UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
INSTALACIÓN DE HONEYNETS DE SEGUNDA GENERACION BAJO OPENBSD

Jose Alberto Avalos Velez

10 de agosto de 2009

DEDICADA A LA MEMORIA DE MI TAN QUERIDA TIA GUADALUPE QUIEN SIEMPRE ME IMPULSO HACIA ADELANTE Y ME DIO SU CARIÑO INCONDICIONAL

Agradecimientos

Agradezco a Dios por haberme dado la fuerza, paciencia y sabiduria para terminar mis estudios en ingeniería.

Agradezco a mis padres y mi tia tan queridos por la enorme oportunidad, el apoyo y el gran impulso que me brindaron para poder estudiar y superarme.

A mis hermanos por el inmenso apoyo que me brindaron durante toda mi vida escolar y por confiar en mi.

A Lissette por haberme permitido ser parte de tu vida y por haber dedicado muchos momentos e ideas durante la elaboración de esta tesis.

Agradezco de una manera muy especial a Ximena, por tu invaluable apoyo en los tramites de esta tesis y por siempre estar ahí y al pendiente.

A mis amigos tan queridos Enrique, Laida, Fabian, Rober, Emmanuel, Javo, Nacho, Mena por siempre estar al pendiente y apoyarme en cualquier situación.

A la Facutlad de Ingeniería de la UNAM, por la excelente educación que me dió y por las gratas experiencias que viví durante mi estancia en ella, así como por tener el privilegio de tomar clases con profesores excelentes.

A mi director de Tesis Ing. Fernando Javier Martínez Mendoza por creer en este proyecto de tesis y compartir conmigo un poco de todo su conocimiento.

Al Instituto de Física de la UNAM, así como a sus directivos en especial a: Lic. Neptali González Gómez, Lic. Alejandro Juárez Robles, Ing. Fernando Javier Martínez Mendoza por todo los consejos (personales y profesionales) y apoyo brindado.

Y en general a toas aquellas personas que creyeron en mí y que siguieron muy de cerca este logro en mi vida.

Introducción

Hoy en día uno de los problemas que más preocupa a los administradores de redes y seguridad de la información es el incremento en las actividades ilícitas y los ataques informáticos.

Actualmente una institución que presnta dicho problema es el Instituto de Física de la UNAM, ya que en sus sistemas se han detectado diversos intentos de ataques que atentan contra el activo más importante de la institución, su información.

Ante dicha situación se ha desarrollado este proyecto, el cual tiene el objetivo de crear una *Honeynet* (red trampa) para poder identificar a detalle los patrones de ataque empleados por los intrusos, cual es su comportamiento y finalmente cuales son sus motivos para atacar a los sistemas.

Se ha decidido crear una *Honeynet de Segunda Generación*, debido a que en este tipo de redes cualquier servicio que se vaya a ofrecer hacia *Internet* es real,es decir, que se crearán servicios como un sistema de correo, un sistema de bases de datos y un sistema de servicio *web* totalmente reales y funcionales pero con la caracter'sitica que serán monitoreados por el administrador de la *Honeyent* además de que toda actividad será grabada para su análisis posterior. A diferencia de las *Honeynets de primera generación* donde todos los servicios son simulados mediante el uso de software especializado para este fin.

Por otro lado se ha empleado *OpenBSD* como sistema principal de la *Honeynet* denominado *Honeywall*. El *Honeywall* es el sistema encargado de administrar a la *Honeynet* en cuanto al numero de conexiones salientes y entrantes desde y hacia la*Honeynet*, por lo que este sistema será el principal responsable de que un intruso no usea la *Honeynet* como plataforma para atacar a otros sistemas que se encuentren fuera del Intituto. Debido a la gran responsabilidad que tiene este sistema requiere de una seguridad sólida para protegerse así mismo y a toda la *Honeynet*, por lo que se implento bajo *OpenBSD* uno de los sistemas tipo *Unix* más seguros en el mundo.

Índice general

1. Conceptos Generales **1**
1.1. Redes de computadoras . . . 1
1.1.1. Definición . . . 1
1.1.2. Tipos de redes . . . 1
1.1.3. Topologías . . . 2
1.1.4. Protocolos de red . . . 6
1.1.5. Hardware en redes . . . 9
1.1.6. Seguridad . . . 11
1.2. Servicios de la seguridad . . . 13
1.2.1. Panorama de la seguridad . . . 13
1.3. OpenBSD . . . 15
1.3.1. Antecedentes . . . 15
1.4. Sistema de detección de intrusos . . . 18
1.5. Sniffer . . . 20
1.5.1. Sistema de bitácoras . . . 21
1.6. Tcpdump . . . 22

2. Honeynets **23**
2.1. Honeypots . . . 23
2.1.1. Definición de Honeypots . . . 23
2.1.2. Tipos de Honeypots . . . 23
2.1.3. Uso de los Honeypots . . . 24
2.2. Honeynets . . . 25
2.2.1. Definición de Honeynet . . . 25
2.3. Honeynets de primera generación . . . 26
2.3.1. Arquitectura de Honeynets de primera generación . . . 26
2.3.2. Instalación y configuración de un Firewall para el control de datos primario . . . 30
2.3.3. Instalación y configuración de un IDS para la captura de datos primario . . . 30
2.3.4. Preparando y conectando la Honeynet a Internet . . . 30
2.4. Honeynets de segunda generación . . . 31
2.5. Control y captura de datos en Honeynets de segunda generación . . . 32
2.6. Honeynets virtuales . . . 34
2.7. Honeynets virtuales híbridas . . . 35
2.8. Honeynets virtuales auto-contenidas . . . 36
2.9. Honeynets sobre VMWARE . . . 37
2.10. Honeynets sobre GSX Server . . . 37
2.11. Honeynets sobre User-mode-Linux . . . 38
2.12. Honeynets distribuidas . . . 39
2.12.1. Granjas de Honeypots . . . 40

3. Instalación de la Honeynet **42**
3.1. Instalación y configuración del servidor de correo . . . 42
3.1.1. Instalación de paquetes necesarios para un servidor de correo . . . 42
3.1.2. Configuración de Sendmail . . . 44
3.1.3. Configuración de dovecot . . . 45
3.1.4. Configuración de sendmail.mc (funciones de sendmail) . . . 47
3.1.5. Squirrelmail . . . 47
3.1.6. Instalación de Sebek cliente sobre Fedora Core . . . 51
3.2. Instalación y configuración del Servidor de Base de Datos . . . 52
3.2.1. Instalación de paquetes para el Servidor de Base de Datos . . . 52
3.2.2. Instalación de Sebek-Cliente sobre Ubuntu . . . 53
3.3. Configuración de Sebek cliente en Ubuntu . . . 55
3.4. Instalación de Sebek cliente en Windows 2000 . . . 57
3.5. Honeywall sobre OpenBSD . . . 59
3.5.1. Instalación y configuración de OpenBSD . . . 60
3.5.2. Configuración de OpenBSD en modo bridge . . . 61
3.6. Sistema Detector de Intrusos . . . 63
3.6.1. Instalación de IDS Snort . . . 63
3.6.2. Configuración de Snort . . . 65
3.6.3. Configuración de Mysql . . . 66
3.7. Sebek Server . . . 69
3.7.1. Instalación de Sebek Server . . . 70

4. Automatización del control de conexiones **72**
4.1. Estructura de la automatización . . . 72
4.1.1. Módulo de instalación . . . 73
4.1.2. Módulo: Administración del Honeywall . . . 74
4.1.3. Módulo: Honeypots . . . 76
4.1.4. Módulo: Control de conexiones . . . 77
4.1.5. Módulo: Ejecutar programa . . . 79
4.1.6. Módulo: Rollback . . . 80

Pruebas y Resultados **81**

Conclusiones Generales **83**

Bibliografa **84**

A. Código fuente para automatizar el control de datos y de conexiones de una *Honeynet* de segunda generación bajo OpenBSD **85**

Índice de figuras

1.1. Topología en Estrella . . . 2
1.2. Topología en Malla . . . 3
1.3. Topología en BUS . . . 4
1.4. Topología en Anillo . . . 4
1.5. Topología en Arbol . . . 5
1.6. Arquitectura de STP . . . 10
1.7. Firewall con servicio de NAT . . . 17
1.8. Firewall con servicio Bridge . . . 18
1.9. Arquitectura de Balanceador de Cargas . . . 18
1.10. Arquitectura de un NIDS . . . 19
1.11. Arquitectura de un HIDS . . . 19

2.1. Arquitectura de Honeynet de generación I . . . 27
2.2. Arquitectura de Honeynet de generación II . . . 31
2.3. Arquitectura de Honeynet Router . . . 33
2.4. Arquitectura de un Honeywall . . . 34
2.5. Honeynet Híbrida . . . 35
2.6. Honeynet Autocontenida . . . 36
2.7. Honeynets distribuidas . . . 39
2.8. Granja de Honeypots . . . 40

3.1. Selección de paquetes para Sendmail . . . 43
3.2. Selección de paquetes para Sendmail . . . 43
3.3. Sesion de Telnet SMTP . . . 44
3.4. Autenticación con AUTHSASL . . . 45
3.5. Control de accesos . . . 45
3.6. Activación de los servicios de POP3 e IMAP . . . 46
3.7. Arranque del sistema con Dovecot . . . 46
3.8. Reinicio de Sendmail con Dovecot . . . 46
3.9. Análisis de maillog . . . 47
3.10. Interfaz de configuración de Squirrelmail . . . 48
3.11. Interfaz de configuración de Squirrelmail . . . 49
3.12. Activación del dominio de correo . . . 49
3.13. Activación de plug-ins de Squirrelmail . . . 50
3.14. Acivación del servicio con Squirrelmail . . . 50
3.15. Arranque del servicio web . . . 51
3.16. Arranque de Squirrelmail vía Web . . . 51
3.17. Obtención de código por SVN . . . 54
3.18. Instalación de Sebek sobre Windows 2000 . . . 57
3.19. Instalación de Sebek sobre Windows 2000 . . . 58
3.20. Instalación de Sebek sobre Windows 2000 . . . 58
3.21. Configuración de Sebek sobre Windows 2000 . . . 59
3.22. Configuración de Sebek sobre Windows 2000 . . . 59
3.23. Configuración de las interfaces de red en OpenBSD . . . 60
3.24. Selección de paquetes para el Honeywall . . . 60

3.25. Proceso de instalación de paquetes del Honeywall 61
3.26. Instalación finalizada del Honeywall 61
3.27. Arranque del bridge en el Honeywall 62
3.28. Reglas básicas para OpenBSD modo puente 62
3.29. Configuración de BASE 68
3.30. Configuración de BASE 68
3.31. Instalación de BASE 69
3.32. Instalación de BASE 69
3.33. Captura de actividades de un intruso 71

4.1. Módulos principales 73
4.2. Menú Principal 73
4.3. Administración del Honeywall 75
4.4. Estatus del Bridge 76
4.5. Datos obtenidos de la Base de Datos 77
4.6. Proceso de Rollback 80

Capítulo 1

Conceptos Generales

1.1. Redes de computadoras

1.1.1. Definición

Una red de computadoras o también llamada red informaítica se puede definir como *el conjunto de computadoras y/o dispositivos conectados por medios físicos e inalambricos.*

Hoy en día los sistemas electrónicos e informáticos se han convertido en una de las herramientas más importantes para el ser humano ya que le permite realizar la mayoría de sus actividades cotidianas como: cálculos matemáticos, financieros, estadísticos, publicidad, negocios, transferencias bancarias y de información, entre otras.

Todas estas actividades que se realizan día a día generan un gran volumen de información, la cual es el recurso más importante dentro de dichas actividades. Debido a esto se le debe de brindar la mayor seguridad posible para así garantizar la **Confidencialidad, Integridad y Disponibilidad** de la misma.

En este sentido las redes de computadoras juegan un papel muy importante, ya que mediante una red de computadoras podemos realizar todas y cada una de las actividades antes menciondas sin límite alguno, es decir, las podemos hacer de manera local y mundial, no importando la ubicación geográfica ni el horario.

1.1.2. Tipos de redes

Redes públicas: Este tipo de redes pueden ser usadas por cualquier persona para compartir información. Permite comunicar a cualquier usuario sin importar su ubicación geográfica.

Redes privadas: Estas redes tienen el propósito de brindar servicio sólo a un conjunto de computadoras en específico, además de contar con un control de acceso personal. Por lo regular estas redes son implementadas en las organizaciones, instituciones,etc. Con el fin de que sólo personal autorizado pueda tener acceso a los o recursos de la organización y/o institución y así realizar sus actividades.

Redes de Área Personal Inalámbricas: Es una red de computadoras que se emplean para la comunicación de diversos dispositivos que se encuentren cercanos a un punto de acceso como: *Access Point, PDAs, Teléfonos celulares, impresoras.* Regularmente este tipo de redes abarcan muy pocos metros debido a que la señal depende de la distancia que exista entre el dispositivo y el punto de acceso.

Redes de Área Local: Significa la interconexión de varias computadoras bajo un mismo ambiente o lugar, su extensión es limitada ya que la máxima distancia entre una computadora y otra es de 100m, ya que si se sobre pasa esta distancia se provoca la interferencia electromagnética en el cableado perdiendo así la comunicación entre los equipos.

Redes de Área Metropolitana: Estas son redes de alta velocidad de transmisión y que dan cobertura a una área geográfica muy extensa. Los medios de transmisión empleados son fibra óptica y par trenzado. Estas redes son muy estables y las velocidades que a las que opera son de 10Mbps, 20Mbps, 45Mbps y 75Mbps sobre pares de cobre y de 100Mbps, 1Gbps, 10Gbps mediante fibra óptica.

Redes de Área Amplia: Son redes de computadoras capaces de dar cobertura en distancias de 100Km hasta 1000Km ofreciendo la posibilidad de intercomunicar a un país o continente entero. Un ejemplo de este tipo de red es *Internet*. Las redes WAN son empleadas por empresas particulares a nivel mundial y por los proveedores de servicios de *Internet* para dar servicio a sus clientes.

1.1.3. Topologías

Una topología se puede definir como la disposición física y lógica de una red de computadoras, apoyándose en la combinación de estándares y protocolos. La topología de una red únicamente determina la configuración de las conexiones entre los nodos, la distancia, la tasa de transferencia de datos, entre otras configuraciones. A continuación se describen las diversas topología de red.

Estrella

Esta topología reduce la posibilidad de fallo de red, conectando todos los nodos de una red a un nodo central. Los dispositivos adecuados para este tipo de topología son los *switch*[1].Estos dispositivos reenvían todas las transmisiones recibidas de cualquier nodo periférico de la red, en algunas ocasiones las reenvía al mismo nodo que la envio En la figura 1.1 se muestra una red en estrella.

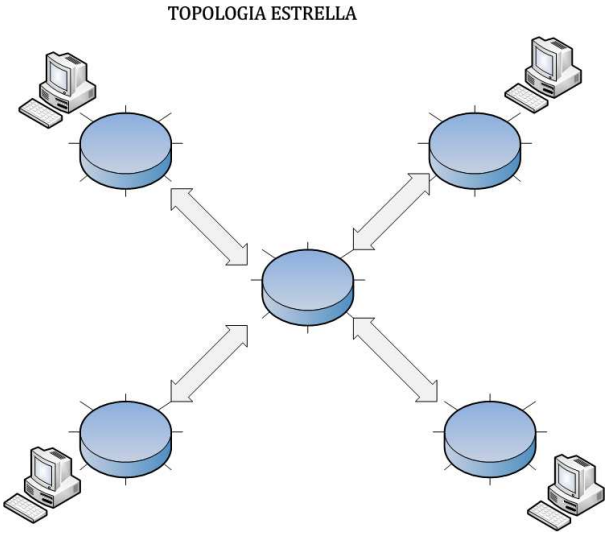

Figura 1.1: Topología en Estrella

[1]es un dispositivo electrónico de interconexión de redes de computadoras que opera a nivel de enlace de datos (capa 2) del modelo OSI

Malla

En una topología en malla, cada dispositivo tiene un enlace punto a punto y dedicado con cualquier otro dispositivo. El término dedicado significa que el enlace conduce el tráfico únicamente entre los dos dispositivos que conecta.Por tanto, una red en malla completamente conectada necesita *n(n-1)/2* canales físicos para enlazar n dispositivos. Para acomodar tantos enlaces, cada dispositivo de la red debe tener sus puertos de entrada/salida (E/S).

Una malla ofrece varias ventajas sobre otras topologías de red. En primer lugar, el uso de los enlaces dedicados garantiza que cada conexión sólo debe transportar la carga de datos propia de los dispositivos conectados, eliminando el problema que surge cuando los enlaces son compartidos por varios dispositivos. En segundo lugar, una topología en malla es robusta, es decir, si un enlace falla, no inhabilita todo el sistema.

Otra ventaja que brinda esta topología es la seguridad, ya que cuando un mensaje viaja a través de una línea dedicada, solamente lo ve el receptor adecuado. La figura 1.2 describe esta topología.

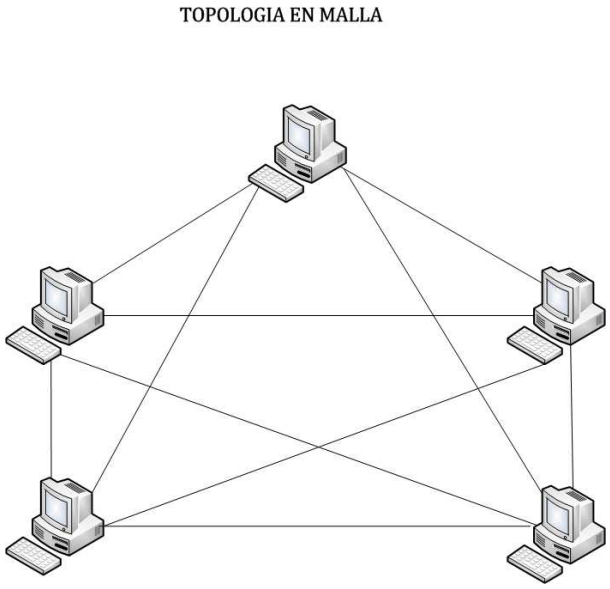

Figura 1.2: Topología en Malla

Bus

Esta topología se caracterízan por que todas las estaciones de trabajo están conectadas a un único canal de comunicaciones. Además una topología de este tipo permite que todos los dispositivos de una red puedan ver todas las señales de todos los demás dispositivos, lo que de alguna manera puede ser ventajoso, si es que se desea compartir la información. Por otro lado representa una desventaja ya que producen problemas de tráfico y colisiones de datos. En la figura 1.3 se muestra una red en Bus.

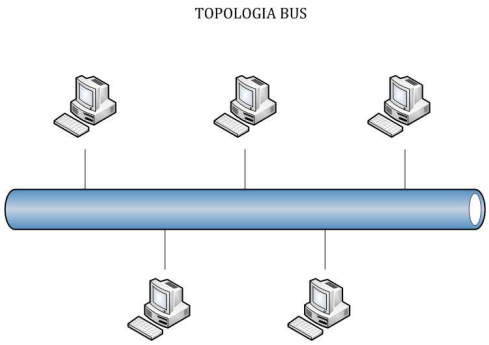

Figura 1.3: Topología en BUS

Anillo

En esta topología las estaciones de trabajo se conectan formando un anillo. Cada estación está conectada a la siguiente y la última a la primera. Cada estación tiene un receptor y un transmisor que hace la función de un repetidor[2], pasando así la señal a la siguiente estación del anillo. La figura 1.4 muestra una red en anillo.

Figura 1.4: Topología en Anillo

Árbol

La topología en árbol es una variante de la de estrella. Como en la estrella, los nodos del árbol están conectados a un concentrador central que controla el tráfico de la red. Aunque no todos los dispositivos se conectan directamente al concentrador central, la mayoría de los dispositivos se conectan a un concentrador secundario que, a su vez, se conecta al concentrador central. En la figura 1.5 se muestra dicha distribución.

[2]Dispositivo electrónico que recibe una señal de bajo nivel y la transmite a un nivel más alto

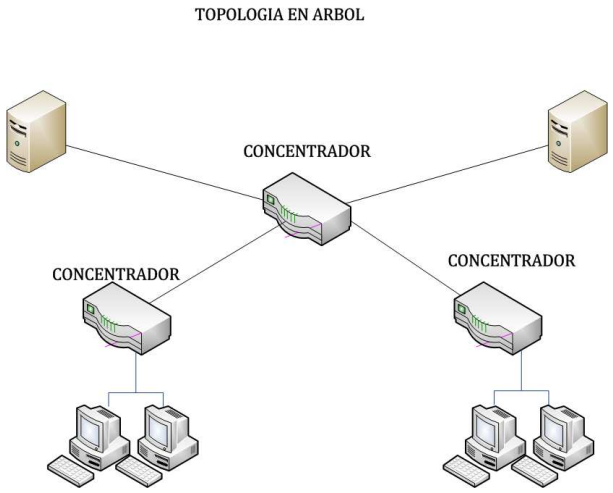

Figura 1.5: Topología en Arbol

Modelo OSI

En el año de 1977 la Organización Internacional de Estándares(ISO por sus siglas en inglés) creó una comisión para organizar los estándares de comunicaciones de datos que establecieran la accesibilidad universal e interoperabilidad entre productos de diferentes fabricantes. Como resultado surge el modelo de Interconexión de Sistemas Abiertos el cual consta de 7 capas en su modelo, dichas capas se describen a continuación:

Capa 1 Física: Esta capa es la que se encarga de las conexiones en un ambiente físico de una computadora hacia la red. Dentro de la capa se consideran dos tipos de medios para la transmisión de datos que son:

- Medios guiados
- Medios no guiados

En los medios guiados se comprende el cable coaxial, de par trenzado, fibra óptica, y otros tipos de cables. Mientras que en los medios no guiados se comprenden lo que es radio, infrarrojos, microondas, láser y otras redes inalámbricas.

Capa 2 Enlace: Esta capa se ocupa del direccionamiento físico, de la topología de la red, así como de su acceso a la misma y la notificación de errores. Dentro de esta capa la transmisión de datos debe ser fiable y sin errores. En este nivel se crean y reconocen los límites de una trama[3] para evitar la saturación de datos en una red.

Capa 3 Red: La función de esta capa es hacer que los datos lleguen desde el origen al destino. Además de controlar la congestión de la red, la cual sucede cuando un nodo llega a una saturación de datos y tira toda la red. Los dispositivos que realizan esta tarea se les conoce como routers.

[3]Unidad de envío de datos. Viene a ser sinónimo de paquete de datos o paquete de red

Capa 4 Transporte: La función de esta capa es aceptar los datos enviados por las capas superiores, fragmentarlos de ser necesario, para posteriormente pasarlos a la capa de red y asegurarse de que lleguen correctamete al otro lado de la comunicación.

Capa 5 Sesión: En esta capa se determinan servicios cruciales para una comunicación como son:

- Control de la sesión a establecer (emisor y receptor)
- Control de la concurrencia (dos cumunicaciones al mismo tiempo)
- Mantener puntos de verificación (para la reanudación de comunicaciones de forma rápida en caso de pérdidad de la misma).

Capa 6 Presentación: Esta capa se realiza la presentación de los datos, es decir, que aunque los equipos tengan difrerentes representaciones internas de caracteres (*ASCII, Unicode, EBCDIC, sonido o imágenes*) lleguen de una forma reconocible.

Capa 7 Aplicación: Esta capa ofrece al usuario la posibilidad de acceder a los servicios de las demás capas, además de definir los protocolos para el intercambio de datos.

1.1.4. Protocolos de red

Desde el principio de las redes los programas de comunicaciones eran capaces de hacer que una computadora se entendiera con otra a través de reglas e instrucciones conocidas como *protocolos*. En un prinicipio estos protocolos debían ser los mismos en ambas computadoras, debido a que una computadora sólo se entendia con otra del mismo fabricante.

Hoy en día es muy común hablar de protocolos en capas y del modelo OSI. Como se mencionaba anteriormente los protocolos están organizados en capas comunmente denominadas *pilas* y cada capa únicamente se comunica con la capa superiro o inferior de la pila, es decir que si los datos van de salida pasan de manera descendente por cada una de las capas, mientras que cada capa tiene la tarea de agrgegar una encabezado a los datos y pasarlo a la capa siguiente en donde se hará la misma tarea hasta que los datos sean transferidos por el medio físico, para el caso de que los datos entren se sigue el mimso proceso pero en sentido inverso, es decir que los datos pasan sobre la pila de protocolos de manera ascendente y en este caso cada capa tiene la tarea de leer el encabezado que le corresponde y va realizando una decapsulación de datos hasta que dichos datos llegan a su destino.
Como ejemplo de lo anterior se tiene que:

capa 7: Encabezado 1 -> DATOS
capa 6: Encabezado 2 -> Encabezado 1 -> DATOS
capa 5: Encabezado 3 -> Encabezado 2 -> Encabezado 1 -> DATOS
capa 4: Encabezado 4 -> Encabezado 3 -> Encabezado 2 -> Encabezado 1 -> DATOS
...
etc.

El modelo OSI ofrece dos grandes ventajas que son:

1. Fácil de implementar
2. Fácil de extender

Por lo que este modelo de comunicaciones permite a cualquier fabricante de hardware una pila de protocolos de manera que su equipo pueda comunicarse con el de otro fabricante sin problema alguno.

Según la clasificación del modelo *OSI*, la comunicación de varios dispositivos *ETD*[4] se puede estudiar dividiéndola en 7 niveles, que van desde el nivel más alto hasta el más bajo. El cuadro 1.1 muestra las capas del modelo OSI.

Cuadro 1.1: Capas del Modelo OSI

Capa	Nivel	Acción
Capa 7	Nivel de Aplicación	Aplicación
Capa 6	Nivel de Presentación	Aplicación
Capa 5	Nivel de Sesión	Aplicación
Capa 4	Nivel de Transporte	Aplicación
Capa 3	Nivel de Red	Transporte de Datos
Capa 2	Nivel de Enlace	Transporte de Datos
Capa 1	Nivel Físico	Transporte de Datos

Estas capas pueden ser subdivididas en capas superiores y capas inferiores. Las primeras 4 capas trabajan con las aplicaciones y las 3 capas restantes con el transporte de datos.

Por otro lado se han desarrollado diferentes familias de protocolos para comunicación en sistemas *Unix*. El más ampliamente utilizado es el *Internet Protocol*, comunmente conocido como *TCP/IP*.

El modelo *TCP/IP* es la base de *internet* que al igual que el modelo OSI sirve para enlazar computadoras que utilizan diferentes sistemas operativos, incluyendo PCs, minicomputadoras y computadoras centrales sobre una red de área local (LAN) o una red de área extensa (WAN).

El modelo *TCP/IP* consta de 4 capas como se muestran en el cuadro 1.2.

Cuadro 1.2: Capas del Modelo TCP/IP

Capa	Nivel	Protocolo
Capa 4	Aplicación	http,ftp,DNS
Capa 3	Transporte	tcp,udp,rtp,rctp
Capa 2	Internet	IP
Capa 1	Físico y enlace	Codificación,T1,E1,Ethernet,TokenRing,PPP

Cada capa cuenta con una interfaz bien definida. Generalmente una capa se comunica con la capa inmediata superior. En cada una de las capas tanto del modelo *OSI* como del modelo *TCP/IP* actúan los protocolos de red de una manera particular.

Es importante aclarar que ambos modelos tienen ciertas diferencias entre sí. Para el caso del modelo TCP/IP no es estrictamente necesario el uso de todas las capas, debido a que este modelo no define los servicios, los protocolos y las interfaces, además de que hay protocolos de apliación que operan directamente sobre IP y otros más que lo hacen por arriba de IP. Mientras que en el caso del modelo OSI el uso de cada capa es estricto, es decir, se debe de llevar una secuencia en el proceso de la comunicación capa por capa, además de que este modelo sí hace la definición de servicios, protcolos e interfaces, para lo cual tiene designada una capa para cada uno de ellos. Por otro lado ambos modelos también poseen similitudes y la más importante es que ambos fueron creados con el fin de que las comunicaciones entre dispositivos no dependa del hardware de dichos dispositivos ni del software en éstos, sino que sea independiente a estos factores y se puedan intercomunicar entre ellos.
Entre los protocolos más reconocidos para el intercambio de datos a nivel de la capa de aplicación son:

[4]Cualquier equipo informático emisor o receptor final de datos

http(Protocolo de transferencia de Hypertexto): Es un protocolo para la navegación Web.

ftp(File transfer Protocol): Protocolo para la transferencia de datos.

smtp(Simple mail Transfer Protocol): Protocolo para la transferencia simple de correo.

pop/imap: Protocolo para el envio de correo a usuarios finales.

ssh: Protocolo para realizar sesiones remotas seguras.

Telnet: Sesiones remotas inseguras.

Por otro lado los protocolos más reconocidos para el transporte de datos son los siguientes:

- ***TCP****(Transfer Control Protocol)*: Este es un protocolo utilizado dentro de una red de datos compuesta por computadoras para crear conexiones entre ellas. Además garantiza que los datos serán entregados a su destinatario ya que es un protocolo orientado a conexión, es decir, que por cada paquete que recibe envía un *ACK*(acuse de recibo), y si existe algún paquete mal formado o corrupto envía una petición hacia el origen de la petición requiriéndole le envíe nuevamente ese paquete.

- ***UDP****(User Datagram Protocol)*: Este protocolo se basa en el envío de datagramas[5] que son enviados mediante la red sin que se haya establecido una conexión previa. Este protocolo no garantiza la entrega de los paquetes a sus destinatarios debido a que es un protocolo no orientado a conexión no envía *ACK*(acuse de recibo), por lo que si un paquete esta mal formado o corrupto lo desecha y no pide por su retransmisión.

Existen además protocolos de enrutamiento que son empleados por los *routers* para comunicarse entre ellos y para compartir información con lo cual les permite tomar la decisión de cual es la ruta más adecuada para el envío de un paquete.

Los protocolos de enrutamiento más utilizados son los siguientes:

- ***RIP****(Router Information Protocol)*: Es un protocolo empleado por los *routers* para funcionar como *gateways* hacia internet, aunque también pueden actuar como equipos con el fin de intercambiar información entre redes *IP*. El protocolo *RIP* hace uso del protocolo *udp* para el envío de sus mensajes y el puerto 520. Por otro lado, el protocolo *RIP* determina cual es el camino más corto hacia la red destino haciendo uso del algoritmo de *vector-distancia*.

- ***OSPF***: Este protocolo es probablemente el tipo de protocolo *IGP*[6] más empleado en redes de grandes magnitudes. Este protocolo puede operar con seguridad haciendo uso de *MD5*[7] para autenticar a sus puntos antes de recargar nuevas rutas. *OSPF* no se basa en protocolos *tcp* ni *udp* sino directamente en *IP*.

- ***IGRP****(Internal Gateway Routing Protocol)*: Es un protocolo de encaminamiento de enlace interior, patentado y desarrollado por *CISCO*[8] y se emplea con el protocolo *tcp/ip* siguiendo el modelo *OSI*. Este protocolo esta basado en el algoritmo de *vector-distancia*, es decir, que determina el mejor camino para los paquetes de datos basándose en el ancho de banda, la confiabilidad y la carga del enlace.

- ***EIGRP***: Es un protocolo de encaminamiento híbrido. Se considera un protocolo avanzado que se basa en las características normalmente asociadas con los protocolos del estado del enlace.

[5] Es una fragmento de paquete con la suficiente información para ser enrutado hacia el equipo receptor
[6] Protocolo de Gateway Interno
[7] Algoritmos de reducción criptográfica de 128 bits
[8] Empresa multinacional dedicada a la fabricación, venta, mantenimiento y consultoría de equipos de telecomunicaciones

1.1.5. Hardware en redes

Los dispositivos empleados para la creación de redes de datos se caracterízan por la forma en que éstos conectan a una red así como sus ventajas y desventajas que presentan en cada una de sus funciones. A continuación se describen estos dispositivos.

Definición de switch

Una vez definido el modelo *OSI*, el concepto de un *switch* es *Un dispositivo electrónico situado en la capa 2 del Modelo OSI que hace la tarea de puente multipuerto*, y a diferencia de un *Hub*, un *switch* tiene la capacidad de tomar deciciones en base a la dirección MAC, dicha acción la realizan conmitando datos sólo desde el puerto al cual esta conectado el host correspondiente, lo cual garantiza la entrega de datos al host destino seleccionado, por lo tanto un *switch* hace que una red LAN sea más eficiente.

Ventajas y desventajas

El uso de un *switch* ofrece ciertas ventajas, como se presentan a continuación:

- Fusión de múliples redes en una sola
- Capacidad de administración para ofrecer más control y seguridad sobre las redes interconectadas.
- Control en el envío de paquetes a los destinatarios
- Aumento en el ancho de banda
- Permiten múltiples conexiones entre segmentos
- Evitan que el tráfico de un segmento pase a otro
- No propaga tramas defectuosas o paquetes mal formados

Estas ventajas hacen que la administración de una red sea más sencilla, eficiente y segura. En contraste las desventajas que ofrecen estos dispositivos son las siguientes:

- Alta latencia (depende del tamaño de la trama)
- Retardos, esto se da si la trama atraviesa por varios *switches*

Bucles de red

Uno de los puntos críticos de los *switches* son los bucles de red o ciclos, los cuales consisten en habilitar caminos o rutas para llegar de un equipo a otro a través de un conjunto de *switches*. Los bucles de red son producidos debido a que dos *switches* están en redundancia, es decir, que cada *Switch* transmite paquetes hacia todos los dispositivos que están conectados a él, mientras que el otro *Switch* hace lo propio provocando así un flujo circular de paquetes lo cual impactará a la red provocandos su saturación o caída.

Spanning Tree Protocol

El *Spanning Tree Protocol (STP)* es un protocolo de red empleado en la capa 2 del modelo *OSI*. La función principal de este protocolo es la de controlar la presencia de bucles de red en las diversas topologías de redes debido a la existencia de enclaces redundantes. Este protocolo permite a los dispositivos de interconexión en redundancia activar o desactivar puertos de conexión en el flujo de paquetes. El *STP* lo que hace es calcular cuál de los puertos de los *switchs* será bloqueado para romper con el ciclo que está generando el bucle e impacatando la red. La figura 1.6 presenta como el protocolo de *STP* bloquea unos de los puertos en una arquitectura en redundancia para que no se generen ciclos y saturación de paquetes en una red.

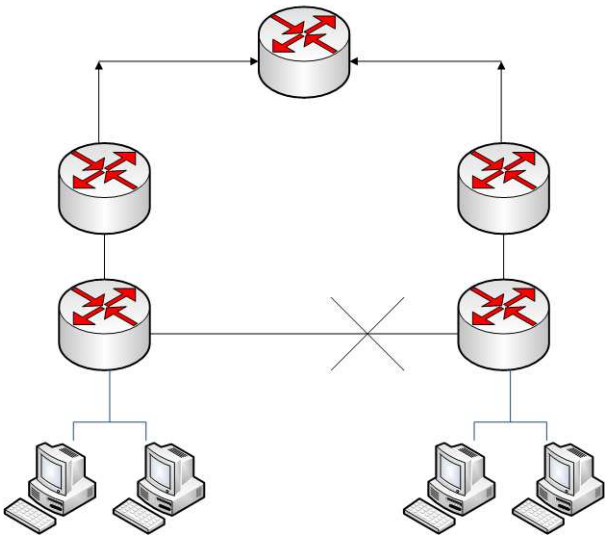

Figura 1.6: Arquitectura de STP

Definición de hub

Un *hub* o concentrador es *un dispositivo que permite unificar o centralizar a un conjunto de computadoras.*

Tiene la función de repetir cada paquete de datos a cada uno de los puertos con los que cuenta, es decir, que todos los equipos conectados a un *hub* pueden tener acceso a la información que circula por la red ya que este dispositivo no impone ningún tipo de restricción. Se encarga además de enviar una señal de choque a todos y cada uno de sus puertos se detecta una colisión en la red.

Estos dispositivos operan en la capa 1 del modelo *OSI* al igual que los repetidores. Además los *hubs* pueden ser implementados únicamente con tecnología analógica, ya que sólo une conexiones y no altera las tramas que llegan.

Ventajas y desventajas

Los *Hubs* ofrecen muchas desventajas en un ambiente de red en el aspecto de infraestructura y velocidad de transmisión de los datos. Además que su costo es muy bajo y son ideales para la implementación de redes pequeñas. Estos dispositivos son los más adecuados para la imeplementación de topologías de tipo *bus*, en donde las computadoras están interconectadas mediante una línea en común.

Definición de router

Un *Router* o *Gateway* es un dispositivo de red que tiene la función de determinar la ruta más óptima para la transmisión de datos de una red a otra. Estos dispositivos operan sobre la capa tres del m Los *routers* al operar en la capa 3 del modelo OSI, utiliza el direccionamiento *IP* y no un direccionamiento *MAC* como lo hacen los *switches*.

Los *Routers* son componentes escenciales para redes extensas que usen *TCP/IP*, ya que tienen la capacidad de organizar el creciemiento de una red no importando la zona geográfica.

Características

Los *Routers* cuentan con dos adaptadores de red mediante los cuales se les asigna una dirección IP para su gestión que son adaptadores Seriales y Ethernet. Además cuentan con dos clases de puertos denominados *puerto de consola* y *puerto de red*.

Puerto de consola: Un *Router* usa un puerto de consola para permitir al administrador su configuración, control y administración, por lo regular este tipo de puerto siempre esta presente en todos los *Routers*.

Puerto de red: El *Router* cuenta con diferentes cantidades de puertos de red dependiendo el modelo del equipo.

Funciones de un Router

Este dispositivo realiza dos funciones principales:

- *Determinación de rutas*
- *Reenvío de paquetes*

Determinación de rutas

Los *Routers* deben mantener sus tablas de rutas para asegurar que otros *Routers* conozcan dichas rutas y así al momento de intercomunicarse sean capaces de determinar cual es la mejor ruta entre ellos. Estos dispositivos realizan esta tarea mediante el uso de un protocolo de rutas, el cual trasmite la información de red de un *Router* hacia la tabla de rutas de otro *Router*.

Reenvío de paquetes

Los *Routers* hacen uso de su tabla de rutas para determinar a donde tienen que enviar paquetes. Estos dispositivos envían paquetes a través de las interfaces de red hacia una red destino, tomando como identificador la dirección IP de el paquete origen.

1.1.6. Seguridad

Dentro de la seguridad existen dos conceptos fundamentales: *Seguridad de la información y seguridad informática*, los cuales en muchas ocasiones se piensa que se trata del mismo concepto, ya que se confunde el término de *Información* con el de *Informática* siendo conceptos totalmente diferentes. Para aclarar el panorama de lo que es la seguridad se mencionan sus diferentes conceptos:

Seguridad de la información

La información como elemento fundamental y crítico de los sistemas de cómputo se define como *"todo aquel conjunto de datos que son generados a partir de la interacción de un individuo y una computadora"*.

Partiendo de la definición anterior el concepto de *Seguridad de la información* se puede definir de la siguiente manera: " *es todo lo que se refiere a las precauciones que se realizan para evitar cualquier acción que comprometa a la información"*.

Seguridad informática

Debido al incremento de ataques informáticos e intrusiones no deseadas en los sistemas de información, surge la necesidad de brindar protección a los sistemas. Todas estas precauciones se engloban de

forma general en el concepto de seguridad informática definido como sigue:

"*conjunto de herramientas automatizadas que tienen la función de proteger y brindar confidencialidad, integridad y disponibilidad a toda la información almacenada en un sistema informático*".

Es importante mencionar que se ha avanzado mucho en el campo de la seguridad informática, aunque también se debe mencionar que las técnicas de intrusión para comprometer a los sistemas día con día tienen mayor impacto y son más agresivas. En este sentido esta tesis se enfocará profundamente en la creación de una *Honeynet* funcional de segunda generación con el fin de brindar un laboratorio completo de pruebas e investigación para los administradores.

Es importante mencionar que antes de implementar o brindar seguridad a la información se deben plantear las siguientes interrogantes: *¿Qué se quiere proteger?, ¿Para qué se quiere proteger?, ¿De quién se quiere proteger? y ¿Cómo se quiere proteger?*. Respondiendo a estas interrogantes un administrador de sistemas o de redes tendrá un panorama amplio de cuales son sus necesidades de seguridad y con que recursos cuenta para brindar dicha seguridad.

Seguridad de la red

Debido al avance significativo de las intrusiones a los sistemas de información, la seguridad requiere de la valoración de un tercer concepto: *Seguridad de la red*. Este concepto surge a raíz de la creciente interconectividad de equipos de cómputo y de las redes de información, por lo que el concepto de seguridad de la red lo podemos definir de la siguiente manera: *Conjunto de herramientas y normas de seguridad para la protección de equipo activo dentro de una red.*

En las redes los ataques informáticos se clasifican en dos: *internos* y *externos*. Los ataques internos son aquellos ataques que son realizados por personal interno en nuestra red. Los ataques internos son más fáciles de realizar ya que el intruso no tiene la necesidad de preocuparse por como entrar a la red, ya que se encuentra dentro de ella, talvez porque es un empleado de alguna empresa o un estudiante de alguna institución, lo cual hacer más difícil controlar este tipo de ataques, ya que estadísticamente más del 80 % de los ataques informáticos son desde el interior de la red.

Por otro lado los ataques externos ocupan el resto de este porcentaje estadístico, pero no por este hecho dejan de ser menos importantes que los internos, ya que en ambos la información es el objetivo principal. La única diferencia entre ambos es la complejidad que requiere para realizar cada uno de ellos.

Una vez que se tiene conocimiento de los tipos de ataques informáticos es importante mencionar que existen dos tipos de topologías de redes a este respecto que son las conocidas como: *Redes cerradas y redes abiertas.*

Redes cerradas: Son aquellas redes que proporcionan conctividad únicamente a usuarios y sitios totalmente conocidos, por lo que éste tipo de redes no permiten concexiones a redes públicas, como ejemplo de red pública se tiene *Internet*.

Redes abiertas: Este tipo de redes a diferencia de las redes cerradas, si proporcionan conexiones hacia redes públicas, ya sea *Internet* o alguna otra red corporativa que sea pública. Esto es debido a que hoy en día las corporaciones requieren de conexiones a redes públicas para poder realizar transacciones, operaciones, y brindar servicios.

Seguridad física

Otra área que cubre la seguridad es la *Seguridad Física*. Esta seguridad hace referencia a las barreras físicas y a los mecanismos de control en el entorno de un sistema informático para poder brindar protección al *hardware* de amenazas físicas.

La seguridad física se basa en mecanismos, es decir, en una serie de normas a seguir para garantizar en un primer plano el entorno adecuado para el óptimo funcionamento y operabilidad de los equipos de cómputo. Por otro lado se trata de garantizar la integridad de los mismos basándose en un análisis de terreno y/o climatológico del lugar.

Todo el análisis de la seguridad física actúa directamente sobre dos tipos de amenazas latentes: *la naturaleza y el hombre.*

Como ejemplos de amenazas naturales se tienen: *Sismos, inundaciones, humedad e incendios.* Por otro lado ejemplos de amenazas humanas son: *Acceso a personal no autorizado, uso de dispositivos electrónicos no autorizados(USB, ipods, teléfonos celulares, etc).*

1.2. Servicios de la seguridad

1.2.1. Panorama de la seguridad

Debido a que los sistemas informáticos son cada día más importantes y necesarios en el manejo de los negocios, escuelas y en muchas otras áreas, toda la información se genera electrónicamente debe de tener muchas de las funciones usualmente interpretadas por los documentos en papel, como ejemplo: las firmas, fechas, correciones, escrituras, etc.

Sin embargo se cuenta con una desventaja, ya que en un documento en papel se puede distinguir si éste se trata de una versión original o una copia, mientras que en un documento electrónico no se puede detectar esta diferencia debido a que un documento electrónico es una secuencia de bits.

Definición

Un servicio de seguridad *es aquel que mejora el entorno de un sisema de información y el flujo de la información de una organización.*

Los servicios de seguridad son: *Confidencialidad, Autenticación, Integridad, No repudio, Control de acceso y Disponibildad.*

Confidencialidad

Por confidencialidad debemos entender que es la capacidad que se tiene para asegurar que sólo personas autorizadas tengan acceso a cierta información.

El servicio de la confidencialidad es algo a lo que se enfrentan diariamente las pesonas y organizaciones debido a que ambos requieren que su información se encuentre segura de cualquier froma posible, ya que, si ésta información es descubierta o reveldada por alguien no autorizado, puede tener consecuencias de gravedad, como ejemplo: *conocer secretos de desarrollo de una empresa, contraseñas de tarjetas de crédito, vida personal de algún individuo, etc.*

Es importante resaltar que existen personas que quieren tener acceso a información que no les pertenece sin autorización con el fin de poseer información confidencial de alguna empresa, otras tantas lo hacen por diversión o por demostrar superioridad, tal como es el caso de los *Script kiddies*[9] y de los intrusos (blackhats). Por tanto el servicio de la confidencialidad brinda protección en el almacenamiento de los recursos y la información asegurando que nadie pueda leerlo y/o tener acceso a la información sin una autorización previa.

Cabe mencionar que uno de los mecanismos para implementar la confidencialidad en la información es la *criptografía.* La criptografía *es un conjunto de técnicas y métodos que se encargan de ocultar o*

[9]Es un cracker inexperto que usa programas, scripts, exploits, troyanos, nukes, etc, creados por terceros para romper con la seguridad de un sistema

cifrar los mensajes que vienen en texto claro, ya sea con símbolos o diferentes alfabetos con el fin de que sólo sea entendible para personas autorizadas a leer dicho texto.

Autenticación

Este servicio de seguridad se enfoca principalmente en la comprobación de identidad de una persona u objeto. Este servicio de seguridad hace que los usuarios adopten medidas de seguridad como tener contraseñias fuertes, cambio de contraseñias en un periodo de 6 meses mínimo, etc.

En la vida cotidiana un individuo realiza este servicio, como ejemplo, lo aplica con el hecho de saber en donde esta su casa, cuando se visita la casa de algún amigo o familiar se identifica el domicilio previamente ubicando la dirección, etc. en el aspecto de las comunicaciones este servicio se encarga de que una comunicación sea autentica, es decir, le asegura al receptor de una o mensaje que dicho mensaje proviene de la fuente que el espera. Este servicio es empleado por un sistema para verificar si dicha entidad es quien dice ser y se realiza principalmente a través de lo siguiente:

- *Algo que se sabe:* contraseñas, un número personal, es decir, que el sistema verifica estos datos comparándolos con los que cuenta en su base de datos.
- *Algo que se tiene:* Tarjetas o pasaportes, lo cual es procesado por el sistema para verificar su identidad.
- *Algo que se es:* La voz, huellas digitales, etc. Con estos datos el sistema puede verificar la identidad del individuo.

Integridad

Este servicio provee de controles (sello o firma) para asegurar que el contenido de un mensaje no haya sido alterado y garantizar que la secuencia de los datos se mantenga durante una transferencia de datos. Con la implemetación de este servicio se garantiza que la información no sufra modificaciones o alteraciones por usuarios que no posean este privilegio. Otra de las bondades que ofrece este servicio de seguridad es la de asegurar que los mensajes envaidos por un emisor se reciban tal cual fueron enviados de un receptor sin tener alteraciones o modificaciones. La integridad es de dos tipos: *integridad de contenido* e *integridad de secuencia del mensaje.*

- *Integridad de contenido:* Brindan pruebas de que el contenido de un mensaje no ha sido alterado ni modificado.
- *Integridad en la secuencia del mensaje:* Ofrecen pruebas de que la secuencia de un mensaje se ha mantenido durante una transmisión de datos.

No repudio

Este servicio de seguridad actúa directamente sobre los emisores y receptores de mensajes, es decir, que tanto un emisor como un receptor no pueden negar o desconocer un mensaje transmitido. El no repudio se aplica al problema de la negación de información que se recibe de otros o de la información que se envía otros usuarios. El servicio de no repudio ofrece pruebas que pueden ser demostradas a una tercera entidad como:

- *No repudio de origen:* Ofrece pruebas del origen de los datos, con ello se previene a la entidad de origen o emisora de cualquier negación falsa al transmitir datos.
- *No repudio de envío:* Ofrece pruebas del envío de datos, por tanto previene al receptor de dichos datos de una negación falsa al recibir la información.
- *No repudio de presentación:* Ofrece pruebas de presentación de los datos, con ello protege contra cualquier intento falso de negar que los datos fueron presentados para el envío.
- *No repudio de transporte:* Ofrece pruebas del transporte de datos, con los que se protege cualquier intento de negar que los datos no fueron transmitidos.

- *No repudio de recepción:* Ofrece pruebas de la recepción de los datos, con ellos se protege al emisor de que el receptor niegue haber recibido la información.

Como ejemplo del servicio de no repudio se tienen las firmas digitales, las cuales son empleadas porque la característica de ser creadas por el firmante y verificadas por otros.

Control de acceso

En este servicio se definen los diversos controles que se pueden tener para el acceso a la información. El acceso a un medio de información puede ser controlado a través de un dispositivo como puede ser el caso de una puerta a través de un dispositivo activo tal es el caso de un monitor. Es importante anotar que este servicio de seguridad va estrechamente ligado con el servicio de la autenticación, ya que se tiene que autenticar en primer lugar al usuario para poder determinar si se le otorga el acceso a un recurso o lugar.

También una de las características importantes de este servicio es que controla los privilegios o permisos de un usuario dentro de un sistema o una red en general. Como ejemplos de permisos o privilegios se tiene:

- Creación o destrucción
- Lectura o escritura
- Adición o modificación
- Exportación o importación
- Ejecución

Estos privilegios pueden ser modificados y revocados únicamente por el administrador de sistemas o de la red. El control de acceso puede ejecutarse de acuerdo con los niveles de seguridad (dependen de los privilegios) y de los recursos de la red.

Disponibilidad

Este servicio se cumple si las entidades autorizadas pueden acceder a la información o recurso cuando así lo requieran y tantas veces sea necesario.

Dentro de la disponibilidad el factor importante es el tiempo, ya que si la información no está disponible para el momento en que ésta se requiere se considera que existe una no disponibilidad de la misma independientemente de si la información esta o no correcta.

1.3. OpenBSD

1.3.1. Antecedentes

OpenBSD surge como parte del proyecto de NetBSD[10] . Es un sistema operativo robusto, portable y multiplataforma (actualmente corre sobre 17 plataformas distintas). Incorpora seguridad con criptografía integrada lo cual lo hace unos de los sistemas operativos más seguros a en el mundo.

[10]sistema Operativo de la familia *Unix* creado por la marca comercial ATT

El sistema operativo *OpenBSD* es un sistema basado en BSD[11], es decir, que es un sistema operativo derivado del sistema *unix*. Cabe anotar que *OpenBSD* tiene similitudes con los sistemas unix[12] y linux pero no es un clon de ellos.

Durante su proceso de creación y separación de *NetBSD*, el equipo de *OpenBSD* le dio un enfoque total de seguridad, basándose en extensas a revisiones y auditorias de su código fuente en cada una de sus versiones, lo que hace de *OpenBSD* en la actualidad uno de los sistemas operativos *open-source* más seguros.

Filosofía

La razón principal por la que *OpenBSD* existe es la seguridad y la filosofía que lo rige es (*Free, Functional and Secure*) Libre, Funcional y Seguro. Esta filosofía ha llevado al sistema operativo *OpenBSD* a convertirse en uno de los sistemas operativos más seguros en el mundo.

Características

El sistema operativo *OpenBSD* cuenta con características que lo hacen original, funcional y uno de los sistemas operativos más seguros del mundo. Las características del sistema operativo son las siguientes:

- Portabilidad
- Auditoría de código
- Criptografía integrada
- Robusto
- Empleo de Packet Filter
- Multiplataforma
- Cumplimiento de normas (iso-17799)
- Licencia libre (BSD)

Estas son algunas de las cualidades por las cuales el sistema operativo se ha convertido en uno de los líderes en el campo de la seguridad.

OpenBSD como firewall

Antes de hablar de *OpenBSD* como firewall se deben considerar algunos conceptos importantes para la correcta compresión del tema.

Firewall: Es un componente o conjunto de componentes que restringen y controlan el acceso entre dos o más redes.

Host: Es una computadora que esta presente en una red de computadoras

Bastion Host: Es un sistema de cómputo que debe ser fuertemenete asegurado, ya que es vulnerable a ataques debido a que usualmente esta expuesto hacia *Internet* y es el punto principal de contacto de los usuarios de una red interna

Dual-Homed-Host: Es un sistema de propósito general que tiene al menos dos interfaces de red

[11] La licencia BSD se emplea para la identificación de un S.O derivado de un Unix derivado de los aportes realizados a ese S.O por la Universidad de Berkeley

[12] Sistema operativo multitareas y multiusuarios desarrollado por los laboratorios Bell de ATT

Una vez definidos estos conceptos, se explicarán las tres formas que tiene *OpenBSD* para configurarse y brindar el servicio de firewall en una red.

Los servicios con los que *OpenBSD* puede ser configurado son:

- NAT (Network Address Translation)
- Bridge (modo puente)
- Load-Balancing (Balanceador de cargas)

NAT*(Network Address Translation)*: La traducción de direcciones de red (NAT) es un procedimiento mediante el cual el *router* se encarga de modificar la dirección IP de los hosts contenidos en una red local. Este servicio se basa en el concepto de *Dual-homed host* en el cual una de las interfaces de red alimenta de conexión a la red protegida con un rango de direcciones IP privadas (no homologadas) con el fin de que éstas no sean ruteables desde *Internet*. Mientras que la segunda interfaz de red tiene asignada una dirección IP pública (homologada) con el fin de que cada petición hecha por las computadoras de la red protegida hacia *Internet* se enruten con la dirección IP de la interfaz externa del firewall y no con la dirección IP con la que cuentan, brindando así una protección hacia la red protegida de cualquier ataque o amenaza del exterior. La figura 1.7 esquematiza este servicio.

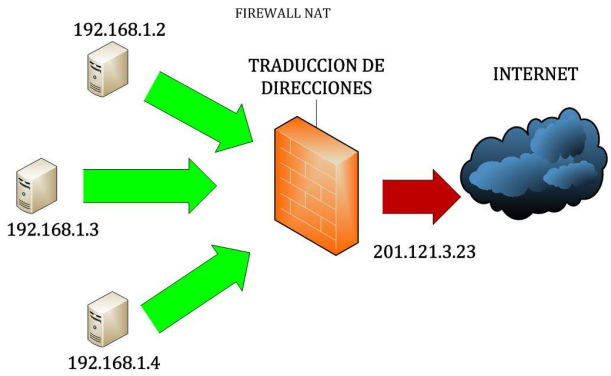

Figura 1.7: Firewall con servicio de NAT

Bridge*(modo puente transparente)*: La configuración de *OpenBSD* como bridge (puente) es un procedimiento mediante el cual se brinda protección a una red interna mediante políticas de seguridad y filtrado de paquetes sin la necesidad de hacer modificaciones de configuración a las computadoras que pertenezcan a la red interna y/o protegida.

En esta configuración *OpenBSD* se basa en el concepto de *Dual-Homed-host*, pero con la característica de que ninguna de las dos interfaces de red cuenta con una dirección IP asignada, por lo que, las computadoras pertenecientes a la red interna mantienen su configuración actual y no necesitan ser reconfiguradas con direcciones IP privadas para que puedan tener acceso a *Internet.* La configuración de *bridge* hace que el *firewall* no sea ruteable (invisible) en la red, debido a que carece de direcciones IP en sus interfaces de red, por lo que los hosts internos de la red tendrán una dirección IP pública y un *Gateway* real para poder tener acceso hacia *Internet*. Estas configuraciones no se las puede ofrecer el *firewall* debido a su transparencia, por lo tanto solo se encargara de filtrarel tráfico entrante y saliente de la red.

La principal función del bridge es gestionar el tráfico entrante y saliente hacia una red interna mediante el uso de políticas de seguridad y reglas de filtrado mediante el uso de *Packet filter*.

El *packet filter* permite o bloquea paquetes cuando se rutean de una red hacia otra (los más frecuentes son los paquetes que provienen de *Internet* y que tiene como destino una red interna). En la figura 1.8 se ilustra un *firewall* en modo *Bridge.*

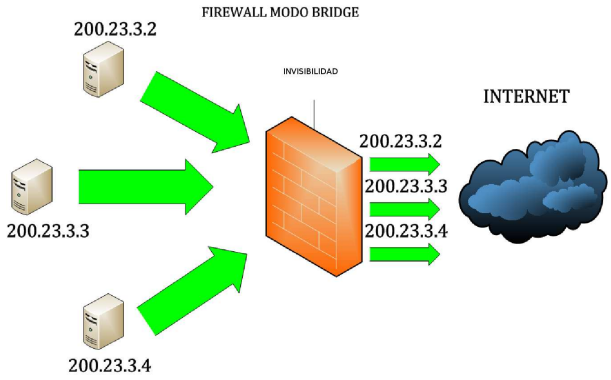

Figura 1.8: Firewall con servicio Bridge

Load Balancing*(Balanceador de Cargas)*: El balanceador de cargas es un procedimiento mediante el cual se configura al sistema operativo *OpenBSD* para que haga la redirección de conexiones entrantes o salientes, con el fin de evitar un congestionamiento de conexiones en la red de producción. La figura 1.9 muestra de manera esquemática un balanceador de cargas.

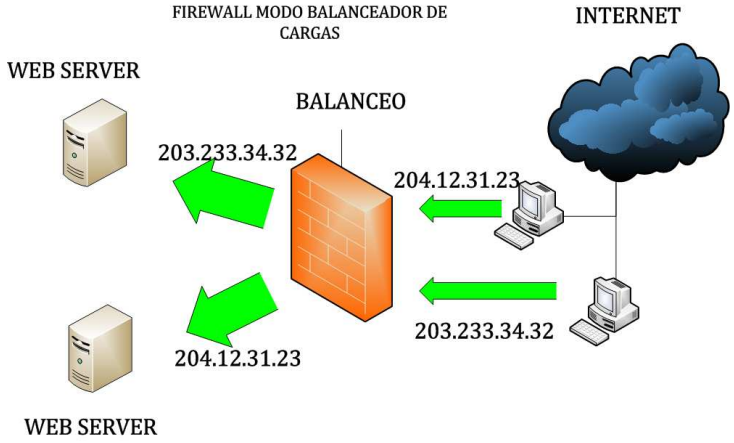

Figura 1.9: Arquitectura de Balanceador de Cargas

1.4. Sistema de detección de intrusos

Defínción

Un sistema detector de intrusos *(IDS)* es una aplicación empleada para la detección de accesos no autorizados a un *host* o a una red. Esta aplicación hace uso de sensores virtuales para detectar anomalías que pueden ser indicio de falsas alarmas o de ataques potenciales.

Tipos de IDSs

NIDS: Son sistemas Detectores de Intrusos basados en red que se encargan de hacer la comparación de cada paquete entrante o existente en una red o contra una lista de ataques conocidos que poseé. La función en términos generales de un Sistema Detector de Intrusos de Red es emitir alertas a los administradores de red informando sobre un posible tráfico sospechoso el cual puede ser un indicio de un ataque potencial en la red. Los *NIDS* son aplicaciones que son implementadas en el perímetro de una red de producción con el fin de que analice y elabore un diagnóstico del tráfico entrante y saliente de

dicha red, enfocándose principalmente en la identificación de ataques y/o incidentes de seguridad.

En la figura 1.10 se observa la implementación de un *NIDS.*

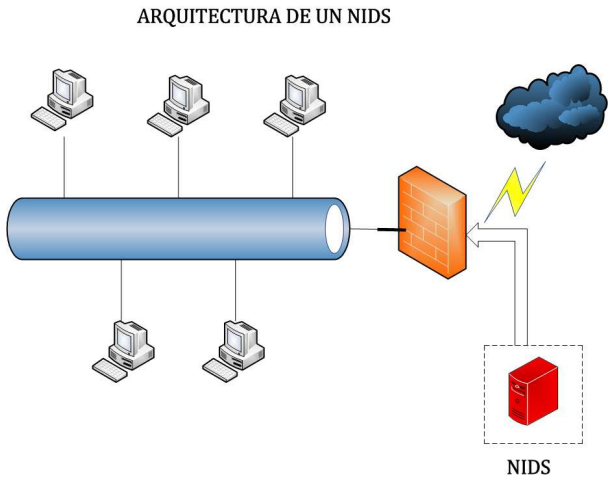

Figura 1.10: Arquitectura de un NIDS

HIDS: Son Sistemas Detectores de Intrusos enfocados en un *host* en particular. Su principal función es buscar anomalías que representen un riesgo potencial para el *host* y sus actividades. En la actualidad existen diversos tipos de *HIDS* como: *Analizadores de Logs, Analizadores de Integridad de archivos, entre otros.*La figura 1.11 muestra la arquitectura de un HIDS.

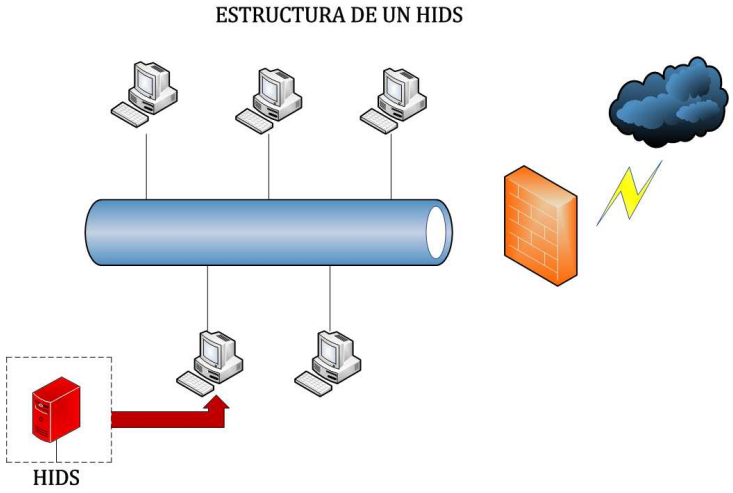

Figura 1.11: Arquitectura de un HIDS

Analizador de logs: Es un *HIDS* que tiene como función hacer una búsqueda minuciosa en el tráfico entrante a un sistema con el fin de identificar patrones de posibles ataques.

Analizador de Integridad de Archivos: Es un *HIDS* que emite una alerta cuando algún archivo del sistema ha sido modificado o alterado, por lo que se considera comprometido dicho *host.*

Los sistemas detectores de intrusos se pueden clasificar en varios tipos:

- ***Por tipo de respuesta***

  Pasivos: Son aquellos que emiten alertas a las autoridades competentes del área de seguridad, pero no actúa contra el atacante.

  Activos: Generan un tipo de respuesta en contra del sistema atacante o fuente de ataque. Este tipo de respuesta puede ser la emisión de una alerta al administrador, y se proceda a cerrar la conexión hacia la red.

- ***Por los medios de uso***

  Basados en Host: Se encargan de recolectar toda la información posible sobre un sistema en particular y sus recursos. Posteriormente hace un análisis de posibles incidentes de seguridad.

  Basados en Red: Estos sistemas funcionan con una interfaz de red en modo promiscuo para monitorear la red en busca de indicios de ataques conocidos.

  Basados en Aplicación: Son *IDSs* que recolectan información de una aplicación activa en el sistema y hacen una búsqueda de indicios de ataques en éstos datos.

  Basados en Objetivo: Son *IDSs* dedicados para garantizar la integridad de un archivo o aplicación objetivo.

  Por tipo de análisis: Son *IDSs* que hacen una búsqueda sobre un patrón de ataque perfectamente identificado.

Falsos positivos y falsos negativos

Falso positivo: Es un evento que es interpretado como un problema cuando es insignificante o inexistente. Ocurre cuando un dispositivo de monitoreo *(IDS)* clasifica a dicho evento como una acción o condición anómala (posible intrusión vulnerabilidad) cuando en realidad es una acción legítima.

Falso negativo: Es un evento que es interpretado como un evento normal cuando es significativo o existente. Ocurre cuando un dispositivo de monitoreo no clasifica a dicho evento como una acción condición anómala, cuando en realidad ésta es una acción no permitida.

1.5. Sniffer

Definición

Es una aplicación que monitorea y analiza el tráfico de una red de computadoras, detectando los cuellos de botella y problemas asociados con la red. La principal función de un *sniffer* es captar el tráfico que circula por una red de producción, no importándole si es o no legítimo dicho tráfico. El modo de operación de estas aplicaciones depende de la topología de la red de producción y del medio de transmisión que compartan dos o más sistemas dentro de la red.

Este tipo de aplicaciones son una de las herramientas más empleadas a por los intrusos, ya que les permite interceptar información, conversaciones, etc.

Debido a la importancia que representa un *sniffer*, esta investigación de tesis estudia a nivel de infraestructura las diversas maneras que existen de implementar un *sniffer* con el fin de tener los mejores resultados.

Ventajas y desventajas

Los *sniffers* son aplicaciones que se han ejecutado en la redes desde hace tiempo. En la actualidad existen dos tipos de *sniffers*:*los comerciales y los libres.* Ambos tipos de *sniffers* son empleados frecuentemente para le gestión y mantenimiento de una red, mientras que con fines o maliciosos son empleados para robo de datos, información, etc.

Las ventajas principales que ofrece un *sniffer* son:

- Fáciles de implementar y operar
- Multiplataforma
- Soporte de más de 300 protocolos
- Portables

Por otro lado la captura de datos, la cual es una de las actividades propias de un *sniffers* en ocasiones suele representar una desventaja para los administradores de redes debido a que los intrusos pueden capturar contraseñas de servidores e incluso de los mismos administradores, además de que se puede conocer la topología de una red.

Ejemplos de sniffers

Ethereal: Es un analizador de redes con una interfaz gráfica y permite reconocer una gran cantidad de protocolos.

Ksniffer: Es un recolector de estadísticas de redes. Tiene la capacidad de observar el tráfico de red de cualquier interfaz conectada a la a computadora.

Dsniff: Es una aplicación que permite auditar redes. Cuenta con una gran variedad de test de penetración.

Snort: Es un *sniffer/logger* de paquetes flexible que detecta ataques, esta basado en la biblioteca *libpcap* y adicionalmente puede ser empleados como *IDS.*

1.5.1. Sistema de bitácoras

Definición

Un sistema de bitácoras es un servidor dedicado al almacenamiento de datos que corresponden al tráfico que circula por una red de datos, así como toda actividad realizada en un *host* en particular o en toda una red.

Syslog

Fue desarrollado por *Erick Allman* como parte del proyecto *sendmail* en 1980. Gracias a su gran utilidad se empezó a usar *syslog* en otras aplicaciones. Hoy en día *syslog* está presente en casi todos los sistemas *unix* y *GNU/Linux* y cuenta con una implementación especial para *Windows. Syslog* es el estándar para el envío de mensajes de registro en una red de computadoras. Considera tanto el protocolo de red como a la aplicación biblioteca que envía los mensajes de registro. Un mensaje de registro generalmente contiene información de la seguridad del sistema, aunque también puede contener cualquier información. Una de las actividades más importantes de *syslog* es registrar lo siguiente:

- Intento de acceso con contraseñas erróneas
- Accesos correctos al sistema
- Variaciones en el funcionamiento normal del sistema
- Información sobre las actividades del sistema operativo
- Errores de Hardware

Adicionalmente es posible registrar el funcionamiento normal de los programas, como por ejemplo las peticiones que recibe un servidor Web. Los mensajes de *syslog* suelen ser enviados por el protocolo *UDP* por el puerto 514, con un formato de texto plano.

1.6. Tcpdump

Tcpdump es una herramienta de *unix* para la recolección de tráfico de red, puede ser usado para el monitoreo y análisis de redes, además de ser la base para otras herramientas de monitoreo.[13]

Servidor syslog remoto

Es un sistema en el cual los *Honeypots* o cualquier tipo de sistemas envía todos sus *logs* obtenidos en su sistema. En otros términos un servidor de *syslog remoto* es un repositorio en donde llegan a almacenarse todos los *logs* que se generan en una red o en un *host*, todo con la finalidad de garantizar la integridad de esos datos ante un ataque contra el *syslog* local de un servidor.

[13]Herramienta de red empleada para el análisis de tráfico en una red

Capítulo 2

Honeynets

2.1. Honeypots

2.1.1. Definición de Honeypots

Antes de hablar de las redes trampa (*Honeynets*) es importante hablar de los *Honeypots*. Los *Honeypots* son tecnologías nuevas y muy dinámicas,aunque no representan una solución para la seguridad de los sitemas, simplemente son herramientas muy flexibles con información diversa de aplicaciones de seguridad.

Una definición formal de *Honeypot* es:Ës un sistema de información con muchos recursos, y su principal valor radica en el uso ilícito o no autorizado del mismo". Esta definición fue desarrollada y adaptada por miembros de la lista de correo del proyecto de *Honeynet*, así como un foro público en *Internet* formado por más de 4,500 profesionales de la seguridad.

La definición no fue fácil de encontrar y adaptar dedibo a que un *Honeypot* puede a ser implementado de muchas formas y de diferentes capacidades. Es importante aclarar que la definición antes mendionada no indica como un *Honeypot* debe comportarse o cual es su propósito. Lo que si podemos decir es que los *Honeypots* trabajan bajo un mismo concepto: *Nadie debería de usarlos o interactuar con ellos, ya que cualquier interacción o actividad en ellos es por definición no autorizada.*

Los *Honeypots* son sistemas que no tienen un valor propio dentro de un entorno de producción debido a que todos los servicios que llegan a brindar son simulados, es decir, sin efecto alguno en la red productiva. Incluso un *Honeypot* no necesariamente debe de ser representado por una computadora, sino que puede ser cualquier entidad digital, normalmente llamadas *Honeytokens*[1]

2.1.2. Tipos de Honeypots

Para la mejor comprensión de los *Honeypots* se pueden dividir en dos categorías generales: Interacción baja e interacción alta. Los *Honeypots* de baja interacción son sistemas que principalmente simulan cualquier tipo de servicio y sistema. Existe software como *BackOfficer Friendly y Specter (Windows)* que es capaz de simular hasta 7 servicios como *ftp, ssh, telnet, http, etc*, y para el caso de *Specter* hasta 13 sistemas operativos y 14 servicios.

En cuanto a lo que ofrece este tipo de *Honeypots* es muy poco, debido a que las actividades de los intrusos son limitadas porque al estar interactuando con un sistema y servicio simulado sus comandos son básicos y limitados, por lo que no se puede aprender mucho con ellos.

[1]Es una entidad digital que se encuentra dentro de un conjunto de información y que representa un medio de fuga de la misma

Por otro lado, estos sistemas son muy fáciles de implementar, mantener y operar debido a la existencia de software antes mencionado, el cual cuenta ya con variedad de opciones compiladas e integradas en el propio emulador, que le da una amplia gama de opciones de operación al administrador.

Uno de los *Honeypots* más conocidos es el *HoneyD*. *HoneyD* es un *Honeypot* de código abierto desarrollado por *Niels Provos* en Abril del 2002. Este *Honeypot* proveé a cualquier usuario acceso a su código fuente, fue desarrollado y diseñado para plataformas *Unix*, aunque también poseé soporte para *Windows*. Una de las características que poseé este software es la capacidad de simular cientos de sistemas operativos y servicios, así como la configuración personalizada para cada uno de éstos. Como ejemplo se puede configurar un *Linux Kernel 2.6* con servicio de *ftp* atendiendo por el puerto 21 y con una dirección ip personalizada y asignada por el administrador.

La diferencia entre los *Honeypots* de interacción alta y los de interacción baja es que todos los sistemas operativos y servicios que ofrecen son reales y no simulados, por lo que cuando exista una intrusión en uno de estos sistemas el intruso realmente estaría interactuando con el sistema y el servicio. Las ventajas que ofrece este tipo de *Honeypots* son grandes, ya que fueron diseñados para capturar grandes cantidades de información, adicionalmente le da ciertas capacidades a los intrusos de interacción como es la capacidad de atacar un servicio con la finalidad de escalar privilegios y ganar acceso al sistema operativo, y de que los intrusos pongan al descubierto todos sus conociemientos y técnicas intrusivas dando como resultado información crítica de donde aprenderemos de los intrusos.

Otra característica importante que ofrece este tipo de *Honeypots* es un comportamiento nuevo, desconocido e inesperado. Es decir, tienen la capacidad de capturar nuevas actividades desde protocolos IP no estandar usados para *tunneling*[2] (envío de datos encapsulados) y ocultar las comunicaciones. El nivel de complejidad de estos sistemas depende del administrador y ésta es inversamente proporcional al riesgo que respresentan estos *Honeypots*.

Como ejemplos de *Honeypots* de interacción alta tenemos a las *Honeynets* y a *Symantec Decoy Server*. *Symantec Decoy Server* es un *Honeypot* comercial vendido por *Symantec*[3] . Como un *Honeypot* de interacción alta no simula servicios ni sistemas operativos, este *Honeypot* únicamente se ejecuta sobre un sistema operativo *Solaris* en sus dos plataformas: *SPARC e Intel*.

La forma en que opera este *Honeypot* es mediante la creación de jaulas*(cages)*. El software toma el *host* existente y crea 4 jaulas idénticas, siendo cada una de ellas un *Honeypot*. Cada una de estas jaulas tiene un sistema operativo real totalmente independiente de las demás.

2.1.3. Uso de los Honeypots

Los *Honeypots* son herramientas sumamente flexibles que pueden ser empleadas para cualquier propósito. En general el uso de los *Honeypots* se pueden desglozar en dos categorías : producción e investigación. Dentro de los ámbitos de producción los *Honeypots* más empleados son los de interacción baja debido a la manera de operar de éstos y su impacto en una red productiva. Mientras que en entornos de investigación los *Honeypots* más empleados son los de interacción alta debido a que en un entorno de éstos lo más importante es la recolección de información para su análisis y así brindar futuras soluciones a incidencias de seguridad. Inherentemente a ambos entornos los *Honeypots* pueden emplearse indistintamente en uno o en otro, ya que no existe una regla que así lo disponga, además de que ninguno de los propósitos de los *Honeypots* es mejor uno que otro.

Estas dos categorías fueron impuestas únicamente con el fin de que identifique el propósito de los *Honeypots* en cualquiera de los entornos antes mencionados y así poder crear una implemantación adecuada de esta tecnología por ejemplo, si es un *Honeypot* de interacción baja ayudará a una red en producción a detectar, prevenir y responder a ataques contra sus servicios críticos. Y si es el caso de un

[2]Es una técnica que consiste en encapsular un protocolo de red sobre otro creando un tunel dentro de una red de computadoras.

[3]Empresa proveedora de software de seguridad y antivirus

Honeypot de interacción alta su propósito será el recolectar toda la información posible de una o varias organizaciones con el fin de analizar dicha información para observar el comportamiento de los intrusos, sus motivos y técnicas.

Ventajas y desventajas

En general los *Honeypots* ofrecen ventajas y desventajas, a continuación se mencionan las ventajas que ofrecen:

Reducción de falsos positivos: Los *Honeypots* reducen en gran medida los falsos positivos debido a que todo acceso o actividad es clasificada como no autorizada, minimizando así el problema que falsos positivos que proporcionan las tecnologías de detección, por lo que cualquier administrador debe poner total atención a cualquier acceso o actividad.

***Captura de falsos negativos*:** A diferencia de las tecnologías de detección los *Honeypots* por su naturaleza son capaces de capturar cualquier tipo de actividad y/o cualquier ataque contra él, sin importar si cuenta con una base de datos de ataques conocidos como lo manejan las tecnologías de detección.

Captura de actividades encriptadas: Los *Honeypots* son capaces de capturar ataques o comunicaciones encriptadas con *ssh, IPsec y SSL*. Debido a que la tendencia en las organizaciones es la adopción de métodos criptográficos para proteger su información.

Flexibilidad: Los *Honeypots* son altamente adaptables a cualquier tipo de ambiente, que va desde una red de producción sencilla (máximo a 10 usuarios) hasta una red productiva de mayor volúmen y tráfico.

En contraste como cualquier tecnología los *Honeypots* presentan desventajas, a continuación sólo se menciona la que es de total relevancia:

Limitación de vista: Un *Honeypot* sólo es capaz de atender a su propio sistema, es decir, no tiene la capacidad de alertarnos de que otro sistema ha sido comprometido. En otras palabras no tienen la interacción con otros sistemas, más que son el propio.

2.2. Honeynets

2.2.1. Definición de Honeynet

Una vez entendido el concepto de un *Honeypot* se puede entender que es una *Honeynet* debido a la siguiente definición: *Una Honeynet es un tipo de honeypot de alta interacción que actúan sobre una red entera, orientado especialmente a la investigación* . Este tipo de redes trampa *(Honeynets)* tienen la finalidad de recolectar información sobre ataques en contra de la información de los sistemas, accesos ilícitos por parte de los intrusos, explotación de vulnerabilidades en aplicaciones, etc. Toda esta información es analizada por la gente encargada de la seguridad de un sistema con el fin de investigar como suceden este tipo de actividades y como podrá afectar a sus ambientes de trabajo, para así encontrar una solución óptima para contrarrestar este tipo de eventos en las redes y los sistemas de información.

Valor de las Honeynets

La razón principal por la que existen las *Honeynets* es la información. Esta información tiene diferente valor para diferentes personas y esta depende de lo que se quiera lograr con ella. Las redes trampa han demostrado su enorme valor como herramienta de investigación en el área de la seguridad de la información. Las *Honeynets* han sido empleadas por muchos profesionales y gente encargada de la seguridad de la información, tanto en el sector público como privado, con la finalidad de reforzar la seguridad de sus instituciones en base a la implementación de estas redes trampa que les proporcionan información

vital sobre las debilidades y/o fallos encontrados en sus redes o aplicaciones. Las *Honeynets* son una herramienta con un valor muy alto por su naturaleza, pero a su vez involucra niveles de riesgo muy altos.

Tipos de Honeynets

Existen varios tipos de *Honeynets*, las cuales pueden ser implementadas para diferentes fines de investigación. Los diferentes tipos de las *Honeynets* son los siguientes:

Honeynets de primera generación: Este tipo de *Honeynets* fueron desarrolladas en 1999 y fue el *Honeynet Project*[4] el primero en intentar la implementación de la tecnología de las *Honeynets*. Este tipo de *Honeynets* son ideales para la captura de actividades automatizadas tales como:*worms, script Kiddies*, *auto-rooters* y *mass-routers*. Actualmente no es muy recomendado implementar este tipo de *Honeynets* debido a que todos los servicios *(ftp, ssh, etc)* son simulados y son muy fáciles a de detectar por un intruso.

Honeynets de segunda generación: Este tipo de *Honeynets* fueron desarrolladas en 2002 y son redes más avanzadas. Estas *Honeynets* tienen el propósito de ser implementadas de manera más sencilla, difíciles de detectar y seguras de mantener. Estas redes son implementadas para capturar actividades más avanzadas por parte de un intruso, para este fin las *Honeynets* de segunda generación hacen uso de la herramienta Sebek[5] para su fase de captura de datos.

Honeynets de tercera generación: Este tipo de *Honeynets* permiten la recolección de datos de una manera más profunda mediante el uso de sistemas centralizados para dicha recolección de datos.

Honeynets virtuales: Las *Honeynets* virtuales están diseñadas para hacer implementaciones de *Honeynets* mucho más fácil de administrar y mucho más accesibles en costo, debido a que toda la implementación de una *Honeynet* se puede hacer dentro de un sólo sistema. (ver apartado 2.5)

Honeynets distribuidas: Una *Honeynet* distribuida es un conjunto de *Honeynets* implementadas en una red grande o en *Internet*. (ver apartado 2.6)

2.3. Honeynets de primera generación

2.3.1. Arquitectura de Honeynets de primera generación

Una de las primeras cosas que se tienen que tener en cuenta es que las *Honeynets* no son un *software*, ni una aplicación que se pueda instalar. Las *Honeynets* de primera generación son muy simples de implementar debido a que utiliza el control de datos mediante un *firewall* por filtrado de paquetes que puede ser un *Linux (iptables)* o un *OpenBSD (PF)* y la captura de datos la realiza mediante un *IDS*. La figura 2.1 muestra la arquitectura de una *Honeynet* de primera generación.

[4]Grupo de investigadores en seguridad informatica www.honeynetproject.org
[5]Herramienta de captura de datos que ayuda a recrear los eventos en un sistema

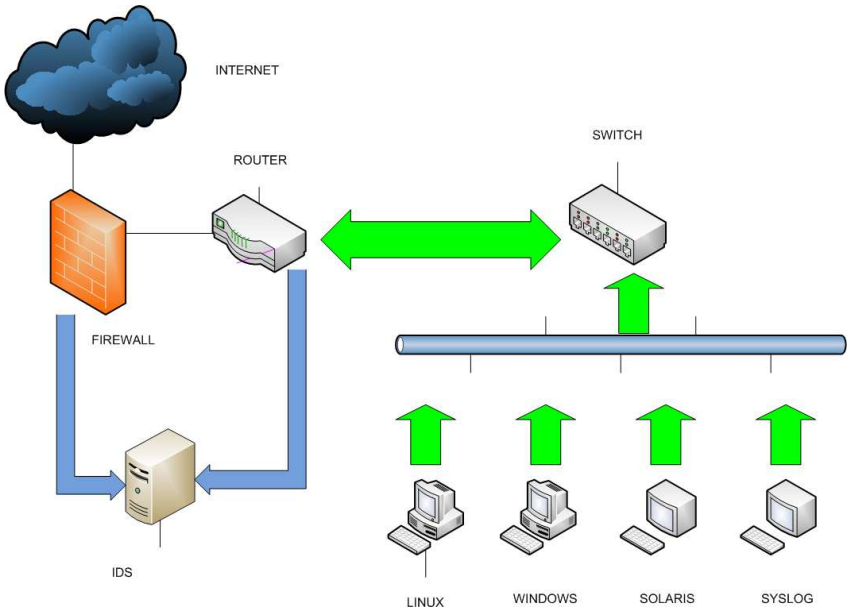

Figura 2.1: Arquitectura de Honeynet de generación I

Como se muestra en la figura 2.1 el *firewall* cuenta con tres interfaces de red con una dirección IP asociada a cada una de las interfaces. La primera de ellas es la que proporciona la conexión a *Internet*. La segunda administra a la *Honeynet* y la tercera es utilizada para la administración de los *logs*. Tiene cuatro estaciones de trabajo *(Honeypots)* que corren diferentes sistemas operativos. Cuenta con un *IDS* o *sniffer* con dos interfaces de red, pero sólo una de ellas tiene asociada una dirección IP para la administración del *IDS* y la otra no cuenta con una dirección IP asociada ya que sólo esta captando todo el tráfico que pasa hacia la *Honeynet*. Finalmente cuenta con un *firewall*, el cual es el elemento más importante dentro de esta arquitectura, ya que es el que controla las conexiones desde y hacia la *Honeynet*. Este *firewall* por el hecho de tener direcciones IP asociadas a cada una de sus interfaces de red, hace que sea visible para los intrusos y puede ser blanco de escaneos.

Características de las Honeynets de primera generación

Las *Honeynets* de primera generación de caracterizan por implementar únicamente los mecanismos básicos para controlar el flujo de los datos, estos mecanismos son: *Control y Captura de datos.*

Control de datos

El control de datos es uno de los requerimientos más importantes dentro de una *Honeynet*, ya que este controla todo el tráfico dentro de una *Honeynet* y es el dispositivo que lleva acabo la acción del control de los datos es el *firewall* y puede ser implementado mediante un *iptables* en *Linux* o con un *Packet Filter* en *OpenBSD*. Este mecanismo es el que mitiga los riesgos dentro de una *Honeynet*. Por riesgo debemos de entender que existe siempre un intruso con el potencial suficiente para usar a la *Honeynet* y atacar a otros sistemas de la misma red o para hacer daño a sistemas que no están dentro de la *Honeynet*. En general el propósito de toda *Honeynet* es hacer el mayor esfuerzo posible para que cuando un intruso se encuentre dentro de una *Honeynet* no pueda lanzar ni por accidente ni concientemente ataques contra otros sistemas fuera de la *Honeynet*.

El administrador debe de ser muy cuidadoso en el manejo y administración de la *Honeynet*, ya que debe de permitir al intruso actuar con toda la libertad posible, sin que éste se percate de que esta siendo monitoreado. Cabe mencionar que entre más actividades realice un intruso más se aprenderá de ellos. En contraste entre más libertad se le conceda a un intruso dentro de la *Honeynet* aumentará el riesgo, por lo que es recomendable encontrar un equilibrio entre estos dos aspectos para poder tener una *Honeynet* lo más controlada posible así como el mayor número de actividades realizadas por los intrusos.

El *Honeynet Project* sugiere algunas directrices con respecto al control de datos: la *Honeynet* debe de contar con almenos dos capas para el control de datos para protegerla contra fallos. El control de datos puede ser implementado vía respuesta automática o intervención manual. En el caso de que las capas del control de datos fallen, el sistema debería responder automáticamente para prevenir todos los accesos entrantes y salientes de la *Honeynet*. El esquema del control de datos debe de ser configurable por el administrador en cualquier momento, incluyendo la administración remota.

Todas las actividades dentro de la *Honeynet* deben de ser fácilmente controladas y monitoreadas. El administrador debe de ser capaz de mantener el estado de todas las conexiones entrantes y salientes.

Captura de datos

Este requerimiento es el encargado de monitorear y almacenar en bitácora todas las actividades de los intrusos dentro de la *Honeynet*. Todas las actividades capturadas son analizadas posteriormente con el fin de aprender de las herramientas empleadas, las tácticas y los motivos que a llegan a tener los intrusos para comprometer a los sistemas de información. La meta principal de este mecanismo es capturar tantos datos y actividades como sean posibles, sin que el intruso se percate de este proceso. Para hacer más difícil a un intruso el detectar estos procesos se deben tomar en cuenta los siguientes puntos:

- Realizar las menos modificaciones posibles a los *Honeypots*, ya que entre más modificaciones se hagan más crece la posibilidad de detección.
- Las actividades capturadas no deben de ser almacenadas dentro de los mismos *Honeypots*, ya que un intruso puede percatarse del almacenamiento de los mismos y podrá modificarlos o eliminarlos.
- Todos los datos capturados deben de almacenarse en un sistema aparte como un *syslog*.

En este sentido el *Honeynet Project* suguiere las sugiere las siguientes recomendaciones con respecto a la captura de datos:

- Las actividades capturadas en la *Honeynet* deben ser almacenadas por un periodo de 1 año.
- El administrador debe de ser capaz de monitorear remotamente las actividades de la *Honeynet* en tiempo real.
- Debe haber almacenamiento de datos automáticos para futuros análisis.
- El administrador debe mantener un *log* estandarizado de cada *Honeypot* implementado.
- El administrador debe de mantener un documento detallado de cada *Honeypot* comprometido.
- Los recursos empleados para la captura de datos deben ser seguros para que garanticen las integridad de los datos.

Otra de las características de este tipo de *Honeynets* es que todos los servicios que ofrece son simulados (no reales), lo cual convierte a la *Honeynet* en una red de bajo riesgo debido a que si un servicio llegara a ser comprometido por un intruso no causaría ningún impacto dentro de la *Honeynet*, ni mucho menos en la arquitectura de la misma.

Métodos empleados

El método típico empleado en las *Honeynets* de primera generación es la captura de datos *Data Capture*, el cual es un método que nos permite tener el control sobre los intrusos y las actividades que éstos realizan dentro de una red comprometida. Sin embargo, siempre cabe la posibilidad de encontrarnos con un intruso hábil que tenga la capacidad de detectar que se esta usando un dispositivo para controlar el tráfico de la *Honeynet* y el exterior *(Internet)*.

Este método empleado en estas *Honeynets* se implementan mediante *Iptables* en *Linux* y *Packet Filter* en *OpenBSD*. Es decir, el dispositivo que controla tanto el tráfico entrante como saliente de la *Honeynet* es administrado por el *firewall* en cualquiera de esas dos modalidades, de ahí que el *firewall* es el elemento más importante dentro de la arquitectura de una *Honeynet*. El *Data Capture* o captura de datos es una funcionalidad que esta orientada a la recolección y almacenamiento de evidencias generadas por ataques en contra de las propias *Honeynets* y de las *Honeynets* hacia *Internet*.

Es importante hacer notar que toda la actividad capturada debe de ser administrada dentro del ambiente de la *Honeynet*, por lo que la captura de datos no sólo es un análisis de *logs* ni un análisis del tráfico en la red, sino que estos dos métodos de análisis se combinan con un monitoreo exhaustivo de los sistemas y con aspectos suaves de monitoreo como: obtención de información *(Intelligence Gathering)*, observación del intruso *(modus operandi)*, obtención de perfiles, etc. Por otro lado, se emplean tecnologías que se combinan con los métodos antes descritos para obtener mejores resultados como son: *Snort* y *TcpDump*. Es importante resaltar que tanto los métodos y las tecnologías empleadas en las *Honeynets* de primera generación también se emplean en *Honeynets* de segunda generación.

Requerimientos de Hardware y Software

Uno de los primeros pasos para la implementación de una *Honeynet* implica el obtener el *hardware y software* necesarios como: *servidores, PCs, una buena conexión a Internet, etc.*

Requerimientos de Hardware.

Los principales elementos para implementar una *Honeynet* de primera generación son:

- 3 computadoras intel o superiores con interfaces de red
- un Hub
- Un servidor o PC con 2 interfaces de red (Control de datos)

Al menos dos de las computadoras deberán tener dos tarjetas de red. En cuanto al *firewall* lo recomendable es una PC *Pentium* o superior, pero incluso una 486 es suficiente (dependiendo de la velocidad de conexión a Internet). En el peor de los casos si se presenta un ataque severo de *flood*[6], el *firewall* causará un cuello de botella[7] y empezará a bloquear las conexiones. Para la PC en donde se implementará el *IDS* debe de ser una máquina con los suficientes recursos para que pueda capturar todo el tráfico, generar alertas, y ejecutar herramientas de análisis. La máquina que será la víctima *(Honeypot)* deberá ser más rápida que una 486 para generar un ambiente lo más real posible.

Hardware para la conexión a Internet.

El *hardware* necesario para la conexión a *Internet* puede ser un *ISDN*[8], un modem de cable o una *DSL*[9]. Cabe mencionar que una conexión *dial-up* teóricamente es posible, pero dificultaría al intruso realizar sus actividades y una conexión T1 sería buena pero no es necesaria. Los requerimientos de *software*

[6] Repetición desmesurada de un mansaje en una red de comunicaciones
[7] Fase en una cadena de producción que alenta los procesos
[8] Red Digital de Servicios Integrados o ISDN por sus siglas en inglés: Red que facilita conexiones digitales extremo a extremo
[9] Digital Subscriber Line: Protocolo de banda ancha para Internet

que se necesitará para la implementación de una *Honeynet* será los siguientes: *Linux Red Hat* (o cualquier otra distribución de Linux).Este *software* será implementado en los *Honeypots*, y se tiene la ventaja de que las distribuciones de *Linux* en particular *Red Hat* incluyen *firewall* con es el caso de *iptables*, un limitador de ancho de banda como es el caso de *TC(Trafic Control)*, y *tcpdump* para el monitoreo del tráfico. Además, cuentan con base de datos como *Mysql* utilizada para el almacenamiento de alertas y datos. *Snort Network IDS*. El sistema detector de intrusos *Snort* es el núcleo para la captura de los datos. *Snort* cumple la función de un generador de alertas contra ataques y también realiza una captura completa de tráfico que circula por la red. Este *IDS* también hace la captura de sesiones en protocolos de texto plano como son *ftp, telnet y http*.

2.3.2. Instalación y configuración de un Firewall para el control de datos primario

Como se ha mencionado anteriormente el *firewall* es el elemento más importante dentro de la arquitectura de una *Honeynet* ya que es el dispositivo que realiza el mecanismo del control de los datos y de las actividades que se lleven acabo dentro de la red. La instalación y configuración de este *firewall* puede realizarse con *Linux* como puedes ser *Red Hat, Debian, Suse*, etc. Más sin embargo esta tesis se enfocará a la instalación y configuración de este *firewall* sobre *OpenBSD*. Para la instalación del *firewall* se requiere de una PC Pentium 3 en adelante. Esta máquina debe de contar con dos tarjetas de red ya que se generarán dos redes, una de ellas hacia *Internet* y la segunda generará la red de la *Honeynet*.(ver tema 3)

2.3.3. Instalación y configuración de un IDS para la captura de datos primario

La máquina designada para esta configuración deberá de tener instalado el sistema operativo *OpenBSD 4.3* y sobre este sistema debe de ser instalado y configurado el sistema detector de intrusos *snort*(ver cap.3). La máquina designada para esta configuración debe de contar con dos tarjetas de red. Una de las tarjetas de red deberá contener una dirección que pertenezca al rango de direcciones IP designadas para esta tesis, con el fin de que esta tarjeta de red este capturando todo el tráfico generado dentro de la *Honeynet*, mientras que la segunda tarjeta deberá de estar conectada a la red de producción con el fin de que capture toda actividad entre el *firewall* y la *Honeynet*.

2.3.4. Preparando y conectando la Honeynet a Internet

Se debe de realizar una sesión de pruebas para garantizar que las instalaciones y configuraciones que se le han hecho a la *Honeynet* funcionan correctamente. Una de las primeras pruebas que se debe de realizar son las de conectividad, es decir, se debe garantizar que todas y cada una de las máquinas víctimas *(Honeypots)* tengan configurada una dirección IP, un *gateway* y un servidor de *DNS*, para que puedan tener comunicación y acceso a la red. Posteriormente hacer un escaneo de puertos al *IDS* para verificar que tiene abiertos los puertos necesarios. Además verificar las conexiones de *ssh* desde el *IDS* hasta el *firewall* y verficar que no existen otros métodos de conexión como *telnet, ftp,etc*, y viceversa.

Por otro lado se debe de probar la disponibilidad del *firewall* para el envío de mensajes de *syslog* hacia el *IDS* así como verificar la posibilidad para hacer conexiones desde el *firewall* hacia el *IDS* por cualquier método de conexión. Se debe de realizar pruebas con *Snort* y *tcpdump* para garantizar la captura de los datos que se generan desde y hacia la *Honeynet*. Es de vital importancia probar la funcionalidad del *bash loggin*, es decir, se debe de garantizar que las sesiones de cada usuario que tenga acceso a la *Honeynet* debe de ser capturado y almacenado en archivos de sesión.

Finalmente debe de sincronizarse el reloj tanto en el *IDS* y el *firewall*. Esta acción se puede realizar mediante el demonio de *ntpd*. Una vez que se han realizado todas las pruebas necesarias para garantizar la buena configuración y el buen funcionamiento de la *Honeynet*, el siguiente paso a realizar es conectar la *Honeynet* a la red.

Una vez conectada la *Honeynet* a *Internet* lo primero que se debe de hacer es probar la conectividad de entrada y salida de la *Honeynet*, así como el flujo de datos y el almacenamiento de las sesiones de los usuarios que acceden a la *Honeynet*.

2.4. Honeynets de segunda generación

Los mecanismos para el control y captura de datos implementados en las *Honeynets* de primera generación pueden ser mejorados, siendo este el principal motivo por el cual se desarrollan las *Honeynets* de segunda generación. Las *Honeynets* de segunda generación tienen como principales características el que sean más fáciles de implementar y que sean más difíciles de detectar.

Arquitectura de Honeynets de segunda generción

La arquitectura de las *Honeynets* de segunda generación tiene algunas diferencias con respecto a la arquitectura de las *Honeynets* de primera generación. La figura 2.2 presenta la arquitectura de una *Honeynet* de segunda generación.

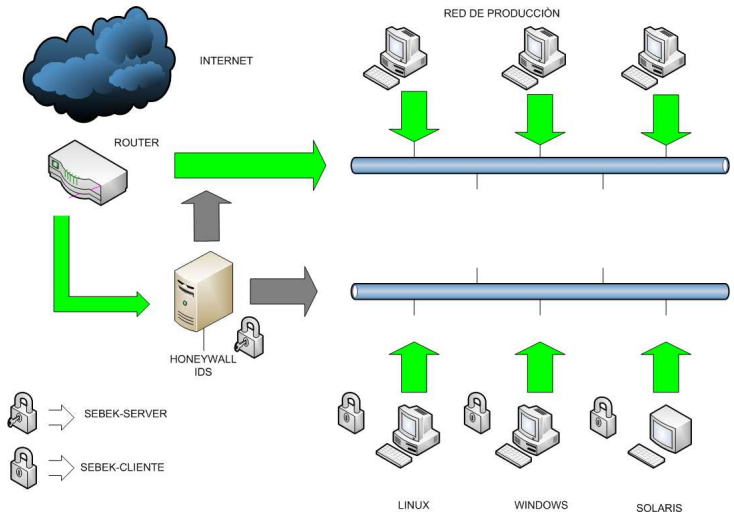

Figura 2.2: Arquitectura de Honeynet de generación II

Como se muestra en la figura 2.2 aparece un nuevo componente llamado *Honeynet gateway*.

El *Honeywall* es el nuevo componente crítico que poseé una *Honeynet* de segunda generación. Este dispositivo contiene todos los mecanismos empleados en una *Honeynet* de primera generación, como son la captura y el control de datos, lo cual hace que la *Honeynet* sea más fácil de mantener.

El *Honeywall* es un dispositivo que actuaría como un *gateway* en capa 2 del modelo *OSI*, es decir, como *bridge* (puente). Este dispositivo hace que la *Honeynet* sea más difícil de detectar, debido a que como esta configurado en modo *bridge*, sus interfaces de red no poseen direcciones IP, por lo que este dispositivo es invisible en la red.

2.5. Control y captura de datos en Honeynets de segunda generación

Como se señaló anteriormente, existen dos segmentos de red (LAN1 y LAN2) que son interconectados mediante el *Honeywall*. Las acciones que realiza este dispositivo es reenviar los paquetes *ethernet* (frames) desde la *LAN1* hacia la *LAN2*, lo cual se realiza a través de las direcciones *MAC* de los dispositivos conectados a la red.

Una vez que se ha implementado el *gateway*, el siguiente paso a seguir es la limitación de las conexiones, con el propósito de controlar el número de conexiones entrantes y salientes para que al alcanzar el número máximo de conexiones (definidas por el administrador) se bloqueé cualquier otra conexión adicional, previniendo así un escaneo masivo o un ataque de *negación de Servicio (DoS)*. Para realizar dicha acción empleamos el *Packet Filter* de *OpenBSD*, con el cual podremos controlar la cantidad de ocasiones que un intruso puede iniciar una conexión saliente por los protocolos *TCP, UDP, ICMP* u otros. El administrador de la *Honeynet* debería ser muy cuidadoso al definir el número máximo de conexiones salientes, ya que un intruso podría dedicarse a realizar cualquier número de conexiones salientes esperando saber si éstas son bloqueadas y así detectar que se encuentra en una red trampa, lo cual, ilimitaría al administrador a aprender de las tácticas y habilidades del intruso.

El *Honeynet Project* realiza la limitación de conexiones mediante el uso de *iptables* en conjunto con *snort-inline*. Esta tesis utilizara por completo *PF*(*Packet Filter*) de *OpenBSD* mediante el uso completo de un conjunto de reglas para realizar dicho control. (ver cap.4)

Las *Honeynets* de segunda generación aplican la misma técnica empleada en las *Honeynets* de primera generación, pero empleando herramientas y métodos mejorados. En general, la información es obtenida de tres diferentes formas, mediante el *logging* del *firewall*, el *logging* del *IDS* y el sistema de *logging* de cada *Honeypot*. Cada una de estas formas proporciona información parcial sobre un posible ataque, y la combinación de las tres nos dará toda la información necesaria para comprender y analizar un ataque.

Firewall logging

Este es el mecanismo que se encarga de registrar todas las conexiones entrantes y salientes en la siguiente ruta */var/log/messages*. Esta información capturada es de vital importancia ya que es la primera información que tenemos sobre las actividades del intruso, además de ser nuestra primera alarma contra ataques que son dirigidos hacia el exterior de la *Honeynet*.

IDS logging

El principal propósito del *IDS logging* es capturar cada paquete junto con todo su contenido o *payload*[10] que entra o sale de la red. Aunque esta acción también puede ser realizada por el *firewall logging*, es recomendable para el administrador de la *Honeynet* no hacer una sobrecarga de tareas en estos mecanismos, ya que si ocurre algún incidente de seguridad que involucre a cualquiera de éstos, se pueda contar con la información de los otros mecanismos. Una particularidad sobre este método es que debe de asociar el *sniffer* a la interfaz interna del *(Honeywall)*, ya que si éste se asocia a la interfaz externa además de capturar tráfico que circula por la *Honeynet*, también se capturará el tráfico fuera de la red, lo cual podría contaminar a los datos capturados dentro de la red y ante algún suceso de seguridad los datos no serían confiables para su estudio posterior.

System logging

Este es uno de los elementos que requiere de más atención ya que captura toda la actividad realizada por un intruso dentro del mismo *Honeypot*. Pero como se comento anteriormente, los intrusos hoy en día utilizan canales *SSH* o *3DES*[11] para comunicarse con otras máquinas a comprometidas. Una de las

[10] En términos informáticos significa la recarga de datos a una máquina o servidor

[11] En criptografía el Triple *DES* se llama al algoritmo que hace triple cifrado del *DES*. También es conocido como *TDES*

ventajas que ofrece este método es que cualquier dato o sesión que se cifra es descifrada en la máquina destino (otro Honeypot), y una vez que es posible capturar esto datos cifrados, se podrá evitar las comunicaciones cifradas. Así este método también ofrece una desventaja y es que ya no es posible capturar las pulsaciones de las teclas con sólo monitorear el segmento de red, si no que ahora se o deben de obtener de manera local en el propio *Honeypot*.

La captura de datos en las *Honeynets* de segunda generación se implementa dentro del *Honeywall* el cual nos proporciona una administración, mantenimiento y monitoreo centralizado. Una de las principales características que nos ofrecen estas redes de segunda generación es la capacidad de tratar con intrusos que emplean conexiones cifradas dentro de la *Honeynet*.

Honeynet Router

El *Honeynet Router* es un dispositivo que genera una segunda subred de la *Honeynet* que es muy restringida y en la cual sólo se encuentra el servidor de *Syslog remoto*. Este dispositivo proveé un ambiente más estricto de seguridad con respecto al ambiente de los *Honeypots* empleando *Packet Filtering*. En este ambiente de trabajo que genera el *Honeynet Router* permite que los paquetes *UDP* circulen desde la *Honeypots* hasta el servidor de *syslog remoto*, adicionalmente todo el tráfico *icmp* esta permitido. La figura 2.3 muestra la arquitectura de un *Honeynet Router* en una *Honeynet*.

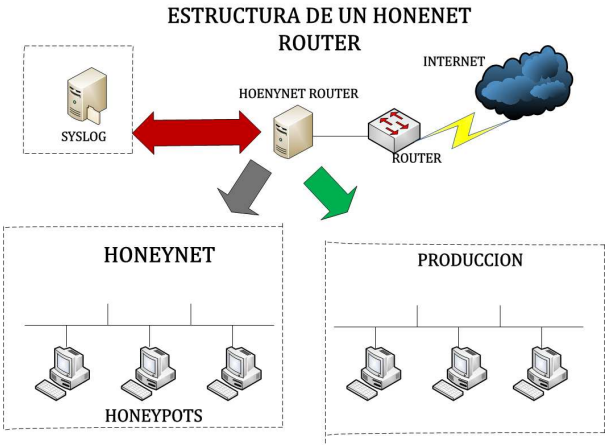

Figura 2.3: Arquitectura de Honeynet Router

Honeynet Gateway (Honeywall)

El *Honeywall* es un dispositivo de seguridad dentro de la *Honeynet* que esta configurado en modo *bridge* (puente) y tiene la función de conectar dos segmentos de la *Honeynet* (la red de producción y la red de los *Honeypots)*. A continuación se explican las fases que se requieren para la configuración y puesta en marcha de un *Honeywall*:

- **Fase 0:** En esta fase de hacen los preparativos iniciales, como son instalación del sistema operativo *(OpenBSD)*, actualizar el sistema operativo (parches de seguridad), así como la plena identificación de todos las configuraciones que serán requeridas.

- **Fase 1:** En esta fase se debe de tener el sistema funcionando con sus parches de seguridad previamente instalados. Con estas características se tiene un sistema con los mínimos requerimientos para realizar pruebas al sistema.

o *3DES*, fue desarrollado por *IBM* en 1978.

- **Fase 2:** En esta fase se debe de tener la configuración completa del sistema, es decir, se deben de tener las herramientas que se requieren completamente instaladas y probadas.

La figura 2.4 representa la arquitectura del *Honeywall* en una *Honeynet*

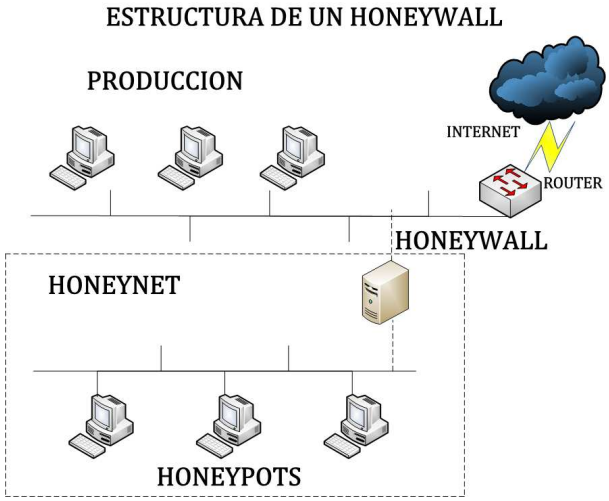

Figura 2.4: Arquitectura de un Honeywall

2.6. Honeynets virtuales

Definición de Honeynets virtuales

Las *Honeynets* virtuales son un concepto que combina todos los elementos físicos de una *Honeynet* dentro de un sólo sistema, empleando tecnología de *software* de virtualización. Este tipo de *Honeynets* ofrecen ventajas y desventajas con respecto a las *Honeynets* tradicionales. Las ventajas que ofrece una *Honeynet* virtual es la reducción de los costos y la facilidad de su administración y mantenimiento, ya que todo esta implementado en una sola máquina y sistema. Por tanto ya a no es necesario ocupar ocho o nueve máquinas para implementar una *Honeynet*, sin embargo estas ventajas tienen un costo:

- *Limitaciones:* Se tienen limitaciones en cuanto a los sistemas operativos a implementar, ya que éstos deben de estar soportados tanto por el *hardware* y *software* de virtualización. Por mencionar un ejemplo la mayoría de las *Honeynets* virtuales están implementadas y soportadas sobre arquitecturas X86, lo cual limita a implementar sistemas operativos bajo esa arquitectura y sería imposible implementar una *Sparc Workstation* o un *Cisco Router*, etc.

- *Alto Riesgo:* Los intrusos pueden tener la habilidad de comprometer el *software* de virtualización y tomar el control de toda la *Honeynet*, obteniendo así el control de todos y cada uno de los sistemas *(Honeypots)* de la red, así como anular o evadir los mecanismos de la captura y control de los datos.

- *Riesgo de Fingerprinting:* El *Fingerprinting* es la forma en que se identifica que sistema operativo esta implementado en una máquina, esta acción se puede determinar de forma local o remota.

2.7. Honeynets virtuales híbridas

Las *Honeynets* híbridas se caracterízan por ser sistemas totalmente separados y aislados. Estas redes están compuestas de la combinación de una *Honeynet* clásica y el *software* de virtualización. Así mismo contienen un *Honeywall*, el cual se encarga de hacer los mecanismos del control y la captura de los datos. Debido a que son sistemas aislados se reducen en gran medida los riesgos de ser comprometidos, de tal manera que todos los sistemas víctima de la *Honeynet* se encuentran implementados en un sólo sistema base, mientras que en una máquina totalmente separada de este sistema base a se implementa el *Honeywall*, el cual controla mediante los mecanismos antes mencionados a la red híbrida. Dentro del sistema base se pueden implementar tantos sistemas operativos como permita el *software* de virtualización, de tal forma que todos son independientes entre sí. Un ejemplo de lo anterior se muestra en la figura 2.5

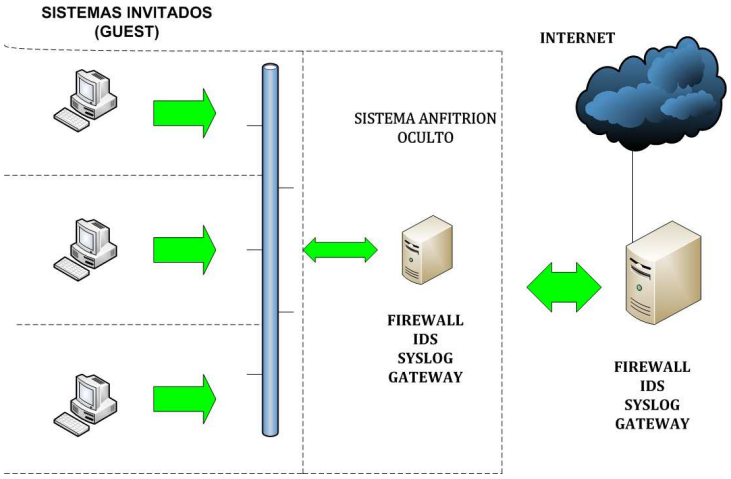

Figura 2.5: Honeynet Híbrida

Las *Honeynets* híbridas ofrecen dos principales ventajas y son las siguientes:

- **Más seguridad:** Con este tipo de redes el único daño que puede causar a un intruso es que sólo puede tener acceso a otros *Honeypots* pero no al *Honeywall*, debido a que éste se encuentra totalmente separado de la *Honeynet*.

- **Más Flexibilidad:** Se puede hacer uso de una gran cantidad de *software* y *hardware* para los mecanismo de control y captura de datos, además a de que se puede añadir otro *Honeypot* en cualquier momento.

Las desventajas que ofrecen este tipo de redes son las siguientes:

- **No son portables:** Debido a que este tipo de redes constan de más de una máquina, se dificulta su movilidad.

- **Costos y espcios:** la inversión de dinero y espacio es muy alta. Además de que posiblemente se requiera invertir dinero para la implementación de otra máquina con fines de espacio para los sistemas.

2.8. Honeynets virtuales auto-contenidas

En contraste con las *Honeynets* híbridas, las *auto-contenidas* son una *Honeynet* completa contenida en una sola máquina mediante *software* de virtualización. En otras palabras las redes *auto-contenidas* son aquellas *Honeynets* que contienen tanto a los *Honeypots* como al *Honeywall* en un sólo sistema base, como se muestra en la figura 2.6

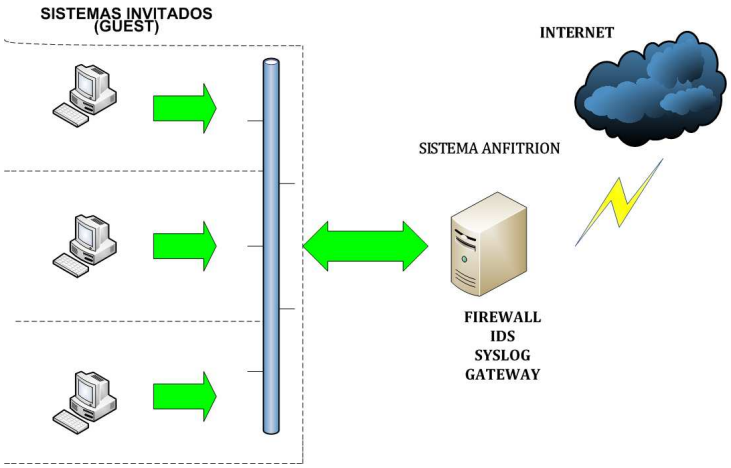

Figura 2.6: Honeynet Autocontenida

Debido a su implementación estas *Honeynets* ofrecen las siguientes ventajas:

- **Portabilidad:** Pueden ser implementadas en una laptop y ser transportadas a cualquier lugar. Como ejemplo de su alta portabilidad estas redes pueden ser operadas y/o manipuladas en una presentación, en una conferencia, etc.

- **Adaptabilidad:** Estas redes pueden ser conectadas a cualquier red y tendrán la capacidad de operar sin problemas.

- **Espacios y costos:** Debido a su implementación estas redes únicamente requieren de una máquina, lo cual reduce los costos, además de que ocupa poco espacio.

Por otro lado las desventajas que presenta son:

- **Fallos en el hardware:** Debido a que estas redes están implementadas en un sólo *hardware*, si se llegase a presentar un fallo en éste la *honeynet* estará totalmente deshabilitada para poder operar.

- **Recursos:** El *hardware* para implementar este tipo de redes debe de contar con los suficientes recursos en cuanto a la memoria y procesador.

- **La seguridad:** Se presenta el riesgo de que un intruso pueda tener acceso a otros recursos del sistema que no sea la *Honeynet*, aunque esto depende en gran medida del *software* de virtualización empleado.

- **Software:** Existen limitaciones de *software*, ya que si el *hardware* es un *Intel*, será imposible implementar algún sistema *Cisco* o algún *Sparc*.

2.9. Honeynets sobre VMWARE

Para realizar la virtualización de las *Honeynets* y de casi cualquier sistema operativo, se requiere de *software* especializado y dedicado a dicho fin, tal es el caso de *VMWARE, GSX*, entre otros. *VMWARE* es un *software* de virtualización disponible para plataformas *Windows y Linux*, que tiene la función de simular a un sistema físico (hardware) con todas las características de dicho sistema. Adicionalmente este *software* tiene la característica de contener tantos dispositivos sean necesarios para el sistema simulado, por ejemplo: se pueden agregar cualquier cantidad de tarjetas de red, como de discos duros, se puede manipular la memoria RAM, tarjetas de sonido, etc. Las ventajas que ofrece este software sobre una *Honeynet* son las siguientes:

- **Variedad de sistemas:** Se pueden ejecutar diversos sistemas operativos dentro del entorno virtual denominado *guestOS*[12] , entre los cuales se distinguen: *Linux, Windows, Solaris, FreeBSD* y otros.

- **Tarjetas de red:** Se pueden manejar las tarjetas de red de dos formas, una en modo *bridge* el cual permite la asignación de direcciones IP homologadas o públicas, y en modo NAT que permite obtener direcciones IP no homologadas a partir de la dirección real del sistema base (no virtual).

- **Movilidad de sistemas:** Debido a que *vmware* crea imágenes de los sistea mas operativos en archivos, genera que los sistemas puedan ser movidos a otros entornos virtuales y poder operar de manera normal.

- **Administración:** *Vmware* debido a sus ambientes gráficos tanto en *Linux* como en *Windows* permiten una instalación y configuración sencilla.

- **Soporte:** Debido a que *vmware* es un producto comercial cuenta con todo el soporte para actualizaciones y parches.

En contraste las desventajas de emplear este tipo de software son:

- **Costo:** Debido a que *vmware* es un producto comercial, cuenta con licencias de uso las cuales tienen un costo elevado (3,000 pesos por licencia). Lo cual limita mucho el uso de esta herramienta en algunos sectores, como tal es el caso de las escuelas, y en algunas empresas pequeñas.

- **Recursos:** Este software debe ser ejecutado bajo un ambiente gráfico y a cada entorno virtual requiere de una ventana, lo cual, repercute en la capacidad de la memoria del *hardware.*

- **Límite de sistemas:** En un entorno *vmware*, se pueden ejecutar sólo 4 máquinas virtuales, lo cual puede provocar una limitante para una a Honeynet.

- **Fingerprinting:** Las máquinas virtuales son vulnerables a los escaneos o reconocimientos de sistema, debido al manejo de direcciones *MAC* en las tarjetas virtuales.

2.10. Honeynets sobre GSX Server

Vmware GSX Server es una versión más robusta que *Vmware Workstation* ya que soporta más sistemas operativos. El *GSX Server* es el idóneo para la implementación de una *Honeynet* debido a las siguientes características: Sistemas operativos soportados: soporta

[12]Son los sistemas simulados en un anfitrió

Windows en sus diversas versiones *Win 95, 98, 2000, XP y .NET*. Soporta además a algunas distribuciones de *Linux*, aunque oficialmente no soporta *BSD* ni *Solaris*.
Opciones de Red: *GSX Server* contiene todas las opciones que contiene *Vmware Workstation*.

Ambientes X: *GSX Server* soporta más sistemas operativos que *Vmware Workstation* debido a que no necesita de un ambiente gráfico.

Administración: Puede ser administrado mediante una interfaz web, es decir, se pueden iniciar, pausar y apagar las máquinas virtuales desde a esta interfaz.

Terminales Remotas: Una de las principales características de *GSX Server* es que se puede hacer uso de una terminal remota para la administración de una red.

Recursos: Soporta hasta 8GB de memoria en cada host, así como hasta 8 CPUs y 2GB de memoria virtual.

Soporte: *GSX Server* incluye soporte para actualizaciones y parches.

En consecuencia las desventajas son las siguientes:
Costos: Una licencia de *GSX Server* cuesta alrededor de 3500 dlls.

Límites en los sistemas: Sistemas como *Solaris y FreeBSD* no son oficialmente soportados lo que afectaría los recursos de la *Honeynet*, pero se pueden instalar.

2.11. Honeynets sobre User-mode-Linux

Herramienta desarrollada por *Jeff Dike*. Es un módulo especial del kernel que permite ejecutar diversas versiones de *Linux* simultáneamente. Algunas de las ventajas de esta herramienta son las siguientes:

- **Licencia:** Se tiene acceso al código fuente, debido a que es abierto.
- **Recursos:** No requiere del uso de X, puede ejecutar varios sistemas con poca memoria.
- **Redes virtuales:** Tiene la capacidad de crear varias redes virtuales completas.
- **Keystrokes:** Tiene la capacidad de almacenar las teclas pulsadas *(keystrokes)* en el propio host, la cual es una característica relevante para una *Honeynet*.

Como consecuencia algunas de sus desventajas son las siguientes:

- **Disponibilidad:** Hasta el momento esta herramienta únicamente está disponible para entornos virtuales de *Linux*, aunque ya se desarrolla una versión para *Windows*.
- **Ambientes gráficos:** Toda la configuración e implementación se realiza desde línea de comandos debido a que carece de interfaz gráfica.
- **Soporte:** Debido a que es una herramienta de código abierto no cuenta con un soporte oficial.

2.12. Honeynets distribuidas

Definición de Honeynets distribuidas

Una *Honeynet* distribuida es un conjunto de *Honeynets* implementadas en diferentes lugares físicos con el fin de observar las actividades en diferentes redes. Existen dos formas de *Honeynets* distribuidas: *Honeynets* de distribución física (Physically Distributed Honeynets) y *Granja de Honeynets (Honeynets Farm)*.

Las *Honeynets* de distribución física es una extensión lógica de las *Honeynets* de generación II, la cual proporciona la capacidad de realizar o análisis centralizados de los datos capturados por múltiples *Honeynets* ubicadas en diferentes zonas geográficas. La granja de *Honeynets* es una extensión más amplia de una *Honeynet* de generación II, ya que se hace una combinación entre las propias *Honeynets* y la tecnología de las Redes privadas virtuales (VPN), con el fin de distribuir *Honeypots* virtuales a través de la red. En la figura 2.7 se muestra una *Honeynet* distribuida.

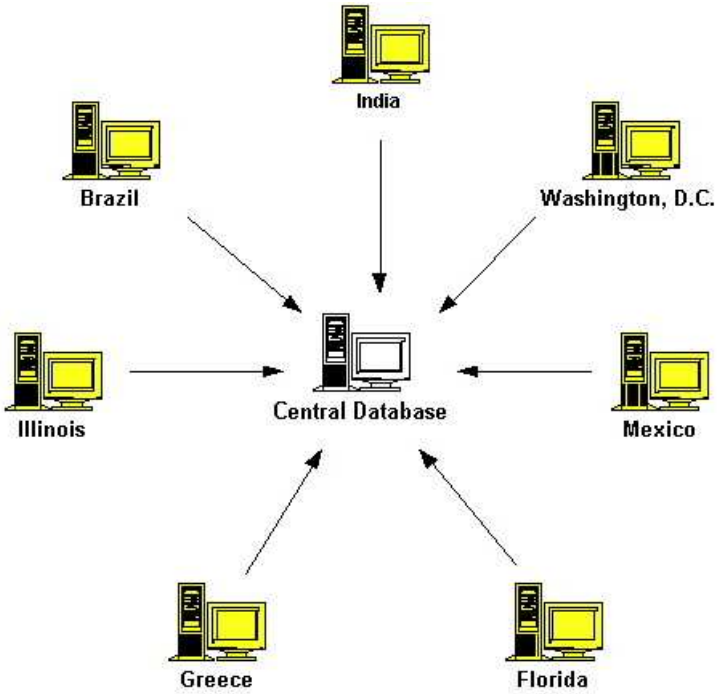

Figura 2.7: Honeynets distribuidas

El valor principal de una *Honeynet* distribuida recae en la operación, ya que un administrador puede determinar el lugar en donde se esta llevando un ataque e incluso puede identificar el comportamiento de un intruso. Otro punto importante es que mediante este tipo de redes un administrador puede determinar que grupos u organizaciones son las víctimas mas frecuentes por los intrusos. Por lo cual las *Honeynets* distribuidas representan una excelente solución a este tipo de problemas.

Distribución física

Como se muestra en la figura 2.7 la distribución física de las *Honeynets*, además de hacer el monitoreo de sus redes locales, envía los datos capturados a un sistema central. Existe una diferencia entre dos *Honeynets* físicamente independientes y una *Honeynet* distribuida, aunque cabe señalar que ambas redes hacen el monitoreo , esta diferencia radica en la capacidad de examinar los datos de las dos redes haciendo uso del mismo sistema.

La principal ventaja de esa técnica es que no se requiere de muchos esfuerzos para el envío de los datos a la base de datos central. Por otro lado, la desventaja es que el harware debe de estar presente en el sitio en donde se ubique el proveedor de servicios de red (ISP), lo cual hace que la implementación de la *Honeynet* sea lenta y muy costosa.

2.12.1. Granjas de Honeypots

Durante los dos últimos años los *Honeypots* han demostrado su valor y su importancia dentro del análisis de vulnerabilidades y ataques informáticos. Los *Honeypots* muestran muchas ventajas, las más importantes son la reducción de los falsos positivos y la capacidad de analizar comportamientos desconocidos en redes corporativas. Debido a estas ventajas las organizaciones están optando por la implementación de grandes cantidades de *Honeypots*.

En contraste una debilidad con la que cuentan los *Honeypots* es que tiene un campo de visión muy limitado, es decir, que sólo tienen la capacidad de ver la actividad que interactúa en el mismo *Honeypot* y no más alla. A diferencia de los *Sistemas Detectores de Intrusos* (IDS) los *Honeypots* no cuentan con la capacidad de controlar y capturar todo el tráfico en una red, por lo que los intrusos deben de probar y comunicarse con el *Honeypot* para que éste pueda realizar su trabajo (obviamente sin que el intruso tenga conociemiento de la existencias del Honeypot). Lo anterior significa que un *Honeypot* para que pueda monitorear y capturar la actividad en una red, debe de ser desplegado en dicha red, y para este fin se debe de realizar un direccionamiento del tráfico mal intencionado en una red hacia el *Honeypot*.

Para realizar dicho direccionamiento de tráfico mal intencionado hacia el *Honeypot* una solición viable es el despliegue de *Granjas de Honeypots* (Honeypot Farms).

Esta técnica se usa para porder realizar el despliegue de grandes cantidades de *Honeypots* y hacer la redirección de tráfico hacia a ellos desde la red de producción.

El concepto de *Granjas de Honeypot* (Farming) es muy simple, porque en lugar de desplegar un gran número de *Honeypots* o un *Honeypot* en cada red, basta con el despliegue de la *Granja de Honeypots* en una buena y consolidada. Esta simple red de *Honeypots* se convierte en un recurso de seguridad dedicado.

Los intrusos entonces serán direccionados hacia esta *Granaja de Honeypots*, es importante volver a mencionar que los *Honeypots* no son capaces de hacer captura de tráfico en la red, por lo que no es necesario que los *honeypots* estén conectados físicamente a la red, lo que significa que pueden estar presentes de forma virtual en la red.

Una vez implementada la *Grnaja de Honeypots*, ésta actúa como un *proxy*, transportando el tráfico mal intencionado para su análisis y este proceso debe de ser totalmente transparente para el intruso. La figura 2.8 muestra la imeplementación de una *Granja de Honeypots*.

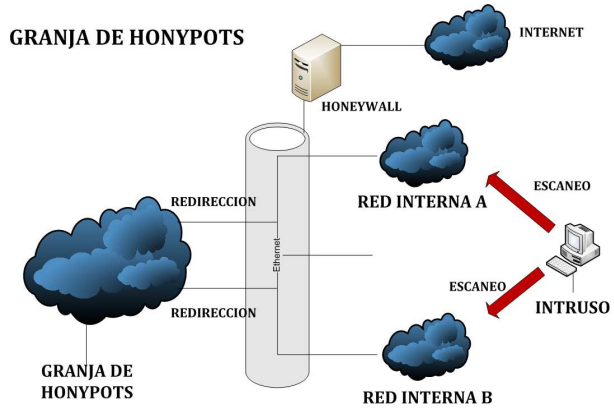

Figura 2.8: Granja de Honeypots

Como se muestra en la figura 2.8, el despliegue de una *Granja de Honeypots* es muy simple y no se requiere de tener personal dedicado a la construcción, configuración y mantenimiento de los *Honeypots* para cada red (teniendo en cuenta que algunas organizaciones tienen cientos de redes interconectadas). En lugar de tener personal dedicado a estas tareas. la *Granja de Honeypots* reduce estos procesos creando una única unidad centralizada de datos. Además de que la *Granja de Honeypots* podría convertirse en parte de un SOC (Security Operation Center) en donde se cuenta con todos los recursos y mano de obra necesaria para generar soluciones de este tipo.

Capítulo 3

Instalación de la Honeynet

En este capítulo se detallará la instalación y configuración de los servicios que contendrán los *Honeypots*, así como de todos los equipos que formarán parte de la *Honeynet* de Segunda Generación.

3.1. Instalación y configuración del servidor de correo

La instalación y configuración de este servicio se hará sobre un sistema operativo *Fedora Core*. Los requerimientos para la instalación del servidor de correo son los siguientes:

- **Sendmail**
- **Dovecot**
- **Sendmail.mc**
- **Squirrelmail**
- **Sebek cliente**

3.1.1. Instalación de paquetes necesarios para un servidor de correo

Sendmail es el Agente de Correo (Mail Transport Agent) más popular, responsable quizá de transportar un poco más del setenta por ciento del correo electrónico en el mundo. Cabe mencionar que es el agente de correo que más reporta incidentes de seguridad.

Sendmail es el único de estos paquetes que viene por default en la instalación de *Fedora Core*, aunque no contiene todos los archivos para su configuración, por lo que es necesario instalar los archivos faltantes así como los paquetes restantes.

Existen dos formas de hacer la instalación de paquetes y que son: mediante el uso del disco de instalación de *Fedora Core* o el uso del gestor de paquetes *YUM*.

Uso del disco de instalación Para instalar paquetes usando el disco de instalación del sistema operativo se requiere de ir a la pantalla de *agregar/quitar* paquetes y seleccionar en esta los paquetes adecuados, como se muestra en las figuras 3.1 y 3.2:

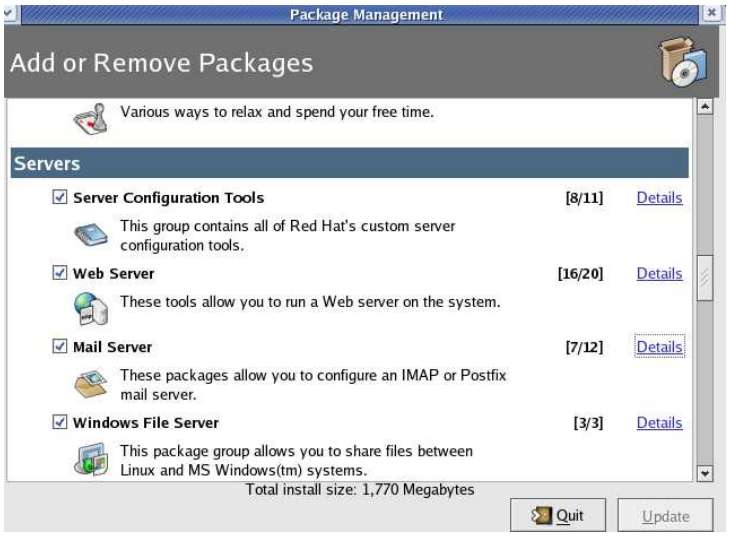

Figura 3.1: Selección de paquetes para Sendmail

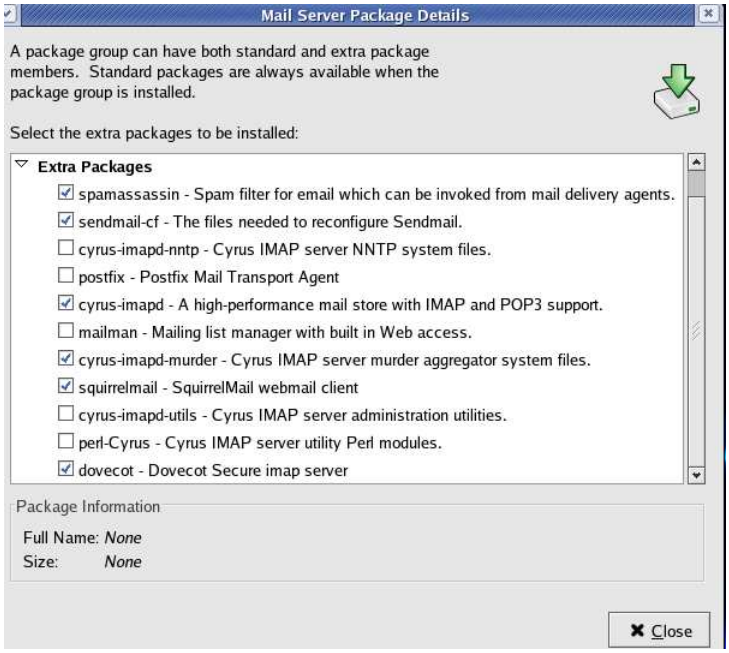

Figura 3.2: Selección de paquetes para Sendmail

Una vez que se han seleccionado todos los paquetes necesarios basta con hacer un *update* al sistema y este indicará cuales son los discos que requiere para poder hacer la instalación de dichos paquetes. Una vez terminado este proceso el sistema operativo contara con los paquetes necesarios para poder configurar el servicio.

Gestor de paquetes YUM

Otra de las formas es hacer uso del gestor de paquetes que contiene Fedora Core, que es el denominado YUM.

Mediante el uso de esta herramienta basta con invocar los paquetes deseados desde una línea de comandos y *YUM* se concectará a sus repositorios para localizar e instalar el paquete seleccionado, a continuación la línea de comandos:

yum install sendmail-cf sendmail dovecto m4 make cyrus-sasl cyrus-sasl-md5

3.1.2. Configuración de Sendmail

Sendmail utiliza el protocolo denominado *SMTP*, el cual es un protocolo estándar a nivel de aplicación utilizado para la transmisión de correo electrónico a través de una conexión *TCP/IP*. *SMTP* es el único protocolo utilizado para la tansmisión de correo electrónico en *Internet*.

Para determinar el servidor *SMTP* para un dominio dado se utilizan los registros MX *(Mail Exchanger)* en la zona de autoridad correspondiente al mismo dominio.

A continuación la figura 3.3 se ejemplifica una sesión *SMTP* entre el remitente (Cliente) y el destinatario (Servidor):

```
[root@fisicamail mail]# telnet 132.248.209.149 25
Trying 132.248.209.149...
Connected to 132.248.209.149 (132.248.209.149).
Escape character is '^]'.
220 fisicamail.fisica.unam.mx ESMTP ; Fri, 21 Nov 200812:13:52 -0600
HELO fisicamail.fisica.unam.mx
250 fisicamail.fisica.unam.mx Hello [132.248.209.149],pleased to meetyou
EHLO fisicamail.fisica.unam.mx
250-fisicamail.fisica.unam.mx Hello [132.248.209.149],pleased to meetyou
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-AUTH DIGEST-MD5 CRAM-MD5
250-DELIVERBY
250 HELP
quit
221 2.0.0 fisicamail.fisica.unam.mx closing connection
Connection closed by foreign host.
[root@fisicamail mail]#
```

Figura 3.3: Sesion de Telnet SMTP

Posteriormente se deben de dar de alta cuentas de usuarios y asignación de claves de acceso. Para la alta de cuentas de usuarios *Sendmail* utilizará *SASL* (ver apartado 3.1.5) para que pueda ser utilizado el método de autenticación para *SMTP*. El procedimiento es el siguiente:

useradd -s /sbin/nologin correo

El comando anterior indica que el usuario correo no debera de tener acceso a ningún interprete de comandos.

passwd correo

Comando para asignar claves de acceso al sistema para permitir autenticar a través de *SMTP* y de los protocolos *IMAP y POP3*.

saslpasswd2 correo

Comando para la asignación de claves de acceso para autenticar *SMTP* a través de métodos de cifrado *(DIGEST-MD5)*. Este proceso se realiza en sistemas con versión de *Sendmail* compilada contra *SASL-2* (Para el caso de *Fedora Core 3* este es el indicado). Finalmente se requiere de la activación del servicio como se muestra en la figura 3.4

```
[root@fisicamail mail]# chkconfig saslauthd on
[root@fisicamail mail]# service saslauthd start
Starting saslauthd:                                        [  OK  ]
[root@fisicamail mail]# service saslauthd status
saslauthd (pid 4287 4286 4285 4284 4283) is running...
[root@fisicamail mail]#
```

Figura 3.4: Autenticación con AUTHSASL

El siguiente paso es establecer los dominios a administrar. Estos dominios deberán ser agregados en el archivo denominado *local-host-names* ubicado en el directorio */etc/mail*. El dominio a administrar para este caso es *fisicamail.fisica.unam.mx* y deberá ir citado en el archivo antes mecionado.

El siguiente paso es una lista de control de accesos, el archivo que se encarga de administar esta lista es access ubicado en */etc/mail*. La lista debe tener una estructura tal y como se muestra en la figura 3.5

```
# Check the /usr/share/doc/sendmail/README.cf file for a description
# of the format of this file. (search for access_db in that file)
# The /usr/share/doc/sendmail/README.cf is part of the sendmail-doc
# package.
#
# by default we allow relaying from localhost...
localhost.localdomain           RELAY
localhost                       RELAY
127.0.0.1                       RELAY

#Direccion IP del Propio Servidor
132.248.209.149                 RELAY

#No se permite enviar correo fuera del propio servidor
132.248.209.149                 OK
```

Figura 3.5: Control de accesos

3.1.3. Configuración de dovecot

Dovecot es un servidor *IMAP y POP3* para sistemas *Linux*, y es la mejor opción para el manejo de ambos protocolos, ademas de que su instalacion es simple y no requiere de una administración especial ni de mucha memoria.

Una vez instalado *Dovecot* en el sistema proporcionará un archivo denominado *dovecot.conf* ubicdo en el directorio */etc*.

Se debe de editar este archivo para poder habilitar los servicios *POP3 e IMAP* como se muestra en la figura 3.6:

```
## Dovecot 1.0 configuration file

# Default values are shown after each value, it's not required to uncomment
# any of the lines. Exception to this are paths, they're just examples
# with real defaults being based on configure options. The paths listed here
# are for configure --prefix=/usr --sysconfdir=/etc --localstatedir=/var
# --with-ssldir=/usr/share/ssl

# Base directory where to store runtime data.
#base_dir = /var/run/dovecot/

# Protocols we want to be serving:
#  imap imaps pop3 pop3s
protocols = imap pop3

# IP or host address where to listen in for connections. It's not currently
# possible to specify multiple addresses. "*" listens in all IPv4 interfaces.
# "[::]" listens in all IPv6 interfaces, but may also listen in all IPv4
# interfaces depending on the operating system. You can specify ports with
"dovecot.conf" 488L, 20289C
```

Figura 3.6: Activación de los servicios de POP3 e IMAP

Una vez habilitados los servicios se agregan al arranque del sistema y se inicia como se muestra en la figura 3.7

```
[root@fisicamail etc]# chkconfig dovecot on
[root@fisicamail etc]# service dovecot start
Starting Dovecot Imap:                                     [  OK  ]
[root@fisicamail etc]# service dovecot status
dovecot (pid 4331) is running...
[root@fisicamail etc]#
```

Figura 3.7: Arranque del sistema con Dovecot

Posteriormente es necesario reiniciar el servicio de *sendmail* como se muestra en la figura 3.8

```
[root@fisicamail mail]# service sendmail restart
Shutting down sendmail:                                    [  OK  ]
Shutting down sm-client:                                   [  OK  ]
Starting sendmail:                                         [  OK  ]
Starting sm-client:                                        [  OK  ]
[root@fisicamail mail]#
```

Figura 3.8: Reinicio de Sendmail con Dovecot

Para depurar posibles errores en el arranque de los servicios se pueden examinar analizando el contenido de la bitácora que se genera en la instalación del servidor de correo denominado *maillog*. Un ejemplo de este proceso se muestra en la figura 3.9

```
[root@fisicamail mail]# tail -f /var/log/maillog
Nov 21 12:58:49 fisicamail dovecot: Killed with signal 15
Nov 21 12:59:03 fisicamail dovecot: Dovecot starting up
Nov 21 13:06:30 fisicamail sendmail[4390]: alias database /etc/aliases rebuilt by correo
Nov 21 13:06:30 fisicamail sendmail[4390]: /etc/aliases: 78 aliases, longest 10 bytes, 802 bytes total
Nov 21 13:06:30 fisicamail sendmail[4395]: starting daemon (8.13.1): SMTP+queueing@01:00:00
Nov 21 13:06:30 fisicamail sm-msp-queue[4405]: starting daemon (8.13.1): queueing@01:00:00
Nov 21 13:07:50 fisicamail sendmail[4434]: alias database /etc/aliases rebuilt by correo
Nov 21 13:07:50 fisicamail sendmail[4434]: /etc/aliases: 78 aliases, longest 10 bytes, 802 bytes total
Nov 21 13:07:50 fisicamail sendmail[4439]: starting daemon (8.13.1): SMTP+queueing@01:00:00
Nov 21 13:07:50 fisicamail sm-msp-queue[4449]: starting daemon (8.13.1): queueing@01:00:00
```

Figura 3.9: Análisis de maillog

3.1.4. Configuración de sendmail.mc (funciones de sendmail)

El archivo encargado de activar o desactivar las funciones para *sendmail* es el denominado **sendmail.mc** ubicado en la ruta */etc/mail*. Los parámetros a activar para el correcto funcionamiento de *sendmail* se deben de configurar dentro del archivo antes mencionado y son los siguientes:

confSMTP.
Este parámetro es el que permite establecer el mensaje de bienvenida cuando se establece la conexión con el servidor. Este parámetro debe de activarse de la siguiente forma en el archivo **sendmail.mc**.

define(‘confSMTP_LOGIN_MSG ’,‘$j ; $b ’)dnl

confAUTH_OPTIONS

Este parámetro viene activado por default y permite la autenticación a través del puerto 25 por cualquier método e incluso el método *PLAIN* que permite autenticación en texto simple, la actuaciones de este parámetro conlleva riesgos de seguridad. Este parámetro se activa de la siguiente forma:

define(‘confAUTH_OPTIONS ’, ‘A ’)dnl

DAEMON_OPTIONS

De modo automático *sendmail* escucha peticiones a través de la interfaz de retorno (loop back) *127.0.0.1* y no a través de otros dispositivos de la red. Basta con eliminar la restricción que tiene para la dirección ip *127.0.0.1* para que reciba correos desde la propia *LAN* e *Internet*. Este parámetro se activa de la siguiente forma:

DAEMON_OPTIONS(‘Port= smtp, Name=MTA ’)dnl

Cw.
Este parámetro hara que ***fisicamail.fisica.unam.mx*** el servidor de correo lo tome como un dominio local. El parámetro *Cw* debe de ir al final del archivo **sendmail.mc** y debe de escribirse sin espacios con C (mayúscula) y w (minúscula).

3.1.5. Squirrelmail

Squirrelmail es un *software* robusto y funcional que permite al usuario final acceder a su correo electrónico desde cualquier navegador ya que muestra las paginas en modo *HTML 4.0* sin la necesidad de emplear *JavaScript*.

Squirrelmail esta escrito en *PHP4* y cumple con todos los estándares para correo electrónico a través de su interfaz *httpd* y el soporte de los protocolos *POP3 e IMAP*.

Instalación de Squirrelmail.
La instalación de *Squirrelmail* se puede realizar mediante el gestor de paquetes *YUM* o mediante el disco de instalación del sistema operativo. Cuando se realiza la instalación mediante *YUM* se específica la siguiente línea de comandos:

yum -y install squirrelmail httpd

Para la instalación con el disco del sistema operativo se debe de seleccionar en el área de *agregar/quitar* el paquete de *squirrelmail* ubicado en la sección de paquetes de *Mail Server*, y en la sección de Web Server seleccionar el paquete de *httpd*.

Posterior a la selección de los paquetes se da la opción de actualizar y el sistema indicará los discos que requiere.

Configuración.
El primer paso para la configuración de *Squirrelmail* es ejecutar el archivo **config.pl** ubicado en la ruta */usr/share/squirrelmail/config*.

cd /usr/share/squirrelmail/config

./config.pl

La ejecución del comando anterior nos mostrará una interfaz de configuración del servidor y de parámetros para personalizarlo. La figura 3.10 muestra ésta interfaz gráfica.

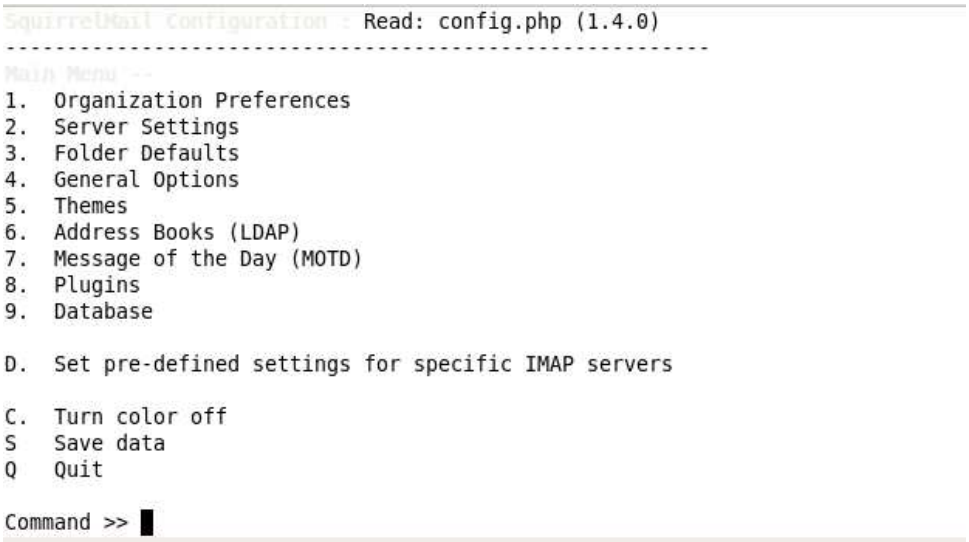

Figura 3.10: Interfaz de configuración de Squirrelmail

En las opciones posteriores aparece un menú en el cual se da la opción de personalizar nuestro correo con los datos particulares, como ejemplo *nombre de la organización, logotipo de la organización, selección del lenguaje,etc*, la figura 3.11 muestra dicho menú y su configuración para este proyecto.

Figura 3.11: Interfaz de configuración de Squirrelmail

Posteriormente se debe declarar el dominio que se va a utilizar para nuestro correo, para este caso el dominio es el siguiente:**fisicamail.fisica.unam.mx.**

Es importante decir que únicamente se debe de declarar el dominio que se usará, como se muestra en la figura 3.12.

Figura 3.12: Activación del dominio de correo

Para las opciones de las carpetas se personaliza como lo considere el administrador. Finalmente se configuran y se dan de alta los *plug-ins* que se requieran (el administrador decide que plug-ins se daran de alta).

En la figura 3.13 se muestran los *plug-ins*[1] que se dieron de alta para este proyecto.

[1]Es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica

```
SquirrelMail Configuration : Read: config.php (1.4.0)
---------------------------------------------------------
Plugins
  Installed Plugins
    1. delete_move_next
    2. squirrelspell
    3. newmail
    4. sent_subfolders
    5. translate
    6. message_details
    7. abook_take
    8. filters
    9. mail_fetch

  Available Plugins:
    10. administrator
    11. fortune
    12. listcommands
    13. bug_report
    14. calendar
    15. spamcop
    16. info

R   Return to Main Menu
C.  Turn color off
S   Save data
Q   Quit
```

Figura 3.13: Activación de plug-ins de Squirrelmail

Una vez terminada la configuración y personalización del servicio se procede a iniciarlo, como ilustra la figura 3.14:

```
[root@fisicamail config]# chkconfig dovecot on
[root@fisicamail config]# service dovecot restart
Stopping Dovecot Imap:                                     [  OK  ]
Starting Dovecot Imap:                                     [  OK  ]
[root@fisicamail config]# service dovecot status
dovecot (pid 4481) is running...
[root@fisicamail config]#
```

Figura 3.14: Acivación del servicio con Squirrelmail

Ya iniciado el servicio basta con reiniciar el servidor web apache para habilitar los servicios de *IMAP y POP3*. El comando que hace esto posible es*service httpd start* y la salida de dicho comando se muestra en la figura 3.15.

```
[root@fisicamail config]# service httpd start
Starting httpd:                                            [  OK  ]
[root@fisicamail config]# netstat -an | grep 80 | grep LISTEN
tcp        0      0 :::80                       :::*                        LISTEN
[root@fisicamail config]#
```

Figura 3.15: Arranque del servicio web

Finalmente se invoca mediante una URL (http://localhost/webmail)al servicio, como se muestra en la figura 3.16:

Figura 3.16: Arranque de Squirrelmail vía Web

3.1.6. Instalación de Sebek cliente sobre Fedora Core

A diferencia de la instalación y configuración de los paquetes anteriores, *Sebek* es diferente ya que no viene contenido ni en el disco de instalación del sistema operativo ni en sus repositorios. La instalación de *Sebek* sigue una serie de procedimientos los cuales deben de seguirse en orden ya que realizaremos compilación de *Kernel* para la inserción de *Sebek cliente* como módulo al kernel.

El primer paso a considerar en la instalación de *Sebek* es descargar el código fuente del *kernel* que se este ocupando en el sistema operativo, para este caso en particular se esta empleando una versión de *kernel 2.6.9-1.667*, por lo que se tiene que descargar las fuentes para dicho *kernel*, como se muestra a continuación:

cd /usr/src

wget http://ftp.riken.jp/Linux/fedora/core/3/SRPMS/kernel-2.6.9-1.667
.src.rpm

Una vez descargadas las fuentes se procede con su instalación en el sistema como se indica:

rpm -ivh kernel-2.6.9-1.667.src.rpm

cd redhat/SPECS

rpmbuild -bp –target=i686 kernel-2.6.spec

*cp /usr/src/redhat/BUILD/kernel-2.6.9/linux-2.6.9/net/packet/af_packet.c\
/lib/modules/2.6.9-1.667/build/net/packet/*

El siguiente paso es descargar la versión adecuada de *Sebek-Cliente* para su instalación en este sistema.

cd /usr

wget http://www.honeynet.org/tools/sebek/3/sebek-lin26-3.1.2b.tar.gz

tar zxvf sebek-lin26-3.1.2b.tar.gz

cd sebek-lin26-3.1.2b

./configure

make

Una vez instalado *Sebek Cliente* se debe de realizar la configuración del mismo para adaptarlo como módulo de *kernel* de *Fedora Core* (ver apartado 3.3).

3.2. Instalación y configuración del Servidor de Base de Datos

La instalación y configuración del Servidor de Base de Datos se hará sobre un Sistema *Linux Ubuntu 7.10*. Los requerimientos para la construcción de este servidor son las siguientes:

Mysql-Server-5.0

Mysql-Client-5.0

Sebek-Cliente

3.2.1. Instalación de paquetes para el Servidor de Base de Datos

Para la instalación de los paquetes requeridos existen dos formas: mediante el uso del disco de instalación de *Ubuntu 7.10* o el uso del gestor de paquetes *apt*.

Uso del disco de instalación

Para instalar paquetes usando el disco de instalación del sistema operativo se requiere de ir a la pantalla de *agregar/quitar* paquetes y seleccionar en esta los paquetes adecuados, como se observa a continuación.

Gestor de paquetes apt

Otra de las formas es hacer uso del gestor de paquetes que contiene Ubuntu 7.10, que es el denominado apt.

Mediante el uso de esta herramienta basta con invocar los paquetes deseados desde una línea de comandos y *apt* se concectará a sus repositorios para localizar e instalar el paquete seleccionado, a continuación la expliación:

sudo apt-get install mysql-server-5.0

Con esta instrucción se procederá con la instalación de *Mysql-Server*. Durante dicha instalación nos pedirá el *password* para el usuario *root*, en caso contrario finalizada la instalación lo realizaremos de forma manual, además se levantará la base de datos como se indica:

*sudo mysqladmin -u root -p password (**password**)*

mysqld_safe

Una vez terminada la instalación de la base de datos, se procede con la creación de la base de datos.

3.2.2. Instalación de Sebek-Cliente sobre Ubuntu

Para la instalación de *Sebek-Cliente* en *Ubuntu* se requieren de los siguientes paquetes:

- **gcc (Compilador)**
- **make**
- **autoconf**
- **automake**
- **patch**
- **lib6-dev (Librerías para Gcc)**
- **linux-headers-server (Fuentes del Kernel)**
- **subversion (Para descarga de código)**

Esta instalación de paquetes se realizará mediante el uso del gestor de paquetes *apt* como se muestra a continuación:

sudo apt-get install gcc make autoconf automake patch libc6-dev linux-headers-server subversion

Terminada la instalacion de los paquetes se descarga la versión adecuada de Sebek para *Ubuntu*. Para este caso se descargara mediante el uso de *subversion* una versión probada para el *kernel 2.6.22* que es con el que cuenta nuestro sistema operativo.

svn co –username fisica https://projects.honeynet.org/svn/sebek/linux-2.6/
trunk sebek

Como se puede observar en el comando anterior, se dió de alta el usuario llamado *fisicatest* en el siguiente sitio del *Honeynet Project https://projects.honeynet*
.org/sebek/register.

El fin de realizar este registro es para poder descargar la versión pre-compilada de *Sebek* para el sistema operativo (Ubuntu 7.10), es decir, con las fuentes de *Kernel* necesarias para su compilación en el sistema.

Ya ejecutada la instrucción anterior el *password* requerido es *fisicatest* para que pueda validar al usuario física y así descargar los archivos fuentes de *Sebek*, como se muestra en la figura 3.17

```
A    sebek/Makefile.in
A    sebek/AUTHORS
A    sebek/configure.in
A    sebek/ChangeLog
A    sebek/depcomp
A    sebek/src
A    sebek/src/sebek.h
A    sebek/src/config.h.in
A    sebek/src/syscall.h
A    sebek/src/util.h
A    sebek/src/af_packet.diff
A    sebek/src/filter.c
A    sebek/src/net.c
A    sebek/src/sebek.c
A    sebek/src/filter.h
A    sebek/src/syscall.c
A    sebek/src/net.h
A    sebek/src/Makefile
A    sebek/src/util.c
A    sebek/config.guess
A    sebek/config.sub
A    sebek/README
A    sebek/gen_fudge.pl
A    sebek/acconfig.h
A    sebek/BUILD
A    sebek/filter.txt
A    sebek/INSTALL
A    sebek/compile_filter.pl
A    sebek/COPYING
A    sebek/Makefile.am
A    sebek/missing
A    sebek/NEWS
A    sebek/aclocal.m4
A    sebek/install-sh
A    sebek/sbk_install.sh
Checked out revision 20.
```

Figura 3.17: Obtención de código por SVN

Posterior a la descarga de los archivos fuente de *Sebek* se genera una carpeta llamada *Sebek* en donde se localizan los archivos para la instalación.

cd /usr/sebek

./configure disable-raw-socket-replacement

make

Al terminar la instalación de *Sebek-Cliente* se procede con la configuración para poder realizar la conectividad con *Sebek-Server*.

3.3. Configuración de Sebek cliente en Ubuntu

En este apartado se realizará la configuración de *Sebek-Cliente* en sistemas *Fedora-Core y Ubuntu*. La instalación de *Sebek-Cliente* en ambos sistemas es diferente, aunque la configuración es la misma para ambos sistemas.

Al concluir la compilación se genera un archivo llamado *sebek_lin26-3.2.0b_bin\
.tar.gz* el cual se movera hacia la ruta de */tmp*, ya que se eliminará después de su configuración.

mv /usr/Sebek/sebek_lin26-3.2.0b_bin.tar.gz /tmp

tar zxvf sebek_lin26-3.2.0b_bin.tar.gz

cd sebek_lin26-3.2.0b_bin

Una vez que se ha descomprimido el archivo *sebek_lin26-3.2.0b_bin.tar.gz* se ingresa a su carpeta y se deben de mostrar los siguientes archivos:

- **compile_filter.pl**
- **databases.o**
- **filter.of**
- **filter.txt**
- **parameters.sh**
- **README**
- **sbk_install.sh**
- **sbk.ko**

El siguiente paso es editar el archivo **sbk_install.sh** el cual contiene los parámetros para que *Sebek-cliente* se comunique con *Sebek-Server*. Los parámetros que se van a modificar son los siguientes:

INTERFACE.

Este parámetro contiene el nombre genárico de la interfaz por donde esta transmitiendo el servidor, para nuestro caso es *eth0*. Cabe aclarar que el nombre de la interfaz puede variar dependiendo la arquitectura del sistema.

DESTINATION_IP

Este parámetro contiene la dirección ip del servidor remoto *(Sebek-Server)*,por otro lado este valor no necesariamente debe de ser correcto, ya que se puede usar una ip falsa y no afectará la transmisión de los datos hacia el *Sebek-Server*.

DESTINATION_MAC

Este parámetro contiene la dirección *MAC* del servidor remoto *(Sebek-Server)*. A diferencia del parámetro *DESTINATION_IP* éste si debe de tener el valor correcto, es decir, debe contener exactamente la dirección *MAC* de *Sebek-Server*.

Un punto muy importante dentro de este parámetro es que si el servidor de *Sebek-Server* se encuentra dentro de la misma red *LAN* el parámetro *DESTINATION_MAC* deberá contener la dirección *MAC* del *Sebek-Server*. Para el caso de que *Sebek-Server* se encuentre fuera de la red LAN de la *Honeynet* el

parámetro *DESTINATION MAC* deberá contener la dirección *MAC* del *GATEWAY* de la *Honeynet.*

SOURCE_PORT.

Este parámetro define el puerto UDP por el cual *Sebek* enviará los datos. Este parámetro es muy importante cuando varios *Honeypots* se encuentran detrás de un mismo NAT ya que es la única forma de distinguir a los hosts de la *Honeynet*. El valor por default de este parámetro es el 1101, pero este valor es modificable.

DESTINATION_PORT.

Este parámetro es de vital importancia, ya que es el que define el puerto destino por el cual *Sebek* enviará datos recolectados hacia *Sebek-Server*. Es muy importante que este parámetro contenga el mismo valor en todos los *Honeypots.*

MAGIC_VAL.

Este parámetro define un valor numérico, el cual se usará para marcar los paquetes enviados por cada *Honeypot* hacia el *Sebek-Server* y así puedan reconocerse, ya que cabe recordar que *Sebek-cliente* envía sus datos capturados por medio del protocolo UDP de manera oculta para el usuario final.

KEYSTROKE_ONLY

Este parámetro permite capturar únicamente pulsaciones de teclas con un valor 1 o capturar todo tipo de datos generados en el *Honeypot* con un valor de 0. Es altamente recomendado usar este parámetro con un valor 1 para mayor nitidez en el análisis posterior de los datos.

SOCKET_TRACKING

Este parámetro se activa con un valor 1 y se desactiva con un valor 0 y es empleado por si requiere capturar conexiones establecidas vía sockets o no. Por default viene configurado con valor 1.

WRITE_TRACKING.

Este parámetro de igual forma que los dos anteriores maneja valores binarios (1 y 0) y es empleado para capturar las pulsaciones de teclas en un *Honeypot*, además de mostrar la salida de dichas pulsaciones en pantalla.

Este parámetro actualmente se encuentra en una fase de pruebas por lo que no es recomendado activarlo. Nota: Para fines de este proyecto se configuró *Sebek-cliente* con este parámetro activo y experimentamos un congelamiento del sistema operativo, por lo cual fue necesario reinstalar el cliente.

TESTING

Este parámetro de valores booleanos (1 y 0) es para definir se el módulo que se instalar de *Sebek-cliente* se hará en forma de testing o no. Si se activa en modo prueba el intruso puede percatarse de la presencia de *Sebek*, poniendo así en riesgo la integridad de la *Honeynet.*

MODULE_NAME.

Este parámetro define el nombre con el cual se instalará el módulo de *Sebek-cliente* en el sistema.

Una vez terminada la configuración de *Sebek-cliente* se procede con la instalación del módulo de la siguiente forma:

sudo ./sbk_install.sh

Si no se presenta ningún problema con la instalación del módulo se da por terminada la instalación de *Sebek-cliente* y el paso final es borrar el archivo *sebek-lin26-3.2.0b-bin* y el directorio de *sebek* generado en la descarga de los archivos fuente, con el fin de eliminar huellas que puedan hacer que el intruso se percate de que esta en un *Honeypot*.

rm -rf /usr/sebek

rm -rf /tmp/sebek_lin26-3.2.0b_bin

3.4. Instalación de Sebek cliente en Windows 2000

A continuación se detallará la instalación y configuración de *Sebek - Cliente* sobre *Windows 2000 Profesional*. Es muy importante decir que actualmente la versión de *Sebek* para *Windows* únicamente funciona y es estable en *Windows 2000* ya que se realizaron pruebas de instalación y estabilidad sobre *Windows 2003, 2005 y 2008* no resultando efectivas porque la instalación provoca un colapso en el sistema operativo haciendolos inoperables.

La versión de *Sebek* para *Windows* contiene 2 archivos llamados *setup.exe* y *wizard.exe*, los cuales permiten la instalación y configuración respectivamente de *Sebek* en el sistema.En la figura 3.18 se muestra la pantalla inicial del proceso de instalación de *Sebek* mediante la ejecución de setup.exe.

Figura 3.18: Instalación de Sebek sobre Windows 2000

Como se puede observar la forma de instalación con respecto de *Linux* es muy diferente ya que en este caso únicamente se le indica a *Windows* la ruta en donde se generarán los archivos de configuración de *Sebek*. La figura 3.19 muestra este proceso.

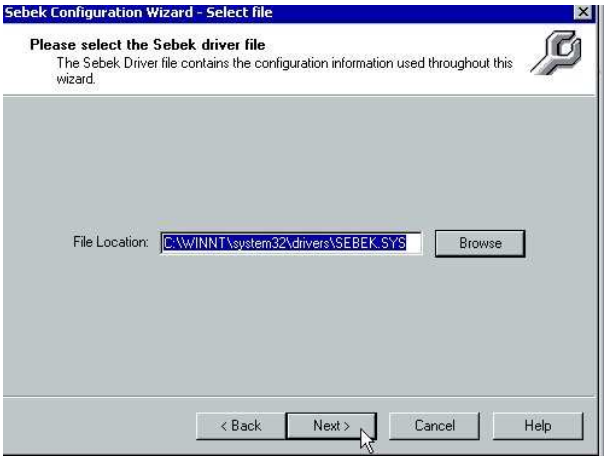

Figura 3.19: Instalación de Sebek sobre Windows 2000

Finalmente y si la instalación del *driver de Sebek* fue exitosa se concluye con la instalación de *Sebek - Cliente* sobre Windows 2000. La figura 3.20 muestra la finalización de la instalación.

Figura 3.20: Instalación de Sebek sobre Windows 2000

Una vez instalado *Sebek* se procede con la configuración del mismo. Para configurar *Sebek* sobre *Windows 2000* basta con ejecutar el programa *wizard.exe* el cual nos guiará paso a paso con la configuración de *Sebek*. En la figura 3.21 se muestra el comienzo de la configuración mediante la ejecución de *wizard.exe*.

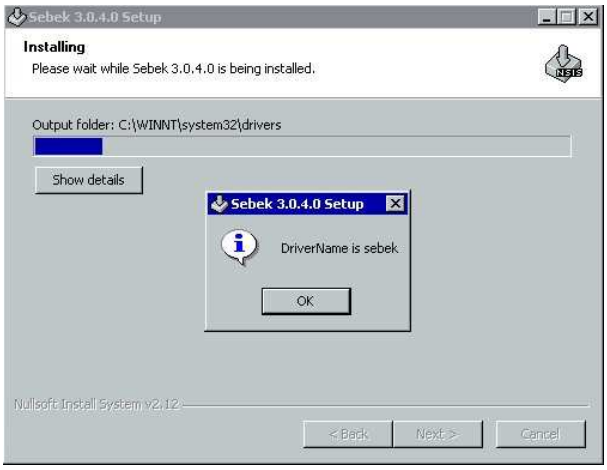

Figura 3.21: Configuración de Sebek sobre Windows 2000

Una vez que se le ha asignado un nombre al driver previamente instalado se procede con la configuración de los siguientes parámetros: *Destination MAC, Destination IP, Destination Port, Magic Value, Network Interface y Configuration File Name*. La pantalla 3.22 muestra la confirmación de estos parámetros de configuración.

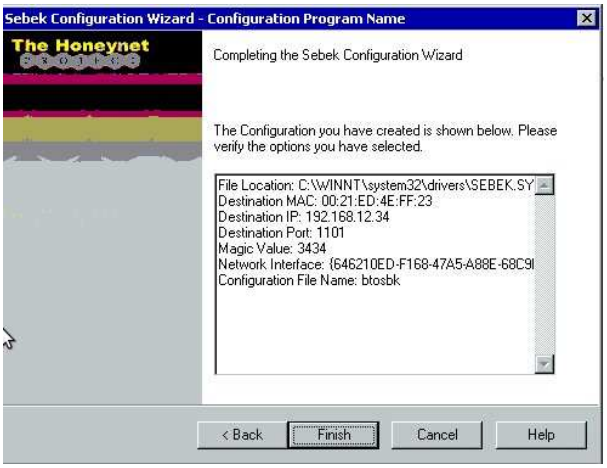

Figura 3.22: Configuración de Sebek sobre Windows 2000

Finalmente se mostrará una ventana con el anuncio que *Sebek* fue configurado exitosamente y que esta listo para su uso. Cabe mencionar que una vez instalado y configurado ya no es posible volver a modificar los parámetros ya que una de las características de *Sebek* es ocultarse en el sistema para que no sea visible a los intrusos, por lo que no será detectado en ningún momento.

3.5. Honeywall sobre OpenBSD

OpenBSD es un sistema operativo tipo *Unix* cuyo propósito es la seguridad de las redes y la información (ver cap.1). Este sistema operativo se configurará como un *Honeywall* dentro de la *Honeynet* para realizar el control de datos entrantes y salientes de la *Honeynet*.

El *Honeywall* es uno de los dispositivos más importantes dentro de la *Honeynet*, ya que será el encargado de controlar todas las actividades realizadas por los intrusos, con el fin de no permitir ataques

a terceros. Además de que es uno de los sistemas que no debe ser descubierto por un intruso ya que se corre el riesgo de perder el control sobre la red completa.

A este respecto se realizará la configuración del *Honeywall* sobre *OpenBSD* en modo *Bridge* con el fin de que no sea ruteable hacia *Internet* y el flujo de los datos entrantes y salientes sea totalmente transparente para el intruso y así garantizar el control sobre la *Honeynet*.

3.5.1. Instalación y configuración de OpenBSD

Como ya se mencionó anteriormente, *OpenBSD* será configurado en modo puente (bridge), por lo cual en la instalación las tarjetas de red no deben de tener direcciones IP asignadas, simplemente deben levantarse como se muestra en la figura 3.23

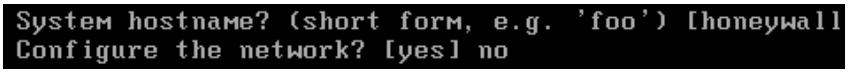

Figura 3.23: Configuración de las interfaces de red en OpenBSD

Otro punto importante es la instalación de los paquetes para el sistema operativo. *OpenBSD* por ser un sistema operativo dedicado a la seguridad no recomienda instalar las interfaces gráficas ya que no es requerida. A continuación se muestran los paquetes a instalar en la figura 3.24

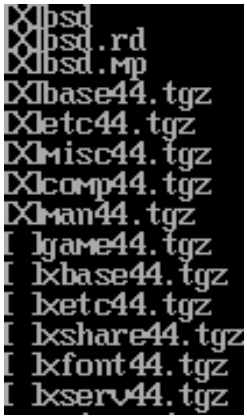

Figura 3.24: Selección de paquetes para el Honeywall

Una vez que se han seleccionado los paquetes requeridos se procede con la instalación de dichos paquetes. La figura 3.25 muestra el proceso de instalación:

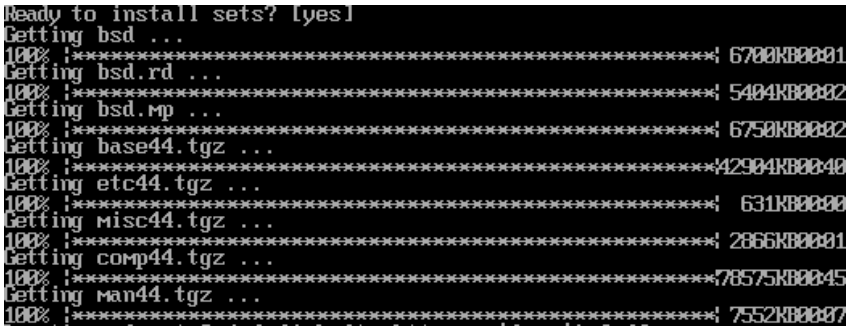

Figura 3.25: Proceso de instalación de paquetes del Honeywall

Una vez que se ha concluido la instalación de los paquetes requeridos y después de haber configurado al sistema operativo con la región de México, se finaliza la instalación, como se muestra en la figura 3.26

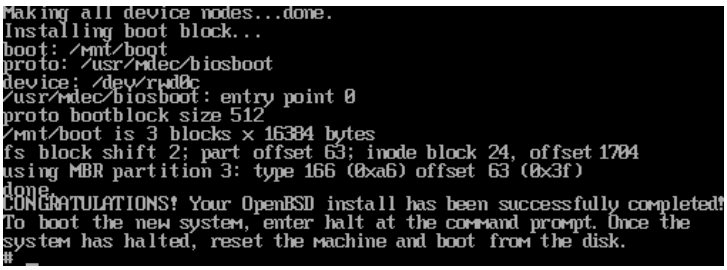

Figura 3.26: Instalación finalizada del Honeywall

3.5.2. Configuración de OpenBSD en modo bridge

El primer punto es la activación de *Packet Filter (PF)* para garantizar que se aplicarán las reglas de filtrado hacia los *Honeypots*. Este proceso consiste en cambiar el valor de la variable *PF* dentro del archivo **rc.conf** ubicado en la ruta */etc* como se indica:

PF = NO

PF = YES

La siguiente variable que se debe activar es **net.ip.inet.forwarding=0** para poder garantizar que los paquetes procesados por las reglas de filtrado sean dirigidos correctamente a sus destinos, como se muestra:

net.inet.ip.forwarding=0

net.inet.ip.forwarding=1

Como segundo paso se debe crear el dispositivo virtual para la configuración del *Bridge*, este proceso se lleva acabo mediante las siguientes líneas:

touch /etc/bridgename.bridge0

Agregar las siguientes líneas al archivo **bridgename.bridge0** creado:

add xl0 add xl1 up

donde *xl0 y xl1* son las interfaces de red reconocidas por el sistema operativo, y es en estas dos interfaces donde se generará el *bridge*.

Finalmente se realiza la activación y ejecución del *bridge* antes creado y configurado como se muestra:

ifconfig bridge0 up

brconfig bridge0 up

La figura 3.27 muestra la inicialización del *bridge* en el sistema opertativo.

Figura 3.27: Arranque del bridge en el Honeywall

Como último paso se edita el archivo **pf.conf** para asignar las reglas de filtrado apropiadas para el *bridge* creado. La figura 3.28 muestra dicha configuración.

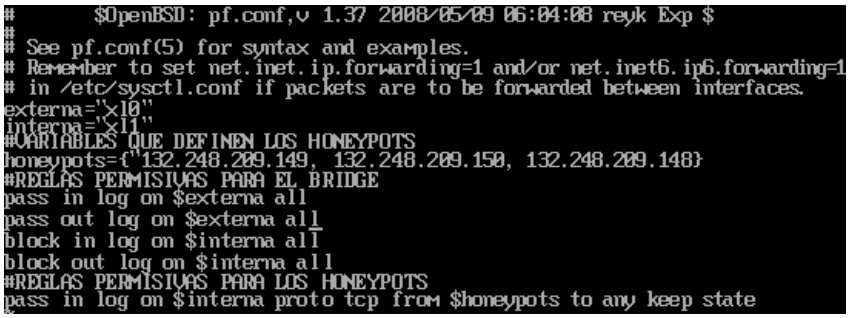

Figura 3.28: Reglas básicas para OpenBSD modo puente

Donde las variables externa e interna contienen el nombre canónico de las interfaces de red detectadas por el sistema y definidas por nosotros como *xl0* la externa y *xl1* la interna.

Se define una variable denominada *honeypots* la cual contiene el *pool* de direcciones IP que perteneces a cada uno de los *honeypots*.

A continuación se asignan las reglas permisivas para que el *bridge* pueda ser funcional. Estas reglas especifican que todo tráfico proveniente de *Internet* es aceptado sólo por la interfaz externa y por la interfaz interna se bloqueara todo tráfico entrante o saliente a menos que las políticas de filtrado así lo permitan.

Como punto final se les asignan políticas permisivas a los *honeypots* para que puedan tener acceso a *Internet* y así brindar libertad a los intrusos.

Es importante hacer notar que todas las reglas permisivas cuentan con la clausula *log* la cual nos registrará cualquier actividad de los *honeypots* que cumplan con esa regla de filtrado y que es lo que nos ayudará a tener el control de las conexiones realizadas por los intrusos.

3.6. Sistema Detector de Intrusos

La presencia de los IDS *(Intrusion Deteccion Systems)* en una *Honeynet* son de vital importancia ya que es un sensor que actúa capturando y alertando al administrador de la red de ataques contra sistemas externos a la *Honeynet* e incluso entre los mismos sistemas trampa *(Honeypots)*.

En este proyecto se utiliazará *Snort* como sensor de alarmas. *Snort* es un sistema de prevención y detección de intrusos de código abierto, el cual en una base de datos que contiene patrones de ataques conocidos, por lo que al momento de que un ataque es analizado por *Snort* y este es comparado con dicha base y si el patron analizado coincide con el que posee *Snort* en su base de datos lanza una alarma al administrador avisándole que se esta intentando llevar a cabo un ataque en la red.

3.6.1. Instalación de IDS Snort

El sistema de prevención y detección de intrusos *Snort* se instalará como una segunda instancia en el sistema de *Syslog* de la *Honeynet*, compartiendo los recursos con *Sebek-Server* (ver apartado 3.6) en un Sistema Operativo *Ubuntu 7.10*, con el fin de que todas las bitácoras, alarmas y captura de datos quede centralizada en un sólo servidor remoto para su posterior análisis.

Para la instalación de *Snort* se requiere de los siguientes paquetes:

- **libpcap0.8-dev**
- **mysql-server-5.0**
- **mysql-client-5.0**
- **libmysqlclient15-dev**
- **flex**
- **bison**
- **apache2**
- **libapache2-mod-php5**
- **php5-gd**
- **php5-mysql**
- **libphp-adodb**
- **php-pear**
- **gcc**
- **libc6-dev**
- **g++**
- **php5**

El primer paso para proceder con la instalación es obtener los privilegios necesarios mediante la siguiente línea de comandos:

sudo -i (ingresar el password de root)

La instalación de los paquetes se realizará mediante el uso del gestor de paquetes APT y mediante la herramienta *wget*.

apt-get install flex gcc g++ libc6-dev bison libpcap0.8-dev php5

El siguiente paso es crear un directorio para concentrar ahí todos los paquetes adquiridos para la instalación y así realizar una instalación limpia.

mkdir InstalacionSnort

cd InstalacionSnort

Dentro de este directorio almacenaremos todos los paquetes a instalar mediante la herramienta *wget*.

wget http://www.snort.org/dl/snort-2.8.0.tar.gz

Una vez descargado el archivo se descomprime dentro del directorio creado anteriormente:

tar -zxvf snort-2.8.0.tar.gz

Para el buen funcionamiento de *Snort* debemos de contar con el set de reglas apropiadas para la versión que se descargo anteriormente y así pueda realizar correctamente el monitoreo de patrones de ataques contra los sistemas de la *Honeynet*.

*wget http://www.snort.org/pub-bin/downloads.cgi/Download/vrt_pr \
/snortrules-pr-2.4.tar.gz*

De igual forma se decomprime el archivo dentro del directorio creado para este fin.

tar -zxvf snortrules-pr-2.4.tar.gz

El siguiente paquete a descargar es el de *pcre (Perl Compatible Regular Expressions)* ya que las reglas de *snort* emplea expresiones regulares en sus reglas de análisis de tráfico.

wget ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-7.4.tar.gz

tar -zxvf pcre-7.4.tar.gz

Para que *snort* pueda mostrar su análisis de red a través de una interfaz gráfica se requiere del paquete llamado *BASE (Basic Analisis and Security Engine)*.

*wget http://downloads.sourceforge.net/secureideas/base-1.3.8.tar.gz?modtime\
=1183896336 big mirror=0*

tar -zxvf base-1.3.8.tar.gz

El paquete *BASE* requiere de una librería para manejar la base de datos mediante la interfaz gráfica en PHP llamada *ADODB*.

*wget http://downloads.sourceforge.net/adodb/adodb502a.tgz?modtime\
=1191343792 big mirror=0*

tar -zxvf adodb502.tar.gz

En este momento tenemos todos los paquetes y librerías necesarias para la instalación de *snort*. El direcotrio *InstallSnort* debe tener la siguiente estructura:

- **adodb5**
- **base-1.3.8**
- **pcre-7.4**
- **snort-2.8.0**

Los archivos ***.tar.gz** seran eliminados para mantener una instalación limpia.

El siguiente paso es proceder con la configuración de todos los paquetes que se descargaron anteriormente.

3.6.2. Configuración de Snort

Para la instalación y configuración de *snort* se realizarán los siguientes pasos:

cd InstallSnort

./configure -enable-dynamicplugin –with-mysql

make

make install

Adicionalmente se deben de crear los directorios */etc/snort* , */etc/snort/rule* y */var/log/snort* ya que se copiarán archivos de confirugarción en ellos.

mkdir /etc/snort

mkdir /etc/snort/rules

mkdir /var/log/snort

Una vez creados los directorios se procede a copiar archivos de configuración de los paquetes descargados enteriormente.

cd /root/InstallSnort/snort-2.8.0/rules

*cp -p * /etc/snort/rules*

cd /root/IsntallSnort/snort-2.8.0/etc

*cp -p * /etc/snort/*

Además copiremos la librería de **libpcre.so.0** a la ruta de */usr/lib*.

cp -p /usr/local/lib/libpcre.so.0 /usr/lib

Finalmente se debe de modificar algunas variables del archivo de configuración de *snort* llamado **snort.conf** ubicado en la ruta */etc/snort/snort.conf*. Dentro de este archivo se dará de alta el segmento de red en la cual *snort* monitoreara todo el tráfico de la *Honeynet*. Las variables a modificar son las siguientes:

```
var HOME_NET 132.248.209.0/24
```

La variable anterior se le da el valor de la red a la que snort monitoreara el tráfico.

```
var EXTERNAL_NET !HOME_NET
```

A la variable $EXTERNAL_NET se le da como valor la variable HOME NET que contiene ya como valor el segmento de red a monitorear.

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

La variable $PATH_RULE leerá el archivo de rules que contiene todas las reglas con las que snort se basa para generar las alarmas.

Finalmente se configura el acceso hacia la base de datos, para que *snort* tenga la posibilidad de conectarse a la base de datos de manera automática y así ir almacenando alertas detectadas y tomando alarmas de la misma para generar alertas.

output database: log, mysql, user=snort password= <password-snort-db> dbname=snort

3.6.3. Configuración de Mysql

En este apartado se iniciará la base de datos que usará *snort* para su buen funcionamiento. El primer paso a realizar es ingresar a la base de datos con el usuario root para obener todos los privilegios.

```
mysql-u -root -p
password: <ingresar password de root
```

Una vez que se ha ingresado se procede con la creación de la base de datos *snort*.

```
mysql> create database snort;
mysql> quit
```

Ya creada la base de datos para *snort*, se procede con la creación de las tablas con las que *snort* gestionará las alarmas de detección de intrusos. Las tablas requeridas por *snort* vienen ya definidas en un directorio llamado **schemas**, por lo que cargaremos el archivo llamado **create_mysql** ubicado en la ruta */root/InstallSnort/snort-2.8.0/schemas/*

```
mysql -D snort -u root -p < /root/InstallSnort/snort-2.8.0/schemas/create_mysql}
```

El paso siguiente es editar el archivo llamado **web-misc-rules** ubicado en la ruta */etc/snort/rules/web-misc.rules*. En este archivo se deben de deshabilitar (o eliminar) las líneas 97, 98 y 452 ya que estas líneas contienen reglas que no son soportadas por la configuración de *snort*.

Las líneas anteriores son:

```
# alert tcp EXTERNAL_NET any -> HTTP_SERVERS HTTP_PORTS (msg:"WEB-MISC ///cgi-bin access";\
 flow:to_server,established; uricontent:" cgi-bin"; nocase; rawbytes; reference:nessus\
 ,11032; classtype:attempted-recon; sid:1143; rev:7;)

# alert tcp EXTERNAL_NET any -> HTTP_SERVERS HTTP_PORTS (msg:"WEB-MISC /cgi-bin/// access";\
 flow:to_server,established; uricontent:" cgi-bin///"; nocase; rawbytes; reference:nessus,\
 11032; classtype:attempted-recon; sid:1144; rev:7;)\

# alert tcp EXTERNAL_NET any -> HOME_NET 8090 (msg:"WEB-MISC TrackerCam ComGetLogFile.php3\
```

```
log information disclosure"; flow:to_server,established; content:"/ComGetLogFile.php3";\
 nocase; pcre:"fn=Eye\d{4}_\d{2}.log/Rmsi"; reference:bugtraq,12592; reference:cve,\
 2005-0481; classtype:web-application-activity; sid:3545; rev:2;)
```

Como paso siguiente es probar las configuraciones realizadas anteriormente y comprobar que funcionan en su totalidad, para esto debemos ejecutar *snort* y si las configuraciones están correctas recibiremos la imágen de snort.

```
snort -c /etc/snort/snort.conf
```

```
Nota: para terminar con la prueba oprimir ctrl + c
```

El siguiente paso es la configuración de *BASE y Apache2* con PHP, no olvidar que tanto *Apache2* como PHP hasta este momento han sido ya instalados mediante el gestor de paquetes apt como se indica en el inicio de este apartado.

Primero se probará que ya contamos con PHP haciendo una prueba de funcionalidad. Para dicha prueba generaremos un archivo llamado **pruebaphp**\
.php en la ruta */var/www*. La función de php que se mandar invocar dentro de este archivo para verificar la funcionalidad y los módulos con los que cuenta php es *phpinfo();*.

```
touch /var/www/pruebaphp.info
```

```
echo <?php phpinfo();?> > /var/www/pruebaphp.php
```

Como paso siguiente reiniciamos el demonio de *apache* para que tome los cambios realizados. Al invocar la función anterior obtenemos con respuesta lo siguiente:

```
/etc/init.d/apache2 restart
```

Para *Adodb y Base* únicamente debemos mover los archivos de configuración de ambos al directorio */var/www/*.

```
mv /root/InstallSnort/adodb5 /var/www
```

Para el caso de *BASE* crearemos un directorio llamado **honeymonitor** dentro de la ruta */var/www/* con el fin de distinguir y tener por separado nuestro *IDS*.

```
mkdir /var/www/honeymonitor
```

```
mv /root/InstallSnort/base-1.3.8 /var/www/honeymonitor
```

Una vez que se han movido los archivos de configuración se edita el siguiente archivo ubicado en */var/www/honeymonitor/base/setup/setup1*, y localizamos la línea que tiene la instrucción *base header* y cambiarla por *header*, ya que en esta versión de php y de *adodb* ya no es soportada.

Por otro lado como se requieren de gráficos para la generación de barras de estadísticas en php, para la presentación de los datos que arroja *snort*, se requiere de la instalación de tres extenciones, como se muestra a continuación:

```
pear install Image_Color

pear install Image_Canvas-alpha

pear install Image_Graph-alpha
```

Una vez terminadas estas instalaciones sólo resta configurar base vía web, para lo cual lo invocamos desde el navergador:

```
http://localhost/honeymonitor/base-1.3.8/setup
```

Esta configuración consta de 5 pasos y son los siguientes:

En la primer pantalla se muestran los permisos de lectura y escritura, así como el despliegue de las variables que realizan los informes de error como se muestra en la figura 3.29.

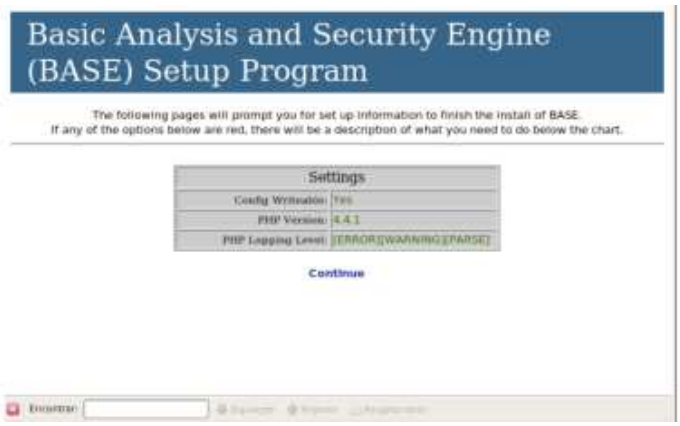

Figura 3.29: Configuración de BASE

En la pantalla siguiente se le tiene que especificar la ruta en donde se encuentra *adodb*, para que base pueda gestionar correctamente la conexión con la *BBDD*. La figura 3.30 mnuestra el proceso.

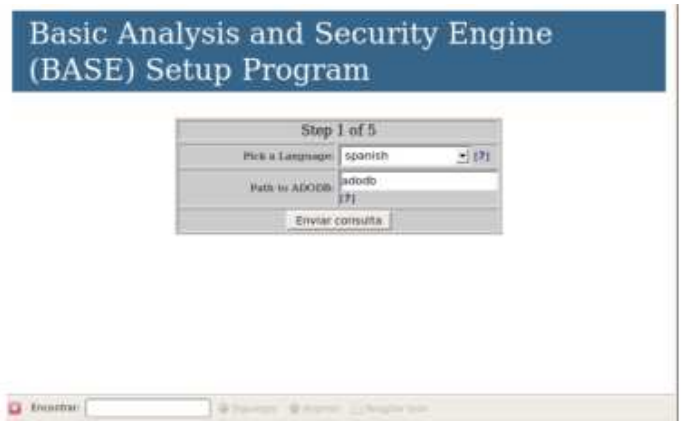

Figura 3.30: Configuración de BASE

Posteriormente la figura 3.31 muestra la especificación de los valores para gestionar la conexión con la *BBDD*, es decir, usuario de la *base, password, host* y el tipo de base de datos a usar.

Figura 3.31: Instalación de BASE

En la pantalla 3.32 es para la autenticación de acceso a la interfaz web de base, si es que así se requiere.

Figura 3.32: Instalación de BASE

En la patalla posterior se le da la opcion de create **BASE AG**, que nos servira para gestionar las tablas creadas para la base de datos de *Snort*.

Finalmente concluidas las configuraciones damos la opción de *continue*.
Como paso final ya solo resta levantar *snort* como se muestra a continuación:

```
snort -c /etc/snort/snort.conf -i eth0 -D
```

3.7. Sebek Server

Sebek, como ya se ha comentado en apartados anteriores, es una herramienta empleada por las *Honeynets* para la captura de datos y actividades de un intruso. La herramienta de *Sebek* consta de dos partes: *Sebek-Cliente y Sebek-Server*.

El fin de este apartado es detallar la instalación y configuración de *Sebek-Server* para detectar, capturar y almacenar cualquier actividad reportada por *Sebek-Cliente* instalado previamente en cada uno de los *Honeypots*.

3.7.1. Instalación de Sebek Server

Para poder realizar la instalación de *Sebek-Server* en nuestro sistema de *Syslog*, es necesario contar con los siguientes paquetes:

- *Sebekd-3.0.3*
- *libpcap8-dev*

La instalación se realizará en un sistema operativo *Ubuntu 7.10*. De igual forma la instalación de los paquetes necesarios se realizará a través de el gestor de paquetes *apt* y la herramienta *wget*.

Mediante el uso del gestor de paquetes apt procedemos con la instalación de la librería **libpcap8-dev** como se muestra:

```
apt-get install libpcap8-dev
```

Por otro lado, para la descarga del paquete de *Sebek-Server*, el primer paso es crear una carpeta llamada *Sbk-Serv* en la ruta */usr*, para posteriormente hacer la descarga del paquete mediante la herramienta *wget*.

```
mkdir /usr/Sbk-Serv

cd /usr/Sbk-Serv

wget http://www.honeynet.org/tools/sebek/Sebekd3.0.3
```

Terminada la descarga del paquete, se procede a descomprimir para su instalación:

```
tar -zxvf Sebekd-3.0.3.tar.gz

cd sebekd3.0.3
```

Para el proceso de instalación basta con seguir las siguientes líneas de comando:

```
./configure

make

make install
```

Finalizada la instalación el sistema contará con 3 nuevos componentes:

- **sbk_extract**
- **sbk_ks_log.pl**
- **sbk_upload.pl**

sbk_extract.

Este componente es capaz de capturar datos de forma directa desde la interfaz, actuando como un rastreador o bien puede extraer datos de un archivo generado por *tcpdump*.

Los datos extraidos por este componente son aquellos que son de entrada *(STDIN)*, desde linea de comandos. Los parámetros que se pueden combinar con este componente son:

- *-f Especifica el archivo de donde se extraeran los datos*
- *-i Especifica la interfaz por donde se capturaran los datos*
- *-p Especifica el puerto udp por el cual los datos arrivarna desde el cliente*

sbk_ks_log.pl

Este componente es un programa desarrollado en *Perl* que se encarga de identificar las pulsaciones de teclas realizadas por el intruso en un *Honeypot* y las envía a la salida estándar *(STDOUT)*.

sbk_upload

Este componente es un programa desarrollado en *Perl* cuya función es almacenar los datos recolectados por los dos componentes anteriores en una base de datos.

La sintaxis correcta para el uso de estos componentes es la siguiente:

```
sbk_extract -i <interfaz-de-red> -p <puerto-udp-sebekcliente> | sbk_ks_log.pl
```

La línea de comando anterior nos permite reazliar un monitoreo desde línea de comandos com lo muestra la figura 3.33

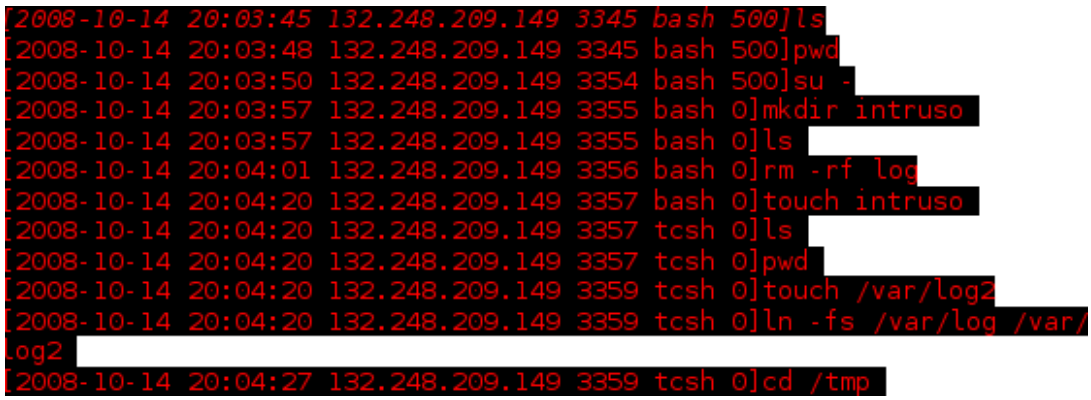

Figura 3.33: Captura de actividades de un intruso

Capítulo 4

Automatización del control de conexiones

En este capítulo se explicará a detalle como automatizar el control de conexiones dentro del *Honeywall*. Esta es una medida adoptada para hacer el monitoreo y el control de conexiones de la *Honeynet* más eficiente para el administrador. Además, como ya se mencionó en el capítulo 2 de esta tesis, el control de las conexiones en la *Honeynet* es un proceso muy importante ya que si no se hace una buena administración de éste proceso un intruso puede lanzar ataques contra sistemas externos a la *Honeynet* y poner así en riesgo la administración de la *Honeynet*.

Para alcanzar este objetivo se realizó un programa en shell, el cual tiene la función de realizar todas las tareas de administración del *Honeywall* para que se realicen de manera automática y se pueda tener un mejor rendimiento del *Honeywall*.

El programa fué adaptado para que opere bajo un sistema *OpenBSD*, ya que la versión original que sugiere el *Honeynet Project* esta adaptado para trabajar bajo un *iptables* de *Linux* llamado *rc.firewall*.

Las principales actividades de este programa es configurar y administrar a *OpenBSD* en modo *firewall transparente*, además de controlar todas y cada una de las conexiones de red que alimentan a la *Honeynet*. Como otras actividades tiene la función de hacer un monitoreo del status de cada *Honeypot* así como permitir o negar permisos de conexión a los *Honeypots* para poder tener el control de las actividades de los intrusos. Finalmente cuenta con la funcionalidad de poder hacer un *rollback* del sistema para regresarlo a su configuración original.

4.1. Estructura de la automatización

El proceso de automatización fue realizado empleando programación en shell, debido a la flexibilidad y libertad que ofrece este ambiente de programación. Se empleó *sqlite3* como motor de base de datos y el uso de herramientas como SED[1] y AWK[2] para el análisis de expresiones reguales. El programa contiene 3 módulos principales que son: *Instalación, Ejecutar programa y Rollback*, la figura 4.1 muestra dichos módulos.

[1] Sed es una potente herramienta de tratamiento de texto para el sistema operativo Unix que acepta como entrada un archivo, lo lee y modifica(con soporte de expresiones regulares) línea a línea mostrando el resultado en pantalla

[2] AWK es un lenguaje de programación diseãdo para procesar datos basados en texto, ya sean archivos o flujos de datos

Figura 4.1: Módulos principales

Además de los módulos principales cuenta también con 3 módulos secundarios para gestionar al sistema y son:*Administración del Honeywall, Honeypots y Control de conexiones*, en la figura 4.2 se muestran éstos módulos.

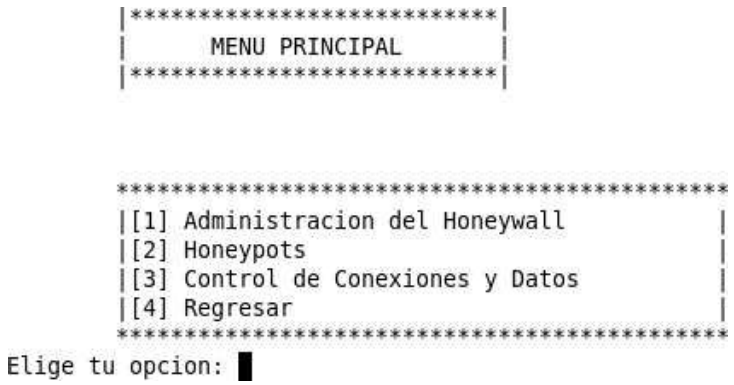

Figura 4.2: Menú Principal

4.1.1. Módulo de instalación

Este módulo tiene como primer tarea hacer una revisión del sistema con el fin de verificar la presencia de archivos importantes(pf.conf,rc.conf y sysctl.conf) y requeridos para que el programa pueda funcionar en su totalidad. Como segunda tarea, una vez detectados todos los archivos requeridos, procede con la creación de respaldos de dichos archivos y con la creación de archivos propios que son necesarios y requeridos por el programa con el fin de que cuente con las suficientes fuentes para ejecutar un proceso de *rollback*[3] y restaurar el sistema completamente. A continuación se presenta la función que es la encargada de realizar el respaldo y creación de los archivos necesarios para la correcta ejecución del programa:

```
function make_backups_honeywall  {
variables_install
if [ ! -f /etc/1 ];then
cp -p /root/$back_main/1 $_path_conf/1
else
cp -p $_path_conf/1 $_path_files_/
rm -rf $_path_conf/1
```

[3]Rollback: es el proceso de dar marcha atras a las configuraciones realizadas

```
cp -p /root/$back_main/1 $_path_conf/1
fi
 }
function create_directory  {
variables_install
if [ ! -d /root/HONEYWALL ];then
mkdir /root/1
touch /root/HONEYWALL/log_sistema
chmod 755 /root/HONEYWALL/log_sistema
touch /root/HONEYWALL/status.txt
chmod 755 /root/HONEYWALL/status.txt
touch /root/HONEYWALL/dump_file.txt
touch /etc/bridgename.bridge0
mv /root/monitor_sistema.sh /root/1/
$_PATH_=/root/1
export $_PATH_
fi
if [ ! -d /root/FLAGS ];then
mkdir /root/2
touch /root/FLAGS/$lag_dump
touch /root/FLAGS/flag_ctl_conn
touch /root/FLAGS/flag_monitor
touch /root/FLAGS/flag_honeypots
touch /root/FLAGS/flag_main
touch /root/FLAGS/flag_bridge
_FLAGS_=/root/2
export _FLAGS_
fi
if [ ! -d /root/backup_bridge ];then
mkdir /root/3
_path_files_=/root/3
fi
if [ ! -d /root/back_main ];then
mkdir /root/4
cp -p /etc/pf.conf /root/4
cp -p /etc/rc.conf /root/4
cp -p /etc/sysctl.conf /root/4
cp -p /etc/bridgename.bridge0 /root/4
fi
 }
```

Otra funcionalidad que toma el módulo de instalación es la creación de la base de datos en *sqlite3*, ya creada la base de datos procede con la creación de tablas y la inserción de los datos correspondientes que alimentaran de información a los demás módulos del programa. Cabe mencionar que los datos que se agregan durante la instalación, son datos que se definen por defecto, es decir, que dichos datos son los ideales para que funcione la *Honeynet*.

Finalmente al terminar con los procesos de la instalación se cargarán los módulos secundarios, los cuales se encargarán de todas las funciones operativas y administrativas dentro del *Honeywall*.

4.1.2. Módulo: Administración del Honeywall

Como ya se mencionón en el capítulo 3 de esta tesis *OpenBSD* será configurado en modo *bridge*, el cual será nuestro *Honeywall*.

Este módulo se encargará de hacer una revisión exhaustiva de archivos necesarios para la configuración del *Honeywall* en modo *bridge*, de no detectar alguno de ellos serán creados. Finalmente revisa el *Honeywall* esta activo en modo *bridge*, en caso contrario lo configura y lo inicia. La figura 4.3 muestra la configuración antes de inicar el *Honeywall* en modo *bridge*:

Figura 4.3: Administración del Honeywall

Una vez que se le ha indicado que se proceda con la configuración del *bridge* realiza el siguiente proceso de activación del *bridge*.Cabe mencionar que si éste modulo es ejecutado y el puente ya se encuentra en estado *activo* lo reconocerá y no realizará ninguna configuración ya que se encuentra activo. En la figura 4.4 se muestra dicho proceso:

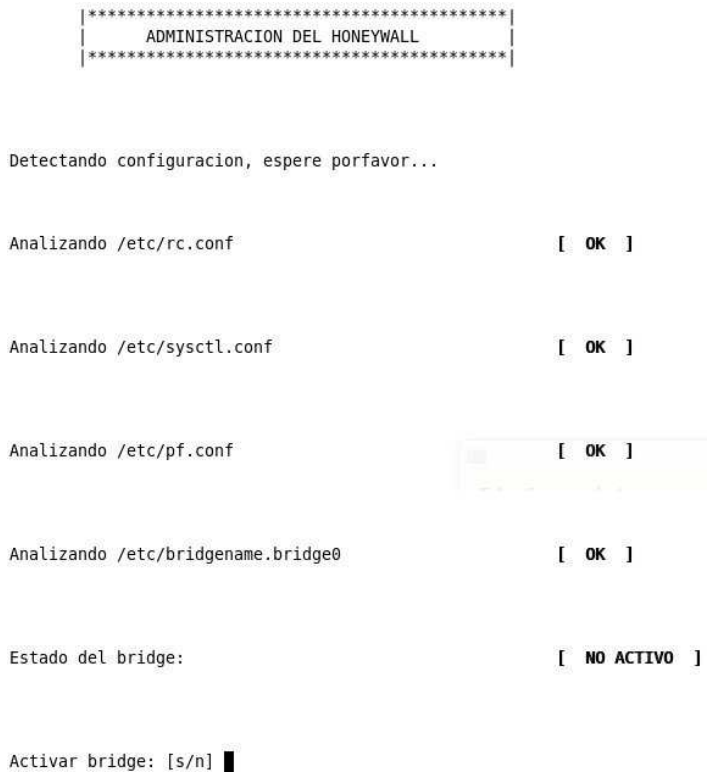

Figura 4.4: Estatus del Bridge

Después de que han llevado acabo los procesos de activación y configuración del *bridge*, el programa muestra una pantalla de monitoreo en la cual se va mostrando información detallada sobre el comportamiento del sistema. Como ejemplo de la información que muestra es: *puertos activos, uso de CPU, procesos activos y conexiones tcp que están siendo bloqueadas por el Honeywall.* El proceso de monitoreo del *Honeywall* es realizado por el módulo llamado *monitoreo_sistema.sh*, el cual es agergado por el módulo de instalación en el cron[4], para más detalles del módulo *monitor_sistema* ver apartado 4.2.

4.1.3. Módulo: Honeypots

El módulo de *Honeypots* tiene la función de monitorear a los *Honeypots*, es decir, muestra información en pantalla de los *Honeypots* y cual es su estado actual. Toda esta información la extrae de la base de datos a través de una función llamada *execute_sql_info*, la cual tiene la siguiente estrucutura:

```
function execute_sql_info  {
variables_install
_GET_QRY_1_1= ‘sqlite3 $DB "select $4 from $1;" ‘
_GET_QRY_1_2= ‘sqlite3 $DB "select $5 from $1;" ‘
_GET_QRY_1_3= ‘sqlite3 $DB "select $6 from $1;" ‘
_GET_QRY_1_4= ‘sqlite3 $DB "select $7 from $1;" ‘
_GET_QRY_1_5= ‘sqlite3 $DB "select $8 from $1;" ‘
_GET_QRY_2_1= ‘sqlite3 $DB "select $4 from $2;" ‘
_GET_QRY_2_2= ‘sqlite3 $DB "select $5 from $2;" ‘
_GET_QRY_2_3= ‘sqlite3 $DB "select $6 from $2;" ‘
_GET_QRY_2_4= ‘sqlite3 $DB "select $7 from $2;" ‘
```

[4]Cron es un programa que permite a un usuario en unix ejecutar comandos o programas de forma automática, además de que permite especificar el día y la hora de ejecución

```
_GET_QRY_2_5= ‘sqlite3 $DB "select $8 from $2;" ‘
_GET_QRY_3_1= ‘sqlite3 $DB "select $4 from $3;" ‘
_GET_QRY_3_2= ‘sqlite3 $DB "select $5 from $3;" ‘
_GET_QRY_3_3= ‘sqlite3 $DB "select $6 from $3;" ‘
_GET_QRY_3_4= ‘sqlite3 $DB "select $7 from $3;" ‘
_GET_QRY_3_5= ‘sqlite3 $DB "select $8 from $3;" ‘
 }
```

Este módulo cuenta con otra funcion llamada *update_attribute*, que tiene la tarea de actualizar cualquier cambio realizado por el administrador referente al número de conexiones permitidas hacia *Internet* por parte de los *Honeypots*. Esta función hace posible que estos cambios se reflejen en la base de datos con el fin de mantener los datos siempre actualizados para que sean mostrados de forma correcta en pantalla. La estructura de dicha función es la siguiente:

```
function execute_sql  {
variables_install
 ‘sqlite3 $DB "$1" ‘

}
function update_attribute_sql  {
variables_install
   echo " UPDATE $1 SET $2= ’$3 ’;"
 }

function update_attribute {
variables_install
sql= ‘update_attribute_sql $1 $2 $3 ‘
execute_sql "$sql"
 }
```

Ya que se ha obtenido y/o actualizado la información en la base de datos, se muestran los datos obtenidos de la BBDD como se ilustra la figura 4.5

```
|================================================================================|
|                                WEB SERVER                                      |
|================================================================================|
|                                                                                |
|Honeypot: webserver                                                             |
|Direccion_IP: 132.248.209.148                                                   |
|Puerto_Sebek: 7811                                                              |
|Estatus: activo                                                                 |
|Conexiones_permitidas: 6                                                        |
|================================================================================|
```

Figura 4.5: Datos obtenidos de la Base de Datos

4.1.4. Módulo: Control de conexiones

Este módulo esta dedicado a gestionar las reglas de filtrado del *Honeywall*, las cuales permiten a los *Honeypots* establecer conexiones hacia *Internet* o no.

Al ejecutar este módulo se cargarán las políticas de acceso por defecto permitidas para los *Honeypots*. Las reglas que se cargan por defecto son las siguientes:

- *Todas las conexiones se filtrarán por la interfaz interna (conectada hacia la Honeynet) del Honeywall, ya que la interfaz externa (conectada hacia internet) permitirá la entrada y salida de cualquier paquete debido a que el Honeywall esta en modo bridge*
- *Sobre la interfaz interna se hará un bloqueo de cualquier paquete y sólo dejara pasar aquella conexión que sea permitida*
- *El número máximo de conexiones hacia Internet por parte de los Honeypots es de 6 conexiones tcp*
- *Se permitirá una conexión por el protocolo udp para que los Honeypots se comuniquen con Sebekserver*
- *Se cargará una tabla en la que se agregaran a aquellos Honeypots que revasen el limite de conexiones y los bloqueará*

Las políticas anteriores se agregarán en el archivo *pf.conf* (ver tema 2) que es el archivo principal que leerá el *Honeywall* para activarlas.

La activación de estas políticas se realiza mediante el uso de la función *insert rules*, y se invocará de la siguiente forma:

insert rules ín "$_interfaz_int"tcp"$HN_WEB_SERVER'6 '

explicando la regla anterior:

- *"in": se le indica al Honeywall que la regla da acceso es para que permita salir paquetes hacia internet, en caso contrario se usa out*
- *"$_interfaz_int": Interfaz de red por la que aplicará la política*
- *"tcp": protocolo que se usará*
- *"$HN_WEB_SERVER": Variable que contiene la dirección ip del Honeypot WebServer*
- *"6": valor que sólo le permitirá a cada Honeypot 6 conexiones tcp hacia internet*

La estructura de la función insert rules es la siguiente:

```
function insert_rules_pf  {
variables_install
echo "pass $1 on $2 proto $3 from $4 to any flags S/SA keep state\
 (max-src-conn $5, max-src-conn-rate 14/5, overload\
  <abusive_hosts> flush)"
 }

function insert_rules  {
variables_install
rules= ‘insert_rules_pf $1 $2 $3 $4 $5 ‘
echo $rules >> $_path_conf/pf.conf
 }
```

Se tiene otra función específica para agregar las políticas permisivas para el protocolo udp, ya que la función *insert_rules* solo aplica para protocolo tcp debido a que tcp siendo un protocolo orientado a conexión requiere de banderas (Flags) para poder realizar la restricción de conexiones, mientras que el protocolo *udp* no es orientado a conexión por lo que no acepta dichas banderas. La función que hace posible insertar políticas para el protocolo *udp* es *insert_rules_sebek* y tiene la siguiente estructura:

```
function insert_rules_pf_sebek  {
variables_install
echo "pass $1 on $2 proto $3 from $4 to $5  keep state "
 }

function insert_rules_sebek  {
variables_install
rules_for_sebek= ‘insert_rules_pf_sebek $1 $2 $3 $4 $5 ‘
echo $rules_for_sebek >> $_path_conf/pf.conf_tmp
 }
```

y la forma de invocar esta función es insert rules sebek "in$_interfaz_int$udpHN_WEB_SERVER SBK_SERVER", donde :

- *"in": se le indica al Honeywall que la regla da acceso es para que permita salir paquetes hacia internet, en caso contrario se usa out*
- *"$_interfaz_int": Interfaz de red por la que aplicará la política*
- *üdp": protocolo que se usará*
- *"$HN_WEB_SERVER": Variable que contiene la dirección ip del Honeypot WebServer*
- *"$SBK_SERVER": variable que contiene la dirección ip del servidor sebek*

Hasta este punto esta es la principal tarea del módulo control de conexiones. Una segunda función de este módulo es permitir al administrador modificar el parámetro que restringe a los *Honeypots* realizar conexiones hacia *Internet*. Esta acción se realiza a través de la función llamda *insert rules modification*, dicha función agrega los parámetros proporcionados por el administrador y los carga en el archivo de configuración del *Honeywall*. Además existen 3 funciones auxiliares llamadas *conexiones_web, conexiones_mail y conexiones_bd*, las cuales invocan a la función *insert_rules_modification*, donde cada una de las funciones auxiliares hace las modificaciones pertinentes para cada *Honeypot* de la red (ver apartado 4.2).

La estructura de la función *insert_rules_modification* es la siguiente:

```
function insert_rules_pf_modification {
variables_install
echo "pass $1 on $2 proto $3 from $4 to any flags S/SA keep state\
 (max-src-conn $5, max-src-conn-rate 14/5, overload\
  <abusive_hosts> flush)"
 }

function insert_rules_modification  {
variables_install
rules_modification= ‘insert_rules_pf_modification $1 $2 $3 $4 $5 ‘
echo $rules_modification >> $_path_conf/pf.conf_tmp
cat \$_path_conf/pf.conf_tmp > $path_conf/pf.conf
rm -rf $_path_conf/pf.conf_tmp
 }
```

4.1.5. Módulo: Ejecutar programa

Este módulo se encarga de detectar si el sistema ya cuenta con la instalacion del programa, para el caso de que el módulo detecte una instalación previa se dirige directamente al menú principal para comenzar con la gestión del sistema.

4.1.6. Módulo: Rollback

El módulo de *rollback* representa un papel muy importante en el programa debido a que este módulo tiene la capacidad de regenerar el sistema completo, regresando a la configuración anterior a la instalación del programa. Cabe mencionar que la ejecución de éste módulo provocara la pérdida de cualquier tipo de información generada durante la gestión del sistema. La figura 4.6 muestra el proceso de *rollback*.

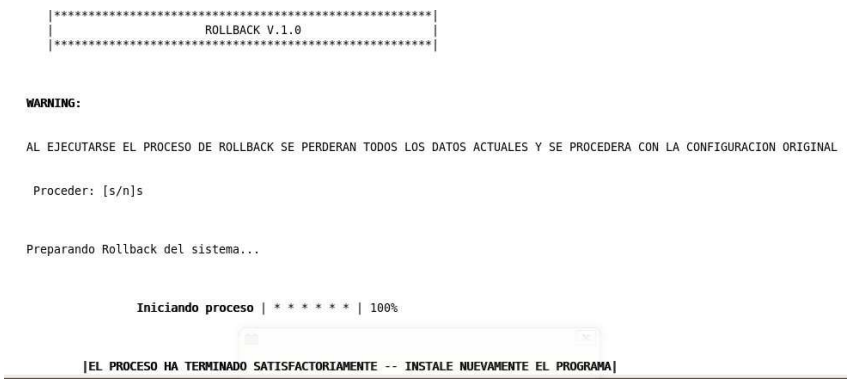

Figura 4.6: Proceso de Rollback

Una vez terminado el proceso de *rollback* el programa esta listo para reinstalarse en el sistema. Todo el código fuente de este programa se puede ver en el apendice A.

Pruebas y Resultados

Las pruebas realizadas fueron sobre la conectividad de los equipos y la configuración de la *Honeynet* (ver cap 3) resultando existosas. Además se cuenta con un plan de pruebas semestrales en donde se revisará nuevamente la configuración de los quipos, por lo que en esta primera revisión y puesta en marcha no reporto ningún problema.

El programa para administrar la *Honeynet* surgió debido a la necesidad de automatizar procesos y agilizar la administración del *Honeywall* en sus tareas de administración de red. Por ejemplo, realiza de manera automática la configuración del sistema operativo *OpenBSD* a modo *Bridge*, además de tener la capacidad de monitorear en tiempo real a todos y cada uno de los *Honeypots* y así saber el estado de cada uno de ellos para poder determinar si cada *Honeypot* cumple con las condiciones iniciales de la *Honeynet*, que por ejemplo es permitir únicamente 6 conexiones hacia *Internet*. Por lo que concluye que dicho programa es una herramienta fundamental dentro del *Honeywall* ya que permite al administrador de la *Honeynet* operar la red de forma más sencilla y eficaz.

El *Honeynet Project* proveé de algunas arquitecturas de red y de ciertos pasos a seguir para la implementación de una *Honeynet* de segunda generación, dichas sugerencias se encuentran publicadas en los *white papers* del proyecto, los cuales pueden ser consultados fácilmente en el sitio oficial del proyecto: *www.honeynetproject.org/papers*.

Tomando en cuenta dichos documentos se desarrolló e implementó esta *Honeynet* con los dos mecanismos necesarios (Control y captura de datos) para que sea de segunda generación, además de que se tuvieron que realizar adaptaciones a esta *Honeynet* debido a la experimentación de ciertos problemas ocurridos durante su implementación.

El mecanismo de control de datos fue implementado en todos y cada uno de los *Honeypots* de la *Honeynet* mediante el uso de la herramienta llamada *Sebek*. Cabe mencionar que esta herramienta se limita exclusivamente a la captura de datos y a su vez es un complememto del programa de gestión del *Honeywall* antes mencionado.

Durante la implementación de esta herramienta en los *Honeypots* se presentaron algunos problemas de compatibilidad con los sistemas operativos. Dichos problemas que se presentaron fueron detectados en las siguientes distribuciones de *Linux*: Centos 4 y 5, Debian Etch, RedHat 4 y 5 y Fedora Core en sus versiones 5,6,7,8 y 9. Después de realizar un análisis se pudo concluir que dichas distribiciones no son compatibles hasta el momento con esta herramienta, debido a que *Sebek* al ser un capturador de datos (*Keylogger*) y al hacer uso de su propia tabla de llamadas al sistema *(syscall table)* requiere de ciertas rutas absolutas en el sistema operativo y las distribiciones antes expuestas por seguridad del sistema operativo no son generadas al momento de la instalación del mismo.

Ante este problema de la rutas absolutas inexistenes en dichas distribiciones, se procedió a crear las rutas de forma manual obteniendo como resultado los siguientes comportamientos:

- Para *RedHat, CentOS y Fedora* no se lograba la compilación exitosa de la herramienta.

- Para Debian se logró la compilación exitosa de la herramienta, pero se experimentó que el sistema

operativo se pasmaba y dejaba de operar. Todo esto debido a que *Sebek* debe de compilarse como un módulo de *kernel*.

Se concluye que para las distribuciones de *Linux* antes mencionadas la herramienta para la captura de datos *Sebek* aún no es compatible y funcional, por lo que se descartaron como opciones para la implementación de los *Honeypots*.

Las distribuciones de *Linux* que resultarón óptimas en estas pruebas de compatibilidad fueron: *Fedora Core 3 y Ubuntu 7.10*. Los servicios de correo y de base de datos fueron implementados bajo estas distribuciones de *Linux*: Fedora Core 3 y Ubuntu 7.10 respectivamente. Del mismo modo el sistema operativo *Windows* presentó problemas de incompatibilidad, por lo que se sometió a las mismas pruebas resultando *Windows 2000* en su versión Server y Profesional las únicas versiones compatibles para la instalación de la herramienta *Sebek*. Así el servicio de *web* fue implementado bajo el sistema operativo *Windows 2000 Server*.

Para la implementación del control de datos se implementó como sistema central de la *Honeynet* un *firewall* bajo el sistema operativo *OpenBSD*. Dicho proceso de implementación resultó exitoso, además de resultar compatible con el resto de sistemas operativos de los *Honeypots*.

Como elemento final de la *Honeynet* se instaló un sistema bajo la distribución *Ubuntu 7.10 de Linux*, el cual poseé dos instancias, una de ellas es como servidor de bitácoras *syslog* y la otra como un sistema detector de intrusos (*IDS*). Para cada una de las instancias se instalo *Sebek* en su version Server y Snort como escaner de red respectivamente. Esta implementación resulto existosa y funcional.

Por otro lado se obtuvo una red totalmente aislada de la red productiva del Instituto de Física que ofrece servicios reales a los usuarios, pero con la particularidad de que es una red totalmente monitoreada y con restricciones en las conexiones salientes, con lo cual se cumplió con uno de los objetivos principales de este proyecto, que es el de tener una red para que sea comprometida por los intrusos y así poder determinar patrones de ataque, herraminetas empleadas y sus motivos.

Finalmente se ha puesto en operación la *Honeynet* resultando viable para su objetivo, por lo que el Instituto de Física de la UNAM cuenta ya con una red alterna de producción para poder enfocarse a la investigacoón de vulnerabilidades actuales y futuras.

Conclusiones Generales

Como se planteó al inicio de este proyecto, el objetivo principal fue de desarrollar e implementar una *Honeynet* de segunda generación teniendo como sistema operativo *OpenBSD*, la cual tiene la función de generar un ambiente de red que proporcione servicios reales hacia internet como son: *correo, web, ssh, ftp,*etc, para investigación de las vulnerabilidades actuales y nuevas sobre estos servicios.

Para poder brindar dichos servicios, este proyecto puso en marcha una infraestructura completamente funcional que hace posible capturar cualquier actividad en donde se involucran estos servicios y generar una bitácora de dichas actividades para su posterior análisis con base en el *Honeynet Project*.

En lo particular es un proyecto bien elaborado y estructurado en el cual además de aprender sobre las técnicas empleadas por los intrusos y sus motivos para atacar, viene a subsanar la problematica actual del Instituto de Física, ya que ahora tiene la posibilidad de aislar el tráfico mal intencionado sobre la *Honeynet* y así ya no exponer a la red productiva.

Durante el desarrollo de este proyecto aprendí muchas cosas sobre los problemas de seguridad actuales, además de saber identificar las tres arquitecturas principales de *Honeynets* y sus alcances lo cual me reflejó un crecimiento profesional en mi carrera dentro del campo de la seguridad de la información.

Experimente algunas dificultades durante el desarrollo de este proyecto, como por ejemplo en las configuraciones de los sistemas ya que las versiones que utilice en un principio no eran compatibles con *Sebek-cliente*, lo cual me representaba un problema muy grande ya que *Sebek* es un aplicación de suma importancia en la *Honeynet* porque es el que realiza el control de datos en la red. Para poder subsanar este problema realice pruebas a fondo sobre distintos sistemas operativos (Linux, Windows y OpenBSD) para poder encontrar la mejor opción para la *Honeynet*.

Cabe mencionar que este proyecto no es una replica exacta del *Honeynet Project*, contiene particularidades que lo hacen diferente, por ejemplo el sistema principal de la red (*Honeywall*) esta implementado sobre *OpenBSD*, el motor de base de datos sobre *sqlite*, además de un script personalizado para la automatización de la *Honeynet*.

Una vez que obtuve la compatibilidad con las aplcaciones y los sistemas operativos procedí a la implementación de la *Honeynet*, particularmente este proyecto me quito muchas horas de sueño y de mi vida social, pero me dejo la enorme satisfacción de aprender un poco más de lo que implica la seguridad de la información y de haber contribuido con el Instituto de Física a controlar los ataques contra la institución, así como de generar un ambiente de investigación sobre seguridad de la información en el Instituto.

Bibliografía

[1] Honeynet Project, *know you Enemy*,Lance Spitzner Anton Chuvakin and Edwar Balas,p.47-52,95-133,183-197,Addison wesley,US,2nd Edition,2004.

[2] Internet and Web Security, *Building Internet Firewalls*, Elizabeth D. Zwicky Simon Cooper and D.Brent Chapman,p.104-119,122-133,241-25,Addison-Wesley,EU,2nd Edition,2000.

[3] *Fundamentos de Seguridad Informática*, M.C Cintia Quezada Reyes y M.C Jaquelina López Barrientos,Facultad de Ingeniería UNAM,2004.

[4] Honeypots, *Tracking Hackers*,Lance Spitzner, Addison-Wesley,p.113-124,168-185,1st Edition, 2003.

[5] Herramientas de seguridad, *Software*,Tony Howlett,Anaya,p.41-42,99-102,263-269,274,325,326329-332,1st Edition, 2005.

[6] *Fundamentals Network Security*,Erik Maiwald,p.67-86,140-158,Mc Graw Hill,US,2003.

[7] Manual de Referencia,*Mysql*,p.45-56,98-123,3th Edition,2004.

[8] *http://es.wikipedia.org/wiki/Berkeley_Software_Distribution acceso:08/Ago/09*

[9] *http://es.wikipedia.org/wiki/Plugins, acceso:05/Ago/09*

[10] *http://manuales.espaciolinux.com/documento-323, acceso:08/Ago/09*

[11] *http://www.securityfocus.com/infocus/1858, acceso:07/Ago/09*

[12] *http://parvinderbhasin.blogspot.com/2007/12/openbsd-42-ids-solution-snort-and-base.html, acceso:08/Ago/09*

[13] *http://www.howtoforge.com/intrusion-detection-with-snort-mysql-apache2-on-ubuntu-7.10, acceso:08/Ago/09*

[14] *http://souptonuts.sourceforge.net/, acceso:06/Ago/09*

[15] *http://docente.ucol.mx/al980347/public_html/modelo_tcp.htm, acceso:7/Ago/09*

[16] *http://www.cisco.com/en/US/docs/internetworking/technology/handbook/Intro-to-Internet.html, acceso: 7/Ago/09*

[17] *www.tcpdump.org, acceso: 9/Ago/09*

[18] *http://www.cisco.com/en/US/docs/internetworking/technology/handbook/Intro-to-Internet.html, acceso: 8/Ago/09*

[19] *http://lamphowto.com/, acceso: 9/Ago/09*

Apéndice A

Código fuente para automatizar el control de datos y de conexiones de una *Honeynet* de segunda generación bajo OpenBSD

En este apartado se presenta el código fuente completo del programa de control de conexiones implementado en el *Honeywall* para la automatización de los procesos mencionados en el apartado 4.1.

```
variables_install ()  {

# EN LA SIGUIENTE VARIABLE SE HACE USO DE LAS UTILERIAS CUT,SED Y AWK
PARA HACER UN FILTRADO FINO Y OBTENER ASI UNICAMENTE LAS INTERFACES DE RED
QUE TIENE EL SISTEMA, ES DECIR, LAS DOS INTERFACES DE RED Y EL DISPOSITIVO
VIRTUAL DEL BRIDGE #

_global_inte=‘ifconfig | cut -f1 -d: | cut -f1 | sed  ’/^$/d’ | tr  ’  ’  ’  ’ |
awk  { ’print $2,$3,$5 ’ }‘
_interfaz_ext=‘echo $_global_inte | awk  { ’print $1 ’ }‘
_interfaz_int=‘echo $_global_inte | awk  { ’print $2 ’ }‘
_bridge=‘ifconfig | grep ’bridge’ | grep ’RUNNING ’ | awk  { ’print $2 ’ } |
 sed ’s/flags=41\ <UP,RUNNING>/RUNNING/ ’‘

# DEFINICION DE LAS VARIABLES ESTATICAS QUE USARA EL PROGRAMA A
LO LARGO DE SU EJECUCION #

HN_WEB_SERVER="132.248.209.148"
HN_MAIL_SERVER="132.248.209.149"
HN_BD_SERVER="132.248.209.152"
SBK_SERVER="132.248.209.151"
DB="honeypots.db"
_FLAGS_=/root/FLAGS  ; export $_PATH_
_path_files_=/root/$backup\_bridge$ ;export _path_files_
_PATH_=/root/HONEYWALL  ; export _PATH_
_path_conf=/etc ; export _path_conf
_path_cron=/var/cron/tabs  ; export _path_cron
 }

# FUNCION PARA HACER UN UPDATE A LA BASE DE DATOS Y ACTUALIZAR
```

```
ASI EL ESTATUS DE LOS HONEYPOTS #

function execute_sql  {
variables_install
 ‘sqlite3 $DB "$1" ‘

 }
function update_attribute_sql  {
variables_install
   echo "UPDATE $1 SET $2= ’$3 ’;"
 }

function update_attribute  {
variables_install
sql= ‘update_attribute_sql $1 $2 $3 ‘
execute_sql "$sql"
}

# FUNCIONES PARA OBTENER DATOS DE LA BASE DE DATOS #

function execute_sql_info  {
variables_install
_GET_QRY_1_1= ‘sqlite3 $DB "select $4 from $1;" ‘
_GET_QRY_1_2= ‘sqlite3 $DB "select $5 from $1;" ‘
_GET_QRY_1_3= ‘sqlite3 $DB "select $6 from $1;" ‘
_GET_QRY_1_4= ‘sqlite3 $DB "select $7 from $1;" ‘
_GET_QRY_1_5= ‘sqlite3 $DB "select $8 from $1;" ‘
_GET_QRY_2_1= ‘sqlite3 $DB "select $4 from $2;" ‘
_GET_QRY_2_2= ‘sqlite3 $DB "select $5 from $2;" ‘
_GET_QRY_2_3= ‘sqlite3 $DB "select $6 from $2;" ‘
_GET_QRY_2_4= ‘sqlite3 $DB "select $7 from $2;" ‘
_GET_QRY_2_5= ‘sqlite3 $DB "select $8 from $2;" ‘
_GET_QRY_3_1= ‘sqlite3 $DB "select $4 from $3;" ‘
_GET_QRY_3_2= ‘sqlite3 $DB "select $5 from $3;" ‘
_GET_QRY_3_3= ‘sqlite3 $DB "select $6 from $3;" ‘
_GET_QRY_3_4= ‘sqlite3 $DB "select $7 from $3;" ‘
_GET_QRY_3_5= ‘sqlite3 $DB "select $8 from $3;" ‘
 }

# FUNCION PARA PF DONDE SE INSERTARAN POLITICAS DE FILTRADO
PARA EL FUNCIONAMIENTO DEL HONEYWALL #

function insert_rules_pf  {
variables_install
echo "pass $1 on $2 proto $3 from $4 to any flags S/SA keep state
(max-src-conn $5, max-src-conn-rate 14/5, overload <abusive_hosts> flush)"
 }

function insert_rules  {
variables_install
rules= ‘insert_rules_pf $1 $2 $3 $4 $5 ‘
echo $rules >> $_path_conf/pf.conf
 }

# FUNCION QUE PERMITE REALIZAR LOS CAMBIOS EN LAS POLITICAS DEL HONEYWALL
(modifica parametro de limite de conexiones) #
```

```
function insert_rules_pf_modification {
variables_install
echo "pass $1 on $2 proto $3 from $4 to any flags S/SA keep state
(max-src-conn $5, max-src-conn-rate 14/5, overload
    <abusive_hosts> flush)"
 }

function insert_rules_modification  {
variables_install
rules_modification= ‘insert_rules_pf_modification $1 $2 $3 $4 $5 ‘
echo $rules_modification >> $_path_conf/pf.conf_tmp
cat $_path_conf/pf.conf_tmp > $path_conf/pf.conf
rm -rf $_path_conf/pf.conf_tmp
 }

# FUNCION PARA INSERTAR REGLAS QUE PERMITIRAN A LOS HONEYPOTS
COMUNICARSE CON EL SEBEK-SERVER A TRAVES DEL PROTOCOLO UDP #

function insert_rules_pf_sebek  {
variables_install
echo "pass $1 on $2 proto $3 from $4 to $5  keep state "
 }

function $insert\_rules\_sebek  {
variables_install
rules_for_sebek= ‘insert_rules_pf_sebek $1 $2 $3 $4 $5 ‘
echo $rules_for_sebek >> $_path_conf/pf.conf_tmp
 }

# FUNCION PARA ACTUALIZAR LA BASE DE DATOS UNA VEZ QUE SE
HAN REALIZADO MODIFICACIONES EN EL HONEYWALL POR EL ADMINISTRADOR #

function execute_sql_modification  {
variables_install
 ‘sqlite3 $DB "$1" ‘
 }

function update_attribute_sql_modification  {
variables_install
   echo "UPDATE $1 SET $2=$3;"
 }

function update_attribute_modification {
variables_install
sql_mod= ‘update_attribute_sql_modification $1 $2 $3 ‘
execute_sql "$sql_mod$"
 }

# FUNCION PARA REALIZAR BACKUPS #

function make_backups_honeywall  {
variables_install
if [ ! -f /etc/$1 ];then
cp -p /root/back_main/$1 $_path_conf/$1
else
```

```
cp -p $_path_conf/$1 $_path_files_/
rm -rf $_path_conf/$1
cp -p /root/back_main/$1 $_path_conf/$1
fi
 }

function make_backups  {
variables_install
if [ -d $_path_files_ ];then
make_backups_honeywall $1
else
echo " No es posible realizar el backup"
sleep 3
exit 0
fi
 }

function create_directory  {
variables_install
if [ ! -d /root/HONEYWALL ];then
mkdir /root/$1
touch /root/HONEYWALL/log_sistema
chmod 755 /root/HONEYWALL/log\_sistema
touch /root/HONEYWALL/status.txt
chmod 755 /root/HONEYWALL/status.txt
touch /root/HONEYWALL/dump_file$.txt
touch /etc/bridgename.bridge0
mv /root/monitor_sistema.sh /root/$1/
_PATH_=/root/$1
export _PATH_
fi
if [ ! -d /root/FLAGS ];then
mkdir /root/$2
touch /root/FLAGS/flag_dump
touch /root/FLAGS/flag_ctl_conn
touch /root/FLAGS/flag_monitor
touch /root/FLAGS/flag_honeypots
touch /root/FLAGS/flag_main
touch /root/FLAGS/flag_bridge
_FLAGS_=/root/$2
export _FLAGS_
fi
if [ ! -d /root/backup_bridge ];then
mkdir /root/$3
_path_files_=/root/$3
fi
if [ ! -d /root/back_main ];then
mkdir /root/$4
cp -p /etc/pf.conf /root/$4
cp -p /etc/rc.conf /root/$4
cp -p /etc/sysctl.conf /root/$4
cp -p /etc/bridgename.bridge0 /root/$4
fi
 }

# FUNCION QUE GESTIONA EL MONITOREO DE LOS HONEYPOTS, ASI COMO SU MONITOREO #
```

```
honeypots ()  {
variables_install
clear

echo "|*****************************************|"
echo "|      ADMINISTRACION DE LOS HONEYPOTS      |"
echo "|*****************************************|"

sleep 3

# EN ESTA SECCION SE PROCEDE CON VALIDACION DEL ESTATUS DE
LOS HONEYPOTS, DICHA INFOMRACION DE RECOGE DE LA BBDD,
ADEMAS DE QUE SE CARGA UN TCPDUMP PARA PODER DECIDIR CUAL ES
EL ESTATUS DE CADA HONEYPOT #

echo "\033[1;38mObteniendo informacion |\033[0;0m \c"

status_hn_web= ‘cat $_PATH_/dump_file.txt | grep  ’132.248.209.148 ’ | sed -e "$=" ‘
echo "* \c"
sleep 2
status_hn_mail= ‘cat $_PATH_/dump_file.txt | grep  ’132.248.209.149 ’ | sed -e "$=" ‘
echo "* \c"
sleep 2
status_hn_bd= ‘cat $_PATH_/dump_file.txt | grep  ’132.248.209.152 ’| sed -e "$=" ‘
echo "* \c"
sleep 2
cd $_PATH_
if [ -z $status_hn_web ];then
update_attribute "webserver" "estatus"  ’activo ’
echo "* \c"
else
update_attribute "webserver" "estatus"  ’activo ’
echo "* \c"
fi

if [ -z $status_hn_mail ];then
update_attribute "mailserver" "estatus"  ’inactivo ’
echo "* \c"
else
update_attribute "mailserver" "estatus"  ’activo ’
echo "* \c"
fi

if [ -z $status_hn_bd ];then
update_attribute "bdserver" "estatus"  ’inactivo ’
echo "* \c"
else
update_attribute "bdserver" "estatus"  ’activo ’
echo "* \c"
fi

# REALIZANDO LOS QUERYS A LA BBDD PARA MOSTRAR EN EL DISPLAY DEL HONEYPOT WEBSERVER #

    execute_sql_info "webserver" "mailserver" "bdserver" "id_honeypot" "ip_add"\
     "pto_sbk_cl" "estatus" "max_conn_limit"
```

```
    sleep 3

echo "* \c"
echo "\033[1;38m |\033[0;0m \c"
echo "\033[1;38m 100% \033[0;0m"
sleep 1
echo "Espere un momento..."
sleep 1

# SE GENERA EL DESPLIEGUE EN PANTALLA DE LOS DATOS OBTENIDOS EN LA BASE DE DATOS #

echo "\n"
echo "\t|================================================================|"
echo "\t|           WEBSERVER     |"
echo "\t|================================================================|"
echo "\t|Honeypot:\033[0;0m $_GET_QRY_1_1                                  |"
echo "\t|Direccion_IP:\033[0;0m $_GET_QRY_1_2                            |"
echo "\t|Puerto_Sebek:\033[0;0m $_GET_QRY_1_                               |"
echo "\t|Estatus:\033[0;0m $_GET_QRY_1_4                                   |"
echo "\t|Conexiones_permitidas:\033[0;0m _GET_QRY_1_5                      |"
echo "\t|================================================================|"
echo "\n"
echo "\n"
echo "\t|================================================================|"
echo "\t|           MAILSERVER               |"
echo "\t|================================================================|"
echo "\t|Honeypot:\033[0;0m $_GET_QRY_2_1                                  |"
echo "\t|Direccion_IP:\033[0;0m $_GET_QRY_2_2                            |"
echo "\t|Puerto_Sebek:\033[0;0m $_GET_QRY_2_3                              |"
echo "\t|Estatus:\033[0;0m $_GET_QRY_2_4                                   |"
echo "\t|Conexiones_permitidas:\033[0;0m $_GET_QRY_2_5                     |"
echo "\t|================================================================|"
echo "\n"

echo "\n"
echo "\t|================================================================|"
echo "\t|         BDSERVER        |"
echo "\t|================================================================|"
echo "\t|Honeypot:\033[0;0m $_GET_QRY_3_1                                  |"
echo "\t|Direccion_IP:\033[0;0m $_GET_QRY_3_2                            |"
echo "\t|Puerto_Sebek:\033[0;0m $_GET_QRY_3_3                              |"
echo "\t|Estatus:\033[0;0m $_GET_QRY_3_4                                   |"
echo "\t|Conexiones_permitidas:\033[0;0m $_GET_QRY_3_5                     |"
echo "\t|================================================================|"
echo "\n"
echo "\033[1;38m Presione q para salir \033[0;0m "
read _q
if [ $_q == "q" ]; then
cd $_PWD_
main
else
echo "\033[1;38m Opcion no valida \033[0;0m "
fi
 }
```

```
honeywall ()  {
variables_install
clear
echo "|*****************************************|"
echo "|      ADMINISTRACION DEL HONEYWALL         |"
echo "|*****************************************|"
sleep 4

# EL SIGUIENTE FRAGMENTO DE CODIGO REALIZA
 UN CHECKEO DEL ESTADO ACTUAL DEL
 HONEYWLALL EN MODO BRIDGE #

echo "\n"
echo " Detectando configuracion, espere porfavor... \c"
sleep 2
cat $_FLAGS_/flag_bridge | grep 'activo ' 2> 2 > 1
if [ $? ==  '1 ' ];then

# haciendo check y backup de los archivos rc.conf, pf.conf, sysctl.conf,
bridgename.bridge0 #

echo "\n"
echo " Analizando $_path_conf/rc.conf \c"
sleep 1
if [ -f $_path_conf/rc.conf ] ;then
sleep 1
echo "\033[1;38m [  OK  ]\033[0;0m"
else
echo "Error: Archivo corrupto o no existe"
echo "Abortando operacion"
exit 0;
fi
echo "\n"
echo " Analizando $_path_conf/sysctl.conf \c"
sleep 1
if [ -f $_path_conf/sysctl.conf ];then
sleep 1
echo "\033[1;38m [  OK  ]\033[0;0m"
else
echo "Error: Archivo corrupto o no existe"
echo "Abortando operacion"
exit 0;
fi
echo "\n"
echo " Analizando $_path_conf/pf.conf \c"
if [ -f $_path_conf/pf.conf ]; then
sleep 1
echo "\033[1;38m [  OK  ]\033[0;0m"
else
echo "Error: Archivo corrupto o no existe"
echo "Abortando operacion"
exit 0;
fi

echo "\n"
echo "Analizando $_path_conf/bridgename.bridge0 \c"
```

```
if [ -f $_path_conf/bridgename.bridge0 ]; then
sleep 1
echo "\033[1;38m [  OK  ]\033[0;0m"
else
echo "Error: Archivo corrupto o no existe"
echo "Generando archivo \c"
exit 0;
fi

# A CONTINUACION SE REVISA CUAL ES EL
ESTADO ACTUAL DE BRIDGE, SI NO ESTA
INICIADO PROCEDE CON SU CONFIGURACION #

echo "\n"
echo " Estado del bridge: \c"

if [ -z $_bridge ]; then
echo "\033[1;38m [  NO ACTIVO  ]\033[0;0m"
echo "\n"
echo " Activar bridge: [s/n] \c"
read activar
if [ $activar == "s" ];then
echo "\n"
echo "Configurando interfaz: $_interfaz_ext \c"
sleep 2
echo "up" > $_path_conf/hostname.$_interfaz_ext
if [ $? ==  '0 ' ];then
sleep 1
echo "\033[1;38m [  OK  ]\033[0;0m"
else
echo "\n"
echo "Error: imposible configurar la interfaz $_interfaz_ext"
exit 0;
fi
echo "\n"
echo "Configurando interfaz: $_interfaz_int \c"
sleep 1
echo "up" > $_path_conf/hostname.$_interfaz_int
if [ $? ==  '0 ' ];then
sleep 1
echo "\033[1;38m [  OK  ]\033[0;0m"
else\\
echo "\n"
echo "Error: imposible configurar la interfaz $_interfaz_int"
exit 0;
fi

# CONFIGURACION DE PARAMETROS PARA ACTIVAR EL BRIGE #

cat $_path_conf/sysctl.conf | sed \ 's/#net.inet.ip.forwarding=1/net.inet.ip.forwarding=1/ ' >
 $_path_conf/sysctl.conf_tmp
cat $_path_conf/sysctl.conf_tmp > $_path_conf/sysctl.conf
rm $_path_conf/sysctl.conf_tmp

# CONFIGURACION DE  ARCHIVOS PARA ACTIVAR EL BRIGE #
cat $_path_conf/rc.conf | sed  's/pf=NO/pf=YES/ ' > $_path_conf/rc.conf_tmp
```

```
cat $_path_conf/rc.conf_tmp > $_path_conf/rc.conf
rm $_path_conf/rc.conf_tmp
echo "\n"
echo "Iniciando el bridge \c"
sleep 1
echo "add $_interfaz_ext add $_interfaz_int up" > $_path_conf/bridgename.bridge0
ifconfig bridge0 up
sleep 1
if [ $? == '0' ];then
echo "\033[1;38m [  OK  ]\033[0;0m"
sleep 2
else
echo "\033[1;38mError: No fue posible el inicio del bridge\033[0;0m "
exit 1
fi

echo "\n"
echo "El estatus del bridge es: \c"
sleep 1
echo "\033[1;38m [  ACTIVO  ]\033[0;0m"
echo "activo" > $_FLAGS_/flag_bridge
sleep 4
      else
    main
      fi
   else
   echo "\033[1;38m [  ACTIVO  ]\033[0;0m"
   echo "activo" > $_FLAGS_/flag_bridge
fi
echo "\n"
echo " Preparando monitoreo del Honeywall..."
sleep 3

# LA SIGUIENTE LIENA DESPLIEGA EN PANTALLA EL MONITOREO DEL SISTEMA #

        less $_PATH_/log_sistema
   else
echo "\033[1;38m |EL BRIDGE YA HA SIDO ACTIVADO |\033[0;0m"
sleep 2
less $_PATH_/log_sistema
 fi
}


rollback () {
variables_install
clear
echo "\t|*********************************************************|"
echo "\t|         ROLLBACK V.1.0                                  |"
echo "\t|*********************************************************|"
echo "\n"
sleep 2

# EL SIGUIENTE FRAGMENTO DE CODIGO HACE UNA
 REVISION DE DIRECTORIOS NECESARIOS PARA PODER REALIZAR UN ROLLBACK #
```

```
if [ -d $_path_files_ ]; then
echo "\033[1;38mWARNING:\033[0;0m "
echo "\n"

echo "AL EJECUTARSE EL PROCESO DE ROLLBACK
SE PERDERAN TODOS LOS DATOS ACTUALES Y SE PROCEDERA CON LA CONFIGURACION ORIGINAL"

echo "\n"
echo " Proceder: [s/n]\c"
read r_back
if [ $r_back == "s" ];then
echo "\n"
echo "Preparando Rollback del sistema..."
sleep 3
echo "\n"
echo "\033[1;38mIniciando proceso...\033[0;0m | \c"
make_backups_honeywall "pf.conf"
sleep 1
echo "* \c"
make_backups_honeywall "rc.conf"
sleep 1
echo "* \c"
make_backups_honeywall "sysctl.conf"
sleep 1
echo "* \c"
make_backups_honeywall "bridgename.bridge0"
sleep 1
echo "* \c"
sleep 1
echo "* \c"
ifconfig bridge0 down
sleep 1
echo "* | \c"
sed  '$d ' $_path_cron/root > $_path_cron/temp_cron.txt;
   cat $_path_cron/temp_cron.txt > $_path_cron/root
   sed  '$d\ ' $_path_cron/root > $_path_cron/temp_cron.txt;
   cat $_path_cron/temp_cron.txt > $_path_cron/root
sleep 1
echo "100 %"
echo "\n"
echo "\033[1;38m|EL PROCESO HA TERMINADO SATISFACTORIAMENTE --
INSTALE NUEVAMENTE EL PROGRAMA|\033[0;0m"
sleep 2
main3

else
install
    fi
else
echo "\033[1;38mERROR: NO ES POSIBLE EL ROLLBACK
| CONSULTE AL ADMINISTRADOR|\033[0;0m"
fi

 }
```

```
# LAS SIGUIENTES 3 FUNCIONES SE USARAN DE FORMA
 AUXILIAR PARA REALIZAR LA MODIFICACION
 DE LAS POLITICAS DE FILTRADO PARA LOS HONEYPOTS #

# LA FUNCIONES AUXILIARES SERAN
 INVOCADAS POR LA FUNCION CONTROL_CONEXIONES #

conexiones_web () {
variables_install

echo "\nRealizando cambios, espere..."
sleep 1
cat $_path_conf/pf.conf | sed  '/132.248.209.148/d ' > $_path_conf/pf.conf_tmp
insert_rules_modification "in" "$_interfaz_int" "tcp" "$HN_WEB_SERVER" "$conn_limit_web"
update_attribute_modification "webserver" "max_conn_limit" "$conn_limit_web"

 pfctl -f $path_conf/pf.conf
 if [ $? ==   '0 ' ];then
echo "\n |Los cambios han sido realizados correctamente|"
sleep 2
echo "\n Mostrando las nuevas reglas cargadas..."
sleep 2
pfctl -sr
sleep 4
echo "\nRegresando..."
sleep 1
main
else
  echo "\n|NO HA SIDO POSIBLE EL CAMBIO|"
sed  '$d  ' _path_conf/pf.conf > $_path_conf/pf.conf_tmp
cat $_path_conf/pf.conf\_tmp > $_path_conf/pf.conf
rm -rf $_path_conf/pf.conf_tmp
sleep 2
main
 fi
 }

conexiones_mail () {
variables_install
echo "\nRealizando cambios, espere..."
 sleep 1
cat $_path_conf/pf.conf | sed  '/132.248.209.149/d  ' > $_path_conf/pf.conf_tmp
insert_rules_modification "in" "$_interfaz_int" "tcp" "$HN_MAIL_SERVER"
"$conn_limit_mail"

update_attribute_modification "mailserver" "max_conn_limit" "$conn_limit_mail"

 pfctl -f $path_conf/pf.conf
 if [ $? ==  '0 ' ];then
echo "\n |Los cambios han sido realizados correctamente|"
sleep 2
echo "\n Mostrando las nuevas reglas cargadas..."
sleep 2
pfctl -sr
sleep 4
echo "\nRegresando..."
```

```
sleep 1
main
 else
echo "\n|NO HA SIDO POSIBLE EL CAMBIO|"
sed   '$d ' $_path\_conf/pf.conf > $_path_conf/pf.conf_tmp
cat $_path_conf/pf.conf_tmp $_path_conf/pf.conf
rm -rf $_path_conf/pf.conf_tmp
sleep 2
main
 fi
 }

conexiones_bd ()  {
variables_install
echo "\nRealizando cambios, espere..."
 sleep 1
cat $_path_conf/pf.conf | sed  '/132.248.209.152/d ' > $_path_conf/pf.conf_tmp
insert_rules_modification "in" "$_interfaz_int" "tcp" "HN_BD_SERVER"
 "conn_limit_bd$"
update_attribute_modification "bdserver" "max_conn_limit" "$conn_limit_bd"

 pfctl -f $path_conf/pf.conf
 if [ $? ==   '0  ' ];then
echo "\n |Los cambios han sido realizados correctamente|"
sleep 2
echo "\n Mostrando las nuevas reglas cargadas..."
sleep 2
pfctl -sr
sleep 4
echo "\nRegresando..."
sleep 1
main
 else
echo "\n|NO HA SIDO POSIBLE EL CAMBIO|"
sed  '$d ' $_path_conf/pf.conf > $_path_conf/pf.conf_tmp
cat $_path_conf/pf.conf_tmp $_path_conf/pf.conf
$_path_conf/pf.conf_tmp
sleep 2
main
 fi
 }

control_conexiones ()  {
variables_install
clear
echo "\t|*****************************************|"
echo "\t|      ADMINISTRACION DE CONEXIONES V1.0    |"
echo "\t|*****************************************|"
sleep 3

cat $_FLAGS_/flag_ctl_conn | grep  'activo ' 2> 2 > 1
if [ $? ==  '0 ' ];then
echo "\n"
echo "\t \033[1;38mLAS REGLAS DE FILTRADO YA HAN SIDO CONFIGURADAS \033[0;0m"
sleep 2
echo "\n"
```

```
echo "'\t\033[1;38mA CONTINUACION SE MOSRTARAN LAS REGLAS DE FILTRADO ACTUALES:\033[0;0m"
sleep 2
pfctl -sr
sleep 1
echo "\n"
echo " Deseas modificar el limite de conexiones hacia internet\ ?: [s/n] \c "
read filter_rule
if [ $filter_rule == "s" ];then
echo "\n"
echo "\t\033[1;38m IMPORTANTE:\033[0;0m "

echo " AL MODIFICAR ESTE PARAMETRO EL
INTRUSO TENDRA MAS LIBERTAD DE REALIZAR CONEXIONES
HACIA EL EXTERIOR, ESTO PUEDE ATENTAR"

echo " CONTRA LA ADMINISTRACION DE LA PROPIA HONEYNET O SISTEMAS EXTERNOS"
sleep 4
echo "\033[1;38m Cargando, espere...\033[0;0m"
sleep 2
while :
 do
    clear
    echo "\033[1;38m |PORFAVOR INDICA EN QUE HONEYPOT RELIZARAS EL CAMBIO| \$033[0;0m"
    echo "\n"
    echo "|****************************************|"
    echo "| [1 ] WEBSERVER                 |"
    echo "| [2 ] MAILSERVER                      |"
    echo "| [3 ] BDSERVER               |"
    echo "| [4 ] Salir                          |"
    echo "|****************************************|"
    echo -n "Elige tu opcion: \c"
    read opcion_mod

    case $opcion_mod in
      1) echo "Indica el limite de conexiones para el Honeypot WEBSERVER: \c"
 read conn_limit_web
 export conn_limit_web
 conexiones_web  ;;
     2) echo "\nIndica el limite de conexiones para el Honeypot MAILSERVER: \c"
 read conn_limit_mail
 conexiones_mail ;;
     3) echo "\nIndica el limite de conexiones para el Honeypot BDSERVER: \c"
 read conn_limit_bd
 conexiones_bd ;;
     4) main ;;
     *) echo "\tERROR: Opcion no valida";
 sleep 2

    esac
done
else
main
fi
else
echo "\n"
echo "\t \033[1;38mINFO:\033[0;0m "
```

```
echo "\n"
echo "\tEL PROGRAMA POR RAZONES DE SEGURIDAD
PROCESARA UNICAMENTE 6 CONEXIONES DESDE LOS HONEYPOTS HACIA INTERNET"

echo "\t LAS CONEXIONES POSTERIORES SEREAN CANCELADAS."
  echo "\t ADEMAS SE CARGARAN POLITICAS DE ACCESO PREDEFINIDO
  EN EL HONEYWALL | LAS CONEXIONES HACIA INTERNET ES UN VALOR CONFIGURABLE|"
sleep 6
echo "\n"
echo "\t\033[1;38mCargando configuracion | \033[0;0m  \c"
sleep 1

# LAS SIGUIENTES LINEAS DE CODIGO ELIMINARAN
 TODAS AQUELLAS OPCIONES QUE NO OCUPAREMOS EN EL HONEYWALL #

sed -e  '1,$d ' $_path\_conf/pf.conf > $_path_conf/pf.conf_tmp
echo "* \c"
sleep 1
cat $_path_conf/pf.conf_tmp > $_path_conf/pf.conf
echo "* \c"
echo "# ===================================== #" >> $_path_conf/pf.conf
echo "# REGLAS DE FILTRADO PARA EL HONEYWALL \#" >> $_path_conf/pf.conf
echo "# ===================================== #" >> $_path_conf/pf.conf

# SE AGREGAN LAS INTERFACES DE RED CON LAS QUE TRABAJARA EL SISTEMA #

echo "interfaz_ext= "$_interfaz_ext" " >> $_path_conf/pf.conf
echo "interfaz_int= "$_interfaz_int" " >> $_path_conf/pf.conf
echo "* \c"
sleep 1

# DEFINICION DE LAS POLITICAS DE ACCESO POR DEFECTO #

echo "# DEFINICION DE LA POLITICA DE ACCESO POR DEFECTO #" >> $_path_conf/pf.conf
echo "pass in log on $_interfaz_ext all" >> $_path_conf/pf.conf
echo "pass out log on $_interfaz_ext all" >> $_path_conf/pf.conf
echo "* \c"
sleep 1

# DEFINICION DE LAS POLITICAS DE NO-ACCESO POR DEFECTO #

echo "# DEFINICION DE LAS POLITICAS DE NO-ACCESO POR DEFECTO #" >> $_path_conf/pf.conf
echo "block in log on $_interfaz_int all" >> $_path_conf/pf.conf
echo "block out log on $_interfaz_int all" >> $_path_conf/pf.conf
echo "* \c"
sleep 1

# DEFINICIO DE LA TABLA DE ABUSOS CONTRA CONEXIONES #

echo "# DEFINICIO DE LA TABLA DE ABUSOS CONTRA CONEXIONES #" >> $_path_conf/pf.conf
echo "table <abusive_hosts> persist" >> $_path_conf/pf.conf
echo "block in quick from <abusive_hosts>" >> $_path_conf/pf.conf
# DEFINICION DE LAS MACROS PARA EL MANEJO DE VARIBLES #

echo "# DEFINICION DE LAS MACROS PARA EL MANEJO DE VARIBLES #" >> $_path_conf/pf.conf
echo "HN_WEB_SERVER=$HN_WEB_SERVER" >> $_path_conf/pf.conf
```

```
echo "HN_MAIL_SERVER=$HN_MAIL_SERVER" >> $_path_conf/pf.conf
echo "HN_BD_SERVER=$HN_BD_SERVER" >> $_path_conf/pf.conf
echo "* \c"
sleep 1

# DEFINICION DE LAS POLITICAS DE ACCESO
PARA LA COMUNICACION DE LOS HONEYPOTS CON
SEBEK-SERVER POR EL PROTOCOLO UDP #

echo "# POLITICAS DE FILTRADO PARA LA COMUNICACION
 ENTRE LOS HONEYPOTS Y EL SEBEK-SERVER #" >> $_path_conf/pf.conf
  insert_rules_sebek$ "in" "$_interfaz_int" "udp" "$HN_WEB_SERVER" $SBK_SERVER"
insert_rules_sebek$ "in" "$_interfaz_int" "udp" "$HN_MAIL_SERVER$" "$SBK_SERVER"
insert_rules_sebek$ "in" "$_interfaz_int" "udp" "$HN_BD_SERVER$" "$SBK_SERVER"

# DEFINICION DE LAS VARIABLES QUE SOLO
PERMITIRAR 6 CONEXIONES COMO MAXIMO HACIA EL EXTERIOR #

insert_rules "in" "$_interfaz_int$" "tcp" "$HN_WEB_SERVER$" "6"
insert_rules "in" "$_interfaz_int$" "tcp" "$HN_MAIL_SERVER$" "6"
insert_rules "in" "$_interfaz_int$" "tcp" "$HN_BD_SERVER$" "6"

echo "* \c"
pfctl -f $_path_conf/pf.conf
if [ $? ==  '0 ' ];then
echo "* \c"
else
echo "\033[1;38mERROR: No ha sido posible cargar las reglas
| CONSULTE AL ADMINISTRADOR | \033[0;0m "
main
fi
sleep 1
echo "\033[1;38m | \033[0;0m \c"
sleep 1
echo "100\%"
echo "\n"
echo "\t \033[1;38m |* La configuracion ha sido cargada con exito,
 para modificar par\'{a}metros vuelva a ingresar a
 esta opci\'{o}n *| \033[0;0m"

sleep 4
echo "activo" > $_FLAGS_/flag_ctl_conn
fi
 }

# FUNCION QUE REALIZA LA INSTALACION POR PRIMERA VEZ EN EL SISTEMA #

install ()  {
variables_install
clear

sleep 2
echo "\t|******************************************************|"
echo "\t|    INSTALADOR DE CONTROL DE CONEXIONES  V1.0          |"
echo "\t|******************************************************|"
echo "\n"
```

```
sleep 3

# EL SIGUIENTE CODIGO GENERA DIRECTORIOS EN
DONDE SE ALOJARAN ARCHIVOS IMPORTANTES DEL SISTEMA #

create_directory "HONEYWALL" "FLAGS" "backup_bridge" "back_main"
variables_install

# A CONTINUACION SE PROCEDE CON LA REALIZACION
DE RESPALDOS PARA GARANTIZAR UN ROLLBACK COMPLETO #

make_backups_honeywall "pf.conf"
make_backups_honeywall "rc.conf"
make_backups_honeywall "sysctl.conf"
cat $_FLAGS_/flag_main | grep \ 'activo\ '
if [ $? ==  '1 ' ];then

touch $_path_conf/bridgename.bridge0
echo "Se ha creado el archivo $_path_conf/bridgename.bridge0" >> $_PATH_/status.txt
if [ -f $_path_conf/hostname.$_interfaz_ext ];then
echo "\t Interfaz $_interfaz_ext correcta" >> $_PATH_/status.txt
else
touch $_path_conf/hostname.$_interfaz_ext
fi
if [ -f $_path_conf/hostname.$_interfaz_int ];then
echo "\t Interfaz $_interfaz_ext correcta" >> $_PATH_$/status.txt
else
touch $_path_conf/hostname.$_interfaz_int
fi
echo "\t Configurando monitor del Honeywall \c"

cat $_FLAGS_/flag_dump | grep \ 'activo\ ' 2> 2 > 1
if [ $? ==  '0 ' ];then
sleep 2
else
echo "*  *  *  *  *    tcpdump -n -e -ttt -i pflog0 -w
$_PATH_/dump_file.txt" >> $_path_cron/root
echo "\t\033[1;38m [  OK  ]\033[0;0m "
fi
echo "\n"
echo "\tConfigurando logs del sistema \c"
chmod +x $_PATH_/monitor_sistema.sh
cat $_FLAGS_/flag_monitor | grep  'activo ' 2> 2 > 1
if [ $? ==  '0 ' ];then
sleep 2
else
ifconfig pflog0 up
echo "* * * * *   sh $_PATH_/monitor_sistema.sh"
 >> $_path_cron/root
echo "\t\033[1;38m [  OK  ]\033[0;0m "
fi

\# EL CODIGO A CONTINUACION GENERARA E INSTALARA LA BASE DE DATOS EN SQLITE \#
\# COMO PRIMER PASO SE CREA UNA BASE DE DATOS
PARA LA GESTION Y AUTENTICACION DE USUARIOS \#
```

```
echo "\n"
echo "\t Creando la Base de Datos \c"
cd $_PATH_
sqlite3 valida.db  "create table usuarios (password INTEGER);"
sleep 2

\# SE PROCEDE CON LA CREACION DE LAS TABLAS WEBSERVER, MAILSERVER
Y BDSERVER QUE CONTENDRAN TODA LA INFORMACION RELACIONADA CON LOS HONEYPOTS \#

sqlite3 honeypots.db "create table webserver
(id_honeypot TEXT, ip_add varchar, pto_sbk_cl INTEGER, estatus TEXT,max_conn_limit INTEGER);"
sleep 2

sqlite3 honeypots.db "create table mailserver
(id_honeypot TEXT, ip_add varchar, pto_sbk_cl INTEGER, estatus TEXT, max_conn_limit INTEGER);"
sleep 2

sqlite3 honeypots.db "create table bdserver
(id_honeypot TEXT, ip_add varchar, pto_sbk_cl INTEGER, estatus TEXT,max_conn_limit INTEGER);"

sleep 2

# GENERANDO LOS DATOS EN LA BASE DE DATOS #

sqlite3 valida.db  "insert into usuarios (password) values (83770);"
sqlite3 honeypots.db  "insert into webserver
(id_honeypot,ip_add,pto_sbk_cl,estatus,max_conn_limit) values
( 'webserver ', '132.248.209.148 ',7811, 'indefinido ',6);"
sqlite3 honeypots.db  "insert into mailserver
(id_honeypot,ip_add,pto_sbk_cl,estatus,max_conn_limit) values
( 'mailserver ', '132.248.209.149 ',7811, 'indefinido ',6);"
sqlite3 honeypots.db  "insert into bdserver
(id_honeypot,ip_add,pto_sbk_cl,estatus,max_conn_limit) values
( 'bdserver ', '132.248.209.152 ',7811, 'indefinido ',6);"
echo "\t\033[1;38m [  OK  ]\033[0;0m"
echo "\n"

echo "\t Generando prueba de conexion \c"
$_test_bd=‘sqlite3 valida.db "select * from usuarios";‘
export _test_bd
if [ $? ==  '0 ' ];then
  sleep 2
  echo "\t\033[1;38m [  OK  ]\b033[0;0m"
sleep 2
echo "\n"
echo "\tLa Base de Datos ha sido instalada con exito"
cd $_PWD_
sleep 2
echo " Espere porfavor..."
echo "activo" > /root/FLAGS/flag_main
sleep 4
main2
else
echo "\n"
echo "tError: ha ocurrido un error en la prueba de conexion...
|CONTACTAR AL ADMINISTRADOR|"
```

```
exit 0
fi
   else
echo "|EL PROGRAMA YA HA SIDO INSTALADO|"
sleep 3
main
 fi
 }

\# LAS SIGUIENTES FUNCIONES SON LAS PRICIPALES Y SE
ENCARGAN DE INTERCONECTAR A TODAS LAS FUNCIONES ANTERIORES Y PERMITEN UNA \#

\# UNA NAVEGACION TOTAL POR TODO EL PROGRAMA \#

main ()  {
variables_install
while :
 do
    clear
    echo "|*************************|"
    echo "|      MENU PRINCIPAL       |"
    echo "|*************************|"
    echo "|*****************************************|"
    echo "|[1] Administracion del Honeywall           |"
    echo "|[2] Honeypots                            |"
    echo "|[3] Control de Conexiones y Datos          |"
    echo "|[4] Regresar                           |"
    echo "|*****************************************|"
    echo -n "Elige tu opcion: \c"
    read opcion
    case \$opcion in
      1) honeywall ;;
      2) honeypots ;;
      3) control_conexiones ;;
      4) main3 ;;
      *) echo "ERROR: Opcion no valida";
 sleep 2
 esac
done
 }

\# FUNCION QUE PRESENTA LA PANTALLA DE ACCESO AL PROGRAMA Y VALIDA AL USUARIO \#

main2 ()  {
variables_install
while true; do
clear

echo -e "|*********************************************|"
echo -e "| HONEYNETS DE SGUNDA GENERACION BAJO OPENBSD |"
echo -e "|       CONTROL DE DATOS Y CONEXIONES         |"
echo -e "|         JOSE ALBERTO AVALOS VELEZ           |"
echo -e "|*********************************************|"


echo -ne "Password: "
```

```
stty -echo
read pass
stty echo
\# HACIENDO LA CONSULTA AL BASE DE DATOS PARA VERIFICAR LA AUTENTICIDAD DEL USUARIO \#
if [ "$_test_bd" == "$pass" ]; then
main
else
echo "\t\033[1;25mError:Password no valido\033[0;0m"
sleep 1
clear
fi
done
 }

main3 () \ {
variables_install
while :
 do
    clear
    echo "|**************************************|"
    echo "|     HONEYNETS DE GENERACION II       |"
    echo "|**************************************|"
    echo "\n"
    echo "|*********************************************|"
    echo "|[1] Instalar                          |"
    echo "|[2] Ejecutar Programa                       |"
    echo "|[3] Restaurar           |"
    echo "|[4] Salir                              |"
    echo "|*********************************************|"
    echo -n "tElige tu opcion: \c"
    read opcion
    case $opcion in
      1) install ;;
      2) main2 ;;
      3) rollback ;;
      4) exit 0 ;;
      *) echo "ERROR: Opcion no valida";
 sleep 2
    esac
done
 }

# FINALMENTE MANDAMOS INVOCAR A LA FUNCION MAIN 3 QUE ES LA QUE INVOCARA
A TODAS LAS ANTERIORES #

main3
```