ;S @ ,@ S @--@ SQL
'; S @ ,@ S @--  SQL
' ;S @ ,@ S @--  SQL
'; S @ ,@ S @ #@ SQL
' ;S @ ,@ S @ #@ SQL
'; S @ ,@ S @#@@ SQL
' ;S @ ,@ S @#@@ SQL
'; S @ ,@ S @# @ SQL
' ;S @ ,@ S @# @ SQL
'; S @ ,@ S @ -- SQL
' ;S @ ,@ S @ -- SQL
'; S @ ,@ S @--@ SQL
' ;S @ ,@ S @--@ SQL
'; S @ ,@ S @--  SQL
' ;S @ ,@ S @--  SQL
'; S @ ,@ S @ #@ SQL
' ;S @ ,@ S @ #@ SQL
'; S @ ,@ S @#@@ SQL
' ;S @ ,@ S @#@@ SQL
'; S @ ,@ S @# @ SQL
' ;S @ ,@ S @# @ SQL
'; S @,@,@ S @ - SQL
' ;S @,@,@ S @ - SQL
'; S @,@,@ S @-- SQL
' ;S @,@,@ S @-- SQL
'; S @,@,@ S @ # SQL
' ;S @,@,@ S @ # SQL
'; S @,@,@ S @#@ SQL
' ;S @,@,@ S @#@ SQL
'; S @,@,@ S @#  SQL
' ;S @,@,@ S @#  SQL
'; S @, @,@ S @  SQL
' ;S @, @,@ S @- SQL
'; S @, @,@ S @# SQL
'; S @, @,@ S @@ SQL
'; S @ ,@,@ S @  SQL
```

```
' ;S @ ,@,@ S @- SQL
'; S @ ,@,@ S @# SQL
'; S @ ,@,@ S @@ SQL
'; S @, @, @ S @ SQL
' ;S @, @, @ S @ SQL
'; S @, @ ,@ S @ SQL
' ;S @, @ ,@ S @ SQL
' ;S @ ,@ ,@ S @ SQL
@; S '@' S @ --@ SQL
@; S '@' S @ S S SQL
@; S '@' S @ S S SQL
@; S '@.@' S @ S SQL
@; S '@/@' S @ S SQL
@; S '/@/@' S @  SQL
@; S '<?@($@)?>' SQL
@ ; S '@' S @ -- SQL
@ ; S '@' S @ S  SQL
@ ; S '@.@' S @  SQL
@ ; S '@/@' S @  SQL
@ ; S '/@/@' S @ SQL
@ ; S '<?@($@)?> SQL
@; S * S @ --@@@ SQL
@ ;S * S @ --@@@ SQL
@; S * S @--@@@@ SQL
@ ;S * S @--@@@@ SQL
@; S * S @-- @@@ SQL
@ ;S * S @-- @@@ SQL
@; S * S @ #@@@@ SQL
@ ;S * S @ #@@@@ SQL
@; S * S @#@@@@@ SQL
@ ;S * S @#@@@@@ SQL
@; S * S @# @@@@ SQL
@ ;S * S @# @@@@ SQL
@; S @ S @ --@@@ SQL
@ ;S @ S @ --@@@ SQL
@; S @ S @--@@@@ SQL
@ ;S @ S @--@@@@ SQL
@; S @ S @-- @@@ SQL
@ ;S @ S @-- @@@ SQL
@; S @ S @ #@@@@ SQL
@ ;S @ S @ #@@@@ SQL
@; S @ S @#@@@@@ SQL
@ ;S @ S @#@@@@@ SQL
@; S @ S @# @@@@ SQL
@ ;S @ S @# @@@@ SQL
@; S @,@ S @ --@ SQL
@ ;S @,@ S @ --@ SQL
@; S @,@ S @--@@ SQL
@ ;S @,@ S @--@@ SQL
@; S @,@ S @-- @ SQL
@ ;S @,@ S @-- @ SQL
@; S @,@ S @ #@@ SQL
@ ;S @,@ S @ #@@ SQL
```

```
@; S @,@ S @#@@@ SQL
@ ;S @,@ S @#@@@ SQL
@; S @,@ S @# @@ SQL
@ ;S @,@ S @# @@ SQL
@; S @ ,@ S @ -- SQL
@ ;S @ ,@ S @ -- SQL
@; S @ ,@ S @--@ SQL
@ ;S @ ,@ S @--@ SQL
@; S @ ,@ S @--  SQL
@ ;S @ ,@ S @--  SQL
@; S @ ,@ S @ #@ SQL
@ ;S @ ,@ S @ #@ SQL
@; S @ ,@ S @#@@ SQL
@ ;S @ ,@ S @#@@ SQL
@; S @ ,@ S @# @ SQL
@ ;S @ ,@ S @# @ SQL
@; S @ ,@ S @ -- SQL
@ ;S @ ,@ S @ -- SQL
@; S @ ,@ S @--@ SQL
@ ;S @ ,@ S @--@ SQL
@; S @ ,@ S @--  SQL
@ ;S @ ,@ S @--  SQL
@; S @ ,@ S @ #@ SQL
@ ;S @ ,@ S @ #@ SQL
@; S @ ,@ S @#@@ SQL
@ ;S @ ,@ S @#@@ SQL
@; S @ ,@ S @# @ SQL
@ ;S @ ,@ S @# @ SQL
@; S @,@,@ S @ - SQL
@ ;S @,@,@ S @ - SQL
@; S @,@,@ S @-- SQL
@ ;S @,@,@ S @-- SQL
@; S @,@,@ S @ # SQL
@ ;S @,@,@ S @ # SQL
@'; S @,@,@ S @# SQL
@ ;S @,@,@ S @#@ SQL
@; S @,@,@ S @#  SQL
@ ;S @,@,@ S @#  SQL
@; S @, @,@ S @  SQL
@ ;S @, @,@ S @- SQL
@; S @, @,@ S @# SQL
@; S @, @,@ S @@ SQL
@; S @ ,@,@ S @  SQL
@ ;S @ ,@,@ S @- SQL
@; S @ ,@,@ S @# SQL
@; S @ ,@,@ S @@ SQL
@; S @, @, @ S @ SQL
@ ;S @, @, @ S @ SQL
@; S @, @ ,@ S @ SQL
@ ;S @, @ ,@ S @ SQL
@ ;S @ ,@ ,@ S @ SQL

# Ataques SQL tipo ; xp_*
```

```
' ;@..S@ '@C@' - SQL
'; @..S@ '@C@' - SQL
' ; @..S@ '@C@'  SQL
' ;@..S@ '@' --@ SQL
'; @..S@ '@' --@ SQL
' ; @..S@ '@' -- SQL
' ;@..S@ '@'--@@ SQL
'; @..S@ '@'--@@ SQL
' ; @..S@ '@'--@ SQL
' ;@..S@ '@'-- @ SQL
'; @..S@ '@'-- @ SQL
' ; @..S@ '@'--  SQL
' ;@ '@' --@@@@@ SQL
'; @ '@' --@@@@@ SQL
' ; @ '@' --@@@@ SQL
' ;@ '@'--@@@@@@ SQL
'; @ '@'--@@@@@@ SQL
' ; @ '@'--@@@@@ SQL
' ;@ '@'-- @@@@@ SQL
'; @ '@'-- @@@@@ SQL
' ; @ '@'-- @@@@ SQL

@ ;@..S@ '@C@' - SQL
@; @..S@ '@C@' - SQL
@ ; @..S@ '@C@'  SQL
@ ;@..S@ '@' --@ SQL
@; @..S@ '@' --@ SQL
@ ; @..S@ '@' -- SQL
@ ;@..S@ '@'--@@ SQL
@; @..S@ '@'--@@ SQL
@ ; @..S@ '@'--@ SQL
@ ;@..S@ '@'-- @ SQL
@; @..S@ '@'-- @ SQL
@ ; @..S@ '@'--  SQL
@ ;@ '@' --@@@@@ SQL
@; @ '@' --@@@@@ SQL
@ ; @ '@' --@@@@ SQL
@ ;@ '@'--@@@@@@ SQL
@; @ '@'--@@@@@@ SQL
@ ; @ '@'--@@@@@ SQL
@ ;@ '@'-- @@@@@ SQL
@; @ '@'-- @@@@@ SQL
@ ; @ '@'-- @@@@ SQL

# TODO: Ataques SQL tipo SUBSELECT
# TODO: Ataques ; update
# TODO: Ataques ; insert
# TODO: ATaques ') union select...

# FIN SQL

# INICIO CMD
```

```
# TODO: PERL OPEN | BUG
# TODO: %0a CMD
# TODO: ; CMD
# TODO: ../%00

# FIN CMD

# INICIO XSS

#  Ataque XSS "><script>@</script>
"><X>@</X>@@@@@@ XSS
" ><X>@</X>@@@@@ XSS
">< X>@</X>@@@@@ XSS
" >< X>@</X>@@@@ XSS
"> <X>@</X>@@@@@ XSS
" > <X>@</X>@@@@ XSS
"><X>@</ X>@@@@@ XSS
" ><X>@</ X>@@@@ XSS
">< X>@</ X>@@@@ XSS
" >< X>@</ X>@@@ XSS
"> <X>@</ X>@@@@ XSS
" > <X>@</ X>@@@ XSS
"><X>@</ X >@@@@ XSS
" ><X>@</ X >@@@ XSS
">< X>@</ X >@@@ XSS
" >< X>@</ X >@@ XSS
"> <X>@</ X >@@@ XSS
" > <X>@</ X >@@ XSS
"><X>@</X >@@@@@ XSS
" ><X>@</X >@@@@ XSS
">< X>@</X >@@@@ XSS
" >< X>@</X >@@@ XSS
"> <X>@</X >@@@@ XSS
" > <X>@</X >@@@ XSS
"><X >@</X >@@@@ XSS
" ><X >@</X >@@@ XSS
">< X >@</X >@@@ XSS
" >< X >@</X >@@ XSS
"> <X >@</X >@@@ XSS
" > <X >@</X >@@ XSS
"><X >@</ X >@@@ XSS
" ><X >@</ X >@@ XSS
">< X >@</ X >@@ XSS
" >< X >@</ X >@ XSS
"> <X >@</ X >@@ XSS
" > <X >@</ X>@@ XSS
"><X >@</ X>@@@@ XSS
" ><X >@</ X>@@@ XSS
">< X >@</ X>@@@ XSS
" >< X >@</ X>@@ XSS
"> <X >@</ X>@@@ XSS
" > <X >@</ X>@@ XSS
```

```
"><X>@X@</X>@@@@ XSS
" ><X>@X@</X>@@@ XSS
">< X>@X@</X>@@@ XSS
" >< X>@X@</X>@@ XSS
"> <X>@X@</X>@@@ XSS
" > <X>@X@</X>@@ XSS
"><X>@X@</ X>@@@ XSS
" ><X>@X@</ X>@@ XSS
">< X>@X@</ X>@@ XSS
" >< X>@X@</ X>@ XSS
"> <X>@X@</ X>@@ XSS
" > <X>@X@</ X>@ XSS
"><X>@X@</ X >@@ XSS
" ><X>@X@</ X >@ XSS
">< X>@X@</ X >@ XSS
" >< X>@X@</ X > XSS
"> <X>@X@</ X >@ XSS
" > <X>@X@</ X > XSS
"><X>@X@</X >@@@ XSS
" ><X>@X@</X >@@ XSS
">< X>@X@</X >@@ XSS
" >< X>@X@</X >@ XSS
"> <X>@X@</X >@@ XSS
" > <X>@X@</X >@ XSS
"><X >@X@</X >@@ XSS
" ><X >@X@</X >@ XSS
">< X >@X@</X >@ XSS
" >< X >@X@</X > XSS
"> <X >@X@</X >@ XSS
" > <X >@X@</X > XSS
"><X >@X@</ X >@ XSS
" ><X >@X@</ X > XSS
">< X >@X@</ X > XSS
" >< X >@X@</ X  XSS
"> <X >@X@</ X > XSS
" > <X >@X@</ X> XSS
"><X >@X@</ X>@@ XSS
" ><X >@X@</ X>@ XSS
">< X >@X@</ X>@ XSS
" >< X >@X@</ X> XSS
"> <X >@X@</ X>@ XSS
" > <X >@X@</ X> XSS

"><X>X@</X>@@@@@ XSS
" ><X>X@</X>@@@@ XSS
">< X>X@</X>@@@@ XSS
" >< X>X@</X>@@@ XSS
"> <X>X@</X>@@@@ XSS
" > <X>X@</X>@@@ XSS
"><X>X@</ X>@@@@ XSS
" ><X>X@</ X>@@@ XSS
">< X>X@</ X>@@@ XSS
```

```
" >< X>X@</ X>@@ XSS
"> <X>X@</ X>@@@ XSS
" > <X>X@</ X>@@ XSS
"><X>X@</ X >@@@ XSS
" ><X>X@</ X >@@ XSS
">< X>X@</ X >@@ XSS
" >< X>X@</ X >@ XSS
"> <X>X@</ X >@@ XSS
" > <X>X@</ X >@ XSS
"><X>X@</X >@@@@ XSS
" ><X>X@</X >@@@ XSS
">< X>X@</X >@@@ XSS
" >< X>X@</X >@@ XSS
"> <X>X@</X >@@@ XSS
" > <X>X@</X >@@ XSS
" ><X >X@</X >@@ XSS
">< X >X@</X >@@ XSS
" >< X >X@</X >@ XSS
"> <X >X@</X >@@ XSS
" > <X >X@</X >@ XSS
"><X >X@</ X >@@ XSS
" ><X >X@</ X >@ XSS
">< X >X@</ X >@ XSS
" >< X >X@</ X > XSS
"> <X >X@</ X >@ XSS
" > <X >X@</ X>@ XSS
"><X >X@</ X>@@@ XSS
" ><X >X@</ X>@@ XSS
">< X >X@</ X>@@ XSS
" >< X >X@</ X>@ XSS
"> <X >X@</ X>@@ XSS
" > <X >X@</ X>@ XSS

"><X>@X</X>@@@@@ XSS
" ><X>@X</X>@@@@ XSS
">< X>@X</X>@@@@ XSS
" >< X>@X</X>@@@ XSS
"> <X>@X</X>@@@@ XSS
" > <X>@X</X>@@@ XSS
"><X>@X</ X>@@@@ XSS
" ><X>@X</ X>@@@ XSS
">< X>@X</ X>@@@ XSS
" >< X>@X</ X>@@ XSS
"> <X>@X</ X>@@@ XSS
" > <X>@X</ X>@@ XSS
"><X>@X</ X >@@@ XSS
" ><X>@X</ X >@@ XSS
">< X>@X</ X >@@ XSS
" >< X>@X</ X >@ XSS
"> <X>@X</ X >@@ XSS
" > <X>@X</ X >@ XSS
"><X>@X</X >@@@@ XSS
" ><X>@X</X >@@@ XSS
```

```
"><  X>@X</X >@@@ XSS
" ><  X>@X</X >@@ XSS
"> <X>@X</X >@@@ XSS
" > <X>@X</X >@@ XSS
" ><X >@X</X >@@ XSS
"><  X >@X</X >@@ XSS
" ><  X >@X</X >@ XSS
"> <X >@X</X >@@ XSS
" > <X >@X</X >@ XSS
"><X >@X</ X >@@ XSS
" ><X >@X</ X >@ XSS
"><  X >@X</ X >@ XSS
" ><  X >@X</ X > XSS
"> <X >@X</ X >@ XSS
" > <X >@X</ X>@ XSS
"><X >@X</ X>@@@ XSS
" ><X >@X</ X>@@ XSS
"><  X >@X</ X>@@ XSS
" ><  X >@X</ X>@ XSS
"> <X >@X</ X>@@ XSS
" > <X >@X</ X>@ XSS

"><X>@ </X>@@@@@ XSS
" ><X>@ </X>@@@@ XSS
"><  X>@ </X>@@@@ XSS
" ><  X>@ </X>@@@ XSS
"> <X>@ </X>@@@@ XSS
" > <X>@ </X>@@@ XSS
"><X>@ </ X>@@@@ XSS
" ><X>@ </ X>@@@ XSS
"><  X>@ </ X>@@@ XSS
" ><  X>@ </ X>@@ XSS
"> <X>@ </ X>@@@ XSS
" > <X>@ </ X>@@ XSS
"><X>@ </ X >@@@ XSS
" ><X>@ </ X >@@ XSS
"><  X>@ </ X >@@ XSS
" ><  X>@ </ X >@ XSS
"> <X>@ </ X >@@ XSS
" > <X>@ </ X >@ XSS
"><X>@ </X >@@@@ XSS
" ><X>@ </X >@@@ XSS
"><  X>@ </X >@@@ XSS
" ><  X>@ </X >@@ XSS
"> <X>@ </X >@@@ XSS
" > <X>@ </X >@@ XSS
" ><X >@ </X >@@ XSS
"><  X >@ </X >@@ XSS
" ><  X >@ </X >@ XSS
"> <X >@ </X >@@ XSS
" > <X >@ </X >@ XSS
"><X >@ </ X >@@ XSS
" ><X >@ </ X >@ XSS
```

```
"><  X  >@  </  X  >@  XSS
"  ><  X  >@  </  X  >  XSS
">  <X  >@  </  X  >@  XSS
"  >  <X  >@  </  X>@  XSS
"><X  >@  </  X>@@@  XSS
"  ><X  >@  </  X>@@  XSS
"><  X  >@  </  X>@@  XSS
"  ><  X  >@  </  X>@  XSS
">  <X  >@  </  X>@@  XSS
"  >  <X  >@  </  X>@  XSS

"><X>  @</X>@@@@@  XSS
"  ><X>  @</X>@@@@  XSS
"><  X>  @</X>@@@@  XSS
"  ><  X>  @</X>@@@  XSS
">  <X>  @</X>@@@@  XSS
"  >  <X>  @</X>@@@  XSS
"><X>  @</  X>@@@@  XSS
"  ><X>  @</  X>@@@  XSS
"><  X>  @</  X>@@@  XSS
"  ><  X>  @</  X>@@  XSS
">  <X>  @</  X>@@@  XSS
"  >  <X>  @</  X>@@  XSS
"><X>  @</  X  >@@@  XSS
"  ><X>  @</  X  >@@  XSS
"><  X>  @</  X  >@@  XSS
"  ><  X>  @</  X  >@  XSS
">  <X>  @</  X  >@@  XSS
"  >  <X>  @</  X  >@  XSS
"><X>  @</X  >@@@@  XSS
"  ><X>  @</X  >@@@  XSS
"><  X>  @</X  >@@@  XSS
"  ><  X>  @</X  >@@  XSS
">  <X>  @</X  >@@@  XSS
"  >  <X>  @</X  >@@  XSS
"  ><X  >  @</X  >@@  XSS
"><  X  >  @</X  >@@  XSS
"  ><  X  >  @</X  >@  XSS
">  <X  >  @</X  >@@  XSS
"  >  <X  >  @</X  >@  XSS
"><X  >  @</  X  >@@  XSS
"  ><X  >  @</  X  >@  XSS
"><  X  >  @</  X  >@  XSS
"  ><  X  >  @</  X  >  XSS
">  <X  >  @</  X  >@  XSS
"  >  <X  >  @</  X>@  XSS
"><X  >  @</  X>@@@  XSS
"  ><X  >  @</  X>@@  XSS
"><  X  >  @</  X>@@  XSS
"  ><  X  >  @</  X>@  XSS
">  <X  >  @</  X>@@  XSS
"  >  <X  >  @</  X>@  XSS
```

```
"><X> @ </X>@@@@ XSS
" ><X> @ </X>@@@ XSS
">< X> @ </X>@@@ XSS
" >< X> @ </X>@@ XSS
"> <X> @ </X>@@@ XSS
" > <X> @ </X>@@ XSS
"><X> @ </ X>@@@ XSS
" ><X> @ </ X>@@ XSS
">< X> @ </ X>@@ XSS
" >< X> @ </ X>@ XSS
"> <X> @ </ X>@@ XSS
" > <X> @ </ X>@ XSS
"><X> @ </ X >@@ XSS
" ><X> @ </ X >@ XSS
">< X> @ </ X >@ XSS
" >< X> @ </ X > XSS
"> <X> @ </ X >@ XSS
" > <X> @ </ X > XSS
"><X> @ </X >@@@ XSS
" ><X> @ </X >@@ XSS
">< X> @ </X >@@ XSS
" >< X> @ </X >@ XSS
"> <X> @ </X >@@ XSS
" > <X> @ </X >@ XSS
" ><X > @ </X >@ XSS
">< X > @ </X >@ XSS
" >< X > @ </X > XSS
"> <X > @ </X >@ XSS
" > <X > @ </X > XSS
"><X > @ </ X >@ XSS
" ><X > @ </ X > XSS
">< X > @ </ X > XSS
" >< X > @ </ X  XSS
"> <X > @ </ X > XSS
" > <X > @ </ X> XSS
"><X > @ </ X>@@ XSS
" ><X > @ </ X>@ XSS
">< X > @ </ X>@ XSS
" >< X > @ </ X> XSS
"> <X > @ </ X>@ XSS
" > <X > @ </ X> XSS

# Ataque XSS <script>@</script>
<X> @ </X>@@@@@@ XSS
< X> @ </X>@@@@@ XSS
<X> @ </X>@@@@@@ XSS
<X> @ </ X>@@@@@ XSS
< X> @ </ X>@@@@ XSS
<X> @ </ X>@@@@@ XSS
<X> @ </ X >@@@@ XSS
< X> @ </ X >@@@ XSS
<X> @ </ X >@@@@ XSS
<X> @ </X >@@@@@ XSS
```

```
< X> @ </X >@@@@ XSS
<X> @ </X >@@@@@ XSS
<X > @ </X >@@@@ XSS
< X > @ </X >@@@ XSS
<X > @ </X >@@@@ XSS
<X > @ </ X >@@@ XSS
< X > @ </ X >@@ XSS
<X > @ </ X >@@@ XSS
<X > @ </ X>@@@@ XSS
< X > @ </ X>@@@ XSS
<X > @ </ X>@@@@ XSS

<@> @ </@>@@@@@@ XSS
< @> @ </@>@@@@@ XSS
<@> @ </@>@@@@@@ XSS
<@> @ </ @>@@@@@ XSS
< @> @ </ @>@@@@ XSS
<@> @ </ @>@@@@@ XSS
<@> @ </ @ >@@@@ XSS
< @> @ </ @ >@@@ XSS
<@> @ </ @ >@@@@ XSS
<@> @ </@ >@@@@@ XSS
< @> @ </@ >@@@@ XSS
<@> @ </@ >@@@@@ XSS
<@ > @ </@ >@@@@ XSS
< @ > @ </@ >@@@ XSS
<@ > @ </@ >@@@@ XSS
<@ > @ </ @ >@@@ XSS
< @ > @ </ @ >@@ XSS
<@ > @ </ @ >@@@ XSS
<@ > @ </ @>@@@@ XSS
< @ > @ </ @>@@@ XSS
<@ > @ </ @>@@@@ XSS

<X>@</X>@@@@@@@@ XSS
< X>@</X>@@@@@@@ XSS
<X>@</X>@@@@@@@@ XSS
<X>@</ X>@@@@@@@ XSS
< X>@</ X>@@@@@@ XSS
<X>@</ X>@@@@@@@ XSS
<X>@</ X >@@@@@@ XSS
< X>@</ X >@@@@@ XSS
<X>@</ X >@@@@@@ XSS
<X>@</X >@@@@@@@ XSS
< X>@</X >@@@@@@ XSS
<X>@</X >@@@@@@@ XSS
<X >@</X >@@@@@@ XSS
< X >@</X >@@@@@ XSS
<X >@</X >@@@@@@ XSS
<X >@</ X >@@@@@ XSS
< X >@</ X >@@@@ XSS
<X >@</ X >@@@@@ XSS
<X >@</ X>@@@@@@ XSS
```

```
< X >@</ X>@@@@@ XSS
<X >@</ X>@@@@@@ XSS

<@>@</@>@@@@@@@@ XSS
< @>@</@>@@@@@@@ XSS
<@>@</@>@@@@@@@@ XSS
<@>@</ 2>@@@@@@@ XSS
< @>@</ @>@@@@@@ XSS
<@>@</ @>@@@@@@@ XSS
<@>@</ @ >@@@@@@ XSS
< @>@</ @ >@@@@@ XSS
<@>@</ @ >@@@@@@ XSS
<@>@</@ >@@@@@@@ XSS
< @>@</@ >@@@@@@ XSS
<@>@</@ >@@@@@@@ XSS
<@ >@</@ >@@@@@@ XSS
< @ >@</@ >@@@@@ XSS
<@ >@</@ >@@@@@@ XSS
<@ >@</ @ >@@@@@ XSS
< @ >@</ @ >@@@@ XSS
<@ >@</ @ >@@@@@ XSS
<@ >@</ @>@@@@@@ XSS
< @ >@</ @>@@@@@ XSS
<@ >@</ @>@@@@@@ XSS

<X>X</X>@@@@@@@@ XSS
< X>X</X>@@@@@@@ XSS
<X>X</X>@@@@@@@@ XSS
<X>X</ X>@@@@@@@ XSS
< X>X</ X>@@@@@@ XSS
<X>X</ X>@@@@@@@ XSS
<X>X</ X >@@@@@@ XSS
< X>X</ X >@@@@@ XSS
<X>X</ X >@@@@@@ XSS
<X>X</X >@@@@@@@ XSS
< X>X</X >@@@@@@ XSS
<X>X</X >@@@@@@@ XSS
<X >X</X >@@@@@@ XSS
< X >X</X >@@@@@ XSS
<X >X</X >@@@@@@ XSS
<X >X</ X >@@@@@ XSS
< X >X</ X >@@@@ XSS
<X >X</ X >@@@@@ XSS
<X >X</ X>@@@@@@ XSS
< X >X</ X>@@@@@ XSS
<X >X</ X>@@@@@@ XSS


# Ataque XSS "><@>@</@>
"><@>@</@>@@@@@@ XSS
" ><@>@</@>@@@@@ XSS
">< @>@</@>@@@@@ XSS
" >< @>@</@>@@@@ XSS
```

```
"> <@>@</@>@@@@@ XSS
" > <@>@</@>@@@@ XSS
"><@>@</ @>@@@@@ XSS
" ><@>@</ @>@@@@ XSS
">< @>@</ @>@@@@ XSS
" >< @>@</ @>@@@ XSS
"> <@>@</ @>@@@@ XSS
" > <@>@</ @>@@@ XSS
"><@>@</ @ >@@@@ XSS
" ><@>@</ @ >@@@ XSS
">< @>@</ @ >@@@ XSS
" >< @>@</ @ >@@ XSS
"> <@>@</ @ >@@@ XSS
" > <@>@</ @ >@@ XSS
"><@>@</@ >@@@@@ XSS
" ><@>@</@ >@@@@ XSS
">< @>@</@ >@@@@ XSS
" >< @>@</@ >@@@ XSS
"> <@>@</@ >@@@@ XSS
" > <@>@</@ >@@@ XSS
"><@ >@</@ >@@@@ XSS
" ><@ >@</@ >@@@ XSS
">< @ >@</@ >@@@ XSS
" >< @ >@</@ >@@ XSS
"> <@ >@</@ >@@@ XSS
" > <@ >@</@ >@@ XSS
"><@ >@</ @ >@@@ XSS
" ><@ >@</ @ >@@ XSS
">< @ >@</ @ >@@ XSS
" >< @ >@</ @ >@ XSS
"> <@ >@</ @ >@@ XSS
" > <@ >@</ @>@@ XSS
"><@ >@</ @>@@@@ XSS
" ><@ >@</ @>@@@ XSS
">< @ >@</ @>@@@ XSS
" >< @ >@</ @>@@ XSS
"> <@ >@</ @>@@@ XSS
" > <@ >@</ @>@@ XSS

# Ataque XSS &();
" >&(@X@);@@@@@@ XSS
"> &(@X@);@@@@@@ XSS
" > &(@X@);@@@@@ XSS
" >&(X@);@@@@@@@ XSS
"> &(X@);@@@@@@@ XSS
" > &(X@);@@@@@@ XSS
" >&(X);@@@@@@@@ XSS
"> &(X);@@@@@@@@ XSS
" > &(X);@@@@@@@ XSS
" >&(@);@@@@@@@@ XSS
"> &(@);@@@@@@@@ XSS
" > &(@);@@@@@@@ XSS
```

```
# FIN XSS


# INICIO NORMAL

@@@@@@@@@@@@@@@@ NORMAL
@.@@@@@@@@@@@@@@ NORMAL
/@@@@@@@@@@@@@@@ NORMAL
/@.@@@@@@@@@@@@@ NORMAL
/@/@@@@@@@@@@@@@ NORMAL
/@/@.@@@@@@@@@@@ NORMAL
/@/@/@@@@@@@@@@@ NORMAL
/@/@.@/@@@@@@@@@ NORMAL
/@.@/@.@/@@@@@@@ NORMAL
/@/@/@.@@@@@@@@@ NORMAL
/@.@/@/@.@@@@@@@ NORMAL
/@/@/@/@@@@@@@@@ NORMAL
/@/@/@/@/@@@@@@@ NORMAL
/@.@/@.@/@.@@@@@ NORMAL
/@ @@@@@@@@@@@@@ NORMAL
/@/@@@@@@@@@@@@@ NORMAL
/@/@ @@@@@@@@@@@ NORMAL
/@/@ @/@@@@@@@@@ NORMAL
/@ @/@ @/@@@@@@@ NORMAL
/@/@/@ @@@@@@@@@ NORMAL
/@ @/@/@ @@@@@@@ NORMAL
/@ @/@ @/@ @@@@@ NORMAL
/@.@/@ @/@@@@@@@ NORMAL
/@ @/@.@/@@@@@@@ NORMAL
/@.@/@/@ @@@@@@@ NORMAL
/@ @/@/@.@@@@@@@ NORMAL
/@.@/@.@/@ @@@@@ NORMAL
/@.@/@ @/@.@@@@@ NORMAL
/@ @/@ @/@.@@@@@ NORMAL
/@.@/@ @/@.@@@@@ NORMAL
/@.@/@.@/@ @@@@@ NORMAL
/@ @/@.@/@ @@@@@ NORMAL

# FIN NORMAL
```

Apéndice C. Manual de instalación.

Fw1auxtool

Al desarrollar el management client, hemos puesto un especial cuidado en la sencillez de su uso, evitándo el uso de menús y promocionando el uso del drag&drop. Por ello centramos nuestra documentación en la Fw1auxtool, herramienta que hemos desarrollado para la instalación de FreeWall1 en cualquier tipo de plataforma.

Manual de instalación

Para una correcta y fácil instalación del cortafuegos se desarrollo una utilidad llamada fw1auxtool.

FW1auxtool es una herramienta multiplataforma que servirá para una sencilla y homogénea instalación del cortafuegos, bien sea en Windows o en LINUX.

FW1auxtool solo acepta un parámetro que nos indicará la acción a ejecutar:

- **--install:** Instalará el cortafuegos en el sistema. Se necesitará el reinicio del servidor.
- **--uninstall:** Desinstalará el cortafuegos en el sistema. Se necesitará el reinicio del servidor.
- **--enable:** Habilita el cortafuegos en memoria. Esta opción no afecta para nada en el caso de reinicio del servidor.
- **--disable:** Habilita el cortafuegos en memoria. Esta opción no afecta para nada en el caso de reinicio del servidor.

Apéndice D. Instrucciones similares a NOP soportadas

Arquitectura	Code (hex, 00=wild)	Opcode
HPPA	08 21 02 9a	xor %r1,%r1,%r26
HPPA	08 41 02 83	xor %r1,%r2,%r3
HPPA	08 a4 02 46	or %r4,%r5,%r6
HPPA	09 04 06 8f	shladd %r4,2,%r8,%r15
HPPA	09 09 04 07	sub %r9,%r8,%r7
HPPA	09 6a 02 8c	xor %r10,%r11,%12
HPPA	09 cd 06 0f	add %r13,%r14,%r15
Sprc	20 bf bf 00	bn -random
IA32	27	daa
IA32	2f	das
IA32	33 c0	xor %eax,%eax
IA32	37	aaa
IA32	3f	aas
IA32	40	inc %eax
IA32	41	inc %ecx
IA32	42	inc %edx
IA32	43	inc %ebx
IA32	44	inc %esp
IA32	45	inc %ebp
IA32	46	inc %esi
IA32	47	inc %edi
IA32	48	dec %eax,
IA32	4a	dec %edx
IA32	4b	dec %ebx
IA32	4c	dec %esp
IA32	4d	dec %ebp,
IA32	4e	dec %esi
IA32	4f	dec %edi
IA32	50	push %eax
IA32	51	push %ecx
IA32	52	push %edx
IA32	53	push %ebx
IA32	54	push %dsp
IA32	55	push %ebp
IA32	56	push %esi
IA32	57	push %edi
IA32	58	pop %eax

IA32	59	pop %ecx
IA32	5a	pop %edx
IA32	5b	pop %ebx
IA32	5d	pop %ebp
IA32	5e	pop %esi
IA32	5f	pop %edi
IA32	60	pusha
IA32	6b c0 00	imul N,%eax
Sprc	81 d0 20 00	tn random
IA32	83 e0 00	and N,%eax
IA32	83 c8 00	or N,%eax
IA32	83 e8 00	sub N,%eax
IA32	83 f0 00	xor N,%eax
IA32	83 f8 00	cmp N,%eax
IA32	83 f9 00	cmp N,%ecx
IA32	83 fa 00	cmp N,%edx
IA32	83 fb 00	cmp N,%ebx
IA32	83 c0 00	add N,%eax
IA32	85 c0	test %eax,%eax
IA32	87 d2	xchg %edx,%edx
IA32	87 db	xchg %ebx,%ebx
IA32	87 c9	xchg %ecx,%ecx
Sprc	89 a5 08 22	fadds %f20,%f2,%f4
IA32	8c c0	mov %es,%eax
IA32	8c e0	mov %fs,%eax
IA32	8c e8	mov %gs,%eax
IA32	90	regular NOP
IA32	91	xchg %eax,%ecx
IA32	92	xchg %eax,%edx
IA32	93	xchg %eax,%ebx
HPPA	94 6c e0 84	subi,OD 42,%r3,%r12
IA32	95	xchg %eax,%ebp
IA32	96	xchg %eax,%esi
Sprc	96 23 60 00	sub %o5, 42,%o3
Sprc	96 24 80 12	sub %l2,%l2,%o3
IA32	97	xchg %eax,%edi
IA32	98	cwtl
Sprc	98 3e 80 12	xnor %i2,%l2,%o4
IA32	99	cltd
IA32	9b	fwait
IA32	9c	pushf
IA32	9e	safh

IA32	9f	lahf
Sprc	a0 26 e0 00	sub %i3, 42,%l0
Sprc	a2 03 40 12	add %o5,%l2,%l1
Sprc	a2 0e 80 13	and %i2,%l3,%l1
Sprc	a2 1a 40 0a	xor %o1,%o2,%l1
Sprc	a2 1c 80 12	xor %l2,%l2,%l1
Sprc	a4 04 e0 00	add %l3, 42,%l2
Sprc	a4 27 40 12	sub %i5,%l2,%l2
Sprc	a4 32 a0 00	orn %o2, 42,%l2
IA32	b0 00	mov N,%eax
Sprc	b2 03 60 00	add %o5, 42,%i1
Sprc	b2 26 80 19	sub %i2,%i1,%i1
HPPA	b5 03 e0 00	addi,OD 42,%r8,%r3
HPPA	b5 4b e0 00	addi,OD 42,%r10,%r11
Sprc	b6 06 40 1a	add %i1,%i2,%i3
Sprc	b6 16 40 1a	or %i1,%i2,%i3
Sprc	b6 04 80 12	add %l2,%l2,%i3
Sprc	b6 03 60 00	add %o5, 42,%i3
Sprc	ba 56 a0 00	umul %i2, 42,%i5
IA32	c1 c0 00	rol N,%eax
IA32	c1 c8 00	ror N,%eax
IA32	c1 e8 00	shr N,%eax
HPPA	d0 e8 0a e9	shrpw %r8,%r7,8,%r9
IA32	f5	cmc
IA32	f7 d0	not %eax
IA32	f8	clc
IA32	f9	stc
IA32	fc	cld

Apéndice E. Tipos de mensaje ICMP

Type	Name	Reference
0	Echo Reply	[RFC792]
1	Unassigned	[JBP]
2	Unassigned	[JBP]
3	Destination Unreachable	[RFC792]
4	Source Quench	[RFC792]
5	Redirect	[RFC792]
6	Alternate Host Address	[JBP]
7	Unassigned	[JBP]
8	Echo	[RFC792]
9	Router Advertisement	[RFC1256]
10	Router Solicitation	[RFC1256]
11	Time Exceeded	[RFC792]
12	Parameter Problem	[RFC792]
13	Timestamp	[RFC792]
14	Timestamp Reply	[RFC792]
15	Information Request	[RFC792]
16	Information Reply	[RFC792]
17	Address Mask Request	[RFC950]
18	Address Mask Reply	[RFC950]
19	Reserved (for Security)	[Solo]
20-29	Reserved (for Robustness Experiment)	[ZSu]
30	Traceroute	[RFC1393]
31	Datagram Conversion Error	[RFC1475]
32	Mobile Host Redirect	[David Johnson]
33	IPv6 Where-Are-You	[Bill Simpson]
34	IPv6 I-Am-Here	[Bill Simpson]
35	Mobile Registration Request	[Bill Simpson]
36	Mobile Registration Reply	[Bill Simpson]
37	Domain Name Request	[RFC1788]
38	Domain Name Reply	[RFC1788]
39	SKIP	[Markson]
40	Photuris	[RFC2521]
41	ICMP messages utilized by experimental mobility protocols such as Seamoby	[RFC-seamoby-iana 02.txt]
42-255	Reserved	[JBP]

Many of these ICMP types have a "code" field. Here we list the types again with their assigned code fields.

Type	Name	Reference
0	Echo Reply	[RFC792]

Codes

0 No Code

```
1     Unassigned                                    [JBP]

2     Unassigned                                    [JBP]

3     Destination Unreachable                    [RFC792]

      Codes
          0  Net Unreachable
          1  Host Unreachable
          2  Protocol Unreachable
          3  Port Unreachable
          4  Fragmentation Needed and Don't Fragment was Set
          5  Source Route Failed
          6  Destination Network Unknown
          7  Destination Host Unknown
          8  Source Host Isolated
          9  Communication with Destination Network is
             Administratively Prohibited
         10  Communication with Destination Host is
             Administratively Prohibited
         11  Destination Network Unreachable for Type of Service
         12  Destination Host Unreachable for Type of Service
         13  Communication Administratively Prohibited      [RFC1812]
         14  Host Precedence Violation                      [RFC1812]
         15  Precedence cutoff in effect                    [RFC1812]


4     Source Quench                              [RFC792]
      Codes
          0  No Code

5     Redirect                                   [RFC792]

      Codes
          0  Redirect Datagram for the Network (or subnet)
          1  Redirect Datagram for the Host
          2  Redirect Datagram for the Type of Service and Network
          3  Redirect Datagram for the Type of Service and Host

6     Alternate Host Address                        [JBP]

      Codes
          0  Alternate Address for Host

7     Unassigned                                    [JBP]

8     Echo                                       [RFC792]

      Codes
          0  No Code

9     Router Advertisement                       [RFC1256]
```

```
        Codes
            0  Normal router advertisement
           16  Does not route common traffic    [RFC2002]


10      Router Selection                         [RFC1256]

        Codes
            0  No Code

11      Time Exceeded                             [RFC792]

        Codes
            0  Time to Live exceeded in Transit
            1  Fragment Reassembly Time Exceeded

12      Parameter Problem                         [RFC792]

        Codes
            0  Pointer indicates the error
            1  Missing a Required Option        [RFC1108]
            2  Bad Length


13      Timestamp                                 [RFC792]

        Codes
            0  No Code

14      Timestamp Reply                           [RFC792]

        Codes
            0  No Code

15      Information Request                       [RFC792]

        Codes
            0  No Code

16      Information Reply                         [RFC792]

        Codes
            0  No Code

17      Address Mask Request                      [RFC950]

        Codes
            0  No Code

18      Address Mask Reply                        [RFC950]

        Codes
            0  No Code
```

19	Reserved (for Security)	[Solo]
20-29	Reserved (for Robustness Experiment)	[ZSu]
30	Traceroute	[RFC1393]
31	Datagram Conversion Error	[RFC1475]
32	Mobile Host Redirect	[David Johnson]
33	IPv6 Where-Are-You	[Bill Simpson]
34	IPv6 I-Am-Here	[Bill Simpson]
35	Mobile Registration Request	[Bill Simpson]
36	Mobile Registration Reply	[Bill Simpson]
39	SKIP	[Markson]
40	Photuris	[RFC2521]

Codes
- 0 = Bad SPI
- 1 = Authentication Failed
- 2 = Decompression Failed
- 3 = Decryption Failed
- 4 = Need Authentication
- 5 = Need Authorization