Universidad Autonoma Metropolitana Unidad
Azcapotzalco

Division de Ciencias Básicas e Ingeniería

Proyecto Terminal en Ingeniería en Computación

Sistema de Filtrado para redes de computadoras con herramientas Gráficas en Linux

Proyecto que presenta:
Misael Azamar Ramos

para obtener el título de:
Ingeniero en Computación

Director de Proyecto:
M. en C. Oscar Alvarado Nava

México, D.F. **Enero de 2010**

Resumen

En la actualidad para la administración de red de computadoras, en las que pueden coexistir diferentes plataformas, distintos sistemas operativos; se requiere contar con mínimo con conocimientos del funcionamiento de una red de datos, asi como la administración de sistemas operativos. La justificación en particular que se tiene con el desarrollo de este proyecto consiste en facilitar a una persona sin conocimentos avanzados en redes el control sobre una red, con ello evitamos que dicha persona tenga actividades extra, como acudir a un administrador de redes o algún conocedor de sistemas, ya que se pretende que el uso de la aplicación sea más fácil y flexible. Para ello se utilizarán tecnicas de *NAT*[1], ya es posible administrar redes a base de aplicación de reglas sobre un servidor.

NAT es una técnica que permite acceder a la internet traduciendo las direcciones privadas en direcciones IP registradas. Incrementa la seguridad y la privacidad de una red local al traducir el direccionamiento interno a una red externa.

NAT tiene varias formas de trabajar según los requisitos del administrador de red; cualquiera de estas es importante al momento de controlar el tráfico hacia una red exterior como lo es la Internet. En los sistemas operativos Linux existe la herramienta *Iptables*, el *firewall* por defecto y que realiza dos funciones; filtrado y traducción de direcciones de red (*NAT*). Es un componente que funciona en el espacio de usuario y permite definir reglas para el filtrado y modificación de paquetes TCP/IP que pasen por las interfaces de red de un equipo, en este caso por el servidor. Existen distintos tipos de NAT empleados por *Iptables* dependiendo si los paquetes ingresan o salen de la red; estas variantes son de origen(*Source NAT*) y destino(*Destination NAT*).

El tipo origen o *Source NAT* cambia las direcciones IP de origen y es el medio mas utilizado para redes locales conectadas a internet a travéz de un servidor NAT.

El tipo destino o *Destination NAT* es usado cuando tenemos algún servidor interno detras del Servidor principal.

Por medio del funcionamiento de *Iptables* y las técnicas existentes de tradución de direcciones, es posible desarrollar aplicaciones gráficas capaces de manipular dichos procesos y a su vez establecer transparencia de funcionamiento al administrador evitando el uso de comandos complejos y difíciles de memorizar.

Inicialmente se hace un análisis de los requerimientos de un administrador de red para así establecer la infraestructura de las reglas detras de la interfaz gráfica. Con las

[1]Network Address Translation, Traducción de direcciones de red

reglas ya establecidas es posible modular el funcionamiento interno de la aplicación de acuerdo a cada decisión que será tomada desde la interfaz en tiempo real, así tambien desarrollar cada interfaz de la aplicación.Finalmente integrar cada una de los módulos con las interfaces mediante algúnas técnicas de procesamiento concurrente.

Con el presente documento se presenta el desarrollo de una aplicación gráfica que integra la funcionalidad de *Iptables* así como técnicas de *NAT* para el filtrado de paquetes en una red de computadoras.

Agradecimientos

- A mi Asesor de Proyecto M. en C. Oscar Alvardo Nava por el constante apoyo para la realización de éste proyecto.
- A la Coordinación de la carrera de Ingeniería en Computación de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco.
- A la División de Ciencias Básicas e Ingeniería de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco.

Dedicatoria

Microprogramar un algoritmo ineficiente no lo convierte en eficiente

. «Ley de Rauscher» .

La Fé por los sueños es un gran don, pero mis sueños fueron más grandes que la fé

. «Misael Azamar Ramos» .

Dedico éste Proyecto...

A mi familia, especialmente a mis padres Pedro Pérez Moya y Ana laura Azamar Ramos quienes día con día me apoyaron tanto moralmente como en el sustento incondicional para la realización de este sueño

A mi amor Susana Flores Romero, quien durante mi estancia universitaria nunca dejo de creer en mí y motivó gran parte de mi inquietud por este logro.

A quienes creyeron y a los que nunca creyeron en mi, por que los hechos son más grandes que las palabras

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Objetivo

Desarrollar una sistema de filtrado de paquetes utilizando *Iptables* [2] con visualización gráfica, que facilite a una persona implantar reglas de filtrado de paquetes en una red de computadoras. En base a este objetivo general se tienen en particular los siguientes objetivos.

- **Recopilar Funcionalidades sobre redes internas** En el objetivo general se requiere en la administración gráfica de una red. Para ello es necesario entender las necesidades de un administrador.
- **Modelar la estructura del sistema** El sistema está estrucurado por varias funciones y con ello es indispensable visualizar al sistema desde un contexto de arquitectura.
- **Desarrollar e implementar** El proceso de desarrollo involucra análisis y estructura; con ello se forman las bases para el proceso de programación del sistema así como su implementación.

1.2. Organización del proyecto

El proyecto consta de 6 capítulos organizados de la siguiente manera: En el capítulo 2 se presenta una introducción a lo que es un *Firewall*. En el mismo se describe a *Iptables* así como el uso de dicha herramienta funcionando como el mismo *Firewall* predeterminado de Linux.En este mismo se describe la metodología de traducción de direcciones de red usando *Iptables* y las reglas de filtrado mas usuales en el contexto de redes internas.

En el capítulo 3 se explica la segunda herramienta usada en el desarrollo de la aplicación. Se describe la utilidad de los *Sniffers* en Linux así como las herramientas de monitoreo a utilizar durante la implementación del sistema.

En el capítulo 4 se describen los aspectos en el contexto de niveles de diseño. En el mismo se define un marco estructural mediante diagramas, representando los módulos de la aplicación.

El capítulo 5 inicia con la descripción de los métodos y algoritmos implementados para la ejecución de las reglas de *Iptables*. Dichas acciones son analizadas para su integración con la interfaz de usuario. En este mismo capítulo se describe la subfunción del monitor en tiempo real así como las metodologías para lectura y retención de paquetes usando *tcpdump* desde la misma aplicación. En este mismo capítulo de muestra el modelo general de la apicación; principalmente se describe la arquitectura general del sistema con todos los componentes encapsulados.

En el capítulo 6 se muestran los resultados de las pruebas obtenidas entre las dos funcionalidades: Filtrado y Monitoreo en tiempo real desde la aplicación. Finalmente las conclusiones se presentan en el capítulo 7.

Capítulo 2

Firewalls

2.1. Introducción

Un firewall es un filtro que controla las comunicaciones entre redes; estos filtros permiten o deniegan el acceso a un dispositivo de red. El firewall examina el tipo de servicio utilizado en una computadora, (por ejemplo Internet, Correo electrónico, Web, etc), y dependiendo de ello, el firewall decide si permite el uso del servicio o no.

Un firewall puede ser un dispositivo software o bien puede ser implementado en hardware. El uso firewalls en hardware es más rápido y eficiente en comparación con implementados en software, pero a consecuencia de ello resultan ser mas costosos. Los firewalls implementados en software utilizan mayores recursos de hardware lo que implica mayor procesamiento y menor rapidez. El hecho de usar firewalls mediante software implica establecer criterios de filtrado a nivel de programas de computadora, incluso se pueden encontrar computadoras muy potentes con softwares especificos que lo único que hacen es monitorizar las comunicaciones entre redes así como establecer criterios de filtrado con funcionamiento mas transparente hacia el usuario de la red.

Tambien pueden ser implementados en combinación de hardware y software. La mayor utilidad de firewalls en software se refleja en la facilidad de uso, ya que es más fácil manipular e interpretar la información de los paquetes, que a nivel de hardware.

Un firewall trabaja de la siguiente manera:

Los mensajes que ingresen o salgan de la red interna pasan a travez de un firewall instalado entre la red externa y la red interna, el firewall examina cada uno de los paquetes y bloquea aquellos paquetes que no cumplen con una regla o criterio de seguridad. La siguiente figura muestra un esquema de ésta aplicación.

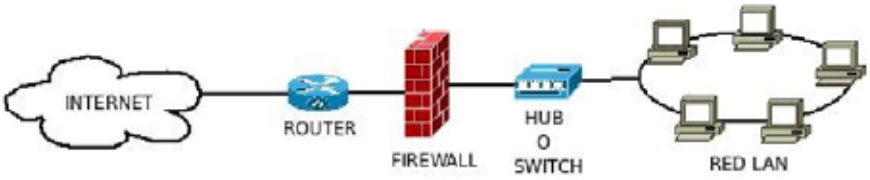

Figura 2.1: Esquema de un *Firewall* entre dos redes, en este caso una red LAN[1] e Internet

2.2. Iptables

Este proyecto se enfoca al desarrollo de un firewall sobre Linux implementado en software a nivel de Interfaces Gráficas capaz de controlar tráfico proveniente de redes internas y externas. Para ello se usa a *Iptables*[3], el firewall de linux por defecto.

Iptables es un módulo en el kernel que se estructura en normas de seguridad sobre tablas y es el componente más sobresaliente construido a partir de un proyecto llamado *Netfilter.Iptables* es el firewall de Linux por defecto que no solamente filtra paquetes, si no que también implementa *NAT*(Traducción de direcciones de Red); es decir, aparte de filtrar puede decidir qué hacer con tales paquetes en base a su dirección de origen o destino.

El proyecto *Netfilter* creado por Rusty Russell en 1998 inició con *Ipchains* el predecesor del actual *Iptables*. Ambos en su etapa de desarrollo y mantenimiento fueron licenciados bajo la licencia GPL(GNU General Public License). *Iptables* fue integrado al kernel de Linux 2.3 en marzo del año 2000.

Antes de *Iptables* el software utilizado era *Ipchains* sobre el kernel de Linux 2.2 e *Ipfwadm*. *Ipchains* trabaja con en concepto de cadenas de reglas, despues de el, *iptables* extendió esto al manejo de tablas, donde ya se decidía entre filtrar o bien procesar paquetes.

Iptables agrupa los criterios de seguridad en cadenas; cada cadena es una lista ordenada de reglas. Estas cadenas son ordenadas en tablas y cada tabla esta asociada a un tipo diferente de procesamiento.Existen tres cadenas básicas Entrada *(INPUT)*, Salida*(OUTPUT)* y Reenvío*(FORWARD)*.

Las tablas incorporadas son principalmente dos: Tabla de filtrado(*Filter table*) y tabla de tradución de direcciones(*Nat table*). La tabla de filtrado se encarga de permitir o bloquear un paquete ya sea de entrada o salida. La tabla de traducción de direcciones se enfoca al procesamiento de los paquetes y trabaja con cadenas de preruteo*(PREROUTING)* (principalmente para direcciones de destino), cadenas de postruteo*POSTROUTING*(para direcciones de origen), así como tambien las cadenas de entrada y salida.

En la siguiente figura muestra la estructura de *iptables* en el proceso de filtrado de paquetes.

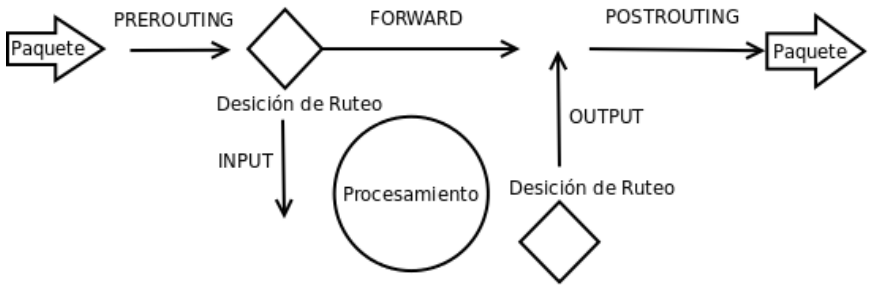

Figura 2.2: Figura que muestra la trayectoria de un paquete a travez del *Firewall Iptables*

El seguimiento de un paquete inicia desde que entra a una interfaz de red(A). EL paquete es analizado y de acuerdo los criterios de *Iptables* se decide qué hacer con tal paquete. Las opciónes pueden ser; entrar mediante la cadena *INPUT* o bién pasar (*FORWARD*).Despues se decide que hacer con el paquete que llega a la interfaz de red(B) desde la cadena *OUTPUT* o bien por paso directo. Dichas decisiones pueden ser rechazar(*REJECT*), aceptar(*ACCEPT*), denegar(*DROP*) o enmascarar(MASQUERADE).

2.2.1. Reglas con Iptables

La estructura general de un comando de Iptables en la consola de Linux y su funcionamiento es de la siguiente forma:

iptables -t [tabla] [operación] [cadena] [descripción de la coincidencia] -j [acción]

La opción *[tabla]*[4] permite al usuario seleccionar una tabla diferente a la tabla predeterminada filter a usar con el comando. Para indicar a *Iptables* qué hacer con los paquetes al intentar entrar o salir de una computadora, se debe crear una coincidencia lo más precisa posible. La idea es que la coincidencia sea inequívoca, tanto como para quien creó la regla (usuario o administrador) como para el kernel.Por último [acción] será para definir las acciones descritas en la figura 2.

Los comandos de *Iptables*[5] varian de acuerdo a los criterios del usuario; por ejemplo se tienen los siguientes casos en particular con el comando correspondiente.

- Bloquear acceso total a internet

  `iptables - t nat -I POSTROUTING -s 192.168.0.3 -o eth0 -j DROP`

- Bloquear acceso a una o mas páginas de Internet

  `iptables -t filter -I FORWARD -s 192.168.0.2 -d www.yahoo.com.mx -p tcp –dport 80 -j DROP`

- Bloquear messenger

  `iptables -t mangle -I PREROUTING -s 172.16.24.1 -p tcp –dport 1863 -j DROP`

- Bloquear comunicaciones por ping

  `iptables -I INPUT -p icmp -s 192.168.0.2 -j DROP`

- Bloquear accesos por protocolo SSH de una maquina a otra

  `iptables -t filter -I FORWARD -s 192.168.0.2 -p tcp –dport 22 -j DROP`

- Bloquear paquetes provenientes del puerto 3306

  `iptables -t filter -I FORWARD -s 192.168.0.2 -p tcp –dport 3306 -j DROP`

- Bloquear accesos por FTP

  `iptables -t filter -I FORWARD -s 192.168.0.2 -p tcp –dport 21 -j DROP`

En este proyecto también se usará NAT[6], puesto que está orientado a direcciones de red, además es control principal dentro de iptables para el redireccionamiento de paquetes entre distintas interfaces. Así como NAT es un método de traducciones de red, *Iptables* tiene la tabla *Nat* destinada para ello. Las funcionalidades de la traducción de direcciones tambien son necesarias cuando en nuestra red interna existe un rango de computadoras mayor al número de computadoras limitadas por nuestro Proveedor de Servicios de internet*(ISP)*, así pues la IP pública de nuestro servidor será la misma para la red interna. NAT es muy utilizado en empresas y redes caseras, ya que basta tener una sola dirección IP pública para poder conectar una multitud de computadoras.

Capítulo 3

Sniffers

3.1. Introducción

Un sniffer es un programa que sirve para monitorear y analizar el tráfico en una red de computadoras y problemas que pudieran surgir en esta.

Un sniffer puede ser utilizado para obtener, lícitamente o no, los datos que son transmitidos en la red. Una interfaz lee cada paquete de datos que pasa por esta; el sistema determina de manera intencional el destino del paquete dentro de la red. Una máquina y un sniffer, pueden leer los datos dentro del paquete así como la dirección de origen y destino.

Un *Sniffer* trabaja capturando el tráfico(en tiempo real) que pasa por las interfaces de red, tal información no es directamente procesada si no solamente interpretada y mostrada en la salida estandar. Existen *Sniffers* orientados al procesamiento de esta información, como *Wireshark*, tales programas capturan los datos, los procesan y son mostrados al usuario como cadenas de texto segmentadas.

Los segmentos de un paquete monitoreado son mostrados según el protocolo, para ello es posible definir al *Sniffer* qué tipo de paquetes especificos se requieren, por ejemplo, paquetes transmitiendose a travez del protocolo icmp(en este caso petición ping), paquetes con destino a web (protocolo TCP/IP), etc.

Existen variedad *Sniffers* comerciales tanto como de código abierto. En este proyecto se integra la utilidad de monitoreo y se utilizará el *Sniffer Tcpdump*.

3.2. Tcpdump

Tcpdump es una aplicación de consola que nos permite analizar el tráfico de la red. Con la misma podremos ver los paquetes que se envían desde y hacia una interfaz de red. En UNIX y otros sistemas operativos, es necesario tener los privilegios del root para utilizar tcpdump.

La estructura de un comando tcpdump varia en función de los paquetes deseados a monitorear. Tambien es posible definir la interfaz sobre la cual se desea ver el paquete

así como paquetes provenientes de alguna IP/Red específica o bién paquetes dirigidos hacia alguna ruta o red en especial.

Tcpdump permite examinar todas las conversaciones, incluyendo mensajes provenientes de una subred . Mientras que sus capacidades en detección de errores están principalmente a nivel de capa de red. En Windows, en lugar del tcpdump se usa el Windump, la cual tiene la misma funcionalidad pero aun así sigue siendo a nivel de comandos.

La utilización frecuente de tcpdump se dá en los casos siguientes.

- Para depurar aplicaciones que utilizan la red para comunicar.
- Para depurar la red misma.
- Para capturar y leer datos enviados por otros usarios o ordenadores. Algunos protocolos como telnet y HTTP no cifran los datos que envían en la red. Un usuario que tiene el control de un router a través del cual circula tráfico no cifrado puede usar tcpdump para lograr contrasenas u otras informaciones.

3.2.1. Ejemplos comunes

A continuación se lista algunos de los pisibles criterios a analizar al momento de iniciar un monitoreo.

- Capturar tráfico cuya dirección IP de origen sea 192.168.3.1

  tcpdump src host 192.168.3.1

- Capturar tráfico cuya dirección origen o destino sea 192.168.3.2

  host 192.168.3.2

- Capturar tráfico con destino a la dirección MAC 50:43:A5:AE:69:55

  ether dst 50:43:A5:AE:69:55

- Capturar tráfico con red destino 192.168.3.0

  tcpdump dst net 192.168.3.0

- Capturar tráfico con red origen 192.168.3.0/28

  src net 192.168.3.0 mask 255.255.255.240 src net 192.168.3.0/28

- Capturar tráfico con destino el puerto 23

  tcpdump dst port 23

- Capturar tráfico con origen o destino el puerto 110

  tcpdump port 110

- Capturar los paquetes de tipo ICMP

  ip proto
  icmp

- Capturar los paquetes de tipo UDP

  ip proto
  udp udp

- Capturar el tráfico Web

  tcp and port 80

- Capturar las peticiones de DNS

  udp and dst port 53

- Capturar el tráfico al puerto telnet o SSH

  tcp and (port 22 or port 23)

- Capturar todo el tráfico excepto el web

  tcp and not port 80

Capítulo 4

Desarrollo técnico

4.1. Introducción

El desarrollo inicialmente consiste en una aplicación capaz de administrar gráficamente una red de computadoras. Para ello utilizaremos las herramientas de desarrollo descritas en el primer reporte parcial. El desarrollo de la aplicación requiere de conceptos orientados a la programación concurrente, para ello se irán describiendo funcionalidades en particular que en este caso son filtrado y monitoreo. Posteriormente se plantean dos arquitecturas sobre las cuales se basa el desarrollo. Se maneja una arquitectura a nivel de datos ya que a pesar de ser un programa que manipulara hardware, se desea que todas las funcionalidades sean transparentes al usuario desde software alto nivel. Con dicha arquitectura se describen los componentes que el sistema manejara así como clases u objetos relacionados entre sí. Al término de la arquitectura anterior se generaliza una arquitectura general la cual se ha sometido a pruebas para la integración de módulos internos así como llamadas desde la interfaz gráfica. Este es quizá el elemento mas importante para el desarrollo de la aplicación ya que sobre esta arquitectura es posible iniciar a programar cada componente.

4.2. Niveles de Desarrollo

4.2.1. Nivel de procesos

El firewall Tendrá como función bloquear o permitir accesos desde o hacia una computadora dentro de la red. El firewall estará contenido en un único proceso que tendrá comunicación directa con una base de datos. EL proceso del firewall validará entradas y salidas a petición del administrador de la red y al mismo tiempo ejecutará a *Iptables* según los datos de entrada, ésto quiere decir que a cada acceso de administrador, el firewall sabrá el estado de red ayudándose de la base de datos central. Con todo este procesamiento del firewall, se accede mediante lectura y escritura a los datos de cada computadora en la red tales como, reglas de entrada, salida, IPs

y nombres de cada host. Ahora bien el monitor tendrá la una funcionalidad similar en cuanto a lectura y escritura en la base de datos. El monitor se encargara mediante un proceso, leer cada paquete transportado de una interfaz de red hacia otra. Solo puede trabajar a petición del administrador de la red mas no durante la ejecución de aplicación de reglas. Otro proceso será el que espere las peticiones desde los clientes o bien del mismo usuario. Con estos datos generados por el proceso de monitoreo, es posible determinar la frecuencia con la que se accede a un cierto sitio web, o bien, qué tipo de servicio se esta usando en cada host, para ello es necesario un almacenamiento de estos datos en una base central. Con esta descripción se puede establecer una arquitectura a nivel de procesos en la siguiente figura .

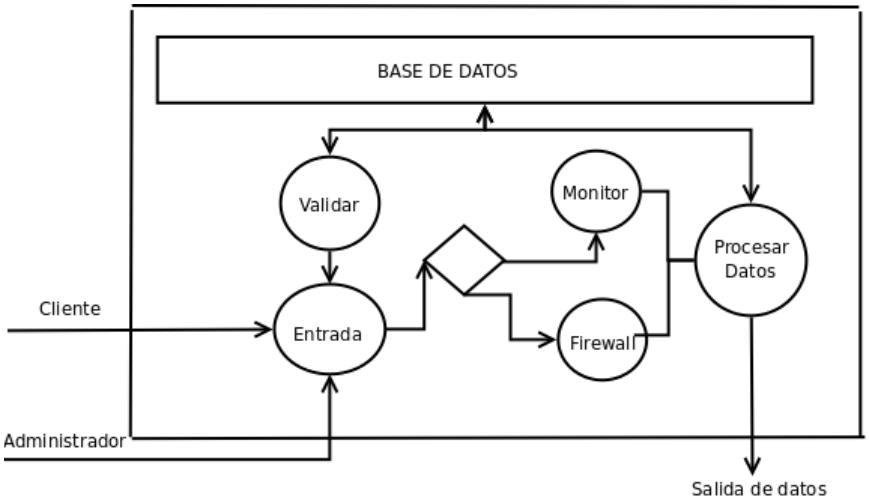

Figura 4.1: Diagrama que muestra la estructura de la aplicación a nivel de procesos[1]

4.2.2. Nivel de datos

En los procesos descritos anteriormente se plantea el uso de una base de datos alojada en el servidor. Ahora bien esta base de datos tiene como objetivo servir a dichos procesos como consultas de estados de cada computadora, datos de cada computadora, paginas visitadas, etc. El siguiente diagrama muestra las clases y sus relaciones que se utilizan para esta aplicación.

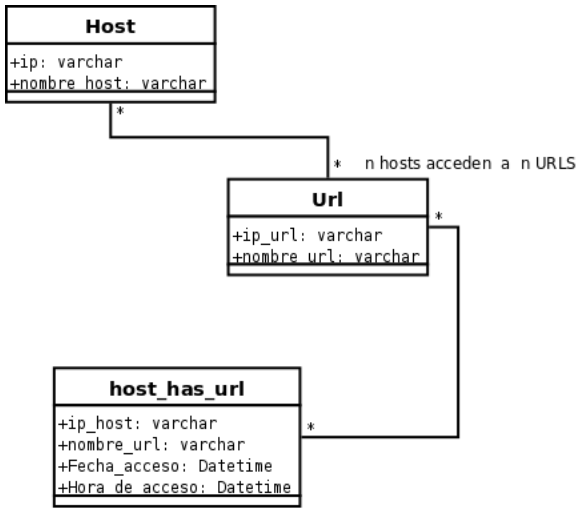

Figura 4.2: Diagrama que muestra la estructura de la aplicación a nivel de datos

4.3. Modelo General

Ya teniendo las particularidades del sistema es posible definir una arquitectura general integrando cada componente para ello se debe tomar en cuenta que posteriormente la aplicación puede estar sujeta a desarrollo y con ello agregar mas funcionalidades.

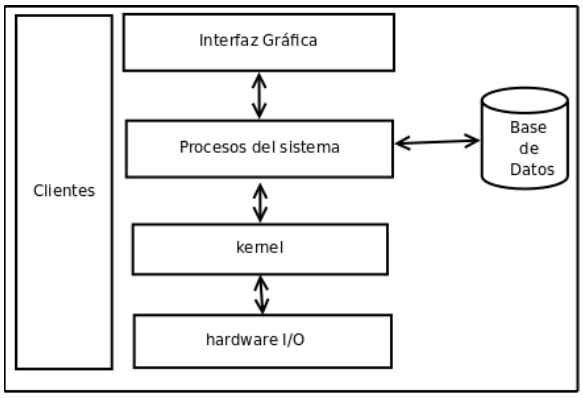

Figura 4.3: Diagrama que muestra la estructura de la aplicación con los elementos ya encapsulados

Capítulo 5

Implementación del Sistema

5.1. Introducción

Inicialmente se modelan las estructuras de cada módulo, en este caso el sistema de filtrado y el monitor. Para ello es necesario especificar las funciones principales dentro de la aplicación al inicio de la secuencia mediante diagramas de flujo.

Teniendo los bloques funcionales de cada módulo, se inicia la codificaciónen lenguaje C++ de cada módulo, filtrado y monitor. En la codificación de las funciones se deben específicar variables globales que sean comunes para ambos módulos.Cada función es ejecutadá mediante hilos, para ello se explicará el proceso de integración.

Posteriormente a la modelación de funciones, se describen las primeras pruebas. El programa es funcional a nivel de consola y está listo para segmentar las funciones en sus respectivos archivos cabeceras.

Después se inicia con la creación de las interfaces con QT-designer. Tales interfaces estan inicialmente independientes de los archivos codificados en lenguaje C++, en este caso cada interfaz se asociará a uno o más módulos de la aplicación.

Por último cada función de los módulos internos, se asocia a un elemento de la(s) interface(s) gráfica(s). Finalmente se muestran los resultados de las pruebas obtenidas.

5.2. Metodología de Filtrado

5.2.1. Consultas en base de datos

De acuerdo al apartado de desarrollo se plantea el nivel de datos. La aplicación tiene la opción de configurar la red previamente a la utilización de filtrado; con ello se establece una comunicación con un medio de almacenamiento centralizado, en este caso una base de datos. La base de datos *zisfilter* esta conformada por las siguientes tablas:

- direcciones

- url
- direccioneshasurl

Se usa el manejador de base de datos MySQL ya que en Lenguaje C es posible utilizar la biblioteca existente *mysql.h* con previa configuración.

De acuerdo a la prioridad de funcionamiento se espera que el administrador decida entre usar una configuración existente o bien cargar los componentes para primer uso, en este caso requiere detectar todos las maquinas de una red y almacenar el registro de cada una de ellas en una fuente de datos permenente; para ello la ejecución inicial corresponde al proceso de detección de dispositivos representado en el siguiente diagrama de flujo.

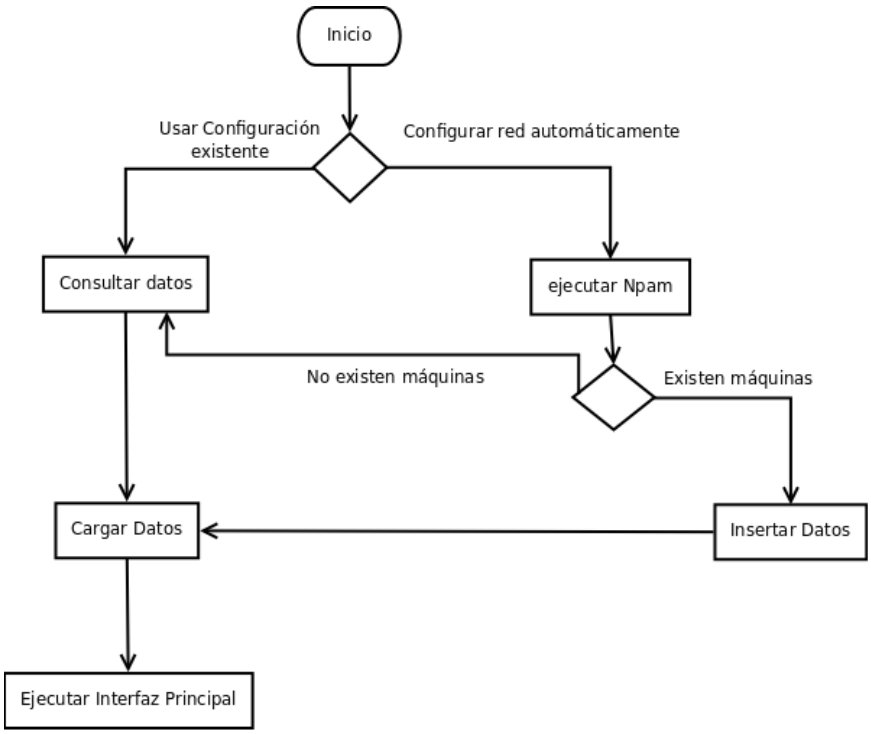

Figura 5.1: Diagrama de flujo que muestra la secuencia inicial de la aplicación

En la figura anterior el programa tiene como procesos al mapeo de dispositivos, consulta e insersión. El método de mapeo , registro en insersión registro se presentan mediante las siguientes funciones en pseudocódigo.

- Detectar dispositivos: Se ejecuta nmap y se volca la salida en salida.txt

```
1                void detectar_host(){
2
3                      system(nmap -v -sp 192.168.0.0/24 > salida.txt);
4
5                 }
6
7
8                crear_hilo(&detectar_host);
9                esperar_programa;
10               sigue_programa;
```

- Instertar Datos: aquí se insertan los registros resultantes de nmap en la base de datos

```
1                void insertar_datos(char *ip){
2
3                conectar_con_base();
4                elegir_base();
5                char *query = insert into zisfilter(ip) values('ip');
6                mysql_query(query);
7                mysql_close();
8                 }
9
10                mientras(maquinas){
11                      char *cadena_ip;
12                              leer(cadena_ip);
13                              insertar_datos(cadena_ip);
14                      }
15               fin mientras
```

- Cargar datos: aquí se cargan todas las máquinas detectadas en la interfaz de usuario

```
1                void Form_principal::carga_maquinas(){
2                char *query = select *from direcciones;
3                conectar_base();
4                       mysql_query(query);
5                        for(i=0; mysql_row[i];i++){
6                        etiqueta_interfaz[i]->setText(tr("%1").arg(mysql_row[i]));
7                           }
8                }
```

5.2.2. Creación de Hilos y archivos

El sistema realiza tareas específicas las cuales están independientes entre sí. Usando hilos ahorramos tiempo de ejecución; cabe mencionar que para el desarrollo de este proyecto se utilizaron hilos*(pthreads)*, ya que estos se ejecutan sobre un solo proceso y una ventaja sobre la creación de procesos es el tiempo. Dado que es poca carga de trabajo no se nota la diferencia en las pruebas realizadas, ya que habría que

probarlo con la cantidad máxima de computadoras posibles a procesar. Otra justificación referente al uso de hilos es que QT se comunica mediante *Signals* y *Slots* por consecuencia requiere de creación de hilos.

Un hilo de ejecución, en sistemas operativos, es una característica que permite a una aplicación realizar varias tareas a la vez (concurrentemente). Los distintos hilos de ejecución comparten una serie de recursos tales como el espacio de memoria, los archivos abiertos, autenticación, etc. Esta técnica permite simplificar el diseño de una aplicación que debe llevar a cabo distintas funciones simultáneamente.

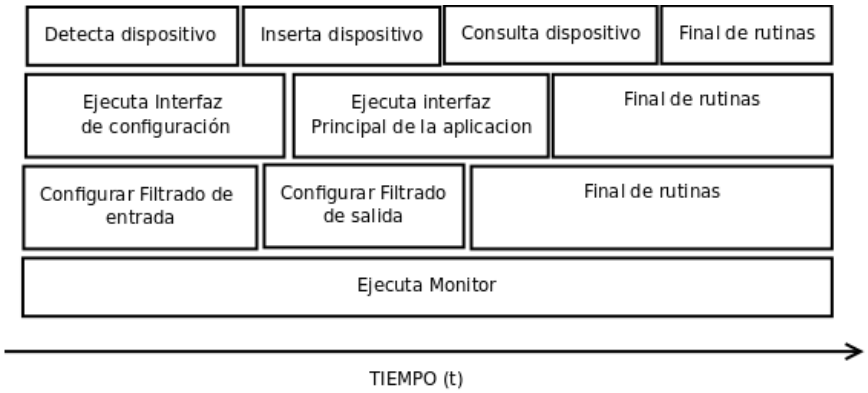

Figura 5.2: Bloque de rutinas que muestra las operaciónes más importantes del sistema así como los criterios de ejecución

La figura anterior describe la secuencia de funciones a las que en el peor caso se somete la aplicación. Decimos que peor caso ya que inicialmente el administrador de red puede o no solamente ejecutar una sola acción, como por ejemplo detectar dispositivos automaticamente; en otro caso puede invocar varias acciones a la vez, como detectar dispositivos y a su vez iniciar el proceso de filtrado o bien monitoreo.

En descripcioón del diagrama de bloques anterior se muestra la ejecución del módulo *detecta dispositivo* concurrente a *ejecuta interfaz* y *configurar filtrado de entrada*. Esto quiere decir que los tres bloques se ejecutarán al mismo tiempo pero en la práctica es posible que no terminen al mismo tiempo. La ejecución de la interfaz debe acabar despues de *configurar filtrado* y tambien de *detectar dispositivos*, y a su vez *detecta dispositivos* debe terminar antes que *configurar filtrado de entrada*.

Los módulos siguientes en el diagrama siguen el mismo contexto, y muestra qué funciones deberían ejecutarse primero y qué otras deben ejecutarse despues.

5.2.3. Formación y ejecución de una regla

La regla se irá formando de acuerdo a la decición tomada en cada interfaz. El siguiente diagrama muestra la secuencia de desciciones que el usuario puede tomar para finalmente formar una regla.

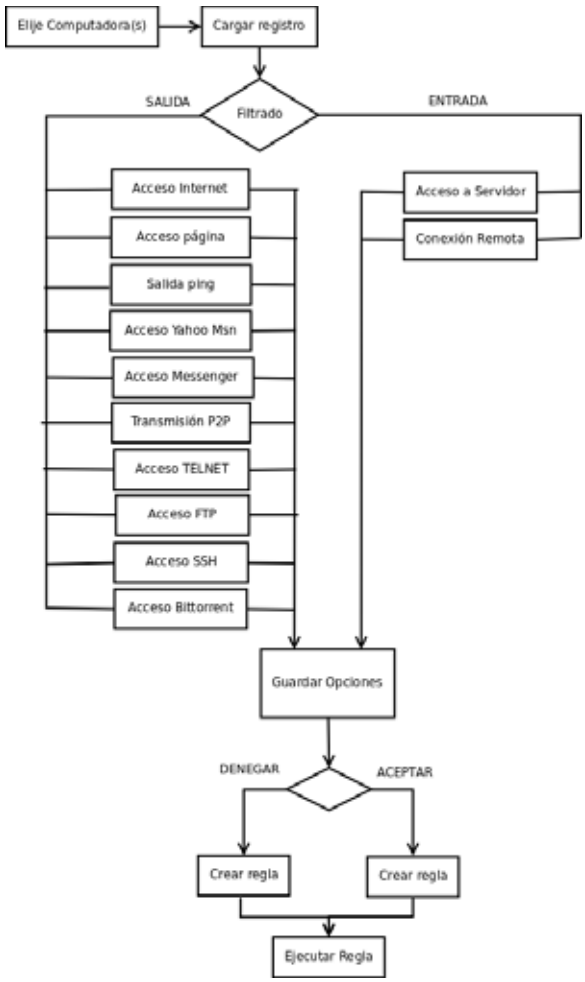

Figura 5.3: En este diagrama se describen las posibles trayectorias en el proceso de formación de reglas desde la interfaz de usuario

El diagrama anterior describe el proceso de creación de una regla desde la interfaz gráfica. Para ello se requieren estrategias claves para la comunicación entre los distintos componentes. A continuación se describe la implementacíon de las rutinas más importantes en base al diagrama anterior.

- Crear Computadora(s) en la interfaz:

  Cuando la aplicación inicia existen opciones de configuración, en este caso la opcion por defecto busca el registro en la base de datos y por cada instancia de computadora encontrada se crea un nuevo objeto con referencia a tal computadora de la red.

  Cada objeto de tipo boton es creado dinamicamente en la interfaz de modo que a cada elemento le corresponde una etiqueta. Mientras la interfaz está en ejecución, la aplicación busca en la base de datos central los registros de cada computadora.

Para este procesamiento se crean dos hilos, uno para cargar el estado de cada computadora y otro para cargar las instancias en la interfaz, en este caso botones y etiquetas asociadas.

- Cargar registro: Se cargaran los datos temporalmente al momento de seleccionar alguna computadora.A continuacion la implementación.

```
1 void  inicia_pc(int i){
2 FILE *entrada;
3 entrada = fopen("iniciado.txt", "w");
4           fprintf(entrada,"%s", la_ip[a]);
5 fclose(entrada);
6
7
8 }
```

  En el código de funciones de la interfaz principal, se crea una función `iniciar()` que estara encargada de inicializar la pc elegida desde la interfaz. En el código anterior se muestra que es guardado en dato en un archivo, pero puede inicialmente guardarse en una variable global.

- Seleccionar opciones de filtrado: Se seleccionan las opciones de filtrado en la interfaz asignada a ésta función.

  Una interfaz A se comunica con una Interfaz B por medio de *(Slots y Signals)* nativos de QT; ahora bien esto implica relacionar clases entre si. En C++ el mecanismo usado es común para todas las colaboraciones entre interfaces.La metodología es la siguiente.

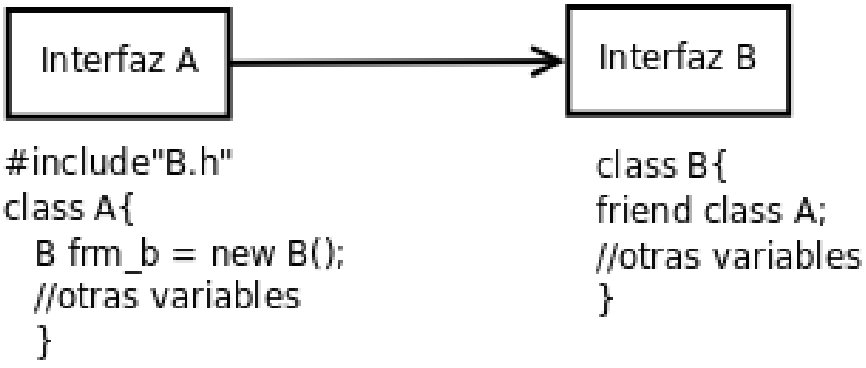

Figura 5.4: Colaboración entre una interfaz A con una interfaz B

La figura anterior muestra el método de comunicación entre interfaces. Primeramente la clase A incluye al archivo cabecera `B.h` de la clase B. En la clase B se indica la clase amiga invocadora; en este caso la clase amiga de B es A.

El problema más frecuente en la creación y comunicación de interfaces se centra en la inclusión de archivos cabecera. Si un archivo cabecera es doblemente incluido por accidente en distintos archivos, se genera un problema de *referencias cruzadas*. Por ello es necesario crear una jerarquía de clases, y con ello evitamos perder la secuencia de comunicación entre archivos.

Por cada invocación a interfaces se irá almacenando un registro según el módulo asociado. En este caso el usuario tiene como opció formar una regla a partir de las opciones de la interfaz.

5.2.4. Formación de una regla

Cada funcion es creada mediante un hilo. En el hilo de creación de una regla se van guardando datos útiles para el sistema ya sea en zona de memoria o bien archivos. Estos datos son extraidos al momento de generar la ultima condición; en este caso la *ACCION*, que en una regla de *Iptables* es el último parámetro. A continuación se describe el método.

- Recibir opción de acción desde la ultima interfaz del módulo de filtrado
- Extraer la opción de filtrado y almacenarla en variable temporal.
- Extraer dirección IP de la computadora seleccionada y almacenarla en variable temporal.
- Extraer la opción de coincidencia.

Al tener tales opciones, el sistema generará una regla mediante una concatenación de cadenas recursiva y se invocará al shell para la ejecución de la instrucción.

*regla = concatena(concatena(regla, opcion1), ...opcion n)

system(regla)

Este es el proceso fundamental en la formación de una regla y por consiguiente es el algoritmo de mayor jerarquía para el armado de una regla.

5.3. Metodología del Monitor

El capturar tráfico en tiempo real, implica manipular datos interpretados por el sistema operativo. En la interfaz gráfica de usuario el administrador tiene que visualizar solo la información necesaria, como por ejemplo las paginas que cierta computadora esta visitando en ese momento.La implementación se describe en el siguiente diagrama seguido de los algoritmos mas relevantes.

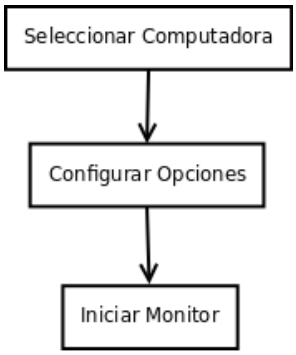

Figura 5.5: Secuencia de pasos para la ejecución del monitor

A primera vista se nota un diagrama sencillo de entender, pero cabe mencionar que el iniciar monitor implica una serie de control cuidadoso en los procesos del kernel. En la secuencia de pasos se toma en cuenta que la funcion *iniciaMonitor()*, es llamada a travez de un *Signal* desde la interfaz. A continuacion se muestra una implementación de el hilo de mayor jerarquía (*iniciaMonitor()*) y sus funciones auxiliares.

- Se activa la bandera del monitor, es decir estará en 1. Y mientras este activa la siguiente secuencia se ejecutara.
- Se crea un archivo donde se alojara la salida del comando
- Se extrae la ip seleccionada desde la interfaz
- Se extrae las opciones configuradas desde la interfaz
- Se forma el comando de monitoreo según las opciones

  *comando = concatena(concatena(comando, opcion1), ...opcion n)
- Se activa una bandera de lectura de paquetes la cual estara en 0(inactiva) mientras no se haga lectura y en 1(activa) mientras se este leyendo tráfico.
- Se crea un hilo el cual se encargará de activar y desactivar la bandera de lectura cada 3 segundos, es decir cada paquete se leera durante 3 segundos y el sistema tendrá tiempo de procesar la información y mostrarla al usuario mientras se leen las interfaces de red.

- Al mismo tiempo del hilo anterior se ejecuta el comando.

  system(comando)

La lectura tardará tres segundos, suficientes para guardar los datos en un medio temporal y extraerlos para su procesamiento. En el momento que la bandera de lectura se desactive o bien cambie a cero, los datos serán mostrados en la interfaz durante dos segundos, y a término de ello se vuelve a activar la bandera de lectura.

Con este ciclo de procesamiento se logra que la interfaz se refresque cada 2 segundos con la funcion sleep(2).En ese intervalo se muestra en la interfaz los puntos de transmisión de datos, en este caso dirección de origen y destino, que en el apartado de resultados se describirá.

En general ese fue el mecanismo empleado por el monitor; el monitor se detiene cuando se oprime detener, en ese instante la bandera de monitor activo se cambia a cero y el monitor termina.El emplear técnicas de programción concurrente implica manejar cuidadosamente las llamadas a procesos del kernel, ya que el mal uso de las técnicas puede disminuir el desempeño de la aplicación;

5.4. Jerarquía de Interfaces

El siguiente diagrama muestra la jerarquía de interfaces. La trayectoria de navegación depende de las deciciones del administrador.

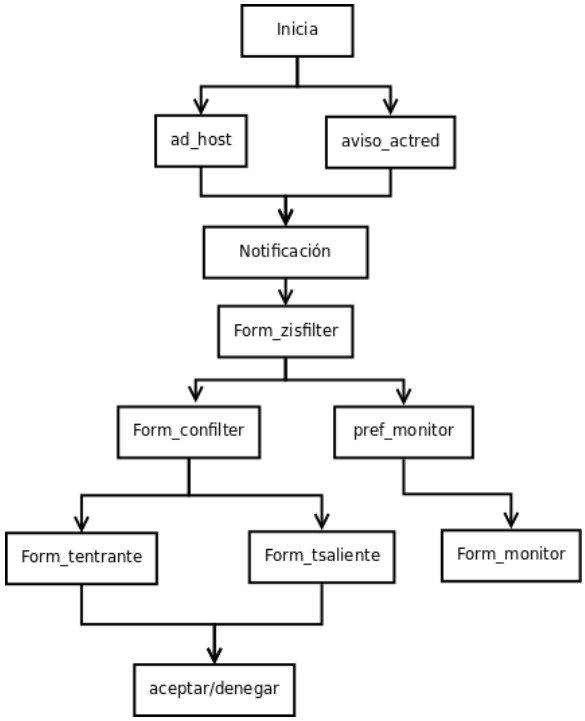

Figura 5.6: Diagrama de jerarquía de interfaces que describe la dependencia entre cada componente de la aplicación a nivel de interfaz

5.5. Qt y La IDE de desarrollo [1]

Qt[7]-designer es un entorno de desarrollo de interfaces gráficas creado por Trolltech [2]. Qt es una biblioteca para desarrollo de interfaces, producida actualmente por la division de software QT de Nokia [3]. Anteriormente Trolltech tenía la propiedad de QT y el 17 de junio del 2008 paso a ser propiedad de la empresa Nokia.

Originalmente la empresa permitía el desarrollo de aplicaciones en código cerrado solo comprando una licencia comercial así como aplicaciones de código abierto obteniendo una licencia Free Qt.

En septiembre del ano 2000, Trolltech ofreció la biblioteca de Qt v2.1 bajo la licencia GPL[4] exclusivo para versiones Linux. La versión para Mac OS X no se publico

[1]Entorno de desarrollo integrado

[2]Empresa creadora del proyecto QT, más acerca de trolltech en `qt.nokia.com`

[3]Nokia actualmente es el líder en la fabricación de teléfonos móviles en el mundo y principal empresa en el sector de las telecomunicaciones.

[4]Licencia Pública General de GNU

bajo la GPL si no hasta junio del 2003 y para windows en junio del 2005.

Qt es utilizado en el escritorio KDE[5] de Linux y utiliza al mismo lenguaje C++ de forma nativa aunque puede ser utilizado por otros lenguajes de programación.

Un programa hecho con Qt puede ser de consola o bien interfaz gráfica, para ello hay dos opciones; abrir un editor de texto y programar, o bien adquirir un IDE de desarrollo, en este caso puede ser Qt-designer. Qt sigue el paradigma orientado a objetos de C++, con ello es posible implementar clases de la misma estructura. La diferencia a ello es que hay que incluir las bibliotecas necesarias para la ejecución.

Un proyecto de Qt puede ser creado de diferentes formas. En la creación de un proyecto completo es más fácil el uso de la IDE de desarrollo, en este se es posible de manera mas rápido crear las interfaces involucradas; al término de la creacion de una interfaz se tiene que guardar como `interfaz.ui` en una carpeta vacia.

Para generar el proyecto primero es necesario crear un archivo main.cpp en la carpeta del proyecto que sera quien llame la instancia del tipo de dato en las cabeceras, despues de ello, abrir la terminal y dirigirse al directorio donde se encuentra la `interfaz.ui` en este caso debe estar en la misma carpeta de `main.cpp` y ejecutar los siguientes comandos.

- qmake -project

  Generará el proyecto

- qmake

  Creará el archivo Makefile necesario para la compilación.

- make

  Compilación del archivo

- ./ejecutable

  Ejecución del programa.

Los archivos `.h` y `.cpp` son creados al momento de ejecutar qmake. El archivo Makefile debe seguir ciertos criterios a la hora de compilación, es decir no basta con generarlo si no que hay que modificarlo para incluir las librerías adicionales como por elemplo las librerías para conexiones con MySQL. El archivo Makefile configura los parametros INCPATH y LIBS, y para ello se tienen que modificar dichos parametros de la siguiente forma.

```
1
2 INCPATH  = -I/usr/share/qt3/mkspecs/default -I. -I. -I/usr/include/qt3
3            -I/usr/include/mysql  -DBIG_JOINS=1  -fno-strict-aliasing   -DUNIV_LINUX
4
5
6 LIBS     = $(SUBLIBS) -Wl,-Bsymbolic-functions -rdynamic -L/usr/lib/mysql -lmysqlclient
7            -L/usr/share/qt3/lib -L/usr/X11R6/lib -lqt-mt -lXext -lX11 -lm -lpthread
```

[5]Proyecto de software libre basado en la creacion de un entorno de escritorio para los sistemas Linux.

Capítulo 6

Pruebas y Resultados

6.1. Configuración Inicial de Clientes y Servidor

La prueba de funcionamiento de realizó en un Servidor con las siguientes especificaciones de software:

-Sistema operativo Linux Ubuntu v9.04

-Servidor Apache v2.2.11

-MySQL server v5.0

-tcpdump 3.9.8

-nmap 4.76

-g++ 4.3

-qt-designer 3.0

Seis computadoras internas conectadas a un switch. Las computadoras internas solamente requieren de un interfaz de red para la conexión a internet. El sistema operativo es independiente en la red interna.

En la prueba de funcionamiento el servidor se configura para primer uso. El diagrama de conexión es el siguiente.

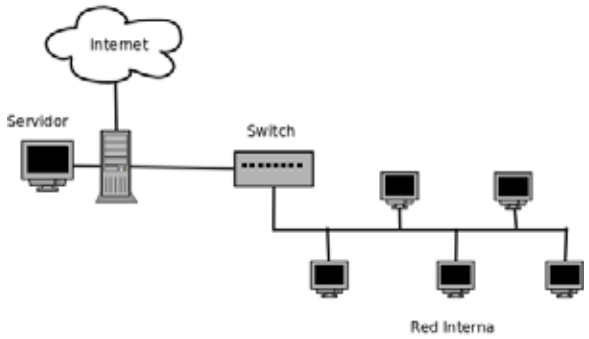

Figura 6.1: Diagrama de conexíon basado en la arquitectura Cliente-Servidor para pruebas de funcionamiento

En el diagrama anterior se muestra la conexión con la topología estrella[1] mediante un switch

Inicialmente se configura en el servidor las interfaces de red, para ello debemos asignar una dirección a la red interna, en este caso se asignó la direccion 192.168.0.04 mientras que la red que se conecta a internet tiene desde el router la dirección 192.168.1.04. El mecanismo consiste en filtrar datos provenientes desde la red 192.168.0.04 (para cualquier computadora), ahora para ello se configuran las computadoras clientes. La configuración del servidor se realizó de la siguiente forma.

- Configurar Interfaces de red

  `root@localhost dhclient eth0`

  `root@localhost ifconfig eth1 192.168.0.1`

- Permitir el reenvío de paquetes provenientes de la red interna y además traducir las direcciones de red para la comunicación externa.

  root@localhost iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -d 0.0.0.0/24 -j MASQUERADE

- Activar el Reenvío de paquetes *FORWARD*.

  root@localhost echo 1 > proc/sys/net/ipv4/ip_forward

  Con ello ya se tuvo el servidor configurado y listo para ejecutar la aplicación.

Un cliente con sistema Linux se configura de la siguiente forma:

- Configurar interfaces de red en cliente linux

  root@localhost ifconfig eth0 192.168.0.2

  root@localhost route add default gw 192.168.0.1

  En este caso la dirección de red para el cliente es 192.168.0.2 y se comunicará a través de la puerta de enlace 192.168.0.1. Esto se hace para cada cliente en caso de tener como sistema operativo Linux.

- Configurar interfaces de red en cliente Windows

  Para sistemas windows se realizo la prueba con clientes Windows XP y Windows Vista. La configuración en un cliente con Windows XP se realizó de la siguiente forma.

  Las configuraciónes de clientes con Windows Vista y Windows XP de realizaron de la siguiente forma.

[1] Es el tipo de red en la cual cada computadora están conectadas directamente a un punto central en este caso un switch y las comunicaciones pasan a través de éste.

Todas las computadoras clientes deben comunicarse a través de la interfaz eth01 configurada en el servidor, en este caso el servidor conecta la puerta de enlace(eth01) al switch y por consecuencia éste será encontrado automaticamente por cualquier cliente.

Cada computadora Windows debe tener una IP asignada 192.168.0.XX, donde XX es el número de computadora, en éste caso no se tiene que repetir las IPs en la red. Todas las computadoras Windows deben acceder a la red mediante la puerta de enlace 192.168.0.1. La diferencia es que Windows no detecta el servidor DNS y para ello es necesario asignar la dirección *Broadcast* [2] en el DNS pedido por el sistema Windows.

6.2. Prueba de funcionamiento de la Aplicación

En seguida se muestra el proceso de ejecución de la aplicación a través de capturas de imagen.

- Funcionamiento del sistema de filtrado. Caso de uso en particular , Bloquear Página WEB.

Figura 6.2: Pantalla de configuración inicial

Se seleccionó usar la configuración existente ya que previamente se detecto la red en (actualizar red completa). El sistema ya sabe que hay cinco computadoras y un servidor; en este caso coloca al servidor en la pantalla.

En esta toma de imagen, se muestra como la aplicacion carga las computadoras y bien ahora cada una puede ser seleccionada para iniciar a filtrar datos. Ahora se seleccionó una computadora y a tal acción se abre una pantalla de opciones.

[2]Dirección de difusión, es la dirección que tiene en su parte de host todos los bits del rango de las IPs permitidas en la red. A unos sirve para comunicarse con todas las computadoras de la red

Figura 6.3: Pantalla donde se muestra las computadoras de la red cargadas con su dirección Ip correspondiente

Figura 6.4: Pantalla de opciones en la cual se elige entre filtrar o monitorear tráfico

En este caso se selecciono (Tráfico saliente) y despues el sistema nos mostró las opciones.

Figura 6.5: Pantalla de opciones de filtrado asociado a la computadora seleccionada

Ahora bien para corroborar que las reglas se ejecutaron, se prueba en el cliente el acceso hacia la URL. En un caso en particular se tomó el caso de uso (Salida Página WEB). Entonces se seleccionó la opción (Salida a Página WEB), y con ello se ejecuta la pantalla donde se ingresara la durección URL a filtrar.

Figura 6.6: Pantalla donde se ingresó la página web a filtrar

Finalmente a la elección de alguna regla, en la siguiente pantalla el sistema pregunta qué se desea hacer con la opción de tráfico seleccionada. En este caso el sistema pregunta que hacer con el acceso hacia la página ingresada.

Figura 6.7: Pantalla de acciones donde se elige entre permitir o denegar

De acuerdo con el caso de uso anteriór la forma de probar que funcionó se refleja en el cliente seleccionado. En la computadora cliente se ingresó a la página seleccionada, y el navegador notificó que no se encontró tal ubicación. Para ello se tomó una captura de la terminal en el servidor que muestra el estado resultante de *Iptables* al término de la configuración de la regla.

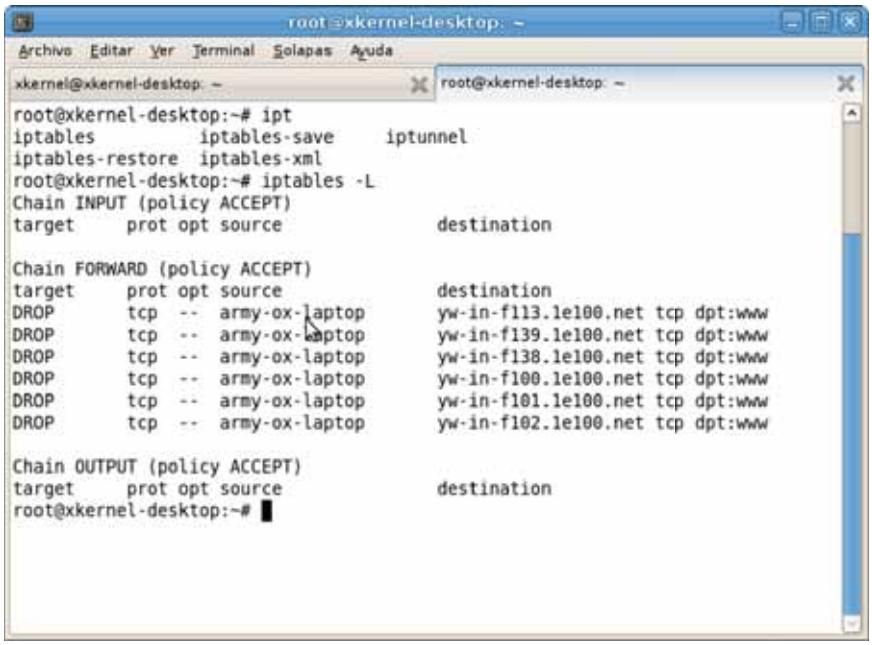

Figura 6.8: Toma de pantalla de la terminal mostrando la configuración de Iptables despues de asignar reglas desde la aplicación

- Funcionamiento del sistema de monitoreo. Caso de uso en particular , Capturar tráfico de toda la red dirigido hacia cualquier URL.

  El monitor se ejecutó al momento de seleccionar una computadora. En este caso si la computadora no accede a internet, el sistema no mostrará nada en la pantalla; en caso contrarío se desplegará la información referente a la computadora de origen y la dirección de destino. La secuencia fue la siguiente.

  Se inicia el monitor desde las opciones en la pantalla siguiente.

Figura 6.9: Toma de pantalla de la terminal mostrando la configuración de Iptables despues de asignar reglas desde la aplicación

Despues de ello se ejecutó la pantalla siguiente donde es posible configurar criterios de captura de tráfico. En este caso se seleccionó a opción de capturar tráfico dirigido a Internet exclusivamente hacia la página www.youtube.com.

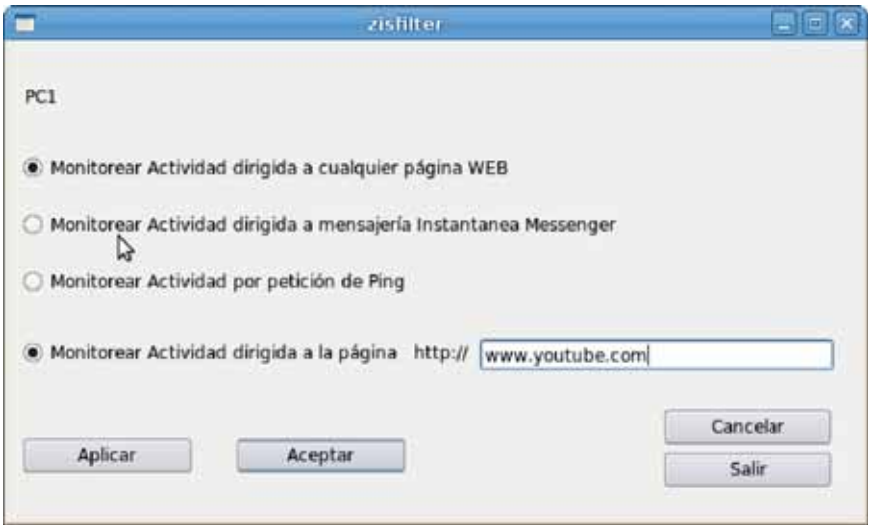

Figura 6.10: Pantalla donde se configuraron las opciones de captura de tráfico

Con las opciones seleccionadas fué posible iniciar el monitor. En la siguiente pantalla se muestra la salida del monitor en tiempo real.Tal función tiene como opciones iniciar y detener el monitor. Aún ejecutandose el monitor es posible aplicar configuraciones de filtrado.

Figura 6.11: Pantalla donde se mostró la salida del monitoreo en tiempo real, en este caso Lisa.Laptop:192.168.0.2 es una computadora en la red interna

6.3. Resultados

En este apartado se describió el uso en la aplicación en una red. Los resultados obtenidos fueron satisfactorios y los objetivos planteados desde el inicio del proyecto fueron cumplidos por el hecho de que las reglas configuradas en el servidor reflejaron funcionalidad para cada uno de los clientes de la red.

Capítulo 7

Conclusiones y trabajo futuro

7.1. Conclusión

El contar con herramientas gráficas para la manipulación es de gran utilidad, actualmente existen variedad de estas aplicaciones, pero lo fundamental es satisfacer propósitos especificos y manejo flexible dentro de la administración. Con ello gran parte de los programadores desarrollan aplicaciones estandares en las empresas para el control de la red, sin embargo solo ellos son capaces de entender el dominio de tales programas.

Durante el desarrollo se tiene que plantear una arquitectura y bien hacia que administrador estará orientada la aplicación. En el apartado de desarrollo, la programación del trabajo interno debe ser capaz de adaptarse a algun medio visual. Lo difícil se encuentra en la manera de adaptar los procesos a dichas llamadas externas, tambien es necesario cuidar que el kernel no genere procesos aleatorios ni tampoco que mantenga procesos dormidos en ejecución.

En un principio se analizó el sistema y la plataforma de desarrollo, en este caso Linux es el entorno sobre el cual se desarrolló la aplicación; para ello lenguaje C++ junto con las librerias de QT. En vista de que el lenguaje de programación C++ se extiende del nativo lenguaje C, se entiende que en la programación de sistemas sobre linux, es más usual acceder a los recursos desde programas codificados en C. QT mediante (*(Signals)* y *(Slots)*) accede a tales recursos, en consecuencia requiere de librerías estandares de UNIX codificadas en lenguaje C para la misma comunicación con métodos externos de otra clase o bien acciónes sobre algún proceso del kernel; en este caso para el sistema de filtrado se requieren librerias para la programación de hilos y creación de procesos.

Las herramientas de desarrollo como QT-designer solo auxilian y aceleran el proceso de diseño, esto quiere decir que el desarrollo total del sistema no depende de la (*IDE*) de desarrollo. Fué necesario generar un árbol de jerarquia de interfaces, ya que en el paradigma orientado a objetos destina a las clases a colaborar entre sí, y con ello establecer comunicación con el motor interno.

Es posible utilizar otros medios para la detección de cada dispositivo de red, si bien la metodologia de mapeo la realiza *Nmap*, ahora bien el interpretar la información y acceder a los recursos del sistema operativo queda al ingenio del programador de sistemas. Las herramientas existen pero hay que establecer medios flexibles de manipulación de estas y por supuesto esto requiere de extender mas el dominio sobre el entorno de las redes.

7.2. Trabajo futuro

Un trabajo posterior a este proyecto podría ser un desarrollo de clientes gráficos que puedan ejecutarse en clientes remotos independientes la platafoma; con ello agragar funcionalidad a la aplicación de tal manera que permita detectar redes segmentadas o bien subredes provenientes de distintos ruteadores.

En cuanto a la manera de acceder a los recursos del kernel es indispensable ahorrar trabajo al sistema operativo, lo que significa que se podría acceder directamente a los modulos del kernel para el control de iptables, esto seria mejor en cuanto a desempeño del hardware, ya que se evita invocar al *shell* para la ejecucion de instrucciones.Con esto, la aportación que se pretende es sobreescribir los medios de acceso al sistema operativo de tal modo que se evite la recurrencia de llamadas a procesos auxiliares como la misma consola. Estas metodologías quedan expuestas al ingenio del programador y sobre todo al desarrollador de software desde el contexto de arquitectura de software.

Apéndice A

Archivos fuente y Bibliotecas

A.1. Archivos fuente

Árbol de directorios de los archivos fuente

```
1 |-- Modulos
2 |    |-- avisos
3 |    |    |-- adhost.cpp
4 |    |    |-- adhost.h
5 |    |    |-- adhost.ui
6 |    |    |-- aviso_actred.cpp
7 |    |    |-- aviso_actred.h
8 |    |    |-- aviso_actred.ui
9 |    |    |-- aviso_agregado.cpp
10 |    |    |-- aviso_agregado.h
11 |    |    |-- aviso_agregado.ui
12 |    |    |-- aviso_agregado.ui.h
13 |    |    |-- aviso_hostad.cpp
14 |    |    |-- aviso_hostad.h
15 |    |    |-- aviso_hostad.ui
16 |    |    |-- inicia.cpp
17 |    |    |-- inicia.h
18 |    |    |-- inicia.ui
19 |    |    |-- moc_adhost.cpp
20 |    |    |-- moc_aviso_actred.cpp
21 |    |    |-- moc_aviso_agregado.cpp
22 |    |    |-- moc_aviso_hostad.cpp
23 |    |    `-- moc_inicia.cpp
24 |    |-- main
25 |    |    |-- Makefile
26 |    |    |-- aceptar_denegar.o
27 |    |    |-- adhost.o
28 |    |    |-- aviso_actred.o
29 |    |    |-- aviso_agregado.o
30 |    |    |-- aviso_hostad.o
31 |    |    |-- avisos
32 |    |    |    |-- adhost.cpp
33 |    |    |    |-- adhost.h
34 |    |    |    |-- aviso_actred.cpp
35 |    |    |    |-- aviso_actred.h
36 |    |    |    |-- aviso_agregado.cpp
37 |    |    |    |-- aviso_agregado.h
38 |    |    |    |-- aviso_hostad.cpp
39 |    |    |    |-- aviso_hostad.h
40 |    |    |    |-- moc_adhost.cpp
```

```
41 |    |    |    |-- moc_aviso_actred.cpp
42 |    |    |    |-- moc_aviso_agregado.cpp
43 |    |    |    `-- moc_aviso_hostad.cpp
44 |    |    |-- confilter.o
45 |    |    |-- form_intropage.o
46 |    |    |-- frames.txt
47 |    |    |-- hosts.txt
48 |    |    |-- inicia.o
49 |    |    |-- iniciado.txt
50 |    |    |-- inicial
51 |    |    |    |-- base_datos.h
52 |    |    |    |-- funciones.h
53 |    |    |    |-- inicia.cpp
54 |    |    |    |-- inicia.h
55 |    |    |    |-- inicia.ui.h
56 |    |    |    `-- moc_inicia.cpp
57 |    |    |-- laspcs.txt
58 |    |    |-- main
59 |    |    |-- main.cpp
60 |    |    |-- main.o
61 |    |    |-- main.pro
62 |    |    |-- mapeo.txt
63 |    |    |-- moc_aceptar_denegar.o
64 |    |    |-- moc_adhost.o
65 |    |    |-- moc_aviso_actred.o
66 |    |    |-- moc_aviso_agregado.o
67 |    |    |-- moc_aviso_hostad.o
68 |    |    |-- moc_confilter.o
69 |    |    |-- moc_form_intropage.o
70 |    |    |-- moc_inicia.o
71 |    |    |-- moc_puente_01.cpp
72 |    |    |-- moc_puente_01.o
73 |    |    |-- moc_tentrante.o
74 |    |    |-- moc_tsaliente.o
75 |    |    |-- moc_zisfilter.cpp
76 |    |    |-- moc_zisfilter.o
77 |    |    |-- motor_firewall.h
78 |    |    |-- opcion.txt
79 |    |    |-- puente_01.cpp
80 |    |    |-- puente_01.h
81 |    |    |-- puente_01.o
82 |    |    |-- res_make
83 |    |    |-- tabla_de_reglas.txt
84 |    |    |-- tentrante.o
85 |    |    |-- tsaliente.o
86 |    |    |-- url.txt
87 |    |    |-- zisfilter.cpp
88 |    |    |-- zisfilter.h
89 |    |    |-- zisfilter.o
90 |    |    `-- zisfilter.ui.h
91 |    |-- mod_confilter
92 |    |    |-- confilter.cpp
93 |    |    |-- confilter.h
94 |    |    |-- extrae.h
95 |    |    |-- moc_confilter.cpp
96 |    |    |-- mod_aceptar_denegar
97 |    |    |    |-- aceptar_denegar.cpp
98 |    |    |    |-- aceptar_denegar.h
99 |    |    |    `-- moc_aceptar_denegar.cpp
100 |    |    |-- mod_tentrante
101 |    |    |    |-- moc_tentrante.cpp
102 |    |    |    |-- tentrante.cpp
103 |    |    |    |-- tentrante.h
104 |    |    |-- mod_tsaliente
105 |    |    |    |-- form_intropage.cpp
106 |    |    |    |-- form_intropage.h
```

```
107 |    |    |    |-- moc_form_intropage.cpp
108 |    |    |    |-- moc_tsaliente.cpp
109 |    |    |    |-- tsaliente.cpp
110 |    |    |    |-- tsaliente.h
111 |    |    `-- puente_01.ui
112 |    |-- mod_monitor
113 |    |    |-- monitor.cpp
114 |    |    |-- monitor.h
115 |    |    `-- prefmonitor
116 |    |         |-- prefmonitor.cpp
117 |    |         |-- prefmonitor.h
```

A.2. Bibliotecas

Bibliotecas utilizadas:

```
1
2 /usr/lib/gcc/i486-linux-gnu/4.3/include/stdio.h
3 /usr/lib/gcc/i486-linux-gnu/4.3/include/string.h
4 /usr/lib/gcc/i486-linux-gnu/4.3/include/stdlib.h
5 /usr/lib/gcc/i486-linux-gnu/4.3/include/sys/types.h
6 /usr/lib/gcc/i486-linux-gnu/4.3/include/pthread.h
7 /usr/lib/gcc/i486-linux-gnu/4.3/include/unistd.h
8 /usr/lib/gcc/i486-linux-gnu/4.3/include/mysql/mysql.h
```

Apéndice B

Interfaces y descripción

B.1. Interfaces gráficas

En seguida se describirán las interfaces involucradas y su brebe descripcion y a grandes rasgos la funcionalidad principal.

- Nombre de la interfaz: `inicial.ui`

  Interfaces amigas: `Formzisfilter.ui`, adhost.ui, avisoactred.ui.

  Objetivo: Configurar por primer uso la aplicación.

  Descripción: Esta es la interfaz de entrada, en ella el administrador tiene la opción de decirle al sistema que configure la red automaticamente o bien agregar una nueva computadora manualmente. La diferencia de ello es que al reconfigurar la red es posible que los datos de las computadoras cambien según las modificaciones de configuración que haya hecho el administrador sobre alguna computadora en especial. Sin embargo el registro de los accesos a las páginas y las reglas configuradas para cada computadora permanecerán estables al menos de que el administrador decida restablecer la configuración predeterminada.

Figura B.1: Interfaz Gráfica inicial.ui

- Nombre de la interfaz: Formzisfilter.ui

  Interfaces amigas: Formconfilter.ui, Formmonitor.ui

  Objetivo: Mostrar las computadoras en la red mediante iconos y preparar las acciones sobre las mismas.

  Descripción: Esta interfaz se encarga de mostrar al usuario todas las computadoras dentro de la red. Así como tambien iniciar el proceso para la aplicación de reglas, uso del firewal y el monitor. Esta es la pantalla de la aplicación principal. Cada componente tendra su propio procesamiento y sabran a qué widget dirigir dichas acciones.

Figura B.2: Interfaz Gráfica Formzisfilter.ui

- Nombre de la interfaz: Formconfilter.ui

  Interfaces amigas: Formtentrante.ui, Formtsaliente.ui

  Objetivo: inicial las opciones del firewall sobre la computadora seleccionada.

  Descripcion:

  Esta interfaz se encarga de iniciar el procesamiento del firewall sobre la computadora seleccionada en Formzisfilter.ui. Para ello primeramente se elije que tipo de acceso hay que trabajar (entrada o salida).

Figura B.3: Interfaz Gráfica Formconfilter.ui

- Nombre de la interfaz: Formtentrante.ui

  Interfaces amigas: .ui

  Objetivo: inicial las opciones del firewall sobre la computadora seleccionada para todo acceso hacia dicha computadora.

  Descripción: Esta interfaz estará encargada de configurar las básicas acciones sobre accesos hacia una computadora. Por ejemplo el permitir o no que se localice por ping, acceder a compartir archivos, entradas de mensajes, etc.

Figura B.4: Interfaz Gráfica Formtentrante.ui

- Nombre de la interfaz: Formtsaliente.ui

  Interfaces amigas: aceptdeng.ui

  Objetivo: inicial las opciones del firewall sobre la computadora seleccionada para todo acceso desde dicha computadora hacia internet o bien hacia otra computadora.

  Descripción: Esta interfaz estará encargada de configurar las básicas acciones sobre accesos desde una computadora hacia la red externa. Por ejemplo el permitir o no que acceda a internet, bloquear una pÃ¡gina web, permitir uso de mensajería, etc.

Figura B.5: Interfaz Gráfica Formtsaliente.ui

- Nombre de la interfaz: Formtaceptdeng.ui

  Interfaces amigas: Ninguna

  Objetivo: Aceptar o Denegar la norma formada en Formtentrante.ui o Formtsaliente.ui Descripción:

  En esta interfaz solamente se elije aceptar o denegar la regla formada en las interfaces anteriores.

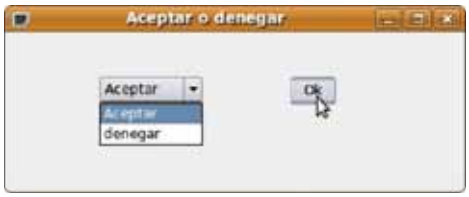

Figura B.6: Interfaz Gráfica Formtaceptdeng.ui

- Nombre de la interfaz: Formmonitor.ui

  Interfaces amigas: Prefmonitor.ui

  Objetivo: Ejecutar el sistema de monitoreo

  Descripción: En esta interfaz se inicia el proceso de monitoreo de todo lo que el sistema detecte por las interfaces de red.

Figura B.7: Interfaz Gráfica Formmonitor.ui

- Nombre de la interfaz: Prefmonitor.ui

  Interfaces amigas: Ninguna

  Objetivo: Establecer criterios previos al monitoreo

  Descripción: En esta interfaz se eligen criterios previos al monitoreo para capturar específicamente uno o varios tipos de paquetes.

- Nombre de la interfaz: adhost.ui

  Interfaces amigas: Avisohostad.ui

  Objetivo: Agregar un host manualmente

  Descripción: En esta interfaz se puede agregar un host a la base de datos de forma manual.

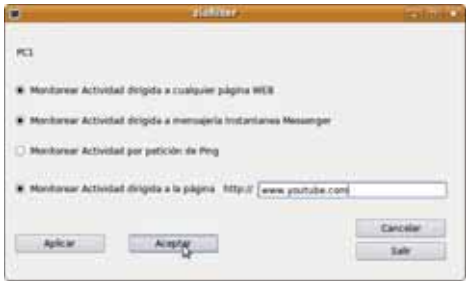

Figura B.8: Interfaz Gráfica Prefmonitor.ui

Figura B.9: Interfaz Gráfica adhost.ui

- Nombre de la interfaz: avisohostad.ui avisoactred.ui

  Interfaces amigas: Ninguna

  Objetivo: Notificar los cambios realizados o bien cancelar operaciones.

  Descripción: En estas interfaces solamente se notificara al administrador los cambios realizados en la configuración del sistema, o bien también podrá cancelar dichas operaciones.

Figura B.10: Interfaz Gráfica avisohostad.ui y avisoactred.ui

Bibliografía

[1] *Introduccion a los sistemas operativos*. Sitesa, Delaware E.U.A, addison wesley edition, 1987.

[2] *Cisco CCENT/CCNA*. ciscopress, Madrid, wendell odom ccie edition, 2008.

[3] *http://es.tldp.org/ManualesLuCAS/dociptablesfirewall/*. consultado el 28 de diciembre del 2009.

[4] *http://www.tuchemnitz.de/docs/lindocs/RH73/RHDOCS/rhlrges7.3/s1iptablesÂ-Â/options.html*. consultado el 28 de diciembre del 2009.

[5] *http://www.lugmen.org.ar/pipermail/luglist/2004September/030693.html*. consultado el 28 de diciembre del 2009.

[6] *http://www.pello.info/filez/firewall/iptables.html*. consultado el 28 de diciembre del 2009.

[7] *Qt-c++ programming*. Prentice hall edition, 2007.