Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Proyecto terminal :

Sistema de seguridad basado en contenido de correos electrónicos

Castro Hernández Natalia - 206201915

Asesor:
M. en C. Oscar Alvarado Nava

20 de Agosto del 2013

Dedicado a mis padres

Sólo es capaz de realizar
los sueños el que,
cuando llega la hora,
sabe estar despierto.
León Daudí

Agradecimientos

Mahatma Gandhi escribió: Nuestra recompensa se encuentra en el esfuerzo y no en el resultado. Un esfuerzo total es una victoria completa.

Durante mucho tiempo se fueron construyendo sueños, sueños que algún día se convirtieron en lejanos y que hoy se hacen realidad. Hoy se que todo ese esfuerzo, comienza a dar sus frutos, hoy puedo decir que las cosas se logran con dedicación, con ilusiones, con esfuerzo. La verdadera fortaleza esta en nunca abandonar lo que de verdad quieres lograr y que por muy obscuro que se vea el camino, siempre habrá una luz al final.

Hoy quiero dar las gracias a mis padres, que a pesar de todo nunca dudaron de mi, siempre se mantuvieron a mi lado, gracias por todas las palabras de aliento, gracias por el esfuerzo que hicieron para que llegara hasta aquí, y sobre todo gracias por la gran herencia que hoy me dejan, mi educación,

Quiero dar las gracias a mi asesor de proyecto al M. en C. Oscar Alvarado Nava, por confiar en mi proyecto, por el tiempo que dedico en solucionar mis dudas, por que a pesar del tiempo que tarde en concluirlo, nunca abandono el interés de realizarlo.

Y en especial quiero agradecer a Alejandro Gonzalez Torres, quien siempre estuvo conmigo, apoyándome, aún cuando llegue a pesar que nunca terminaría este proyecto, aún cuando no teníamos idea de como hacer las cosas, gracias por todo tu apoyo, tu dedicación, tu esfuerzo, tu interés.

Resumen

El presente proyecto muestra una solución al entorno empresarial, ya que provee una alternativa para evitar las fugas de información, por medio de lo que hoy se considera un recurso de vital importancia para la comunicación, el correo electrónico.

Pensando en una de las amenazas constantes que presentan las empresas, se implemento un sistema de seguridad, que permite filtrar el contenido de los correos electrónicos, a partir de un patrón que clasificará los correos en base a su contenido.

Los clientes de la red corporativa podrán solicitar el acceso al servidor de correo local, de modo que, cuando el acceso es permitido y la conexion se establece, se captura el correo antes de ser enviado al servidor local. La información contenida es analizada con base en el patrón de datos que indicará si dicho correo contiene o no información importante. Si el contenido no coincide con el patrón de datos, el correo es enviado al servidor local; el cual se encargará de enviarlo al servidor de correos exterior; en caso contrario, el correo será filtrado y se generará una alerta de infiltración de información, la cual dependerá del grado riesgo que éste pudiera generar.

El patrón de datos utilizado, dependerá de las especificaciones y requerimientos de la empresa, es decir, esto se define a partir de palabras claves que se generan de la información que se considere confidencial.

El desarrollo del sistema se seguridad, se dividió en módulos, los cuales nos permitirán tener control sobre el funcionamiento del mismo, ya que cada módulo estará destinado a realizar una función en específico y son trascendentales para establecer la seguridad en la información, que permitirá realizar el filtrado de los correos electrónico.

Índice

Agradecimientos **II**

Resumen **III**

1. Objetivos **1**
1.1. Objetivo General 1
1.2. Objetivo Particulares 1

2. Introducción **1**

3. Justificación **3**

4. Antecedentes **4**
4.1. Referencias Internas 4
4.2. Referencias Externas 5

5. Fundamentos Teóricos **5**
5.1. Sniffer 5
5.2. Libpcap 6
5.2.1. S0 Inicialización 7
5.2.2. S1 Establecimiento del Filtrado 7
5.2.3. S2 Captura de Paquetes 8
5.2.4. S3 Extracción de Datos 8
5.2.5. S4 Procesamiento de Datos 8
5.3. SMTP(Simple Mail Transfer Protocol) 9
5.3.1. Modelo SMTP 9
5.3.2. Comandos y respuestas de SMTP 10
5.3.3. Modo de comunicación SMTP 11
5.4. POP3 (Post Office Protocol) 13
5.5. IMAP (Internet Message Access Protocol) 13
5.6. DNS(Domain Name Service) 13
5.7. TCP (Transmission Control Protocol) 14
5.8. Iptables 14
5.8.1. NAT (Network Address Translation) 15
5.8.2. Tablas 15
5.8.3. Cadenas 15
5.8.4. Reglas 16
5.8.5. Destinos/Objetivos 16
5.8.6. Seguimiento de Conexiones 17
5.9. Qmail 18
5.9.1. Características 18
5.9.2. Módulos de qmail 19
5.9.3. Proceso de envió de un mensaje 19

6. Desarrollo Técnico **20**
6.1. Desarrollo General 21
6.2. Módulos de desarrollo 23
6.2.1. Módulos de clasificación de direcciones IP 23
6.2.2. Módulo de administración del servidor de correo Qmail 24
6.2.3. Módulo de filtrado de correos electrónicos 24
6.2.4. Módulo de generación de alertas 28
6.2.5. Módulo de estadísticas 28
6.2.6. Proceso de filtrado 29

7. Especificación Técnica **30**
7.1. Investigación Previa 30
7.2. IDE de desarrollo 31
7.3. Valores de Entrada y Salida. 32
7.4. Características Mínimas 32
7.5. Posibles alcances a futuro 32

8. Pruebas y Resultados **32**
8.1. Hardware 32
8.2. Software 33
8.3. Herramientas adicionales utilizadas 33
8.4. Cuentas de usuarios 33
8.5. Pruebas No.1.- Envió entre clientes del mismo dominio 33
8.6. Prueba No.2.- Envió de correos entre clientes de dominios distintos 37
8.7. Prueba No.3.- Filtrado de correos 38

9. Conclusión **43**

A. Manual de Instalación **44**
A.1. Configuración del servidor de correos qmail 44
A.2. Configuración del DNS 53
A.3. Configuración de Iptables 54
A.4. Inicializacón de los archivos de arranque 56
A.4.1. Configurar el inicio automático de qmail 56
A.4.2. Inicializar Bind 56
A.5. Configuración de NetBeans para crear un proyecto con jNetPcap 57
A.5.1. Creación de una nueva biblioteca bajo Netbeans 57
A.6. Manual de uso del sistema de seguridad 59

B. Código Fuente **61**
B.1. Interfaz.java 61
B.2. IpList.java 71
B.3. JNetPcap 74
B.4. Smtp.java 86

C. Scripts de inicio **92**
C.1. Script de inicio qmail, bind 92
C.2. Eliminar Iptables 93
C.3. ón Sistema de Seguridad 94

D. Bibliografía **94**

Índice de figuras

1. Elementos involucrados en el proceso de captura. 6
2. Esquematización de un programa con Libpcap. 7
3. Modelo básico de SMTP. 10
4. Secuencia de intercambio de comandos del protocolo SMTP. 12
5. Pila de protocolos de Internet. 14
6. Filtrado de paquetes con Iptables. 17
7. Arquitectura de qmail. 20
8. Maqueta de comunicación y pruebas. 21
9. Diagrama de tiempos del proceso de filtrado de correos. 22
10. Módulos del sistema de seguridad. 23
11. Configuración del buffer de error. 25
12. Obtención de la interfaz activa. 25
13. Método para obtener el nombre de la interfaz. 25
14. Método para obtener abrir una interfaz. 26
15. Método para bucles de envío. 27
16. Método para recibir paquetes. 27
17. Finalizar Pcap. 27
18. Filtrado de correos. 28
19. Diagrama de flujo del proceso de filtrado. 30
20. Arquitectura de API jNetPcap. 31
21. Configuración de un cliente. 34
22. Configuración del servidor de un cliente. 34
23. Configuración del puerto. 35
24. Envío de correo. 35
25. Captura de tráfico de red. 36
26. Secuencia de comunicación cliente-servidor. 36
27. Secuencia de comunicación cliente-servidor. 37
28. Envió de correos desde el servidor qmail.zone.com. 37
29. Captura de tráfico de red desde qmail.zone.com 38
30. Secuencia de comandos y respuestas del servidor-cliente 38
31. Interfaz del administrador 39
32. Correo de prueba para el filtro. 39
33. Correo en proceso. 40
34. Correo invalido. 40
35. Dirección bloqueada. 41
36. Iptables bloqueada. 41
37. Mensaje recibido por el destinatario. 42
38. Mensaje recibido por el administrador. 42
39. Nuevo mensaje. 43

Índice de cuadros

1. Comandos de SMTP 11
2. Respuestas de SMTP 11
3. Tablas de Iptables 15
4. Destinos de Iptables 16
5. Estados de Iptables 17
6. Módulos centrales de qmail. 19

1. Objetivos

1.1. Objetivo General

Desarrollar un sistema de seguridad que permita la clasificación de correos electrónicos a partir de su contenido.

1.2. Objetivo Particulares

- Diseñar un esquema para la clasificación de direcciones IP[1] que permita y determine la supervisión de los correos electrónicos.
- Operar un servidor de correo electrónico[2] para administrar y consultar correos externos desde el mismo sistema.
- Diseñar e implementar un módulo de filtrado de correos electrónicos a partir del reconocimiento de un patrón de datos, para la clasificación y cancelación del envió del correo.
- Diseñar e implementar un módulo para la generación de alertas, que determine la acción a realizar a partir de los correos filtrados.
- Diseñar e implementar un módulo que registre las estadísticas de conexión, para la generación de reportes.
- Integrar los módulos.

2. Introducción

El correo electrónico y la mensajería instantánea, son hoy, sin lugar a dudas, las herramientas más utilizadas como medio de comunicación tanto personal como empresarial y se han convertido ya no en una manera de comunicarnos, sino en "la" manera como nos comunicamos.

En la actualidad, las organizaciones se apoyan de distintos medios de comunicación en sus procesos comunicativos, entre los que está adquiriendo una importancia cada vez mayor el correo electrónico, considerándolo como un elemento vital para generar relaciones comerciales con empresas y organizaciones de todas partes del mundo, así como, la forma más sencilla de compartir información con los demás miembros de la organización.

La información es uno de los principales activos de las empresas y por ello las nuevas tecnologías de la información y la comunicación se han convertido en una herramienta imprescindible para desarrollar cualquier actividad económica, así como en un factor clave para mejorar la productividad, por lo tanto, dicha información debe de ser protegida de

[1]Dirección IP: El número que identifica a cada dispositivo dentro de una red con protocolo IPv4.
[2]Servidor de Correo: Se encarga de gestionar los correos de los usuarios de su dominio o empresa.

un modo adecuado.

Los modelos de seguridad tradicionales (*firewall* [3], *proxy* [4]) se enfocan en mantener alejados a los atacantes externos. La realidad es que existen amenazas tanto dentro como fuera de la organización. El robo de los secretos comerciales y sabotaje por parte de los empleados es una de las constantes preocupaciones de las amenazas internas que enfrentan las empresas.

Aunque el correo electrónico ha demostrado sus grandes beneficios, al utilizarlo hay que observar reglas básicas de seguridad que permitan a las empresas estar protegidas contra los accesos inadecuados de la información que sale de su red.

El correo electrónico y los sniffers[5] son las tecnologías de interés para este proyecto, se aprovechara el auge y los beneficios que ofrecen para establecer seguridad en la información en el entorno empresarial. Se propone una solución que implemente un sistema de seguridad estratégico para identificar el tipo de información que se transmite a través de un correo electrónico y prevenir la pérdida de datos.

En este proyecto se desarrolló un sistema de seguridad que permite filtrar los correos electrónicos en base a su contenido, ya que permite identificar qué tipo de información se envía a través de la red, lo cual representa una solución empresarial para prevenir la pérdida de datos, determinar las posibles fuentes de la fuga, clasificar esa información e iniciar la medida correctiva; por ejemplo: eliminación de contenido de los correos.

Para el desarrollo de la aplicación se diseñaron módulos de configuración y programación con base en los objetivos particulares del proyecto, en los cuales cada uno cumple una función específica y son trascendentales para establecer la seguridad en la información, que permitirá realizar el filtrado de los correos electrónico.

Para realizar las observaciones, pruebas y resultados del correcto funcionamiento del sistema de seguridad se implementó una red LAN[6] para la interacción entre los diferentes dispositivos, formada por un equipo de computo en el que se implementó el sistema de seguridad y en el que se configuró el servidor de correo, esté equipo es el encargado de cumplir con la función de filtrado y comunicación con el servidor, dos equipos de computo, los cuales serán clientes del servidor de correo y que pertenecen a la red empresarial, un equipo de computo que será un host que pertenece a otra red, dicha función de estos equipos será el envió de correos entre ellos, tanto dentro como fuera de la red.

[3]Firewall: Sistema para filtrado de paquetes y protección de una red interna, de acceso restringido y con información privada.

[4]Proxy: Es un software o dispositivo que sirve para permitir el acceso a Internet a todos los equipos de una organización.

[5]Sniffer: Es un software que permite colocar la interfaz de red de una estación en modo promiscuo(es aquel en el que una computadora conectada a una red compartida, captura todo el tráfico que circula por la red) y capturar toda la información que circula por la red.

[6]LAN: Red de Área Local, para intercambio de información en distancias cortas.

3. Justificación

La evolución de hardware y el desarrollo de software en la actualidad se ve reflejada en las formas de comunicarnos, el desarrollo de internet gracias a la fibra óptica, las computadoras personales y portátiles, o los dispositivos móviles, son sin duda una muestra de los alcances del desarrollo tecnológico. En las empresas, universidades o en el hogar tenemos la necesidad de informarnos y comunicarnos con las personas sin importar donde nos encontremos.

Debido a las tendencias actuales de utilizar elementos como las redes sociales, el correo electrónico, la computación en nube y los dispositivos personales móviles en la empresa, se ha percibido un cambio en el entorno de riesgo que enfrentan las organizaciones, por el creciente aumento del acceso a la información. Aunque las tecnologías informáticas han demostrado su fiabilidad, al utilizarlas hay que observar unas reglas básicas de seguridad que permitirán estar protegidos contra los accesos inadecuados a la información, la pérdida de datos y problemas similares.

Las soluciones de seguridad de la información tradicionales que se enfocan en las amenazas externas podrían exponer a las organizaciones a otras formas de ataque, especialmente desde adentro, las herramientas de prevención de pérdida de datos trabajan para evitar que las personas con acceso a información privilegiada copien esta a discos extraíbles o le envíen archivos por correo electrónico a un competidor.

La seguridad de la información debe ser un elemento base de la estrategia general de administración de riesgos de una organización. Resulta esencial contar con un programa estratégico para fortalecer la seguridad a largo plazo, este le permitirá tratar amenazas actuales y futuras, reducir el costo de las medidas de seguridad y facilitará el manejo de incidentes de seguridad de la información.

Razón por la cual se presenta un interés de desarrollar una herramienta de prevención de pérdida de datos que permita verificar el contenido de los correos electrónicos, prevenir las fugas de información y disminuir los riesgos en las organizaciones.

Para este proyecto se eligió la librería Libpcap[7], por ofrecer una interfaz para capturar los paquetes que circulan por la red, además de su portabilidad en diferentes sistemas operativos y por ser la base de distintas aplicaciones que funcionan como sniffers(Wireshark, Tcpdump, Netsniff-ng, entre otros), la cual nos permitirá establecer el filtrado de los correo electrónicos.

[7]Libpcap: es una librería open source escrita en C que ofrece una interfaz desde la que capturar paquetes en la capa de red

4. Antecedentes

4.1. Referencias Internas

El desarrollo de este proyecto tiene relación con temas como: Seguridad, Redes y Reconocimiento de datos, por lo cual se tomarán como referencias proyectos que tienen relación a dichos temas. En la UAM Azcapotzalco se tiene desarrollo de proyectos como:

- Dispositivo de seguridad implementado en un sistema empotrado con software libre, por el alumno Ramos León Anthony Damián y como asesores de proyecto el M. en C. Arturo Zúñiga López y el Dr. Juan Villegas Cortez.

  En este proyecto el alumno implementa un dispositivo de seguridad que será conectado a una red LAN, donde su principal funcionalidad será la protección frente a amenazas como: Spam[8], virus y amenazas web.

  En el proyecto a desarrollar la principal funcionalidad será el análisis de correos electrónicos, que determinará si un correo contiene o no información confidencial y a partir de esto se realizará la clasificación de los mismo.

- Diseño e implementación de una LAN Virtual con switches de red, por el alumno Castillo Cabrera Juan Alberto y como asesor de proyecto el Prof. José Ignacio Vega Luna.

  En este proyecto el alumno realiza el diseño e implementación de subredes mediantes switches de red, para controlar la propagación y difusión de información en la red, proporcionando un método de segmentación de dominios de difusión, con el objeto de reducir costos, mejorar el rendimiento y la seguridad de la red.

  El desarrollo de este proyecto se realizará bajo la implementación de una red LAN y en la cual se realizará el análisis de la información que se propaga en los correos electrónicos a través de la red.

- Aplicación de Distintas Técnicas de Minería de Datos para el Tratamiento de Información, por la alumna Guzmán González Nancy y como asesor de proyecto el Prof. Héctor Javier Vázquez.

  En este proyecto el alumno implementa una aplicación cuya funcionalidad se basará en los sistemas de información que poseen gran cantidad de datos incongruentes, lo cual obstaculiza el estudio y extracción de datos. De este modo se podrán hacer

[8]Spam: Práctica de enviar mensajes de correo electrónico no solicitados.

mejores estimaciones, deducir relaciones y patrones fiables.

En el desarrollo de este proyecto se utilizará una técnica de reconocimiento de patrones de datos para realizar la clasificación y filtro de los correos en base a su contenido.

4.2. Referencias Externas

- *MailMarshal*: Es una herramienta que permite definir y aplicar políticas de seguridad y de uso aceptable para el correo electrónico. Examina el contenido de cada mensaje, bloqueando el paso a los virus, al código malintencionado y a patrones de comportamiento sospechosos. Evita la difusión de datos confidenciales.

  El proyecto a desarrollar permitirá, en base a la generación de scripts, la clasificación de correos cuyo contenido se relaciona con información confidencial de una empresa.

- *Google Message Discovery*: Es un producto de seguridad de Google, que establece políticas de correo electrónico centralizadas para gestionar el contenido y hacer cumplir los requisitos de conformidad.

  En el proyecto a desarrollar se establece un patrón de datos para gestionar el contenido de los correos electrónico y así realizar el filtrado y clasificación de los mismos.

5. Fundamentos Teóricos

5.1. Sniffer

Los sniffers, también conocidos como analizadores de paquetes, son programas que tienen la habilidad de interceptar el tráfico que pasa por la red. La captura de paquetes es la acción que realizan los sniffers para coleccionar dichos los datos.

Cuando un host está ejecutando un sniffer, se puede decir que es un espía dentro de la red, ya que tiene acceso a la información que no va dirigida directamente a él. El sniffer captura las tramas que circulan por la red, examina las cabeceras, recoge los datos que se incluyen en el campo de información de la trama y lo sube a través del stack(pila) TCP/IP para que dicha información sea tratada por los protocolos de nivel superior. Para ello se coloca la interfaz de red en modo promiscuo, de esta manera toda la información que circula por la red será capturada.

Cuando se utiliza un sniffer, cada vez que la interfaz de red recibe o transmite un paquete, se envía una copia del buffer de la interfaz a un bloque de la memoria del Kernel, entonces, se determina qué tipo de paquetes se han recibido o transmitido.

En la Figura No.1, se ilustra el proceso de captura de paquetes que realiza un sniffer.

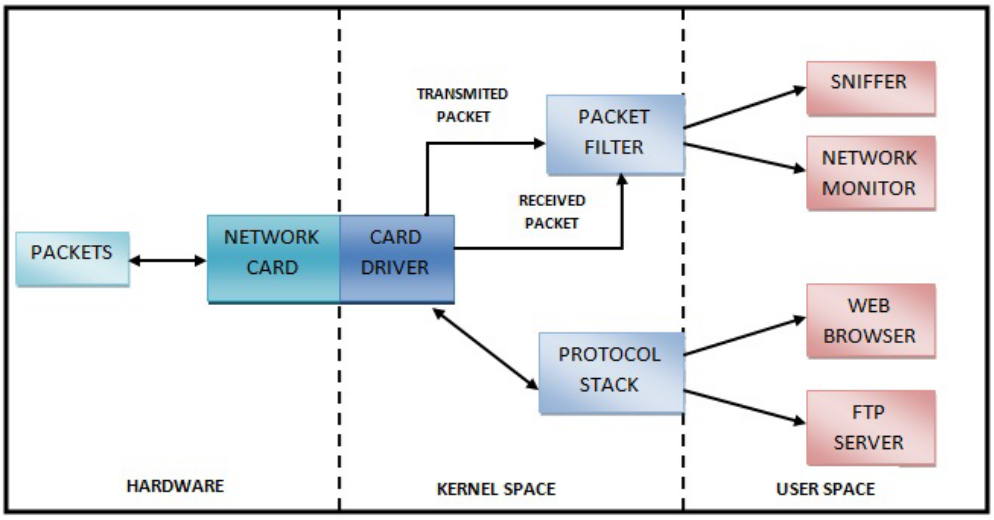


Figura 1: Elementos involucrados en el proceso de captura.

5.2. Libpcap

Libpcap es una librería de código abierto que proporciona una interfaz de alto nivel para la captura de paquetes que circulan a través de una red. Fue creada en 1994 por McCanne, Leres and Jacobson como parte del proyecto para investigar y mejorar TCP[9], así como, el desempeño de internet.

El objetivo principal de Libpcap fue crear una plataforma independiente (API[10]) que elimine la necesidad de módulos de captura de paquetes dependientes del sistema operativo para cada aplicación. La API de Libpcap está diseñada para ser usada desde el lenguaje C y C++, sin embargo, se pueden usar para lenguajes como: Perl, Python, Java o Ruby.

Libpcap es una librería de funciones y los procesos que ejecutan estas funciones, lo hacen en el nivel de usuario (user space). Sin embargo la captura real de datos tiene lugar en las capas más bajas del sistema operativo, en el Kernel.

La Figura No.2, muestra la esquematización de un programa realizado con Libpcap.

[9]TCP: Protocolo de Control de Transmisión/ Transmission Control Protocol.
[10]API: Interfaz de Programación de Aplicaciones / Application Programming Interface

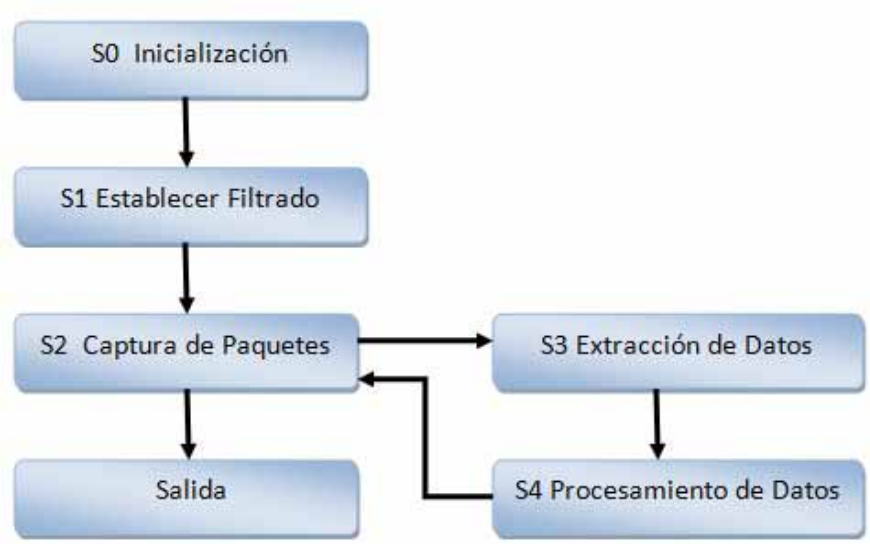


Figura 2: Esquematización de un programa con Libpcap.

5.2.1. S0 Inicialización

La fase de inicialización engloba las funciones capaces de obtener información del sistema: las interfaces de red instaladas, la configuración de estas interfaces (Mascara de Red, Dirección de Red), etc. Las funciones más relevantes para obtener esta información son:

- *char *pcap_lookupdev(char *errbuf)*: Devuelve un puntero al primer dispositivo de red válido para ser abierto para captura, en caso de error se devuelve NULL y una descripción del error en *errbuf*.
- *int pcap_findalldevs(pcap_if_t **alldevsp, char *errbuf)*: Está función nos devuelve todos las interfaces de red que pueden ser abiertas para capturar datos.
- *int pcap_datalink(pcap_t *p)*: Esta función devuelve el tipo de enlace de datos asociado a una interfaz de red.

5.2.2. S1 Establecimiento del Filtrado

En esta fase se establecen los mecanismos de filtrado, para indicar los paquetes que se desean tratar de entre todos aquellos que se reciben. La *expresión* que se usa para definir el filtro tiene una serie de primitivas y tres posibles modificadores. Si la expresión es verdadera, el paquete se pasa a la zona de usuario y si es falsa el paquete se descarta sin llegar a salir en ningún caso de la zona Kernel. Los 3 posibles modificadores son:

- Tipo: Puede ser *host*, *net*, *port*. Para hacer referencia a un host, una red o un puerto.

- DIR: Indica el sentido de la transferencia que interesa. Se representa por *src*, *dst*, para identificar que el identificador referenciado es el *origen* o el *destino*.
- PROTO: Permite especificar paquetes que transportan un cierto protocolo, *ether, arp, ip, tcp, udp*.

5.2.3. S2 Captura de Paquetes

Una vez que se estableció cómo obtener el listado de las interfaces instaladas en el sistema operativo y sus configuraciones, se comienza con la captura. Existen varias funciones para capturar paquetes, las principales diferencias entre ellas son: El número de paquetes que se desean capturar, el modo de captura, normal o promiscuo y la manera en que se definen sus funciones de llamada o Callbacks (la función invocada cada vez que se captura un paquete).

- *pcap_t *pcap_open_live(char *device, int snaplen,int promisc, int to_ms, char *errbuf)*: Prepara las estructuras de datos necesarias para comenzar la captura de paquetes, devolviendo un descriptor que podrá ser utilizado en las restantes operaciones de captura.
- *int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user)*: Esta función procesa tantos paquetes se indiquen.
- *u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h)*: Devuelve un puntero al siguiente paquete que se haya capturado y se tenga almacenado en el buffer de entrada.

5.2.4. S3 Extracción de Datos

Cuando una aplicación quiere enviar datos a través de la red, se añaden las cabeceras de los protocolos que se emplearán en la transmisión, en esta fase, se extraen e interpretan dichas cabeceras.

5.2.5. S4 Procesamiento de Datos

Libpcap ofrece la posibilidad de almacenar los datos en un fichero para procesarlos después, mediante funciones especificas:

- *pcap dumper t *pcap_dump_open(pcap_t *p, char *fname)*: Esta función se utiliza para abrir un fichero de salida en el cual se guardaran los datos que se van capturando.
- *pcap open offline(filename, errbuf)*: Esta función sirve para abrir un fichero que contiene paquetes guardados.

- *u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h)*: Devuelve un puntero al siguiente paquete que se haya capturado y se tenga almacenado en el buffer de entrada.

5.3. SMTP(Simple Mail Transfer Protocol)

El protocolo SMTP(Simple Mail Transfer Protocol), es el protocolo estándar de Internet para el intercambio de correo electrónico. SMTP necesita que el sistema de transmisión ponga a su disposición un canal de comunicación fiable y con entrega ordenada de paquetes, con lo cual, el uso del protocolo TCP (puerto 25) en la capa de transporte, es lo adecuado. Para que dos sistemas intercambien correo mediante el protocolo SMTP, no es necesario que exista una conexión interactiva, ya que este protocolo usa métodos de almacenamiento y reenvío de mensajes.

Éste es un protocolo que funciona en línea, encapsulado en una trama TCP/IP. El correo se envía directamente al servidor de correo del destinatario. El protocolo SMTP funciona con comandos de textos enviados al servidor SMTP, a cada comando enviado por el cliente le sigue una respuesta del servidor SMTP compuesta por un número y un mensaje descriptivo.

Sin embargo, debido a las limitaciones de este protocolo para mantener una cola de mensajes en el servidor que los recibe, es usado normalmente con uno o dos protocolos adicionales, POP3[11] o IMAP[12], con los que se permite al usuario guardar los mensajes en un buzón del servidor y descargarlos periódicamente a su ordenador desde allí.

5.3.1. Modelo SMTP

Tanto el cliente y el servidor SMTP deben tener dos componentes básicos: UA[13] y MTA[14] local.

Hay pocos casos de envío de mensajes de correo electrónico a través de redes. En el primer caso en la comunicación entre el emisor y el receptor a través de la red, el UA del remitente prepara el mensaje, genera el sobre y pone mensaje en el sobre. El MTA transfiere el correo a través la red, y por el puerto 25 del receptor. El segundo caso de la comunicación es entre el host de envió (cliente) y el host receptor (servidor), existe un MTA en el sitio de emisor y uno en el sitio de recepción, otros MTA, en calidad de cliente

[11]POP3 (Post Office Protocol): Un protocolo que se utiliza para recuperar e-mail desde un servidor de correo.

[12]IMAP (Internet Message Access Protocol): Un protocolo de Internet utilizado por los clientes de correo para recuperar mensajes almacenados en los servidores.

[13]UA (User Agent): Un componente SMTP que prepara el mensaje, crea el sobre, y coloca el mensaje en el sobre.

[14]MTA (MAIL TRANSFER AGENT): El software que se ejecuta en un servidor de correo y realiza la entrega de los mismos.

o servidor, puede transmitir el correo electrónico a través de la red.

La Figura No.3 Se muestra un modelo simple de los componentes del sistema de SMTP.

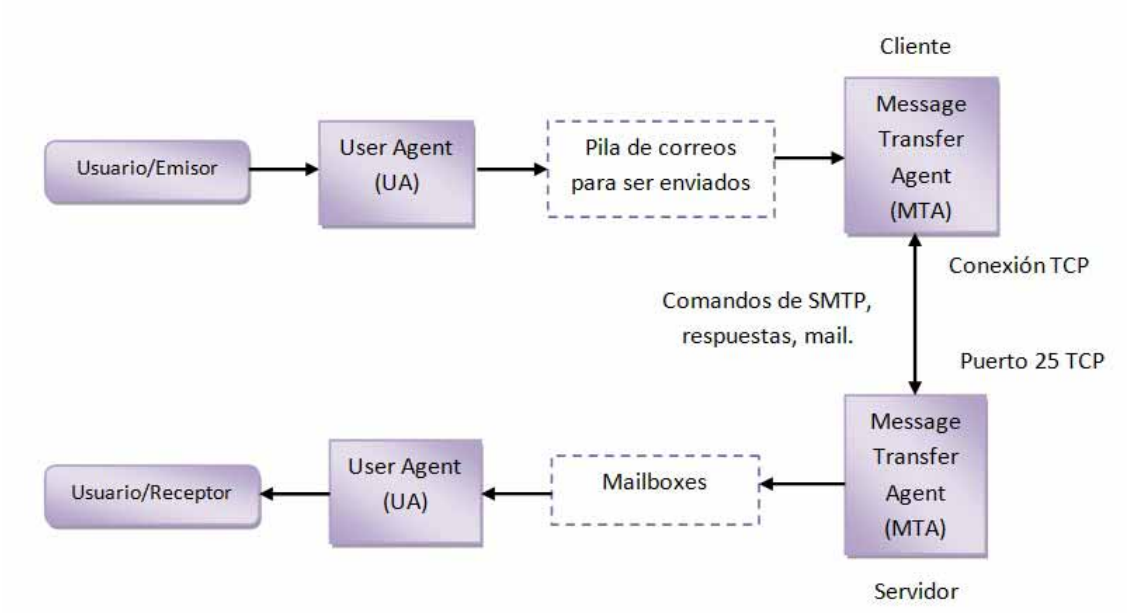


Figura 3: Modelo básico de SMTP.

5.3.2. Comandos y respuestas de SMTP

La transferencia de correo consiste en el intercambio de unidades de datos de protocolo SMTP referidas como comandos. Estos comandos, codificados en cadenas de caracteres ASCII, incluyen:

- Un número de tres dígitos.
- Un comando textual.
- Ambos valores.

En el proceso de apertura de un canal, se produce un intercambio de datos que tiene como fin la identificación de los host. Tras establecer el canal se darán tres pasos para la realización de una transacción SMTP, el primero de ellos se produce para identificar al usuario emisor. A continuación se identifican los usuarios finales a quienes se les quiere hacer llegar el mensaje. El tercer paso es el envió del mensaje de correo en sí.

Las Tablas No.1 y No.2 Muestran algunos comandos y respuesta SMTP respectivamente, para la transacción de un correo.

COMANDOS SMTP

Comandos	Descripción
HELO	Identifica el remitente al destinatario.
MAIL FORM	Identifica una transacción de correo e identifica al emisor.
DATA	Permite enviar una serie de líneas de texto.El tamaño máximo de una línea es de 1000 caracteres.
RSET	Aborta la transacción de correo actual.
QUIT	Cierra la conexión.
SEND	Identifica una transacción de correo e identifica al emisor.

Cuadro 1: Comandos de SMTP

RESPUESTAS SMTP

Código	Descripción
211	Indica el estado del sistema.
220	<domain>. Servicio preparado.
221	Servicio cerrando el canal de transmisión.
250	Solicitud completada con éxito.
354	Indroduzca el texto.
251	Usuario no local, se enviará a....

Cuadro 2: Respuestas de SMTP

5.3.3. Modo de comunicación SMTP

Cuando un servidor de SMTP, requiere transmitir un mensaje a otro servidor SMTP, el emisor (servidor que inicia la sesión SMTP) establece una conexión con el receptor (servidor que recibe petición de establecer sesión SMTP). Esta conexión es unidireccional, es decir, el emisor puede enviar correo al receptor, pero durante esa conexión, el receptor no puede enviar correo al emisor. Si el receptor tiene que enviar correo al emisor, tiene que esperar a que finalice la conexión establecida y establecer otra en sentido contrario, cambiando los papeles de emisor y receptor.

Una vez establecida la conexión, el emisor envía comandos y mensajes. Por lo tanto, el diseño de SMTP se basa en el siguiente modelo de comunicación:

1. Como respuesta a una solicitud de un usuario de enviar un correo electrónico, el emisor SMTP establece una conexión con el receptor SMTP. El receptor SMTP debe ser el destinatario último del correo o un intermediario. Para ello el emisor genera los comandos SMTP en formato ASCII y los envía al receptor y el receptor genera las respuestas a los comandos enviados por el emisor.

2. Una vez establecido el canal de transmisión, el emisor envía el comando MAIL para indicando que él es el emisor del correo. Si el receptor puede aceptar correo responde con el comando OK.

3. El emisor envía el comando RCPT identificando el destinatario del correo. Si el receptor puede aceptar correo para ese destino responde con una respuesta OK; si no, responde rechazando el correo para ese destino.

4. Una vez negociado el destino, el emisor comienza a enviar datos (cabeceras y cuerpo), terminando con una secuencia especial. Si el receptor ha procesado correctamente los datos, responde con el comando OK.

La Figura No.4 muestra el modo de comunicación SMTP.

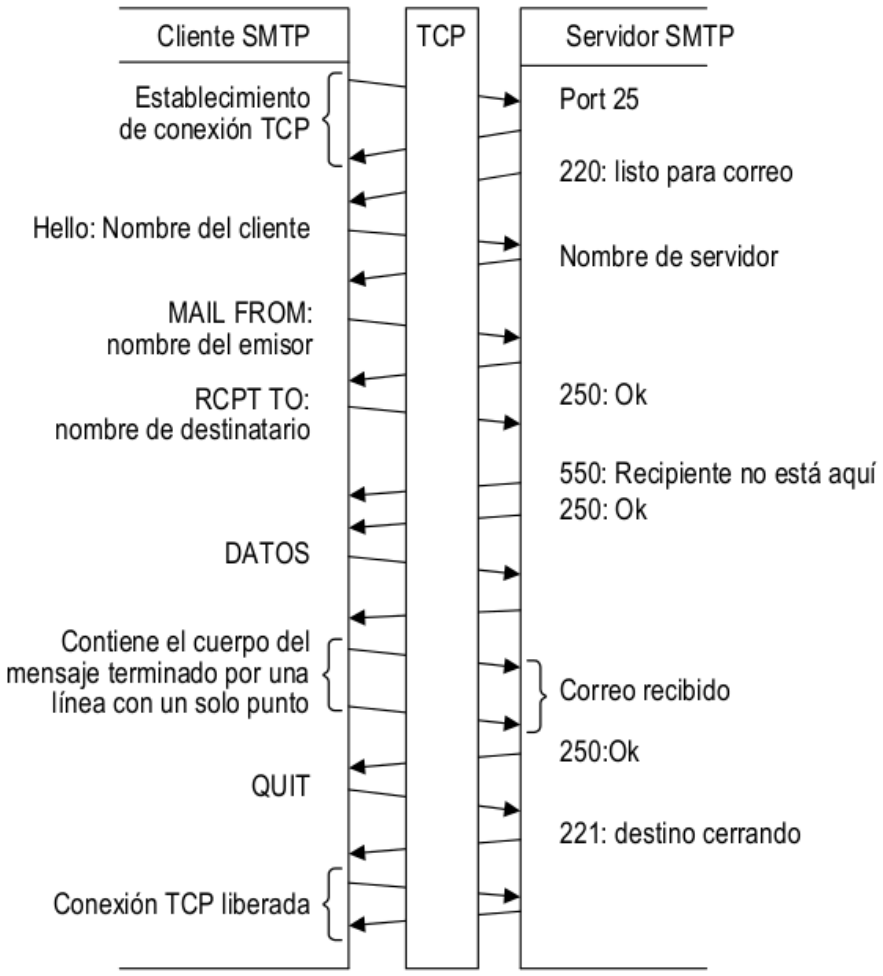


Figura 4: Secuencia de intercambio de comandos del protocolo SMTP.

5.4. POP3 (Post Office Protocol)

EL protocolo POP(Post Office Protocol) versión 3 es utilizado para transferir correo desde un servidor hasta una estación de trabajo. El servicio POP3 se encuentra disponible de forma estándar en el puerto TCP 110. Cuando un cliente desea utilizar dicho servicio, establece una conexión TCP con el servidor.

Puesto que con SMTP lo único que se consigue es la transferencia del correo entre buzones, es necesario un protocolo como POP (o también IMAP) con el que se podrá descargar el mensaje desde el buzón.

5.5. IMAP (Internet Message Access Protocol)

El IMAP4 (Internet Message Access Protocol) es un protocolo cliente/servidor que permite el acceso y manipulación de los mensajes de correo electrónico en el servidor, así como de archivos de mensajes remotos, llamados "casillas electrónicas", de modo similar a las casillas postales. Permite que un cliente fuera de línea resincronize sus casillas electrónicas con el servidor.

- IMAP permite administrar diversos accesos de manera simultánea
- IMAP permite administrar diversas bandejas de entrada
- IMAP brinda más criterios que pueden utilizarse para ordenar los correos electrónicos

5.6. DNS(Domain Name Service)

Un sistema de nombres de dominio como DNS(Domain Name Service) es necesario para poder referenciar los recursos de red mediante nombre que sean cómodos de manejar por los usuarios. El sistema de nombre traslada estas cadenas proporcionadas por los usuarios a direcciones de red manejables por los programas y dispositivos de red.

El sistema de nombres de dominio (DNS) es un directorio que utiliza SMTP para convertir un nombre, a una lista de servidores que pueden recibir conexiones de ese nombre y para encontrar la dirección IP de un servidor específico. Al buscar la dirección de un servidor de destino en el DNS, el servidor de envío puede enrutar correctamente un mensaje a un destinatario. DNS utiliza dos tipos de registros: registros Mail Exchanger(MX) y un registro A. Un registro MX asigna un nombre de dominio a los nombres de uno o más servidores de correo. Un registro A asigna un nombre de host a la dirección IP de un servidor.

5.7. TCP (Transmission Control Protocol)

El Protocolo de Control de Transmisión (TCP) es el método más eficiente y seguro de mover el tráfico de red entre un cliente y un servidor. En las conexiones TCP, cuando un cliente decide establecer una comunicación con un servidor, es necesario que ambos estén de acuerdo en participar, de lo contrario la comunicación no se podrá llevar a cabo.

Este es uno de los protocolos más importantes en internet, es un protocolo orientado a conexión y asegura que los datagramas IP enviados lleguen correctamente a su destino y los retransmite si es necesario. En la recepción de los datagramas, se encarga de ensamblarlos en mensajes con la secuencia apropiada.

Este protocolo garantiza a SMTP la fiabilidad y retransmisión de información para la correcta comunicación entre los participantes.

En la Figura No.5 se muestra la Pila de protocolos de internet y su iteración con SMTP.

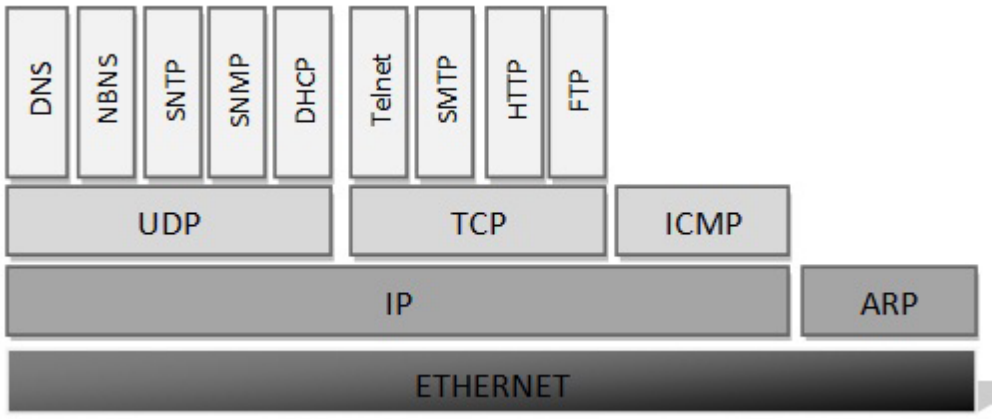


Figura 5: Pila de protocolos de Internet.

5.8. Iptables

Iptables es el componente más conocido del proyecto Netfilter(que comenzó en 1998 con Rusty Russell) y es una herramienta que funciona en el espacio de usuario y que permite definir reglas para el filtrado y la modificación de paquetes TCP/IP que pasen por cualquiera de las interfaces de red de un equipo.

Iptables realiza dos funciones principalmente, filtrado de paquetes y traducción de direcciones de red (Network Address Translation (NAT)), permite al administrador del sistema definir reglas acerca de qué hacer con los paquetes de red. Iptables es el soporte a seguimiento de conexiones (connection tracking), ya que permite actuar como firewall, mejorando de forma considerable la seguridad. Por ello, cuando un paquete llega a la pila

de red, el kernel consulta las reglas de iptables, y actúa de acuerdo a ellas.

5.8.1. NAT (Network Address Translation)

Iptables provee dos tipos de traducción de direcciones:

- SNAT (Source Network Address Translation) Para traducir direcciones origen. Su uso común es permitir que una red con UPS privadas acceda a internet a través de una IP pública.
- DNAT (Destination Network Address Translation) está diseñado para convertir direcciones de destino. Este tipo de traducción nos permite situar distintos servicios en distintos servidores compartiendo una dirección IP.

5.8.2. Tablas

Iptables está construida por cuatro tablas: *filter*, *mangle*, *nat* y *raw*,cada una de las cuales contiene ciertas cadenas predefinidas.

En la Tabla No.3 se definen las principales características de cada tabla.

Tabla	Descripción
NAT	Esta tabla permite realizar la traducción de direcciones, pudiendo traducir tanto la dirección de origen como destino. Los objetivos válidos en esta tabla son SNAT, DNAT, MASQUERADE o REDIRECT.y cadenas predefinidas: PREROUTING, POSTROUTING y OUTPUT.
FILTER	Esta tabla es usada para filtrar paquetes, con los objetivos ACCEPT, DROP y REJECT y con cadenas predefinidas: FORWARD, INPUT y OUTPUT.
MANGLE	Esta tabla esta diseñada para manipular paquetes. Ciertos objetivos sólo son aplicables en esta tabla:TOS, TTL, MARK, SECMARK y CONNSECMARK. Con ellos podemos modificar los paquetes o marcarlos para luego realizar controles de seguridad. Cadenas predefinidas: PREROUTING, POSTROUTING, OUTPUT, INPUT y FORWARD.
RAW	Esta tabla permite marcar paquetes que no serán controlados con estado de conexión. Para ello se usa el objetivo NOTRACK y las cadenas PREROUTING o OUTPUT.

Cuadro 3: Tablas de Iptables

5.8.3. Cadenas

Las cadena son un conjunto de reglas que son analizadas de forma ordenada. Existen un conjunto de cadenas predefinidas que son usadas por defecto por iptables, y además

se pueden definir nuevas cadenas.

Las cadenas definidas por el usuario tienen que ser llamadas explícitamente. A una cadena predefinida se le puede asociar una política predeterminada cuando el paquete acaba de recorrerla. Las políticas válidas son ACCEPT o DROP.

5.8.4. Reglas

La regla especifica qué propiedades debe tener el paquete para que la regla coincida, como un número de puerto o dirección IP. Si la regla no coincide, se continúa con la regla siguiente. Si la regla, por el contrario, coincide con el paquete, las instrucciones de destino de las reglas se siguen (y cualquier otro procesamiento de la cadena normalmente se aborta). Algunas propiedades de los paquetes solo pueden examinarse en ciertas cadenas, algunos destinos solo pueden usarse en ciertas cadenas y/o en ciertas tablas.

5.8.5. Destinos/Objetivos

Los destinos se utilizan para especificar la acción que se realizará cuando una regla coincide con un paquete y también para especificar las políticas de la cadena. Existen distintos destinos que se construyen en iptables.

La Tabla No.4 describe los destinos incorporados.

Destino	Descripción
ACCEPT	Este destino hace que Iptables acepte el paquete, es decir, deja pasar el paquete a la siguiente etapa de procesamiento.
DROP	Este destino hace que Iptables descarte el paquete sin ningún otro tipo de procesamiento.
REJECT	Este destino hace que Iptables descarte los paquetes enviando un mensaje de error al equipo de origen del paquete.
LOG	Este destino irve para obtener registros con información detallada sobre los paquetes que cumplan cierta regla.
SNAT	Este destino se usa para realizar SNAT (traducción de direcciones de origen).
MASQUERADE	Este destino actúa como SNAT, pero sin dirección de origen, ya que esta diseñado para conexiones DHCP.
DNAT	Este destino se usa para realizar DNAT (traducción de direcciones de destino).
QUEUE	Este destino hace que el paquete sea enviado a una cola en el espacio de usuario.
RETURN	Hace que el paquete en cuestión deje de circular, por la cadena en cuya regla se ejecutó el destino RETURN.

Cuadro 4: Destinos de Iptables

5.8.6. Seguimiento de Conexiones

El seguimiento de conexiones le permite al Kernel llevar cuenta de todas las conexiones o sesiones lógicas de red y de este modo relacionar todos los paquetes que pueden llegar a formar parte de esa conexión. La traducción de dirección de red (NAT) depende de esta información para traducir todos los paquetes relacionados de la misma manera e iptables puede usar esta información para actuar como un firewall.

En la Tabla No.5 se describen los cuatro estados para la clasificación de los paquetes:

Estado	Descripción
NEW	Intentando crear una conexión nueva.
ESTABLISHED	Parte de una conexión ya existente.
RELATED	Relacionada, aunque no realmente parte de una conexión existente.
INVALID	No es parte de una conexión existente e incapaz de crear una conexión nueva.

Cuadro 5: Estados de Iptables

En la Figura No.6 se muestra un ejemplo de tráfico de paquetes con filtrado iptables.

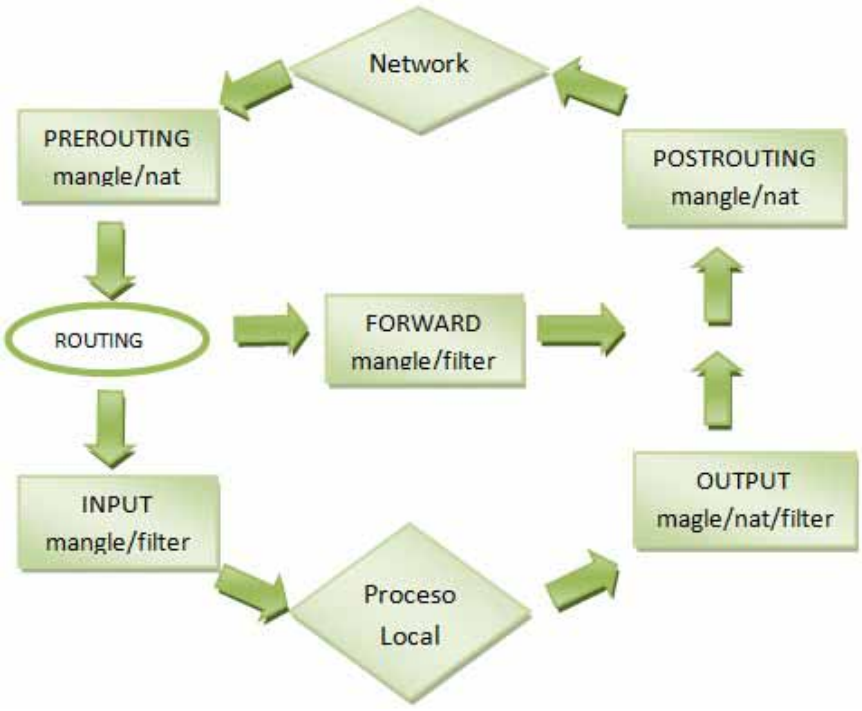


Figura 6: Filtrado de paquetes con Iptables.

5.9. Qmail

Qmail fue desarrollado por Daniel J. Bernstein entre 1995-98, en la búsqueda de una mejor alternativa a Sendmail, y otros MTAs. Qmail es un Agente de Transporte de Correo (MTA) para sistemas operativos tipo UNIX, utiliza: el Simple Mail Transfer Protocol (SMTP) para intercambiar mensajes con los MTA de otros sistemas, el formato *maildir*[15] para almacenar mensajes (un archivo por mensaje), eliminando varios problemas asociados al manejo del formato *mbox*[16].

5.9.1. Características

Configuración

- Separación nítida entre direcciones, ficheros y programas.
- Reducción al mínimo del código que se ejecuta como root.

Construcción de mensajes

- Longitud de línea de la cabecera limitada sólo por la memoria disponible
- Enmascaramiento de máquina y de usuario.

Servicio SMTP

- No hay interferencia entre el control de relay (control de retransmisión) y los alias de correo.
- Reconocimiento automático de direcciones IP locales.
- Retransmisión (relaying) y reescritura de mensajes para clientes autorizados.

Gestión de colas

- Tratamiento instantáneo de los mensajes que se agregan a la cola.
- Control automático para cada destinatario.
- Visionado de las colas.

Rutados por dominios

- Cualquier número de nombres para la maquina local.
- Cualquier número de dominios virtuales

[15]Maildir: Es un formato de correo electrónico que no bloquea los ficheros para mantener la integridad del mensaje, porque los mensajes se almacenan en ficheros distintos con nombres únicos.

[16]Mbox: Es un término genérico para una familia de formatos de fichero que se usan para almacenar conjuntos de correos electrónicos. Todos los mensajes en un buzón mailbox están concatenados en un único fichero.

5.9.2. Módulos de qmail

Qmail es modular. Cada una de estas funciones la lleva a cabo un programa separado. Como resultado, los programa son mucho más pequeños, más sencillos, y menos tendentes a contener fallos funcionales o de seguridad. Para incrementar la seguridad aún más allá, los módulos de qmail se ejecutan con diferentes privilegios.

En la Tabla No.6 se muestran los módulos centrales de qmail.

Módulo	Función
qmail-smtpd	Acepta/rechaza mensajes vía SMTP.
qmail-inject	Construye un mensaje y lo inyecta localmente usando qmail-queue.
qmail-queue	Coloca un mensaje en la cola
qmail-rspawn/qmail-remote	Gestión entregas remotas.
qmail-lspawn/qmail-local	Gestión entregas locales.
qmail-send	Gestiona la cola de correo y se encarga de hacer el envío tanto local como remoto.
qmail-clean	Limpia la cola.

Cuadro 6: Módulos centrales de qmail.

5.9.3. Proceso de envió de un mensaje

A continuación se describen algunos de los posibles recorridos que realiza un mensaje:

1. Origen local: Un usuario Linux operando en el sistema, remite un mensaje usando un MUA[17] como mail. Esto normalmente originará una llamada a qmail-inyect el cual llevará el mensaje al programa de encolamiento qmail-queue. Este lo almacenará en la cola de mensajes /var/qmail/queue. Acto seguido, qmail-send intentará remitirlo a su destino mediante los programas qmail-lspawn o qmail-rspawn.

2. Origen remoto exterior: Un usuario de Internet ha enviado un mensaje hacia nuestra red. Este mensaje debe provenir de otro servidor de email mediante SMTP. qmail-smtpd descubre que el destinatario pertenece a la red, y acepta el mensaje, el cual pasa a la fase de encolamiento.

3. Origen remoto de la LAN: Un cliente de la red desea enviar un mensaje. Para esto ha configurado su MUA a fin de remitir hacia el servidor qmail. Esta remisión normalmente se efectúa usando SMTP, y por tanto nuevamente qmail-smtpd es el encargado de la recepción.

4. Destino local: Los mensajes con destino local son guardados en el mailbox o qmail-local para ser recogidos por los MUAs de los usuarios que trabajan en el servidor.

[17]MUA (Mail User Agent): También conocido como el cliente de correo, se utiliza para leer, archivar, enviar, y procesar e-mail,

En el caso de que el usuario no esté en el servidor, sino, en una estación de trabajo, entonces su MUA deberá conectarse a un servidor IMAP o POP para obtener los mensajes del mailbox. En cualquier caso, qmail-lspawn es el encargado de controlar el agente de delivery local.

5. Destino remoto: Los mensajes remotos se remiten con SMTP hacia otros MTA. qmail-rspawn es el encargado de los mensajes que se envían remotamente.

La Figura No.7 proporciona una visión esquemática de estos módulos que constituyen el sistema de correo electrónico.

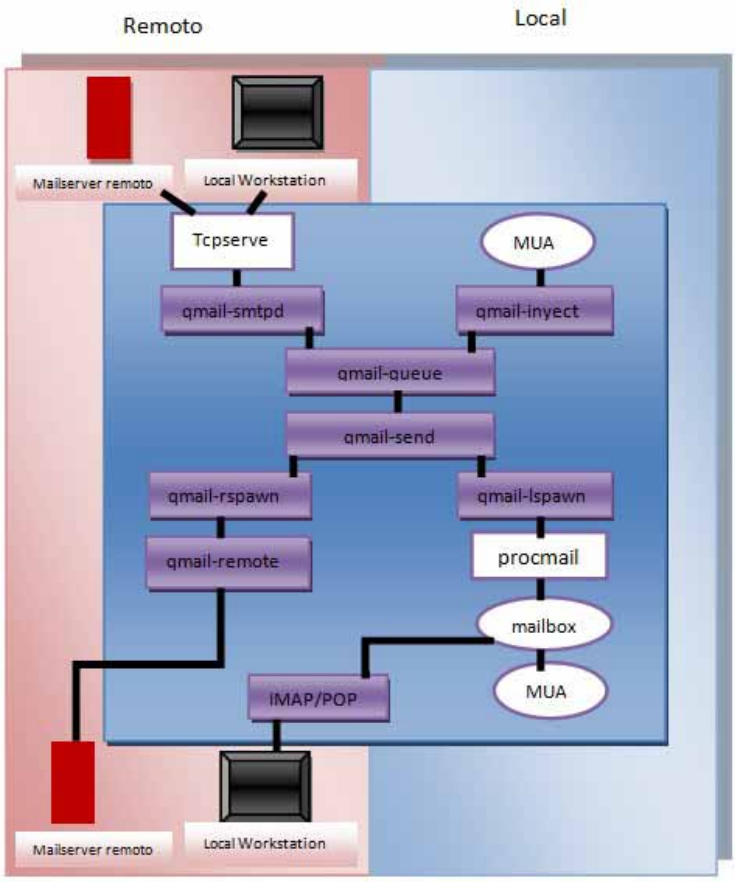


Figura 7: Arquitectura de qmail.

6. Desarrollo Técnico

El sistema de seguridad que se desarrolló permite filtrar los correos electrónicos que contengan información clasificada como confidencial, para lo cual se toma como base un patrón de datos generado por dicha información.

6.1. Desarrollo General

En el proyecto se implementó el análisis de los correos electrónicos que envían los usuarios de una red, hacia los usuarios de otra red, debido a las condiciones que establecen algunos servidores de correos (Gmail, Hotmail) para la recepción de correos electrónicos, al detectar un dominio no conocido, se clasifican dichos correos como *Spam*[18], por tal motivo se implementó una red conformada por dos dominios distintos.

En la Figura No.8 se muestra la implementación de la red LAN, con los componentes que se necesitan para el desarrollo y el funcionamiento del sistema de seguridad.

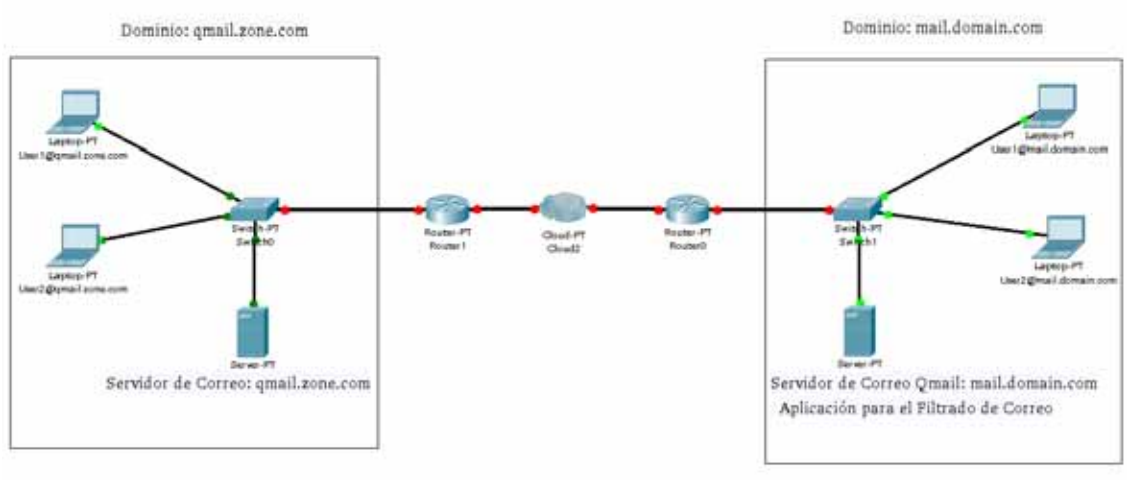


Figura 8: Maqueta de comunicación y pruebas.

El equipo de cómputo en el cual se implementó el sistema de seguridad, cuenta con una distribución GNU/Linux, que proporcionó soporte para la librería jNetPcap. La cual, por medio de diferentes funciones proporcionaron las características necesarias para la captura de los paquetes que navegan por la red.

Como se muestra en la figura anterior, existen dos dominios en los cuales se genera el envió de los correos electrónicos. El servidor de correos local*(mail.domain.com)* fue el encargado de gestionar los mensajes que se envían o reciben dentro de su dominio, de modo que si un usuario desea enviar un correo, el servidor determina si pertenece a su dominio, lo almacena en su cola, para después envíar el correo a su destino.

Los clientes de la red corporativa, donde se analizan los correos enviados, podrán solicitar el acceso al servidor de correos, de modo que cuando se detecta una petición de conexión hacia el puerto 25 (puerto donde se reciben peticiones SMTP), se autentifica su acceso, determinando la red a la que pertenece y hacia donde será enviada su petición, lo cual genera que la información que se desea enviar sea redirigida hacia la aplicación de

[18]Spam: correo basura o mensaje basura, mensajes no solicitados, no deseados o de remitente no conocido (correo anónimo)

filtrado de correos.

Cuando el acceso es permitido y la conexión se establece, utilizando iptables se definen las reglas para denegar el envío de datos a través de la interfaz de salida, lo que permite capturar el paquete al mismo tiempo que se envía el correo hacia el servidor de correos, es decir, analizar la información la mismo tiempo que se esta enviando. La información capturada es analizada con base al patrón que indica si dicho correo contiene o no información confidencial.

La información analizada, no solo se basará en el mensaje, sino que tomará en cuenta la información del asunto del mensaje, por lo que, si el contenido coincide con el patón de datos, la información es eliminada, y el correo se envía como un correo vació, ya que no contendrá asunto y mensaje, en caso contrario el correo es enviado tal y como se capturo.

El patrón de datos utilizado, depende de las especificaciones y requerimientos de la empresa, por lo cual dependiendo a la información que se desea capturar de ingresaran las palabras claves que detectarán el filtrado de información.

El administrador del sistema de seguridad, podrá consultar los correos en los que su envió fue denegado, ya que la aplicación le permitirá tener un registro de los correos cancelado, así como la información detallada, dirección del destinatario, nombre del receptor y el contenido del mensaje, con esto se obtendrá la información necesaria para la generación de estadísticas.

En la Figura No.9 se muestra el diagrama de tiempos de cómo se lleva a cabo el proceso de filtrado delos correos, desde la autentificación del usuario hasta que el mensaje es enviado.

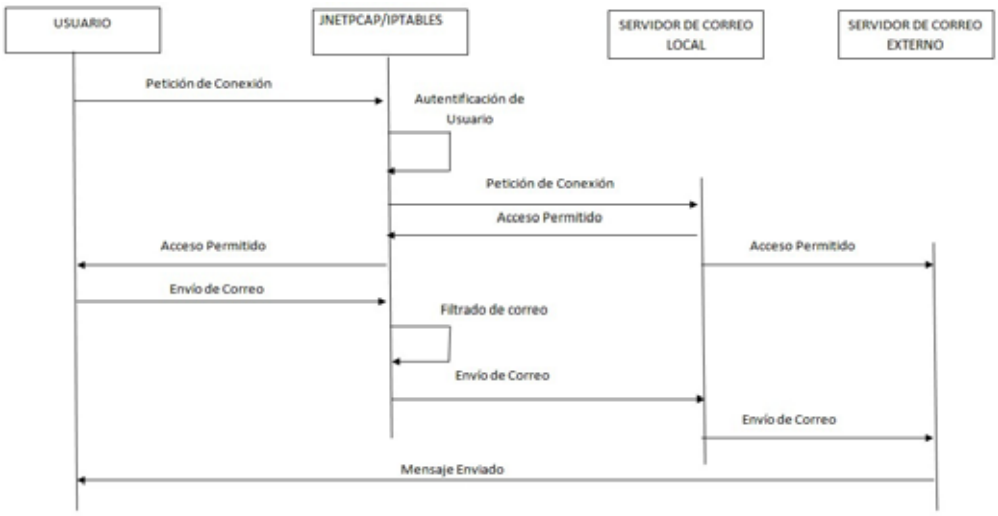


Figura 9: Diagrama de tiempos del proceso de filtrado de correos.

6.2. Módulos de desarrollo

Según los objetivos, para generar el sistema de seguridad, el proyecto se dividió entre los siguientes módulos:

- Módulo de clasificación de direcciones IP.
- Módulo de administración del servidor de correos.
- Módulo de filtrado de correos electrónicos.
- Módulo de generación de alertas.
- Módulo de estadísticas.

En la Figura No.10 se muestra un diagrama general de los módulos que integran el sistema de seguridad.

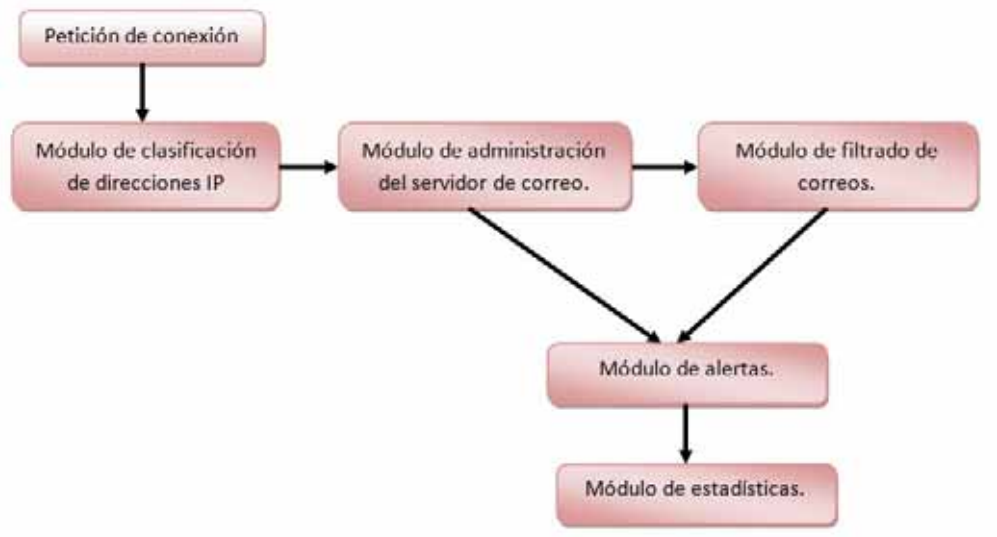


Figura 10: Módulos del sistema de seguridad.

6.2.1. Módulos de clasificación de direcciones IP

Para generar la clasificación de las direcciones IP, en las cuales se establecerá el filtrado se correos, se utilizó la herramienta de **iptables**, la cual nos permitió redirigir el tráfico de la red hacia el host que contienen el programa de filtrado.

Para establecer las reglas de direccionamiento se tomará en cuenta:

- La dirección IP del host o red, hacia la cual se enviará el correo electrónico (dirección destino).
- La dirección IP del host o red, que envía el correo electrónico (dirección fuente).
- El puerto destino al cual llegará el correo electrónico (en nuestro caso, puerto 25).
- La interfaz de salida en la cual viajará el correo electrónico.

La siguiente imagen muestra las reglas que se establecieron para el redireccionamiento del tráfico hacia el host del filtrado.

```
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o wlan0 -j MASQUERADE
iptables -t nat -A POSTROUTING -s 192.168.8.0/24 -o wlan0 -j MASQUERADE
```

6.2.2. Módulo de administración del servidor de correo Qmail

Para gestionar los correos que son enviados y recibidos, en nuestra red, se utilizó el servidor de correos de qmail, el cual nos permite:

- Definir los usuarios y grupos con los que qmail trabajará.
- Definir el nombre de dominio de nuestra red.
  ./config-fast qmail.zone.com
- Seleccionar el tipo de buzón de correo predeterminado, en el cual almacenar los correos(Mbox, Maildir, etc).
  echo ./Maildir/ >/var/qmail/control/defaultdelivery
- Definir el control de acceso de SMTP, para que los host puedan enviar mensajes con nuestro dominio.
  echo '127.:allow,RELAYCLIENT='>>/etc/tcp.smtp
  qmailctl cdb
- Crear el sistema de los alias

6.2.3. Módulo de filtrado de correos electrónicos

Proceso de captura de paquetes

- **Java Package: org.jnetpcap**: Es el paquetes que provee las funciones necesarias para establecer la captura de los paquetes de red. Los pasos de programación para trabajar con este paquete son:

1. Configurar un buffer de error: Permitirá establecer un buffer para almacenar los errores detectados durante la captura de paquetes. La Figura No.11 muestra la instrucción necesaria para establecer el buffer para los mensajes de error.

Figura 11: Configuración del buffer de error.

2. Obtener una lista de todas las interfaces disponibles y elegir una (Pcap.findAllDevs()): Si desea capturar los paquetes directamente desde una red activa, primero se debe adquirir una lista de las interfaces de red disponibles y luego elegir la que desee abrir. En la lista proporcionada por *Pcap.findAllDevs()* se adquiere el nombre de la interfaz de usuario o la aplicación en la que se desea capturar. Podemos obtener el nombre de la interfaz mediante *PcapIf.getName()*. La imagen No.12 y 13 muestran el método para obtener la lista de interfaces activas, así mismo el nombre de las mismas.

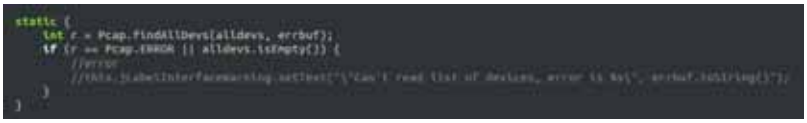


Figura 12: Obtención de la interfaz activa.

Figura 13: Método para obtener el nombre de la interfaz.

3. Abrir una interfaz de red activa, o abrir una conexión, un archivo de captura (Pcap.openLive() o Pcap.openOffline()). Una vez que hemos elegido una interfaz de red, es hora para la siguiente etapa, que consiste en abrir la interfaz de red para la lectura. La principal llamada que hace esto es Pcap.openLive ()

y será la función con la cual se trabajará en la implementación de esté proyecto.

Este método toma una serie de parámetros, pero el más importante es el nombre de la interfaz de red. En la imagen No. 14 se muestra la implementación de está función.

```
public void setup(String name) {

    pcap = Pcap.openLive(name, snaplen, flags, timeout, errbuf);


    PcapBpfProgram program = new PcapBpfProgram();
    String expression = "dst port 25 and net 192.168.1.0/24";
    //String expression = "src port 8084  dst port 8084";
    //String expression = "host 192.168.26.32";
    int optimize = 0;
    int netmask = 0xFFFFFF00; /// Revisar que hongo

    if (pcap.compile(program, expression, optimize, netmask) != Pcap.OK) {
        System.err.println(pcap.getErr());
        return;
    }

    if (pcap.setFilter(program) != Pcap.OK) {
        System.err.println(pcap.getErr());
        return;
    }

    if (pcap == null) {
        System.err.printf("Error while opening device for capture: "
                + errbuf.toString());
        //return;
    }
}
```

Figura 14: Método para obtener abrir una interfaz.

Donde:

name: Es el nombre de la interfaz.
snaplen: Establece el número máximo de bytes para capturar desde la red.
flags: Modo en el cual se configura la interfaz de red, el parámetro que se utiliza es *MODE_PROMISCOUS*, ver todo el tráfico de red que su tarjeta de red puede capturar.
timeout: El tiempo de espera de lectura.
errbuf: Esperar un mensaje de error presente en la memoria intermedia.

Es posible configurar un filtro de paquetes directamente en la captura pcap y sólo tener paquetes entregados a nuestra aplicación y que coinciden o pasen la expresión de filtro.

El filtro está construido a partir de una expresión textual, mediante la sintaxis de filtro de libpcap. La expresión, se representa como una cadena, el filtro se compila mediante *PcapBpfProgram()* que contiene la representación binaria de la expresión de filtro, y luego el programa bpf se encuentra en la captura PCAP utilizando textslPcap.setFilter().

4. Leer ya sea un paquete a la vez (Pcap.nextEx()) o configurar un bucle de envío

(Pcap.loop() o Pcap.dispatch()): En el sistema de seguridad a implementar se necesitan leer demasiados paquetes de la manera más eficiente posible. Libpcap proporciona 2 funciones (Pcap.loop() y Pcap.dispatch()) para configurar los bucles de envío. En la imagen No. 15 se muestra la implementación de la función que genera los bucles de envío.

```
public void jPcapLoop() {
    pcap.loop(-1, jpacketHandler, "");
}
```

Figura 15: Método para bucles de envío.

5. Si está usando un loop de despacho, espera en el método de devolución de llamada (.nextPacket ()) y recibir paquetes entrantes: Por esa razón varios manejadores de devolución de llamada diferentes están disponibles, en este caso ocuparemos: . ya que tratan con objetos de paquetes, proporciona capacidades de decodificación de buffer de paquetes, y proporciona un acceso más avanzada en el paquete para la manipulación en el mismo. La imagen No. 16 muestra la implementación de para recibir correos.

```
public JNetPcap() {
    this.jpacketHandler = new PcapPacketHandler<String>() {
        @Override
        public void nextPacket(PcapPacket packet, String user) {
            Tcp tcp = new Tcp();
```

Figura 16: Método para recibir paquetes.

6. Siempre el último paso es cerrar pcap (Pcap.close ()) para permitir liberar todos los recursos. La imagen No. 17 muestra los métodos para cerrar Pcap.

```
public void jPcapBreakLoop() {
    pcap.breakloop();
    //pcap.close();
}

public void jPcapClose() {

    pcap.close();
}
```

Figura 17: Finalizar Pcap.

Proceso de filtrado de paquetes

Para establecer las reglas de filtrado, se necesitó establecer el patrón de datos con el cual se compara el mensaje que es enviado dentro del correo electrónico, y con el cual se realizó el filtrado de los mismos.

El patrón de datos se establece de acuerdo a las condiciones y especificaciones de la empresa, y en la que se estableció una sección para ingresar las palabras que hacen relación a la información que se desea detectar.

Para esto se realizó la implementación de un método, en el cual una vez que se captura el paquete, se extrae el mensaje y el asunto de correo para después poderlo comparar con el patrón de datos y determinar si el mensaje se clasifica como válido o inválido.

En la imagen No. 18, se muestra la implementación del método de filtrado.

```
public void validate(String string) {

    StringTokenizer st = new StringTokenizer(string);
    List<String> messageTokens = new ArrayList();

    while (st.hasMoreTokens()) {
        messageTokens.add(st.nextToken());
    }
    for (String invalidToken : this.listInvalidWords) {

        for (String messageToken : messageTokens) {
            System.out.println("--" + invalidToken + "--" + messageToken);
            if (messageToken.contains(invalidToken)) {
                this.isValid = false;
                break;
            }
        }
    }
}
```

Figura 18: Filtrado de correos.

6.2.4. Módulo de generación de alertas

El módulo de generación de alertas se relaciona con las acciones a realizar cuando se detecta que un correo electrónico contiene información confidencial, el uso de las iptables permitió denegar o aceptar el acceso para el envió de los mismos.

A todos los correos, cuando son analizados por el sistema de seguridad, se les deniega el acceso para continuar su envió, esto mientras se realiza el análisis, una vez que se detecta el tipo de información contenida y se realiza la clasificación, si el correo no contiene información confidencial, se permite el envió a la red, sin realizar modificación alguna.

En caso contrario, si el correo contiene información confidencial, tanto el mensaje como el asunto, son eliminados del mismo para después ser enviado como un mensaje vació. De esta forma se garantiza que la información no sea enviada a la red.

6.2.5. Módulo de estadísticas

El módulo de estadísticas permite visualizar la información necesaria para general los reportes de correos bloqueados. El administrador podrá accesar a los registro de los correos, es decir, el sistema de seguridad esta configurado para que se tenga acceso a los correos que son bloqueados, los correos que contienen información confidencial, en dichos registro se puede consultar, la dirección del emisor, el nombre del receptor, así como el

mensaje que fue enviado en el correo.

Esto permitirá obtener tener un control de los usuarios que envían información confidencial y dependiendo de los requerimientos de la empresa, establecer las acciones correspondientes, al grado de riesgo que causaría la información si fuera enviada.

6.2.6. Proceso de filtrado

El proceso de filtrado, es aquel que se lleva a cabo cuando un se captura un paquete, el cual consiste en los siguientes pasos:

1. El primer paso consiste en determinar si el destino del paquete es el puerto 25, si es verdadero se bloquea la interfaz, de lo contrario el paquete sigue hacia su destino.

2. Ena vez bloqueada la interfaz de salida, el siguiente paso es determinar si el dominio del paquete pertenece al dominio del servidor local, es decir el usuario pertenece al dominio del servidor, en caso de ser verdadero, la interfaz de salida se desbloquea y el paquete se envía tal y como se capturo, en caso negativo, el paquete pasa a la sección de análisis.

3. En la fase de análisis, se extrae el asunto y mensaje del paquete, y se compara con el patrón de datos.

4. A continuación se determina si el paquete coincide con el patrón de, si es falso, se debloquea la interfaz de salida y el paquete se envía a su destino, sin modificación alguna. En caso verdadero, se bloque la IP fuente, para que no pueda volver a enviar correos, hasta que el administrador le permita nuevamente el acceso.

5. Cuando se determina que el paquete contiene información confidencial, se envía el correo al administrador informando que el correo se rechazo y se adjunta el contenido del mismo, se toman los datos del mensaje original y se eliminan los datos del asunto y del mensaje, se desbloquea la interfaz de salida, para después envía el correo como vació.

En la siguiente Figura No.19 se muestra un diagrama de flujo de la secuencia que recorre el correo electrónico, cuando es capturado por el sistema de seguridad.

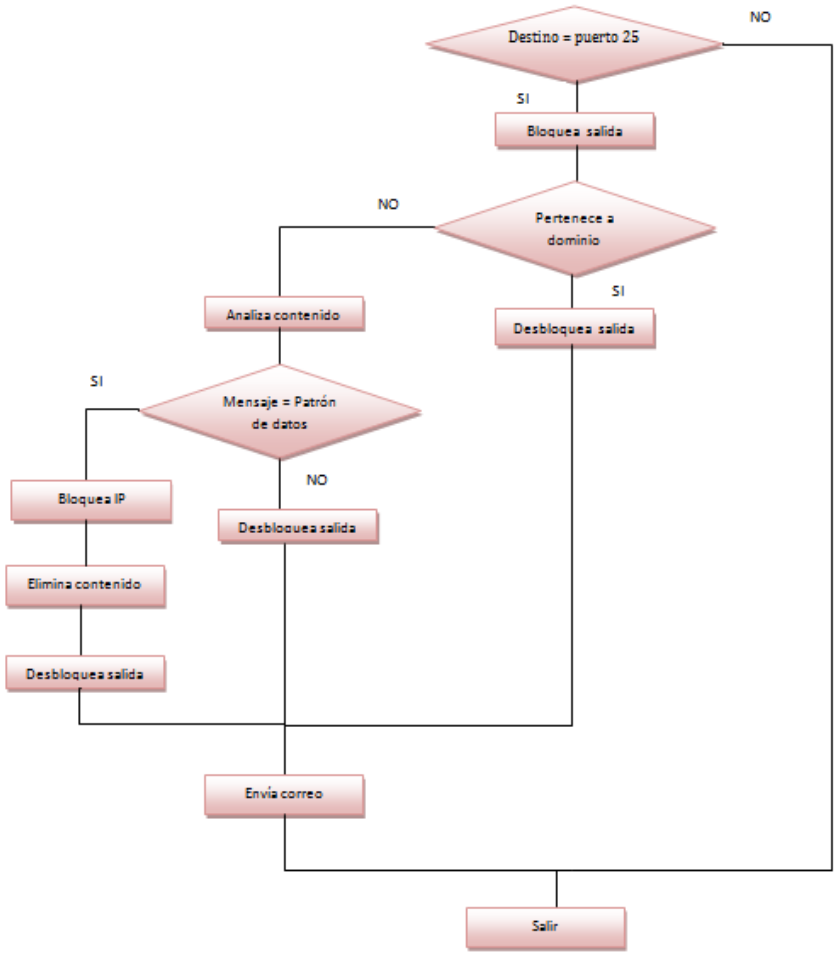


Figura 19: Diagrama de flujo del proceso de filtrado.

7. Especificación Técnica

7.1. Investigación Previa

El primer paso para el desarrollo de este proyecto fue la investigación de las herramientas a utilizar, de acuerdo a los beneficios, características y uso que nos proporcionarían. Para lo cual se utilizó **Qmail** como servidor de correo, ya que ofrece seguridad, rendimiento, fiabilidad y simplicidad.

El siguiente paso fue hacer una revisión a la API de Libpcap en la documentación oficial de Tcpdump, lo cual permitió identificar las funciones que permitieron realizar la captura de los paquetes de red.

7.2. IDE de desarrollo

NetBeans es el entorno de programación de aplicaciones que se uso para desarrollar la captura de los paquetes de red, incluye un editor de texto, un compilador, un intérprete y un depurador. Este entorno es compatible con varios lenguajes de programación, sin embargo se trabajo con el compilador de Java[19]. Podemos descargar NetBeans en el sitio web oficial y de forma gratuita.

Para trabajar con paquetes de red, NetBeans ofrece una librería de código abierto llamada **jNetPcap** que ofrece:

- Un wrapper de Java para casi todas las llamadas nativas de la biblioteca libpcap.
- Los paquetes capturados son decodifica en tiempo real.
- Proporciona una gran biblioteca de protocolos de red (protocolos básicos).
- Los usuarios pueden añadir fácilmente sus propias definiciones de protocolo utilizando Java SDK.

jNetPcap utiliza una mezcla de la implementación nativa y java para un rendimiento óptimo para la decodificación de paquetes. Además de que jNetPcap ofrece documentación necesaria para el desarrollo de un programa.

En la Figura No.20, se muestra la estructura de la API jNetPca.

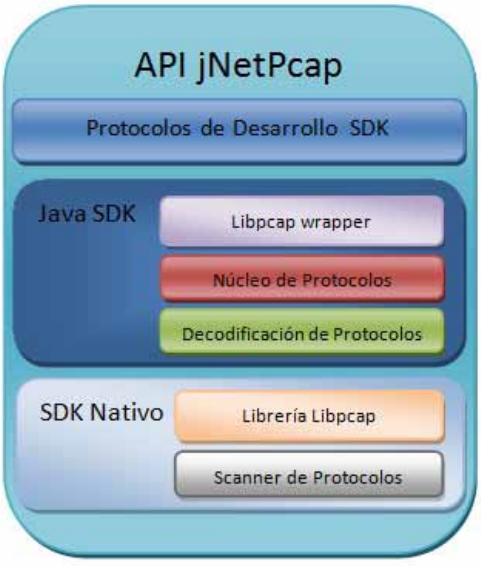


Figura 20: Arquitectura de API jNetPcap.

[19]Java: Lenguaje de programación de alto nivel, diseñado por James Gosling y Sun Microsystems en 1995

7.3. Valores de Entrada y Salida.

Las variables de entrada para todo el sistema de seguridad son: las direcciones IP tanto del emisor como del servicio al que se pretende acceder, las cuales se registran al momento de solicitar la petición de conexión, además del correo electrónico que se pretende ser enviado a través de la red, y que es capturado en el módulo de filtrado para su clasificación.

Las variables de salida del sistema son las alertas que dependen de la clasificación de los correos y que generarán una acción dependiendo al tipo de riesgo este pudiera haber causado si llega a su destino, así como un reporte con estadísticas que se obtienen del establecimiento de las conexiones permitidas en el módulo de análisis.

7.4. Características Mínimas

El sistema de seguridad que se implementó, permite hacer las capturas de los correos electrónicos, para que este proyecto se pueda dar por concluido, es necesario a partir de la captura se puedan reconocer los correos en los cuales su contenido se clasifique como confidencial, dependiendo de la comparación con el patrón de datos y se establezca alguna acción para el envío del mismo, así como las alertas y estadísticas que se generarán a partir de su identificación.

7.5. Posibles alcances a futuro

Los posibles alcances que se pueden desarrollar en un futuro para este proyecto son:

- Añadir el análisis a mensajes encriptados.
- Añadir una fase de bloqueo para mensajes que contengan virus.
- Añadir el análisis para el filtrado de información basado en mensajería instantánea.
- Añadir el análisis para los mensajes adjuntos.

8. Pruebas y Resultados

Para realizar las pruebas de filtrado de los correos se implementó una red LAN con los siguientes componentes:

8.1. Hardware

- Laptop 1
  - Procesador Intel Atom
  - Memoria RAM 2Gb
  - Disco duro 320 Gb
  - Sistema Operativo GNU/Linux Ubuntu

- Laptop 2
  - Procesador Intel Atom
  - Memoria RAM 2Gb
  - Disco duro 120 Gb
  - Sistema Operativo GNU/Linux Ubuntu
- Laptop 3
  - Procesador Intel Core i3-370M
  - Memoria RAM 2Gb
  - Disco duro 320 Gb
  - Sistema Operativo Fedora

8.2. Software

- Servidor de Correos qmail
- Librería Libpcap

8.3. Herramientas adicionales utilizadas

- Wireshark: Para la captura de tráfico en la red, instalado en el servidor de la red LAN.

8.4. Cuentas de usuarios

- Red 1
  - Red 192.168.1.0/24
  - Dominio: qmail.zone.com
- Red 2
  - Red 192.168.26.0/24
  - Dominio: mail.domain.com

8.5. Pruebas No.1.- Envió entre clientes del mismo dominio

La primera prueba consiste en comprobar el envió de correos entre clientes del mismo dominio y la misma red. Ya que lo establecido en el sistema de seguridad que se implemento, el envío de correos entre usuarios que pertenecen al mismo dominio, no pasará por la sección de filtrado, ya que solo se pregunta si pertenece al mismo dominio, si esto es verdadero, el sistema de seguridad permite el envío del correo sin realizar algun análisis o modificación.En las Figura No.21, 22 y 23, se muestran las configuraciones(establecer el servidor de correos y el puerto de envío) necesarias para el usuario1

del dominio mail.domain.com.

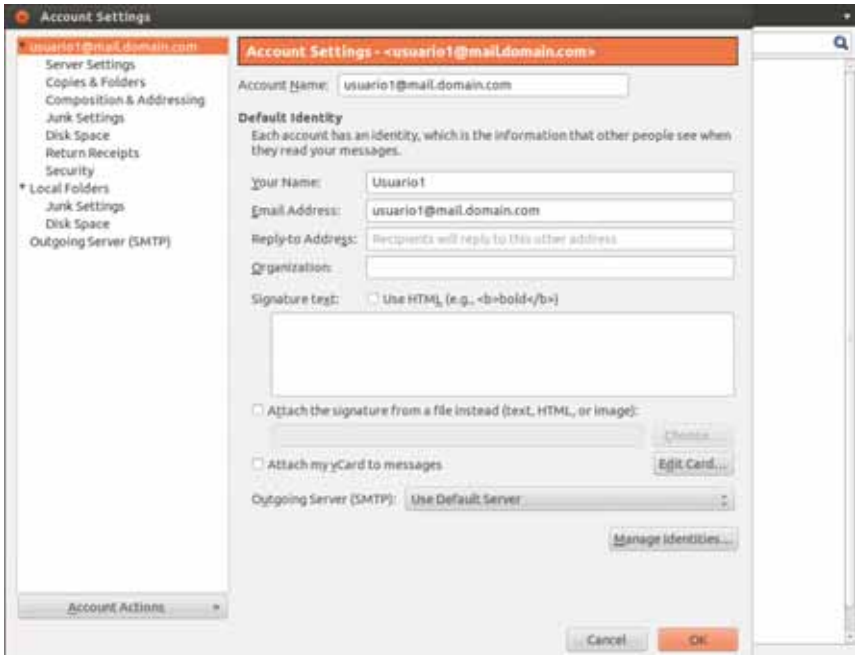


Figura 21: Configuración de un cliente.

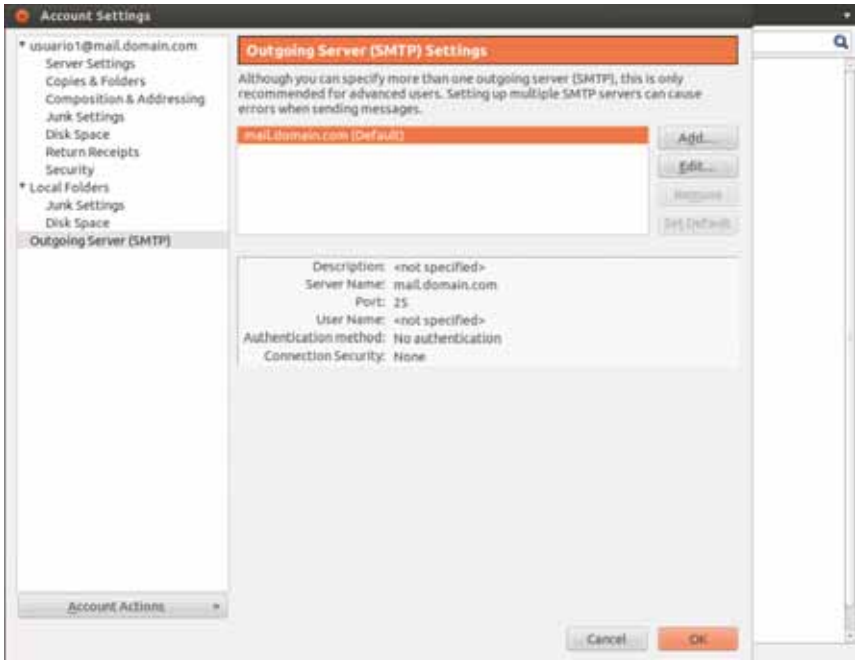


Figura 22: Configuración del servidor de un cliente.

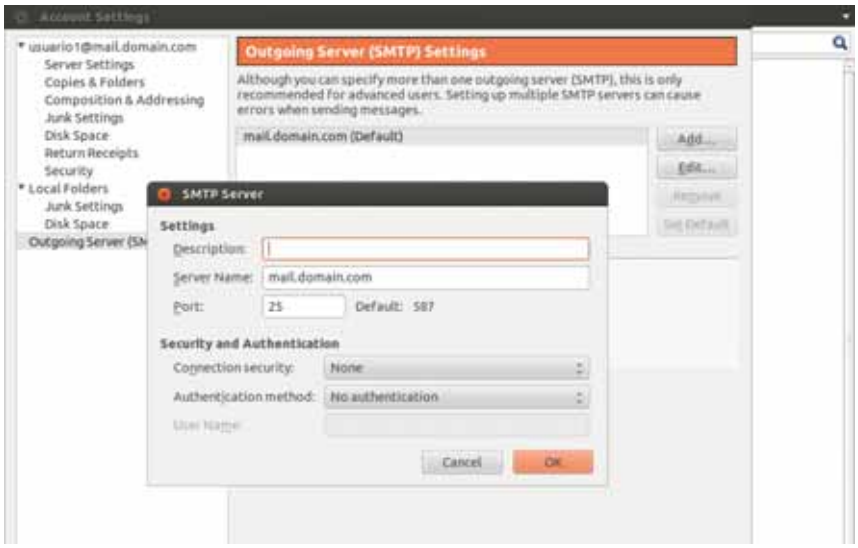


Figura 23: Configuración del puerto.

Una vez configurado el cliente de ese dominio, se puede comenzar a enviar correos. La Figura No.24 muestra el proceso para enviar un correo.

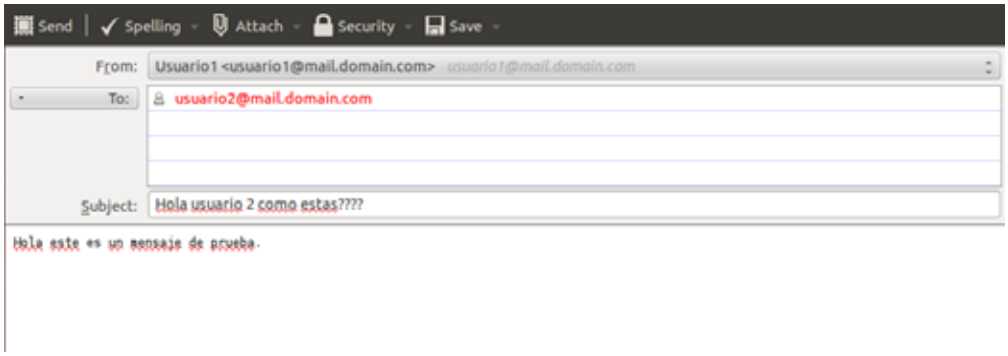


Figura 24: Envío de correo.

El *usuario1* tiene la IP 192.168.1.101 y el *usuario2* tiene la IP 198.162.1.102, en la Figura No.25 se muestra el proceso de intercambio del cliente con el servidor qmail para el envió del correo con destino al usuario2.

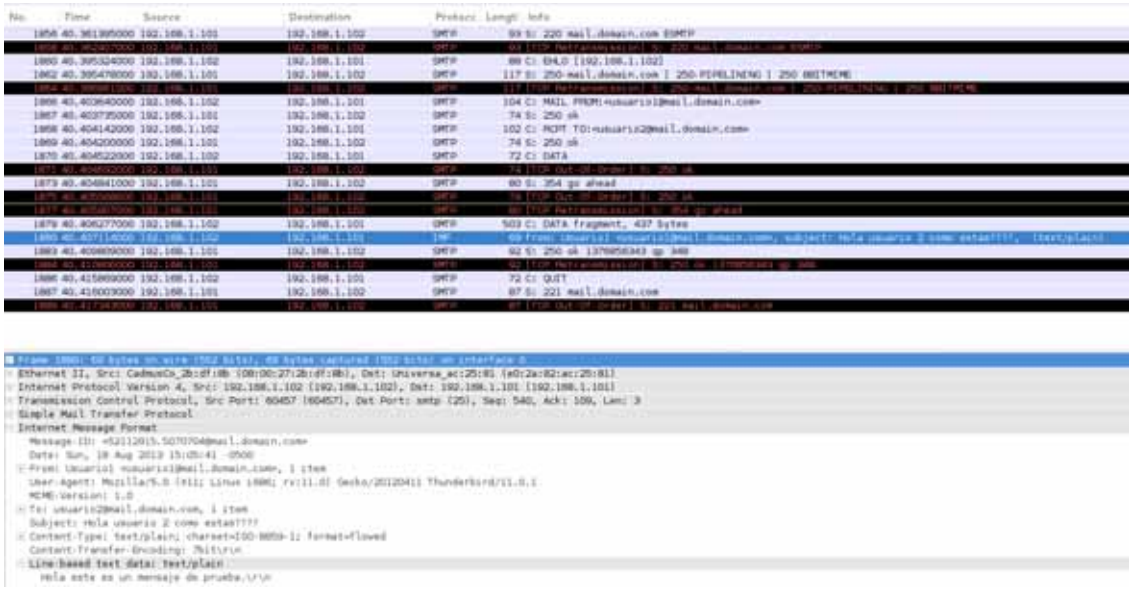


Figura 25: Captura de tráfico de red.

En la Figura No.26 se muestra la secuencia de comandos y respuestas que se envian el cliente y el servidor en el proceso de envió de un correo electrónico mismo que sirve de basa para la extracción de los datos a análizar.

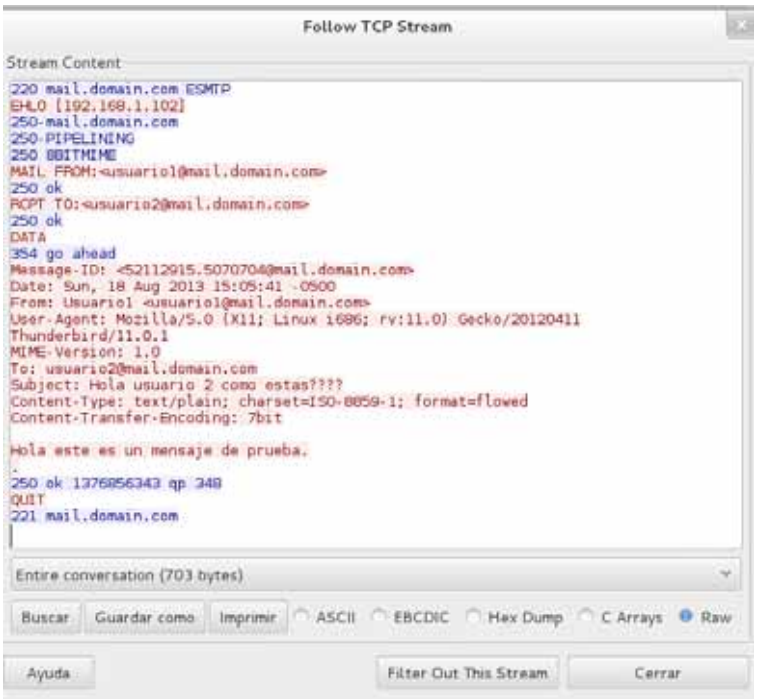


Figura 26: Secuencia de comunicación cliente-servidor.

A continuación se comprueba que ya conexión existe, ya que se verifica que el correo ha llegado a su destino, por lo cual de esta prueba se concluye que la comunicación entre los clientes y los servidores existe. La Figura No.27 demuestra que el correo se entrego a su destinatario.

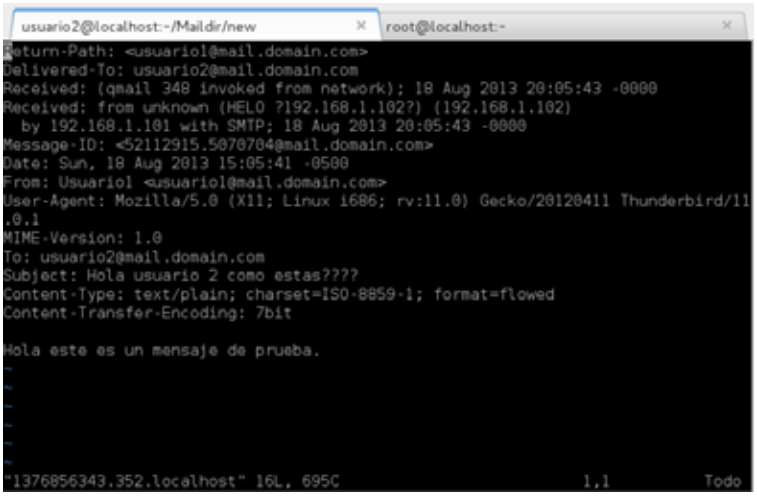


Figura 27: Secuencia de comunicación cliente-servidor.

8.6. Prueba No.2.- Envió de correos entre clientes de dominios distintos

El desarrollo de está prueba muestra el intercambio de correos entre usuarios que pertenecen a dominios de red distintos, esto para asegurar que los mensajes que se envían llegan a los destinatarios.

En las pruebas a realizar se establecieron dos dominios *mail.domain.com y qmail.zone.com*. En la Figura No.28 se muestra la petición para enviar un correo hacia mail.zone.com.

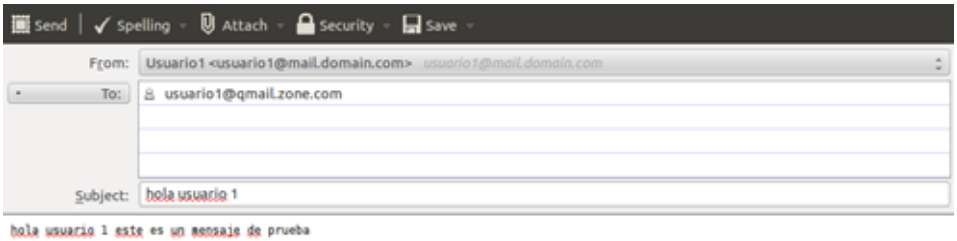


Figura 28: Envió de correos desde el servidor qmail.zone.com.

Cuando el mensaje es enviado desde el emisor, con wireshark podemos observar el estado en los que se envía el mensajes. En la Figura No.29, se muestra la captura de los paquetes que circulan por la red y tienen relación con el envió del correo.

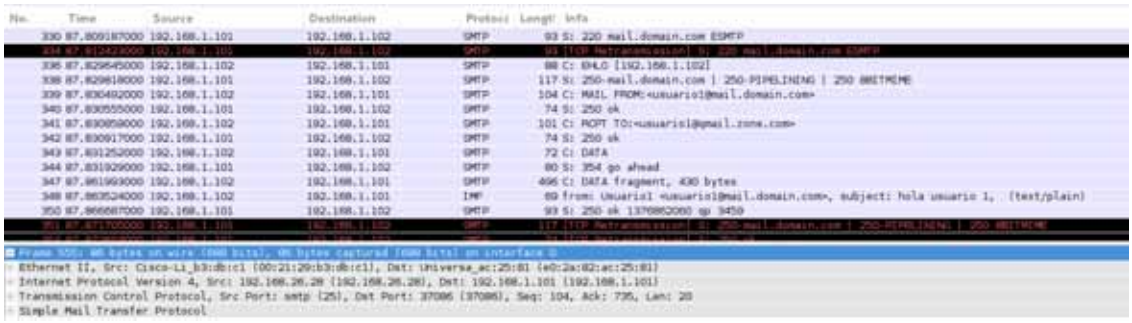


Figura 29: Captura de tráfico de red desde qmail.zone.com

Así mismo, en la Figura No.30 se muestra la secuencia de comandos y respuestas que son enviados entre el cliente y el servidor, desde que se establece la conexión hasta se recibe el correo.

Figura 30: Secuencia de comandos y respuestas del servidor-cliente

8.7. Prueba No.3.- Filtrado de correos

Una vez que se comprobó que los correos son enviados tanto entre clientes del mismo dominio como en clientes de dominios distintos, se puede establecer el sistema de seguridad para realizar el filtrado de los correos, en este caso, está prueba se realiza especialmente para capturar los correos que contienen información confidencial.

El primer paso es establecer en la aplicación de filtrado el patrón de datos con el cual se trabajará, para lo cual se establece una sección para ingresar dichas palabras, una vez hecho esto se comienza la captura. En la Figura No.31 se visualizan los datos para análizar.

Figura 31: Interfaz del administrador

Cuando se inicia la aplicación para capturar los paquetes, se envía un correo el cual contendrá información confidencial, para lo cual utilizamos cualquier usuario de nuestra red. En la Figura No.32 muestra como se envía un correo con información que coincidirá con el patrón de datos.

Figura 32: Correo de prueba para el filtro.

Una vez que se inicio la captura de los paquetes, el sistema de seguridad muestra en la interfaz que contiene un paquete que esta en proceso, lo que significa que comenzo a analizar el mensaje. En la Figura No. 33 se muestra el la información de captura del paquete.

Figura 33: Correo en proceso.

El correo que se envío contiene información confidencial por lo caul será filtrado y calsificado como correo bloqueado por lo que se mostrará el la lista de correos bloqueados, y en el cual se mostrará la información correspondiente a dicho mensaje además de que se contabiliza en en los correos invalidos. Como se muestra en la Figura No.34.

Figura 34: Correo invalido.

Cuando el correo se clasificó como invalido, la dirección del remitente fue bloqueada para el envío de correo, es decir, el usuario no podrá enviar correo ya que su dirección se encuentra bloqueada. En la Figura No. 35 se muestra la lista donde se encuentran todas las direcciónes IP que han sido bloqueadas.

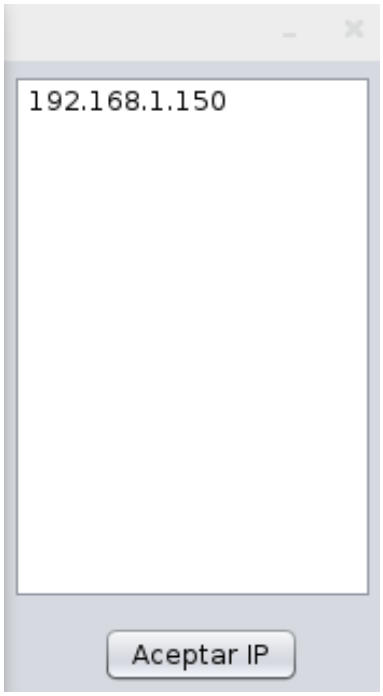


Figura 35: Dirección bloqueada.

Se puede comprobar que la dirección IP se encuentra bloqueado su acceso al puerto 25 o a la dirección donde se encuentra el servidor de correos, en la Figura No.36, se muestran las reglas de iptables donde se muestra la dirección fuente que esta bloqueada.

```
[neo@localhost PT]$ sudo iptables -L INPUT -n -v
Chain INPUT (policy ACCEPT 160 packets, 12972 bytes)
 pkts bytes target     prot opt in     out     source               destination
   13   712 DROP       tcp  --  wlan0  *       192.168.1.150        192.168.1.101       tcp dpt:25
```

Figura 36: Iptables bloqueada.

El correo que se envía, es un correo vació ya que se elimino la información. En la Figura No.37 se muestra el correo que se envía al destinatario, al cual se elimino el mensaje y el asunto.

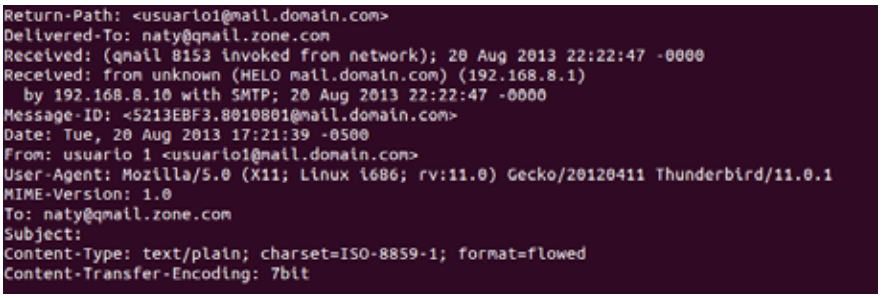


Figura 37: Mensaje recibido por el destinatario.

Para generar las alertas que indican el envío de mansajes invalidos, se envía un correo al administrador indicando los datos del remitente así como el mensaje que envío. En la Figura No.38, se muestra el mensaje que recibe el administrador.

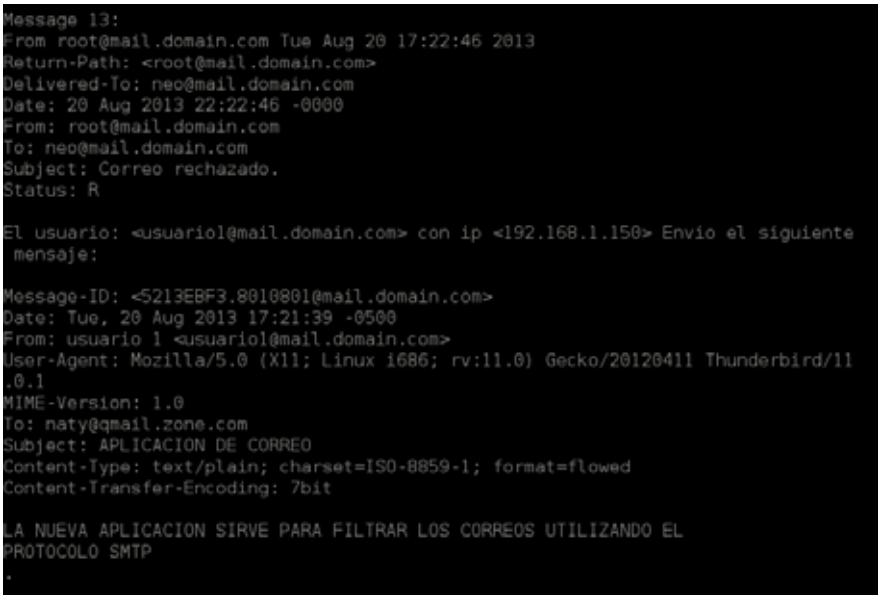


Figura 38: Mensaje recibido por el administrador.

Por lo que si el usuario bloqueado quisiera enviar un nuevo correo, este ya no se le permitirá el envío. En la Figura No. 39 se muestra como se deniega el acceso al usuario para enviar correos.

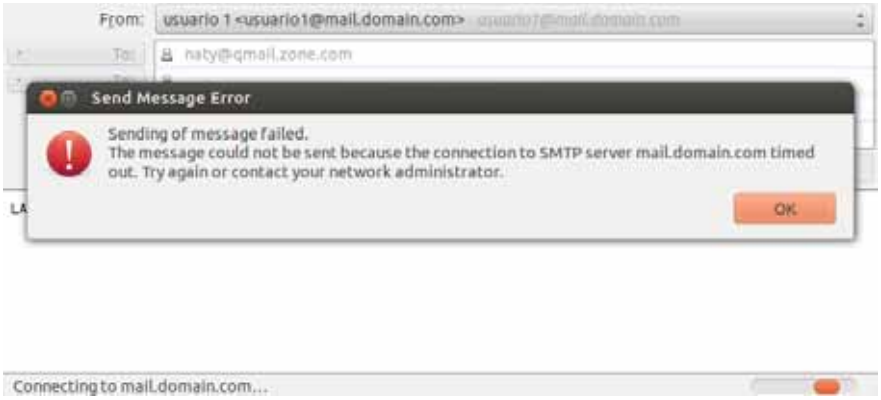


Figura 39: Nuevo mensaje.

9. Conclusión

En la actualidad la seguridad de la información es un asunto de gran importancia para las empresas, ya que de esto depende el futuro de la misma, un acceso inadecuado a la información pondría en riesgo imagen de la empresa, por lo cual, la importancia del desarrollo de esté proyecto, radica en la proteger la información que circula a través de la red.

El desarrollo de esté proyecto permitió que todos los objetivos particulares se cumplieran satisfactoriamente, a partir de la construcción de los módulos se permitió realizar el sistema de seguridad para clasificar y filtrar los correos electrónicos en base a su contenido, cumpliendo con los requerimientos mínimos establecidos para dar por concluido el proyecto, refiriéndonos a que la información confidencial no sea enviada a través de los correos.

En cuanto al módulo administración del servidor de correos, nos permitió realizar las configuraciones necesarias para establecer un nuevo dominio en el cual se realizaron las pruebas del envió de los mismos. Ya que, algunos servidores de correos, no permiten la recepción de correos con dominio desconocido, y tienden a clasificarlo como Spam.

En este proyecto se implementaron y aprovecharon las características de tecnologías muy poderosas, la programación de alto nivel, librerías de código abierto y uso de protocolos estándares, éstos nos permiten crear herramientas poderosas y útiles para dar solución a problemas que actualmente enfrentan las empresas con respecto a la seguridad de su información. Esto es posible ya que el tipo de tecnologías como los sniffers, ofrecen a los usuarios y desarrolladores acceso a su documentación para poder realizar el desarrollo o implementación de los mismos, dependiendo a los requerimientos que se tengan.

A. Manual de Instalación

A.1. Configuración del servidor de correos qmail

Para realizar la instalación y configuración de qmail, una vez que se cuenta con los requisitos necesarios, el primer paso es descargar el código fuente de qmail y cualquier otros complementos. Para lo cual se necesitará qmail, así como, también obtener ucspi-tcp y daemontools :

Requerimientos para la instalación de qmail

- 10 MB de espacio libre en el área donde se instalará qmail.
- Suficiente espacio de disco para la cola en un sistema de archivos local apropiado.
- Un sistema operativo Unix o similar.
- El acceso a un sistema de resolución de nombres de dominio (DNS).
- Un entorno de desarrollo completo que incluye un compilador C, enlazador, archivos de cabecera y bibliotecas.
- Contar con los siguientes archivos qmail, ucspi-tcp y daemontools.

Instalación de qmail

- Copiar o mover los archivos netqmail y ucspi-tcp en el directorio `/usr/local/src`.

  ```
  mv netqmail-1.06 ucspi-tcp-0.88 /usr/local/src
  ```

- Dado que el programa de instalación de qmail crea los subdirectorios, sólo es necesario para crear el directorio donde reside qmail

  `mkdir /var/qmail`.

- **Crear los usuarios y grupos**: La manera más fácil de crear los usuarios y grupos necesarios es crear un pequeño script. En el directorio de netqmail (/usr/local/src/netqmail), se encuentra un archivo llamado `INSTALL.ids`. Este archivo contiene las líneas de comandos para muchas plataformas, por lo hay que copiar el archivo a otro nombre para poder editarlo.

  ```
  cd /usr/local/src/netqmail-1.06
  ```

  ```
  cp INSTALL.ids IDS
  ```

  Luego, con cualquier editor, se elimina el archivo, excepto las líneas que desee, según el sistema operativo. En nuestro caso los IDS para Linux serian los siguientes:

```
groupadd nofiles
useradd -g nofiles -d /var/qmail/alias alias
useradd -g nofiles -d /var/qmail qmaild
useradd -g nofiles -d /var/qmail qmaill
useradd -g nofiles -d /var/qmail qmailp
groupadd qmail
useradd -g qmail -d /var/qmail qmailq
useradd -g qmail -d /var/qmail qmailr
useradd -g qmail -d /var/qmail qmails
```

Después de editarlo, se usa `chmod`, se cambian los permisos del archivo para convertirlo en un archivo ejecutable:

```
cchmod 700 IDS
./IDS
```

- **Compilar qmail**:

Una vez que ya se crearon los grupos y usuarios, editamos el archivo `/usr/local/src/netqmail` para indicarle que trabajaremos con un compilador C:

```
gcc -O2

This will be used to compile .c files.
~
~
~
~
"/usr/local/src/netqmail-1.06/conf-cc" [Sólo lectura] 3L, 48C 3,38          Todo
```

De igual forma editamos el archivo `/usr/local/src/netqmail-1.06/conf-ld`

```
gcc -s

This will be used to link .o files into an executable.
~
~
~
~
~
"/usr/local/src/netqmail-1.06/conf-ld" [Sólo lectura] 3L, 63C 3,54          Todo
```

Ya que se tienen creados los grupos, usuarios y se especifico el compilador con el cual se trabajará, podemos realizar la compilación de qmail.

```
make setup check
```

Después de que la compilación se ha completado, se tiene que hacer la configuración posterior a la instalación, en la cual se establece el dominio de correo.

```
./config-fast qmail.zone.com
```

qmail ya está instalado en el sistema y está listo para correr. A continuación se describen los pasos para iniciar y probar qmail.

- **Instalar ucspi-tcp**: Ucspi-tcp consiste en un conjunto de utilidades, entre las que se destacan tcpserver y tcpclient, que sirven para construir aplicaciones cliente-servidor.

  Anteriormente, se desempaqueto los archivos qmail, ucspi-tcp y daemontools. Ahora cambiamos a la ucspi-tcp para instalarlo, ejecutando los pasos siguientes:

```
cd /usr/local/src/ucspi-tcp-0.88
patch </usr/local/src/netqmail-1.06/other-patches/ucspi-tcp-0.88.errno.patch
make
make setup check
```

- **daemontools**: Daemontools, es una colección de herramientas para administrar los servicios de UNIX.

  Ahora cambiamos a la carpeta que contiene a daemontools-0.76 para instalarlo, ejecutando los pasos siguientes:

```
cd /package/admin/daemontools-0.76
cd src
patch </usr/local/src/netqmail-1.06/other-patches/daemontools-0.76.errno.patc
cd ..
package/install
```

- **Inicializar qmail**

  El directorio /var/qmail/boot contiene ejemplos de los scripts de arranque para diferentes configuraciones, para esta instalación , se utilizó el siguiente script que editaremos en el archivo **/var/qmail/rc**

```
chmod 755 /var/qmail/rc
mkdir /var/log/qmail
```

- **Tipo de buzón**

  Para seleccionar el tipo de buzón de correo predeterminado,se introduce el valor *defaultdelivery* de la tabla en /var/qmail/control/defaultdelivery. Para este desarrolló utilizamos *Maildir*.

```
echo ./Maildir/ >/var/qmail/control/defaultdelivery
```

- **Archivos de arranque del servidor**

  Script de qmailctl: qmailctl se utilizará para iniciar qmail de forma automática cada vez que se arranca el sistema.Esto se logra mediante la creación de un script de inicio/apagado en */var/qmail/bin/qmailctl*, como el siguiente:

```
#!/bin/sh
PATH=/var/qmail/bin:/bin:/usr/bin:/usr/local/bin:/usr/local/sbin
export PATH
QMAILDUID=‘id -u qmaild‘
NOFILESGID=‘id -g qmaild‘
case "$1" in
start) echo "Starting qmail"
if svok /service/qmail-send ; then
svc -u /service/qmail-send /service/qmail-send/log
else
echo "qmail-send supervise not running"
fi
if svok /service/qmail-smtpd ; then
svc -u /service/qmail-smtpd /service/qmail-smtpd/log
else
echo "qmail-smtpd supervise not running"
```

```
fi
if [ -d /var/lock/subsys ]; then
touch /var/lock/subsys/qmail
fi
;;
stop)
echo "Stopping qmail..."
echo "qmail-smtpd"
svc -d /service/qmail-smtpd /service/qmail-smtpd/log
echo "qmail-send"
svc -d /service/qmail-send /service/qmail-send/log
if [ -f /var/lock/subsys/qmail ]; then
rm /var/lock/subsys/qmail
fi
;;
stat)
svstat /service/qmail-send
svstat /service/qmail-send/log
svstat /service/qmail-smtpd
svstat /service/qmail-smtpd/log
qmail-qstat
;;
doqueue|alrm|flush)
queue)
qmail-qstat
qmail-qread
;;
reload|hup)
echo "Sending HUP signal to qmail-send."
svc -h /service/qmail-send
;;
pause)
echo "Pausing qmail-send"
svc -p /service/qmail-send
echo "Pausing qmail-smtpd"
svc -p /service/qmail-smtpd
;;
cont)
echo "Continuing qmail-send"
svc -c /service/qmail-send
echo "Continuing qmail-smtpd"
svc -c /service/qmail-smtpd
;;
echo "Flushing timeout table and sending ALRM signal to qmail-send."
/var/qmail/bin/qmail-tcpok
```

```
svc -a /service/qmail-send
;;
restart)
echo "Restarting qmail:"
echo "* Stopping qmail-smtpd."
svc -d /service/qmail-smtpd /service/qmail-smtpd/log
echo "* Sending qmail-send SIGTERM and restarting."
svc -t /service/qmail-send /service/qmail-send/log
echo "* Restarting qmail-smtpd."
svc -u /service/qmail-smtpd /service/qmail-smtpd/log
;;
cdb)
tcprules /etc/tcp.smtp.cdb /etc/tcp.smtp.tmp </etc/tcp.smtp
chmod 644 /etc/tcp.smtp.cdb
echo "Reloaded /etc/tcp.smtp."
;;
help)
cat HELP
stop - stops mail service (smtp connections refused, nothing goes out)
start - starts mail service (smtp connection accepted, mail can go out)
pause - temporarily stops mail service (connections accepted, nothing
leaves
cont - continues paused mail service
stat - displays status of mail service
cdb - rebuild the tcpserver cdb file for smtp
restart - stops and restarts smtp, sends qmail-send a TERM & restarts
it
doqueue - schedules queued messages for immediate delivery
reload - sends qmail-send HUP, rereading locals and virtualdomains
queue - shows status of queue
alrm - same as doqueue
flush - same as doqueue
hup - same as reload
HELP
;;
*)
echo Usage: $0 start|stop|restart|doqueue|flush|reload|stat|pause|cont|cdb|qu
exit 1
;;
esac
exit 0
```

A continuación se convierte el script qmailctl a ejecutable y se vincular a un directorio en su ruta de acceso:

```
chmod 755 /var/qmail/bin/qmailctl
ln -s /var/qmail/bin/qmailctl /usr/bin
```

- **Scripts *supervice***

  La utilidad *supervise* monitorea los servicios. Inicia los servicios, y los reinicia en el caso que se caigan.

  Ahora se crean los directorios *supervice* para los servicios de qmail:

```
mkdir -p /var/qmail/supervise/qmail-send/log
mkdir -p /var/qmail/supervise/qmail-smtpd/log
```

  Creamos el archivo `/var/qmail/supervise/qmail-send/run`:

```
#!/bin/sh
exec /var/qmail/rc
```

  Creamos el archivo `/var/qmail/supervise/qmail-send/log/run`:

```
#!/bin/sh
exec /usr/local/bin/setuidgid qmaill /usr/local/bin/multilog t /var/log/qmail
```

  Creamos el archivo `/var/qmail/supervise/qmail-smtpd/run`:

```
#!/bin/sh
QMAILDUID=‘id -u qmaild‘
NOFILESGID=‘id -g qmaild‘
MAXSMTPD=‘cat /var/qmail/control/concurrencyincoming‘
LOCAL=‘head -1 /var/qmail/control/me‘
if [ -z "$QMAILDUID" -o -z "$NOFILESGID" -o -z "$MAXSMTPD" -o -z "$LOCAL"
]; then
echo QMAILDUID, NOFILESGID, MAXSMTPD, or LOCAL is unset in
echo /var/qmail/supervise/qmail-smtpd/run
exit 1
fi
if [ ! -f /var/qmail/control/rcpthosts ]; then
echo "No /var/qmail/control/rcpthosts!"
echo "Refusing to start SMTP listener because it’ll create an open relay"
exit 1
fi
exec /usr/local/bin/softlimit -m 2000000
/usr/local/bin/tcpserver -v -R -l "$LOCAL" -x /etc/tcp.smtp.cdb -c "$MAXSMTPI

-u "$QMAILDUID" -g "$NOFILESGID" 0 smtp /var/qmail/bin/qmail-smtpd 2>&1
```

  Creamos el archivo de control *concurrencyincoming*:

```
echo 20 >/var/qmail/control/concurrencyincoming
chmod 644 /var/qmail/control/concurrencyincoming
```

Creamos el archivo /var/qmail/supervise/qmail-smtpd/log/run:

```
#!/bin/sh
exec /usr/local/bin/setuidgid qmaill /usr/local/bin/multilog t /var/log/qmail
```

Convertimos a ejecutables dichos archivos:

```
chmod 755 /var/qmail/supervise/qmail-send/run
chmod 755 /var/qmail/supervise/qmail-send/log/run
chmod 755 /var/qmail/supervise/qmail-smtpd/run
chmod 755 /var/qmail/supervise/qmail-smtpd/log/run
```

Entonces, levantamos el servicio de los directorios *log*:

```
mkdir -p /var/log/qmail/smtpd
chown qmaill /var/log/qmail /var/log/qmail/smtpd
```

Finalmente ligamos los directorios *supervise* a */service*:

```
ln -s /var/qmail/supervise/qmail-send /var/qmail/supervise/qmail-smtpd
/service
```

El sistema de qmail podría comenzar automáticamente después de que los links son creados.

- **SMTP Access Control**: Permite a los host locales inyectar mensajes vía SMTP:

```
echo '127.:allow,RELAYCLIENT=" " '>>/etc/tcp.smtp
qmailctl cdb
```

- **Crear el sistema de los alias**

  Hay tres sistemas de aliases que podrían ser creados en todas las instalaciones de qmail:

  Para crear estos aliases, se tiene que decidir donde se quiere cada uno de ellos(un usuario local o en una dirección remota) y crear los archivos apropiados. Para el desarrolló de este proyecto se utilizó lo siguiente:

```
echo usuario >/var/qmail/alias/.qmail-root
echo usuario >/var/qmail/alias/.qmail-postmaster
ln -s .qmail-postmaster /var/qmail/alias/.qmail-mailer-daemon
```

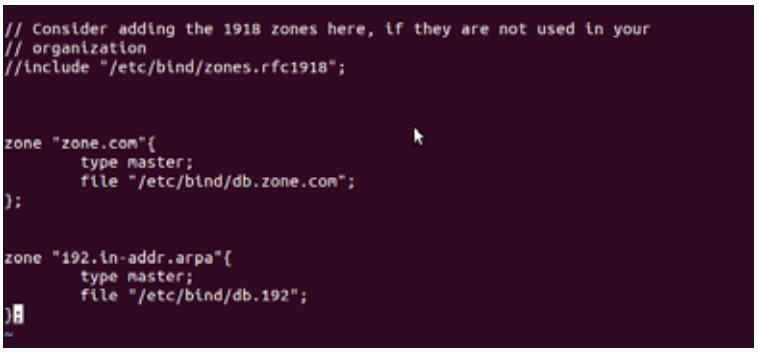
```
ln -s .qmail-postmaster /var/qmail/alias/.qmail-abuse
chmod 644 /var/qmail/alias/.qmail-root /var/qmail/alias/.qmail-postmaster
```

A partir de estas configuraciones el sistema de correo funciona correctamente para lo cual se puede realizar su ejecución y enviar correos.

A.2. Configuración del DNS

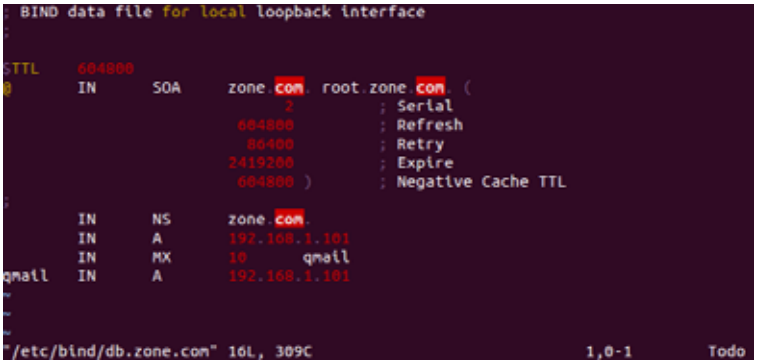
La configuración del DNS nos ayuda a proporcionar el servicio de email a la organización "zone.com"para la cual se definió el dominio en las configuraciones de qmail, la cual utilizará a “qmail” como servidor de correo. Por lo que tendremos direcciones de la forma “usuario@qmail.zone.com”.

Para ello, definimos la zona con la que trabajara nuestro dominio, en archivo `/etc/bind/named.conf.local` en el cual definimos el nombre de nuestra zona, como se muestra en la siguiente imagen.



Ya que se definió la zona con la cual nuestro dominio trabajará, el siguiente paso será aceptar todad las peticiones de nuestra red, y definir el reenvió de las mismas hacia nuestra dirección IP, ya que esta será la encargada de resolver los nombres de dominio. En la siguiente imagen se muestran las modificaciones a realizar en el archivo `/etc/bind/named.conf.options`.

En la configuración del nameserver, nos interesa la configuración del registro MX para el dominio de nuestra organización. Asumiendo que el nameserver utiliza bind, el archivo de la zona qmail.zone.com `/etc/bind/db.zone.com` contiene lo que se muestra a continuación:



Donde definimos el registro MX, la dirección en la cual se encuentra nuestro servidor.

El último paso a realizar es definir ld dirección del nameserver, para ello editamos el archivo `/etc/resolv.conf`, en donde definimos la dirección de localhost(127.0.0.1), como el encargado de resolver los nombres de dominio.

A.3. Configuración de Iptables

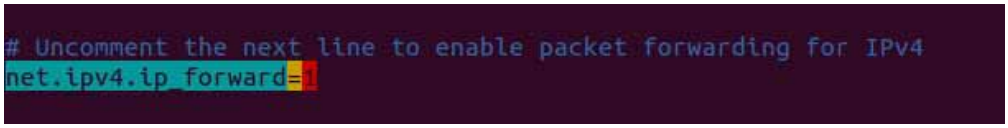
Por defecto, la política IPv4 en kernels de Linux/Unix deshabilita el soporte para reenvío de IP. Esto evita que máquinas que corran alguna distribución de estos, funcionen como un ruteador. Para habilitar el reenvío de IP, se usa el siguiente comando *sysctl -w net.ipv4.ip_forward=1*, al aplicarlo se muestra lo siguiente:

```
naty@Administrador:~$ sudo sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

Este cambio en la configuración sólo es válido para la sesión actual; no persiste luego de un reinicio de equipo o del servicio de red. Para establecer el reenvío de IP permanente, editamos el archivo/etc/sysctl.conf como sigue:

- Ubicar la siguiente línea: `net.ipv4.ip_forward = 0`,
- Y editarla para que se lea como: `net.ipv4.ip_forward = 1`

Como se muestra en la siguiente imagen:



El último paso para habilitar en reenvío IP es usar el siguiente comando para habilitar el cambio en el archivo sysctl.conf:

sysctl -p /etc/sysctl.conf.

En algunas distribuciones de Linux/Unix cuentan, por defecto, con algunas reglas en el firewall, por ello utilizamos un script que nos permita eliminar esas reglas establecidas, para generar las nuestras.

Script para eliminar reglas establecidas en el firewall

```
#!/bin/sh
#/usr/local/bin/fw_flush
#flush script, which deletes all active rules
#and chains, and resets default policies to "accept"
#this is like having no firewall at all
#define variables
ipt="/sbin/iptables"
echo "The firewall is now being shut down. All policies are set to
ACCEPT, all rules and chains are deleted, all counters are set to zero."
#Set default policies to ACCEPT everything
ipt -P INPUT ACCEPT
ipt -P FORWARD ACCEPT
ipt -P OUTPUT ACCEPT
ipt -t nat -P OUTPUT ACCEPT
ipt -t nat -P PREROUTING ACCEPT
ipt -t nat -P POSTROUTING ACCEPT
ipt -t mangle -P INPUT ACCEPT
ipt -t mangle -P OUTPUT ACCEPT
ipt -t mangle -P FORWARD ACCEPT
ipt -t mangle -P PREROUTING ACCEPT
ipt -t mangle -P POSTROUTING ACCEPT
#Zero out all counters
ipt -Z
ipt -t nat -Z
```

```
ipt -t mangle -Z
# Flush all rules, delete all chains
ipt -F
ipt -X
ipt -t nat -F
ipt -t nat -X
ipt -t mangle -F
ipt -t mangle -X
```

A.4. Inicializacón de los archivos de arranque

A.4.1. Configurar el inicio automático de qmail

Ahora vamos a configurar el sistema para que siempre se ejecute qmail al reiniciarse el equipo. Para ello se debe averiguar el UID y el GID del usuario "qmaild" y del grupo "nofiles" respectivamente, a través del siguiente comando:

id qmaild

```
naty@Administrador:~$ id qmaild
uid=1002(qmaild) gid=1001(nofiles) grupos=1001(nofiles)
```

Ahora, simplemente se añaden los siguientes comandos al final del archivo */etc/rc.d/rc.local.*:

`csh -cf '/command/svscanboot &' /usr/sbin/tcpserver -u 1003 -g 1002 0 smtp /var/qmail/bin/qmail-smtpd &`

O bien, ejecutamos esos mismos comando desde la terminal, como se muestra el la siguiente imagen:

```
root@Administrador:/home/naty# csh -cf '/command/svscanboot &'
[1] 6651
root@Administrador:/home/naty# /usr/sbin/tcpserver -u 1003 -g 1002 0 smtp
        /var/qmail/bin/qmail-smtpd &
[1] 6711
root@Administrador:/home/naty# qmailctl start
Starting qmail
```

A.4.2. Inicializar Bind

Una vez que ya se tienen configurados los archivos correspondientes del DNS, para inicializarlo necesitamos el siguiente comando:

`/etc/init.d/bind9 start`

```
root@Administrador:/home/naty# /etc/init.d/bind9 start
 * Starting domain name service... bind9                          [ OK ]
root@Administrador:/home/naty#
```

A.5. Configuración de NetBeans para crear un proyecto con jNetPcap

Hay varias maneras en que jNetPcap se puede añadir a un proyecto Java existente en NetBeans. Los pasos generales para realizar una compilación con jNetPcap son los siguientes:

- Crear una biblioteca jNetPcap que añade el archivo jNetPcap-*.jar a la ruta de compilación
- Crear una nueva configuración del proyecto de ejecución que incluye biblioteca nativa
- Añadir el archivo .jar de jNetPcap al proyecto de construcción, pero copiar la biblioteca nativa necesariamente al directorio de bibliotecas del sistema (/usr/lib).

Lo primero que se tiene que hacer es descargar el paquete de instalación jNetPcap. Debido a que cada paquete de instalación jNetPcap se instala bajo una ruta de directorio único, se puede fácilmente tener varias versiones de la biblioteca y cambiar entre ellos cuando sea necesario.

Para ello necesitamos descargar los siguientes paquetes: *jNetPcap-1.4.b0004-1.zip, jNetPcap-src-1.4.b0004-1.zip* y *jNetPcap-javadoc-1.4.b0004-1.zip.*. Y extraemos el contenido del paquete jNetPcap-1.4.b0004-1.zip, por lo que ahora contamos con los siguientes recursos:

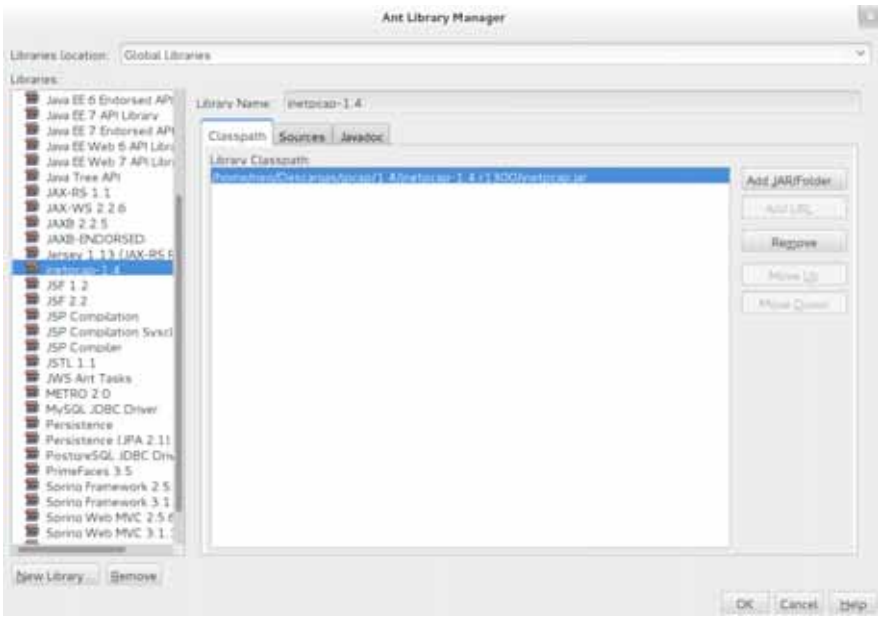
- jNetPcap.jar
- jNetPcap-1.4.b0004-1.zip
- jNetPcap-javadoc-1.4.b0004-1.zip
- jNetPcap-src-1.4.b0004-1.zip

A.5.1. Creación de una nueva biblioteca bajo Netbeans

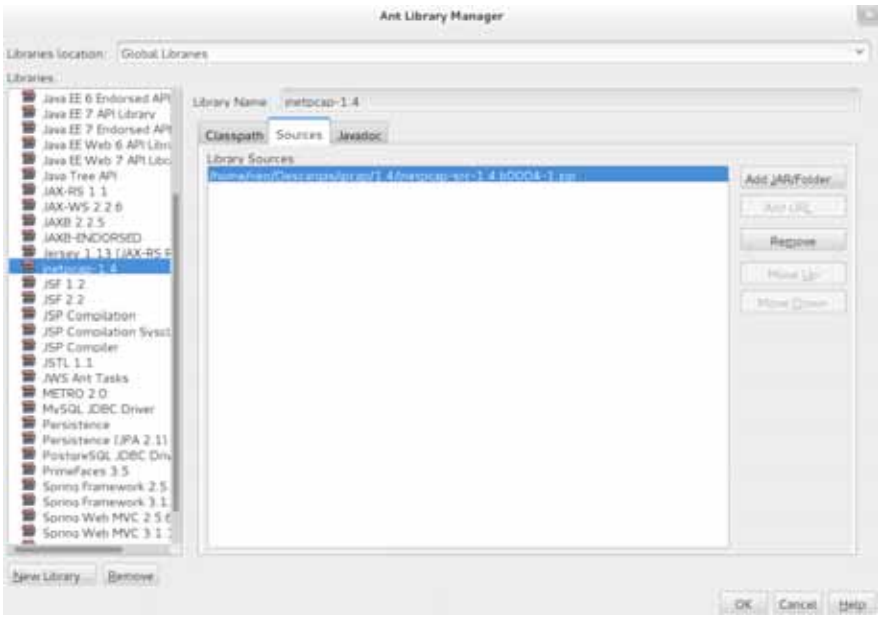
La forma recomendada de configurar el entorno de NetBeans es configurar una nueva librería global en las “Bibliotecas”, la cual sólo contendrá la ruta de acceso al archivo jNetPcap-*.jar.

En el menú “Tools”, hacemos clic en la opción de menú “Libraries”, esta mostrará una ventana “Library Manager”, hacemos clic en “New Library ...” Escribimos un nombre para la nueva declaración de la biblioteca, en este caso utilizamos “jNetPcap-1.4” para nuestro ejemplo. En la casilla de selección “Library Type”, elegimos “Class Libraries” y aceptamos.

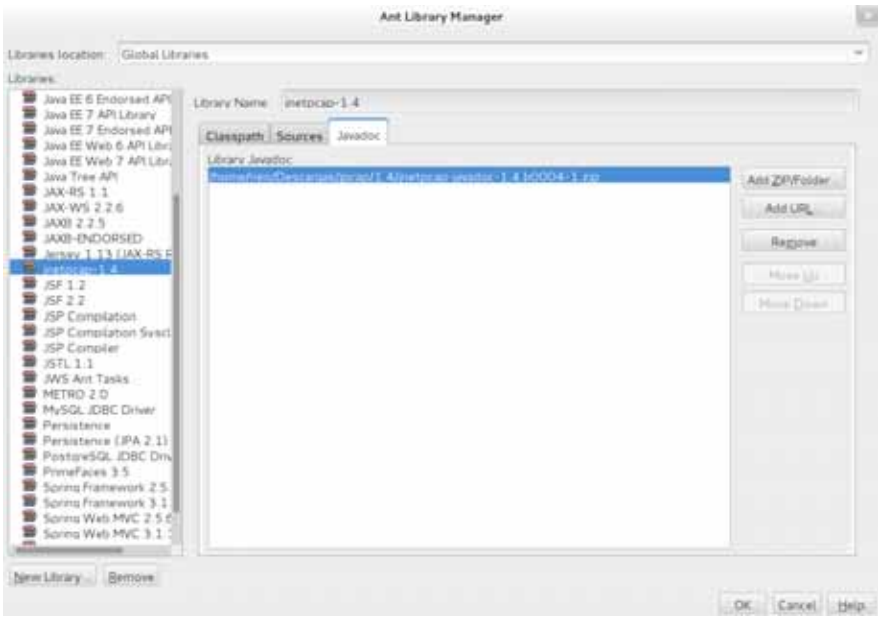
Ahora podemos definir el resto de los elementos que componen esta biblioteca. Haga clic en la pestaña “Classpath” y haga clic en “Add JAR/Folder ...”. Buscamos en el directorio donde se tienen los paquetes que descargamos y seleccionamos el archivo “jNetPcap.jar”. Como se muestra en la siguiente figura:



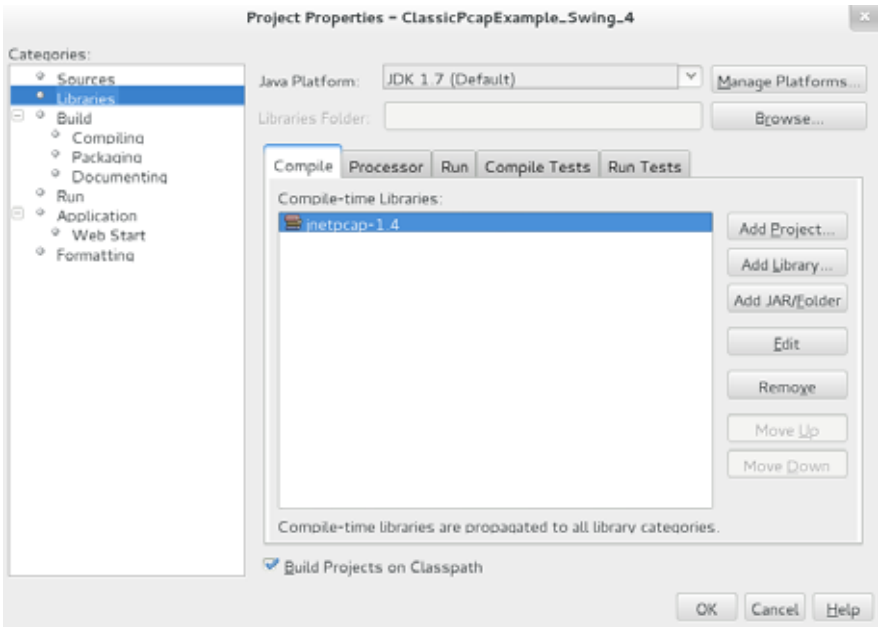
Luego damos clic en la pestaña “Sources”, y agregamos en “Add JAR/Folder” el archivo “jNetPcap-src-1.4.b0004-1.zip” y aceptamos.



Luego haga clic en la pestaña “Javadoc”. A continuación, seleccionamos el archivo “jNetPcap-javadoc-1.4.b0004-1.zip” y lo agregamos en “Add JAR/Folder” y damos clic en aceptar.



Por último aplicamos la biblioteca a nuestro proyecto java. En el Explorador de proyectos, hacemos clic derecho sobre el elemento "Libraries" y seleccionar "Add Library", en esta ventana escogemos nuestra biblioteca recién creada "jNetPcap-1.4" y haga clic en el botón "Add Library".



A.6. Manual de uso del sistema de seguridad

La siguiente imagen muestra la estructura de la interfaz con la que trabajará el administrador y en la cual de obtendrá la información necesaria para realizar cualquier

informe.

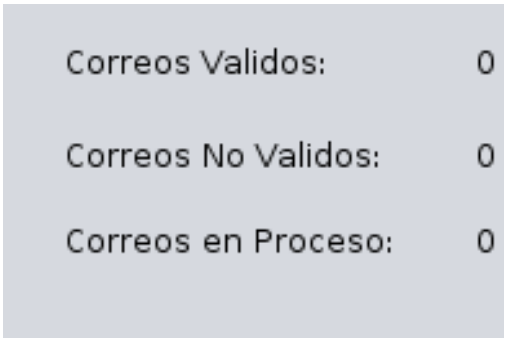
De esta interfaz de desprenden diferentes secciones que ayudaran al filtrado de correos, entre estas encontramos la sección de palabras invalidas, donde podemos ingresar todas aquellas palabras que formen nuestro patrón de datos, y las cuales serán parte esencial para la clasificación de los correos.

Una vez que se definen las palabras invalidas y la interfase de red a analizar se puede comenzar con la captura de los correos, en lo cual se obtiene:

La sección de correos bloqueados, en la cual se mostrará una lista de los correos que son bloqueados,

de tal forma que cuando se seleccione un correo de la lista, en la parte inferior se mostrará la información referente a dicho correo, con los cual se tendrán los datos disponibles para su consulta.

Por último se encuentra la sección de contadores en donde se muestra el número de correos que son detectados tanto validos como inválidos y correos en proceso.



B. Código Fuente

B.1. Interfaz.java

```
package com.view;
```

```
import java.util.ArrayList;
import java.util.List;
import java.util.StringTokenizer;
import javax.swing.DefaultComboBoxModel;
import javax.swing.DefaultListModel;
import jnetpcap.JNetPcap;
import protocols.Smtp;

/**
 *
 * @author Natalia Castro Hernandez
 */
public final class Interfaz extends javax.swing.JFrame {

   //jNetPcap: variable de la Clase JNetPcap que permite captura de correo
    private JNetPcap jNetPcap;
    //listInvalidWords: lista de palabras invalidas
    private List<String> listInvalidWords;
    //invalidEmailsListModel: Modelo para la JList que muestra los correos bloquead
    private static DefaultListModel invalidEmailsListModel;
    ////Modelo para JList que muestra IP's bloqueadas
    private static DefaultListModel invalidIpListModel;
    //JFrame contiene lista de IP's bloqueadas
    private IpList ipList;

    public Interfaz() {
        initComponents();
        invalidIpListModel = new DefaultListModel();
        this.ipList = new IpList(jNetPcap);
        this.ipList.setModel(invalidIpListModel);
        invalidEmailsListModel = new DefaultListModel();
        this.jListInvalidEmails.setModel(invalidEmailsListModel);
        setDevs();
    }

    /*
     * Agrega al Modelo invalidEmailsListModel un correo invalido
     */
    public static void setInvalidMail(String email) {

        invalidEmailsListModel.addElement(email);
    }

    /*
     * Agrega al modelo invalidIpListModel una IP invalida
```

```
 */
public static void setIP(String ip) {
    invalidIpListModel.addElement(ip);
}


public void setDevs() {
    DefaultComboBoxModel model = new DefaultComboBoxModel();

    List<String> devsNames = JNetPcap.getDevsNames();
    for (String name : devsNames) {
        model.addElement(name);
    }
}

/*
 * Este metodo es llamado con el contructor para inicializar el JFrame
 *
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:init
private void initComponents() {

    jToggleButtonStart = new javax.swing.JToggleButton();
    jScrollPane2 = new javax.swing.JScrollPane();
    jTextAreaInvalidWords = new javax.swing.JTextArea();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    jScrollPane3 = new javax.swing.JScrollPane();
    jListInvalidEmails = new javax.swing.JList();
    jLabel5 = new javax.swing.JLabel();
    jLabel6 = new javax.swing.JLabel();
    jLabel7 = new javax.swing.JLabel();
    jLabel8 = new javax.swing.JLabel();
    jTextFieldIP = new javax.swing.JTextField();
    jTextFieldFrom = new javax.swing.JTextField();
    jTextFieldTo = new javax.swing.JTextField();
    jScrollPane1 = new javax.swing.JScrollPane();
    jTextAreaMessage = new javax.swing.JTextArea();
    jLabel10 = new javax.swing.JLabel();
    jLabel11 = new javax.swing.JLabel();
    jLabel12 = new javax.swing.JLabel();
    jLabelValidEmails = new javax.swing.JLabel();
    jLabelInvalidEmails = new javax.swing.JLabel();
    jLabelInProcess = new javax.swing.JLabel();
```

```
        jLabel1 = new javax.swing.JLabel();
        jTextFieldDateTime = new javax.swing.JTextField();
        jLabel9 = new javax.swing.JLabel();
        jTextFieldSubject = new javax.swing.JTextField();
        jLabel13 = new javax.swing.JLabel();
        jMenuBar1 = new javax.swing.JMenuBar();
        jMenu2 = new javax.swing.JMenu();
        jMenuItem1 = new javax.swing.JMenuItem();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Analisis de Correo Electronico");
        setResizable(false);

        jToggleButtonStart.setText("Iniciar Captura");
        jToggleButtonStart.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jToggleButtonStartActionPerformed(evt);
            }
        });

        jTextAreaInvalidWords.setColumns(20);
        jTextAreaInvalidWords.setRows(5);
        jScrollPane2.setViewportView(jTextAreaInvalidWords);

        jLabel3.setText("Palabras Invalidas");

        jLabel4.setText("Correos Bloqueados");

        jListInvalidEmails.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                selectEmail(evt);
            }
        });
        jScrollPane3.setViewportView(jListInvalidEmails);

        jLabel5.setText("IP Origen:");

        jLabel6.setText("Para:");

        jLabel7.setText("De:");

        jLabel8.setText("Mensaje");

        jTextFieldIP.setEditable(false);
```

```
        jTextFieldFrom.setEditable(false);

        jTextFieldTo.setEditable(false);

        jTextAreaMessage.setEditable(false);
        jTextAreaMessage.setColumns(20);
        jTextAreaMessage.setRows(5);
        jScrollPane1.setViewportView(jTextAreaMessage);

        jLabel10.setText("Correos Validos:");

        jLabel11.setText("Correos No Validos:");

        jLabel12.setText("Correos en Proceso:");

        jLabelValidEmails.setText("0");

        jLabelInvalidEmails.setText("0");

        jLabelInProcess.setText("0");

        jLabel1.setText("Fecha/Hora:");

        jTextFieldDateTime.setEditable(false);

        jLabel9.setText("Asunto:");

        jTextFieldSubject.setEditable(false);

        jLabel13.setText("Interfaz de Captura ");

        jMenu2.setText("IP's Bloqueadas");

        jMenuItem1.setText("Mostrar");
        jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem1ActionPerformed(evt);
            }
        });
        jMenu2.add(jMenuItem1);

        jMenuBar1.add(jMenu2);

        setJMenuBar(jMenuBar1);
```

```
javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
            .addGroup(layout.createSequentialGroup()
                .addGap(27, 27, 27)
                .addComponent(jLabel3)
                .addGap(30, 30, 30)
                .addComponent(jLabel4)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                .addComponent(jToggleButtonStart))
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
                    .addGroup(layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.Gr
                            .addComponent(jLabel1)
                            .addComponent(jLabel9))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentP
                        .addGroup(layout.createParallelGroup(javax.swing.Gr
                            .addComponent(jTextFieldDateTime, javax.swing.G
                            .addComponent(jTextFieldSubject, javax.swing.Gr
                    .addGroup(layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.Gr
                            .addComponent(jLabel5)
                            .addComponent(jLabel7)
                            .addComponent(jLabel6))
                        .addGap(26, 26, 26)
                        .addGroup(layout.createParallelGroup(javax.swing.Gr
                            .addComponent(jTextFieldFrom, javax.swing.Group
                            .addComponent(jTextFieldTo, javax.swing.GroupLa
                            .addComponent(jTextFieldIP, javax.swing.GroupLa
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
                    .addComponent(jLabel10)
                    .addComponent(jLabel11)
                    .addComponent(jLabel12))
                .addGap(31, 31, 31)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
                    .addComponent(jLabelInProcess)
                    .addComponent(jLabelInvalidEmails)
                    .addComponent(jLabelValidEmails)))
            .addGroup(layout.createSequentialGroup()
```

```
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
                        .addGroup(layout.createSequentialGroup()
                            .addContainerGap()
                            .addComponent(jScrollPane1, javax.swing.GroupLayout
                        .addGroup(layout.createSequentialGroup()
                            .addGap(10, 10, 10)
                            .addComponent(jScrollPane2, javax.swing.GroupLayout
                            .addGap(18, 18, 18)
                            .addComponent(jScrollPane3, javax.swing.GroupLayout
                    .addGap(0, 0, Short.MAX_VALUE)))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VA
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequ
                .addGap(0, 0, Short.MAX_VALUE)
                .addComponent(jLabel8)
                .addGap(276, 276, 276))
            .addGroup(layout.createSequentialGroup()
                .addGap(239, 239, 239)
                .addComponent(jLabel13)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VA
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jLabel13)
                        .addGap(28, 28, 28)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
                            .addComponent(jLabel3)
                            .addComponent(jLabel4)))
                    .addComponent(jToggleButtonStart, javax.swing.GroupLayout.Align
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELAT
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
                    .addComponent(jScrollPane2)
                    .addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED_S
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
                    .addComponent(jLabel5)
                    .addComponent(jLabel10)
                    .addComponent(jLabelValidEmails)
                    .addComponent(jTextFieldIP, javax.swing.GroupLayout.PREFERRED_S
                .addGap(9, 9, 9)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
                    .addComponent(jLabel7)
```

```
                .addComponent(jTextFieldFrom, javax.swing.GroupLayout.PREFERRED
                .addComponent(jLabel11)
                .addComponent(jLabelInvalidEmails))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
                .addComponent(jLabel6)
                .addComponent(jTextFieldTo, javax.swing.GroupLayout.PREFERRED_S
                .addComponent(jLabel12)
                .addComponent(jLabelInProcess))
            .addGap(6, 6, 6)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
                .addComponent(jLabel1)
                .addComponent(jTextFieldDateTime, javax.swing.GroupLayout.PREFE
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignm
                .addComponent(jLabel9)
                .addComponent(jTextFieldSubject, javax.swing.GroupLayout.PREFER
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED
            .addComponent(jLabel8)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED
            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
            .addContainerGap(28, Short.MAX_VALUE))
    );

    pack();
}// </editor-fold>//GEN-END:initComponents


/*
 * Este metodo detecta un evento en el boton jToggleButtonStart
 * el cual inicia o detiene la captura de paquetes.
 */
private void jToggleButtonStartActionPerformed(java.awt.event.ActionEvent evt)
    //
    if (jNetPcap == null) {
        jNetPcap = new JNetPcap(this.jLabelValidEmails, this.jLabelInvalidEmail
    }
    if (!jNetPcap.isAlive()) {

        this.jTextAreaInvalidWords.setEditable(false);

        String text = this.jTextAreaInvalidWords.getText().toLowerCase();
        StringTokenizer invalidWords = new StringTokenizer(text);
        this.listInvalidWords = new ArrayList();
        while (invalidWords.hasMoreTokens()) {
```

```
            this.listInvalidWords.add(invalidWords.nextToken());
        }

        jNetPcap.setup();
        jNetPcap.setListInvalidWords(listInvalidWords);
        jNetPcap.start();
    } else {
        jNetPcap.interrupt(); //detiene el hilo en ejecucion
        jNetPcap = null;

        //borra IP's invalidas
        invalidIpListModel = new DefaultListModel();
        this.ipList = new IpList(jNetPcap);
        this.ipList.setModel(invalidIpListModel);

        this.jTextAreaInvalidWords.setEditable(true);
        this.jLabelInProcess.setText("0");
        this.jLabelValidEmails.setText("0");
        this.jLabelInvalidEmails.setText("0");
    }

}//GEN-LAST:event_jToggleButtonStartActionPerformed

/*
 * Al capturar un correo que no sea valido se muestra en la lista
 * jListInvalidEmails, la cual al seleccionar un correo invalido
 * despliega su contenido mostrando:
 * 1) IP Fuete
 * 2) Remitente
 * 3) Destinatarios
 * 4) Fecha
 * 5) Asunto
 * 6) Cuerpo del Mensaje Bloqueado
 */
private void selectEmail(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_sele
    int index = jListInvalidEmails.getSelectedIndex();
    List<Smtp> invalidEmails = jNetPcap.getInvalidEmails();
    if (!invalidEmails.isEmpty()) {
        Smtp email = invalidEmails.get(index);
        this.jTextFieldIP.setText(email.getIpSource());
        this.jTextFieldFrom.setText(email.getFrom());
        List<String> toList = email.getToList();
        String temp = "";
        for (String string : toList) {
            temp += string + ", ";
```

```
            }
            temp = temp.substring(0, temp.length() - 1);
            this.jTextFieldTo.setText(temp);

            this.jTextFieldDateTime.setText(email.getDate());

            this.jTextFieldSubject.setText(email.getSubject());

            this.jTextAreaMessage.setText(email.getBodyMessage());
        }
    }//GEN-LAST:event_selectEmail

    /*
     * Muestra JFrame que contiene lista de las IP's bloqueadas
     */
    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-F
        ipList.setVisible(true);

    }//GEN-LAST:event_jMenuItem1ActionPerformed

    /*
     * Metodo main
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the defa
         * For details see http://download.oracle.com/javase/tutorial/uiswing/looka
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException | InstantiationException | IllegalAccessExc
            java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.u
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
```

```
                Interfaz interfaz = new Interfaz();
                interfaz.setVisible(true);

            }
        });
    }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JLabel jLabelInProcess;
    private javax.swing.JLabel jLabelInvalidEmails;
    private javax.swing.JLabel jLabelValidEmails;
    private javax.swing.JList jListInvalidEmails;
    private javax.swing.JMenu jMenu2;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JMenuItem jMenuItem1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane3;
    private javax.swing.JTextArea jTextAreaInvalidWords;
    private javax.swing.JTextArea jTextAreaMessage;
    private javax.swing.JTextField jTextFieldDateTime;
    private javax.swing.JTextField jTextFieldFrom;
    private javax.swing.JTextField jTextFieldIP;
    private javax.swing.JTextField jTextFieldSubject;
    private javax.swing.JTextField jTextFieldTo;
    private javax.swing.JToggleButton jToggleButtonStart;
    // End of variables declaration//GEN-END:variables
}
```

B.2. IpList.java

```
 package com.view;

import java.io.IOException;
```

```
import javax.swing.DefaultListModel;
import jnetpcap.JNetPcap;

/**
 *
 * @author Natalia Castro Hernandez
 */
public class IpList extends javax.swing.JFrame {
    //invalidIpListModel: Modelo para la JList que muestra las IP's Bloqueadas
    private DefaultListModel invalidIpListModel;
    //jNetPcap: variable de la Clase JNetPcap que permite captura de correo
    private JNetPcap jNetPcap;

    public IpList(JNetPcap jNetPcap) {

        this.jNetPcap = jNetPcap;
        initComponents();

    }

    public void setModel(DefaultListModel invalidIpListModel) {
        this.invalidIpListModel = invalidIpListModel;
        this.jList1.setModel(invalidIpListModel);
    }

    /*
     * Este metodo es llamado con el contructor para inicializar el JFrame
     *
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:init
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        jList1 = new javax.swing.JList();
        jButton1 = new javax.swing.JButton();

        setResizable(false);

        jScrollPane1.setViewportView(jList1);

        jButton1.setText("Aceptar IP");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
```

```
        }
    });

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequ
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VA
            .addComponent(jButton1)
            .addGap(40, 40, 40))
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VA
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELAT
            .addComponent(jButton1)
            .addContainerGap())
    );

    pack();
}// </editor-fold>//GEN-END:initComponents
/*
 * Cuando se bloquea una IP se cargan el la JList
 * si se selecciona una ip bloqueada y se presiona el boton "Aceptar IP"
 * se borra la ip bloqueda de la lista de iptable
 */
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIR
    if (this.jList1.getSelectedIndex() != -1) {
        dropRule();
        invalidIpListModel.remove(this.jList1.getSelectedIndex());
    }

}//GEN-LAST:event_jButton1ActionPerformed

/*
 * Este metodo permite borrar una regla de una IP bloqueda.
 */
public void dropRule() {
```

```
        Integer integer = new Integer(this.jList1.getSelectedIndex());
        int index = integer.intValue();
        String comando = "iptables -D INPUT " + Integer.toString(++index);
        try {
            Runtime.getRuntime().exec(comando);
            System.out.println(comando);

            System.out.println("Se borro regla iptables .");
        } catch (IOException ex) {
            System.out.println("Error al borrar regla iptables.");
        }
    }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JList jList1;
    private javax.swing.JScrollPane jScrollPane1;
    // End of variables declaration//GEN-END:variables
}
```

B.3. JNetPcap

```
 package jnetpcap;

import com.view.Interfaz;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JLabel;
import org.jnetpcap.Pcap;
import org.jnetpcap.PcapBpfProgram;
import org.jnetpcap.PcapIf;
import org.jnetpcap.packet.PcapPacket;
import org.jnetpcap.packet.PcapPacketHandler;
```

```
import org.jnetpcap.protocol.network.Ip4;
import org.jnetpcap.protocol.tcpip.Tcp;
import protocols.Smtp;

/**
 *
 * @author Natalia Castro Hernandez
 */
public class JNetPcap extends Thread {

    //alldevs: contiene la lista de todas las interfaces de red sin contar el loop
    private static final List<PcapIf> alldevs = new ArrayList<>();
    //localDomain: contiene el dominio local del servidor qmail
    private static final InetAddress localDomain;
    //externalDomains: contiene la lista de los dominios externos
    private static final List<InetAddress> externalDomains = new ArrayList<>();
    //invalidIPs: contiene la lista de IP's invalidas
    private static final List<String> invalidIPs = new ArrayList<>();
    //errbuf: almacena los errores
    private static StringBuilder errbuf = new StringBuilder();
    //snaplen: permite capturar todos los paquetes entrantes
    private int snaplen = 64 * 1024;
    //flags: Habilita la interfaz en modo promiscuo
    private int flags = Pcap.MODE_PROMISCUOUS;
    //timeout: establece el tiempo de espera
    private int timeout = 10 * 1000;
    //emails: contiene la lista de todos los correos invalidos e invalidos
    private final static List<Smtp> emails = new ArrayList();
    //invalidEmails: Contiene la lista de todos los correos invalidos
    private final static List<Smtp> invalidEmails = new ArrayList();
    //
    private Pcap pcap;
    //listInvalidWords: contiene la lista de las palabras invalidas establecidas po
    private List<String> listInvalidWords;
    //jpacketHandler: permite el manejo de paquetes
    private PcapPacketHandler<String> jpacketHandler;
    //validEmailsCount: muestra la cuenta de correos validos
    private JLabel validEmailsCount;
    //invalidEmailsCount: muestra la cuenta de correos invalidos
    private JLabel invalidEmailsCount;
    //inProcessCount: Muestra la cuenta de correos en proceso
    private JLabel inProcessCount;
    //valid: correos validos
    private int valid = 0;
    //invalid: correos invalidos
```

```
private int invalid = 0;
//correos en proceso
private int process = 0;

static {
    /*
     * busca todas las interfaces disponibles
     */
    int r = Pcap.findAllDevs(alldevs, errbuf);
    if (r == Pcap.ERROR || alldevs.isEmpty()) {
        System.err.println("Error al buscar interfaces");
    }

    /*
     * Carga el dominio que utiliza qmail
     */
    File file = new File("/var/qmail/control/me");
    String me = null;
    try {
        FileReader fileReader = new FileReader(file);
        BufferedReader bufferedReader = new BufferedReader(fileReader);

        me = bufferedReader.readLine();

    } catch (FileNotFoundException ex) {
        System.err.println("El archivo \"me\" de qmail no se puede leer.");
    } catch (IOException ex) {
        System.err.println("El archivo \"me\" de qmail no se puede leer.");
    }

    localDomain = getIpFromName(me);
    System.out.println("El dominio es: " + localDomain.getHostName() + " - " +

}

public int getEmailsSize() {
    return emails.size();
}

public static int getInvalidEmailSize() {
    return invalidEmails.size();
}

public static int getValidEmailSize() {
    return emails.size();
```

```
}

/*
 * el siguiente metodo regresa una lista de todas las interfaces
 * que cuentan con una direccion de red.
 */
public static List<String> getDevsNames() {
    List<String> stringDevs = new ArrayList();
    for (PcapIf pcapIf : alldevs) {

        try {
            byte[] mac = pcapIf.getHardwareAddress();
            if (mac != null && !pcapIf.getName().equals("lo")) {
                stringDevs.add(pcapIf.getName());
            }

        } catch (IOException ex) {
            Logger.getLogger(Interfaz.class.getName()).log(Level.SEVERE, null,
        }

    }
    return stringDevs;
}

public JNetPcap(JLabel ve, JLabel ie, JLabel ip) {
    this.validEmailsCount = ve;
    this.invalidEmailsCount = ie;
    this.inProcessCount = ip;


    /*
     * Se inicia PcapPacketHandler que permite analisar el siguiente
     * mensaje enviado por el cliente.
     */
    this.jpacketHandler = new PcapPacketHandler<String>() {
        @Override
        public void nextPacket(PcapPacket packet, String user) {
            Tcp tcp = new Tcp();
            Ip4 ip4 = new Ip4();

            if (packet.hasHeader(ip4) && packet.hasHeader(tcp)) {


                byte[] payload = tcp.getPayload();
                String stringPayload = new String(payload);
```

```
/*
 * si la lista de correos esta vacia se crea un objeto Smtp nue
 */
if (emails.isEmpty()) {
    emails.add(new Smtp(tcp.hashCode(), listInvalidWords));

    if (!isIpInvalid(ip4.source())) {
        inProcessCount.setText(Integer.toString(++process));
    }

} else {
    boolean exist = false;
    for (Smtp smtpTemp : emails) {
        /*
         * Si existe previamente el objeto smtp en la lista
         * de correos
         */
        if (smtpTemp.compareSmtpHashCode(tcp.hashCode())) {

            if (stringPayload.startsWith("RCPT TO:<")) { //MAIL
                String domain = stringPayload.substring(stringP
                domain = domain.substring(domain.indexOf("@") +
                InetAddress inetAddress = null;

                /*
                 * si el dominio del correo entrante no pertene
                 * al domino de qmail entonces se bloquea la di
                 * del destino del correo saliente
                 */
                if (!domain.equals(localDomain.getHostName()))
                    smtpTemp.setIslocal(false);

                    inetAddress = getIpFromName(domain);

                    if (inetAddress == null) {

                        System.out.println("no existe el domino
                    } else {
                        externalDomains.add(inetAddress);

                        setIptableRule(inetAddress.getHostAddre

                    }
```

```
    } else {
        /*
         * Si el destinatario esta dentro del domin
         * no se valida el correo
         */
        validEmailsCount.setText(Integer.toString(+
        if (process > 0) {
            inProcessCount.setText(Integer.toString
        }
    }

}

Smtp smtp;
if ((smtp = smtpTemp.set(stringPayload)) != null) {
    if (process > 0) {
        inProcessCount.setText(Integer.toString(--p
    }

    /*
     * si el correo recibido no es valido
     * se agrega a la lista de correos no validos
     */
    if (!smtp.isValid()) {
        invalidEmails.add(smtpTemp);
        String s = Integer.toString(invalidEmails.s
        s += ": " + smtpTemp.getIpSource();
        List<String> toList = smtpTemp.getToList();
        for (String toString : toList) {
            s += " To: " + toString;
        }

        Interfaz.setInvalidMail(s);
        invalidEmailsCount.setText(Integer.toString

        //se envia un mensaje al administrador
        sendMessage(smtp);
        //se borra el asunto y contenido del mensaj
        rewriteFile(smtp.getRouteMessage(), smtp.ne

        //se borra la regla de salida del correo pa
        dropRuleOutput();
        //se establece regla de entrada para bloque
        //para que no envie correos nuevamente.
        setIptableRule(smtp.getIpSource(), "INPUT")
```

```
                                    if (!isIpInvalid(ip4.source())) {
                                        Interfaz.setIP(smtp.getIpSource());
                                    }
                                    //se agrega la ip bloqueada
                                    invalidIPs.add(smtp.getIpSource());



                                } else {
                                    /*
                                     * Si el correo es valido se borra la regla
                                     */
                                    validEmailsCount.setText(Integer.toString(+
                                    dropRuleOutput();
                                }

                            }
                            exist = true;
                            break;
                        }
                    }
                    /*
                     * El correo no existe en la lista de correos se agrega el
                     */
                    if (!exist) {

                        emails.add(new Smtp(tcp.hashCode(), listInvalidWords));

                        if (!isIpInvalid(ip4.source())) {
                            inProcessCount.setText(Integer.toString(++process))
                        }
                    }

                }

            }

        }
    };
}

/*
 * El siguiente metodo revisa si la ip se encuentra en la lista de IP's invalid
 */
```

```
public boolean isIpInvalid(byte[] ipsource) {

    StringBuilder bufsource = new StringBuilder();
    for (int j = 0; j < 4; j++) {
        if (bufsource.length() != 0) {
            bufsource.append('.');
        }

        bufsource.append(new Integer((ipsource[j] < 0) ? ipsource[j] + 256 : ip
    }

    boolean exist = false;

    for (String invalid : invalidIPs) {

        if (bufsource.toString().equals(invalid)) {

            exist = true;
            break;
        }
    }

    return exist;


}
/*
 * Establece las reglas de iptables para bloquear correo entrate o saliente
 */
public void setIptableRule(String ip, String inout) {
    String comando = "";


    if (inout.startsWith("OUTPUT")) {
        comando = "iptables -A OUTPUT -o p1p1 -p tcp --dport 25 -d " + ip + " -
    }
    if (inout.startsWith("INPUT")) {
        comando = "iptables -A INPUT -i wlan0 -p tcp --dport 25 -s " + ip + " -
    }

    try {
        Runtime.getRuntime().exec(comando);

        System.out.println("Se bloqueo la siguiente ip: " + ip);
    } catch (IOException ex) {
```

```
            System.out.println("No se pudo aplicar la siguiente regla \n" + comando
        }

    }

    /*
     * El siguiente metodo borra la primer regla de salida de iptables
     */
    public void dropRuleOutput() {
        String comando = "iptables -D OUTPUT 1";
        try {
            Runtime.getRuntime().exec(comando);
            System.out.println(comando);

            System.out.println("Se borro regla iptables.");
        } catch (IOException ex) {
            System.out.println("Error al borrar regla iptables.");
        }
    }

    /*
     * El siguiente correo borra la primer regla iptable de entrada.
     */

    public void dropRuleInput() {
        String comando = "iptables -D INPUT 1";
        try {
            Runtime.getRuntime().exec(comando);
            System.out.println(comando);

            System.out.println("Se borro regla iptables.");
        } catch (IOException ex) {
            System.out.println("Error al borrar regla iptables.");
        }
    }

    /*
     * Sobre escribe el mensaje del correo y borra el Asunto y cuerpo del mensaje
     */

    public void rewriteFile(String route, String message) {

        File file = new File(route);
        FileWriter fw = null;
        PrintWriter pw = null;
```

```
        try {

            fw = new FileWriter(file);
            pw = new PrintWriter(fw);

            pw.print(message);

        } catch (IOException ex) {
            System.err.println("Error al abrir el archivo");
        } finally {
            try {
                fw.close();
            } catch (IOException ex) {
                System.err.println("Error al cerrar el archivo");
            }
        }

    }

    public void sendMessage(Smtp smtp) {
        String comando = "/var/qmail/bin/qmail-inject";
        try {

            Process exec = Runtime.getRuntime().exec(comando);
            try (BufferedWriter bufferedWriter = new BufferedWriter(new OutputStrea
                bufferedWriter.write("To: root\n");
                bufferedWriter.write("Subject: Correo rechazado.\n");
                bufferedWriter.write("El usuario: <" + smtp.getFrom() + "> con ip <
                        + smtp.getMessage());
                bufferedWriter.flush();
            }
        } catch (IOException ex) {
            System.err.println("Error al enviar correo a root.");
        }
    }

    /*
     * El siguiente metodo establece el filtro de captura de paquetes
     * el filtro, solo filtra paquetes que van dirigidos al puerto 25 en la red
     * 192.168.1.0/24
     */
    public void setup() {

        pcap = Pcap.openLive("wlan0", snaplen, flags, timeout, errbuf); //establece
```

```
        PcapBpfProgram program = new PcapBpfProgram();
        String expression = "dst port 25 and net 192.168.1.0/24"; //expresion
        int optimize = 0;
        int netmask = 0xFFFFFF00; //mascara

        if (pcap.compile(program, expression, optimize, netmask) != Pcap.OK) {
            System.err.println(pcap.getErr());
            return;
        }

        if (pcap.setFilter(program) != Pcap.OK) {
            System.err.println(pcap.getErr());
            return;
        }

        if (pcap == null) {
            System.err.printf("Error al abrir el dispositivo para la captura: "
                    + errbuf.toString());
        }
    }

    /*
     * El siguiente metodo resuelve el nombre de una ip
     */
    public static InetAddress getIpFromName(String host) {
        InetAddress inetAddress = null;
        try {
            inetAddress = InetAddress.getByName(host);

        } catch (UnknownHostException ex) {
            System.err.println("El dominio: " + host + " no se puede resolver");
            inetAddress = null;
        }
        return inetAddress;
    }

    public void setListInvalidWords(List<String> st) {
        this.listInvalidWords = st;
    }

    /*
     * El siguiente metodo inicia la caputa de paquetes.
     */
    public void jPcapLoop() {
```

```
        pcap.loop(-1, jpacketHandler, "");
    }

    /*
     * El siguiente metodo termina la captura de paquetes
     */
    public void jPcapBreakLoop() {
        pcap.breakloop();

    }

    /*
     * El siguiente metodo cierra la captura
     */
    public void jPcapClose() {

        pcap.close();
    }

    public List<Smtp> getInvalidEmails() {
        return invalidEmails;
    }

    /*
     * El siguiente metodo sobreescribe el metodo run para iniciar la captura de pa
     */
    @Override
    public void run() {

        jPcapLoop();
    }

    /*
     * El siguiente metodo manda una interrupcion del hilo en ejecucion
     * y borra todas las reglas de las IP's invalidas.
     */
    @Override
    public void interrupt() {
        //borra reglas iptables de entrada de pauqetes al servidor
        for (int i = 0; i < this.invalidIPs.size(); i++) {
            dropRuleInput();
        }
        //termina la captura de paquetes
        jPcapBreakLoop();
        jPcapClose();
```

```
    }
}
```

B.4. Smtp.java

```
 package protocols;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;
import java.util.StringTokenizer;

public class Smtp {

    private String ipSource; //IP Fuente
    private String from; //Remitente
    private List<String> toList;  //Lista de destinatarios
    private String subject; //Asunto
    private String messageID; //ID del Mensaje
    private String date; //Fecha
    private String userAgent;  //UserAgent
    private String mimeVersion;  //MimeVersion
    private String contentType;  //ContentType
    private String contentTransfer;//ContentTransfer
    private String message = ""; //Mensaje Completo
    private String bodyMessage = ""; //Cuerpo del mensaje
    private String tmp = ""; //
    private String routeMessage; //Ruta del correo en la cola de qmail
    private int stmpHashCode; //Hash Code del correo
    private boolean isMessage = false; //
    private boolean isBodyMessage = false;
    private boolean quit = false;
    private boolean enter = false;
    private boolean isValid = true;
    private boolean isEndMessage = false;
    private boolean islocal = true;
    private List<String> listInvalidWords; //palabras invalidas pasadas por la inte

    public Smtp(int smtpHashCode, List<String> listInvalidWords) {
        this.stmpHashCode = smtpHashCode;
        this.toList = new ArrayList<>();
        this.listInvalidWords = listInvalidWords;
    }
```

```
/*
 * Si se recibio un correo invalido el siguiente metodo
 * borra el Asunto y Cuerpo del Mensaje
 * para que lleguen al destinatario como borreos vacios.
 */
public String newMessage(){
    String[] split = this.message.split("\n");
    String temp = "";
    for (String string : split) {
        if(string.startsWith("Subject: ")){
            temp += "Subject: " + "\r\n";
            continue;
        }
        if(string.startsWith("Content-Transfer-Encoding:")){
            temp += string + "\r\n";
            break;
        }


        temp += string + "\r\n";
    }
    return temp;
}

public String getMessage() {
    return message;
}

public String getRouteMessage() {
    return routeMessage;
}

public String getSubject() {
    return subject;
}

public List<String> getToList() {
    return toList;
}

public void setIslocal(boolean islocal) {
    this.islocal = islocal;
}
```

```
public String getIpSource() {
    return ipSource;
}

public String getFrom() {
    return from;
}

public String getBodyMessage() {
    return bodyMessage;
}

public boolean isValid(){
    return this.isValid;
}

public String getDate() {
    return date;
}

public String getMessageID() {
    return messageID;
}

/*
 * El siguiente metodo
 */
public Smtp set(String string) {

    if(quit){
        return null;
    }

    /*
     * Entra a este bloque siempre y cuando no sea el mensaje que se va a envia
     */
    if (!isMessage ) {

        if (string.startsWith("EHLO [")) { //EHLO [192.168.1.102]
            this.ipSource = string.substring(string.indexOf("[") + 1, string.in
            return null;
        }
        if (string.startsWith("MAIL FROM:<")) { //MAIL FROM:<neo@mail.domain.co
            this.from = string.substring(string.indexOf("<") + 1, string.indexO
            return null;
```

```
        }
        if (string.startsWith("RCPT TO:<")) { //RCPT TO:<neo@mail.domain.com>
            String to = string.substring(string.indexOf("<") + 1, string.indexO
            toList.add(to);
            return null;
        }
        if (string.startsWith("DATA")) { //DATA
            isMessage = true;
            return null;
        }
    }

    //si es un mensaje que va dirigido al mismo dominio no se almacena
    if(islocal)
        return null;

    //si se termina la transmision del mensaje se muestra el contenido en la te
    if (string.startsWith("QUIT")) {
        quit = true;
        System.out.println("_____________________________________");
        System.out.println(this);
        System.out.println("_____________________________________");
        return this;
    }

    /*
     * Entra a sete bloque cuando es el mensaje que se envio
     */
    if (isMessage) {

        this.message += string;
        String[] stringSplit = string.split("\r\n");


        for (String s : stringSplit) {

            /* se captura unicamente el cuerpo del mensaje
             * y se valida para determinar si el mensaje enviado es valido
             */

            if(isBodyMessage){
                this.bodyMessage += s + "\n" ;
                this.validate(this.bodyMessage);
                continue;
            }
```

```
if (s.startsWith("Message-ID:")) {
    this.messageID = s.substring(s.indexOf("<") + 1, s.indexOf(">")
    String temp = getRuteFileEmail(this.messageID);
    if(temp != null){
        this.routeMessage = temp.substring(0,temp.indexOf(":"));
        System.out.println(this.routeMessage);
    }else
        System.err.println("No se puede encontrar el archivo del Me

    continue;
}

if (s.startsWith("Date:")) {
    this.date = s.substring(s.indexOf(":") + 2, s.indexOf("-") - 1)
    continue;
}

if (s.startsWith("From:")) {
    continue;
}

if (s.startsWith("User-Agent:")) {
    this.userAgent = s.substring(s.indexOf(":") + 2);
    continue;
}

if (s.startsWith("MIME-Version:")) {
    this.mimeVersion = s.substring(s.indexOf(":") + 2);
    continue;
}

if (s.startsWith("To:")) {
    continue;
}

//valida el asunto del correo y determina si el correo es valido o
if (s.startsWith("Subject:")) {
    this.subject = s.substring(s.indexOf(':') + 2);
    this.validate(this.subject);
    continue;
}

if (s.startsWith("Content-Type:")) {
    this.contentType = s.substring(s.indexOf(":") + 2);
```

```
                continue;
            }

            if (s.startsWith("Content-Transfer-Encoding:")) {
                this.isBodyMessage = true;
                this.contentType = s.substring(s.indexOf(":") + 2);
                continue;
            }
        }

    }
    return null;
}

/*
 * Este metodo compara el Hash Code de los correos para
 * determinar si pertenecen al mismo correo enviado
 */
public boolean compareSmtpHashCode(int smtp2HashCode) {
    if (smtp2HashCode == this.stmpHashCode) {
        return true;
    }
    return false;
}

//Valida las palabras tanto del cuerpo del mensaje como del asunto
public void validate(String string) {
    string = string.toLowerCase();
    StringTokenizer st = new StringTokenizer(string);
    List<String> messageTokens = new ArrayList();

    while (st.hasMoreTokens()) {
        messageTokens.add(st.nextToken());
    }
    for (String invalidToken : this.listInvalidWords) {

        for (String messageToken : messageTokens) {

            if (messageToken.contains(invalidToken)) {
                this.isValid = false;
                break;
            }
        }
    }
}
```

```
    //determina la ruta del archivo que contiene el correo invalido
    public String getRuteFileEmail(String string) {
        String command = "grep -r " + string + " /var/qmail/queue/mess";
        String route = null;
        try {

            Process exec = Runtime.getRuntime().exec(command);

            BufferedReader stdInput = new BufferedReader(new InputStreamReader(exec

            route = stdInput.readLine();

        } catch (IOException ex) {
            System.err.println("No se pudo ejecutar: " + command);
        }
        return route;
    }

    /*
     * Se sobreescribe el metodo toString
     */

    @Override
    public String toString() {
        String email = "El correo fue enviado desde: \n"
                + ipSource + "\n"
                + "From: " + from + "\n";

                for (String toString : toList) {
                    email += "To: " + toString + "\n";
                }

                email += "Asunto: " + this.subject + "\n"
                + "Mensaje: \n" + this.message + "\n";

        return email;
    }
}
```

C. Scripts de inicio

C.1. Script de inicio qmail, bind

```
#!/bin/sh
```

```
csh -cf '/command/svscanboot &'
qmailctl start
sudo /usr/sbin/tcpserver -x /etc/tcp.smtp.cdb -u 1002 \
-g 1001 0 smtp /var/qmail/bin/qmail-smtpd &

../fw_flush

sudo iptables -F
sudo iptables -X
sudo iptables -Z
sudo iptables -t nat -F

sudo iptables -P INPUT ACCEPT
sudo iptables -P OUTPUT ACCEPT
sudo iptables -P FORWARD ACCEPT
sudo iptables -t nat -P PREROUTING ACCEPT
sudo iptables -t nat -P POSTROUTING ACCEPT

sudo iptables -t nat -A POSTROUTING -s 192.168.8.0/24 -o p1p1 -j MASQUERADE
```

C.2. Eliminar Iptables

```
 #!/bin/sh
##/usr/local/bin/fw_flush
#flush script, which deletes all active rules
#and chains, and resets default policies to "accept"
#this is like having no firewall at all
#define variables
ipt="/sbin/iptables"
echo "The firewall is now being shut down. All policies are set to
ACCEPT, all rules and chains are deleted, all counters are set to zero."
#Set default policies to ACCEPT everything
$ipt -P INPUT ACCEPT
$ipt -P FORWARD ACCEPT
$ipt -P OUTPUT ACCEPT
$ipt -t nat -P OUTPUT ACCEPT
$ipt -t nat -P PREROUTING ACCEPT
$ipt -t nat -P POSTROUTING ACCEPT
$ipt -t mangle -P INPUT ACCEPT
$ipt -t mangle -P OUTPUT ACCEPT
$ipt -t mangle -P FORWARD ACCEPT
$ipt -t mangle -P PREROUTING ACCEPT
$ipt -t mangle -P POSTROUTING ACCEPT
#Zero out all counters
$ipt -Z
```

```
$ipt -t nat -Z
$ipt -t mangle -Z
# Flush all rules, delete all chains
$ipt -F
$ipt -X
$ipt -t nat -F
$ipt -t nat -X
$ipt -t mangle -F
$ipt -t mangle -X
```

C.3. Inicialización Sistema de Seguridad

```
 #!/bin/sh

java -Djava.library.path="./jnetpcap-1.4.r1300" -jar "ClassicPcapExample_Swing_12.j
```

D. Bibliografía

[1] A. S. Tanenbaum, "Network Security", *Computer Networks*, 4th ed.,NJ: Prentice Hall, 2003, pp. 721-828.

[2] E. Mailwald, *Network Security a Beginner's Guide*, 2th ed.,New York: McGraw-Hill, 2003.

[3] C. Alcócer, "Servicios de aplicación: TELNET, FTP, TFTP, NFS, SMTP, POP3, IMAP4, WWW Y GOPHER", *Redes de computadoras*, 1th ed., Lima: Infolink, 2000, pp. 318-337.

[4] A. López, A. Novo, *Protocolos de Internet : diseño e implementación en sistemas UNIX*, 1th ed., D.F., Alfaomega, 2000.

[5] S. Garfinkel, G. Spafford and A. Schwartz, *Practical Unix and Internet Security*, 3th ed.,USA: O'Reilly, 2003.

[6] G. N. Purdy, *Linux iptables Pocket Reference*, 1th ed., USA: O'Reilly, 2004.

[7] D. Sill, *The qmail Handbook*, 1th ed. Usa: Apress, 2002.

[8] K. Wheeler, *Qmail Quickstarter*, 1th ed., USA: Packt Publishing, 2007.

[9] A. D. Ramos León, "Dispositivo de Seguridad Implementado en un Sistema Empotrado con Software Libre", Propuesta de Proyecto Terminal de Ingeniería en Computación, Departamento de Electrónica, UAM-Azc.,Ciudad de Méx., D.F., Méx., 2011.

[10] N. Guzmán Gonzalez, "Aplicación de Distintas Técnicas de Minería de Datos para el Tratamiento de Información", Propuesta de Proyecto Terminal de Ingeniería en Computación, Departamento de Sistemas, UAM-Azc.,Ciudad de Méx., D.F., Méx., 2010.

[11] J. A. Castillo Cabrera, "Diseño e Implementación de una LAN Virtual con Switches de Red", Propuesta de Proyecto Terminal de Ingeniería en Computación, Departamento de Electrónica, UAM-Azc.,Ciudad de Méx., D.F., Méx., 2009.

[12] jNetPcap:Sly Technologies, Mayo 2013. [Online]. Disponible en: *http://www.jnetpcap.com/.*

[13] A. López, "Aprendiendo a programar con Libpcap", 2005.

[14] L. M. Gracia, "Programming with Libpcap - Sniffing the Network From Our Own Application", *HAKING, practical protection hard core it security magazine*, vol. 3, pp. 38-46, Febrero 2008.