UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

“Sistema de prevención de intrusos basado en OpenBSD y Snort”

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACIÓN

P R E S E N T A N:

MARCO ANTONIO MOLINA ANGUIANO

VÍCTOR ENRIQUE GONZÁLEZ SALCEDO

Directora de Tesis: M.C. Ma. Jaquelina López Barrientos

México JUNIO 2008

Índice General

Índice de Tablas y Figuras

Introducción

El fundamento de Internet es TCP/IP, un protocolo de transmisión que asigna a cada computadora que se conecta a la red un número específico y único, llamado dirección IP como por ejemplo 92.250.26.11. El protocolo TCP/IP sirve para establecer una comunicación entre dos puntos remotos mediante el envío de información en paquetes. Al transmitir un mensaje o una página con imágenes, el bloque completo de datos se divide en pequeños paquetes que viajan de un punto a otro en la red entre dos números IP determinados, cada uno de los paquetes viaja a través de una ruta disponible. La información viaja por muchas computadoras intermedias a modo de repetidoras hasta alcanzar su destino, lugar en el cual todos los paquetes se juntan y reordenan regenerando la información original.

El protocolo TCP/IP fue diseñado inicialmente para transmitir información entre distintas computadoras, resolviendo una necesidad básica de comunicación, pero nunca se pensó en que la información viajara de forma segura, por lo tanto el protocolo no tiene seguridad incluida por defecto. Debido a esta carencia se pueden aprovechar los errores de configuración, administración, programación y/o errores humanos para poder obtener privilegios en los servidores que aloja Internet. Esta situación donde el protocolo no brinda una seguridad se traduce en un riesgo constante donde estamos expuestos literalmente a una fauna nociva que va desde los ya viejos conocidos virus hasta los más recientes caballos de Troya y amenazas combinadas. Los riesgos no sólo son por los defectos que tiene el protocolo sino que se le suman los defectos de todo aquel software que esté expuesto en la red y accesible desde cualquier punto en Internet.

Junto con el crecimiento exponencial de Internet y los riesgos en los que nos encontramos al conectarnos a una red pública tan grande, ha nacido la necesidad de crear una disciplina que nos mantenga, o trate de hacerlo, en la certeza de que nuestro equipo y nuestra información se mantendrán como nosotros queremos que lo hagan. Esta disciplina que se encuentra dentro de las Tecnologías de la Información se llama Seguridad Informática. La Seguridad Informática nos brinda una serie de lineamientos, buenas prácticas, estándares informáticos, estándares de calidad que nos ayudan a mantener nuestra información segura y fuera de la mayoría de los riesgos.

Actualmente dentro de la seguridad informática podemos encontrar diversas herramientas tanto para proteger nuestro entorno de red como nuestros sistemas operativos y los servicios que montamos sobre ellos, haciendo una combinación podemos brindar un mejor desempeño a nuestras aplicaciones.

Para hacer frente a las amenazas y disminuir los riesgos que encontramos al conectarnos a una red pública como lo es Internet se definen una serie de servicios para proteger los sistemas de proceso de datos y de transferencia de información de una organización. Estos servicios hacen uso de uno o varios mecanismos de seguridad para poder cumplir con sus objetivos. Actualmente la mayoría del software y los sistemas operativos cumplen con la mayoría de los servicios de seguridad haciendo uso de varios mecanismos, así podemos encontrar herramientas muy poderosas que nos ayudan a proteger nuestras redes y aplicaciones. Dos de las herramientas más importantes son los Firewalls y los sistemas detectores de intrusos. Que entre otros servicios cumplen con confidencialidad, integridad, disponibilidad y control de acceso. Entre los sistemas operativos en los que podemos encontrar un buen desempeño de seguridad está la familia de sistemas operativos UNIX.

En la década de los cincuentas, Ken Thompson quien trabajaba para los laboratorios Bell decidió realizar un sistema operativo para la computadora PDP-11/20 denominado UNICS (*Uniplexed Information and Computing System*), más adelante se le denominó UNIX. Uno de los objetivos de dicho sistema era que pudiera permitir el trabajo de varios usuarios al mismo tiempo, el proyecto entusiasmó tanto a sus colegas que pronto se le unió Dennis Ritchie. Casi al mismo tiempo Dennis Ritchie junto con Brian Kernighan desarrollaron el lenguaje de programación C, junto con su compilador lo que permitió que UNIX se escribiera en ese lenguaje permitiendo entonces la portabilidad a cualquier computadora para la cual existiera un compilador de C. En 1984 AT&T lanzó el primer producto comercial de UNIX.

Una de las Universidades que adquirió UNIX fue la universidad de California Berkeley y como se distribuía junto con el código fuente en dicha universidad comenzaron a mejorar el sistema, y entre dichas mejoras se introdujo el uso de redes a través del protocolo TCP/IP. La versión desarrollada por Berkeley se denominó BSD

(*Berkeley Software Distribution*). Estos actos propiciaron que existieran dos grandes versiones de UNIX, la de AT&T y la de Berkeley.

OpenBSD es un sistema operativo libre tipo UNIX, multiplataforma, basado en 4.4BSD de la universidad de Berkeley, es un descendiente de NetBSD, centrado en seguridad y criptografía. Este sistema operativo se concentra en la portabilidad, cumplimiento de normas y regulaciones, seguridad proactiva y criptografía integrada. Se distribuye bajo la licencia BSD, aprobada por la OSI (*Open Source Initiative*). Su uso está muy difundido en Internet como herramienta de red y seguridad como Firewalls y sistemas detectores de intrusos por brindar una flexibilidad mayor a los sistemas de código cerrado.

El presente trabajo de tesis se desarrolla para cumplir con los siguientes objetivos:

- Instalación de un perímetro de seguridad utilizando ambientes UNIX con OpenBSD.
- Hacer una integración, mediante una nueva herramienta, de dos de las utilerías básicas de seguridad, Firewalls y detectores de intrusos para que funcionen en conjunto con un sistema de prevención de intrusos.
- Desarrollo de la herramienta de seguridad que realice la integración mencionada.
- Realización de pruebas de rendimiento con ataques reales en ambientes controlados.

Capítulo

1. Fundamentos de redes

1.1 Modelo OSI

En la década de 1980 existían muchas tecnologías diferentes para la creación de redes de computadoras, en esa época las grandes empresas veían en el uso de las redes una gran oportunidad de aumentar su productividad, por tanto las redes comenzaron a crecer casi sin control. Tiempo después, se vio la necesidad de comunicar las distintas redes existentes, esto no siempre era posible debido a la incompatibilidad entre las tecnologías empleadas para la implementación de cada una de las redes, haciendo muy difícil y en algunos casos, imposible, la comunicación entre ellas. En ese tiempo las implementaciones que más figuraban eran tres, SNA (*Systems Network Architecture*) de IBM, DECnet de Digital y OSI (*Open System Interconnection*) de ISO (*International Organization for Standardization*); el gobierno de Estados Unidos decidió utilizar y apoyar productos que estuvieran basados en OSI por eso su utilización fue más difundida. El modelo OSI divide los problemas de comunicación entre redes en tareas más sencillas acomodando cada tipo de problema en una capa determinada. El modelo de referencia OSI tiene siete capas, que a continuación se describen con detalle.

Capa siete: Aplicación

La capa de aplicación es la que se encuentra en la parte superior del modelo de referencia OSI y es por tanto la que interactúa directamente con las aplicaciones de usuario, entre sus funciones está la de proporcionar servicios de red a programas como los procesadores de texto, hojas de cálculo, etc. También tiene la función de sincronizar y establecer acuerdos en los procedimientos para la recuperación de errores e integridad en el control de datos. Algunos ejemplos de protocolos de capa siete son FTP y HTTP.

Capa seis: Presentación

La capa de presentación se encarga de asegurarse que la información que fluye entre las capas de aplicación de dos sistemas que están en comunicación va a poder leerse adecuadamente, esto lo hace traduciendo múltiples formatos de datos a un formato común, es también en esta capa donde se da el proceso de cifrado y descifrado de datos. Algunos protocolos comunes de esta capa son SSL y TLS, que se encargan de las comunicaciones cifradas.

Capa cinco: Sesión

Como su nombre lo indica, la capa de sesión se encarga de establecer, administrar y finalizar la comunicación entre dos hosts diferentes. Esta capa también tiene la función de dar servicio a la capa superior encargándose de sincronizar el diálogo entre las capas de presentación de hosts diferentes. Un protocolo que se encuentra en esta capa es NetBios, que se encarga de brindar una especificación para conectarse a la red.

Capa cuatro: Transporte

La capa de transporte puede decirse que es la frontera entre los protocolos de aplicación y los de transmisión de datos. La capa de transporte se encarga de segmentar los datos para que puedan ser transmitidos desde un host a otro, encargándose de ocultar los detalles de la implementación del transporte a las capas superiores. Para brindar un servicio confiable se emplea la detección y recuperación de errores en el transporte así como la información en el control de flujo. Ejemplos de protocolos en esta capa son, TCP y UDP.

Capa tres: Red

La capa de red se encarga de brindar la conectividad y seleccionar la mejor ruta para el flujo de la información entre dos hosts que pueden estar ubicados en lugares geográficamente muy apartados. Además esta capa se encarga del direccionamiento lógico. Ejemplos de protocolos de esta capa son IP, IPX.

Capa dos: Enlace de datos

Esta capa proporciona un tránsito de datos fiable a través de un enlace físico. Esta capa se encarga del direccionamiento físico, de la topología de la red, del acceso a la red, de la notificación de errores, de la distribución ordenada de tramas y del control del flujo. En esta capa se encuentran las siguientes tecnologías de red, Ethernet, Frame Relay, ATM, PPP y Wi-Fi.

Capa uno: Física

Define las especificaciones mecánicas y eléctricas para activar, mantener y desactivar el enlace físico entre sistemas finales. Aquí se definen entre otras cosas los niveles de voltaje, la velocidad en la transmisión de datos, los tipos de conectores, así como el tipo de material empleado para la conexión entre otras cosas. Muchas personas identifican a esta capa por el tipo de cables que se utilizan para conectarse al medio, algunos pueden ser, cable coaxial, fibra óptica, cable por par trenzado, microondas y cable serial (RS-232).

1.2 Modelo TCP/IP

TCP/IP surgió de la necesidad del departamento de la defensa de los Estados Unidos de crear una red que pudiera sobrevivir a cualquier condición, inclusive un ataque nuclear. El objetivo era que la información pudiera fluir de un punto a otro aún cuando alguna parte de la red estuviera dañada o fuera de funcionamiento. Esto representaba un problema de diseño muy grande y complejo que derivó en la creación del modelo TCP/IP sobre el cual Internet tuvo su crecimiento.

El modelo TCP/IP está dividido en cuatro capas: aplicación, transporte, Internet y de acceso a red. Cabe mencionar que algunas de las capas de este modelo tienen el mismo nombre que en el modelo de referencia OSI, sin embargo, esto no significa que tengan el mismo funcionamiento, de hecho algunas de las capas del modelo TCP/IP pueden abarcar más de una capa del modelo OSI. TCP/IP combina las capas de presentación y sesión del modelo OSI en la capa de aplicación, las capas de enlace de datos y física en la capa de acceso a red, como se muestra en la Figura 1.1. A continuación describimos con detalle cada una de las cuatro capas del modelo TCP/IP.

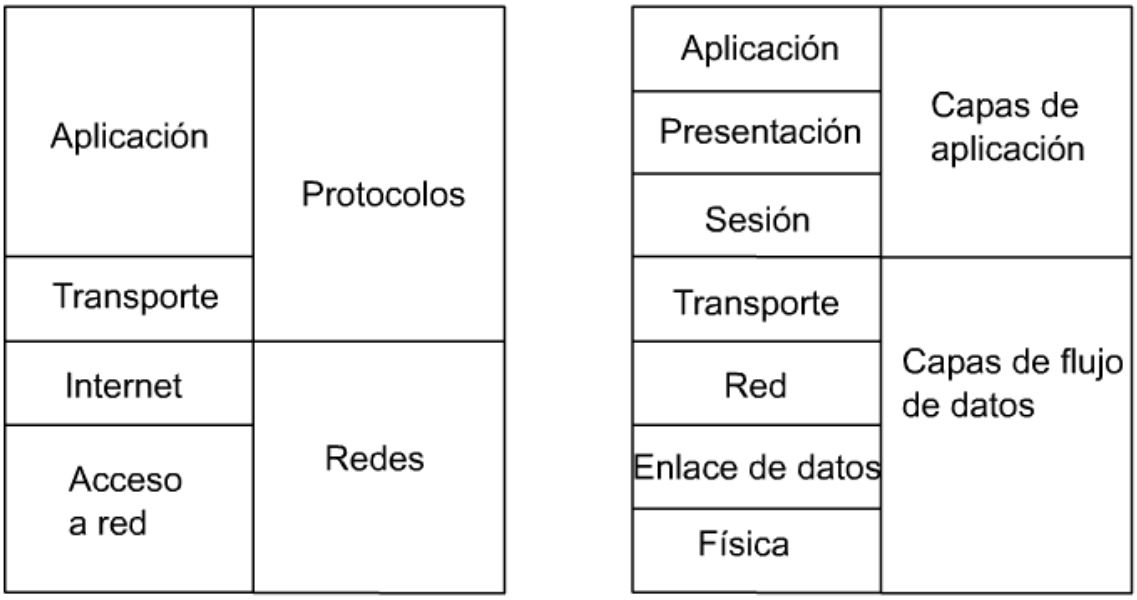

Figura 1.1 Modelo TCP/IP y OSI

Capa de acceso a red

La capa de acceso a red también es conocida como capa de host a red. Es la capa que se encarga de crear un enlace físico con el medio de red. Una de las funciones de esta capa es dirigir (rutear) los datos entre los hosts de una LAN. Los servicios que provee esta capa son control de flujo y control de errores de datos que viajan entre los distintos hosts. Esta capa provee los manejadores (*drivers*) para que la capa superior, es decir, la capa de Internet pueda tener acceso al medio a través de las distintas tecnologías como pueden ser Ethernet, Frame Relay, ATM, PPP y Wi-Fi.

Capa de Internet

El propósito de la capa de Internet es enviar paquetes desde un dispositivo a otro, utilizando dentro de esta capa el protocolo adecuado. En esta capa se determina cuál es la mejor ruta para entregar los paquetes. Dentro de esta capa se encuentra el protocolo IP que realiza las funciones de capa de red del modelo OSI para el modelo TCP/IP. Dentro de esa capa operan los protocolos IP, ICMP, ARP y RARP.

Capa de Transporte

Esta capa se encarga de proporcionar los servicios de transporte de información desde un host a otro, construyendo una conexión lógica entre ambos hosts. Los protocolos de esta capa se encargan de

segmentar los datos para poder ser enviados, además de reensamblarlos cuando llegan a su destino, también se encargan del envío de segmentos desde un dispositivo final a otro dispositivo final. La capa de transporte asume que puede utilizar la red como una nube para enviar paquetes de datos desde un origen hasta un destino. En esta capa trabajan los protocolos TCP y UDP.

Capa de aplicación

La capa de aplicación, manipula protocolos de alto nivel y temas de presentación, codificación y control de diálogo, dentro de estos protocolos se encuentran HTTP, FTP, NFS, SMTP, SNMP, DNS, así como Telnet. Esta capa es la que engloba más capas del modelo OSI, TCP/IP contempla además de la capa de aplicación la de transporte y la de presentación, por lo que también en esta capa podemos encontrar protocolos de cifrado como SSL y TLS además de aplicaciones que utilizan estos protocolos como SSH.

1.3 Direccionamiento IP

El protocolo IP es la aplicación más popular de un esquema de direccionamiento de red y es el utilizado por Internet. Cuando la información fluye por las capas de los modelos de referencia TCP/IP y OSI los datos se van encapsulando. En la capa de red los datos se encapsulan en paquetes, también llamados datagramas. El protocolo IP determina la forma de la cabecera del paquete, que incluye direccionamiento e información de control sin importar los datos, acepta cualquier dato que se pase de las capas superiores. El paquete de la capa tres (OSI) se convierte en los datos de la capa dos, que se encapsulan en tramas. De forma similar el paquete de la capa IP se forma con la información de las capas superiores además de una cabecera IP, que está formada por los campos que se muestran en la Figura 1.2 que posteriormente se explican con detalle.

- VERS (Versión): Indica la versión de IP utilizada (cuatro bits).

- HLEN (Longitud de la cabecera IP): Indica la longitud de la cabecera del datagrama IP en palabras de 32 bits (cuatro bits).

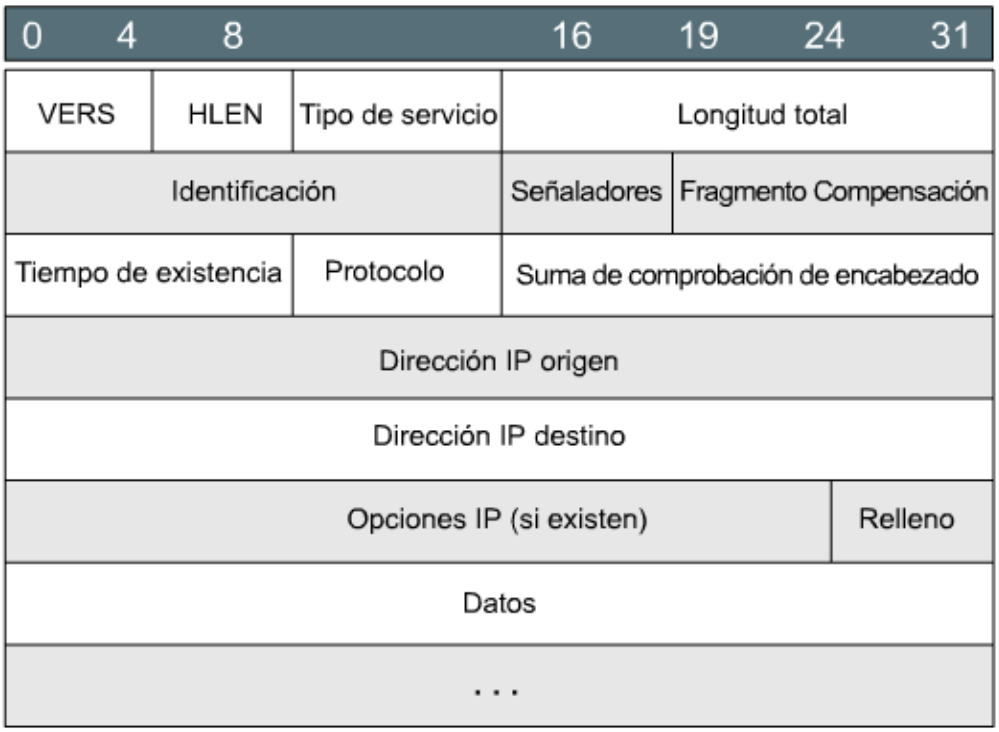

Figura 1.2 Cabecera IP

- Tipo de servicio (TOS): Especifica el nivel de precedencia del paquete que ha sido asignado por un protocolo de capa superior particular (ocho bits).

- Longitud total: Especifica la longitud del paquete IP completo, en bytes, incluyendo datos y cabecera (16 bits).

- Identificación: Contiene un entero que identifica al datagrama actual, es el número de secuencia (16 bits)

- Señaladores (FLAGS): Controlan la fragmentación (tres bits).

- Fragmento de compensación: Ayuda a reconstruir los fragmentos del datagrama (13 bits).

- Tiempo de existencia (TTL): Mantiene un contador que gradualmente disminuye hasta cero, punto en el que el datagrama se descarta evitando que los paquetes entren en un ciclo infinito (ocho bits).

- Protocolo: Indica qué protocolo de capa superior recibe los paquetes entrantes después de que se haya completado el proceso IP (ocho bits).

- Suma de comprobación de encabezado: Ayuda a garantizar la integridad del encabezado IP (16 bits).

- Dirección IP origen: Especifica la dirección IP del nodo emisor (32 bits).

- Dirección IP destino: Especifica la dirección IP del nodo receptor (32 bits).

- Opciones: Permite que IP soporte varias opciones, como la seguridad (longitud variable).

- Relleno: Se agregan ceros adicionales a este campo para garantizar que el encabezado IP siempre sea un múltiplo de 32 bits.

- Datos: Contiene información de la capa superior (longitud variable, máximo 64 Kb).

1.3.1 Direcciones IP

La dirección IP contiene la información necesaria para dirigir (enrutar) un paquete a través de la red. Cada campo de la dirección IP en el datagrama, origen y destino, contiene una dirección de 32 bits. La primera es la dirección IP del dispositivo que envía el paquete (origen) y el segundo contiene la dirección IP del dispositivo que lo recibe (destino).

Una dirección IP se puede representar por un número binario de 32 bits escrito como cuatro octetos (grupos de ocho bits) separados por puntos, como se muestra en la Figura 1.3.

También se puede ver en la Figura 1.3 que las direcciones IP se expresan como números decimales. El valor decimal máximo que se puede expresar con ocho bits es 255.

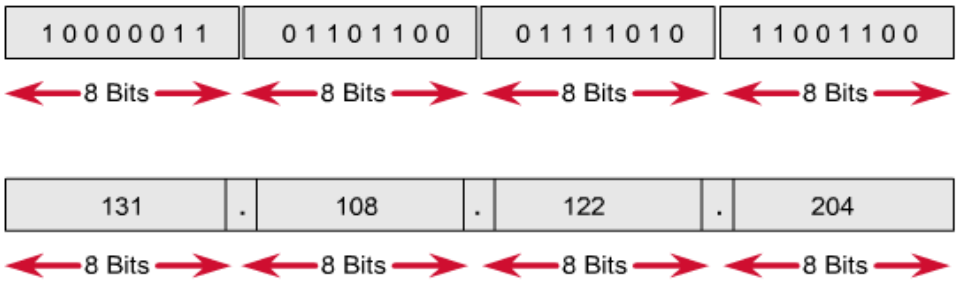

Figura 1.3 Octetos de una dirección IP

La dirección IP de un dispositivo puede dividirse en dos partes. El número de red que identifica la red a la que pertenece el dispositivo y el número de host, que identifica de manera única al dispositivo dentro de una red, ver Figura 1.4.

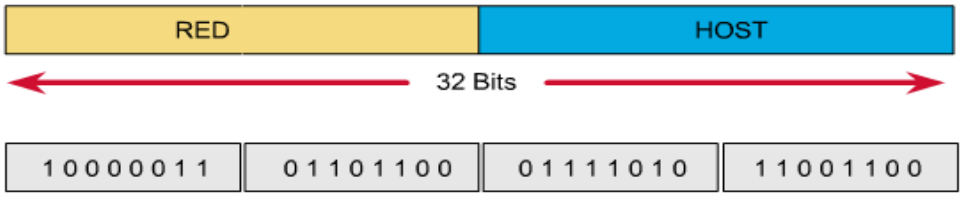

Figura 1.4 Número de red y host

Como las direcciones IP se componen de 32 bits se puede utilizar cualquier cantidad de bits para identificar el número de red y el resto para identificar el número de host. Regularmente se utilizan uno, dos o tres octetos completos para identificar el número de red; de forma similar se pueden usar hasta tres octetos para identificar la parte del host de una dirección IP.

1.3.2 Clases de IP

Hay tres clases de direcciones IP que una organización puede recibir: Clase A, B y C. Adicionalmente existen dos clases extra, D y E, la primera se le conoce como dirección de multicast y la segunda se reservó para uso futuro.

Clase A: Las direcciones de la clase se diseñaron para crear redes extremadamente grandes. Debido a necesidades históricas de este tipo de redes se supuso que serían mínimas, se desarrolló una arquitectura que maximizaba el número posible de direcciones de host pero limitaba severamente el número potencial de redes de clase A que se podrían definir. El primer bit de la dirección de Clase A siempre es cero Figura 1.5. Todas las direcciones IP de Clase A utilizan solamente los primeros ocho bits para identificar la parte de

la red de la dirección. Los tres octetos restantes se pueden utilizar para la parte del host de la dirección.

Clase B: Los primeros dos bits de una dirección de Clase B siempre son uno y cero, Figura 1.5. Las direcciones IP de Clase B siempre tienen valores que van del 128 al 191 en su primer octeto. Todas las direcciones IP de Clase B utilizan los primeros 16 bits para identificar la parte de la red de la dirección. Los dos octetos restantes de la dirección IP se encuentran reservados para la parte del host de la dirección.

Clase C: Los tres primeros bits de una dirección de Clase C siempre son uno, uno y cero, Figura 1.5. Las direcciones IP de Clase C siempre tienen valores que van del 192 al 223 en su primer octeto. Todas las direcciones IP de Clase C utilizan los primeros 24 bits para identificar la porción de red de la dirección. Sólo se puede utilizar el último octeto de una dirección IP de Clase C para la parte de la dirección que corresponde al host.

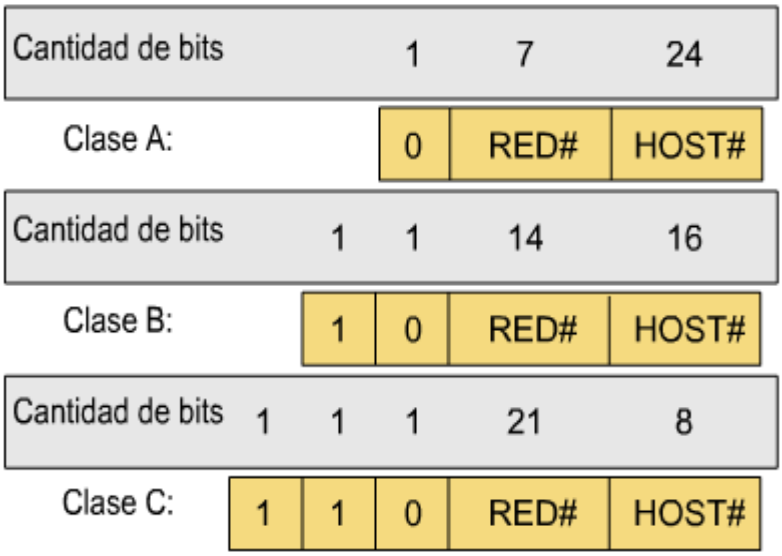

Figura 1.5 Clases de IP

La dirección IP 0.0.0.0 no se utiliza para direccionar un host en Internet, los valores nulos pueden confundir a los sistemas. Pero hay un caso especial con la red de clase A, la red 127.0.0.0. Esta red se ha reservado para acomodar una dirección especial, 127.0.0.1. Esta dirección se utiliza como loopback, los datos que se envían a esta dirección son reenviados en ciclo de vuelta al emisor sin atravesar red alguna. Todos los hosts en TCP/IP utilizan esta dirección para referirse a ellos mismos (*localhost*). Debido a que cada dispositivo permitido por IP cree que está conectado a la red

127.0.0.0 el enrutamiento a esta red es problemático y no es usada en Internet.

Otra dirección IP especial es la dirección de broadcast (difusión). Una difusión ocurre cuando un emisor envía datos a todos los dispositivos de una red. Para asegurar que todos los dispositivos de la red presten atención a la difusión el emisor debe utilizar una dirección IP de destino que todos ellos puedan reconocer y seleccionar. Las direcciones IP de difusión terminan con unos binarios en toda la parte del host de la dirección, esto significa que las direcciones de difusión ocupan el número decimal 255.

1.3.3 CIDR

Una función añadida recientemente al direccionamiento IP es el enrutamiento entre dominios sin clase, CIDR (*Classless Inter-Domain Routing*). CIDR nació de la crisis que acompañó al crecimiento explosivo de Internet a principio de los años noventa. Las razones específicas para la creación de esta función fue el agotamiento de las direcciones de red IPv4 que quedaban por asignar, el espacio de Clase B estaba en peligro de agotamiento y el rápido crecimiento de las tablas de enrutamiento de Internet como resultado del crecimiento de la red. Para evitar el colapso el IETF (*Internet Engineering Task Force*) visualizó que eran necesarias dos soluciones, una a corto plazo y otra a largo plazo. A largo plazo la única solución era una versión completamente nueva de IP con espacio ampliado y arquitecturas de dirección, que se materializó con IPv6, aún en desarrollo. Lo más complicado a corto plazo era disminuir el agotamiento de las direcciones restantes no asignadas. Parte de la respuesta fue eliminar las ineficientes clases de direcciones a favor de una arquitectura de direccionamiento más flexible.

CIDR permite que los ruteadores de Internet agreguen información de enrutamiento de forma más eficiente. Una sola entrada en una tabla de enrutamiento puede representar los espacios de direcciones de varias redes. Esto reduce mucho el tamaño de las tablas lo que se traduce en un incremento de la escalabilidad.

Cada dirección de red compatible con CIDR se anuncia con una máscara de bits específica, la cual identifica la longitud del prefijo de

red. Por ejemplo la dirección 192.125.61.8/20 especifica una dirección de red de 20 bits.

1.3.4 Protocolo ICMP

Todos los hosts TCP/IP implementan el protocolo de mensajes de control de Internet (ICMP). Los mensajes ICMP se transportan en los datagramas IP y se utilizan para enviar mensajes de error y control. ICMP utiliza los siguientes tipos de mensajes:

- Destination Unreachable (Destino inalcanzable)
- Time to Live Exceeded (Tiempo de existencia superado)
- Parameter Problem (Problema de parámetros)
- Source Quench (Suprimir origen)
- Redirect (Redirigir)
- Echo (Eco)
- Echo Reply (Respuesta de eco)
- Timestamp (Marca horaria)
- Timestamp Reply (Respuesta de marca horaria)
- Information Request (Petición de información)
- Information Reply (Respuesta de información)
- Address Request (Petición de dirección)
- Address Reply (Respuesta de dirección)

Si un ruteador recibe un paquete que no puede enviar a su destino final, envía al origen un mensaje ICMP destino inalcanzable al origen. Es posible que el mensaje no se pueda entregar porque no hay ninguna ruta conocida hacia el destino. Una respuesta de eco es una respuesta exitosa a un comando ping. Sin embargo, los

resultados pueden incluir otros mensajes de ICMP como por ejemplo, límite de tiempo excedido.

1.3.5 Protocolo ARP

ARP se utiliza para resolver o asignar una dirección IP conocida a una dirección de subcapa MAC (*Media Access Control*) para permitir la comunicación a través de un medio de acceso múltiple como Ethernet. Para establecer una conexión, el host origen debe determinar la dirección MAC del host destino con el que desea comunicarse, para esto, el host origen verifica una tabla denominada caché ARP donde están asociadas todas las direcciones IP con su respectiva dirección MAC de todos los hosts que se encuentren dentro de la misma red (o subred).

Si la dirección MAC no figura en la tabla, ARP envía un broadcast que reciben todos los hosts de la red, al recibir este mensaje todos los host responden, enviando su propia dirección MAC con lo que el host origen podrá actualizar su tabla ARP y entonces establecer la comunicación.

El término ARP local se utiliza para describir la búsqueda de una dirección cuando el host que la solicita y el host destino comparten el mismo medio o cable. Antes de emitir el ARP, se debe consultar la máscara de subred[1] que determina qué nodos se encuentran en la misma subred.

1.4 Protocolos de capa de transporte

Dentro de la capa de transporte se lleva a cabo el establecimiento de la conexión para el intercambio de información entre dos hosts o computadoras distintas, para esto, dentro de esta capa se cuenta con dos protocolos muy importantes los cuales son TCP y UDP.

[1] La máscara de subred sirve para identificar los bits que se utilizan para identificar el número de red y el número de host.

1.4.1 Protocolo TCP

Como se mencionó, TCP (*Transmission Control Protocol*) trabaja sobre la capa de transporte y se encarga de proporcionar una transmisión fiable de datos orientada a la conexión. TCP es el responsable de dividir la información en pequeños paquetes de datos llamados segmentos, también se encarga de reensamblar los segmentos en el momento que se reciben, esto lo logra identificando de alguna manera cada uno de ellos. También se encarga de asegurarse que los segmentos hayan llegado a su destino y si no, de reenviar los segmentos que no se hayan recibido. Los protocolos más comunes que utilizan TCP son HTTP (*HyperText Transfer Protocol*), SMTP (*Simple Mail Transfer Protocol*) y SSH (*Secure Shell*). La Figura 1.6 muestra el formato que tiene un segmento TCP.

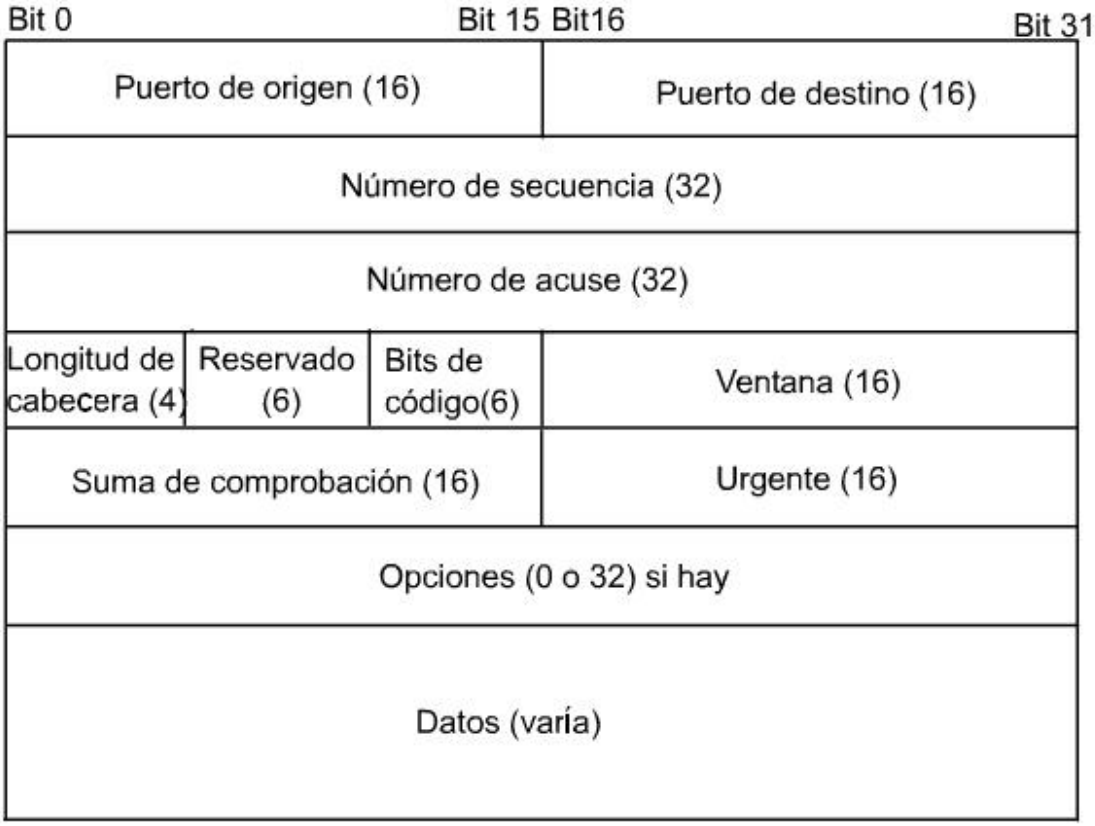

Figura 1.6 Formato segmento TCP

- **Puerto de origen**: Número del puerto que realiza la llamada.
- **Puerto de destino**: Número del puerto que es llamado.
- **Número de secuencia**: Número utilizado para garantizar la secuenciación correcta de la llegada de datos.
- **Número de acuse**: Siguiente octeto TCP esperado.
- **Longitud de cabecera**: Número de palabras de 32 bits de la cabecera.
- **Reservado**: Establecido a cero.
- **Bits de código**: Funciones de control (como configuración y finalización de una sesión).
- **Ventana**: Número de octetos que el emisor es capaz de aceptar.
- **Suma de comprobación**: Suma de comprobación que incluye la cabecera y los campos de datos.
- **Puntero urgente**. Indicación del final de datos urgentes.
- **Opciones**: Una opción definida actualmente es el tamaño máximo del segmento TCP.
- **Datos**: Datos del protocolo de capa superior.

1.4.2 Intercambio de señales en tres pasos

Con un protocolo orientado a la conexión como lo es TCP existen un conjunto de procedimientos que toman lugar para establecer una conexión. Existen varios estados durante una conexión TCP de este tipo. Para establecer una conexión entre dos hosts se debe establecer el inicio de la misma, para esto se deben sincronizar sus ISN (Números de secuencia inicial, por sus siglas en inglés). La sincronización para el intercambio de información se efectúa mediante el envío de segmentos especiales que portan un bit llamado SYN (por sincronización en inglés) el segmento que lleva el bit SYN también se denomina SYN. La sincronización requiere que cada parte envíe su número de secuencia inicial y reciba una

confirmación ACK (por agradecimiento en inglés) desde el otro extremo, en pocas palabras cada lado debe recibir el ISN y enviar una señal ACK. Para efectuar esto se siguen los siguientes pasos:

- La computadora A envía a B su número de secuencia inicial “X”, con el bit ACK ajustado a cero (desactivado) mientras que el bit SYN se encuentra ajustado a uno (activado).

- La computadora B recibe el segmento y contesta hacia A con su número de secuencia “Y” así como el número de secuencia de A añadido en una unidad, es decir X + 1. Los bits SYN y ACK se encuentran activados por lo que A ya podrá ajustar su número de secuencia y también habrá recibido su mensaje ACK.

- A regresa a B un segmento con su correspondiente número de secuencia, es decir Y + 1 con el bit SYN desactivado y el bit ACK activado por lo que B contará ahora con su mensaje ACK y su número de secuencia ajustado.

La Figura 1.7 muestra este intercambio de segmentos para el establecimiento de una conexión. Este intercambio se denomina intercambio de señales en tres pasos. Es un mecanismo de conexión asíncrono ya que los números de secuencia no están vinculados a un reloj global de la red.

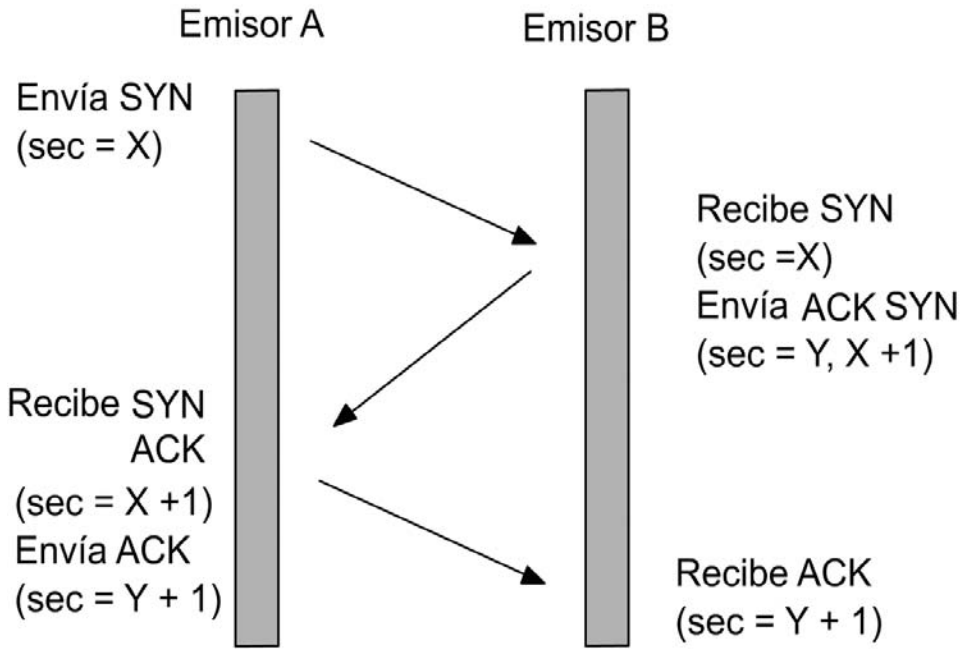

Figura 1.7 Intercambio de señales en tres pasos

1.4.3 Protocolo UDP

UDP es un sencillo protocolo que se ocupa de intercambiar datagramas sin confirmar la garantía de entrega. El procesamiento de errores y la retransmisión deber ser manipulados por protocolos de las capas superiores. Por ejemplo si un operador envía cierto archivo a través de FTP y la comunicación se ve interrumpida por alguna circunstancia el operador deberá volver a enviar la información hasta tener éxito. UDP no utiliza sistema de ventanas (*Windowing*) ni acuses de recibo, por lo tanto son los protocolos de la capa de aplicación los que se encargan de esta tarea. UDP está diseñado para aplicaciones que no requieren la agrupación de secuencias o segmentos. La Figura 1.8 muestra el formato de un segmento UDP

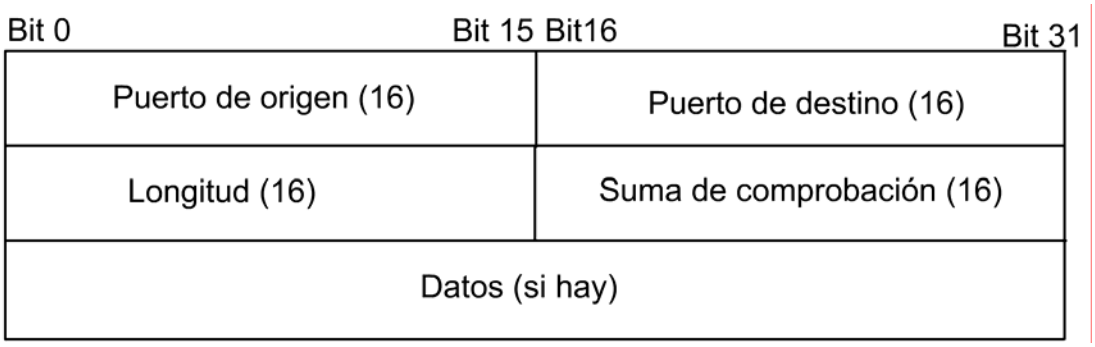

Figura 1.8 Formato segmento UDP

- **Puerto de origen**: Número del puerto que realiza la llamada.
- **Puerto de destino**: Número del puerto que es llamado.
- **Longitud**: Número de bytes, incluyendo la cabecera y los datos.
- **Suma de comprobación**: Suma de comprobación que incluye la cabecera y los campos de datos.
- **Datos**: Datos del protocolo de capa superior.

1.5 Servicios de capa de aplicación

La parte con la que los seres humanos interactuamos con las computadoras es en la mayoría de los casos con la capa de aplicación y específicamente con los distintos servicios que ésta

ofrece. Los servicios de capa de aplicación utilizan puertos específicos para poder interactuar con las capas más bajas, algunos de los servicios que ofrece esta capa son HTTP, Telnet, FTP, SMTP, entre otros.

1.5.1 Puertos de servicios

Los puertos de servicios permiten a un servidor brindar distintos servicios de red sobre los protocolos TCP y UDP, permitiendo así la conexión entre las aplicaciones de distintos hosts. Dentro del segmento TCP o UDP que llega a un determinado host, se encuentra el puerto de servicio al que va dirigido, esta información permite al protocolo dirigir el segmento hacia el servicio de red correspondiente. Todos los servicios de red tienen un puerto de servicio único y diferente por el cual escuchan las distintas solicitudes, por ejemplo, el servicio Web regularmente tiene asignado el puerto 80, por otro lado el servicio de correo electrónico cuyo protocolo es SMTP escucha en el puerto 25.

De acuerdo con IANA (*Internet Assigned Numbers Authority*) existen tres categorías para los puertos de servicios: los puertos bien conocidos, los puertos registrados y los puertos dinámicos y/o privados. Los puertos bien conocidos van del 0 al 1023 y en la mayoría de los sistemas operativos sólo pueden ser usados bajo los privilegios del usuario administrador (root). Los puertos registrados van del 1024 al 49151 y los puertos dinámicos y/o privados van del 49152 al 65535. En la tabla 1.1 se presenta una lista con los servicios de red más comunes y los números de puerto que le corresponden.

Nombre de puerto	Puerto Número / Protocolo
FTP	21/TCP/UDP
SSH	22/TCP/UDP
Telnet	23/TCP/UDP
SMTP	25/TCP
Nicname	43/TCP
Domain	53/TCP/UDP
Finger	79/TCP

Nombre de puerto	Puerto Número / Protocolo
HTTP	80/TCP/UDP
POP3	110/TCP/UDP
Auth	113/TCP
Nntp	119/TCP
Ntp	123/TCP/UDP
Https	443/TCP/UDP

Tabla 1.1 Servicios y sus puertos

Capítulo

2. Fundamentos de seguridad

2.1 Seguridad en cómputo

Para definir seguridad en cómputo primero se explica que es la seguridad en términos no relacionados con las computadoras. La seguridad nos da la certeza de que nuestra salud, integridad física, dinero o cualquier cosa que nosotros apreciemos permanezca como nosotros lo deseamos.

La seguridad nos brinda la confianza necesaria para seguir con nuestras actividades normales sin tener que pensar en protegernos todo el tiempo; si levantamos una barda con alambre de púas alrededor de nuestra casa podemos tener la confianza de permanecer dentro de ella con toda tranquilidad.

De acuerdo a la definición de seguridad dada anteriormente se puede definir la seguridad en cómputo de la siguiente manera: un sistema de cómputo es seguro si se puede confiar en que se comportará como se espera que lo haga, que la información contenida en él no será alterada y se mantendrá accesible durante el tiempo que el dueño de la información lo desee para los usuarios que él mismo determine.

También se dice que un sistema de cómputo es seguro cuando es ajeno a todo riesgo, que está libre de amenazas, que no posee vulnerabilidades, que es confiable y que funciona sin fallas.

La seguridad en cómputo está orientada a ofrecer tranquilidad y confianza a usuarios, empleados, clientes y todos aquellos personajes que se vean involucrados en un sistema de cómputo expuesto a un mundo hostil como lo es Internet.

2.1.1 Amenaza

Recurriendo nuevamente a analogías con definiciones fuera del campo técnico de las computadoras una amenaza es un hecho que pone en peligro algo que nosotros apreciemos de una forma especial, por ejemplo, un asesino es una amenaza latente a la vida de cualquier persona.

Una amenaza es una violación potencial a la seguridad en cómputo. La violación no necesita realmente ocurrir para que haya una amenaza, ya que éstas se encuentran presentes en todo momento

aunque no lleguen a ser consumadas. Las amenazas son factores, circunstancias o eventos que pueden causar un daño a un sistema de cómputo.

Se puede clasificar como amenaza todo aquello que pretenda e intente destruir o alterar los activos[1] de la organización. Cuando la amenaza es consumada se convierte en un ataque.

Las amenazas se pueden clasificar en cuatro amplias clases:

- Acceso: Acceso no autorizado a la información.
- Engaño: Aceptación de información falsa.
- Interrupción: Prevención de la correcta operación del sistema.
- Usurpación: Control no autorizado de cierta parte de un sistema.

La clasificación anterior engloba todas las amenazas en forma general pero no le pone nombre ni apellido a todas aquellas que podemos encontrar al exponer recursos dentro de una red empresarial o Internet, simplemente las ordena. Los nombres de todas aquellas amenazas son tan variados como la forma en que se convierten en ataques, podemos encontrar algunas que son muy comunes como el escaneo de puertos, los virus, los ataques de negación de servicio, sin embargo, la tarea de conocerlas todas es muy difícil por su gran variedad.

Frecuentemente las amenazas aprovechan una vulnerabilidad antes de convertirse en un ataque.

2.1.2 Vulnerabilidad

Una vulnerabilidad es una debilidad de software, hardware o de procedimientos que puede llegar a causar que un atacante pueda abrir una puerta hacia el interior de la red o de los sistemas.

También es la predisposición de un sistema a ser afectado por un agente perturbador, como lo son los atacantes y las amenazas. Las vulnerabilidades pueden ser una mala configuración en un servidor, sistemas operativos sin actualizaciones, un puerto abierto innecesariamente, una sesión abierta sin vigilancia, seguridad física

1 Un activo es algo de valor para la organización.

débil que permita que cualquier persona entre al sitio donde están los servidores, entre otras.

A las vulnerabilidades se les conoce comúnmente como hoyo de seguridad. Las vulnerabilidades son el recurso que utilizan los atacantes para poder penetrar dentro de un sistema de cómputo.

2.1.3 Riesgo

El riesgo es una conjugación de las amenazas, las vulnerabilidades y los ataques. Es la probabilidad de que una vulnerabilidad sea explotada de acuerdo con su nivel de exposición y con el peligro involucrado. El peligro es la probabilidad de que se presente una amenaza. El grado de exposición es el número de activos que pueden llegar a ser dañados por el ataque. Para poder contabilizar el riesgo podríamos cuantificar con unidades numéricas cada uno de los puntos (peligro, vulnerabilidad y exposición) y obtener una fórmula para el nivel de riesgo de un sistema de cómputo:

*riesgo = grado de exposición * vulnerabilidad * peligro.*

Esta fórmula es simple y nos ayuda a contabilizar el grado de riesgo al que se encuentra expuesto un sistema de cómputo.

2.2 Administración de la seguridad

Conocer los procedimientos o metodologías que ocupan las personas que intenta realizar un ataque contra una estructura de cómputo es un recurso de defensa muy importante debido a que podemos prevenir patrones que se han repetido en ataques anteriores. La administración de la seguridad ayuda a comprender la forma en la que podemos defendernos y planear estrategias que mitiguen los riesgos y las vulnerabilidades. Así se puede tener un riesgo menor al exponer sistemas a redes públicas.

2.2.1 Atacantes y motivaciones

Los criminales que cometen los delitos informáticos son conducidos por los mismos motivos que los criminales convencionales de los cuales estamos más acostumbrados a escuchar, para ambos

delitos los motivos son una pequeña lista de categorías: financieros, políticos y personales (psicológicos).

La clasificación de los atacantes es muy variada y puede realizarse partiendo de diferentes factores; uno de ellos muy conocido es por el color de su sombrero o por el bien o mal que le hacen a la comunidad de la seguridad en cómputo. La clasificación, dependiendo el color del sombrero de los atacantes, es blanco, gris y negro.

- Sombrero blanco (*White Hat*): Es un investigador encargado de buscar nuevas vulnerabilidades en los sistemas, desarrolla la prueba de concepto (*proof of concept*) y avisa al fabricante del producto sobre la vulnerabilidad descubierta y su explotación antes de publicarlo.
- Sombrero gris (*Gray Hat*): Es un investigador que encuentra nuevas vulnerabilidades en los sistemas, desarrolla la prueba de concepto y lo publica sin avisarle al fabricante del producto.
- Sombrero negro (*Black Hat*): Son investigadores que explotan nuevas vulnerabilidades en los sistemas pero jamás dan aviso ni al fabricante ni tampoco publican sus investigaciones, sólo las usan para beneficio personal.

Existen otras clasificaciones de los atacantes dependiendo del nivel de experiencia del atacante. El nivel de motivos puede ser muy variado, como también los ataques y las vulnerabilidades, aquí exponemos los más convencionales.

2.2.2 Tipos de ataques

Los ataques se clasifican en pasivos y activos, dependiendo de cómo se lleven a cabo y del impacto del daño. Un ataque pasivo es aquel en el que el atacante no tiene contacto directo con el objetivo del ataque, simplemente se dedica a observar y recopilar información que le sea útil, ejemplos de ataques pasivos es recopilar información del objetivo del ataque en bases de datos públicas, consultas al DNS y espiar la red con sniffers.

Los ataques activos son aquellos en los que el atacante se hace presente, si no tenemos en cuenta la importancia de la seguridad puede ser que no nos demos cuenta de que el atacante está presente aunque esté realizando un ataque activo; en un ataque

activo la pérdida de datos es latente pues estos ataques buscan destruir, corromper, sustituir, interrumpir, interceptar, modificar o sacar provecho de forma más generosa que su contraparte pasiva.

2.2.3 Defensas

Para mitigar los riesgos y tener una mayor administración de éstos se pueden ocupar muchos de los recursos de cómputo que estén a nuestro alcance como es el uso de la criptografía. Si en nuestro entorno de red ocupamos esquemas de cifrado y firmas digitales podemos devaluar los activos hacia el atacante. A un atacante no le sirven datos cifrados sin conocer las claves y llaves para descifrarlos, se vuelven datos basura para él pues no puede sacar información valiosa de ellos.

Otra estrategia de defensa contra ataques es mitigar las vulnerabilidades aplicando parches de seguridad a los sistemas operativos y aplicaciones, así el rango de vulnerabilidades se reduce demasiado aunque siempre aparecerán nuevas vulnerabilidades. Otra estrategia para mitigar las vulnerabilidades es el uso de software de prevención como son las herramientas de seguridad. En la presente tesis se propone el uso de software de prevención como son los Firewalls y los sistemas detectores de intrusos.

2.2.4 Modelos y políticas de seguridad

Un modelo de seguridad es la base con la cual se puede construir la normatividad (políticas de seguridad) que va a dar la pauta para fortalecer un entorno. Un modelo de seguridad es la lista de recomendaciones en las que debemos fijarnos para poder construir una seguridad fuerte en la que esté contemplada la solución a la mayoría de las vulnerabilidades. El modelo de seguridad se convierte entonces en la filosofía que va a seguir la empresa o institución para obtener y mantener un estado de seguridad. El modelo de seguridad nos va a llevar de la mano para escribir políticas de seguridad.

Las políticas de seguridad son una serie de reglas que todos los directores, administrativos, jefes, empleados y usuarios deben de cumplir para mantener el estado de seguridad. Estas reglas deben

ser difundidas y publicadas de manera que sean del conocimiento de todos los usuarios para su correcto cumplimiento. Las políticas de seguridad son reglas que se deben de cumplir estrictamente, por lo que no son opcionales.

Para que una política de seguridad sea efectiva debe de ser conocida y apoyada. Si está publicada pero no es debidamente apoyada es igual de inefectiva como si no lo estuviera o no existiera. Los mandos superiores son los primeros que deben apoyar y apegarse a las políticas, de lo contrario se genera una vulnerabilidad y el modelo de seguridad no será eficaz.

En el trabajo de tesis se encontraron muy útiles los siguientes modelos de seguridad pues reúnen las características básicas para un trabajo de seguridad modesto pero muy robusto hasta un complejo modelo como lo es la serie 27000.

El modelo de seguridad RFC 2196 también conocido como libro de seguridad (*Security Hand Book*) es muy flexible. Útil en empresas pequeñas y medianas, recomendado para la mayoría de las arquitecturas de red. Es muy amigable en el caso de la escritura de políticas pues muestra varios ejemplos de ellas usadas frecuentemente. No es muy recomendable su uso para hardware.

Criterios comunes (*Common Criteria for Information Technology Security Evaluation*, ISO 15048) es un modelo muy completo que define los requerimientos básicos y muy avanzados para garantizar una buena seguridad en las Tecnologías de la Información. Criterios Comunes establece un conjunto de componentes como una manera estándar de expresar los requerimientos funcionales de la seguridad. Es una guía cuando se quiere garantizar ciertos niveles de seguridad.

El ISO 27001 contiene los requisitos para implementar un sistema de gestión de seguridad de la información dentro de una organización. Tiene su origen en el estándar BS 7799-2:2002

El ISO 27002 (el nombre anterior de este ISO es ISO/IEC 17799) es un estándar para la gestión de la seguridad de la información. Presenta guías, recomendaciones y criterios básicos para establecer la seguridad informática. Estas van desde los conceptos de seguridad física hasta los de seguridad lógica. El 27002 proviene de la norma elaborada por la BSI (*British Standards Institution*)

aunque después fue adoptada por la ISO (*International Standards Organization*) y por la IEC (*International Electronic Commision*) como el estándar más completo e incluyente. Su uso es recomendable en organizaciones grandes que buscan garantizar la seguridad de sus sistemas.

Existen además algunos otros estándares como ITIL que en sus apartados de seguridad hacen referencia al ISO/IEC 27002.

En la tabla 2.1 muestra la orientación o enfoque que tienen los estándares descritos.

Nombre estándar	Enfoque
Security Hand Book RFC 2196	– Estándar orientado a buenas prácticas para aplicarse dentro de las organizaciones.
Criterios comunes ISO 15048	– Estándar orientado a controles en el desarrollo e implementación de productos de TI
ISO/IEC 27001	– Estándar orientado a implementación de procesos en las organizaciones
ISO/IEC 27002	– Estándar orientado a buenas prácticas que describe controles para implementar seguridad informática

Tabla 2.1 Enfoque de los estándares

2.2.5 Servicios de la seguridad (CID)

La seguridad en cómputo está estructurada en tres conceptos fundamentales que se pueden y se deben aplicar a cualquier tema relacionado con ella. Los tres conceptos son Confidencialidad, Integridad y Disponibilidad, comúnmente se les identifica con el acrónimo CID (CIA en inglés), también se les conoce como vectores de la seguridad en cómputo.

- Confidencialidad: Un sistema posee la propiedad de confidencialidad si la información manipulada por éste no es

disponible ni puesta en descubierto para usuarios, entidades o procesos no autorizados.

- Integridad: Un sistema posee la propiedad de integridad si los datos manipulados por éste no son alterados o destruidos por usuarios, entidades o procesos no autorizados.

- Disponibilidad: Un sistema posee la propiedad de disponibilidad si, la información es accesible (está disponible) en el momento en que así lo deseen los usuarios, entidades o procesos autorizados.

Cualquier herramienta de software, modelo de seguridad, esquema de red, debe de cumplir cabalmente con estos conceptos para poder brindar seguridad, si no está basado directamente en ellos no se puede garantizar su efectividad.

2.3 Control de acceso

El término control de acceso es mencionado por algunos autores como un vector de seguridad extra. Se refiere a los privilegios que tienen los usuarios y equipos autorizados cuando interactúan con la red y los sistemas. El control de acceso se divide en autenticación y autorización.

2.3.1 Autenticación

La autenticación es él área de la seguridad en cómputo donde se definen los métodos de validación de usuarios.

El servicio de autenticación es posible dividirlo en tres fases: registro, identificación y autenticación. El registro se refiera a generar una lista de usuarios válidos para el sistema, la identificación consiste en dar a cada usuario un nombre único e irrepetible y por último la autenticación es cualquier método para garantizar la identidad del individuo, este puede darse de tres formas distintas, por lo que se sabe[2], lo que se tiene o por lo que se es.

2 Por lo que se sabe hacer, ya sea una habilidad única como firmar en una hoja de papel, o por lo que se sabe de algún conocimiento, por ejemplo una contraseña.

El más común es por lo que se sabe, en esta categoría entran las contraseñas. Actualmente las organizaciones que buscan tener un control de usuarios más eficiente y ocupan métodos no tan comunes como las contraseñas; utilizan autenticación por métodos biométricos, lo que se es; y por posesión de credenciales magnéticas de identificación o algún dispositivo (*netkey*), por lo que se tiene.

2.3.2 Autorización

La autorización se refiere a las acciones que tienen permitidas los usuarios autorizados por un medio de autenticación, en particular se refiere a restricciones sobre información confidencial. Generalmente siempre existen diferentes niveles de usuarios, tanto en el sistema operativo como en las aplicaciones. Los administradores la mayoría de las veces son los usuarios con mayor privilegio sobre datos confidenciales y los usuarios convencionales los que menor privilegio tienen.

Dos de las herramientas más utilizadas para garantizar autorización son los Firewalls y los sistemas detectores de intrusos.

2.3.3 Contabilidad de accesos

La contabilidad de accesos se refiere al registro de los accesos de los usuarios o las acciones que éstos ejecutan, aun cuando tengan autorización para realizar determinadas tareas o privilegios para ver determinada información, la contabilidad de acceso permite saber qué acciones ejecuta el usuario y cómo las ejecuta.

2.4 Proceso general de ataque

Cuando un atacante ha tomado la decisión de iniciar un ataque buscando introducirse a una red para comprometer la seguridad de un objetivo debe de seguir una metodología que lo lleve a lograr una intrusión, esta metodología puede variar dependiendo de si el atacante es interno o externo, también puede variar en factor de los conocimientos del agresor. Nosotros describimos una metodología general donde se engloban los pasos necesarios para consumar una intrusión con la cual se logran los objetivos del ataque satisfaciendo los motivos del agresor.

Los pasos que nosotros describimos como necesarios para un ataque son los siguientes: selección del objetivo, reconocimiento, enumeración, explotación, ganando acceso, consolidación, manteniendo acceso, evitar la detección, realizar una meta.

La selección del objetivo depende mucho de la motivación del ataque, la motivación da el parámetro de decisión para poder determinar un objetivo adecuado. Si el atacante busca desprestigiar a su competencia comercial buscará empresas líderes en el ramo en el que se desempeña, si los motivos son políticos, el atacante buscará una institución por la cual se sienta amenazado, así podríamos enfocar muchos objetivos que satisfagan a cierta motivación.

En el reconocimiento el atacante se da a la tarea de recolectar información útil utilizando ataques pasivos. En esta parte del ataque el agresor ratifica la buena selección del objetivo, si cumple con las expectativas sigue buscando información en sitios públicos como las bases de datos de ARIN[3], donde se consulta a las bases de datos whois, consultas en el sitio netcraft[4] y consultas directas a los servidores DNS. Con esta información el atacante busca conocer mejor el perfil del objetivo.

En la enumeración el atacante recopila información de las direcciones IP de los equipos objetivo, esta información de red le sirve para conocer la infraestructura con la que cuenta el objetivo y buscar puntos débiles; en esta fase el atacante arma un esquema donde tal vez estén involucrados Firewalls, detectores de intrusos, ruteadores y otros elementos de red. El atacante deberá estar preparado para brincar todas estas restricciones. En esta fase también el agresor hace un análisis de vulnerabilidades que puede ser pasivo o activo, en la parte activa puede utilizar herramientas automatizadas como los escáneres de puertos.

La explotación es la parte fundamental del ataque, si ésta tiene éxito, el atacante abrirá una brecha de seguridad tomando control sobre el objetivo, si la brecha es crítica, el control sobre el objetivo será total. Hay ataques muy específicos en los que el control total no necesariamente sería tener acceso al objetivo, por ejemplo, en

1 American Registry for Internet Numbers se utilizan para identificar el dueño de una dirección IP o un dominio de Internet
2 Sitio que provee información sobre servidores Web.

un ataque de SQL Injection la consumación total del ataque sería tener una copia fiel de la base de datos del objetivo o insertar datos no confiables en ella para ganar recursos como dinero o servicios. La fase de explotación es la parte en donde el ataque se vuelve realmente activo, durante la enumeración se pueden usar técnicas pasivas para detectar vulnerabilidades.

Ganando acceso se refiere a escalar privilegios dentro del objetivo hasta poder llegar a privilegios de administración sobre él, los privilegios de administración son siempre los más deseados por los atacantes pues con ellos pueden ejecutar cualquier acción sobre el equipo objetivo.

Consolidación es la fase en la que el atacante se cerciora de que sólo él tendrá la posibilidad de comprometer la seguridad de ese equipo, para lograr esta fase el atacante deberá componer la falla por la cual tuvo acceso, con esto el atacante logra que él será el único que podrá vulnerar ese equipo.

Manteniendo acceso es la fase en la que el atacante asegura su libre entrada al equipo para poder regresar cuando a él le convenga, generalmente a esta parte se le conoce como instalación de puertas traseras.

Evitar la detección es una fase en la que el atacante busca que los dueños del equipo objetivo no visualicen su presencia. Para lograr esto comúnmente los atacantes usan herramientas para limpiar las bitácoras de acceso y otras que les sirven para esconder los procesos que ejecuten, por lo general ocupan versiones troyanas de las versiones originales.

Realizar una meta es cuando el atacante se ha visto satisfecho en sus motivaciones y continúa haciendo un abuso del equipo objetivo, este abuso puede ser tan prolongado como el atacante desee o hasta que el dueño del equipo objetivo realice una auditoría de seguridad donde deberá encontrarse que el equipo está comprometido.

Capítulo

3. Firewalls

3.1 Definición

Un Firewall es un dispositivo que controla el acceso entre distintas redes. Esta tarea la realiza examinando el tipo de tráfico que circula entre dichas redes y bloqueando aquel que considera peligroso, inapropiado o ambos. Se utiliza generalmente para proteger una red privada y los servidores que se encuentran dentro de ésta. El Firewall por lo regular se ubica entre la red privada y la red pública, aunque podrían darse muchas variantes.

El Firewall impone restricciones a los paquetes de datos que entran y salen de una red privada. Todo el tráfico que circula de dentro hacia fuera y viceversa es verificado, pero sólo el tráfico permitido podrá pasar a través de él.

Para entender qué significa esto, necesitamos comprender la estructura de un Firewall. Consiste en dos interfaces (aunque podría hacerse sólo con una), la pública y la privada. La interfaz pública, generalmente Internet, es el lado del Firewall al cual todos tienen acceso. La interfase privada, es el lado de la red que se quiere proteger, puede haber muchas interfaces privadas, de acuerdo con el número de segmentos de red que se desee aislar. El Firewall usa un conjunto de reglas, aplicadas a cada interfaz, para determinar qué tipo de tráfico puede pasar del lado público al privado. Si se utiliza una política restrictiva todo el tráfico que no esté explícitamente determinado por las reglas será bloqueado, mientras que si se emplea una política permisiva sólo se bloqueará el tráfico especificado en las reglas, en la sección 3.4 se explican más detalles.

Un Firewall puede brindar muchos servicios, los cuales representan ventajas y desventajas, por ejemplo, debido a los altos costos de los dispositivos y al poco presupuesto con que cuentan muchas compañías, a veces se colocan servicios adicionales sobre el Firewall como por ejemplo: servidor de nombres de dominio (DNS), servidor de autenticación, servidor de redes privadas virtuales (VPN), etc. Esto pude representar riesgos para la seguridad del Firewall, ya que mientras más servicios tengamos funcionando, daremos más oportunidades de que vulneren nuestro sistema a través de cualquiera de estos servicios. Una vez vulnerado el Firewall toda la red está desprotegida.

La fuerza de un Firewall se halla en la habilidad para filtrar paquetes, se basa en un conjunto de reglas, las cuales se determinan de acuerdo con las políticas de seguridad y las necesidades de la red que requerimos proteger. Un conjunto incompleto de reglas o mal definido puede provocar en el mejor de los casos inestabilidad en nuestra red, ya que podríamos estar bloqueando algunos servicios importantes, pero en el peor de los casos estaríamos abriendo una brecha, que podría utilizar un atacante atentando contra la seguridad.

Uno de los objetivos de un Firewall es asegurar la confidencialidad de la información que se encuentra y se transmite del lado de la red privada, permitiendo el acceso desde la red pública sólo a los usuarios autorizados. Como se menciona en el capítulo dos la confidencialidad es un servicio de seguridad que permite sólo a las persona autorizadas ver la información y que tengan acceso a la misma.

3.2 Clases de Firewall

Existen muchos productos que hacen funciones de Firewall, la mayoría de ellos son desarrollados por empresas que tienen mucha presencia en el campo de la seguridad en cómputo, regularmente son productos muy costosos. También hay muchos que son basados y distribuidos bajo licencias de software libre y open source, lo que permite reducir los gastos ampliamente.

Algunos trabajan específicamente en una capa del modelo TCP/IP mientras que otros pueden trabajar en varias capas al mismo tiempo. Hay tres clases de Firewall, filtrado de paquetes, filtrado de paquetes con estado y proxys de capa de aplicación, se pueden encontrar productos libres y comerciales de las tres clases de Firewalls.

3.2.1 Filtrado de paquetes

Este tipo de Firewall determina qué paquetes pueden pasar a través de él, analizando dos campos de la cabecera IP, los cuales son: dirección IP origen y dirección IP destino y dos campos de la cabecera TCP o UDP que son: puerto origen, y puerto destino. La información de estos cuatro campos puede combinarse para

determinar el conjunto de reglas qué determinan que tipo de tráfico puede pasar o no. Figura 3.1

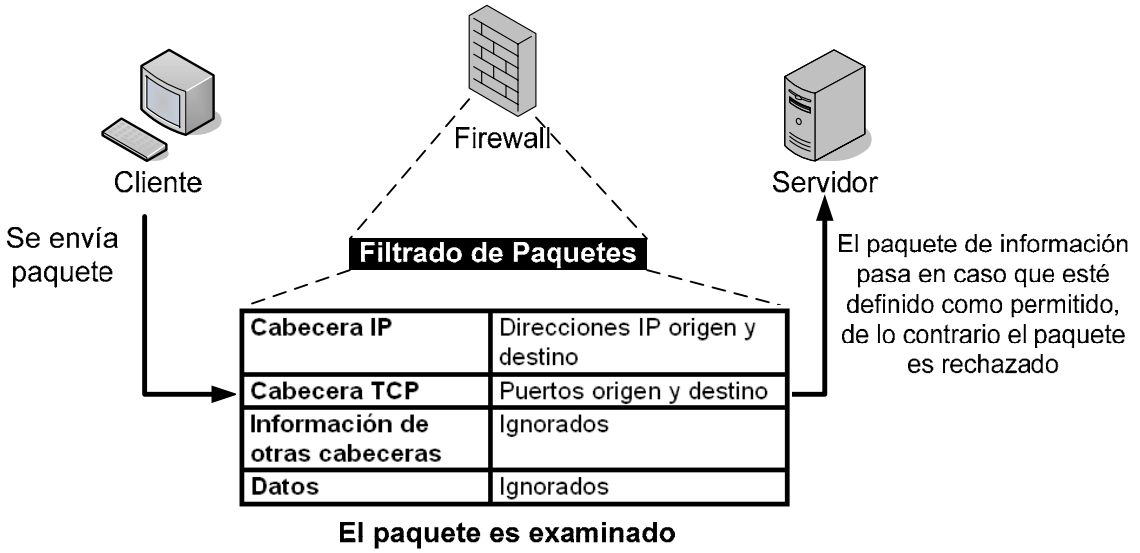

Figura 3.1 Filtrado de paquetes.

Sin embargo este tipo de Firewall no es cien por ciento confiable ya que es vulnerable a los ataques de suplantación de dirección IP (*IP spoofing*). Este ataque busca suplantar o modificar la dirección IP origen en el campo de la cabecera IP. Este tipo de paquetes con direcciones suplantadas generalmente no son bloqueados por este tipo de Firewall. También es vulnerable a los paquetes que cuenten con la cabecera IP legítima, pero que contengan datos peligrosos, como ataques de desbordamiento de búfer (*buffer overflow*).

3.2.2 Filtrado de paquetes con estado

El término filtrado de paquetes con estado hace referencia al seguimiento de las conexiones TCP, empezando por el protocolo de intercambio de señales en tres pasos, que se realiza al comienzo de cada sesión TCP y finaliza con el último paquete de la sesión con una bandera de finalización (FIN) o con una de reset (RST). Después que el Firewall haya verificado como válida una transacción dada, basado en las direcciones IP y en los puertos destino y origen, monitorea el estado del protocolo TCP en la conexión iniciada.

Si se completa en un tiempo razonable, las cabeceras TCP del resto de los paquetes subsecuentes para esa transacción son verificadas contra la tabla temporal estable del Firewall permitiendo el paso de dichos paquetes hasta que dicha sesión es finalizada.

Específicamente, cada dirección IP origen, puerto origen, dirección IP destino, puerto destino y número de secuencia TCP son controlados. En la figura 3.2 se muestra cómo trabaja este tipo de Firewall.

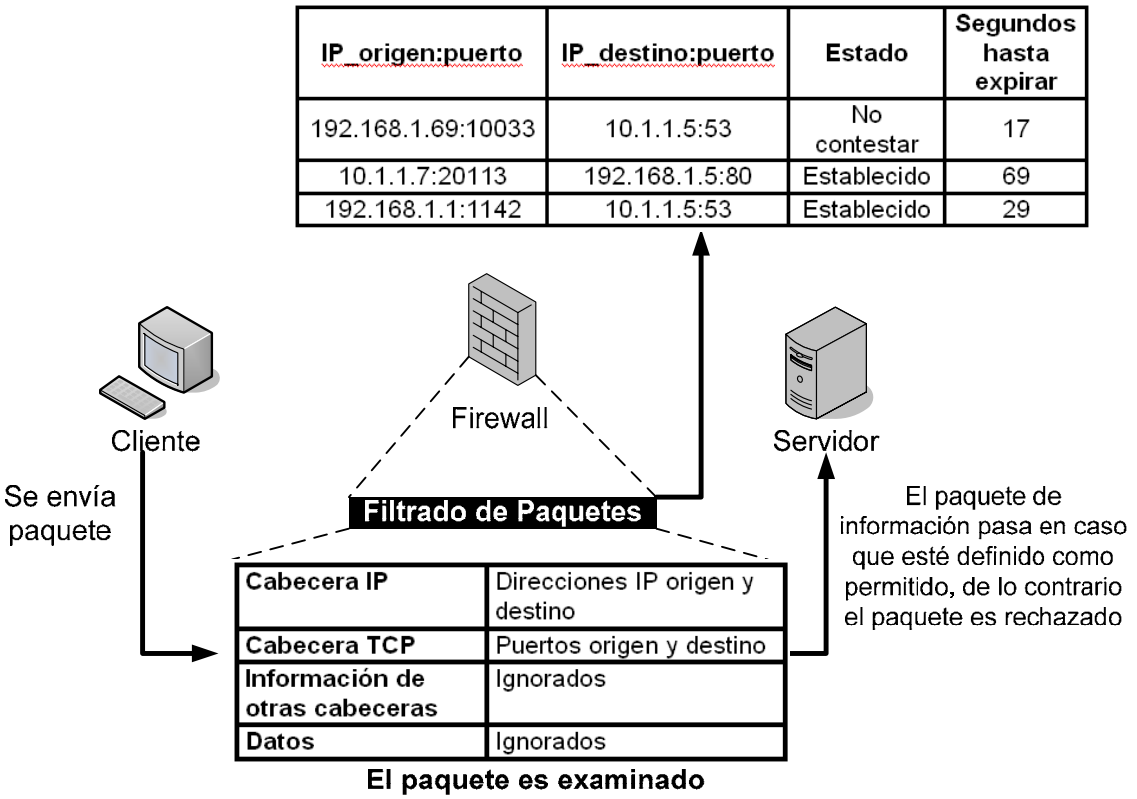

Figura 3.2 Filtrado de paquetes con estado

3.2.3 Proxy de capa de aplicación

Como se muestra en la Figura 3.3 a diferencia de los Firewalls anteriores, un proxy actúa como un intermediario en todas las transacciones que lo atraviesan.

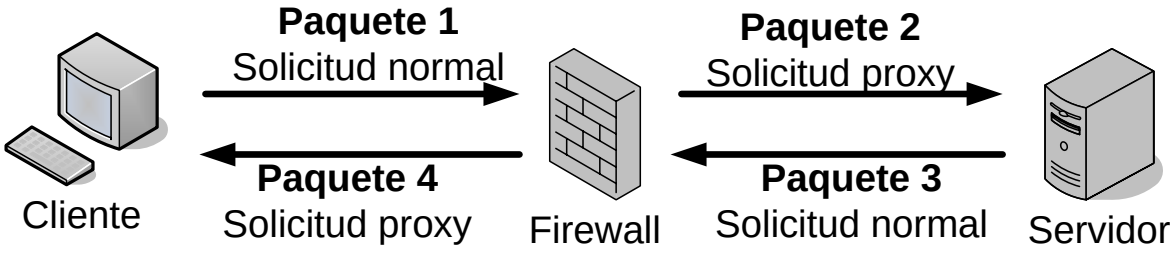

Figura 3.3 Proxy de aplicación

Se les denominan proxy para capa de aplicación porque, al contrario de otros tipos de proxy que realzan el rendimiento pero no necesariamente los aspectos de seguridad, los Firewalls tipo proxy

de capa de aplicación normalmente tienen gran inteligencia sobre los servicios específicos de la aplicación que verifican.

Un proxy para capa de aplicación no sólo distingue entre direcciones IP y puertos permitidos, también distingue los comportamientos permitidos y rechazables de la aplicación, en otras palabras los clientes interactúan directamente con el Firewall y es éste el que interactúa con el servidor (HTTP, SMTP, FTP, etc.).

Algunas de las desventajas de este tipo de Firewall son el rendimiento y la flexibilidad. En el momento en que un Firewall proxy participa activamente en lugar de únicamente monitorear, emplea mucho más recursos de cómputo para cada transacción que un filtro de paquetes con estado. Un Firewall proxy para capa de aplicación sólo puede proporcionar protección a una reducida variedad de servicios, generalmente los más importantes.

3.3 Arquitecturas de Firewall

A continuación comentaremos brevemente cuáles son algunas de las arquitecturas de red para colocar un Firewall más comunes, utilizadas en redes pequeñas y grandes. Regularmente existen varias formas de implementar estas arquitecturas pero cada una de ellas cuenta con sus propios riesgos de seguridad.

3.3.1 Multi-homed host

Multihomed-host se refiere a una computadora casera que posee más de una interfaz de red. Cada una de las interfaces extra es utilizada para conectar física y lógicamente más de un segmento de red diferente. Dual-hommed host es una computadora con sólo dos interfaces de red y es la versión más utilizada de multi-homed host.

En dual-homed host cada interfaz de red está conectada a redes diferentes. Comúnmente una de las tarjetas de red está conectada a una red externa e insegura, mientras que la segunda tarjeta está conectada a la red interna que es la que se busca proteger.
Un problema que tiene esta sencilla arquitectura es que el direccionamiento (ruteo) de paquetes debe ser desactivado en el Firewall por problemas de seguridad, los paquetes no deben de

pasar directamente de una red a otra, el Firewall siempre debe de ser intermediario entre ambas redes.

3.3.2 Red protegida (Screened subnet)

Una de las arquitecturas más comunes que tiende a verse hoy en día es la que se muestra en la Figura 3.4. En el diagrama, se puede ver en primer lugar un ruteador que es por lo regular el punto donde toca la acometida de Internet, este ruteador implementa cierto nivel de filtrado de paquetes.

Detrás del primer ruteador se ubican los servidores públicos, la red donde se ubican estos servidores se conoce como perímetro exterior. Después se utiliza un segundo ruteador el cual sirve para aislar la red interna del perímetro exterior.

Los servidores públicos como el Web, SMTP y FTP son los que tienen que estar en contacto directo con Internet, lo cual tiene un alto riesgo de que alguno de estos servidores pueda ser comprometido por un atacante, lo que implica comprometer prácticamente cualquier punto de la red detrás del primer ruteador.

Es aquí donde adquiere gran valor el segundo ruteador ya que si alguno de los servidores públicos es comprometido, el atacante tendría que penetrar a través del segundo ruteador para llegar a la red interna, haciendo la tarea más complicada

Una ventaja de esta arquitectura es que se puede aprovechar mejor los recursos de los ruteadores en la transmisión de información entre el perímetro exterior e Internet y la transmisión de información entre la red interna y el perímetro exterior.

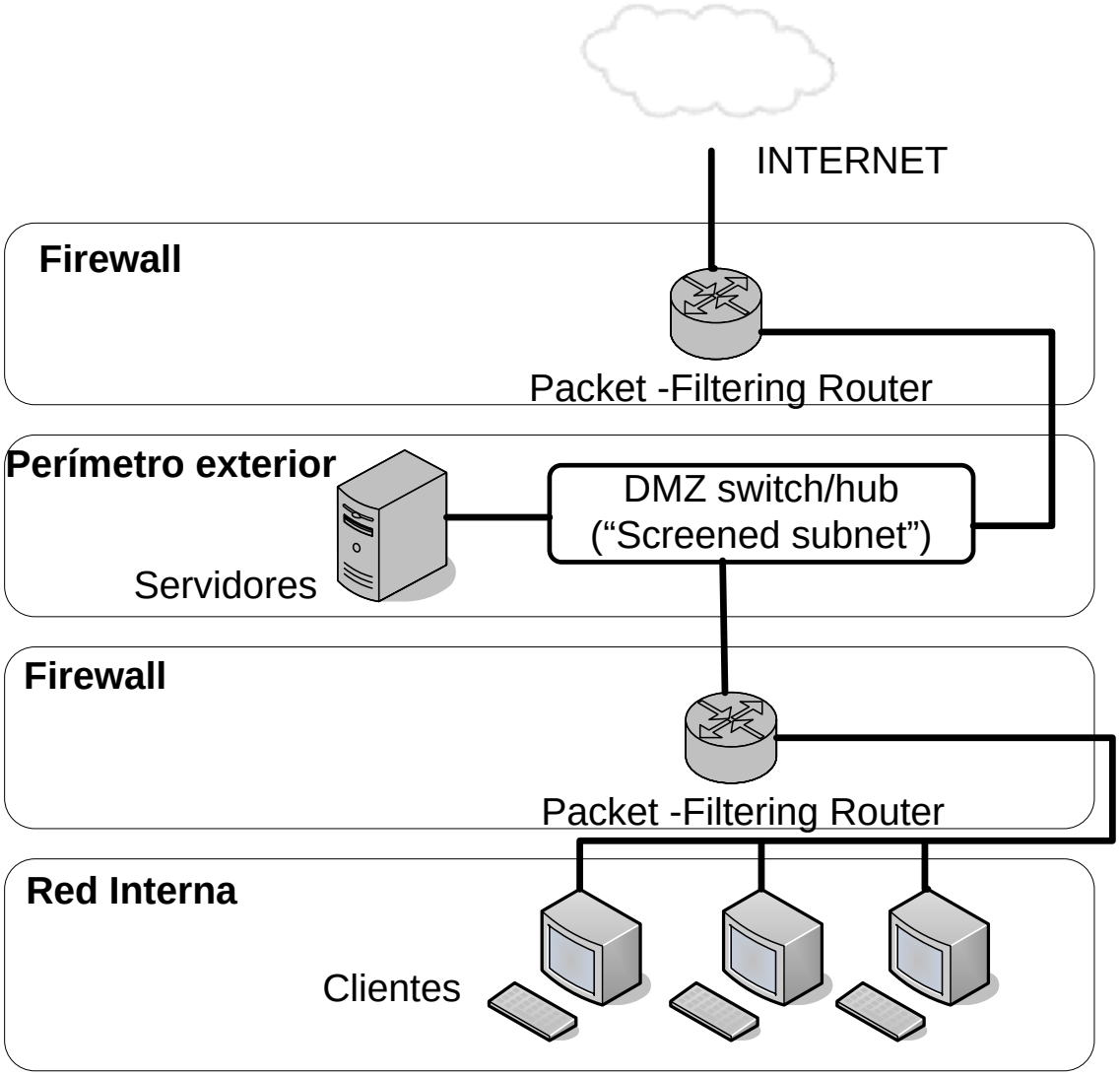

Figura 3.4 Screened subnet

3.3.3 Zona desmilitarizada (*Screened subnet DMZ*)

Una zona desmilitarizada es cualquier parte de la red alcanzable por el público, pero aislada de la red interna, en la Figura 3.5 se muestra un ejemplo de este tipo de arquitectura.

Como puede apreciarse este tipo de Firewall debe tener tres interfaces de red, una que recibe la acometida del ruteador, la segunda que está conectada a la red interna y la tercera la cual está conectada a la zona desmilitarizada. El Firewall debe utilizar distintas reglas para evaluar el tráfico que pasa a través de él en las siguientes direcciones:

- De Internet a la zona desmilitarizada.

- De la zona desmilitarizada a Internet.
- De Internet a la red interna.
- De la red interna a Internet.
- De la zona desmilitarizada a la red interna.
- De la red interna a la zona desmilitarizada.

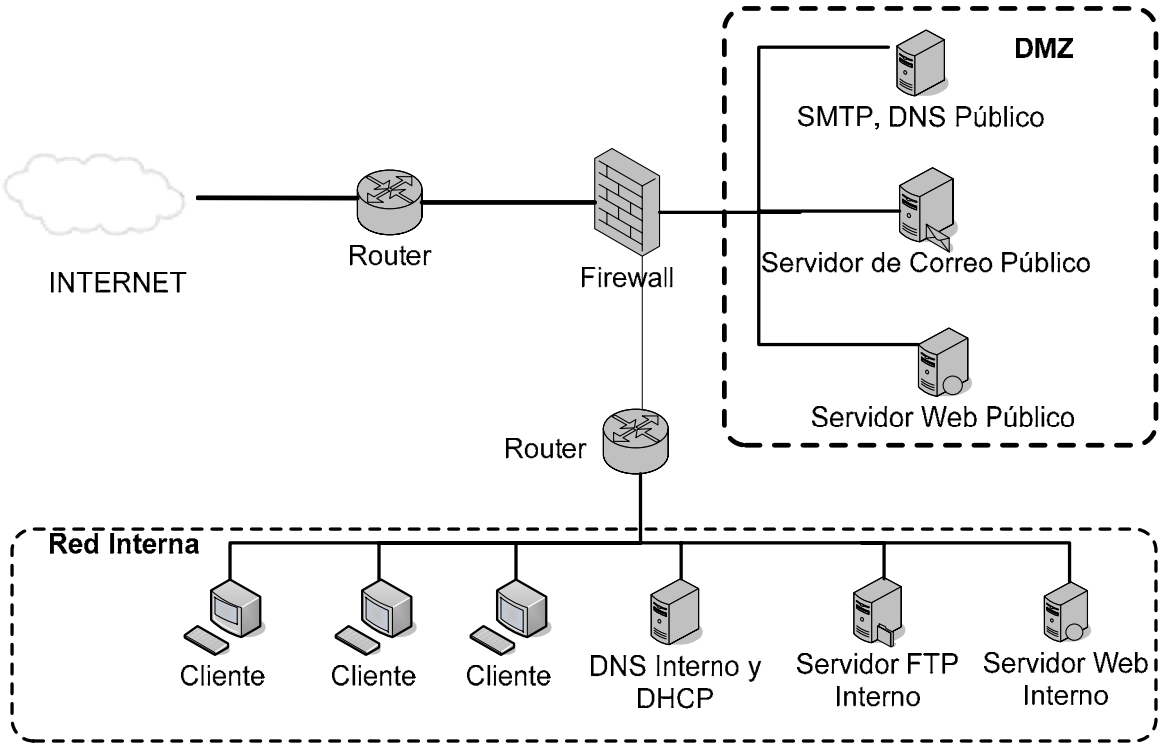

Figura 3.5 Zona Desmilitarizada (DMZ)

La ventaja de esta arquitectura es que divide el tráfico que va dirigido hacia los servidores públicos del que va hacia la red interna, permitiendo un control estricto en el tráfico dirigido desde o hacia la red que queremos proteger, es decir la red interna.

Sin embargo la desventaja de este tipo de arquitecturas es que si se requiere la transmisión de grandes volúmenes de información entre dos de las tres redes, se podría ver afectado el rendimiento en la transmisión de la tercera red debido a que todo el tráfico es distribuido por un solo dispositivo, por ejemplo si se transmite mucha información desde Internet hacia la DMZ, podría disminuir el rendimiento del dispositivo en la transmisión de información desde la red interna hacia Internet. Para evitar el problema de rendimiento y contar con la ventaja de dividir el tráfico en distintas redes para lograr mayor seguridad al filtrar el tráfico se puede pensar en un dimensionamiento más eficiente de los equipos de comunicaciones, para esto nos conviene adquirir equipos con mayor throughput que soporten la tasa de transferencia actual de transacciones más un

porcentaje extra para evitar sorpresas inesperadas de pérdidas de paquetes.

3.4 Políticas

La consideración más efectiva en el uso de Firewalls es el diseño de políticas de seguridad. Una política de seguridad para un Firewall es una declaración de cómo debe funcionar. Una política es aquella que nos permite definir las reglas por las cuales el tráfico que entra y sale de nuestra red debe ser aceptado o rechazado. Conforme las necesidades de seguridad y de servicios en nuestra red vayan cambiando, las políticas de seguridad lo harán también. Una efectiva lista de políticas de seguridad debe de brindar la base para la implementación y configuración de un Firewall.

Generalmente las políticas de un Firewall se pueden dividir en dos grandes apartados, las permisivas y las restrictivas. Las permisivas son reglas con una seguridad débil, pero fáciles de implementar. Las restrictivas dan un mayor fortalecimiento a la red pero son más complicadas de implementar. Una política permisiva se puede enunciar de la siguiente forma, todo lo que no está prohibido está permitido. En cambio, una política restrictiva tiene mayor control, y se puede enunciar como sigue, todo lo que no está permitido está prohibido.

Las políticas de un Firewall también nos ayudan a identificar los riesgos a los que están expuestos tanto la red interna como los servidores. Por lo que nos permite crear planes para mitigar los riesgos y saber cuáles son los activos más importantes que debemos proteger.

No debemos perder de vista que la misión y los objetivos de la organización definen los objetivos de seguridad, con base en éstos deben definirse las políticas de acceso de un Firewall. Por ejemplo, si para cumplir con los objetivos de la organización es necesario transmitir información a través del protocolo FTP las políticas del Firewall deben permitir el paso de información a través de este protocolo.

3.5 OpenBSD y Packet Filter

Para este trabajo de tesis se escogió el Firewall Packet Filter (PF) del sistema operativo OpenBSD utilizando la arquitectura Multi-Homed host. Se hizó esta elección debido a que el Sistema Operativo OpenBSD brinda un Firewall muy avanzado, además de que se puede usar su código fuente por ser un sistema libre. Su libertad es tan grande que permite modificar y reutilizar la mayoría de su código fuente tanto para fines particulares y comerciales.

OpenBSD es un sistema operativo que puede ser utilizado tanto para usuarios finales como para servidores, aunque su uso está mucho más difundido en el mundo de los servidores y en particular para aquellos que se requieran implementar con seguridad por defecto. OpenBSD es reconocido por muchos profesionales en seguridad como uno de los sistemas operativos más seguros del mundo por contar con algoritmos criptográficos y por integrar las últimas tecnologías en seguridad para el uso de Firewalls.

PF es parte del kernel del sistema operativo OpenBSD desde su versión 3.0 y es el programa utilizado para filtrar el tráfico TCP/IP, una de sus características principales es que cuenta con un filtrado de paquetes con estado.

Para habilitar el filtrado de paquetes desde que inicia la computadora basta con editar y agregar la siguiente directiva al archivo de configuración /etc/rc.local, pf=YES.

En la tabla 3.1 se muestran algunos de los comandos más utilizados en la administración de PF, estos pueden ser introducidos directamente en una terminal de intérprete de comandos del sistema operativo, necesitan ser estrictamente ejecutados bajo los permisos del usuario administrador (root) para tener efecto.

Comando	Descripción
pfctl –e	Habilita PF
pfctl –d	Deshabilita PF
pfctl –f archivo	Carga las reglas descritas en archivo
pfctl –nf archivo	Revisa la sintaxis de las reglas descritas en archivo pero no las carga.
pfctl –sr	Muestra las reglas de filtrado activas.

Tabla 3.1 Comandos de PF

En el archivo de configuración, que generalmente se encuentra en /etc/pf.conf se pueden establecer las siguientes directivas que son las más básicas para crear un conjunto de reglas:

- Macros: Son variables definidas por el usuario y en ellas podemos alojar los nombres de los hosts, las direcciones IP o los nombres de las interfaces de red. Son muy útiles para tener mayor flexibilidad.
- Reglas de Filtrado: Deciden cuáles son los paquetes que pasan o son bloqueados por las interfaces de red.

Un ejemplo sencillo de reglas para una política restrictiva se puede llevar a cabo de la siguiente manera:

```
ext_if=”rl0”
int_if=”dc0”

# Política reestrictiva por defecto

block in all
block out all

# Permitir todo en localhost (127.0.0.1)

pass in on lo0 from any to any keep state
pass out on lo0 from any to any keep state

# Permitir el puerto 22, para administración del Firewall

pass in on $ext_if proto tcp from any to 192.168.1.66 port 22 keep state
```

Donde ext_if es un macro para definir la interfaz de red externa, ahí es donde va conectado el cable de Internet, int_if es otra macro para definir la interfaz interna que es la red que queremos proteger. Las líneas que comienzan con el símbolo # son comentarios y el Firewall no los tomará en cuenta. Después se define una política donde se deniega todo el tráfico, tanto el de entrada *in*, como el de salida *out*, esto se logra con la sentencia *block*. Como el filtrado de paquetes se hace a nivel núcleo del sistema operativo hay que

aceptar como válido el tráfico de entrada y de salida que se genere en la interfaz local lo0, para especificar cual es la interfaz donde se genera la regla se utiliza la sentencia *on*. Después se muestra como permitir el tráfico del protocolo SSH en la interfaz externa desde una dirección IP, para lograr esto utilizamos la sentencia *pass*, adicionalmente utilizamos la sentencia *proto* para determinar que tipo de protocolo se utiliza, en este caso es TCP.

Capítulo

4. Sistemas de detección de intrusos (IDS)

4.1 Definición

Antes de hacer una definición formal del término detección de intrusos es vital definir intrusión. Establecer una explicación clara del concepto de intrusión representa fijar los alcances que pueden tener los sistemas de detección y prevención de intrusos. Una intrusión es una secuencia activa de eventos relacionados que tratan deliberadamente de causar daño. La definición se refiere a ambas tentativas, las que se consumaron y las que fracasaron.

En un nivel básico la detección de intrusos en una red representa exactamente el proceso de determinar cuando algunas personas no autorizadas realizan intentos o logran consumar una intrusión.

Mantener a los atacantes fuera o extraerlos de la red una vez que han penetrado es un problema totalmente diferente. Obviamente mantener los intrusos fuera de nuestra red es una tarea que carece de sentido si no se sabe si éstos han penetrado ya y como lo han hecho.

Un IDS observa todo el tiempo las actividades realizadas en la red y es capaz de decidir cuando algo le parece sospechoso.

Los IDS se componen básicamente de tres elementos, sensor, consola y motor central.

- Sensor: Su labor es detectar eventos de seguridad.

- Consola: Se encarga de monitorear los eventos emitidos y de administrar los sensores.

- Motor central: Registra los eventos generados por los sensores en bitácoras y genera las alertas de los eventos de seguridad recibidos.

4.2 Clasificación por arquitectura

Básicamente existen dos tipos de IDS en una clasificación por arquitectura, basados en host y basados en red, la clasificación de arquitectura se refiere al lugar en donde están situados en la red y el nivel de cobertura.

4.2.1 IDS basado en host

Los IDS basados en Host (HIDS, *Host-based Intrusion Detection System*) implican utilizar programas instalados en los sistemas que se requiere sean monitoreados. Dichos programas están diseñados para monitorear, detectar y responder a la actividad de los usuarios o del sistema, así como ante eventuales ataques. Algunas herramientas más robustas ofrecen administración de las políticas de auditoría definidas para el sistema, sirven como fuente para análisis forense y en ciertas ocasiones proporcionan medidas para el control de acceso.

Los HIDS están orientados a combatir amenazas internas, debido a la capacidad que tienen de monitorear y responder a acciones específicas de los usuarios y el acceso a los archivos del sistema. También se encargan de revisar qué aplicaciones del sistema tienen acceso a qué recursos del mismo, así como monitorear que no se modifique el estado interno del sistema como es el sistema de archivos, la cantidad de memoria RAM entre otras cosas.

4.2.2 IDS basado en red

Los IDS basados en red (NIDS, *Network Intrusion Detection System*) se encargan de los datos que fluyen a través de los cables que conectan al host con el exterior. Comúnmente llamados sniffers[1], que es una pieza de software que permite visualizar de alguna manera el tráfico que pasa por una interfaz de red. Es decir, todo lo que en el medio físico de red esté pasando, los NIDS se encargan de los paquetes de información que viaja a través de distintos medios de comunicación y protocolos, regularmente TCP/IP. Una vez capturados los paquetes éstos son analizados de diferentes maneras, algunos NIDS simplemente comparan los paquetes con una base de firmas obtenidas a partir de ataques conocidos mientas que otros se basan en actividades anómalas que pueden indicar un comportamiento malicioso.

En cualquier caso los NIDS deben ser considerados como el primer miembro en el perímetro de defensa dentro de las organizaciones.

[1] “Herramienta para la detección de intrusos” Sánchez Verdejo Rommel.

La principal diferencia entre los IDS basados en host y los basados en red es que éstos últimos se encargan de analizar el tráfico que fluye de un lado a otro entre los distintos equipos, mientras que el HIDS sólo se ocupa de analizar lo que ocurre al interior del host donde se encuentra instalado.

4.3 Clasificación por detección

Por otro lado los IDS también se clasifican por la forma en que trabajan y detectan los ataques, éstos pueden ser basados en firmas o en anomalías.

4.3.1 IDS basados en firmas

Los IDS basado en firmas inspeccionan el contenido de las secciones de cabecera y carga neta del paquete, en busca de correspondencias con ataques conocidos, la inspección de la carga neta se conoce como inspección a fondo del paquete.

Actualmente los desarrolladores de estos sistemas de detección examinan nuevos métodos de análisis con el fin de reducir la latencia causada por almacenar los paquetes para completar la inspección a fondo de paquete.

Un IDS basado en detección por firmas pude ser configurado contra amenazas específicas. Esto permite a los administradores de seguridad determinar rápidamente la naturaleza del ataque con tan solo examinar la regla que ha motivado que se dispare la alerta.

Para detectar los últimos ataques conocidos es necesario contar con la actualización de firmas más reciente. Por lo anterior es importante mantener el conjunto de reglas actualizado constantemente. Por lo tanto se recomienda realizar este tipo de actualizaciones en periodos de tiempo regulares, de acuerdo con las políticas de seguridad que se hayan establecido.

El inconveniente al usar un catálogo de firmas es que el atacante también tiene acceso al conjunto de reglas, lo que le permitiría diseñar el ataque de manera que no provoque ninguna alerta, esto resulta de un alto grado de dificultad ya que los ataques regularmente disparan más de una alerta, sin embargo, el atacante

podría provocar que se disparen muchas alarmas al mismo tiempo para ocultar el ataque real que se está llevando a cabo.

Es recomendable instalar este tipo de IDS en lugares donde se encuentren sistemas de misión crítica o lugares expuestos a redes públicas.

4.3.2 IDS basados en anomalías

Los IDS basados en anomalías tienen por objetivo detectar actividades fuera de lo normal en un sistema o el tráfico de una red. Para esto deben contar con un perfil que les permita determinar cual debería ser la actividad normal del sistema o el tráfico común de la red.

Existen dos métodos para generar el perfil: a partir de estadísticas y a partir de especificaciones. Ambos se pueden aplicar en redes, protocolos o sistemas. Sin embargo, las técnicas usadas en cada método para generar un perfil son absolutamente diferentes[2].

La gran diferencia entre los sistemas basados en anomalías y los basados en firmas es que los primeros pueden detectar ataques aún no documentados mientras que los basados en firmas sólo pueden detectar los ataques que se encuentren en la lista activa de firmas.

Se recomienda utilizar este tipo de IDS en ambientes donde se tenga un conocimiento preciso sobre el comportamiento del tráfico de información a través de la red, ya que de esta manera será fácil detectar y alertar sobre cualquier comportamiento anormal.

En la Tabla 4.1 se muestra una comparativa de ventajas y desventajas de los IDS basado en firmas y anomalías.

IDS basado en firmas	
Ventajas	– Detectan ataques perfectamente identificados – Es una tecnología probada que ha demostrado su eficiencia
Desventajas	– Los atacantes también conocen las firmas y

[2] Para mayor referencia visitar http://www.securityfocus.com/infocus/1600

	pueden diseñar ataques no identificados – Requieren constante actualización de firmas
IDS basado en anomalías	
Ventajas	– Pueden detectar ataques no documentados de acuerdo al comportamiento de la red – No requiere actualización de firmas
Desventajas	– Es una tecnología poco probada – Puede presentar errores y tener falsos positivos y falsos negativos (ver 4.4)

Tabla 4.1 Ventajas y desventajas de los IDS

4.4 Falsas Alarmas

Existen dos tipos de falsas alarmas relacionadas con la detección de intrusos, los falsos positivos y los falsos negativos. Un falso positivo es cuando el IDS emite una alarma de intrusión cuando el evento no fue legítimo. Un falso negativo es cuando el IDS no emite una alarma de intrusión cuando el evento realmente sí lo fue.

En ambos sistemas de detección, por anomalías y por firmas, se presentan tanto falsos positivos como falsos negativos. Aunque la tendencia del desarrollo de sistemas de detección dice que los falsos negativos se minimizarán casi en su totalidad cuando los sistemas de detección por anomalías maduren.

4.5 Sistemas de prevención de Intrusos (IPS)

Las organizaciones involucradas con la seguridad actualmente están interesadas no solamente en detectar los ataques, sino en la manera de prevenirlos dando paso a sistemas de prevención de intrusos, IPS (*Intrusion Prevention System*).

El término prevención se refiere a que las intrusiones detectadas no sean consumadas y podamos dejar fuera de nuestra red incidentes mayores de seguridad. Su funcionamiento parte de una acción reactiva bloqueando el tráfico que sea catalogado como malicioso, si habláramos de detección de intrusos la acción sería únicamente pasiva. Cabe notar que los ataques y las intrusiones se seguirán presentando, los sistemas de prevención de intrusos previenen que los atacantes consumen una intrusión y tomen control de nuestra red o servidores.

Un IPS es un dispositivo de seguridad que ejecuta el control de acceso para proteger a un sistema o una red. La tecnología de prevención de intrusos es considerada por algunos como una extensión de la tecnología de detección de intrusos, IDS, sin embargo, actualmente es una forma más de establecer el control de acceso.

4.5.1 Intercepción de sesiones IPS

Existe una clasificación de sistemas de prevención de intrusos que debido a su funcionamiento se utiliza una arquitectura de red diferente. Los diferentes tipos de IPS son dos, intercepción de sesiones y gateway IPS. Este tipo de IPS se encarga de dar fin a una sesión de TCP enviando un paquete RST. En la Figura 4.1 se muestra un IPS ejecutándose como interceptor de sesiones. Cuando un ataque es detectado un paquete de tipo RST se envía al host atacante finalizando así la comunicación.

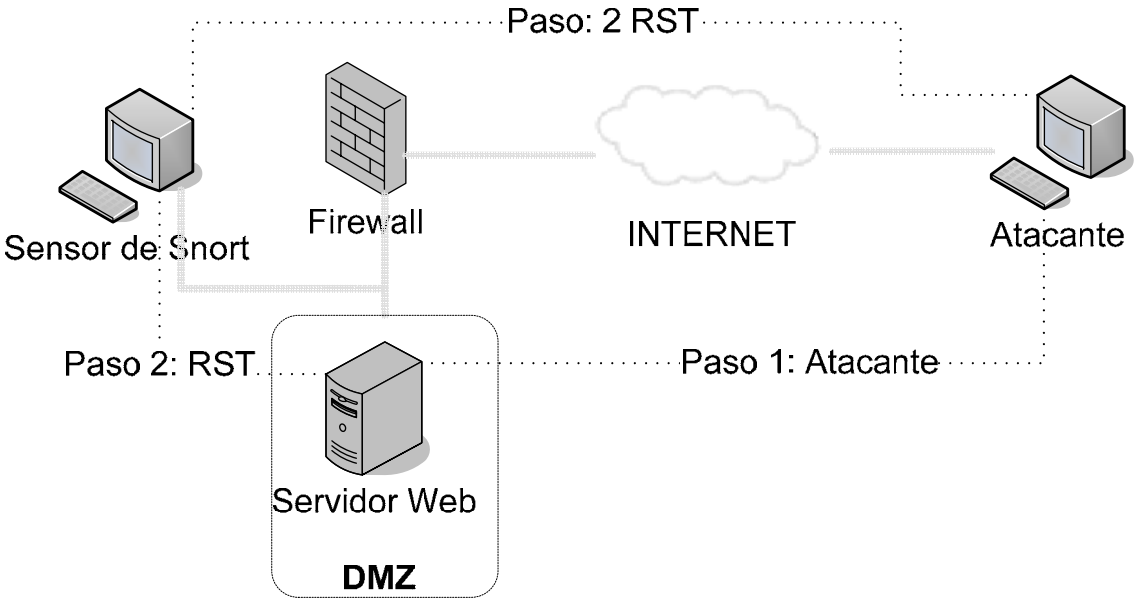

Figura 4.1 Intercepción de sesiones

4.5.2 Gateway IPS

Un IPS puede bloquear tráfico hostil auxiliándose de otras herramientas como Firewalls, los cuales permiten interactuar directamente con el sistema de filtrado del kernel del sistema operativo donde éste implementado, así se podrá manipular la política de reglas de manera dinámica.

En la Figura 4.2 se muestra un sensor actuando como Firewall, router, e IPS. Cuando un ataque es detectado, se bloquea el tráfico proveniente del atacante.

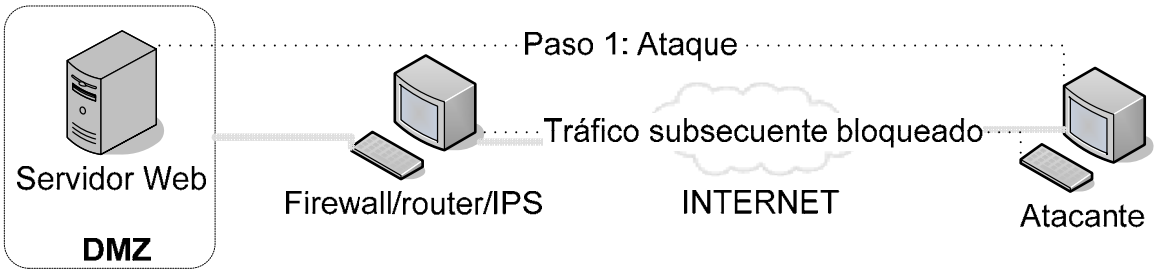

Figura 4.2 Gateway IPS

Una opción más es instalar sensores en lugares estratégicos de la red, manipulando la política de reglas del Firewall de manera remota con la información proveniente de los sensores, así se podrán enviar mensajes al Firewall para bloquear el trafico malicioso. En la Figura 4.3 se muestra un sensor analizando todo el tráfico de red que llega desde Internet y manipulando la política de acceso de acuerdo con las alarmas que se ejecuten.

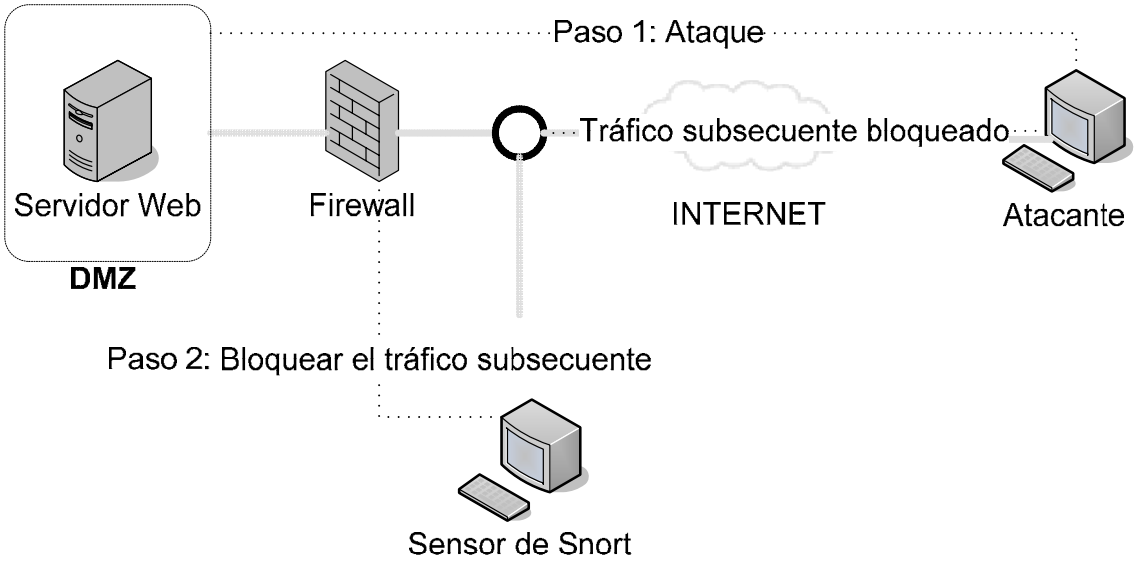

Figura 4.3 Gateway IPS con manipulación

En la tabla 4.2 se muestran las ventajas y desventajas de utilizar un IPS de tipo gateway y manipulador de sesiones.

IPS Interceptor de sesiones	
Ventajas	No se requiere un equipo con grandes recursos En caso de existir un falso positivo no se bloquearía tráfico válido
Desventajas	Se podría provocar que el IPS envíe un número indeterminado de mensajes RST provocando un ataque de negación de servicio
Gateway IPS	
Ventajas	Es una solución económica ya que se requiere de un solo equipo para su implementación
Desventajas	Se tienen muchos servicios en un solo equipo En caso de existir un falso positivo se bloquearía tráfico valido Se requiere un equipo con una gran cantidad de recursos
Gateway IPS con manipulación	
Ventajas	No se requieren un equipo con grandes recursos
Desventajas	En caso de existir un falso positivo se bloquearía tráfico valido

Tabla 4.2 Ventajas y desventajas de los IPS

De manera general la implementación de un IPS puede provocar la negación de servicios válidos, por lo que es recomendable colocar estos dispositivos en lugares donde se tenga un conocimiento amplio sobre el tipo y cantidad de tráfico de información que circula por el punto de red donde se desea colocar el dispositivo.

En la tabla 4.3 se muestran los criterios de selección entre un IDS y un IPS

Criterios de selección entre IDS e IPS	
IDS	– Sólo se requiere monitorear el tráfico de red y alertar en caso que exista un posible ataque, sin tomar ninguna acción inmediata – Se tiene un conocimiento moderado del tipo de tráfico de red – La disponibilidad es alta y no se puede permitir la negación de servicios
IPS	– Se requiere tomar acción inmediata en caso de que se detecte un ataque – Se necesita un conocimiento amplio sobre la cantidad

	y el tipo de tráfico de red donde se va a implementar el dispositivo de manera tal que se reduzca el riesgo de una negación de servicios provocada por falsos positivos. – Se puede permitir la negación de servicios durante el periodo de análisis del presunto ataque

Tabla 4.3 Criterios de selección IDS e IPS

4.6 Snort

Snort es un sistema detector de intrusos basado en software libre. Snort esta diseñado principalmente para funcionar desde la línea de comandos, además ha sido integrado para formar parte de varias aplicaciones. Snort también ha sido transportado a varias plataformas.

Snort se puede ocupar con una detección basada en anomalías aunque su uso por defecto es utilizando un sistema de detección basado en firmas. Snort es un sistema detector de intrusos de red (NIDS) en el cual el número de falsos positivos se puede minimizar configurando correctamente la mayoría de sus directivas de configuración.

Debido a que Snort es software libre se pude manipular su código fuente, lo cual permite que se puedan hacer extensiones en su funcionalidad. Una modificación interesante es la forma en la que puede entregar las alertas generadas por la detección. Snort permite que las alertas vayan dirigidas a un socket tipo AF_UNIX, de esta forma se pueden manipular las alertas y poder crear diferentes aplicaciones, como un IPS, haciendo de Snort un ente reactivo al disparar una alerta.

4.6.1 Configuración básica en OpenBSD

Para la instalación de Snort en OpenBSD se necesita configurar una variable de ambiente que se refiere a un lugar donde el sistema operativo puede encontrar paquetes de instalación, esta variable de ambiente se llama PKG_PATH y su valor puede ser la siguiente dirección IP

ftp://ftp.openbsd.org/pub/OpenBSD/4.0/packages/i386/

Sin embargo, puede haber otra dirección en donde también se encuentren paquetes para el sistema operativo OpenBSD.

Una vez que se haya configurado la variable se puede proseguir con la instalación del paquete de Snort para OpenBSD. Cabe notar que para asignarle el valor a la variable de entorno varía mucho dependiendo del intérprete de comandos que se ocupe[3].

Para la instalación del paquete de Snort se debe teclear el siguiente comando con privilegios de administración.

pkg_add -v Snort-2.4.5p0

Después de que termine de ejecutarse el comando, tendremos Snort instalado pero no listo para utilizarse, antes debemos configurarlo.

El punto más crucial para el funcionamiento de Snort es la configuración de las firmas de ataques conocidos, para obtener este conjunto de firmas es necesario estar registrado en el sitio Web de Snort y descargar el compendio de firmas. Después de descargar el archivo hay que colocarlo en algún lugar del sistema de archivos e indicar la ruta en el archivo de configuración snort.conf, a continuación indicamos las directivas necesarias para el funcionamiento básico de Snort como NIDS.

var HOME_NET <valor>
var EXTERNAL_NET <valor>
var RULE_PATH <valor>
config logdir: <valor>

Las directivas mencionadas arriba son las más básicas, el archivo de configuración tiene muchas directivas más con las que se puede optimizar y expandir el funcionamiento de Snort. HOME_NET se refiere a la red interna, EXTERNAL_NET se refiere la red externa, en RULE_PATH se indica la ruta del directorio donde se colocaron las firmas de los ataques con los que Snort va a hacer la detección. La directiva config logdir: se refiere al lugar en donde se van a colocar los archivos de bitácoras de Snort incluido el archivo de alertas.

3 para el caso de bash se puede utilizar el siguiente comando:
export PKG_PATH=ftp://ftp.openbsd.org/pub/OpenBSD/4.0/packages/i386/

Para levantar el daemon de Snort desde la línea de comandos hay que teclear el siguiente comando con privilegios de administración.

snort -D -c /etc/snort/snort.conf

Otra consideración que hay que tomar es si el equipo se llega a reiniciar por falta de suministro eléctrico debemos de tener en cuenta que el daemon de Snort debe de iniciar cuando arranque el equipo pues la mayoría de las veces estos equipos están situados en cuartos de comunicaciones donde no hay personal en sitio. Para lograr esto hay que agregar las siguientes líneas al archivo /etc/rc.local

```
if [ -x /usr/local/bin/snort ]; then
        echo -n 'Snort'; /usr/local/bin/snort -D -c /etc/snort/snort.conf
fi
```

Capítulo

5. Sockets

5.1 Definición

Muchas comunicaciones entre computadoras y procesos utilizan el modelo cliente-servidor que consiste básicamente en un programa llamado cliente que realiza peticiones a otro programa al que se le conoce como servidor que les da respuesta.

La comunicación entre los dos elementos, cliente y servidor, se logra implementado un protocolo, el cual define la forma en la que estos dos elementos se “hablarán”. El canal de comunicación puede ser a través de una red o en la misma computadora (localhost). La Figura 5.1 Muestra el modelo cliente-servidor.

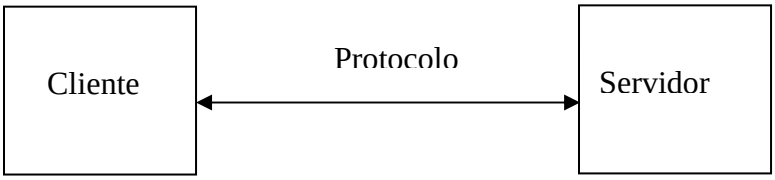

Figura 5.1 Modelo cliente-servidor

Un socket es un ente abstracto por el cual dos programas se van a poder comunicar y transmitir un flujo de datos confiable utilizando el modelo cliente-servidor. Para definir un socket se necesita una dirección IP, un protocolo y un puerto.

Los sistemas operativos actuales basan las comunicaciones internas entre procesos a través de sockets. Todos los servicios de red como HTTP y SMTP, basan también su comunicación entre los clientes y el servidor con sockets. Para poder lograr una comunicación entre computadoras son necesarios los sockets.

La definición original de un socket está en el RFC 147 y fue establecida para la red ARPA, predecesora de Internet, en 1971.

En este capítulo se hace referencia al capítulo 1 “Fundamentos de redes” por que los sockets son la parte de software que logra la comunicación entre redes, con ellos se logran implementar los protocolos, los puertos, las direcciones IP, etc.

5.2 Sockets API

El acrónimo de API por sus siglas en inglés significa *Application Programming Interface*, interfaz de programación de aplicaciones. Es el conjunto de funciones y procedimientos que ofrece cierta biblioteca de programación para ser utilizados en otro programa como una capa de abstracción.

El API para el uso de sockets fue originalmente desarrollada para el sistema operativo 4.2BSD en 1983 por el grupo de investigación de sistemas de cómputo (CSRG, *Computer Systems Research Group*) de la Universidad de Berkeley. Esta API ha sobrevivido a través de los años con cambios estructurales y nuevas implementaciones derivados de los cambios en los protocolos, además de que es el estándar para la comunicación con sockets en muchas versiones de UNIX. En la figura 5.2 se puede ver la evolución de esta API.

Esta API brinda todas las funciones necesarias para la creación de programas robustos que necesiten comunicación entre computadoras y procesos. Su desarrollo está hecho para su uso en el lenguaje de programación estructurado C. Existen otras implementaciones de sockets para lenguajes de programación diferentes, la mayoría toma como base la API de BSD.

5.3 Address Family

Address Family (dirección de familia) es un parámetro en los sockets que determina y especifica el formato de las estructuras de direcciones que pueden manipular las funciones descritas en el API de sockets, es muy importante que no confundamos estas direcciones con las relacionadas al protocolo IP.

También determina la familia de protocolos para proveer la comunicación a través de la red de los datos de aplicación de una computadora a otra o de un proceso a otro en la misma computadora.

Existen varias familias de sockets, hay algunas bastante difundidas y de uso muy general como lo son las que vamos a ocupar en este trabajo y hay otras que son muy particulares de otros tipos de redes como Xerox Network Systems y Appletalk.

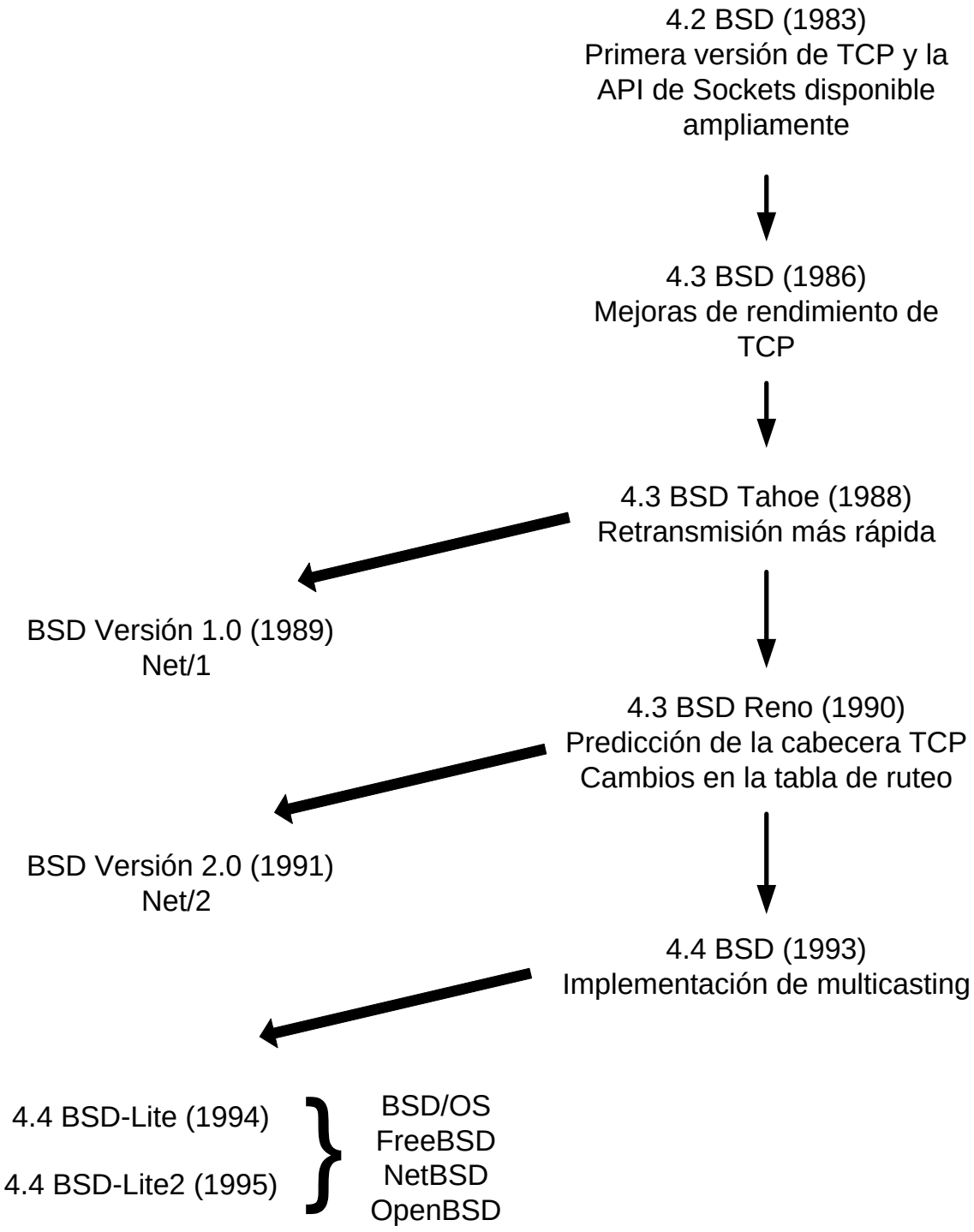

Figura 5.2 Historia de red de BSD

5.3.1 AF_UNIX

Esta familia es utilizada para comunicación entre procesos en sistemas operativos tipo UNIX y tanto el programa cliente como el servidor deben residir en la misma computadora. La dirección en

este tipo de familia es realmente el nombre de una trayectoria válida dentro del sistema de archivos de la computadora.

La estructura asociada con este tipo de familia es la siguiente:

```
struct sockaddr_un {
unsigned char sun_len;
unsigned char sun_family;
char sun_path[104];
};
```

Donde sun_family define la familia, en este caso AF_UNIX, sun_path la trayectoria dentro del sistema de archivos donde se va a crear el socket y sun_len la longitud de la dirección del socket.

Las bibliotecas que hay que incluir para poder hacer uso de esta API con sockets UNIX son sys/socket.h y sys/un.h.

5.3.2 AF_INET

Esta familia provee comunicación entre procesos en la misma computadora y entre diferentes computadoras; por lo que el programa cliente y servidor pueden residir en la misma computadora o en diferentes que es lo más utilizado. Dentro de esta familia se encuentran los sockets de Internet. Las direcciones de esta familia están compuestas por una dirección IP y por un número de puerto.

La estructura asociada con este tipo de familia es la siguiente:

```
struct sockaddr_in {
        u_int8_t sin_len;
        sa_family_t sin_family;
        in_port_t sin_port;
        struct in_addr sin_addr;
        int8_t sin_zero[8];
};
```

Donde sin_len define la longitud de la dirección del socket, sin_family define la familia, en este caso AF_INET, sin_port define el número del puerto, sin_address contiene a la dirección IP y

sin_zero es un campo reservado que contiene ceros en hexadecimal.

Las bibliotecas que hay que incluir para utilizar esta API con sockets de Internet son sys/socket.h, netinet/in.h y arpa/inet.h.

5.4 Tipos de sockets

El tipo de socket provee la información de cómo serán transportados los datos de una computadora a otra, o de un proceso a otro. Principalmente hay dos tipos de sockets, sock-stream y sock-dgram. Sock-stream y sock-dgram pueden ser utilizados con las familias AF_UNIX Y AF_INET.

5.4.1 Sock-stream

Este tipo de socket está orientado a la conexión y es el utilizado para servicios que necesiten la comprobación de que los datos son recibidos en el cliente y en el servidor, sin errores ni duplicaciones además de que los datos son recibidos en el orden en el que fueron enviados. Este tipo de socket es utilizado en las aplicaciones que necesiten ser desarrolladas con el protocolo TCP. En la figura 5.3 se muestra la secuencia de funciones para crear un sock-stream, cabe notar que el intercambio de señales en tres pasos se da después de la función accept en el programa servidor y después de la función connect en el programa cliente.

5.4.2 Sock-Dgram

Este tipo de socket no está orientado a la conexión y no garantiza la entrega de los datos entre el cliente y el servidor. Se puede llegar a perder información en la entrega, por lo que no es confiable, además de que los datos no llegan en el mismo orden en el que fueron enviados. Este tipo de socket es utilizado por las aplicaciones que necesitan ser desarrolladas bajo el protocolo UDP. En la figura 5.4 se muestra la secuencia de funciones para crear un sock-dgram.

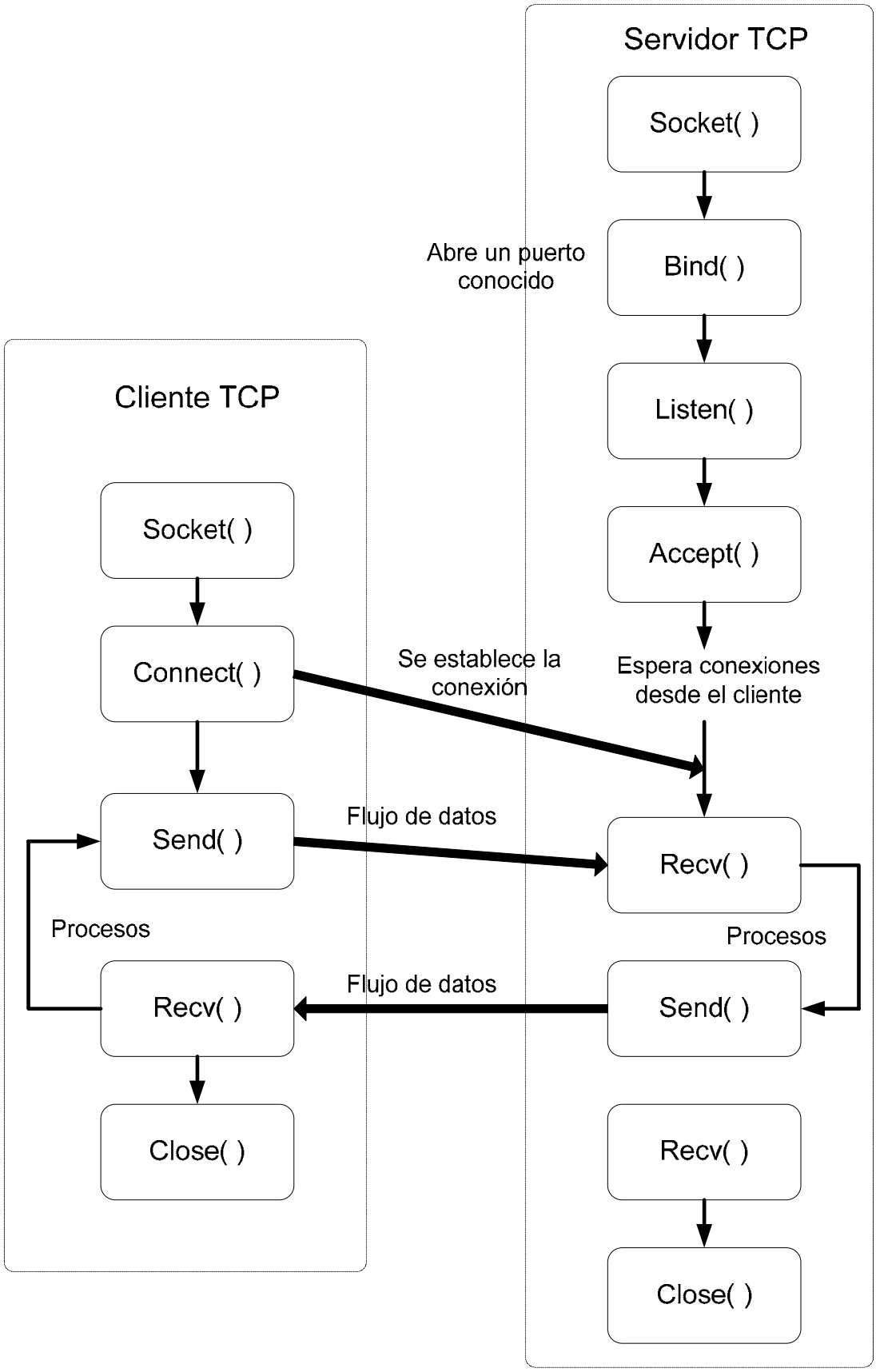

Figura 5.3 Secuencia de funciones para sock-stream

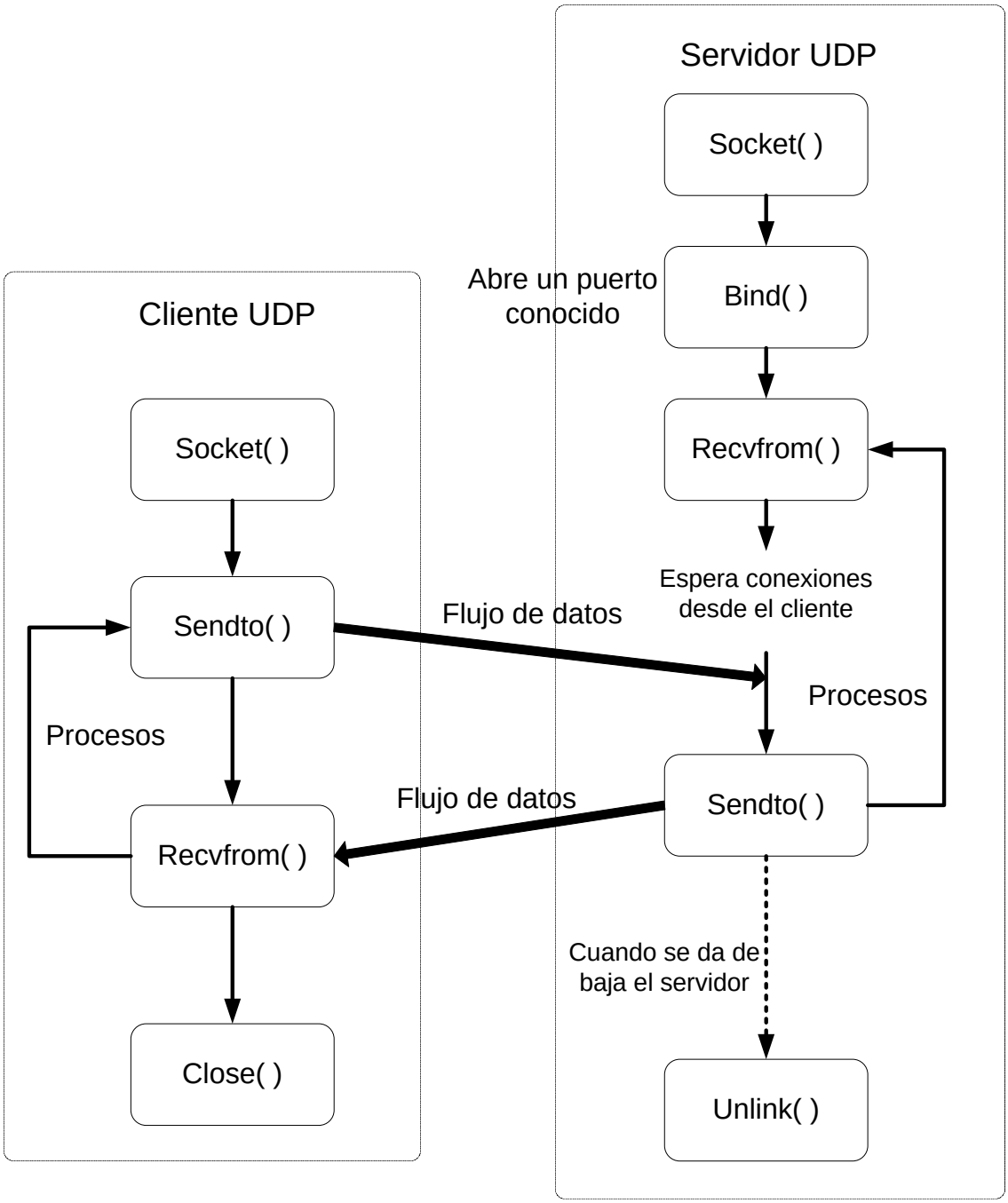

Figura 5.4 Secuencia de funciones para sock-dgram

En la tabla 5.1 podemos ver un resumen de la correspondencia y el uso de los tipos de sockets, tanto los orientados a la conexión como los que no.

Familia	Localmente	En red	Sock-stream	Sock-dgram
AF_UNIX	Sí	No	No	Sí
AF_INET	Sí	Sí	Sí	No

Tabla 5.1 Tipos de sockets

5.5 Funciones de sockets en UNIX

Las funciones mencionadas en las figuras 5.4 y 5.5 son las necesarias para crear una comunicación TCP y UDP entre dos programas, cliente y servidor. El orden descrito en esas figuras es necesario para establecer un protocolo de comunicación sencillo.

Las funciones que describimos no son las únicas que están implementadas en la API, pero sí son las necesarias para realizar aplicaciones sencillas y son las que nosotros utilizamos durante la programación de la herramienta desarrollada.

5.5.1 Socket

La primera función que se tiene que utilizar para crear una comunicación a través de la red o entre procesos en la misma computadora es la función socket. Esta función especifica la familia a la que va a pertenecer el nuevo socket y el tipo.

Está definida de la siguiente manera, mencionamos también las bibliotecas necesarias para su uso:

#include <sys/types.h>
#include <sys/socket.h>
int socket (int family, int type, int protocol);

Donde family determina la familia a la que va a pertenecer (AF_UNIX o AF_INET), type el tipo de socket que vamos a utilizar (sock-dgram o sock-stream), protocol determina qué tipo de protocolo es el que se va utilizar, para obtener una lista de protocolos válidos podemos ver el archivo /etc/protocols de OpenBSD, la alternativa más utilizada es poner el número cero, así el sistema operativo determinará cual es el protocolo más apropiado.

La función socket devuelve un descriptor de archivo referenciado al socket creado si funcionó normalmente, si hubo algún error devuelve el valor -1.

5.5.2 Bind

La función bind permite ponerle un nombre a un socket sin nombre, esto es necesario para poder identificarlo y poder hacer referencia a él. Cuando creamos un socket con la función socket le damos los atributos pero no lo dotamos de un nombre.

La función bind está definida de la siguiente manera, definimos también las bibliotecas necesarias para su uso:

```
#include <sys/types.h>
#include <sys/socket.h>
int bind (int s, const struct sockaddr *name, socklen_t namelen);
```

Donde s recibe un entero que hace referencia a un descriptor de archivo creado con la función socket, name recibe un apuntador de tipo const struct sockaddr que apunta a la dirección de memoria de un tipo de dato creado con las estructuras que definimos en las familias de sockets, sockaddr_un y sockaddr_in, finalmente namelen recibe el tamaño de name.

La función bind devuelve cero si todo ocurrió con normalidad o -1 si hubo algún tipo de error.

5.5.3 Listen

La función listen sólo puede ser utilizada para un socket que haya sido definido como tipo sock-stream y para un programa que vaya a ser utilizado como servidor orientado a la conexión (TCP). Esta función es utilizada para aceptar conexiones de los clientes.

Está definida de la siguiente manera, definimos también las bibliotecas necesarias para su uso:

```
#include <sys/types.h>
#include <sys/socket.h>
int listen(int s, int backlog);
```

Donde s recibe un entero que hace referencia a un descriptor de archivo creado con la función socket, el tipo de socket debe estar definido como sock-stream, backlog recibe un entero que define la máxima longitud de la cola de espera para conexiones.

La función listen devuelve cero si todo ocurrió con normalidad o -1 si hubo algún tipo de error.

5.5.4 Accept

La función accept sólo puede ser utilizada para un socket que haya sido definido como tipo sock-stream y para un programa que vaya a ser utilizado como servidor orientado a la conexión (TCP). Mediante esta función el servidor lee las peticiones de los clientes.

Está definida de la siguiente manera, definimos también las bibliotecas necesarias para su uso:

#include <sys/types.h>
#include <sys/socket.h>
*int accept(int s, struct sockaddr *addr, socklen_t *addrlen);*

Donde s recibe un entero que hace referencia creado con la función socket, el tipo de socket debe de estar definido como sock-stream, addr recibe un apuntador que define la dirección del socket donde se van almacenar los datos del cliente, finalmente addrlen recibe el tamaño de addr.

La función accept devuelve un descriptor de archivo referenciado al socket aceptado como cliente si funcionó normalmente, si hubo algún error devuelve -1.

5.5.5 Connect

La función connect sólo puede ser utilizada para un socket que haya sido definido como tipo sock-stream y para un programa que vaya a ser utilizado como cliente orientado a la conexión (TCP). Mediante esta función el cliente inicia una comunicación con el servidor.

Está definida de la siguiente manera, definimos también las bibliotecas necesarias para su uso:
#include <sys/types.h>

#include <sys/socket.h>
*int connect(int s, const struct sockaddr *name, socklen_t namelen);*

Donde s recibe un entero que hace referencia a un descriptor de archivo creado con la función socket, el tipo de socket debe de estar definido como sock-stream, name recibe un apuntador que define la dirección del socket donde se van almacenar los datos del servidor, finalmente namelen recibe el tamaño de name.

La función accept devuelve cero si todo ocurrió con normalidad o -1 si hubo algún tipo de error.

5.5.6 Send

La función send sólo puede ser utilizada para un socket que haya sido definido como tipo sock-stream y para un programa que vaya a ser utilizado como servidor o cliente orientado a la conexión (TCP). Mediante esta función se manda un mensaje a otro socket.

Está definida de la siguiente manera, definimos también las bibliotecas necesarias para su uso:

#include <sys/types.h>
#include <sys/socket.h>
*ssize_t send(int s, const void *msg, size_t len, int flags);*

Donde s recibe un entero que hace referencia a un descriptor de archivo creado con la función socket, el tipo de socket debe de estar definido como sock-stream, msg es un apuntador que recibe el socket a través del cual se van enviar los datos, len es la longitud de los datos a enviar y flags hace una distinción en la forma en la que se envían los paquetes.

La función send devuelve el número de bytes que fueron enviados o -1 si hubo algún tipo de error.

5.5.7 Recv y recvfrom

Las funciones recv y recvfrom se utilizan para recibir datos de otro socket. La función recvfrom sólo puede ser utilizada para un socket que haya sido definido como tipo sock-dgram y para un programa que vaya a ser utilizado como servidor o cliente no orientado a la conexión (UDP). La función recv sólo puede ser utilizada para un

socket que haya sido definido como tipo sock-stream y para un programa que vaya a ser utilizado como servidor o cliente orientado a la conexión (TCP).

La función recv está definida de la siguiente manera, definimos también las bibliotecas necesarias para su uso:

#include <sys/types.h>
#include <sys/socket.h>
*ssize_t recv(int s, void *buf, size_t len, int flags);*

Donde s recibe un entero que hace referencia a un descriptor de archivo creado con la función socket, el tipo de socket debe de estar definido como sock-stream, buf es un apuntador al socket que va ser encargado de recibir los datos de otro socket, len es la longitud del socket buf, finalmente flags hace una distinción en la forma en la que se reciben los paquetes.

La función recvfrom está definida de la siguiente manera, definimos también las bibliotecas necesarias para su uso:

#include <sys/types.h>
#include <sys/socket.h>
*ssize_t recvfrom(int s, void *buf, size_t len, int flags, struct sockaddr *from, socklen_t *fromlen);*

Donde s recibe un entero que hace referencia a un descriptor de archivo creado con la función socket, el tipo de socket debe de estar definido como sock-dgram, buf es un apuntador al socket que va ser encargado de recibir los datos de otro socket, len es la longitud del socket buf, flags hace una distinción en la forma en la que se reciben los paquetes, from es un apuntador al socket que envía los datos, finalmente fromlen el tamaño de from.

Las funciones recv y recvfrom devuelven el número de bytes que fueron recibidos o -1 si hubo algún tipo de error.

5.5.8 Close y unlink

La función close se utiliza para borrar un descriptor de archivo. En los temas relacionados con la API de sockets es utilizada para finalizar las comunicaciones entre un cliente y un servidor. Generalmente es más utilizada por clientes con sockets del tipo sock-stream.

La función close está definida de la siguiente manera, definimos también las bibliotecas necesarias para su uso:

#include <unistd.h>
int close(int d);

Donde d es el descriptor de archivo del socket que deseamos finalizar.

La función unlink remueve la trayectoria de un archivo, se utiliza para finalizar las comunicaciones entre un cliente y un servidor. Es utilizada en sockets del tipo sock-dgram. La función unlink está definida de la siguiente manera, definimos también las bibliotecas necesarias para su uso, donde path es la trayectoria del socket que deseamos finalizar.

#include <unistd.h>
*int unlink(const char *path);*

Finalmente en la tabla 5.2 se puede observar un resumen de las funciones que se describieron anteriormente, detallando su uso para TCP o UDP y si la función es indispensable para ambos casos.

Función	Indispensable	UDP	TCP
socket	Sí		
bind	Sí		
Listen		No	Sí
Accept		No	Sí
Connect		No	Sí
Send		No	Sí
Recvfrom		Sí	No
Recv		No	Sí
Close	Sí		

Tabla 5.2 Funciones de sockets

Capítulo

6. Análisis, Diseño y Desarrollo

6.1 Metodologías de desarrollo de software

La utilización de una metodología para el desarrollo de software hace más eficiente el proceso de creación de piezas de software. Existen muchas metodologías que durante años se han utilizado en la ingeniería de software, entre las más comunes están el modelo en espiral, metodología por prototipos, metodología de desarrollo de software por etapas y metodología de desarrollo de software en cascada. Hacemos una breve descripción de cada una de ellas y finalmente explicamos los motivos por los cuales escogimos la metodología con la cual desarrollamos nuestra herramienta de seguridad.

6.1.1 Modelo en espiral

El Desarrollo en Espiral es un modelo de ciclo de vida de software desarrollado por Barry Boehm en 1985. Las actividades de este modelo son una espiral en donde cada vuelta representa una iteración en el desarrollo del software

Las etapas de este modelo son:

- Planificación: planificación de los pasos a seguir con las actividades
- Desarrollo y pruebas: desarrollo de la nuevas versiones del software
- Análisis de riesgo: análisis de los riesgos inherentes al nuevo desarrollo.
- Determinación de objetivos: establecer los objetivos para la evolución y mejoramiento del software.

En la figura 6.1 se muestra una gráfica donde se pueden apreciar las iteraciones cíclicas de las cuatro etapas de esta metodología.

6.1.2 Metodología por prototipos

La metodología por prototipos, se inicia con la definición de los objetivos globales para el software, después se identifican los requisitos conocidos y las áreas donde es necesario una mayor definición. A continuación se plantea con rapidez una iteración de construcción de prototipos y se presenta el modelado.

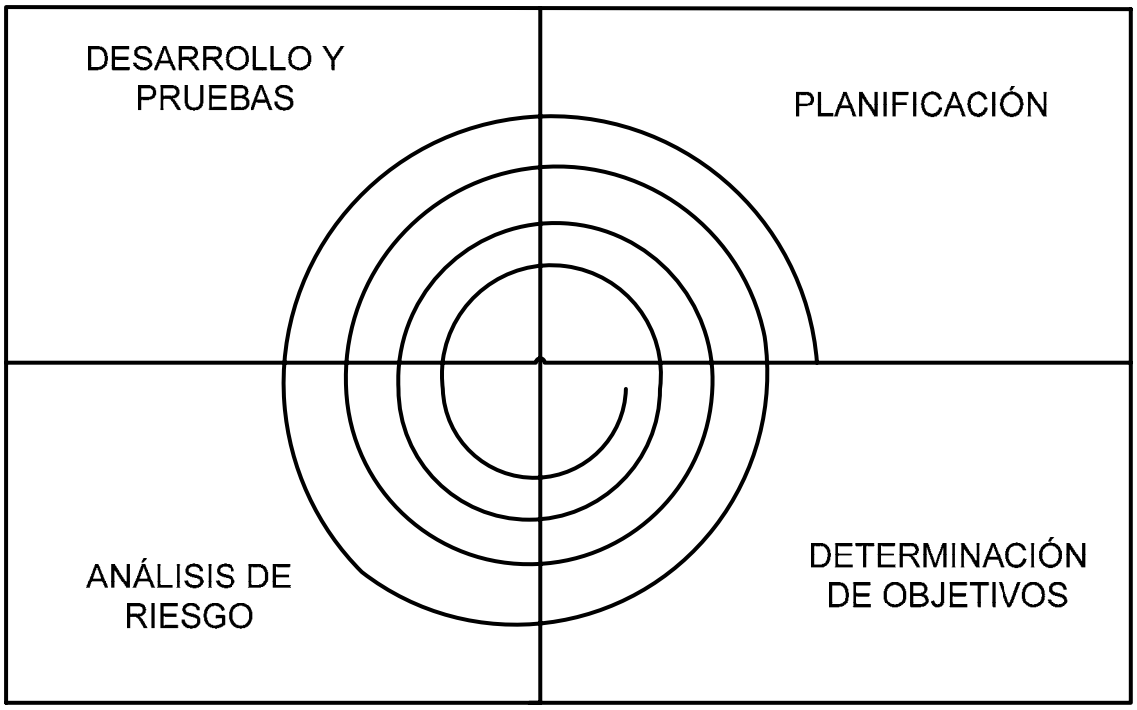

Figura 6.1 Modelo en espiral

El diseño rápido se centra en una representación de aquellos aspectos del software que serán visibles para el cliente o el usuario final (por ejemplo, la configuración de la interfaz con el usuario y el formato de los despliegues de salida). El diseño rápido conduce a la construcción de un prototipo, el cual es evaluado por el cliente o el usuario para una retroalimentación, gracias a ésta se refinan los requisitos del software que se desarrollará. La iteración ocurre cuando el prototipo se ajusta para satisfacer las necesidades del cliente. Esto permite que al mismo tiempo el desarrollador entienda mejor lo que se debe hacer y el cliente vea resultados a corto plazo.

6.1.3 Metodología de desarrollo de software por etapas.

El modelo de desarrollo de software por etapas es similar al modelo de prototipos ya que se muestra al cliente el software en diferentes estados sucesivos de desarrollo, se diferencian en que las especificaciones no son conocidas en detalle al inicio del proyecto y por tanto se van desarrollando simultáneamente con las diferentes versiones del código.

Pueden distinguirse las siguientes fases:

- Especificación conceptual
- Análisis de requerimientos
- Diseño inicial
- Diseño detallado, codificación, depuración y liberación

6.1.4 Metodología de desarrollo de software en cascada

En ingeniería de software el desarrollo en cascada, también llamado modelo en cascada, es el enfoque metodológico que ordena rigurosamente las etapas del ciclo de vida del software, de forma tal que el inicio de cada etapa debe esperar a la finalización de la inmediata anterior.

Las etapas de este modelo pueden resumirse de la siguiente manera:

1. Análisis de requisitos
2. Diseño del Sistema
3. Diseño del Programa
4. Codificación
5. Pruebas
6. Implantación
7. Mantenimiento

La palabra cascada sugiere, mediante la metáfora de la fuerza de gravedad, el esfuerzo necesario para introducir un cambio en las fases más avanzadas de un proyecto.

En este trabajo de tesis hemos decidido utilizar esta metodología para mostrar de mejor manera la herramienta de seguridad que se ha desarrollado. Esta metodología es útil para generar la documentación adecuada para el desarrollo de la herramienta.

6.2 Análisis de Requisitos

Basándonos en el modelo en cascada podemos visualizar que los requisitos que debe cumplir nuestro sistema son los siguientes:

a) El sistema debe utilizar software con licencia tal que permita su libre desarrollo, esto es con la finalidad de que cualquier persona pueda hacer uso del mismo de manera libre. Por lo

anterior se decidió utilizar herramientas con licencias libres entre las que se encuentran las licencias GPL y BSD.

b) El sistema requiere de un sistema operativo estable y con suficiente soporte por lo que para este componente se seleccionó el sistema operativo OpenBSD como se mencionó anteriormente cuenta con un tipo de licenciamiento tal que es posible la manipulación del código fuente. También se optó por OpenBSD por que el desarrollo de éste tiene como característica importante la auditoría continua de su código fuente, lo que hace de OpenBSD uno de los sistemas operativos más robustos que existen.

c) El sistema requiere de un Firewall de software por lo que seleccionó Packet Filter (PF) ya que éste está integrado a nivel del kernel en el sistema operativo OpenBSD.

d) El sistema requiere de un sistema detector de intrusos (IDS). Para este requisito se seleccionó Snort por ser una herramienta de software libre, además del gran soporte que brinda la comunidad que lo desarrolla. Cabe mencionar que Snort es utilizado como base para otros IDS cuya licencia es propietaria.

e) El sistema debe funcionar bajo una arquitectura tipo cliente-servidor, esto es con el objetivo que se puedan colocar varios sensores de Snort en diferentes puntos de la red y que éstos manden la señal de bloqueo al Firewall.

f) El sistema debe utilizar un lenguaje de programación de alto nivel que permita el desarrollo ágil de la aplicación, pero al mismo tiempo que sea un lenguaje compilado para aumentar el rendimiento en tiempo de ejecución ya que se requiere una velocidad de respuesta muy alta.

g) Para el desarrollo de la aplicación se requiere el lenguaje de programación C, por ser el lenguaje en el cual está desarrollado OpenBSD así como la interfaz de desarrollo del Firewall PF y el IDS Snort.

h) El sistema debe detectar los ataques que provengan de redes externas como lo es Internet. Esto debido a que muchos de los ataques provienen de redes públicas.

i) Se requiere que en ningún momento se desactive el Firewall por lo que deben poder agregarse reglas de forma dinámica que permitan bloquear el origen de los ataques detectados.

j) Se requiere que no existan reglas repetidas, ya que mientras más reglas existan se afecta el rendimiento del Firewall, por que se deber verificar que no se agreguen reglas que ya se encuentran configuradas.

6.3 Diseño del Sistema

Siguiendo con los pasos del modelo en cascada a continuación se muestra el algoritmo de funcionamiento de la herramienta.

1. El atacante selecciona un objetivo y lanza su ataque a través de la red.

2. El sensor con Snort instalado identifica el patrón del ataque asociándolo a una firma dentro de su base datos interna.

3. Snort genera una alerta y envía la alerta a un socket AF_UNIX ubicado del lado del sensor enviando la información de direcciones IP origen y destino, el número de puerto origen y destino así como el protocolo utilizado.

4. El socket AF_UNIX manda los datos a un socket cliente AF_INET quien establece comunicación con el socket servidor AF_INET y envía dentro de una estructura la información antes mencionada.

5. El socket servidor AF_INET se encarga de entregar la información recibida desde el cliente AF_INET al programa principal.

6. El programa principal toma la información proporcionada por el socket, la procesa y agrega una regla al estado original del Firewall.

La figura 6.2 muestra el flujo de información del sistema en general.

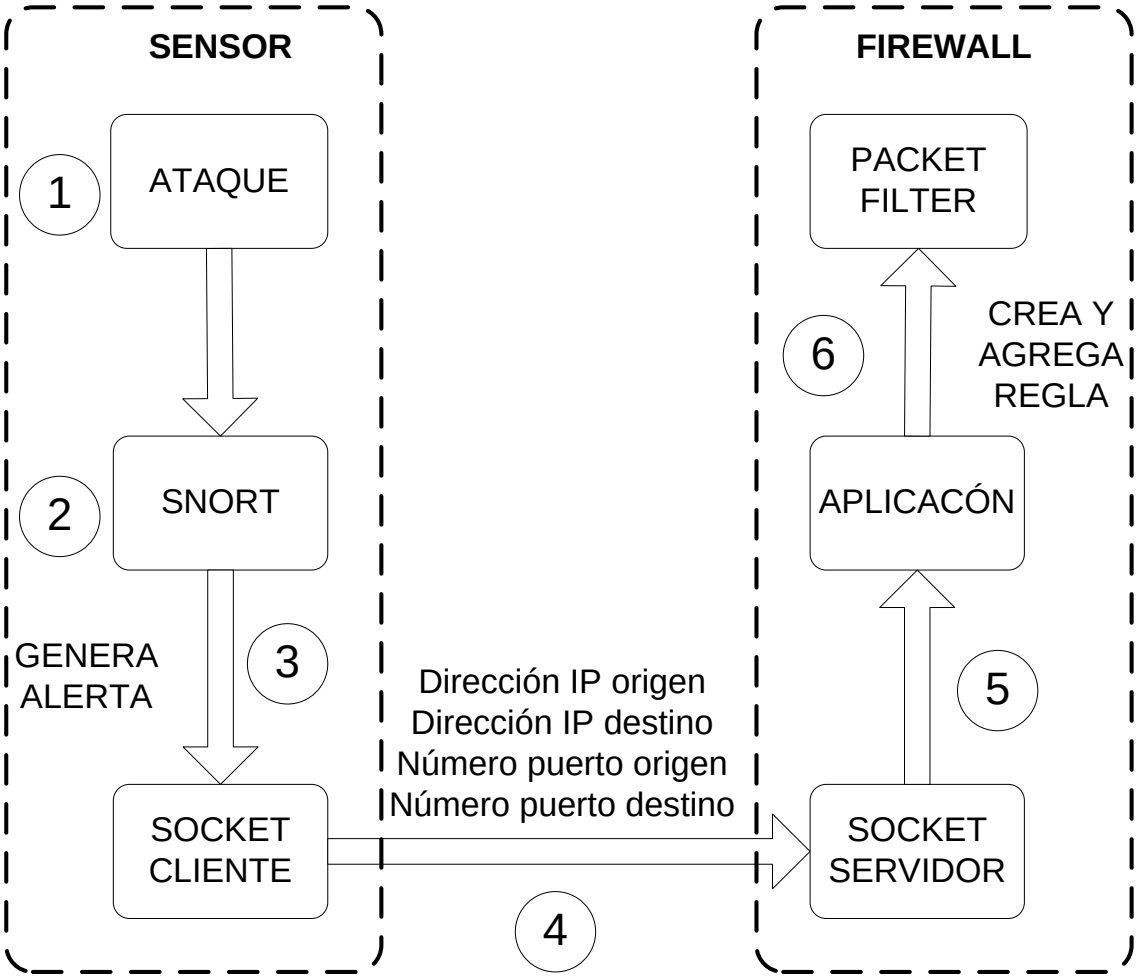

Figura 6.2 Flujo de información del sistema

6.4 Diseño del programa

A continuación se muestra el algoritmo básico de desarrollo del programa principal, aquí se describe básicamente cómo llega la información al programa principal y como éste la procesa para que a través del funcionamiento interno de PF se pueda agregar una regla al Firewall.

1. El programa principal recibe la información de direcciones IP origen y destino, puertos origen y destino así como el protocolo a través de una estructura llamada mipaq vía el socket AF_INET cliente.

2. Desde la función principal se llama a otra función denominada mantiene_set() la cual se encarga de obtener el conjunto activo de reglas, y almacenar todos los valores en una estructura denominada practivo. Esta función realiza una

copia de todos los valores y la almacena en una nueva estructura denominada prnuevo que es un conjunto inactivo de reglas.

3. Haciendo uso de un case se selecciona el tipo de protocolo que emitió la alerta, para configurar este valor en la nueva regla se llama a la función verifica_regla() la cual se encarga de verificar que no exista ya en el conjunto de reglas practivo, en caso de que no exista se llama a la función load_set() para agregar una nueva regla al conjunto inactivo prnuevo.

4. La función load_set() se encarga de configurar la información para generar una nueva regla agregando ésta al final del conjunto inactivo prnuevo. Después la función se encarga de convertir el conjunto prnuevo en el conjunto activo de reglas.

5. El sistema siempre está a la espera de nuevos eventos, por lo que el proceso se repite por cada nueva alerta detectada por Snort.

La figura 6.3 muestra el diagrama de flujo básico del programa principal.

6.5 Implementación del programa

El procedimiento para instalar el sistema es de propósito general y deberá servir para una instalación estándar de todos los componentes de la herramienta, sin embargo pudiera presentarse algunos puntos diferentes.

Como primer paso se requiere instalar el sistema operativo OpenBSD, para información acerca de cómo realizar esta instalación dirigirse a la guía de instalación oficial del proyecto[1]

El siguiente paso es la instalación y configuración de Snort como se menciono en el capítulo cuatro, debido a la instalación para OpenBSD. Para mayor información se puede acceder a la página oficial para la instalación genérica de Snort[2].

Por último se deben copiar los archivos con el código fuente de la

1 http://www.openbsd.org/faq/faq4.html
2http://www.Snort.org/docs/#docs

aplicación a un directorio del sistema operativo, estos son: ipsserv.c el cual se encarga de recibir los mensajes de alerta y manipular los acceso del Firewall, el archivo ipscli.c es el cliente y se encarga de recibir las alertas de Snort y mandar la información hacia el programa ipsserv.c. Para la correcta compilación deben de copiarse también los archivos de bibliotecas de Snort.

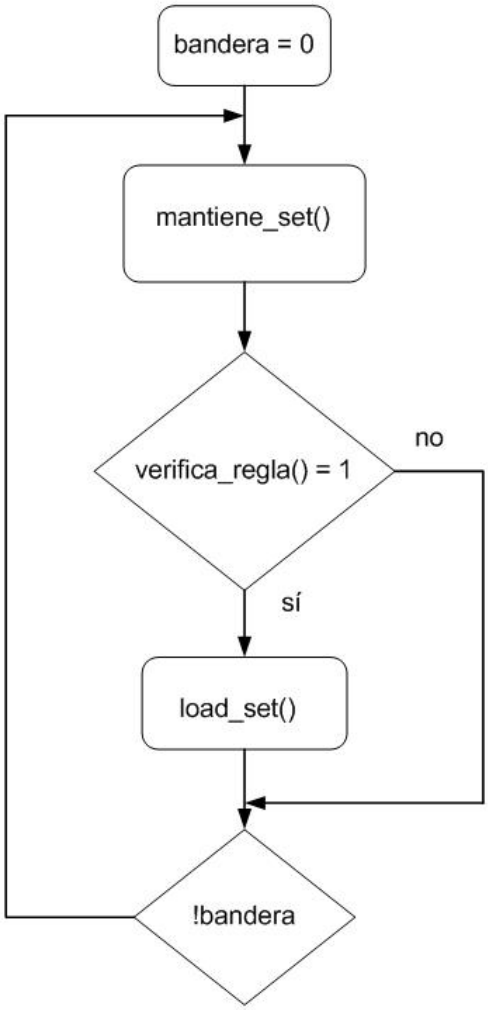

Figura 6.3 Diagrama de flujo programa principal

Una vez que se haya instalado y configurado Snort de la manera como se mencionó en el capítulo cuatro debe modificarse el archivo de configuración agregando la siguiente línea, esto es para lograr que Snort pueda entregar las alertas a un socket del tipo AF_UNIX y poderlas manipular:

output alert_unixsock

A continuación se debe compilar y ejecutar la parte del servidor de la herramienta. Como se muestra a continuación.
Para compilar el código se debe ejecutar el siguiente comando:

gcc ipsserv.c -o ipsserv

Después se debe ejecutar el programa de la siguiente manera con privilegios de administración para poder abrir los puertos necesarios para la comunicación:

./ipsserv

En el siguiente paso debemos de compilar el cliente de la herramienta en la ruta del sistema de archivos que se definió en el archivo de configuración para la salida de las alertas que genera Snort.

gcc ipscli.c -o /var/snort/log/ipscli

Se manda a ejecutar el código poniendo como argumento la dirección IP donde se encuentra la parte del servidor de nuestro programa, en una instalación donde todos los componentes se encuentran en el mismo servidor se especifica la dirección IP local como se muestra a continuación.

 # ./ipscli 127.0.0.1

Por último, en una Terminal diferente se inicia Snort utilizando el siguiente comando

snort -D -c /etc/snort/snort.conf

Donde la opción –c se utiliza para indicar la ruta de directorios donde se encuentra nuestro archivo de configuración y la opción -D para iniciar Snort como un demonio de UNIX.

Por último debe verificarse que el Firewall esté activo, esto se hace con el siguiente comando que se mencionó en el capítulo tres.

 # pfctl -e

Capítulo

7. Pruebas

7.1 Plan de pruebas

Para la correcta satisfacción de los objetivos de la tesis, siguiendo el modelo en cascada, la herramienta desarrollada se debe someter a una fase de pruebas donde se buscaron y corrigieron errores de programación y diseño. Esta fase de pruebas cumple también con la parte más significativa de los objetivos de la tesis, bloquear un ataque real, el plan de pruebas se desarrolló pensando en detectar fallas en el proceso de bloquear el ataque.

Para satisfacer la prueba de bloquear un ataque real se puede escoger alguno de los protocolos con los que se trabajó en la tesis, TCP, UDP o ICMP. Para efectos de mayor impacto se decidió escoger un ataque para TCP demostrando primero cómo se lleva a cabo la intrusión sin seguridad y después cómo reacciona la herramienta desarrollada al estímulo del ataque.

Para el caso de los protocolos UDP e ICMP se hicieron pruebas durante la fase de desarrollo las cuales también fueron satisfactorias al poder bloquear la herramienta ataques sobre estos protocolos. Durante las pruebas que se efectuaron durante el desarrollo de la herramienta se encontró la particularidad de que los ataques sobre el protocolo ICMP no cuentan con un puerto víctima pues en el protocolo no se implementan puertos para comunicación, lo cual implicó cambios en la forma de reportar alertas y agregar reglas dinámicamente al Firewall.

Se eligió realizar la fase de pruebas con un ataque sobre el protocolo TCP, se escogió esta prueba por ser representativa al poder lograr una intrusión real, además que es uno de los protocolos más utilizados por servicios en la capa de aplicación. Cabe notar que el diseño e implementación de la herramienta permite detectar cualquiera de los ataques conocidos por las firmas de Snort en los 3 protocolos TCP, UDP e ICMP, por lo que se hicieron pruebas menores también sobre los protocolos restantes.

El plan de las pruebas es sencillo, primero se hace el ataque real sin seguridad y se muestra la intrusión, la segunda fase es activar la herramienta y verificar paso a paso cómo se bloquea el ataque evitando la intrusión.

Para llevar a cabo el esquema de pruebas se utilizo una maqueta con las siguientes características: Procesador Pentium III a 450 MHz con 128 MB de memoria RAM y disco duro de 6 GB, sistema operativo OpenBSD 4.0 y Snort versión 2.4.5.

7.2 Esquema de pruebas

La fase de pruebas se realiza en un ambiente de red controlado explotando una vulnerabilidad remota a un servicio de red. Para lograr una prueba de cómo se llevaría en un ambiente real en producción necesitamos por lo menos tres elementos básicos, un servidor víctima, un ruteador y un cliente atacante. En la primera fase de la prueba el ruteador sólo servirá para tener dos redes separadas, la red interna donde estará el servidor víctima y la red externa simulando Internet que va alojar al cliente atacante. Con esta configuración se puede llevar a cabo el ataque explotando la vulnerabilidad y obteniendo un intérprete de comandos remoto con privilegios de administrador. La segunda parte de la prueba es habilitar el sistema desarrollado en esta tesis, que consiste en un Firewall y un sistema detector de Intrusos.

El Firewall tendrá entonces las reglas convencionales de los servicios de red que aloja el servidor víctima y el sistema detector de intrusos contará con la configuración necesaria para detectar ataques provenientes de la red externa. La herramienta estará lista para esperar las alertas del detector de intrusos y generar las reglas de Firewall necesarias para bloquear el acceso a los atacantes haciendo así una prevención de intrusos sencilla por dirección IP, puerto y protocolo. En la figura 7.1 se ve el diagrama de red utilizado para llevar a cabo la fase de pruebas.

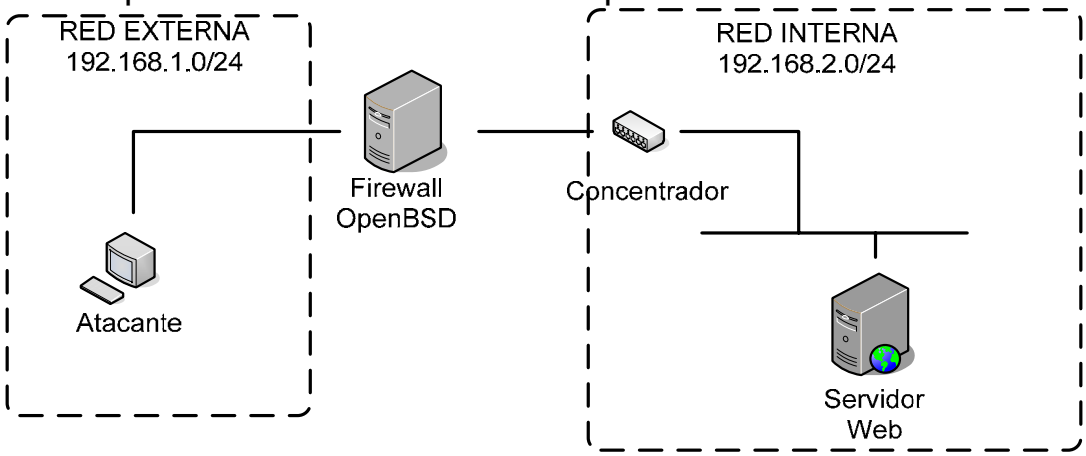

Figura 7.1 Esquema de red de las pruebas

7.3 Ataque real en un ambiente controlado

Como se mencionó anteriormente, el ataque se realizará en dos etapas, esto para detectar puntualmente errores lógicos en el diseño del programa y errores técnicos en la implementación y desarrollo de la herramienta. La primera etapa es un ataque con la metodología general de ataque descrita anteriormente. La segunda etapa es el mismo ataque con seguridad brindada por nuestra herramienta.

7.3.1 Ataque sin seguridad

El ataque en un ambiente controlado que llevamos a cabo está basado en el método general de ataque expuesto anteriormente de una forma práctica explotando una vulnerabilidad en un ambiente parecido a la producción cotidiana de un servidor real.

El primer punto del proceso general de ataque es la selección de un objetivo, en la prueba hemos configurado un servidor Web que simula ser un portal corporativo de una empresa de servicios http://www.empresadeservicios.com. Suponemos un atacante el cual tiene por objetivo desprestigiar la imagen del portal corporativo de la empresa de servicios. Para cumplir con su objetivo el atacante deberá hacer una modificación a la página principal del portal, este tipo de ataque se le conoce como Web defacement o graffiti. En la figura 7.2 se ve la página del portal de la empresa de servicios en su funcionamiento normal.

El segundo punto del proceso general de ataque es el reconocimiento el cual se refiere a la obtención de datos que puedan ser de utilidad. En la prueba se utiliza una consulta al DNS para conocer la dirección IP del servidor Web de la empresa de servicios. Para lograr la consulta se hace uso del comando *nslookup* como se muestra en la figura 7.3. En el reconocimiento es en donde se llevan a cabo con más frecuencia los ataques pasivos ya que no se tiene contacto real con el servidor víctima.

El tercer punto que mencionamos en el proceso general de ataque es la enumeración. La enumeración es un equilibrio entre los ataques pasivos y activos, aunque los activos son los que mayor éxito tienen a la hora de recopilar información útil para la realización

del ataque. En el ataque controlado que se realizó para este método de prueba se hizo el análisis de los puertos abiertos en el servidor que aloja la página Web del la empresa de servicios. Para el análisis de puertos utilizamos una herramienta muy conocida en esta categoría que se llama nmap, la cual brinda la información del sistema operativo que está instalado en el servidor, los puertos abiertos, los servicios detallados con los números de versión que se alojan en los puertos abiertos.

En la imagen 7.4 se muestran los puertos abiertos que tiene el servidor de la empresa de servicios. En la imagen se ve el detalle de los servicios por cada puerto abierto. Entre las cosas interesantes que nos da este comando es saber que la empresa de servicios tiene más puertos abiertos de los que se pensaría inicialmente.

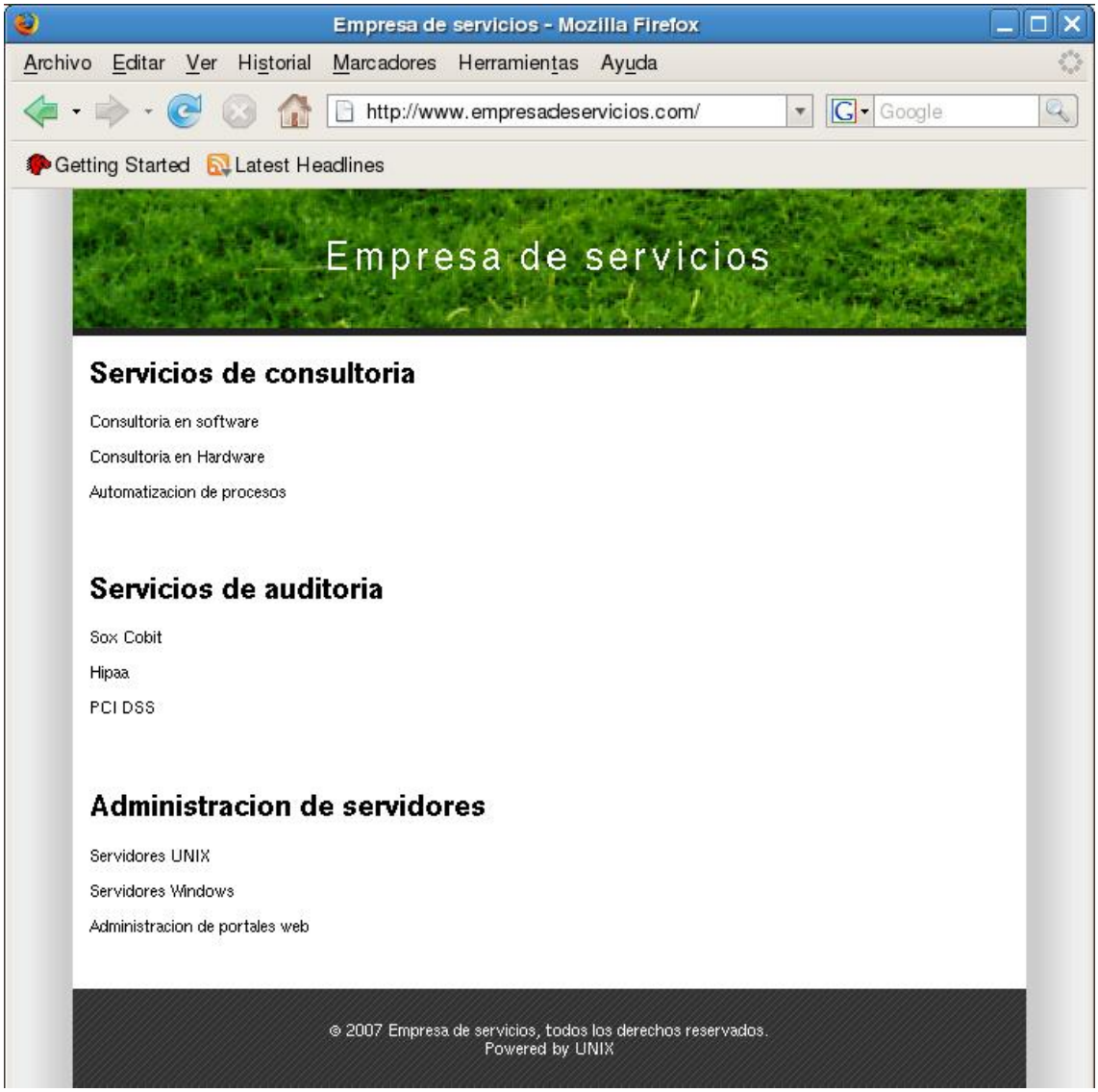

Figura 7.2 Empresa de servicios

```
root@rutabaga:~# nslookup www.empresadeservicios.com
Server:         192.168.3.254
Address:        192.168.3.254#53

Name:   www.empresadeservicios.com
Address: 192.168.2.33
```

Figura 7.3 Reconocimiento

En primera instancia se esperaría que sólo tuviera abierto el puerto 80 para la comunicación Web, pero se puede ver en la imagen que cuenta con varios puertos abiertos entre los que podemos encontrar aplicaciones para administración remota (SSH), compartir archivos con Windows (netbios-ssn) y transferencia remota de archivos (FTP).

```
root@rutabaga:~# nmap 192.168.2.33

Starting Nmap 4.03 ( http://www.insecure.org/nmap/ ) at 2007-11-18 16:36 CST
Interesting ports on 192.168.2.33:
(The 1662 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
21/tcp  open  ftp
22/tcp  open  ssh
25/tcp  open  smtp
37/tcp  open  time
79/tcp  open  finger
80/tcp  open  http
111/tcp open  rpcbind
113/tcp open  auth
139/tcp open  netbios-ssn
515/tcp open  printer
587/tcp open  submission
901/tcp open  samba-swat

Nmap finished: 1 IP address (1 host up) scanned in 0.565 seconds
```

Figura 7.4 Puertos y servicios abiertos

Dentro de la enumeración es necesario obtener más información con el fin de determinar si un servicio es vulnerable. Para poder lograr esto hay que obtener las versiones de los servicios que están abiertos. Nmap nos puede brindar información detallada de las versiones de los servicios abiertos como se muestra en la imagen 7.5.

Otro detalle que se puede encontrar en el la figura 7.5 es que nmap nos puede dar información del sistema operativo que está instalado en el servidor víctima. En la imagen se puede observar el apartado

OS details donde menciona que el servidor esta funcionado con algún sistema operativo con kernel Linux, lo que denota que puede ser alguna de las muchas distribuciones del sistema operativo GNU/Linux.

Analizando la información que se tiene hasta ahora se puede buscar por servicios vulnerables. Uno que llama la atención por su larga historia de problemas de seguridad es Samba, el cual es accesible en el servidor víctima a través del puerto 139/TCP. Para obtener más información de este servicio se utilizó una herramienta llamada nbaudit la cual nos brinda detalles de la versión del servicio, de los usuarios válidos y de los recursos compartidos como se ve en la figura 7.6.

```
root@rutabaga:~# nmap -A -T5 -F 192.168.2.33

Starting Nmap 4.03 ( http://www.insecure.org/nmap/ ) at 2007-11-18 16:37 CST
Interesting ports on 192.168.2.33:
(The 1221 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE     VERSION
21/tcp  open  ftp         ProFTPD 1.2.5
22/tcp  open  ssh         OpenSSH 3.2.3p1 (protocol 1.99)
25/tcp  open  smtp?
37/tcp  open  time         (32 bits)
79/tcp  open  finger      Linux fingerd
80/tcp  open  http        Apache httpd 1.3.26 ((Unix))
111/tcp open  rpcbind      2 (rpc #100000)
113/tcp open  ident       OpenBSD identd
139/tcp open  netbios-ssn Samba smbd (workgroup: WORKGROUP)
515/tcp open  printer?
587/tcp open  submission?
901/tcp open  http        Samba SWAT administration server
Device type: general purpose
Running: Linux 2.4.X|2.5.X|2.6.X
OS details: Linux 2.4.0 - 2.5.20, Linux 2.4.7 - 2.6.11
Service Info: OSs: Unix, Linux, OpenBSD

Nmap finished: 1 IP address (1 host up) scanned in 106.671 seconds
```

Figura 7.5 Versión de los servicios abiertos y del sistema Operativo

En la información obtenida con el programa nbaudit se puede encontrar que se muestran detalles de información muy útil como la versión del servicio Samba y la versión del sistema operativo instalado. En el caso de Samba ahora ya sabemos que el servidor víctima está utilizando la versión 2.2.4 y en el caso del sistema operativo confirmamos la información obtenida con nmap, el sistema operativo es Slackware GNU/Linux versión 8.1. Estos datos son de vital importancia por que ahora el atacante puede hacer

búsquedas en Internet con los detalles obtenidos para localizar posibles vulnerabilidades conocidas.
Haciendo una búsqueda en la página de seguridad y versiones del producto Samba se encuentra que todas las versiones entre la 2.2.2 y la 2.2.7 tienen múltiples problemas de seguridad referenciadas con un número CVE (*Common Vulnerabilities and Exposures*) relacionado que es el CVE-2003-0085.

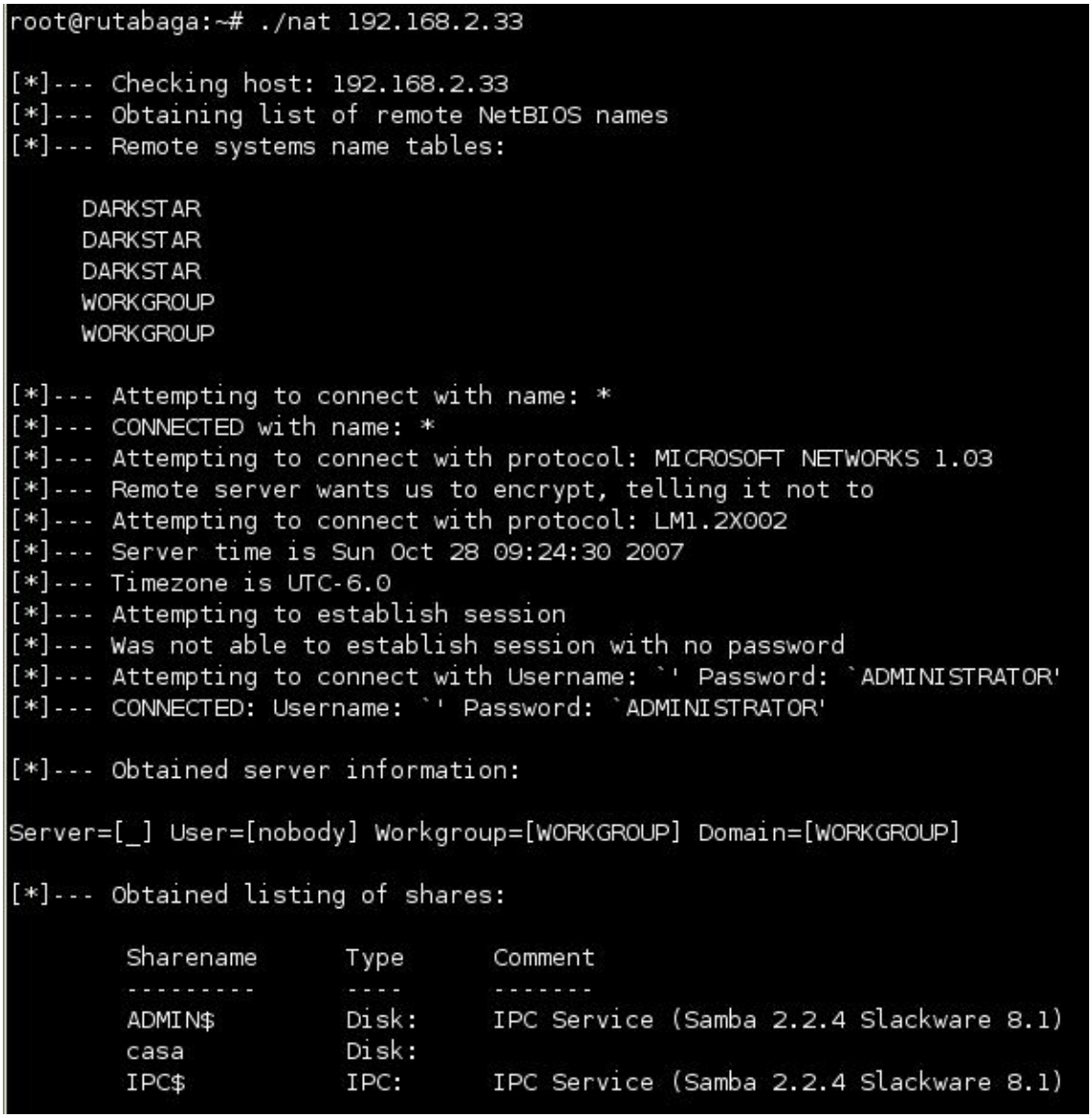

Figura 7.6 Detalle del servicio samba

En este momento el atacante ha encontrado una brecha de seguridad y utilizará la herramienta metasploit para llevar a cabo la explotación que es el siguiente paso dentro de la metodología general de ataque.

Metasploit es un framework para explotar sistemas, entre sus características más importantes es que cuenta con una gran cantidad de exploits para muchos y diferentes servicios, otra característica importante es que brinda la interfaz para poder ejecutar una terminal de intérprete de comandos remota con los permisos obtenidos después de lanzar el ataque, generalmente estos permisos son de administración. En la figura 7.7 se muestra la interfaz Web de la herramienta metasploit.

Figura 7.7 Interfaz Web de Metasploit

Metasploit cuenta con un exploit para la vulnerabilidad mencionada en el número CVE-2003-0085, esta vulnerabilidad es conocida como *Samba Fragment Reassembly Overflow* y se muestra en la figura 7.8. Entre las ventajas que tiene este exploit es que cuenta con varias versiones dependiendo del sistema operativo que tenga la víctima, en la figura 7.8 se puede ver también que existe una versión de este exploit para la versión del sistema operativo que corre el servidor Web de la empresa de servicios que es Slackware 8.1. Con esta información ya confirmada, la versión del sistema operativo y la versión de Samba, se puede realizar la explotación.

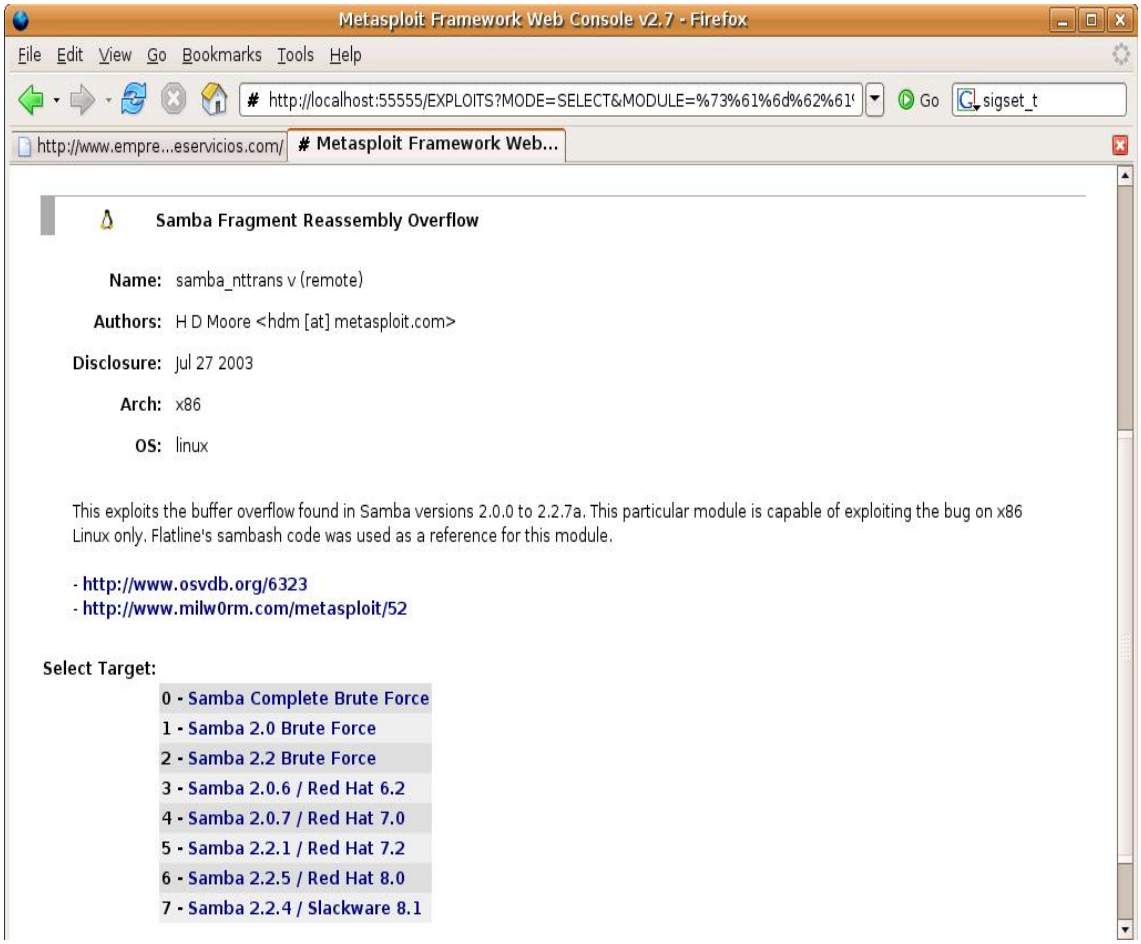

Figura 7.8 Exploit para la versión de samba 2.24

En la figura 7.9 se ve el proceso de explotación. La herramienta es tan automatizada que solamente necesitamos introducir la dirección IP del servidor víctima en el campo RHOST y presionar el botón Exploit para lanzar el ataque.

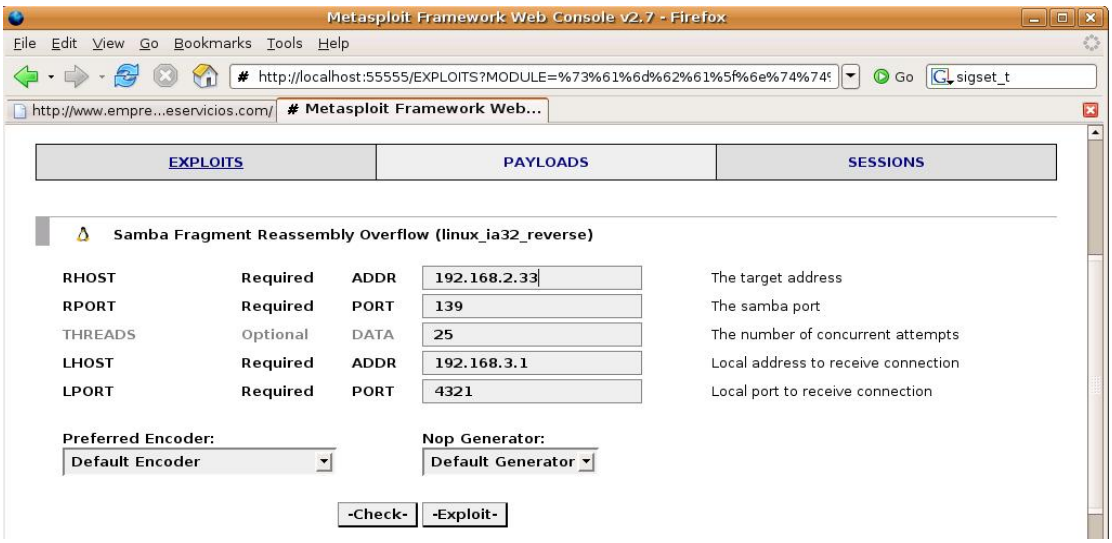

Figura 7.9 Explotación

Una vez lanzado el ataque, en caso de tener éxito se abre una sesión en donde podemos teclear comandos directamente en el equipo comprometido. En la figura 7.10 se pueden apreciar los comandos que el atacante ha tecleado para satisfacer sus motivaciones.

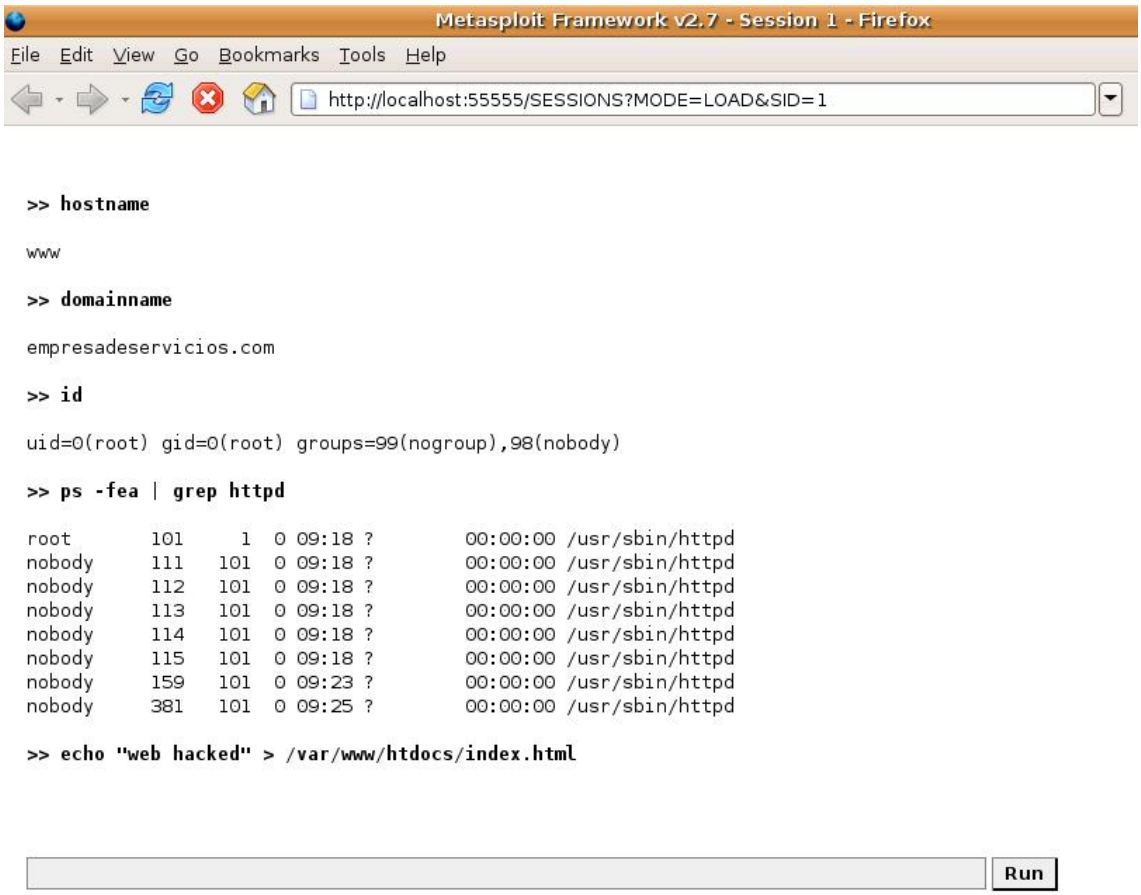

Figura 7.10 Ejecutando comandos en el equipo comprometido

En particular el último comando tecleado es el que realiza la ejecución de las motivaciones, a este punto se le conoce como realización de una meta. Con el comando se sobrescribe la leyenda *Web hacked* en el archivo index.html que es el que contiene el código HTML para mostrar la página Web, como se muestra en la figura 7.11. Al aparecer esta leyenda en lugar del código original de la página Web de la empresa de servicios se hace un desprestigio de la credibilidad y seriedad de la empresa. Al verse comprometida su seguridad se deteriora la calidad de su imagen ante los clientes que tal vez querían adquirir servicios con dicha empresa.

En el método general de ataque se mencionan más pasos que los que realizamos en la prueba, en particular se mencionan los

siguientes pasos extras, ganando acceso, consolidación y evitar la detección.

- Ganando acceso: es un punto que no se utiliza en esta prueba por el tipo de ataque que se lanza regresa los privilegios máximos en el servidor víctima.
- Consolidación: es el punto en el que se debe de asegurar el atacante que él sea el único que pueda vulnerar el equipo, por la naturaleza de las pruebas se necesita que el servicio siga vulnerable para poder hacer la segunda parte de la prueba.
- Evitar la detección: en este punto se busca eliminar las huellas dejadas por el ataque, en la prueba no se hace por las múltiples pruebas que se hacen sobre el servidor víctima.

Figura 7.11 Realización de una meta

7.3.2 Ataque con seguridad

La segunda fase de la prueba es habilitar el sistema desarrollado en la tesis. Para lograr esto vamos a mencionar temas de los que hablamos en los capítulos anteriores, en particular los mencionados en la implementación del programa en el apartado 6.5.

A continuación se describen los pasos necesarios para poder hacer una prueba de ataque con seguridad incluida por la herramienta desarrollada en este trabajo de tesis. Esta prueba se va a desarrollar en el mismo ambiente controlado y con la misma configuración de servicios que en el punto anterior, Ataque sin seguridad.

Lo primero que se tiene que hacer es habilitar el uso del Firewall en OpenBSD como se mencionó en el apartado 3.5 de la tesis. Es necesario también contar con un set de reglas como se menciona en ese apartado; nuestro sistema puede trabajar con un set de reglas casi nulo o con uno más elaborado.

Después hay que habilitar la detección de intrusos con Snort y que la salida de las alertas este dirigida a un socket de tipo AF_UNIX. Para lograr esto hay que seguir los pasos descritos en los apartados 4.6 y 6.5 de la tesis, siendo de vital importancia la correcta configuración de las variables para la red externa e interna.

Levantar los servicios de Firewall y la detección de intrusos se pueden automatizar para que cada vez que se encienda el equipo ya estén presentes. Esto también se explica detalladamente en los capítulos anteriores.

Posteriormente hay que levantar el servidor de nuestra aplicación él cual está encargado de recibir las alertas del cliente y generar las reglas de Firewall necesarias para hacer la prevención de intrusos. Es recomendable levantar el servidor en una ventana de shell individual si se esta conectado remotamente o en una terminal virtual si se esta visualizando directamente en el monitor del equipo. Para lograr esto hay que seguir los pasos relacionados al servidor en el apartado 6.5 de la tesis.

Enseguida hay que levantar el cliente de nuestra aplicación el cual recibe directamente de Snort las alertas de intrusión y manda al servidor la información necesaria al Firewall para que éste pueda crear la regla. De forma similar al servidor hay que tener una ventana individual si se está conectado remotamente u otra terminal virtual si se esta visualizando directamente en el monitor del equipo. Para lograr esto hay que seguir los pasos relacionados al cliente en el apartado 6.5 de la tesis.

En el siguiente punto se lanza nuevamente el ataque como se mostró en el punto anterior de este capítulo, Ataque sin seguridad. En la figura 7.9 se ve la interfaz Web de la herramienta metasploit en la cual el atacante con dirección IP 192.168.3.1 hace un intento de intrusión, en la figura 7.10 se ve el resultado de la intrusión al obtener una shell remota del equipo víctima; con nuestro sistema ya habilitado se lanza el ataque de la misma forma que en la figura 7.9

pero el resultado del ataque no será ahora satisfactorio pues será bloqueado por reglas generadas a partir de nuestro sistema, en la figura 7.12 se ve el intento fallido de intrusión.

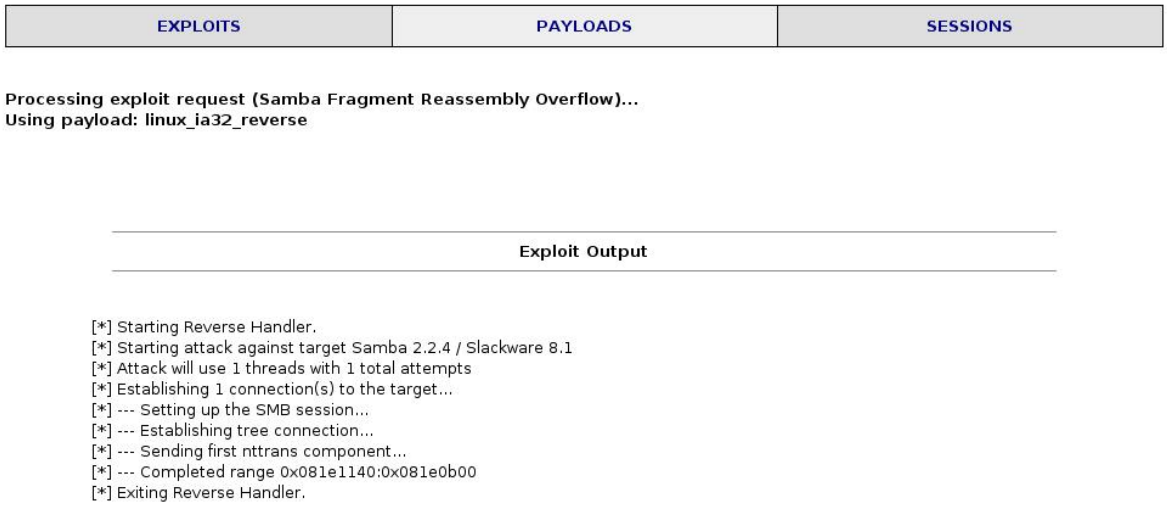

Figura 7.12 Intento fallido de intrusión

Para verificar las acciones que realizó el sistema primero hay que visualizar las ventanas del servidor y la del cliente, en ellas encontramos la información relacionada con el ataque, vamos a ver el nombre de la regla que disparó la alerta, las direcciones IP y puertos origen y destino, figura 7.13.

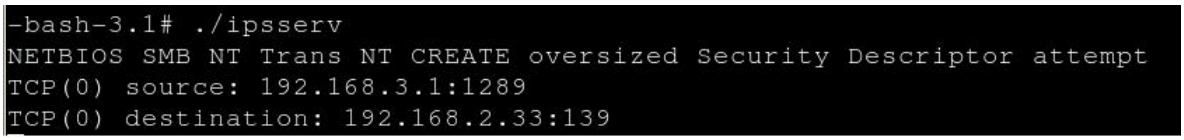

Fig 7.13 Ventana del servidor

El siguiente punto para verificar las acciones que realizó el programa es revisar las nuevas reglas agregadas al set activo del Firewall después del ataque. En particular se generan dos reglas por cada ataque detectado. Una que bloquea la entrada del atacante por puerto, protocolo y dirección IP, el puerto relacionado es por donde el atacante lanzó el intento de intrusión. La otra regla bloquea todo el tráfico de salida de la dirección IP víctima hacia la dirección IP atacante. En la figura 7.14 se ve el set activo de reglas, al final de éste se ven las dos reglas agregadas por nuestro sistema cuando el atacante con dirección IP 192.168.3.1 lanzó un intento de intrusión.

```
-bash-3.1# pfctl -sr
block drop in all
pass in on lo0 all keep state
pass out on lo0 all keep state
pass in on rl0 proto icmp all
pass out on rl0 proto icmp all
pass quick on dc0 all
pass in on rl0 proto tcp from any to any port = www
pass in on rl0 proto tcp from any to any port = netbios-ssn
pass in on rl0 proto udp from any to any port = domain
block drop out log on rl0 inet from 192.168.2.33 to 192.168.3.1
block drop in log on rl0 inet from 192.168.3.1 to 192.168.2.33 port = netbios-ssn
```

Figura 7.14 Set activo de reglas después del ataque

Finalmente hay que revisar las bitácoras en línea del Firewall para visualizar los paquetes que se están rechazando por las reglas creadas a partir del ataque. Una vez que el atacante manda el ataque queda bloqueada toda comunicación del interior al exterior. En la figura 7.15 se ven los paquetes bloqueados por las reglas generadas.

```
-bash-3.1# tcpdump -n -e -ttt -i pflog0
tcpdump: WARNING: pflog0: no IPv4 address assigned
tcpdump: listening on pflog0, link-type PFLOG
May 23 06:27:00.690321 rule 10/(match) block in on rl0: 192.168.3.1.1313 > 192.168.2.33.139: [|tcp] (DF)
May 23 06:27:03.612488 rule 10/(match) block in on rl0: 192.168.3.1.1313 > 192.168.2.33.139: [|tcp] (DF)
May 23 06:27:04.623461 rule 9/(match) block out on rl0: 192.168.2.33.80 > 192.168.3.1.1151: [|tcp] (DF)
May 23 06:27:09.647833 rule 10/(match) block in on rl0: 192.168.3.1.1313 > 192.168.2.33.139: [|tcp] (DF)
May 23 06:27:21.618338 rule 10/(match) block in on rl0: 192.168.3.1.1313 > 192.168.2.33.139: [|tcp] (DF)
May 23 06:27:35.516502 rule 9/(match) block out on rl0: 192.168.2.33.80 > 192.168.3.1.1315: [|tcp] (DF)
May 23 06:27:38.521202 rule 9/(match) block out on rl0: 192.168.2.33.80 > 192.168.3.1.1315: [|tcp] (DF)
May 23 06:27:44.537213 rule 9/(match) block out on rl0: 192.168.2.33.80 > 192.168.3.1.1315: [|tcp] (DF)
May 23 06:35:07.344018 rule 9/(match) block out on rl0: 192.168.2.33.139 > 192.168.3.1.1289: [|tcp] (DF)
```

Figura 7.15 Bitácora del Firewall

Conclusiones

La utilización OpenBSD y Snort permitió implementar un perímetro de seguridad con herramientas de distribución libre. Además OpenBSD y Snort son proyectos que cuentan con una amplia documentación con lo cual se desarrolló una herramienta que permite integrar éstos dos proyectos en un sistema de prevención de intrusos.

El software se desarrolló de manera que no es necesario reiniciar el Firewall cada vez que sea necesario agregar una nueva regla de bloqueo.

La aplicación se desarrolló utilizando el lenguaje C ya que éste es un lenguaje de alto nivel y al mismo tiempo un lenguaje compilado que permite a la aplicación trabajar en ambiente de producción de alto rendimiento.

Durante el desarrollo de la aplicación se detectó que es posible su implementación bajo arquitectura cliente-servidor por lo cual la herramienta se desarrollo en dos partes, la parte del cliente y la parte del servidor.

También se observo la oportunidad de implementar la inspección de distintos protocolo de comunicación por lo que se desarrolló la inspección de paquetes para los protocolos TCP, UDP e ICMP.

Además se vio que era necesario desarrollar el programa de tal manera que se verificara que no se repitieran reglas por lo que se agregó la funcionalidad para evitar la duplicidad de reglas.

Para llevar a cabo la pruebas de rendimiento se implemento una maqueta con todos los paquetes necesarios, se buscaron vulnerabilidades en diversos paquetes de software y se implementó una intrusión con base en la metodología general de ataque.

Dentro de los aspectos que es necesario implementar para las siguientes versiones se encuentra incluir una lista blanca cuyo objetivo sea definir un conjunto de direcciones IP confiables que permitan el paso por el IPS sin inspección, evitando de esta manera la negación de servicios a las direcciones IP confiables.

También se visualiza como un siguiente paso desarrollar la funcionalidad que permita comparar reglas nuevas con una política general para evitar que se agreguen reglas innecesariamente.

Además se requiere ampliar la funcionalidad para que la aplicación sea capaz de almacenar las tramas o paquetes de red identificados como un posible ataque y que cada vez que esto suceda se envíe una notificación a los administradores del sistema,

Por último es necesario que el programa escriba la configuración del Firewall en un archivo cada vez que se genere un cambio.

Bibliografía

1. Academia de Networking de Cisco Systems, Guía del primer año CCNA 1 y 3. Cisco Press.

2. Cisco ASA: All-in-One Firewall, IPS, and VPN Adaptive Security Appliance. Jazib Frahim - CCIE No. 5459, Omar Santos.Cisco Press 2005.

3. CISSP Exam Cram™ 2. Michael Gregg. Que 2005.

4. Introduction to Computer Security. Matt Bishop. Prentice Hall PTR 2004.

5. Intrusion Detection & Prevention. Carl Endorf, Eugene Schultz y Jim Mellander. McGraw-Hill 2004

6. Managing Security with Snort and IDS Tools. Christopher Gerg y Kerry J. Cox. O'Reilly

7. Mastering FreeBSD and OpenBSD Security. Yanek Korff, Paco Hope y Bruce Potter. O'Reilly

8. Official (ISC)2 Guide to the CISSP Exam. Susan Hansche, John Berti y Chris Hare.

9. Penetration Testing and Network Defense. Andrew Whitaker, Daniel P. Newman. Cisco Press 2005.

10. Snort Cookbook. Jacob Babbin, Simon Biles, Angela D. Orebaugh. O'Reilly 2005

11. The Tao of Network Security Monitoring Beyond Intrusion Detection Richard Bejtlich. Addison Wesley 2004.

12. UNIX® Network Programming Volume 1, Third Edition: The Sockets Networking API. W. Richard Stevens, Bill Fenner, Andrew M. Rudoff. Addison Wesley

Mesografía

Wikipedia “Subred”
<http://es.wikipedia.org/wiki/Subred>

Wikipedia “Línea de comandos”
<http://es.wikipedia.org/wiki/Int%C3%A9rprete_de_comandos>

Wikipedia “Desarrollo por etapas”
<http://es.wikipedia.org/wiki/Desarrollo_por_etapas>

Wikipedia “Desarrollo en espiral”
<http://es.wikipedia.org/wiki/Desarrollo_en_espiral>

Wikipedia “Modelo en cascada”
<http://es.wikipedia.org/wiki/Modelo_en_cascada>

Wikipedia “Modelo de prototipos”
<http://es.wikipedia.org/wiki/Modelo_de_prototipos>

Wikipedia “HIDS”
<http://es.wikipedia.org/wiki/HIDS>

Wikipedia “website defacement”
<http://en.wikipedia.org/wiki/Website_defacement>

Wikipedia “Protocol-based intrusion detection system”
<http://en.wikipedia.org/wiki/Protocol-based_intrusion_detection_system>

Wikipedia “Network intrusion detection system”
<http://en.wikipedia.org/wiki/Network_intrusion_detection_system>

Wikipedia “Host-based intrusion detection system”
http://en.wikipedia.org/wiki/Host-based_intrusion_detection_system

Diccionario Real Academia Española “Seguridad”
<http://buscon.rae.es/draeI/>

SecurityFocus “One of These Things is not Like the Others: The State of Anomaly Detection”
<http://www.securityfocus.com/infocus/1600>

SecurityFocus “Evading NIDS, revisited”
<http://www.securityfocus.com/infocus/1852>

IETF “RFC 147”
<http://tools.ietf.org/html/rfc147>

OpenBSD “OpenBSd”
<http://www.openbsd.org/faq/faq4.html>

Snort “Snort”
<http://www.Snort.org/docs/#docs>

Nmap “nmap”
<http://insecure.org/nmap/>

nbaudit “NetBIOS Auditing Tool / Security Kit”
<http://dir.filewatcher.com/d/OpenBSD/3.5/m68k/nbaudit-1.0.tgz.142255.html>

CVE “Common Vulnerabilities and Exposures”
<http://cve.mitre.org/>

Metaesploit “Metasploit”
<http://www.metasploit.com/>

Glosario

Activos: algo de valor para la empresa. Los activos de tecnología de la información son la combinación de activos lógicos y físicos, y son agrupados en clases específicas (información, sistemas, software, hardware, personas).

Amenaza: indicación de un posible acontecimiento indeseable. Se refiere a una situación en la que una persona puede hacer algo indeseable (un atacante inicia un ataque de denegación de servicio contra el servidor de correo electrónico de una organización) o un fenómeno natural que podría provocar un resultado no deseado (un incendio estropeando el hardware de tecnología de la información de una organización). Las amenazas tienen propiedades definidas (activo, actor, motivo, acceso, salida).

Appletalk: es un conjunto de protocolos desarrollados por Apple para redes de computadoras actualmente es obsoletos y se utilizan los protocolos TCP/IP.

ARIN (*American Registry for Internet Numbers*): se utilizan para identificar el dueño de una dirección IP o un dominio de Internet. http://www.arin.net/index.shtml

ARP (*Address Resolution Protocol*): es un protocolo de nivel de red responsable de encontrar la dirección hardware (Ethernet MAC) que corresponde a una determinada dirección IP.

Ataque: se refiere a la intención de una persona de causar daño o robar información de un sistema informático.

Auditoría: se refiere al registro de los accesos de los usuarios o las acciones que éstos ejecutan.

Autenticación: es el área de la seguridad en cómputo donde se definen los métodos de validación de usuarios.

Autorización: se refiere a las acciones que tienen permitidas los usuarios autorizados por un medio de autenticación.

Broadcast: es un modo de transmisión de información donde un nodo emisor envía información a una multitud de nodos receptores de manera simultánea, sin necesidad de reproducir la misma transmisión nodo por nodo.

Buffer overflow: es un error de software que se produce cuando se copia una cantidad de datos sobre un área que no es lo suficientemente grande para contenerlos, sobrescribiendo de esta manera otras zonas de memoria.

CIDR (*Classless Inter-Domain Routing*): es un estándar de red para la interpretación de direcciones IP. Facilita el encaminamiento al permitir agrupar bloques de direcciones en una sola entrada de tabla de rutas. Estos grupos, llamados comúnmente Bloques CIDR, comparten una misma secuencia inicial de bits en la representación binaria de sus direcciones IP.

Confidencialidad: un sistema posee la propiedad de confidencialidad si la información manipulada por éste no es disponible ni puesta en descubierto para usuarios, entidades o procesos no autorizados.

Control de acceso: se refiere a los privilegios que tienen los usuarios y equipos autorizados cuando interactúan con la red y los sistemas

Criptografía: es la ciencia de cifrar y descifrar información utilizando técnicas que hagan posible el intercambio de mensajes de manera segura que sólo puedan ser leídos por las entidades a quienes van dirigidos.

CVE (*Common Vulnerabilities and Exposures*): es una notación estándar que básicamente consiste en definir de forma común a las vulnerabilidades independientemente del tipo o sistema operativo de que se trate.

Disponibilidad: un sistema posee la propiedad de disponibilidad si, la información es accesible (está disponible) en el momento en que así lo deseen los usuarios, entidades o procesos autorizados.

DMZ (*Demilitarized Zone*): es una red local (una subred) que se ubica entre la red interna de una organización y una red externa, generalmente Internet.

DNS (*Domain Name System*): es una base de datos distribuida y jerárquica que almacena información asociada a nombres de dominio en redes como Internet.

Ethernet: es un estándar de redes LAN con transmisión en topología de BUS y transmisión de 10 Mbps, con capacidad de 1,024 nodos.

Firewall: es un elemento de hardware o software utilizado en una red de computadoras para controlar las comunicaciones, permitiéndolas o prohibiéndolas según las políticas de red que haya definido la organización responsable de la red.

Frame Relay: es una técnica de comunicación mediante retransmisión de tramas, introducida por la ITU-T a partir de la recomendación I.122 de 1988. Ofrece mayores velocidades y rendimiento, a la vez que provee la eficiencia de ancho de banda que viene como resultado de los múltiples circuitos virtuales que comparten un puerto de una sola línea.

FTP (*File Transfer Protocol*): protocolo de transferencia de archivos basado en la arquitectura cliente servidor.

GPL (*GNU General Public License*): es una licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software.

Host: se refiere a un equipo personal de cómputo, un servidor o una estación de trabajo.

HIDS (*Host-based Intrusion Detection System*): el IDS basado en host implican utilizar programas instalados en los sistemas que se requiere sean monitoreados.

HTTP (*Hypertext Transfer Protocol*): el protocolo de transferencia de hipertexto es el protocolo usado en cada transacción de la Web (WWW).

IANA (*Internet Assigned Numbers Authority*): es la Agencia de Asignación de Números de Internet. Era el antiguo registro central de los protocolos Internet, como puertos, números de protocolo y empresa, opciones y códigos. Fue sustituido en 1998 por ICANN. www.iana.org

IDS (*Intrusion Detection System*): es un programa usado para detectar accesos no autorizados a una computadora o a una red.

ICMP (*Internet Control Message Protocol*): es el subprotocolo de control y notificación de errores del Protocolo de Internet (IP). Como tal, se usa para enviar mensajes de error, indicando por ejemplo que un servicio determinado no está disponible o que un router o host no puede ser localizado.

Integridad: un sistema posee la propiedad de integridad si los datos manipulados por éste no son alterados o destruidos por usuarios, entidades o procesos no autorizados.

Internet: es un método de interconexión descentralizada de redes de computadoras implementado en un conjunto de protocolos denominado TCP/IP que garantiza que redes físicas heterogéneas funcionen como una red lógica única, de alcance mundial.

IP (Internet Protocol): es un protocolo no orientado a conexión usado tanto por el origen como por el destino para la comunicación de datos a través de una red de paquetes conmutados.

IPS (*Intrution Prevension System*): un IPS es un dispositivo de seguridad que ejecuta el control de acceso para proteger a un sistema o una red. La tecnología de prevención de intrusos es considerada por algunos como una extensión de la tecnología de detección de intrusos, IDS, sin embargo actualmente es una forma más de establecer el control de acceso.

IPv6: es la versión 6 del Protocolo de Internet (*Internet Protocol*), un estándar en desarrollo del nivel de red encargado de dirigir y encaminar los paquetes a través de una red.

ISO (*International Organization for Standardization*): Organización Internacional de Estandarización, es una red de institutos de estandarización que integra a más de 157 países cuya sede central se encuentra en Genova, Suiza.

Licencia BSD: es la licencia de software otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*). Pertenece al grupo de licencias de software Libre.

Localhost: en el contexto de redes TCP/IP, localhost es un nombre reservado que tienen todas las computadoras, router o dispositivo que disponga de una tarjeta de red ethernet para referirse a sí mismo.

MAC (*Media Access Control*): es un identificador de 48 bits que corresponde de forma única a una tarjeta o interfaz de red.

NetBios (*Network Basic Input/Output System*): es una especificación de interfaz para acceso a servicios de red, originalmente desarrollado por IBM y Sytek.

NIDS (*Network Intrusion Detection System*): IDS de red que se encargan de los datos que fluyen a través de los cables que conectan al host con el exterior

NFS (*Network File System*): es un protocolo de nivel de aplicación. Es utilizado para sistemas de archivos distribuido en un entorno de red de computadoras de área local.

OpenBSD: es un sistema operativo libre tipo Unix, multiplataforma, basado en 4.4BSD. Es un descendiente de NetBSD, con un foco especial en la seguridad y la criptografía.

OSI (*Open System Interconnection*): modelo de referencia de interconexión de sistemas abiertos, lanzado en 1984 fue el modelo de red descriptivo creado por ISO; esto es, un marco de referencia para la definición de arquitecturas de interconexión de sistemas de comunicaciones.

Políticas de seguridad: serie de reglas que todos los directores, administrativos, jefes, empleados y usuarios de una organización deben de cumplir para mantener el estado de seguridad, las políticas de seguridad deben ser difundidas y publicadas.

PPP (Protocolo Punto a Punto): permite establecer una comunicación a nivel de enlace entre dos computadoras. Generalmente, se utiliza para establecer la conexión a Internet de un particular con su proveedor de acceso a través de un módem telefónico.

Proxy: es un programa o dispositivo que realiza una acción en representación de otro. La finalidad más habitual es la de servidor proxy, que sirve para permitir el acceso a Internet a todos los equipos de una organización cuando sólo se puede disponer de un único equipo conectado, esto es, una única dirección IP.

RAM (*Random Access Memory*): se compone de uno o más chips y se utiliza como memoria de trabajo para programas y datos. Es un tipo de memoria temporal que pierde sus datos cuando se queda sin energía (por ejemplo, al apagar la computadora), por lo cual es una memoria volátil.

RARP (*Reverse Address Resolution Protocol*): es un protocolo utilizado para resolver la dirección IP de una dirección hardware dada (como una dirección Ethernet).

Riesgo: es la probabilidad de que una vulnerabilidad sea explotada de acuerdo a su nivel de exposición y al peligro involucrado

Ruteador: es un dispositivo de hardware para interconexión de red de computadoras que opera en la capa tres (nivel de red). Este dispositivo permite asegurar el enrutamiento de paquetes entre redes o determinar la ruta que debe tomar el paquete de datos.

Shell: es un programa informático que actúa como interfaz de usuario para comunicar al usuario con el sistema operativo mediante una ventana que espera comandos textuales ingresados por el usuario en el teclado, los interpreta y los entrega al sistema operativo para su ejecución.

SQL Injection: es una vulnerabilidad informática en el nivel de la validación de las entradas a la base de datos de una aplicación.

SMTP (*Simple Mail Transfer Protocol*): protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico entre computadoras o distintos dispositivos.

Sniffer: es una pieza de software que permite visualizar de alguna manera el tráfico que pasa por una interfaz de red. Es decir todo lo que en el medio físico de red este pasando.

SNMP (*Simple Network Management Protocol*): es un protocolo de la capa de aplicación que facilita el intercambio de información de administración entre dispositivos de red.

SSH (*Secure Shell*): es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a hosts remotos a través de una red.

SSL (*Secure Sockets Layer*): protocolo criptográfico que proporciona comunicaciones seguras en Internet capaz de autenticar a los servidores y a los clientes en una comunicación a través de Internet.

Socket: es un ente abstracto por el cual dos programas se van a poder comunicar y transmitir un flujo de datos de manera ordenada y confiable utilizando el modelo cliente-servidor

Telnet (*TELecommunication NETwork*): es el nombre de un protocolo de red (y del programa informático que implementa el cliente), que sirve para acceder mediante una red a otra máquina.

TCP (*Transmission Control Protocol*): es un protocolo de comunicación orientado a conexión y fiable del nivel de transporte, actualmente documentado por IETF RFC 793.

TLS (*Transport Layer Security*) Protocolo criptográfico que proporcionan comunicaciones seguras en Internet capaz de autenticar a los servidores y a los cliente en una comunicación a través de Internet.

Throughput: cantidad de datos por unidad de tiempo que pasan a través de un dispositivo lógico o físico, como una interfaz de red. También puede ser la cantidad de tráfico que pasa por un nodo de red.

UDP (*User Datagram Protocol*): es un protocolo del nivel de transporte basado en el intercambio de datagramas. Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión.

VPN (*Virtual Private Network*): es una tecnología de red que permite una extensión de la red local sobre una red pública o no controlada, como por ejemplo Internet.

Vulnerabilidad: es una debilidad de software, hardware o de procedimientos que puede llegar a causar que un atacante pueda abrir una puerta hacia el interior de una red o sistema.
Web defacement: es cuando un atacante se introduce a un servidor Web y sustituye el contenido original por otro, cuyo principal objetivo es causar daño a la imagen del sitio Web. También puede utilizarse

para distraer a los administradores del sistema mientras se causa más daño a otros servidores de la organización.

Whois: es un protocolo TCP basado en preguntas/repuestas que es usado para consultar una base de datos para determinar el propietario de un nombre de dominio o una dirección IP en Internet. http://www.whois.net/