UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

"La Universidad Católica de Loja"

Escuela de Ciencias de la Computación

ESTUDIO DE TRÁFICO DE ATAQUES A LA RED DE LA UTPL MEDIANTE UN PROTOTIPO DE HONEYPOT

TESIS PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Autor:
Henry Fernando Montalván Celi

Director:
Msc. María Paula Espinoza

Loja - Ecuador

2009

CESIÓN DE DERECHO

Yo, **Henry Fernando Montalván Celi,** declaro conocer y aceptar la disposición del Art. 67 del Estatuto Orgánico de la Universidad Técnica Particular de Loja, que en su parte pertinente textualmente dice: "Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado que se realicen a través o con el apoyo financiero, académico o institucional (operativo) de la Universidad".

_____

Henry F. Montalván C.

Msc.

María Paula Espinoza

DOCENTE INVESTIGADOR DE LA ESCUELA DE CIENCIAS DE LA COMPUTACIÓN DE LA UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

CERTIFICA:

Que una vez concluido el trabajo de investigación con el tema **"ESTUDIO DE TRÁFICO DE ATAQUES A LA RED DE LA UTPL MEDIANTE LA IMPLEMENTACIÓN DE UN PROTOTIPO DE HONEYPOTS"** previa la obtención del título de Ingeniero en Sistemas Informáticos y Computación, realizado por el señor Henry Fernando Montalván Celi egresado de la Escuela de Ciencias de la Computación; haber dirigido, supervisado y asesorado en forma detenida cada uno de los aspectos de la tesis de pregrado.

Además, en mi calidad de DIRECTOR DE TESIS y al encontrar que se han cumplido con todos los requisitos investigativos, autorizo su presentación y sustentación ante el tribunal que se designe para el efecto.

Atentamente,

- -

Ing. María Paula Espinoza

DIRECTOR DE TESIS

AUTORÍA

Las ideas, opiniones, conclusiones, recomendaciones y más contenidos expuestos en el presente informe de tesis son de absoluta responsabilidad del autor.

_____

Henry F. Montalván C.

DEDICATORIA

A mis padres y hermanos, por su cariño y apoyo incondicional.

Henry Fernando

AGRADECIMIENTOS

Mis sinceros agradecimientos a todas las personas que de una u otra manera han contribuido a la culminación de este proyecto, de manera especial a la Msc. María Paula Espinoza Directora de Tesis que con su conocimiento, empeño y motivación, fue la guía durante el desarrollo del presente trabajo de investigación.

Henry Fernando

INDICE DE CONTENIDOS

I. INTRODUCCIÓN.

En la actualidad los ataques a las redes de computadoras y los incidentes informáticos se incrementan día a día en mayor número y con mayor grado de eficacia en sus objetivos. La tarea principal que se ha venido realizando es la constante defensa en contra de ataques que pongan en riesgo los sistemas críticos de las organizaciones. Algunos de los métodos de seguridad actuales todavía utilizados son las practicas de actualización de antivirus, infraestructuras que incluyen Firewalls, IDS, VPN, etc., que son tareas de defensa que son válidas pero no alcanzan a mantener una seguridad altamente confiable, además dejando a un lado la capacidad proactiva que puede ayudar en un grado considerable a la seguridad de la organización. La acción reactiva en contra de estos ataques reduce la capacidad de aprender de estos mismos sucesos. Por lo tanto es muy importante el estudio y la elaboración de estrategias que permitan tener un grado adecuado para protegerse pero teniendo en cuenta la creación de recursos de investigación.

Un nuevo concepto de seguridad es el considerar, qué se debe proteger y principalmente de quién.

Una Honeynet tiene como objetivo reunir información sobre la actividad del intruso. De esta manera seremos capaces de detectar una vulnerabilidad antes de que sea explotada, además de conocer los riesgos a los cuales nuestros sistemas de producción están expuestos.

El administrador podrá utilizar y aplicar este conocimiento en la configuración de sus herramientas y equipos de seguridad como son los firewall, IDS, etc.

El presente proyecto de Tesis consiste en implementar un Honeypot especial denominado Honeynet, en la red de la Universidad Técnica Particular de Loja, con la finalidad de capturar y analizar tráfico en la red y poder establecer patrones de ataques.

Durante la etapa de investigación se fueron elaborando documentos que relatan el estado y resultados del proyecto como: **Implementación de una Honeynet para estudio de tráfico en la red de la UTPL**, **Honeynet como apoyo a la Investigación de Seguridad de Redes en Entornos Universitarios**; los cuales fueron publicados y expuestos en las conferencias del INFOSecurity en Quito y en el Primer Congreso Internacional de Computación y Telecomunicaciones en la cuidad de Lima - Perú.

II. ESTRUCTURA DE LA TESIS.

La estructura de la tesis es la siguiente:
Se inicia con una introducción general a la seguridad, al trabajo realizado y la publicación de los documentos que se han desarrollado.

En el capítulo 1 se trata el estado del arte de los Honeypots, definición y sus tipos especiales como son las Honeynets. Aquí se trata la definición, requerimientos, arquitecturas y funcionamiento de las Honeynets. Para seleccionar la ubicación y la arquitectura correcta en la red se realiza un análisis de las mismas.

En el capítulo 2 se hace un estudio de las herramientas que se utilizarán para el control, captura y análisis de datos que están incluidas en el software de la Honeynet como aquellas herramientas externas que fueron necesarias.

En el capítulo 3 se hace una descripción de las actividades para la implementación de la Honeynet en la red de la UTPL, la descripción de cada uno de los elementos que conforman la red trampa junto con sus requerimientos hardware, sistemas, servicios y la forma de operación de la Honeynet así como los diferentes problemas que surgieron durante este proceso.

Y finalmente en el capítulo 4 se detallan las actividades encontradas en el análisis del tráfico durante aproximadamente seis meses.

III. OBJETIVOS.

General:

- Implementar una HoneyNet en la Red de la UTPL.

Específicos:

- Diseñar e Implementar una Honeynet, como un recurso de seguridad proactivo para la red Universitaria.
- Generar un recurso de investigación que permita generar estadísticas reales sobre los ataques que sufren actualmente la red de datos de la Universidad.
- Crear un portal para dar visibilidad de los resultados obtenidos y estadísticas generadas luego del análisis de tráfico así como la publicación de nuevos proyectos.
- Presentar la experiencia y resultados a través de capacitaciones y/o participaciones.

INTRODUCCION.

En la actualidad el ataques a las redes de computadoras y los incidentes informáticos se incrementan día a día en mayor número y con mayor grado de eficacia en sus objetivos. La tarea principal que se ha venido realizando es la constante defensa en contra de ataques que pongan en riesgo los sistemas críticos de las organizaciones. Algunos de los métodos de seguridad actuales todavía utilizados son las practicas de actualización de antivirus, infraestructuras que incluyen Firewalls, IDS, VPN, etc, que son tareas de defensa que son válidas pero no alcanzan a mantener una seguridad altamente confiable, además dejando a un lado la capacidad proactiva que puede ayudar en un grado considerable a la seguridad de la organización. La acción reactiva en contra de estos ataques reduce la capacidad de aprender de estos mismos sucesos.

Por lo tanto es muy importante el estudio y la elaboración de estrategias que permitan tener un grado adecuado para protegerse pero teniendo en cuenta la creación de recursos de investigación.

Un nuevo concepto de seguridad para mantener una red integra es que se debe considerar que se debe proteger y principalmente de quien.

Una Honeynet tiene como objetivo reunir información sobre la actividad del intruso. De esta manera seremos capaces de detectar una vulnerabilidad antes de que sea explotada, además de conocer los riesgos a los cuales nuestros sistemas de producción están expuestos.

El administrador podrá utilizar y aplicar este conocimiento en la configuración de sus herramientas y equipos de seguridad como son los firewall, IDS, etc.

El presente proyecto de Tesis consiste en implementar un Honeypot especial denominado Honeynet, en la red de la Universidad Técnica Particular de Loja, con la finalidad de capturar y analizar tráfico en la red y poder establecer patrones de ataques.

Durante la etapa de investigación se fueron elaborando documentos que relatan el estado y resultados del proyecto como: **Implementación de una Honeynet para estudio de tráfico en la red de la UTPL, Honeynet como apoyo a la Investigación de Seguridad de Redes en Entornos Universitarios** los cuales fueron publicados y expuestos en las conferencias del INFOSecurity2008 en Quito y en el Primer Congreso Internacional de Computación y Telecomunicaciones en la cuidad de Lima Perú.

ESTRUCTURA DE LA TESIS

La estructura de la tesis es la siguiente:
Se inicia con una introducción general a la seguridad, al trabajo realizado y las publicaciones de los documentos que se han realizado,

En el capítulo 1 se trata el estado del arte de los Honeypots, definición y sus tipos especiales como son las Honeynets. Aquí se trata la definición, requerimientos, arquitecturas y funcionamiento de las Honeynets. Para seleccionar la ubicación y la arquitectura correcta en la red se realiza un análisis de las mismas.

En el capítulo 2 se hace un estudio de las herramientas que se utilizarán para el control, captura y análisis de datos que están incluidas en el software de la Honeynet como aquellas herramientas externas que fueron necesarias.

En el capítulo 3 se hace una descripción de las actividades para la implementación de la Honeynet en la red de la UTPL, la descripción de cada uno de los elementos que conforman la red trampa junto con sus requerimientos hardware, sistemas, servicios y la forma de operación de la Honeynet así como los diferentes problemas que surgieron durante este proceso.

Y finalmente en el capítulo 4 se detallan las actividades encontradas en el análisis del tráfico durante aproximadamente seis meses.

CAPÍTULO I:

ESTADO DEL ARTE

1.1 HONEYPOTS.

1.1.1 INTRODUCCIÓN

En la actualidad las organizaciones disponen de información importante, fundamental en el desarrollo y crecimiento de sus actividades, es por ello que deben mantener la infraestructura necesaria para la detección de posibles intentos e intrusiones en los sistemas que la organización maneja.

Hoy en día existen aproximadamente 134 millones de internautas [1] a nivel de América del Sur, sin embargo; gran cantidad de usuarios, administradores o proveedores conectados a la red no tienen un conocimiento claro de las debilidades o vulnerabilidades de seguridad a las que está expuesta su información ni del crecimiento que ha tenido la actividad de la "Comunidad Black Hat" [2] en los últimos años. De acuerdo en un informe presentado por el Computer Security Institute hasta el 2006 se incrementó el número de ataques y se perdieron cerca de 15 millones de dólares únicamente por infección de virus sin considerar las pérdidas ocasionadas por otros tipos de ataques. [3]

A nivel de Latinoamérica: Brasil, México, Perú, Chile y Argentina cuentan con equipos formales que trabajan en temas de seguridad y que reportan datos estadísticos sobre los comportamientos detectados en cada país.

A nivel de Ecuador, no existe un registro oficial y público sobre el número de incidentes o ataques de seguridad que se presentan. Muchas empresas que ofrecen servicios de seguridad manejan cifras que por cuestiones de confidencialidad no pueden ser expuestas y, por tanto no permiten tener una visión sobre la situación actual del país en cuanto a ataques o incidentes de seguridad. A pesar de que el Proyecto Honeynet capítulo Ecuador (www.honeynet.ec) [4] ha arrancado no se cuenta con una infraestructura global que integre diversas redes.

Por ello, se ha visto en la tecnología Honeypot una herramienta complementaria de seguridad valiosa, no solo por aportar a una seguridad proactiva, sino por constituir un potencial recurso de investigación que permite afrontar la seguridad desde una perspectiva que se alinea a los objetivos de las Universidades. Por tanto, el principal enfoque que le daremos a éste proyecto, será investigativo, con lo que conseguiremos tener un conocimiento amplio sobre el comportamiento de los ataques, perfiles de los atacantes y datos estadísticos a fin de aportar a otros grupos investigativos con información válida para el desarrollo de proyectos relacionados.

1.1.2 DEFINICIÓN DE HONEYPOTS.

Lance Spitzner define a un Honeypot como un recurso de red destinado a ser atacado o comprometido [5] con la finalidad de obtener información acerca de los métodos que utiliza un atacante para ingresar.

Es una trampa que se utiliza para identificar, evitar y, en cierta medida, neutralizar los intentos de secuestrar los sistemas de información y redes. Ejemplo: se puede engañar al hacker para que piense que se encuentra frente a una organización importante que contiene información interesante o valiosa y que es un sistema poco seguro.

Un Honeypot es generalmente utilizado como un sistema de vigilancia o monitoreo, así como un mecanismo de alerta temprana, la tarea mayor de un Honeypot es supervisar toda la actividad ilícita del intruso en el dicho sistema.

Características:

- ✓ La información que se genera es de gran valor, la misma que se utiliza para desarrollar métodos preventivos para evitar estos ataques.
- ✓ Dispone de nuevas herramientas y tácticas que son diseñadas para capturar el tráfico que interactúa con ellos, con esto se provee un alto grado de detección de intrusos, y de esta forma se evita que se den falsos positivos, es decir alertas falsas.
- ✓ Tiene la capacidad de recopilar información de manera detallada de los intrusos internos y externos, lo que permite detectar potenciales riesgos sobre la red.
- ✓ Se requiere de una dirección IP pública como mínimo para su funcionamiento.

1.1.3 TIPOS DE HONEYPOTS.

1.1.3.1 Por su Función.

Honeypots de Producción.

Este tipo de Honeypot es diseñado principalmente para la seguridad y defensa de las redes. No han sido diseñados para recoger información sobre las actividades de hacking, se implementa de manera colateral a las redes de datos o infraestructuras en ambientes reales y son utilizados para proteger a las organizaciones. Este tipo de Honeypot está sujeto a ataques constantes.

Actualmente a este tipo de Honeypot se le da más importancia debido a las herramientas de detección que se pueden utilizar, así como también por la forma de cómo este tipo de Honeypot puede mejorar la protección en la red y en los hosts.

Honeypots de Investigación.

Estos Honeypots son implementados con la finalidad de recolectar información sobre las acciones de los intrusos.

Por lo general, son administrados por instituciones educativas sin fines de lucro, organizaciones de investigación, se utilizan para tener una visión más clara sobre las operaciones, las estrategias, los motivos de ataques, estudiar patrones de ataque y amenazas de todo tipo.

El objetivo principal es identificar las amenazas y encontrar el modo de tratar con estas de una manera eficaz. Estas son difíciles de manejar y desplegar, pero son capaces de reunir una gran cantidad de información.

Existen otros tipos de Honeypots que permiten definir un rango de posibilidades de ataques de un potencial atacante, esto ayuda a entender no solo el tipo de Honeypot con el que se está trabajando, sino también ayuda a definir cuáles de las vulnerabilidades se necesita que un atacante explote. Entre estos tipos tenemos los siguientes:

1.1.3.2 Por su grado de interacción.

Tabla 1.
Tipos de Honeypot según su nivel de interacción. [6]

NIVEL DE INTERACCIÓN	INSTALACIÓN Y CONFIGURACIÓN	DESARROLLO Y MANTENIMIENTO	RECOLECCIÓN DE INFORMACIÓN	NIVEL DE RIESGO
BAJO	Fácil	Fácil	Limitado	Bajo
MEDIO	Considerable	Considerable	Variable	Medio
ALTO	Difícil	Alto consumo de tiempo	Extensivo	Alto

Honeypots de Baja Interacción.

Un Honeypot de baja interacción como su término lo describe, proporciona una interacción limitada entre el atacante y el Honeypot. La cantidad de información recogida es limitada y simplemente se puede obtener básicamente lo siguiente:

- ✓ Hora y fecha de ataque
- ✓ Dirección IP de origen y puerto de origen de la conexión
- ✓ Dirección IP de destino y puerto destino de la conexión. [7]

Estos tipos de Honeypots trabajan únicamente emulando servicios y sistemas operativos. La actividad del atacante se encuentra limitada al nivel de emulación del Honeypot. La ventaja de un Honeypot de Baja Interacción radica principalmente en su simplicidad, ya que estos tienden a ser fáciles de utilizar y mantener con un riesgo mínimo. [8]

El proceso de implementación de un Honeypot de Baja Interacción consiste en instalar un software de emulación de sistema operativo, elegir el sistema operativo y el servicio a emular, establecer una estrategia de monitoreo y dejar que el programa opere por si solo de manera normal.

Los servicios emulados mitigan el riesgo de penetración, conteniendo la actividad del intruso que nunca tiene acceso al sistema operativo real donde puede atacar o dañar otros sistemas. [8]

Entre algunos ejemplos de Honeypots de Baja Interacción tenemos los siguientes:

1. **Specter:** [9]

   Consiste en un PC dedicado y el software de Specter. Es un Honeypot inteligente basado en sistemas de detección de intrusos, vulnerables y atractivos a los atacantes, este sistema proporciona servicios como PHP, SMTP, FTP, POP3, HTTP, y TELNET que atraen facilmente a los atacantes, pero en realidad son trampas que pretenden recolectar información. Algunas ventajas de utilizar Specter son:

   - ✓ Actividades sospechosas de interés en sus redes se pueden detectar inmediatamente.
   - ✓ Los administradores son notificados de actividad hostil cuando esto ocurre, por lo que pueden ver inmediatamente el problema y tomar medidas.

2. **Honeyd:** [10]

El Honeyd es un Honeypot que crea host virtuales en la Red. Los anfitriones pueden ser configurados para ejecutar servicios arbitrarios y su personalidad puede ser adaptado de modo que parezcan estar ejecutando ciertos sistemas operativos. Honeyd mejora la seguridad Cibernética proporcionando mecanismos para la detección y evaluación.

3. **KFSensor:** [11]

El KFSensor es un Honeypot de Windows basado en sistema de detección de intrusos (IDS)**,** actúa para atraer y detectar piratas informáticos y gusanos, vulnerables mediante la situación de los servicios del sistema y troyanos.

4. **PatriotBox:** [12]

El PatriotBox usa como señuelo el sistema de detección de intrusos (IDS), en entornos de red empresarial de forma efectiva la detección temprana de las amenazas de intrusos. También utiliza ayuda para reducir el spam en internet ya que simula un servidor de correo de retrasmisión abierta.

Honeypots de Alta Interacción.

Estos tipos de Honeypots constituyen una solución compleja, ya que implica la utilización de sistemas operativos y aplicaciones reales montados en hardware real sin la utilización de software de emulación e involucrando aplicaciones reales que se ejecutan de manera normal.

Con este tipo de Honeypot se tiene la posibilidad de capturar grandes cantidades de información referentes al modus operandi de los atacantes debido a que los intrusos se encuentran interactuando frente a un sistema real.

Los Honeypots de Alta Interacción no asumen responsabilidad sobre el comportamiento que tendrá el atacante, ya que se provee de un entorno abierto que captura todas las actividades realizadas, además ofrece una amplia gama de servicios, aplicaciones y depósitos de información que pueden servir como blanco potencial para aquellos servicios que se desea comprometer.

Esta capacidad también incrementa el riesgo de que los atacantes puedan utilizar estos sistemas como entradas para lanzar ataques a sistemas internos que no forman parte de los Honeypots.

Una Honeynet [13] es un Honeypot de Alta Interacción.

Entre algunos ejemplos de Honeypots de Alta Interacción tenemos:

1. **ManTrap:** [14]

   Puede crear "software jaula" y simular una red virtual de una máquina. Para ello emula una variedad de diferentes máquinas (FTP, HTTP, SMTP, ODBC) en una sola ManTrap de acogida. Este Honeypot es configurable para enviar una gran variedad de alertas a cualquier dirección e-mail o a un dispositivo con capacidad SNMP, para alertar a los administradores del sistema que alguien ha entrado a la "jaula".

   En un Host con ManTrap, todo reside físicamente en el mismo sistema. Este Host soporta hasta cuatro jaulas lógicas. Cada jaula actúa como un sistema operativo virtual independiente con su propia interfaz de red física que se conecta a la red.

1.1.4 UTILIDADES DE LOS HONEYPOTS.

- ✓ Los Honeypots son útiles para investigaciones forenses, ya que permiten analizar la actividad del hacker o atacante basados en el engaño. Si ya se conoce la identidad del atacante y además se va a tomar acciones en contra del atacante es importante recordar que antes de implementar un Honeypot se debe tener un permiso judicial contra el atacante.
- ✓ Brindan protección, prevención, detección y respuesta a los ataques de baja interacción en sistemas de producción.
- ✓ Facilitan la recolección de la información, esto ayuda a definir tendencias respecto de las actividades del atacante, activación de sistemas tempranos de alarma, predicción de ataques e investigaciones criminales con alta interacción.
- ✓ Permite la detección de ataques "0 days"[1], dado que su objetivo fundamental es la construcción de un perfil del atacante.

1.1.5 VALOR DE LOS HONEYPOTS.

1.1.5.1 Ventajas.

Seguidamente se nombra algunas de las ventajas.

- ✓ Los Honeypots son una tecnología con un concepto muy simple y con una fortaleza muy poderosa.

[1] Es un ataque contra un ordenador que trata de explotar las vulnerabilidades de aplicaciones. Normalmente estas vulnerabilidades son desconocidas, no están publicadas o no existe un parche que las solvente

- ✓ Frente a un IDS[2] la tecnología Honeypot tiene una gran ventaja, ya que este permite la detección de nuevos tipos de ataques, a diferencia de un IDS que se basa en patrones que ya están definidos para realizar la detección de los ataques conocidos.
- ✓ Los recursos necesarios a utilizar son mínimos, de esta forma se puede implementar una plataforma lo suficientemente potente para operar a gran escala. Ejemplo: Una computadora con un procesador Pentium con 128 Mb de RAM puede manejar fácilmente una red de clase B entera.
- ✓ Trabaja en entornos sobre IPv6[3], es decir Honeypot detectará un ataque sobre IPv6 de la misma forma que lo hace con un ataque sobre IPv4.
- ✓ Generan información de mucho valor, cosa que no sucede con otros sistemas de seguridad que generan cientos de megas de ficheros logs[4] con información que no es nada útil.
- ✓ El tráfico cifrado dirigido a los sistemas Honeypot es fácil de analizar.

1.1.5.2 Desventajas.

Como toda tecnología, los Honeypots también presentan debilidades inherentes a su diseño y funcionamiento.

Esto se debe a que éstos no reemplazan a las tecnologías actuales, sino que trabajan con las tecnologías existentes:

- ✓ Los Honeypots solo permiten rastrear y capturar actividad destinada a interactuar directamente con ellos, pues estas no capturan información relacionada a ataques destinados hacia otros sistemas vecinos, a menos que el atacante o la amenaza interactúe con el Honeypot al mismo tiempo.
- ✓ Los Honeypots pueden llegar a constituir un riesgo potencial para la red, esto debido a la atracción que se genera a los posibles atacantes, es por ello que si no se configura adecuadamente el alcance del Honeypot y se lo convierte en un entorno controlado y cerrado, puede ser utilizado como punto de inicio para ataques a otras redes o incluso a la misma red.
- ✓ El valor de un Honeypot termina cuando es detectado, una vez que este ha sido detectado el intruso podrá evitarlo o incluir información errónea que desvíe toda

---

[2] **IDS** (*Intrusion Detection System)* Un **sistema de detección de intrusos** es un programa usado para detectar accesos desautorizados a un computador o a una red.

[3] **IPv6** es la versión 6 del Protocolo de Internet, un estándar en desarrollo del nivel de red encargado de dirigir y encaminar los paquetes a través de una red.

[4] **Logs** son registros oficiales de eventos durante un periodo de tiempo en particular.

investigación. Todo Honeypot es detectable, ya que existen intrusos con los suficientes conocimientos para ello, solo es cuestión de tiempo y paciencia.

1.2 HONEYNETS

1.2.1 INTRODUCCIÓN.

Este capítulo contiene datos generales de lo que es una Honeynet, sus requerimientos críticos como el control y la captura de datos. Se evaluarán los tipos de arquitectura que existen para luego determinar cuál es la que se va a elegir para la implementación del proyecto. También se pone a conocimiento las implicaciones legales y los problemas que surgen al desarrollar una Honeynet.

1.2.2 DEFINICIÓN DE HONEYNETS.

- ✓ Una Honeynet es un tipo de *Honeypot* de alta interacción diseñado específicamente para la investigación, esta recoge información sobre atacantes.
- ✓ Una Honeynet es una arquitectura, no es un producto o un software determinado.
- ✓ Una Honeynet es una red de varios sistemas y aplicaciones las cuales son investigadas y atacadas por los blackhats[5]. Las Honeynets pueden utilizar varios sistemas al mismo tiempo, como Solaris, Linux, Windows NT, router, conmutadores, etc. Esto crea un entorno de red que refleja de forma más realista una red productiva. Además, al tener diferentes sistemas con diferentes aplicaciones, como un servidor DNS en Linux, un servidor Web Windows IIS (Internet Information Server), y un servidor de bases de datos en Solaris, podemos aprender sobre diferentes herramientas y tácticas. Quizás algunos blackhats se centran en sistemas específicos, aplicaciones o vulnerabilidades. Teniendo una variedad de sistemas operativos y aplicaciones, somos capaces de trazar con más exactitud el perfil de las tendencias y rasgos de los blackhats. [13]
- ✓ El objetivo es el de hacer creer al atacante que está ante una red “real”, entonces de deben añadir los distintos elementos que conforman una arquitectura de red. [15]
- ✓ Los sistemas de seguridad tradicionales como los IDs, firewalls son de carácter correctivo ya que al detectar intrusiones estos de inmediato toman medidas de corrección, mientras que las Honeynets son de carácter proactivo y se dedican al

[5] También llamados "crackers", son los hackers que se especializan en el ingreso no autorizado. Pueden utilizar las computadoras para atacar los sistemas con fines de lucro, por diversión o por motivaciones políticas o como parte de una causa social. Tal ingreso a menudo implica la modificación y/o destrucción de datos, y se realiza sin autorización.

estudio de estos ataques y atacantes para obtener patrones de comportamiento y nuevos métodos de ataque.

1.2.3 REQUISITOS Y REQUERIMIENTOS DE UNA HONEYNET.

Uno de los mayores problemas que tienen los sistemas tradicionales como los firewalls, IDs, es que se tiene que hacer un control, una depuración de los eventos registrados por este tipo de sistemas con la finalidad de comprobar falsos positivos y falsos negativos. Las Honeynets aplican un concepto simple para evitar lo anteriormente mencionado, son redes controladas y monitoreadas, diseñadas para ser atacadas y comprometidas, por lo tanto toda actividad que se registre en estos sistemas son catalogados como tráfico sospechoso.

Para implementar una Honeynet de manera satisfactoria, se deben llevar a cabo correctamente dos requisitos o elementos críticos: *el control de datos y la captura de datos*.

1.2.3.1 Control de Datos.

Este es uno de los aspectos más críticos de la implementación de la Honeynet. Hay que tener en cuenta que si la Honeynet es atacada con éxito, puede servir para atacar otros sistemas. Debemos estar preparados para resolver esta situación.

El Control de Datos es una actividad de contención. Cuando se trata con *blackhats* siempre hay riegos, se debe reducir este riesgo. Hay que asegurar que, cuando un Honeypot sea comprometido no pueda ser utilizado para dañar a algún sistema que no sea la Honeynet. Sin embargo, el reto es controlar el flujo de datos sin que el blackhat sospeche. Una vez que el sistema está comprometido, el blackhat a menudo querrá conectarse a Internet, para descargar herramientas, establecer conexiones IRC[6], o enviar correos electrónicos. Tenemos que darle flexibilidad para ejecutar estas acciones, y estos pasos son los que queremos aprender y analizar. Además, los blackhats pueden sospechar si observan que no pueden realizar conexiones al exterior. [13]

Para solucionar esto, es necesario permitir cierta cantidad de tráfico de salida desde la Honeynet al exterior, además de limitar los servicios a los cuales se puede conectar desde la Honeynet.

---

[6] **IRC (Internet Relay Chat)** es un protocolo de comunicación en tiempo real basado en texto que permite debates en grupo o entre dos personas y todos ellos pueden comunicarse entre sí sin tener que establecer comunicación de antemano.

El punto clave en el control de datos es la automatización. Suponga que un Honeypot es comprometido a las 2 de la mañana y se lanza un ataque DoS (Denegación de Servicios) hacia un equipo fuera de la Honeynet. Para cuando el administrador revise el estado de las bitácoras, el daño estará hecho y seguro tendría problemas por ello. Una idea práctica de llevar un control de datos adecuado es el verificar en el tráfico de salida la configuración de banderas en conexiones TCP (Transmission Control Protocol), o el número de paquetes UDP (User Datagram Protocol) que salen de un Honeypot, de igual manera habrá que verificar el tráfico ICMP (Internet Control Message Protocol) en caso de que se utilice para ataques de DoS (p.e Smurf). Cuando se exceda un umbral predefinido, podríamos agregar una regla al firewall para que impida más tráfico desde el Honeypot, y enviar una alerta al administrador ya sea por correo electrónico o por cualquier otro medio, para informarle de dicho suceso. [6]

1.2.3.2 Captura de Datos.

El siguiente paso para la construcción de una Honeynet es instalar algún tipo de herramienta que capture la actividad del sistema. Sebek se utilizará para registrar los detalles de las conversaciones entre los atacantes y la Honeynet.

El objetivo de este paso es recolectar la mayor cantidad de información tanto de las amenazas que atentan contra la red Honeynet, así como de cuáles son los motivos y comportamientos de los intrusos. Esto quiere decir que todo tráfico de entrada y salida así como la actividad dentro de cada equipo de la Honeynet deberá ser registrado para su posterior análisis.

Uno de los puntos más importantes en la captura de datos es la descentralización de los mecanismos de captura. Es decir que, debe existir una infraestructura de captura por capas, de manera que si un sistema de captura de datos llegase a fallar, no esté todo perdido. Otro punto importante es que por ningún motivo se debe almacenar información en los Honeypots, debido a que el intruso podría darse cuenta que se encuentra dentro de un Honeypot y borrar la información obtenida por el Honeypot, o peor aún, modificarla y así se obtendrá información errónea sobre el incidente. [7]

1.2.4 REQUERIMIENTOS DE UNA HONEYNET.

El objetivo de implementar la Honeynet es monitorear el tráfico de la red externa de la UTPL.

Los siguientes requerimientos son necesarios para implementar la red trampa:

***Honeynet en General*:** La honeynet debe cumplir los requisitos críticos como son el control, captura, análisis y recolección de datos.

Honeynet UTPL: Uno de los objetivos de la Honeynet es imitar en lo posible a la red en producción por lo que hay que tomar en cuenta las tecnologías de implementación basadas en ambientes reales.

Red: Adicional a las medidas de Control de Datos proporcionadas por la Honeynet, todo el tráfico de red generados por la red trampa debe pasar por un dispositivo (switch) que pueda ser apagado por el administrador de red en caso de emergencia. Para que los Honeypots dentro de las Honeynet puedan ser visibles por todo el mundo, cada uno debe ser configurado usando una IP pública con los principales puertos abiertos TCP, UDP que son proporcionadas por los administradores de red de la UTPL, inclusive algunos no tradicionales y que son usado para realizar ataques, esto permitirá llamar más la atención, recolección de mayor cantidad de datos y mayor interacción con los atacantes.

Hardware: Se necesitan los siguientes equipos para la red con los requisitos mínimos (revisados en el punto 3.2): Un Swith o Hub, Dos computadores que serviran como Honeypots, un computador que desempeñe el papel de Honeywall con tres interfaces de red, un equipo adicional que sirva para administración remota y se debe tomar en cuenta que los datos que se recolecten requieren gran capacidad de almacenamiento por lo que es indispensable discos duros externos adicionales de mínimo 200GB y el espacio físico para montar esta infraestructura.

1.2.5 ARQUITECTURAS DE UNA HONEYNET.

La arquitectura de las Honeynets ha ido mejorando con el avance del tiempo, dando lugar a dos generaciones de Honeynets denominadas Generacion I y Generacion II.

1.2.5.1 Generación I.

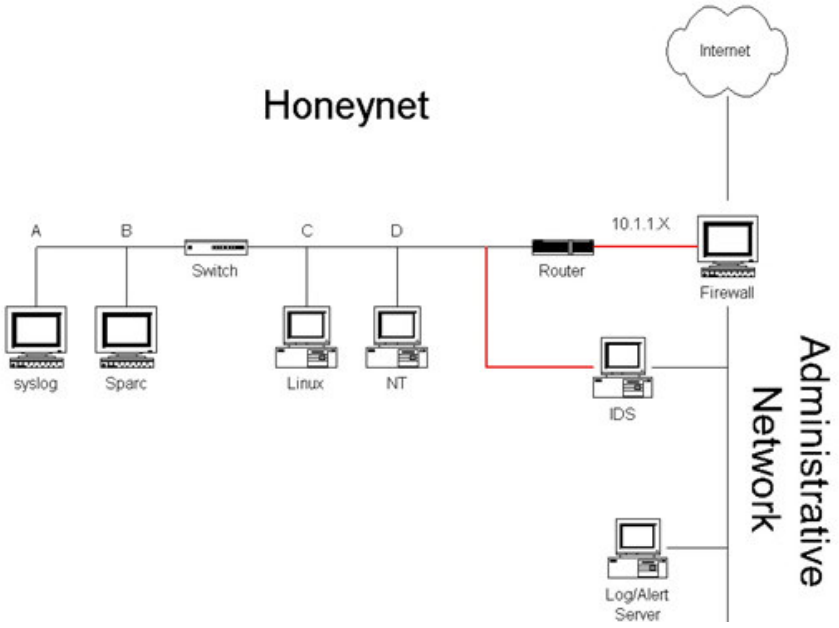


Figura 1. Arquitectura Honeynet – Generación I [16]

Las Honeynets de esta generación fueron las primeras que implantaron el control y la captura de datos con medidas simples pero eficientes. La Figura 1 representa el esquema gráfico de este tipo de arquitectura.

El propósito de las Honeynets de Generación I era capturar la cantidad máxima de actividad de los atacantes, pero esta arquitectura no es eficaz a la hora de actuar, puede ser fácilmente detectado por los atacantes avanzados.

El control de los datos es realizado mediante la utilización de un firewall de capa tres entre los Honeypots y la Internet. El firewall funciona como un gateway en modo NAT (Network Address Traslation) y controla todo el tráfico de entrada y de salida. La captura de datos se realiza mediante capas como se mencionó anteriormente y son transferidos a un equipo remotamente. [7]

1.2.5.2 Generación II.

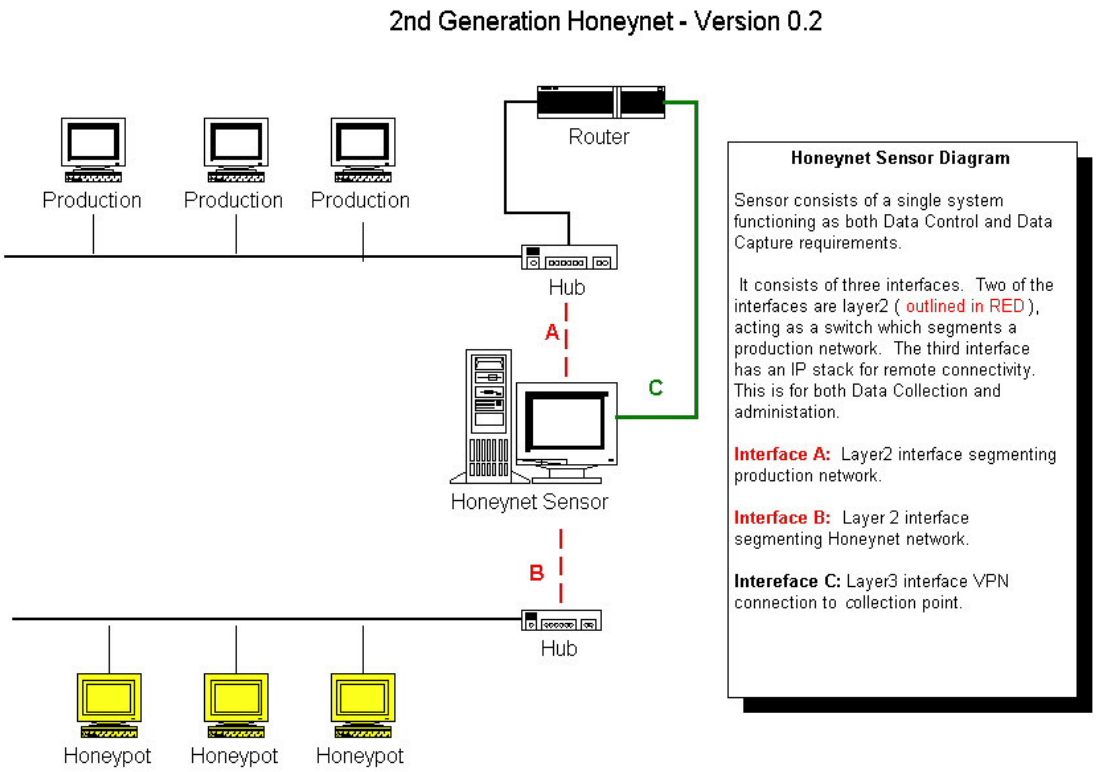


Figura 2. Arquitectura Honeynet – Generación II [16]

En esta generación el control y la captura de datos son efectivos, pero aún pueden ser mejorados. La idea de las Honeynets GenII es crear una solución que sea más fácil de desarrollar y más difícil de detectar. El Control de Datos en GenII ofrece al agresor mayores posibilidades para interactuar con los sistemas comprometidos, teniendo mayor control sobre sus actividades y haciendo más difícil que este control sea detectado. Se espera que al darle al agresor más flexibilidad en sus acciones, especialmente en conexiones salientes, se pueda recoger mayor información de ellas. Esto se consigue creando una respuesta más inteligente y flexible a las acciones del blackhat. En la Figura 2 se presenta el proceso que se realiza en la arquitectura de Honeynet GenII.

1.2.5.3 Honeynets virtuales.

Una Honeynet Virtual se basa en el mismo concepto de la Honeynet pero implementándose dentro de un mismo computador, todos sus dispositivos son virtualizados mediante un software que permita esta tecnología [32].

Dentro de una máquina física se levantan los Honeypots como máquinas virtuales formando la Honeynet Virtual. Dependiendo de la configuración de cada uno, y de la arquitectura de red, podríamos hablar de Honeynets virtuales de I, II, generación. La idea

de virtualizar el sistema es reducir costos por requerimiento de dispositivos, entre más grande es la Honeynet, más dispositivos y espacio físico se necesita. En una Honeynet Virtual todo se encuentra en una sola máquina física. En el caso de aumentar más dispositivos virtuales simplemente se mejora el hardware de la máquina anfitriona.

Entre las limitaciones tenemos: el hardware necesario de la máquina que alberga a la Honeynet, el software que es usado para virtualizar, si el atacante toma en su poder la máquina anfitriona tendría control sobre toda la Honeynet y sería un peligro para los sistemas reales. Las Honeynet Virtuales se dividen en dos grandes tipos: Auto-Contenidas e Híbridas. [32]

Honeynet Auto-Contenidas:

Se caracteriza porque todos sus dispositivos son virtualizados dentro de la misma máquina física.

Ventajas

- ✓ Movilidad: pueden ser instaladas en un portátil y llevadas a cualquier parte.
- ✓ Plug and Play: fácilmente pueden ser conectadas dentro de una red o de otra, ya que la implantación es fácil por ser un solo dispositivo.
- ✓ Económica: ahorra dinero porque no necesita de varios equipos, y ahorra espacio al usar un dispositivo.

Desventajas

- ✓ Punto único de fallo: si falla el hardware toda la Honeynet queda sin funcionar.
- ✓ Máquina potente: es necesario un computador potente para simular una red grande con muchos dispositivos.
- ✓ Seguridad: como se comparten dispositivos físicos como discos duros y unidades, es posible que el atacante pueda acceder a otras partes del sistema. La seguridad depende del software de virtualización.
- ✓ Hardware utilizado: limita la cantidad de sistemas operativos a simular. Si tenemos un PC (arquitectura ix86) nunca podremos emular Solaris de SPARC, un AIX de RS6000 o IRIS de Silicon Graphics.

Honeynets Híbridas:

Llamadas híbridas por combinar una Honeynet Clásica con una Honeynet Virtual, se agrega un dispositivo adicional en la arquitectura. Uno sirve como Honeywall (punto de entrada, control y recolección de información de la Honeynet) y otro levanta la red virtual de Honeypots.

Ventajas

- ✓ Seguridad: eliminan el punto único de fallo y aíslan los datos y el control en otro dispositivo.
- ✓ Flexible: se tiene un dispositivo que contienen diferentes tipos de Honeypot que son máquinas virtuales, las cuales pueden ser de diferentes tipos con diferentes servicios, fáciles de copiar, borrar, duplicar, lo que facilita enormemente en la tarea de administración. Si se daña un Honeypot sólo hay que levantar un duplicado ya pre instalado.

Desventajas

- ✓ Se dificulta la movilidad: debido a que tenemos dos dispositivos.
- ✓ Costosas: se incrementa el costo por hardware y en espacio.

Para cualquier tipo de Honeypot o Honeynet Virtual hay que considerar que pueden ser usadas técnicas de fingerprinting (obtención del tipo y versión del sistema operativo mediante el envío de paquetes IP específicamente construidos) sobre los Honeypots revelándole al atacante la virtualizacion de los sistemas [33].

1.2.6 PROBLEMAS LEGALES CON LA HONEYNET.

La creación de una Honeynet trae consigo algunos problemas. En este ítem se trata sobre los retos que se deben asumir al implantar una Honeynet.

1.2.6.1 Implicaciones Legales.

La supuesta tentativa generada por el uso de Honeypots es uno de estos problemas. Se podría pensar que el hecho de colocar un equipo en la red con la finalidad de que sea comprometido, puede ser tomado como tentativa para comprometer otros sistemas, y fomentar la actividad maliciosa sobre la red, es decir se habilita el medio por el cual el intruso puede atacar a otros sistemas, sin embargo esto no es cierto, [14] ya que la información obtenida de los Honeypots, es utilizada para un fin de investigación, es por ello

que no se estaría incurriendo en el artículo 58 de la Ley de comercio Electrónico del Ecuador [17]. De acuerdo a las Reformas de Código Penal, este artículo dice:

Art. 58.- A continuación del artículo 202, *inclúyanse los siguientes artículos innumerados:*
"Art. ....- El que empleando cualquier medio electrónico, informático o afín, violentare claves o sistemas de seguridad, para acceder u obtener información protegida, contenida en sistemas de información; para vulnerar el secreto, confidencialidad y reserva, o simplemente vulnerar la seguridad, será reprimido con prisión de seis meses a un año y multa de quinientos a mil dólares de los Estados Unidos de Norteamérica.

Si la información obtenida se refiere a seguridad nacional, o a secretos comerciales o industriales, la pena será de uno a tres años de prisión y multa de mil a mil quinientos dólares de los Estados Unidos de Norteamérica.

La divulgación o la utilización fraudulenta de la información protegida, así como de los secretos comerciales o industriales, será sancionada con pena de reclusión menor ordinaria de tres a seis años y multa de dos mil a diez mil dólares de los Estados Unidos de Norteamérica.

Si la divulgación o la utilización fraudulenta se realiza por parte de la persona o personas encargadas de la custodia o utilización legítima de la información, éstas serán sancionadas con pena de reclusión menor de seis a nueve años y multa de dos mil a diez mil dólares de los Estados Unidos de Norteamérica.
Art. - Obtención y utilización no autorizada de información.- La persona o personas que obtuvieren información sobre datos personales para después cederla, publicarla, utilizarla o transferirla a cualquier título, sin la autorización de su titular o titulares, serán sancionadas con pena de prisión de dos meses a dos años y multa de mil a dos mil dólares de los Estados Unidos de Norteamérica"

1.2.6.2 Permisos.

Uno de los requisitos necesarios para la implantación de una Honeynet en una organización, es el permiso de la misma para el proyecto. En este caso se debe contar con un permiso explícito de la Universidad, para monitorear, capturar, y analizar el tráfico que llega a la Honeynet, debido a que esto puede incluir información ajena a la detección de amenazas en la red (conversaciones por mensajero instantáneo, transferencia de archivos, etc.). Para el presente proyecto debemos tratar los temas de legalidad, privacidad y seguridad tomando en cuenta que no somos propietarios de la red donde se desea implantar la Honeynet. Este permiso da sustento al proyecto y el reconocimiento al mismo como un recurso de investigación.

1.2.7 FUNCIONAMIENTO DE LA HONEYNET.

El funcionamiento de la Honeynet puede explicarse de forma simple "*…ud crea una red parecida a una pecera, donde puede ver todo lo que ocurre dentro de ella. Igual que a un pez, puede observar a un hackers interactuar con su entorno virtual. Además, al igual que en una pecera, puede poner allí todo lo que quiera. Esta red controlada se convierte en su Honeynet. Las actividades capturadas le enseñan las herramientas, tácticas y motivos de la comunidad blackhat*" [13]. Con ésta tecnología se simulan sistemas reales en producción, por lo que cualquier tráfico que se genera desde y hacia la Honeynet es sospechoso y podría conducir a un posible ataque, por tanto cualquier interacción con este sistema deberá ser registrado para su posterior análisis.

1.2.8 TÉCNICAS DE ENGAÑO.

La inclusión de técnicas de engaño, evita que los atacantes se den cuenta de cómo y de qué forma exacta pueden hacernos daño. A continuación se exponen algunas técnicas y su funcionamiento.

1.2.8.1 Symantec Decory Server (Man Trap). [14]

Este producto presenta una alternativa innovadora a los planteamientos convencionales de software basados en tecnología Honeynet. Permite instalar, sobre un único sistema operativo, hasta cuatro sistemas operativos que se presentan al usuario como máquinas independientes en la red.

Cada uno de los sistemas operativos que se instalan sobre el sistema operativo base son completamente funcionales en el sentido de que, una vez comprometidos, el atacante puede acceder al sistema de ficheros e interactuar con los procesos que corren en él. No obstante, ninguna de las acciones que lleva a cabo en el equipo virtual comprometido afectan al sistema operativo base. El propio producto se encarga de establecer una sólida política de privilegios que impide el compromiso total o parcial del equipo en el que se ejecuta; de hecho, este equipo es invisible para el atacante.

Con objeto de proporcionar un mayor grado de realismo a la virtualización, cada uno de estos sistemas operativos utiliza una interfaz de red diferente a la del equipo en el que se ha instalado el producto.

1.2.8.2 Decepción Toolkit (DTK). [18]

DTK fue el primer Honeypot OpenSource, publicado en 1997. Escrito por Fred Cohen, DTK es una colección de scripts de Perl y C de código fuente que emula una gran variedad de servicios de escucha. Su principal objetivo es engañar a los atacantes humanos.

DTK utiliza una estrategia sencilla basada en la simulación de servicios de red comunes, en apariencia vulnerables. El objetivo que se persigue es mantener ocupado al atacante con un sistema que no representa ningún riesgo desde el punto de vista de los servicios de negocio ofrecidos; para ello, se le proporciona un método fácilmente identificable de comprometer el sistema, con esto se gana que el intruso este más tiempo intentando vulnerar algún sistema logrando rastrear y captutar sus acciones de ataque.

1.2.8.3 LaBrea Tarpit. [19]

Es OpenSource disponible para los principales sistemas operativos (Windows, UNIX, Solaris, BSD). LaBrea no realiza ningún de tipo de simulación su objetivo no es engañar a un atacante humano, sino detener al robot de red utilizado por los worms para detectar nuevos objetivos.

Esta utilidad se adueña de direcciones IP no utilizadas en la red e instala en ellas servidores virtuales. Estos servidores responden a los intentos de conexión en los puertos más habituales para la propagación de gusanos (por ejemplo, puerto 80 en el caso del gusano CodeRed). Sin embargo, en lugar de presentar al cliente el comportamiento esperado del servicio al que accede, la utilidad intenta mantener la conexión abierta (idle) el mayor tiempo posible. De esta forma se consigue ralentizar considerablemente el proceso de búsqueda de nuevos objetivos y, por tanto, el de infección de nuevas máquinas.

1.2.8.4 Honeyd. [10]

Honeyd es una herramienta potente, es un software de virtualización que permite diseñar una Honeynet relativamente compleja con un grado de realismo muy elevado sin necesidad de utilizar costosos recursos de hardware.

Honeyd puede instalarse en los principales sistemas operativos basados en UNIX (Linux, BSD, Solaris).

Permite emular servicios de red, sistemas operativos a nivel de stack TCP/IP e incluso redes locales completas, con equipos que funcionan como servidores, estaciones de trabajo o elementos de interconexión (routers).

Con Honeyd es posible construir un entorno de red virtual con una arquitectura arbitraria. La configuración recomendada de Honeyd impide el acceso a la máquina en la que se instala: sólo son visibles los hosts virtuales definidos en los ficheros de configuración de la aplicación. A continuación se incluye una breve lista de los sistemas operativos que es capaz de emular:

- ✓ Linux
- ✓ Windows (95/98/NT/2000/Me/XP)
- ✓ Cisco IOS
- ✓ Mac OS
- ✓ Sega Dreamcast

Para poder llevar a cabo la emulación de equipos, Honeyd se adueña de direcciones IP no utilizadas en la red mediante un procedimiento denominado IP takeover; de esta forma consigue simular la presencia de un conjunto de máquinas en un determinado rango de direcciones IP. Estas máquinas virtuales reciben el tráfico dirigido a su dirección IP y responden en función de la configuración definida. El comportamiento de cada una de ellas se establece en base al sistema operativo emulado y los servicios de red activos, con objeto de ofrecer el mayor grado de realismo posible.

El problema de este software es la interfaz de usuario. Toda la configuración se realiza a través de ficheros con una sintáxis predefinida o mediante parámetros del ejecutable en línea de comandos. Tampoco dispone de una herramienta de gestión específica que facilite la labor del administrador. No obstante, en la última sección se describe cómo puede implementarse dicha herramienta con una interfaz web basada en Perl/Javascript.

1.3 DISEÑO DE UNA HONEYNET

1.3.1 INTRODUCCIÓN.

El diseño de una Honeynet es uno de los pasos primordiales al momento de realizar una implementación de este tipo, ya que esto es la base fundamental sobre la cual funcionará la red trampa. Aquí se analizan aspectos muy importantes como: ubicación de la Honeynet en la red de la organización, equipos a utilizar y herramientas de software a implementar.

Desde los inicios de la Honeynet, se han venido desarrollando diferentes arquitecturas, generaciones, y herramientas para la explotación de los datos, los cuales se obtienen de acuerdo a los requerimientos que surgen en base a las diferentes necesidades que tienen

las organizaciones que se encuentran involucradas en la investigación de la seguridad en las redes.

1.3.2 UBICACIÓN DE LOS HONEYPOTS EN LA RED.

Los Honeypots pueden ser ubicados en distintas partes sobre la red, esto dependerá de los requerimientos que tenga la organización. Según [15] tenemos las siguientes ubicaciones sobre la red.

1.3.2.1 Honeypots antes del firewall.[7]

Esta ubicación es la menos riesgosa para la red, ya que se encuentra fuera de la zona protegida por el firewall, puede ser atacado sin ningún tipo de peligro para el resto de la red. La Figura 3 indica el esquema de este tipo de ubicación de un Honeypot.

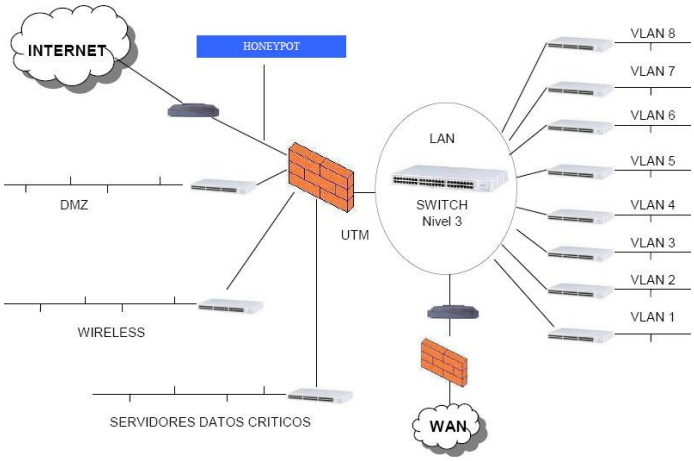


Figura 3. Ubicación de un Honeypot antes del firewall [15]

Esta ubicación permite:

- ✓ Tener un acceso directo a los atacantes, y obtener información real de su comportamiento y estadísticas fiables sobre la cantidad y calidad de ataques que puede recibir nuestra red.
- ✓ Evitar las alarmas de otros sistemas de seguridad de nuestra red (IDS) al recibir ataques en el Honeypot. Existe el peligro de generar mucho tráfico debido a la facilidad que ofrece el Honeypot para ser atacado.
- ✓ Evitar la detección de atacantes internos.

[7] **Firewall** (**cortafuegos**), es un elemento de hardware o software utilizado en una red de computadoras para controlar las comunicaciones, permitiéndolas o prohibiéndolas según las políticas de red.

Al trabajar bajo esta ubicación, cualquier atacante externo lo primero que encuentra será al Honeypot, sin embargo existe el peligro de que se genere mucho tráfico debido precisamente, a la facilidad que este ofrece para ser atacado, también generará consumo de ancho de banda y espacio en los ficheros de log. [6]

1.3.2.2 Honeypots después del firewall.

En esta ubicación el acceso al Honeypot está dirigido por las reglas de filtrado del firewall, su ubicación permite la detección de atacantes internos así como firewalls mal configurados, máquinas infectadas por gusanos o virus y atacantes externos. El esquema de esta ubicación se puede observar en la Figura 4.

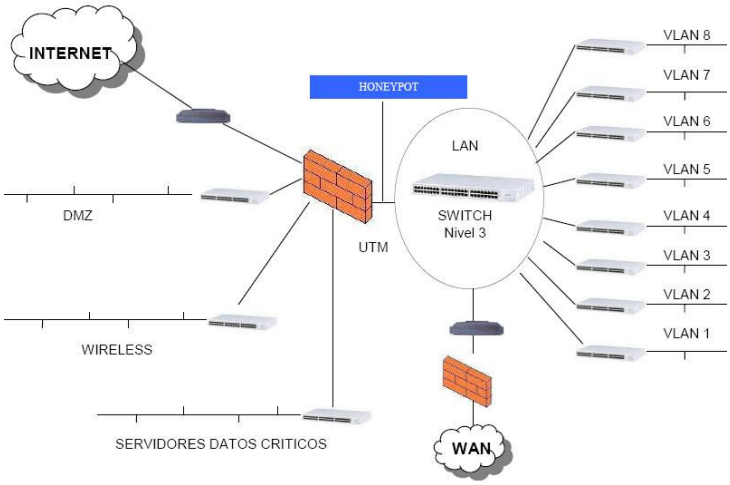


Figura 4. Ubicación de un Honeypot después del firewall [15]

Al utilizar esta ubicación se debe modificar las reglas del firewall para permitir algún tipo de acceso a nuestro Honeypot por posibles atacantes externos, al realizar esto se podría estar introduciendo algún elemento peligroso dentro de nuestra red permitiendo así a un atacante ganar acceso al Honeypot y a nuestra red.

Con este esquema nos vemos en la necesidad de asegurar el resto de nuestra red contra el Honeypot mediante el uso de firewalls extras o sistemas de bloqueo de acceso, ya que si un atacante logra comprometer el sistema tendrá vía libre en su ataque a toda nuestra red.

Un beneficio de esta ubicación es la generación de gran cantidad de alertas por otros sistemas de seguridad de la red (Firewalls, IDS, etc.) al recibir ataques el Honeypot. [6]

1.3.2.3 Honeypots en la zona desmilitarizada.[8]

Esta es la ubicación ideal ya que permite detectar ataques externos e internos, para esto se requiere de una reconfiguración del firewall puesto que se encuentra en la zona de acceso público. La Figura 5 representa la estructura que utiliza un Honeypot ubicado en la zona desmilitarizada.

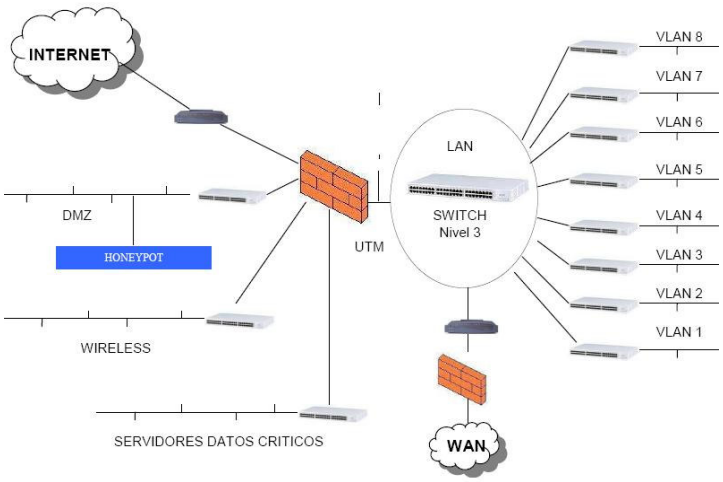


Figura 5. Ubicación de un Honeypot en la zona desmilitarizada [15]

La ubicación en esta zona permite juntar en el mismo segmento a nuestros servidores de producción con el Honeypot y controlar el peligro que carga su uso, ya que tiene un firewall que lo aísla del resto de nuestra red local. Bajo este esquema se eliminan las alarmas de los sistemas internos de seguridad. [7]

1.3.3 ESTUDIO DE LA RED ACTUAL DE LA UTPL.

Para llegar a establecer los requerimientos necesarios para implementar la Honeynet se realizó un estudio de la red actual de la UTPL (en el punto 1.3.4), en la que se analizó el esquema, equipos y configuraciones de la red.

El tener un panorama más amplio y claro de la red de la UTPL, nos ayudará en gran medida para poder determinar la ubicación de la Honeynet dentro de la misma.

El primer paso luego de identificar que el objetivo de implementación de la Honeynet es monitorear el origen de ataques de la red externa o Internet, se evalúo el esquema de red de la Universidad así como sus configuraciones. En su configuración inicial se asigna a la

[8] **DMZ** (demilitarized zone) **zona desmilitarizada** o **red perimetral** es una red local que se ubica entre la red interna de una organización y una red externa, generalmente Internet.

Honeynet una subred dedicada de 8 hosts y su ubicación en la red de la universidad se muestra en la Figura 6.

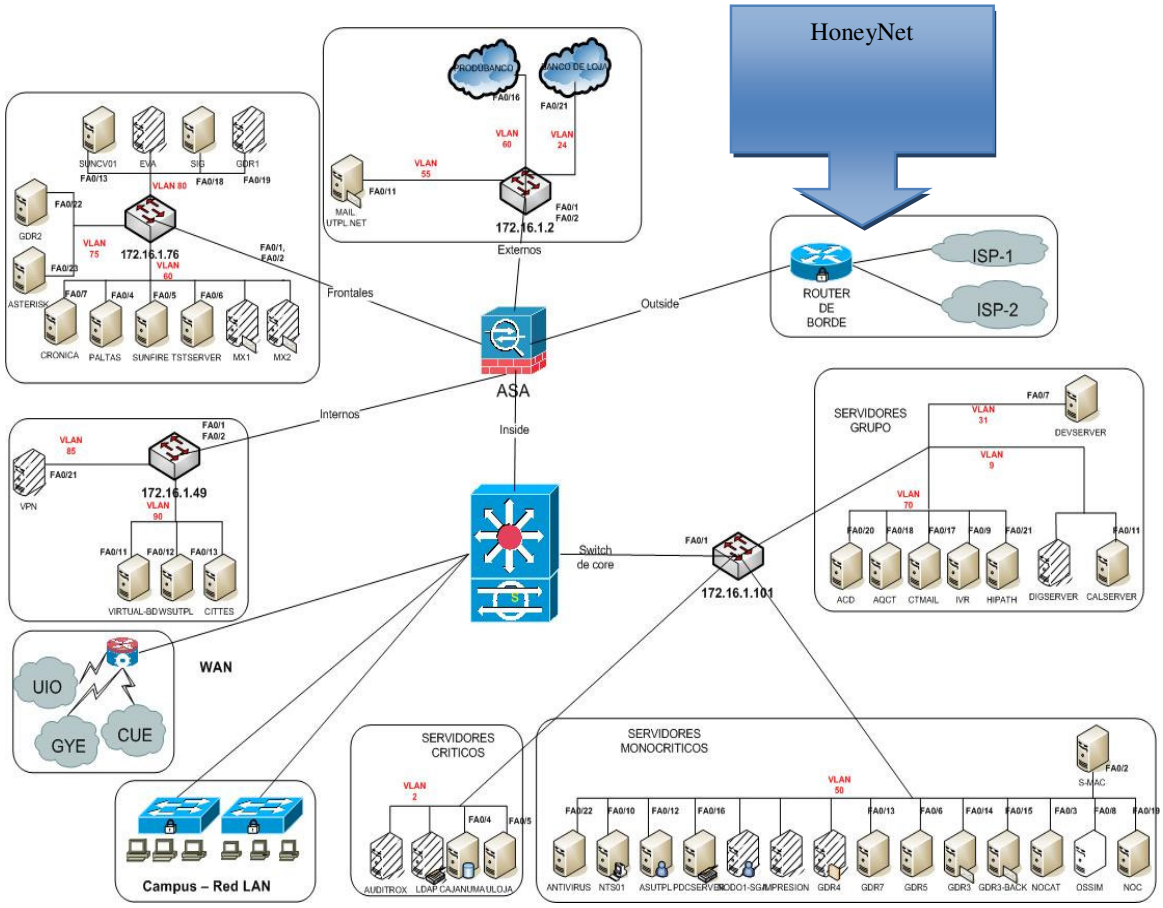


Figura 6. Esquema de red de la UTPL

1.3.4 ANÁLISIS DE LA UBICACIÓN Y SELECCIÓN DE ARQUITECTURA PARA LA IMPLEMENTACION DE LA HONEYNET.

Para la elegir la Arquitectura de la red trampa de la UTPL, se realiza el siguiente análisis.

Como se había mencionado en la documentación de las arquitecturas de las Honeynets (punto 1.2.5), existen de Generación I, Generación II y Virtuales, para la presente se ha decidido implementar en la red de la Universidad una Honeynet de II Generación por ser la que más se apega a los requerimientos y objetivos esperados (referidos el punto 1.2.4) y por ser la generación más segura, estable y se encuentra en vigencia como herramienta de análisis forense.

Las generaciones I y II tienen características comunes teniendo la II técnicas y herramientas mejoradas para el control y captura de datos. Para este el análisis se ha

tomado como una sola Generación que tienen como característica común que se implementan en equipos reales, entonces tenemos dos tipos para elegir:

1. Implementar una Honeynet con los mismos requerimientos de hadware que una red en producción, en la cual todos sus equipos y servicios son físicos y reales respectivamente, nada es virtualizado.
2. Implementar una Honeynet Virtual, virtualizando sistemas operativos y servicios en uno o varios equipos.

Para montar este proyecto se analiza los requerimientos para la implementación de una Honeynet (estudiados en el punto 1.2.4), además se cuenta con los equipos y la infraestructura necesara para implantar la Honeynet.

Para la elección de la arquitectura se ha tomado en cuenta las ventajas y desventajas de usar soluciones ya sean en ambientes virtualizados como en reales (estudiandos en el ítem de arquitecturas de una Honeynet 1.2.5) y tomando en cuenta que las ventajas de utilizar Honeynet con equipos y servicios reales frente a las Honeynets Virtuales dan mayor calidad a la solución esperada, se ha decidido usar Honeynets de Generacion II.

Para la elaboración del diseño de la Honeynet, la red ha sido analizada en su estructura y servicios. En base al análisis realizado, hemos decidido agregar nuestra Honeynet fuera del espacio de direcciones de producción debido a que contiene servicios sensibles y muy susceptibles a ataques. Entre estos servicios se encuentran los diferentes servidores Web, de base de datos, de correo electrónico, proxy, DNS, etc. y los diferentes computadores ubicados dentro del área Administrativa, que son la base de los servicios digitales prestados por la UTPL.

Como se puede apreciar en la Figura 6, la red en producción, donde se encuentran los servidores reales se aísla de la Honeynet eliminando la posibilidad de tráfico entre ellas y los riesgos de seguridad asociados con el comprometimiento de la Honeynet.

Luego del análisis realizado se llegó a determinar que la mejor ubicación donde será implementada la Honeynet es fuera de la red de producción de la UTPL, es la más indicada, la que menos riesgo causa a la integridad de la red de la universidad y el esquema seria el planteado en la Figura 7.

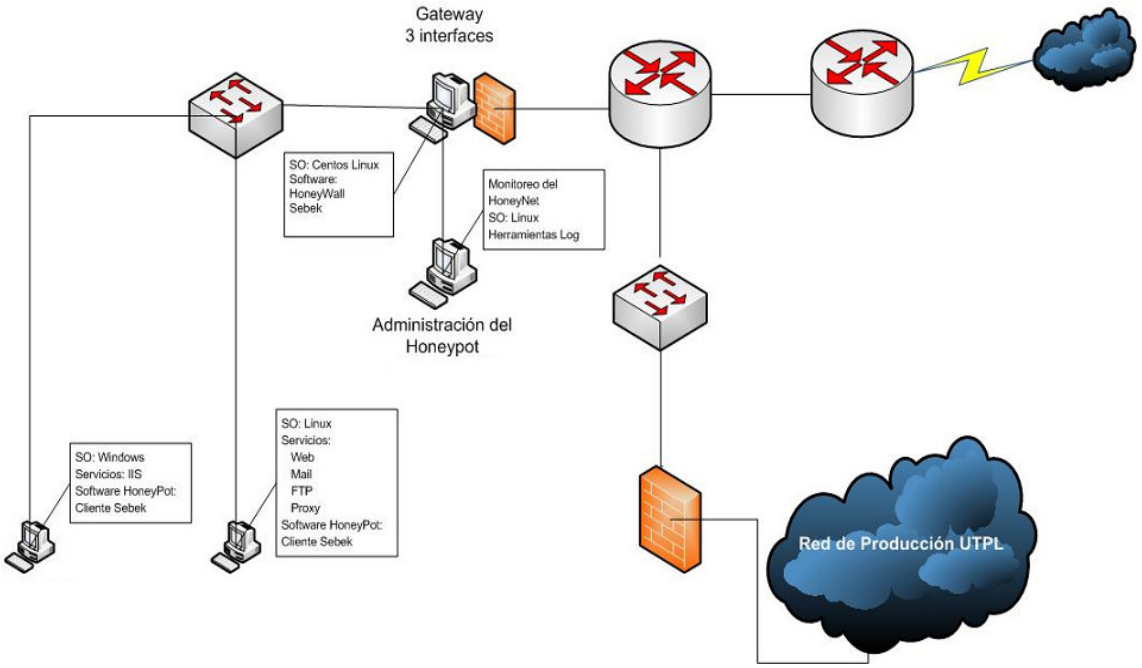


Figura 7. Esquema de la red Honeynet para la UTPL

CAPÍTULO II:

ESTUDIO DEL SOFTWARE PARA LA HONEYNET ACTUAL.

2.1 INTRODUCCIÓN.

En este capítulo se describen las todas herramientas necesarias y que se utilizan para el control, captura y análisis de tráfico en la Honeynet. El estudio de estas herramientas es fundamental para la obtención de datos principalmente, ya que gracias a estos resultados se puede realizar el análisis para determinar qué es lo que hacen los intrusos al acceder a la red.

2.2 HERRAMIENTAS HONEYWALL.

El Honeywall CDROM, es un CD de arranque que contiene todas las herramientas para crear y utilizar una Honeynet. Está basado en una versión reducida de Linux y está diseñado para ser utilizado como aplicación.

Contiene herramientas para gestionar el Honeywall, monitorizar la red y registrar las actividades del atacante, las cuales incluyen:

2.2.1 FIREWALL IPtables. [20]

Iptables es un firewall, que proporciona la funcionalidad de filtrado de paquetes, traducciones de direcciones de red (NAT). Iptables se configuró para permitir transmitir paquetes entre las dos interfaces de red del Honeywall. La primera interfaz que conecta a la red exterior, y la segunda interfaz que sirve de Gateway para los Honeypots. Las reglas son configuradas para evitar IP Spoofing[9] desde la red interna. Estas reglas garantizan que solo los paquetes que tienen origen las direcciones de red interna se pueden ir fuera. Se restringe el tráfico proveniente de internet hacia el Gateway.

Otra cosa importante que se logra con las normas del Iptables es:

- ✓ Reducir la posibilidad de ataques de denegación de servicios (DoS),
- ✓ la regulación de los Honeypots,
- ✓ y la cantidad de tráfico que puede salir desde la Honeynet.

El firewall, es configurado para establecer límites sobre la cantidad de datos que pueden ser enviados desde cada sistema trampa por unidad de tiempo. Los límites utilizan las conexiones salientes en el caso de TCP y el número de paquetes para

[9] **Ip Spoofing.-** Consiste en sustituir la dirección IP origen de un paquete TCP/IP por otra dirección a la cual se desea suplantar.

UDP, ICMP, y otros (una categoría que detecta si los atacantes utilizan un protocolo diferente, como IPv6 dentro de IPv4, para enviar datos desde la red trampa). El tiempo para cada límite puede indicarse en unidades de segundos, minutos, horas, y días. Los paquetes de salida autorizados para pasar a través del firewall son luego enviados a Snort_Inline[10]. Es capaz de tomar acciones contra el tráfico saliente que tenga ataques conocidos.

El objetivo aquí es contar las conexiones de salida, y cuando un cierto límite se ha cumplido, se bloquea cualquier intento de conexión posterior. Esto se utiliza principalmente para reducir el riesgo de la masa de exploración, ataque, o ataques de denegación de servicio, actividad que requiere gran número de conexiones de salida. El número de conexiones a permitir dependerá de cuánto riesgo se está dispuesto a asumir.

Limitar el número de conexiones salientes evita que los atacantes utilicen la red trampa para escanear o atacar un gran número de otros sistemas, o para lanzar ataques de denegación de servicio. Pero hay que tener en cuenta que esto puede delatar el funcionamiento de la Honeynet, un atacante puede ser capaz de detectar su red trampa, simplemente iniciar conexiones de salida y ver si están bloqueados después de un determinado número.

Seguidamente se muestra el número de conexiones de salida permitidas a los protocolos por cada hora:

Escala = hora

- ✓ Limite de conexiones permitidas para TCP=20
- ✓ Limite de conexiones permitidas para UDP=20
- ✓ Limite de conexiones permitidas para ICMP=50
- ✓ Limite de conexiones permitidas para OTROS=10

El script que configura estas reglas se lista en el **ANEXO A**.

2.2.2 Proxy ARP. [21]

Con el fin de que los Honeypots puedan ser accesibles desde Internet, el Gateway fue configurado para responder peticiones ARP de las IPs de los Honeypots. ARP se

[10] **Snort_inline.-** Es un NIPS, es una versión modificada de Snort, utiliza nuevos tipos de reglas para decirle a iptables si el paquete debe ser descartado o modificado sobre la base de un conjunto de reglas de snort.

utiliza para traducir las direcciones IP en direcciones MAC. Una técnica llamada Proxy ARP es que el Gateway se puede configurar para responder a las peticiones de ARP de direcciones IP de los Honeypots que se ha elegido. Esto permite a las computadoras desde el exterior, puedan encontrar y hacer las conexiones con los Honeypots. (NAT) Network Address Translation también se puede utilizar para lograr este objetivo.

2.2.3 Snort. [22]

El sistema de detección de intrusos que se utiliza es Snort[11]. Este olfatea cualquier tráfico que entra y sale de la segunda interfaz de red de la puerta de enlace (esta interfaz proporciona la puerta de entrada a Internet para los Honeypots), además los registra en varios formatos.

Snort envia todo a una base de datos MySQL en el Honeywall. Snort tiene muchas reglas predefinidas para detectar diversos intentos de intrusión. Estas reglas personalizadas clasifican las conexiones basadas en números de puerto, la dirección IP de los Honeypots y otras firmas. La clasificación ayuda en la recopilación de estadísticas. En el **ANEXO B** se puede encontrar el archivo de configuración, algunas reglas y alertas de Snort.

2.2.4 Snort_Inline. [23]

Snort_Inline es un sistema de prevención de intrusiones basado en el detector Snort. Es capaz de tomar acciones contra el tráfico saliente que tenga ataques conocidos. Snort_Inline puede ser configurado en el Honeywall para utilizar tres conjuntos diferentes de reglas por defecto que pueden: *descartar (drop), deshabilitar (disable), o rechazar (reject)* ataques conocidos.

El script del firewall y las reglas de Snort_Inline están en el directorio */etc*. Pueden ser modificados para ajustarse a las necesidades particulares de cada red trampa. Ambos pueden generar registros detallados que pueden ser utilizados para análisis y alertas. Toda la actividad de Snort y Snort_Inline es registrada en /var/log/snort/$DAY, donde $DAY es el valor numérico del día, por ejemplo se ha estandarizado YEARMONTHDAY, de forma que los datos capturados el 30 de agosto de 2009 deberían estar en /var/log/snort/20090830.

---

[11] **Snort** es un sistema de detección de intrusos basados en red, capaz de realizar análisis de tráfico en tiempo real y registro de paquetes en redes IP

2.2.5 POF (Pasive OS Fingerprint). [24]

Analiza el tráfico de red e intenta identificar el sistema operativo en base a parámetros TCP/IP.

2.2.6 Swatch (Simple Watcher of Logfiles). [25]

Investiga los archivos log del Honeywall en busca de eventos definidos mediante expresiones regulares y envía un correo al administrador si encuentra alguna actividad de red sospechosa.

2.2.7 Sebek. [26]

Sebek es una herramienta de captura de datos que ayuda a recrear de forma fiable los eventos sucedidos dentro de un equipo trampa. Es el módulo central, reside en las profundidades del sistema operativo y registra todas las actividades del atacante. Es una herramienta cliente-servidor, que en el lado del cliente se puede ejecutar en plataformas Linux y Windows; la instalación de esta herramienta puede ser revisada con más detalle en el **ANEXO C**.

El principal objetivo de esta herramienta es determinar información como cuándo consiguió entrar el intruso, cómo lo hizo y qué hizo después de conseguir acceso. Con esta información se puede determinar quién es el intruso, cuáles son sus motivos y con qué trabaja. Para determinar qué hizo el intruso después de su acceso es importante tener los datos que proporcionan las pulsaciones de teclado del mismo y el impacto de su ataque.

Entre sus principales propósitos podríamos mencionar los siguientes:

- ✓ Registrar todo o parte de los datos accedidos por los usuarios del sistema,
- ✓ Registrar pulsaciones de teclas en una sesión cifrada,
- ✓ Recuperar archivos copiados con SCP[12],
- ✓ Capturar contraseñas utilizadas para entrar en sistemas remotos,
- ✓ Recuperar contraseñas usadas para habilitar binarios protegidos con Burneye,
- ✓ Cumplir con otras tareas relacionadas con el análisis forense.

[12] **SCP Secure Copy** es un comando de Linux que permite copiar ficheros entre dos máquinas, ofrece la misma seguridad que el **SSH** en la transmisión encriptada de datos.

Las primeras versiones de Sebek fueron desarrolladas para recolectar las pulsaciones de teclas directamente desde el núcleo, para ello utilizaban una llamada a **sys_read** troyanizada. Este sistema registraba las pulsaciones en un fichero oculto y las exportaba a través de la red de forma que pareciera un tráfico UDP cualquiera, como NetBIOS. Este proceso era complejo, fácil de detectar a través del uso de rastreadores por ejemplo sniffers y tenía un rendimiento limitado; por esta razón se hacía más difícil recolectar otro tipo de datos que no fueran pulsaciones de teclas.

La siguiente y actual versión de Sebek, versión 2, fue diseñada no sólo para recolectar pulsaciones de teclas, sino para todos los datos que pasaran por **sys_read**. Recolectando todos los datos, se ha extendido la capacidad de monitorización a toda actividad dentro del equipo trampa incluyendo las pulsaciones de teclas. Si se copia un fichero al equipo trampa, Sebek verá y registrará el fichero obteniendo una copia idéntica. Si un intruso lanza un cliente de IRC o de correo, Sebek verá sus mensajes. Un objetivo secundario es el hacer a Sebek más difícil de detectar, en ocultar el tráfico de los registros para ocultarlos completamente de un blackhat, así cuando uno de ellos ejecute un rastreador para detectar tráfico sospechoso sea incapaz de detectar el tráfico perteneciente a Sebek.

Sebek también proporciona la habilidad de monitorizar los trabajos internos del equipo trampa de forma transparente en comparación con las anteriores técnicas de caja negra. Si un intruso instala un software maligno y sale del sistema, ahora se puede seguir el rastro de las acciones locales del software maligno aunque no acceda a la red.

Arquitectura:

Tiene dos componentes: el cliente y el servidor.

1. El cliente captura los datos del equipo trampa y los envía a través de la red donde son recolectados por el servidor. La Figura 8 representa esta arquitectura.
2. El servidor recolecta los datos de dos posibles fuentes: la primera es capturando paquetes directamente de la red, y la segunda es a través de un archivo de captura de tráfico en formato *tcpdump*.
3. Una vez que se han recolectado los datos se almacenan en una base de datos relacional o se extraen las pulsaciones de teclas.

Las comunicaciones usadas por Sebek están basadas en UDP por ser no orientadas a conexión y no fiables.

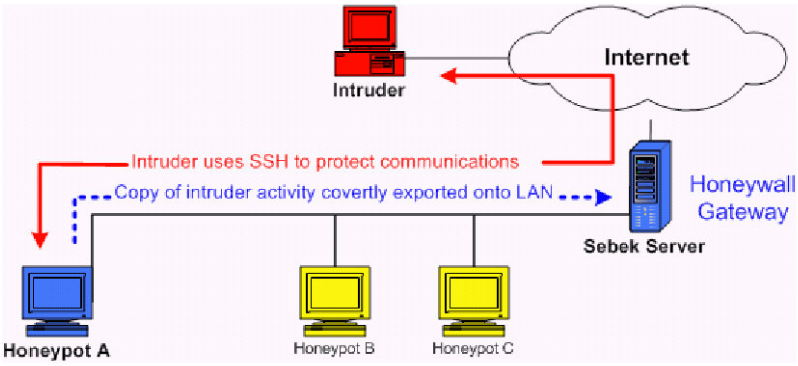


Figura 8. Arquitectura de Sebek [27]

2.2.8 Hflow. [28]

Hflow recibe datos de flujo Argus, IDS Snort, pOf OS fingerprints y datos Sebek. Una vez combinados estos datos son insertados en una base de datos.

Hflow fue desarrollado para combinar cada una de las fuentes de datos mencionadas, en un solo modelo relacional compuesto. Continuamente consume datos desde cada fuente de datos, la fusión se basa en la identificación de las relaciones y después se almacenan todos estos datos en una base de datos llamada Hflow.

Los flujos de datos relacionados, como Argus y Snort, son relacionados en base al número de protocolo IP, la dirección de origen, destino y el número de puerto de aplicación, que entran dentro del mismo periodo de tiempo. Este es un enfoque similar a la forma en que un sistema operativo determina que paquete se relaciona con que socket.

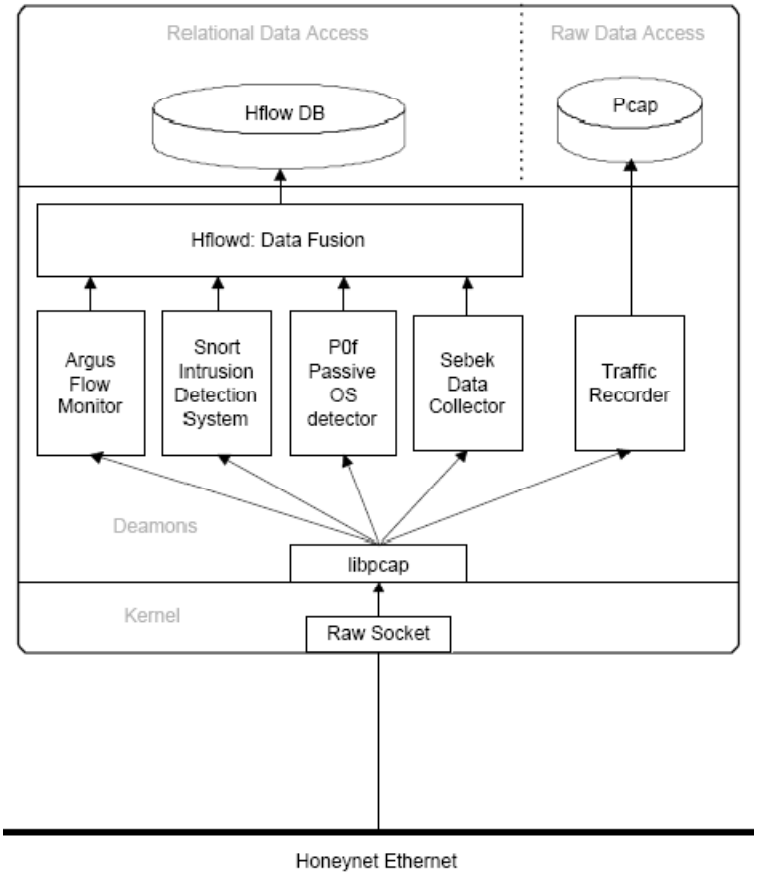


Figura 9. Colección de datos y diagrama de fusión [28]

El resultado de la fusión de datos de Hflow es un conjunto de datos unificados con relaciones identificables entre las categorías lógicas. Estos datos son exportados en el formato necesario para ser cargados en una base de datos relacional.

2.3 HERRAMIENTAS DE ANÁLISIS DE TRÁFICO.

El análisis se lo realiza de dos maneras:

1. Monitoreo y análisis de tráfico remotamente en línea, para esto se utilizó la interfaz web Walleye que ofrece el Honeywall.
2. En la segunda forma, se capturan paquetes denominados PCAP del Honeywall y porteriormente se realiza el análisis de estos con herramientas selelccionadas que permiten la lectura de este tipo de formato.

Se ha elegido Wireshark y Honeysnap debido a que son muy potentes y los más utilizados, ofrecen un resumen completo y detallado del contenido de los PCAPS.

2.3.1 Walleye. [29]

Es una interfaz de usuario basada en navegador, actúa como interfaz grafica entre la información guardada en el Honeywall y su operador. Además de mostrar información del tráfico de red, Walleye permite solicitar información sobre Honeypots individuales.

Walleye representa la información de Sebek como gráficos de procesos, ofreciendo de esta manera al administrador una visión general muy útil de las actividades del atacante.

Walleye también exporta tráfico de red en formato *pcap*: se importa esta información a otras herramientas de análisis como el Wireshark.

Por último está interfaz permite modificar algunas configuraciones del Honeywall.

2.3.2 Wireshark. [30]

Es un analizador de paquetes de red que examina lo que está pasando en el interior de una red. Esta herramienta es considerada como una de los mejores analizadores de paquetes de código abierto.

Permite realizar análisis y solucionar problemas en redes de comunicaciones. La funcionalidad que provee es similar a la de *tcpdump*, pero añade una interfaz gráfica y opciones de organización y filtrado de información. Gracias a su funcionalidad permite ver todo el tráfico que pasa a través de una red; también incluye una versión basada en texto llamada *tshark*.

Wireshark se ejecuta sobre la mayoría de sistemas operativos Unix y compatibles, como: Linux, Solaris, FreeBSD, NetBSD, OpenBSD, y Mac OS X, así como en Microsoft Windows.

2.3.3 Honeysnap. [31]

La herramienta Honeysnap es una herramienta que interpreta los archivos en formato *pcap* en una consola de línea de comandos, el objetivo principal es presentar un análisis resumido de los datos, de esta manera, una vez identificados los datos de interés se pueden utilizar otras herramientas para un análisis más detallado, con esta

herramienta se pueden obtener resúmenes de datos HTTP, TCP, FTP, Sebek, ICMP, IRC, entre otros.

CAPÍTULO III:

CONFIGURACIÓN E IMPLEMENTACIÓN DE UNA HONEYNET.

3.1 INTRODUCCIÓN.

Para la implementación es necesario conocer las características físicas de los equipos que van a formar parte de la Honeynet, en este caso se utilizará un equipo que servirá como Servidor y dos equipos que funcionaran como Honeypots, los mismos que enviarán la información de los atacantes al Servidor. Uno de los Honeypots utilizará plataforma Windows y el otro Linux. Este capítulo incluye también cuales son las seguridades y servicios que tendrán los equipos incluyendo mecanismos de mantenimiento y respaldo de los estados de cada uno.

3.2 CARACTERÍSTICAS DE HARDWARE Y SOFTWARE.

Las características del equipo Honeywall son:

- ✓ Procesador Pentium 4 de 3.0 MHZ
- ✓ 4 GB de memoria RAM
- ✓ 80 GB de capacidad de almacenamiento en disco

Las características de los Honeypots son:

- ✓ Procesador Pentium 4 de 3.0 MHZ
- ✓ 1 GB de memoria RAM
- ✓ 80 GB de capacidad de almacenamiento en disco

Software a instalar:

Para la implementación se considera un equipo Honeywall, en el cual se instaló la herramienta **Honeywall versión 1.4.hw-2009**, que para la captura de datos trae instalado Sebek Server.

En cada uno de los Honeypots se instaló el siguiente software:

- ✓ Equipo Windows

  Sebek cliente – Win32-3.0.3 (captura de datos)

- ✓ Equipo Linux

  Sebek cliente – Linux-2.1.7 (captura de datos)

3.3 IMPLEMENTACIÓN.

El diseño preliminar, tal como lo muestra la Figura 7 (Capítulo 4), estará conformado por un equipo Honeywall (Gateway) que hará de pasarela de capa 2, que actuará como puente y este es totalmente transparente al atacante que no tiene conocimiento de qué tráfico está pasando por esta máquina. La interfaz externa de nuestra pasarela Gateway (eth0) está conectada a la red externa (Internet). La interfaz interna de nuestro Gateway (eth1) está conectada a la red de sistemas Honeynet (Honeypots) que levantarán servicios similares a los que están en producción, con el fin de ser usados como señuelos para ser detectados por intrusos. La interfaz eth2 está conectada a la red interna, desde la cual se podrá administrar la Honeynet. Como puente, tanto la interfaz externa como la interna están en la misma red IP. Esto nos ofrece diversas ventajas. La primera ventaja viene de el hecho de que el dispositivo sea de capa 2, lo que lo hará mucho más difícil de detectar, por cuanto no hay enrutamiento de tráfico ni decrementos en el tiempo de vida (TTL), además el dispositivo está más oculto, así que los atacantes no sabrán que su tráfico está siendo analizado y controlado. La segunda ventaja es que, como pasarela, todo tráfico entrante o saliente debe pasar a través del dispositivo, de esta forma podemos capturar y controlar tanto el tráfico entrante como el saliente desde un único dispositivo.

Tanto el control del flujo de datos como la captura de datos, estarán bajo herramientas libres como sniffers, alarmas y scripts para bloqueo y control de ataques. Todo esto estará corriendo sobre una distribución Linux que soporte funciones de puente, lo cual con Iptables es muy sencillo y eficiente de implementar.

Tres computadoras fueron necesarias para empezar la Honeynet, por lo que una se configuró como un equipo Gateway-Honeywall, otro como un Honeypot Linux y el otro como un Honeypot Windows. Otro equipo fuera de la red se utilizó adicionalmente para acceder y administrar remotamente.

3.4 EL GATEWAY – HONEYWALL.

Este computador es crítico en el funcionamiento de la Honeynet. Es la puerta de entrada a la red y también sirve como un servidor de seguridad, sistema de detección de intrusos y servidor de logs. Tiene una puerta de enlace para los Honeypots, ayuda mucho en el sentido de que permite filtrar el tráfico y la hace fácil de monitorear y administrar toda la actividad de la red asociada con el Honeypot. También proporciona

un sistema seguro de registro y da mejores opciones para asegurar a los Honeypots. El Honeywall actúa como un router entre la red externa y la red de Honeypots. Este equipo está basado en una distribución de CentOS. La instalación y configuración de este servidor se encuentra detallada en el **ANEXO D**, el archivo de configuración puede ser revisado en el **ANEXO E**.

3.4.1 Seguridad del Honeywall.

La seguridad del Gateway es de suma importancia ya que todas las actividades de administración y monitoreo giran en torno a ella. Todos los servicios en el Gateway son absolutamente necesarios.

- ✓ Se ha creado un Script el cual está incluido en el crontab del servidor, este envía al administrador reportes del estado del Honeywall como: Nombre del sistema, fecha, particiones montadas, tamaño de las particiones, procesos del sistema, además de los log messenges e iptables. Esto con el fin de ir monitoreando el estado del servidor e ir almacenado estos reportes para un posterior análisis de algún evento detectado. En el **ANEXO F** se encuentra el script que permite obtener la información del estado del Honeywall.
- ✓ Con el fin de administrar de forma remota el Gateway y el Firewall desde un cliente SSH, se configuró este en un puerto no estándar para mayor seguridad.
- ✓ Todas las cuentas de usuario y servicios innecesarios creados para pruebas en el servidor fueron deshabilitados del sistema.
- ✓ Por último, se probó con Netstat[13]. Netstat genera informes detallados, de las conexiones establecidas y los puertos abiertos o bloqueados, esto con el fin de cerrar aquellos que no son necesarios.

3.5 EL HONEYPOT LINUX.

Para este Honeypot se ha elegido como sistema operativo base la distribución Linux CentOS 5.2 con una configuración básica más los servicios que se desea controlar.

[13]**Netstat** es una herramienta de línea de comandos que muestra un listado de las conecciones activas de un ordenador y los puertos locales y remotos utilizados.

3.5.1 Servicios.

Se decidió ejecutar sobre este Honeypot los siguientes servicios: Servidor Web (Http), FTP, SSH (Secure Shell), Servidor Mail, y servicios de base de datos que son los más usados en los sistemas en producción.

La distribución CentOS Linux viene con el Servidor Web Apache (versión 1.3.23-11), se utiliza como servidor web corriendo en su puerto por defecto (80). Este Servidor Web se ha instalado con las configuraciones por defecto.

Apache-Tomcat, es un Servidor Web basado en Java, se configura en el puerto 8080.

Otros servidores que se instalaron fueron el Servidor Secure Shell (SSH), un protocolo de transferencia de archivos (FTP) server, el servidor MySQL (servidor de base de datos) y el servidor de Sendmail. Todos los servidores que se han configurado utilizan sus propios archivos de configuración. El Servidor FTP fue configurado para aceptar conexiones anónimas.

3.5.2 Cuentas de usuario.

Para aumentar el nivel de interacción se pueden añadir cuentas de usuario con diferentes grados de complejidad de la contraseña con la esperanza de que un hacker pueda romper la contraseña, esto podría atraer a algunos de los piratas informáticos especialmente calificados que encuentran difícil romper en tales sistemas.

3.6 EL HONEYPOT WINDOWS.

Se seleccionó Windows XP Profesional como Sistema Operativo para el Honeypot Windows.

3.6.1 Servicios.

Internet Information Services (IIS), ofrece una serie de servicios para ordenadores que funcionan bajo plataforma Windows. Los servicios levantados en este Honeypot son: HTTP, FTP, SMTP. Estos servicios convierten a este ordenador en un Servidor de Internet en el cual se pueden publicar páginas web.

Aprovechando este servicio, se decidió configurar un entorno virtual de aprendizaje EVA, en la plataforma Moodle-1.9.2, con la finalidad de simular un servicio real de la UTPL.

Actividades realizadas para la simulación.

- ✓ Establecer una interfaz gráfica similar al entorno de la UTPL, como por ejemplo los colores y el logo.
- ✓ Crear cursos. Se crearon varios cursos con temas como los que ofrece la universidad, como por ejemplo el del Taller Web 2.0 que abarca a varios paralelos debido a la afluencia de estudiantes que tiene.
- ✓ Asignar roles a usuarios: Administrador y Tutor de curso.
- ✓ Crear listado de estudiantes. Se crearon usuarios tipo estudiantes.

3.6.2 Otros servicios.

MySQL-5.0. Gestor de base de datos. Su licencia es gratuita, permite la interacción Web, es rápida, flexible, estable, robusta y fácil de utilizar. Abre el puerto 3306.

Todos los servicios de Windows que se cargan por defecto, no fueron modificados para que sean explotados por los atacantes:

- ✓ Puerto 135 (TCP) - para el Servicio de Remote Procedure Call (RPC)
- ✓ Puerto 137 (UDP) - para el Servicio de nombres de NetBIOS
- ✓ Puerto 138 (UDP) - para Netlogon y Browsing de NetBIOS
- ✓ Puerto 139 (TCP) - para la sesión (NET USE) de NetBIOS

Todos estos servicios tienen algunas fallas de seguridad.

3.7 RESTAURACIÓN Y COPIA DE SEGURIDAD (BACKUP).

Después de la instalación y configuración de todo, se utilizó Symantec Ghost[14] para crear una imagen de los Honeypots. Una imagen del sistema es una copia comprimida de todo el sistema almacenado como un archivo. Estas imágenes de los sistemas Honeypots se copian en un CD-ROM de arranque. Si se compromete un Honeypot, la imagen le ayudará en la restauración. También Ghost puede ser usado para crear y

[14] Ghost. Herramienta que permite guardar las imágenes de las máquinas típicas y duplicarlas rápidamente.

guardar una imagen de un sistema comprometido, el cual puede ser guardado para un posterior análisis detallado.

3.8 PLAN DE PRUEBAS.

Cada vez que se activó una Honeynet, fue imprescindible hacer que pase por una serie de pruebas, las cuales garantizaban el correcto funcionamiento de su hardware y software, fue muy útil elaborar una lista de pasos o pruebas (Check List) necesarias con las cuales se pueden determinar si la Honeynet y cada uno de sus elementos funcionan correctamente. Con estas pruebas no solo garantizamos la correcta recolección de datos, sino también la integridad del proyecto como tal, si dejábamos un solo indicio a los atacantes de que se trataba de un Honeypot, podría introducirse ruido en las muestras recogidas, o en el peor de los casos, se intentaría usar la Honeynet como una arma contra nosotros mismos.

Las pruebas iniciaban desde un chequeo del cableado, continuando con la revisión de cada elemento de recolección en la Honeynet, finalmente se revisaba si la Honeynet no podría ser usada como una herramienta contra la red interna o alguna máquina en Internet, para lograrlo se limitaron paquetes salientes.

Seguidamente se muestran las actividades que se revisan constantemente para asegurar la estabilidad, seguridad y control de la Honeynet en la Tabla 2.

Tabla 2.
CHECK LIST: Estabilidad y control de la Honeynet

SECUENCIA	DESCRIPCIÓN	RESULTADO (REALIZADO / NO REALIZADO)
1	Revision de cableado.	
2	Conectividad de equipos de la red Honeynet.	
3	Honeypots estén trabajando normalmente y no estén caídos.	
4	Control de conexiones - Reglas de Firewall	
5	Alertas al correo electrónico	
6	Recolecta de datos y visualización en Walleye.	

Otra acción importante que se realiza es la prueba del funcionamiento del Honeywall su conectividad y captura de datos; a continuación en la Tabla 3 el Check List de los pasos a comprobar:

Tabla 3.
CHECK LIST: Ejecución del Honeywall

SECUENCIA	DESCRIPCIÓN	RESULTADO (REALIZADO / NO REALIZADO)
1	Inicialización de servicios del Honeywall. (Walleye, Snort, Sebek, Pof, Swatch, Argus)	
2	Conectividad entre los Honeypots y el Honeywall	
3	Conectividad entre los Honeypots con el Internet	
4	Registro en los logs	
5	Sebek está realizando captura de datos en los honeypots	
6	Tcpdump está capturando tráfico de red	
7	Walleye este activado y sea accesible via HTTPS	
8	Walleye regsitre y visualice actividad del sistema	
9	Honetwall envía mensajes de alerta	

3.9 PROBLEMAS Y AJUSTES.

En la actualidad, en el proyecto Honeynet UTPL se está trabajando con Honeypots dentro de la red universitaria desde el año 2008, estos resultados servirán para estudiar y las técnicas, tácticas y motivos de los atacantes, también permite contar con un mecanismo adicional en el monitoreo de la red, así como ser un recurso insustituible en las actividades académicas y de investigación.

Para lograr esta estabilidad se fueron corrigiendo algunos detalles de configuración e implementación de hardware y software en el equipo Honeywall.

3.9.1 Puesta en marcha de la interfaz Walleye.

La puesta en marcha de la interfaz Walleye fue uno de los primeros inconvenientes a corregir, pues no se lograba mostrar ninguna información, como son las estadísticas de flujo, los detalles de tráfico, las alertas IDS y ninguna información avanzada de las

actividades del sistema. Al inicio del proyecto se tenía el servidor en un equipo de características PII de 400 Mhz de procesador y 256 Mb de memoria RAM. Para dar solución a este problema fue necesario actualizar las características de hardware del equipo Honeywall, las capacidades limitadas de memoria y procesamiento no permitían procesar y ejecutar las herramientas y subsistemas que hay en este equipo.

Se probó instalando el sistema Honeywall en un PC PIV de 3.0 Mhz de procesamiento y de 1 GB de memoria RAM., inmediatamente se observó que la interfaz Walleye empezó a mostrar gráficamente detalles del tráfico de la red Honeynet. De esta manera se logró solucionar este problema.

Después de que ya funcionaba el sistema, el procesamiento de datos en el servidor era demasiado lento lo que llevó a aumentar la cantidad de memoria RAM. En la actualidad se tiene el Honeywall ejecutándose con 4 GB de RAM lo que mejora notablemente el rendimiento del Honeywall.

3.9.2 Instalación de Sebek Cliente en distribuciones Ubuntu.

Otro problema que se presentó fue al instalar Sebek Cliente en un Honeypot Linux con Ubuntu 7.10. Al intentar actualizar el sistema se obtiene un mensaje de error el cual nos indica que los repositorios para esta versión han expirado.

Se ha intentado instalar en algunas versiones como Ubuntu 7.0 Server, Ubuntu 8.0.

La posible solución a este inconveniente sería compilar el Kernel a nuestras necesidades.

Actualmente se está probando con un Sebek Cliente para la distribución CentOS 5.0, que además es la plataforma en la que corren la mayoría de los servidores de la UTPL.

3.9.3 Instalación de Sebek Cliente en plataformas Windows.

Al instalar versiones de Sebek Cliente menores a 3.0.3 en la máquina Windows, causa inconvenientes como:

- ✓ Inestabilidad en el sistema operativo, en ocasiones bloqueándolo por completo a causa de un volcado de memoria a nivel del Kernel,

- ✓ Reinicia el computador cada 2 o 3 minutos aproximadamente,
- ✓ Colgar el computador cada 2 0 3 minutos aproximadamente.

Este problema es causado porque el módulo de Sebek se instala como parte del Kernel de Windows.

Cuando esto sucede lo que se hace es desinstalar Sebek Cliente a nivel de consola como se indica en el manual de instalación y desinstalación de Sebek Cliente en plataformas Windows. En el **ANEXO C** encuentra con más detalle la instalación de Sebek en plataformas Windows.

Existen versiones actuales de Sebek Cliente 3.0.4, las cuales causan inestabilidad al sistema pero en menor proporción, reinicia el sistema ocasionalemente siendo normal este comportamiento en las máquinas con plataforma Windows.

Cabe señalar que se obtiene el mismo comportamiento en todas las plataformas Windows.

3.9.4 Herramientas de Análisis Walleye

Durante la primera etapa, la interfaz gráfica Walleye estaba sin funcionar, esto debido a inconvenientes en cuanto a las versiones de software y hardware utilizadas. El resto de archivos pcap capturados, no fueron analizados debido a que estos no registraban datos Sebek, es decir no se registraba actividad desde los Honeypots hacia afuera.

Los Honeypots han logrado generar una gran cantidad de tráfico la misma que se procesa en el Honeywall, lo que ha causado que la interfaz web en ocasiones colapse quedándo por momentos sin conexión al servidor. Esto se debe a que la Base de Datos Relacional que tiene el Honeywall no logra almacenar tanta cantidad de tráfico que en ocasiones se llegó a generar especialmente en la segunda etapa en la que se obtuvo la mayor cantidad de Información. Para estabilizar el Honeywall en estas situaciones se tiene que reiniciar los servicios especialmente de mysql y de tcpdump, los cuales son los que almacenan y capturan el tráfico de la red respectivamente.

De la misma forma, por el mismo motivo del problema descrito anteriormente la Honeynet recibe una cantidad ingente de tráfico que normalmente no son ataques, sino sondeos o ataques fallidos, entonces se tiene que los archivos Pcap descargados del Honeywall llegan a tener tamaños muy grandes para su análisis, lo que conlleva a que los equipos en los que se hace el análisis de estos archivos dejen de funcionar y

la aplicación en este caso el Wireshark y el Honeysnap no se logren ejecutar con normalidad. Hay que descargar los Pcaps por horas para disminuir su tamaño o en su defecto a los archivos ya descargados dividirlos en partes de menos tamaño para así lograr que se ejecuten las aplicaciones y realizar el análisis. Estas actividades se las detalla en los manuales de Administración del sistema.

3.10 PROCESOS DE MANTENIMIENTO.

Luego de realizar una instalación exitosa de los Honeypots se debe obtener una imagen del sistema de instalación.

Algunas de las causas por las cuales se debe realizar el mantenimiento constante a un Honeypot son: *fallas en el software de instalación, ataques a los Honeypots en los cuales el daño es muy grande, etc.*, ante estas circunstancias se sugiere desactivar los Honeypots, siguiendo los siguientes pasos:

- ✓ Desconectar el Honeypot de la Honeynet.
- ✓ Recolectar y analizar el estado del sistema.
- ✓ Generar el sistema de imagen y compararlo con la imagen inicial.
- ✓ Reinstalar nuevamente el sistema en el Honeypot de acuerdo a los procedimientos establecidos.

Actualmente existe la imagen de los Honeypots, con el objetivo de realizar un posterior análisis forense.

Revisar constantemente la conectividad de los equipos de la red Honeynet.

Analizar el estado de los Honeypots y si están funcionando correctamente.

Revisar que los servicios del servidor esten trabajando, ejemplo: *mysql, tcpdump*, etc.

Cambiar la contraseña de acceso al servidor constantemente para evitar accesos no autorizados.

Respaldar y extraer los logs del servidor. Especialmente *messages e iptables*, esto como medida de precaución por si sufre algún daño el disco además por la razón que estos archivos van aumentando de tamaño día a día y al existir gran cantidad de tráfico que se registra en los logs puede que el disco se llene en su totalidad.

El mismo proceso se debe realizar con los archivos *Pcaps* que se almacenan en el servidor y utilizan gran capacidad en disco.

CAPÍTULO IV:

RECOLECCIÓN DE DATOS Y ANÁLISIS DE TRÁFICO.

4.1 INTRODUCCIÓN.

En esta sección se mostrará una descripción de la naturaleza y el tipo de ataques que la Honeynet ha registrado.

Se han recolectado datos suficientes como para demostrar que las computadoras conectadas a internet no están a salvo de los atacantes, en cualquier lugar que un equipo este en línea será victima de sondeos, escaneos y ataques.

4.2 HERRAMIENTAS DE ANÁLISIS DE DATOS UTILIZADAS.

Para realizar el análisis de datos se han utilizado herramientas que poseen características muy ventajosas para lograr el cometido de esta fase.

Las herramientas utilizadas con algunas de sus características se describen en la Tabla 4., las mismas que en el capítulo IV, se describieron con mayor detalle.

Tabla 4. Herramientas de análisis

HERRAMIENTA	CARACTERÍSTICAS
Walleye	✓ Presenta información del tráfico de red ✓ Permite solicitar información sobre Honeypots individuales ✓ Presenta una visión general de las actividades del atacante ✓ Exporta tráfico de red en formato pcap ✓ Permite modificar algunas configuraciones del Honeywall
Wireshark	✓ Analiza la información capturada de paquetes en formato pcap ✓ Analiza protocolos ✓ Examina datos de una red en tiempo real ✓ Presenta el flujo reconstruido de una sesión de TCP
Honeysnap	✓ Vista global de conexiones y paquetes. ✓ Extracción de flujo y descifrar comunicaciones basadas en ASCII. ✓ Analizar protocolos de comunicación de Internet más comunes como HTTP. ✓ Resumen de flujo de conexiones entrantes y salientes. ✓ Extracción de pulsaciones de teclado de paquetes Sebek (versión 2 y 3). ✓ Identificación y análisis de tráfico IRC.

4.3 ANÁLISIS DE DATOS OBTENIDOS EN LA HONEYNET.

Los Honeypots están en línea desde el mes de febrero del 2009. Seis meses de registro han producido interesantes y significativos resultados.

En aproximadamente 6 meses, los Honeypots registraron 4345170 conexiones, 15604 direcciones IP diferentes trataron de conectarse de alguna forma a los Honeypots. En general, se produjeron escaneos de puertos simultáneamente en ambos Honeypots, lo que indica que los escaneos fueron generados por scripts o herramientas automáticas de rastreo desde una misma dirección IP o desde varias direcciones IP.

Cabe destacar que el análisis fue realizado en tres etapas, de acuerdo al progreso del proyecto en cuanto a la estabilidad de los procesos del sistema Honeynet. Estas etapas se desarrollaron de la siguiente manera:

- ✓ Etapa 1: Análisis de paquetes capturados en los meses Febrero – Marzo. (Anexo G – Etapa I)
- ✓ Etapa 2: Análisis de paquetes capturados en los meses Mayo – Julio. (Anexo (Anexo G – Etapa II)
- ✓ Etapa 3: Análisis de paquetes capturados en los meses Agosto – Septiembre. (Anexo G – Etapa III)

Cada etapa contiene Información referente a:

- ✓ Cantidad de paquetes TCP, ICMP y UDP,
- ✓ Top de direcciones de origen,
- ✓ Alertas encontradas y su descripción.

A continuación se muestra los resultados más destacados del análisis.

4.4 NÚMERO DE ACCESOS POR PROTOCOLO.

De acuerdo a las etapas mencionadas y a la información obtenida en cada una de ellas se han analizado los protocolos que han sido mayormente escaneados logrando así obtener la siguiente gráfica estadística (Figura. 10):

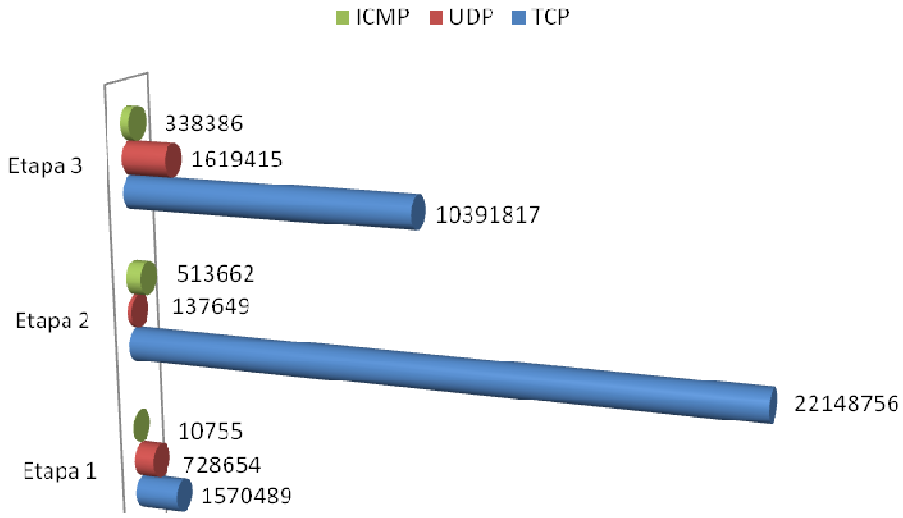


Figura 10. Total de accesos al Honeypot - Protocolos

- ✓ La Figura 10 representa el número total de conexiones a los protocolos TCP, UDP e ICMP, misma que nos indica que de los tres protocolos, el que más número de conexiones produce en una escala considerable es el TCP, se puede concluir que esto se debe en gran mayoría a los puertos abiertos TCP que por defecto tienen los sistemas y que son los más vulnerables.

- ✓ Durante la primera etapa se distingue, que se genera mayor cantidad de tráfico por el protocolo TCP, desde el Honeypot Linux, especialmente a los puertos 6667 que corresponden a los IRC más utilizados, seguido de los puertos correspondientes al NetBios (445, 135, etc.); en cuanto a la cantidad de accesos por los puertos UDP, se puede destacar que también corresponden a los puertos NetBios.

  El resumen obtenido por las herramientas de análisis de tráfico Wireshark y Honeysnap aplicado a los pcaps capturados en los meses de la primera etapa relacionado con el incidente IRC, es el siguiente:

Análisis de tráfico IRC que contiene PRIVMSG[15] en el Honeypot Linux.

Tabla 5. Tráfico IRC

Pattern	IPSource	Sport	IPDestino	Dport	Cant
Privmsg	200.0.30.51	37619	195.197.175.21	6667	1
Privmsg	195.18.164.194	6667	200.0.30.51	6667	72
Privmsg	195.197.175.21	6667	200.0.30.51	37619	71

[15]Privmsg script en Perl, utilizado para extraer archivos binarios en formato Pcap, diálogos en canales IRC.

Privmsg	194.109.20.90	6667	200.0.30.51	47355	128
Privmsg	217.168.95.245	6667	200.0.30.51	60307	40
Privmsg	200.0.30.51	6667	217.168.95.245	6667	1
Privmsg	161.53.178.51	6667	200.0.30.51	58058	36

Luego que los intrusos lograron comprometer el Honeypot Linux en esta etapa por medio del puerto TCP 6667, intentaron comunicarse con servidores de canales de Chat para establecer comunicación IRC. Seguidamente se muestra los servidores de chat que intentaron ser accedidos.

- ✓ Diemen.NL.EU.Undernet.Org
- ✓ Oslo2.NO.EU.undernet.org
- ✓ Oslo2.NO.EU.undernet.org
- ✓ Zagreb.HR.Eu.UnderNet.org
- ✓ Trondheim.NO.EU.Undernet.org
- ✓ Tampa.FL.US.Undernet.org
- ✓ Borning!~Lordx@Borning.users.undernet.org
- ✓ Born2Love!~born@Profi.users.undernet.org

Para acceder a estos servidores se utilizaron los siguientes Nicks para intentar establecer comunicaciones y autenticarse como usuarios.

- ✓ AUTH : 3240
- ✓ #NewHacks: 1207
- ✓ RaydeN : 1206
- ✓ Lordx : 922
- ✓ Born2love: 886
- ✓ Distrus : 813
- ✓ RaydeN : 485
- ✓ Lordx_ : 461
- ✓ _orn2love : 445

- ✓ En la segunda etapa se observó la mayor cantidad de accesos al protocolo 445 TCP del Honeypot Windows con respecto a las otras etapas como se muestra en la Figura 10, se obtuvieron las respectivas alertas que indican los intentos de conexión hacia estos, estas vulnerabilidades afectan a la familia de Sistemas Operativos Windows que

es la plataforma de escritorio más usada en el mundo y tiene los puertos más vulnerables en este caso el puerto TCP 445 fue el punto de ataques. La cantidad de conexiones simultáneas sobre el puerto 445 que está relacionada con el servicio para compartir recursos e intercambio de archivos, han generado 37956 alertas, *ICMP Destination Unreachable Communication Administratively Prohibited,* lo que significa que se estaba escaneando la red automáticamente buscando computadoras con este puerto abierto, lo que indica que un gusano estaba intentando propagarse.

Como medida correctiva al mencionado inconveniente, se bloqueó este puerto con la finalidad de disminuir la cantidad de tráfico desde el Honeypot Windows, ya que de esta manera se estaba generando mucho ruido desviándose y descuidándose así el análisis de otras alertas. Cabe mencionar que durante esta etapa el proyecto alcanzó mayor estabilidad con respecto a la anterior, por lo que se obtuvo mayor cantidad de datos para el análisis.

- ✓ El análisis en la tercera etapa presenta una disminución en la cantidad de conexiones, esto en representación al parche del puerto del protocolo TCP que en la etapa anterior generó mucho tráfico. El comportamiento de la red Honeynet en esta etapa presenta resultados similares a las anteriores, las alertas e intentos de conexión, puertos escaneados hacia los Honeypots casi no varían con la diferencia que se normaliza la cantidad de tráfico en la Honeynet por la acción correctiva que se tomó en la etapa anterior.

  Para más detalle de cada una de las etapas mencionadas revisar **ANEXO G**.

4.5 NÚMERO DE ACCESOS POR PUERTOS.

El análisis de los puertos escaneados se lo realizó desde la segunda etapa, por la razón que en este periodo se tiene mayor estabilidad en la captura de datos en el Honeywall. Se realiza este análisis con el fin de tener un panorama más detallado de los puertos utilizados para el intento de explotar vulnerabilidades y conocer cuáles son los más requeridos por los intrusos.

Los puertos más escaneados y que intentaron ser explotados durante todo este tiempo de recolección y análisis de datos son los que perteneces al protocolo TCP que son los puertos más vulnerables y con más frecuencia atacados. La siguiente tabla

muestra los puertos con más rastreo y que en su mayoría pertenecen al Honeypot Windows.

DESCRIPCION DEL PROTOCOLO TCP

Tabla 7. Accesos a puertos TCP

Puertos TCP	Total
445 (microsoft-ds)	3615997
Daytime	36625
80 (http)	20225
Tcpmux	14963
ident (113)	3302
22 (ssh)	1717
1433(ms-sql-s)	1260
135(epmap)	1090
2967 (ssc-agent)	1040
23 (telnet)	674
netbios- ssn(139)	482
25 (smtp)	375

La gráfica describe la representación de los intentos de acceso por los puertos TCP.

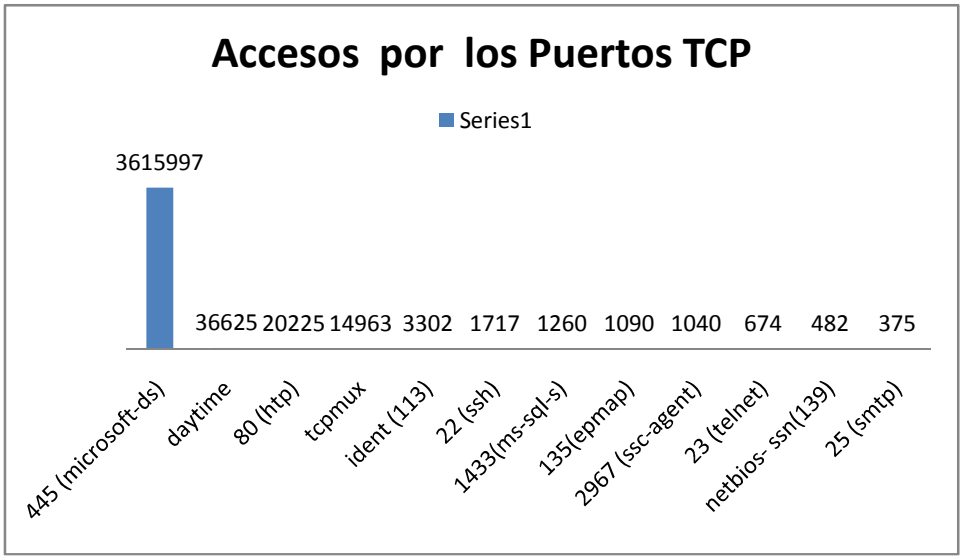


Figura 11. Accesos a puertos TCP

El tráfico recolectado muestra que los siguientes protocolos tuvieron actividad en la red: MICROSOFT-DS, EPMAP, IDENT, MS-SQL-S, DAYTIME, TCPMUX, SMTP, HTTP, SSH, DNS, NETBIOS , TELNET, DNS, IRCD. Otros resultados encontrados en proyectos afines muestran la similitud en los datos presentados. En la Honeynet de la ESPOL los protocolos con más actividad son: FTP, HTTP, SSH, DNS, NETBIOS Y TELNET. Coincidiendo que la mayor cantidad de tráfico es vía TCP por los protocolos http, netbios que son los que más debilidades presentan.

Seguidamente se analizan los protocolos más destacados que se mostraron en el análisis de datos recolectados.

PROTOCOLO MICROSOFT-DS

La tabla 7 muestra que el protocolo Microsoft-ds es de aproximadamente el 98% del tráfico total. Seguidamente de un Daytime con un 2% del tráfico total. Este comportamiento indica que los scaneos se los realiza con el fin de obtener información de los sistemas encontrados en la red, Microsoft-ds y daytime proporcionan datos que le servirán al atacante para realizar posibles conexiones remotas, ambos protocolos permiten comunicar máquinas mediante el puerto 445/TCP y 13TCP/UDP respectivamente. En los resultados presentados por el proyecto Honeynet UNAM se muestra estadísticas con altos porcentajes de tráfico al protocolo 445/TCP, lo que afirma que un comportamiento hacia estos puertos es constante debiendo tomar medidas preventivas para la seguridad de la red.

PROTOCOLO HTTP

El tráfico Web usa el protocolo de transferencia de hipertexto (HTTP), con el protocolo de control de transmisión (TCP). TCP proporciona varios servicios importantes para HTTP, incluyendo la transferencia de datos fiable y de control de gestión. En la Honeynets se ha capturado todas las peticiones y respuestas HTTP generadas por los Honeypors en un periodo de seis meses mostrando 20225 intentos de conexión. Las alertas para este protocolo es *WEB-IIS view source via translate header.* Esta alerta de acuerdo a Snort está clasificada como Acceso a una aplicación web potencialmente vulnerable, tiene una prioridad de 2. Ver Anexo 4: Vulnerabilidades encontradas

PROTOCOLO NETBIOS

NETBIOS (Network Basic Input Output System, Sistema Básico de E/S en la red) es un protocolo de comunicación entre ordenadores que comprende tres servicios (servicio de nombres, servicio de paquetes y servicio de sesión), y actualmente con la difusión de Internet, los sistemas operativos de Microsoft permiten ejecutar NETBIOS sobre el protocolo TCP/IP. Es un protocolo de compartición de archivos en una red, con la intensión de esparcirse a través de la compartición de archivos abiertos. Este protocolo generó 2269 alertas, *NETBIOS SMB-DS IPC$ unicode share access, NETBIOS SMB-DS srvsvc NetrPathCanonicalize unicode little endian overflow attempt* debido a que los Honeypots cuentan con recursos compartidos y el análisis muestra que se intentaron acceder a estos.

Se detalla las definiciones del resto de protocolos en el anexo G, etapa 4 Descripción de los protocolos.

Otros protocolos como el DNS, TELNET, FTP no proporcionan gran cantidad de tráfico en la Honeynet siendo un comportamiento contrario al presentado en la Honeynet UNAM que muestra gran cantidad de actividad vía puerto FTP.

DESCRIPCION DEL PROTOCOLO UDP

En cuanto a los puertos del protocolo UDP los que más intentos de conexión son los que pertenecen a la familia de los NetBios, el más referente es el DGM.

Tabla 8. Accesos a puertos UDP

Puertos UDP	Total
138 (netbios-dgm)	8726
53 (DNS)	4353
137 (netbios-ns)	1851
1434 (ms-sql-m)	486

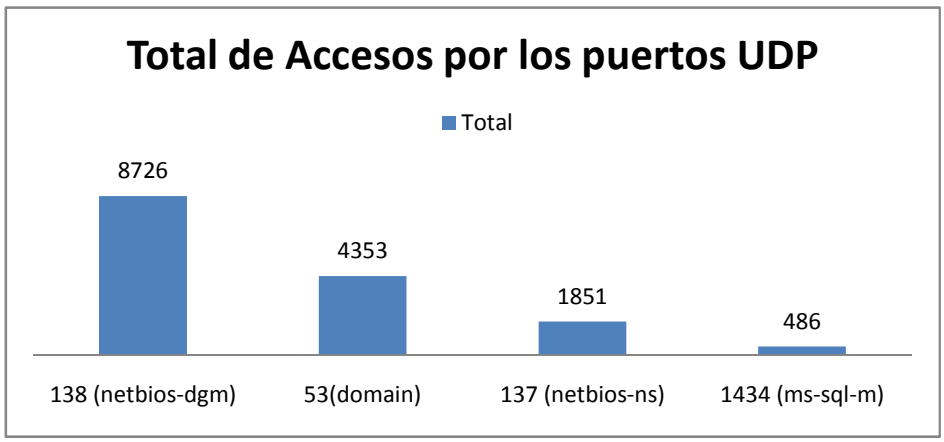


Figura 12. Accesos a puertos UDP

4.6 DIRECCIONES DE IP ORIGEN.

Los resultados del análisis de tráfico muestran que 204 direcciones Ip`s diferentes intentaron conectarse a los Honeypots desde muchos países de todo el mundo. Las direcciones listadas a continuación, son algunas direcciones origen hacia los Honeypots con su respectivo país, están ordenadas de acuerdo al número de paquetes transmitidos hacia los Honeypots.

Tabla 6. Direcciones IP origen

Direcciones	Organización	País	Conex. diarias
200.179.95.132	Embratel	Brasil	1269
85.185.146.4	ISDP	Irán	809
194.109.21.230	XS4ALL Servers	Países Bajos	768
64.210.19.210	Global Crossing	EEUU	684
85.42.132.98	ADI S.P.A.	Italia	441
217.168.95.245	NTEBB IRC Servernet	Noruega	419
161.53.178.240	CARNet backbone	Croacia	415
195.197.175.21	Saunalahti Group Oyj	Finlandia	391
81.17.16.18	ISP Relians	Rusia	333
202.32.71.80	MURATA MACHINERY, LTD.	Japón	290
84.74.121.36	Cablecom GmbH	Zurich	270
217.31.48.106	Ignum s.r.o	Rep. Checa	265
194.42.125.194	TheCloud	Reino Unido	249
220.189.222.68	Cixi jinlun Company	China	123
61.191.191.73	CHINANET Anhui province network	China	120
59.185.97.134	Mahanagar Telephone Nigam Ltd.	India	112
81.216.5.12	Sjofartsverket	Suecia	107

89.43.216.97	SC Bluelink SRL	Rumania	105

Las direcciones subrayadas son algunas direcciones que han sido reportadas como Maliciosas y son parte de listas negras.

Se han utilizado algunas páginas de Internet como: www.ip2location.com, www.xbl.spamhaus.org, www.web.dnsbl.sorbs.net, para identificar el origen, dominios, tipo de conexión y en qué listas negras están declaradas las direcciones IP que se han registrado en el Honeywall. En el **ANEXO G** se muestra en detalle las páginas utilizadas y el listado de direcciones IP que se registraron en todo este tiempo de recolección de datos.
El tipo de conexión a Internet más utilizado es la vía DSL.

EL país origen de las direcciones Ip en la Honeynet UTPL son Brasil, Estados Unidos, países de Asia y Europa. La UNAM muestra resultados en su web que países de Asia son las direcciones mas capturadas, concluyéndose que desde Asia se realizan escaneos a las redes de América.

4.7 ALERTAS RECIBIDAS.

El número total de alertas emitidas por el Snort y mostradas en la interfaz Walleye fueron de 29069 las que dividiremos en dos grupos, las *alertas ICMP* y *Otras alertas*. Las ICMP se las agrupa porque existen muchas alertas de intento de explotar vulnerabilidades de este tipo. Las otras alertas son intentos de accesos a las máquinas.

4.7.1 Alertas ICMP.

La mayoría de las alertas a las que se hace referencia son alertas de tipo ICMP, las mismas que tienen un tipo y un código definido por este protocolo.
El protocolo ICMP fue el que obtuvo más intentos de accesos, esto se concluye por la cantidad de alertas que hay a este protocolo, el análisis de estas alertas indican que son rastreos que realizan los atacantes a redes escogidas al azar con herramientas automatizadas con el fin de encontrar máquinas con vulnerabilidades y explotar las mismas.

En el **ANEXO G** se explica el funcionamiento de cada alerta, su nivel de riesgo, su prioridad, y el análisis detallado de la data que se obtiene de los archivos Pcap.

A continuación se muestra una lista con las alertas más importantes y que se obtuvieron con mayor frecuencia durante el análisis de tráfico de la Honeynet.

Tabla 9. Alertas ICMP

Descripción	Total
ICMP Destination Unreachable Communication Administratively Prohibited (1)	21110
ICMP PING CyberKit 2.2 Windows (2)	3031
ICMP redirect host (3)	1284
ICMP redirect net (4)	808
ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited (5)	328
ICMP PING NMAP	65
ICMP webtrends scanner	13
ICMP Destination Unreachable Communication with Destination Network is Administratively Prohibited	11
ICMP PING speedera	4
Total	**26654**

El siguiente gráfico muestra las alertas ICMP más comunes registradas en el Honeywall.

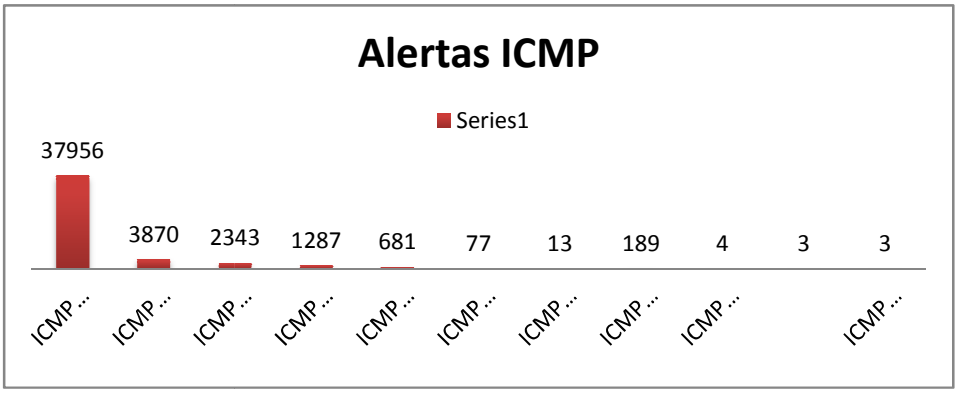


Figura 13. Alertas ICMP registradas en el Honeywall

4.7.2 Otras alertas.

Otras alertas que tambien se generaron son intentos de ataques a las aplicaciones de los Honeypots. El Honeypot Windows recibió más conexiones por lo que las alertas fueron emitidas de este sistema operativo.

Los servicios como el servidor web IIE (Internet Information server), mysql son los que reportaron intentos de vulnerabilidad. Otras alertas son intentos de propagación en la red como el Code Red y Sql Slammer que son menos eficaces a la hora de comprometer plenamente una LAN. En el **ANEXO G** se describe el comportamiento de cada una de las atertas encontradas.

Tabla 10. Otras alertas

Alerta	Junio	Mayo	julio	Total
MS-SQL Worm propagation attempt ; MS-SQL Worm propagation attempt OUTBOUND ; MS-SQL version overflow attempt (1)	740	456	18	1196
NETBIOS SMB-DS IPC$ unicode share access (2)	0	0	631	631
NETBIOS SMB-DS srvsvc NetrPathCanonicalize unicode little endian overflow attempt (3)	0	0	538	538
WEB-IIS view source via translate header (4)	0	0	19	19
BAD-TRAFFIC tcp port 0 traffic	3	16	4	16
SNMP request tcp	6	0	0	6
SNMP AgentX/tcp request	5	0	0	5
SNMP trap tcp	4	0	0	4
Total				**2415**

El siguiente gráfico muestra alertas adicionales al ICMP que se resgistraron en el Honeywall.

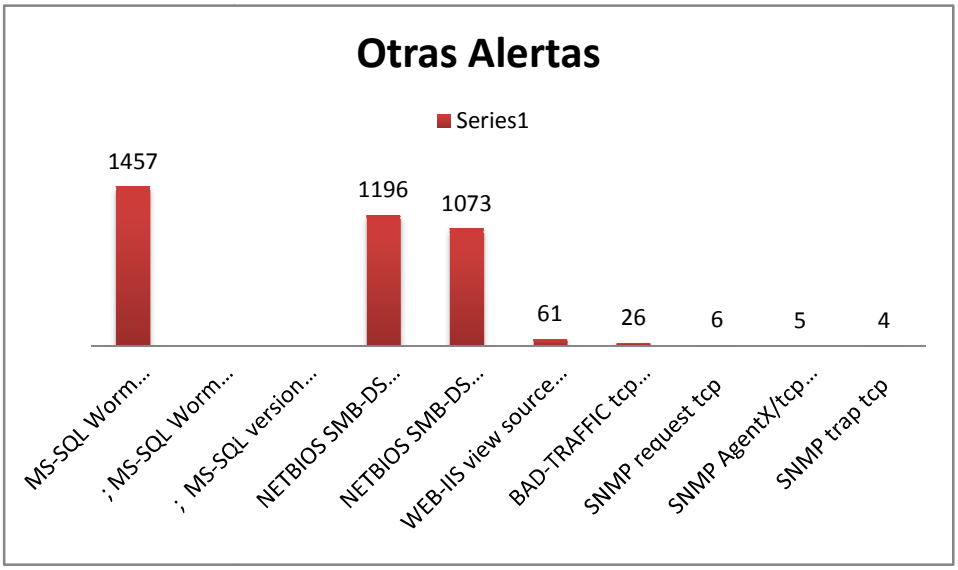


Figura 14. Otras alertas registradas en el Honeywall

4.7.3 Dirección IP de alertas.

De la mayor cantidad de alertas que se reportaron, la mayoría provienen de Asia específicamente de China. Seguidamente algunas IP`s – Dominios de China.

Tabla 11. Direcciones IP de alertas

Dirección	Dominio	País	Cantidad
61.145.123.141	ChinaNet Guangdong Province Network	China	38
60.161.78.144	CHINANET Yunnan province network	China	27
218.22.244.45	CHINANET Anhui province network	China	25
218.75.199.50	CHINANET-HN Zhuzhou node network	China	21
211.99.122.18	Jinan Dadu Hotel	China	17
89.34.153.157	SC UNDERNET SRL	Rumania	6

DISCUSION, CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS.

Discusión

La culminación de este proyecto deja como resultados el aprendizaje y experiencia en algunos campos relacionados a las redes. El conocimiento sobre Honeynet es sumamente extenso, abarca muchos temas; fue complicado tener el dominio completo de estos, y el reto fue ir aprendiendo día a día.

Los temas como redes, Linux, Windows, servidores, esquemas de red de la UTPL, control, captura y análisis de tráfico en la red, el mundo apasionante de la seguridad informática, los atacantes, tipos de intrusos, sus técnicas de ataque, herramientas, vulnerabilidades y parches de los sistemas operativos, sus orígenes y los efectos de sus ataques, fueron el pan de cada día mientras la Honeynet se imprentaba.

No podrían faltar los problemas durante este proceso. Comenzamos con la falta de colaboración por parte de proyectos afines en nuestro mismo medio, hubo un proyecto en nuestro país en otra universidad que fue imposible compartir información con ellos en bien de nuestra comunidad en entornos universitarios estrictamente en el campo de la investigación. Ahora este proyecto es el único en nuestro país y la idea es generar fuentes de conocimiento e investigación a partir de este, generar estadística de cómo estamos siendo atacados y la forma de mitigar estos riesgos, para luego transmitir esta información en nuestro medio y luego con sitios relacionados a este tipo de investigación. Otros problemas que están adjuntos en la documentación de este proyecto se trataron y fueron resueltos, algunos me llevaron mucho tiempo y otro no, al fin el agrado de contar con la Honeynet en la red de la UTPL.

La documentación presentada por el Proyecto Honeynet.org acerca de la implementación en redes trampa en instituciones académicas con infraestructuras muy grandes, característica común entre las redes académicas analizadas en esta sección como lo son Georgia Tech, la red de la ESPOL y la de UTPL, demuestra el valor del tipo de arquitectura implementadas de Generación II, lo cual es comprobado por la red trampa UTPL, ya que es grande la cantidad de datos generados y la importancia de su contenido que es utilizado para el análisis y servirán posteriormente para nuevos proyectos de investigación, lo que es objetivo común de los Proyectos Honeynet a nivel Mundial y que la Honeynet UTPL ha aportado para la constitución de nuevos proyectos de investigación, al contrario sucede con los datos generados por la Honeynet.ec de la ESPOL que tiene montada una infraestructura virtual lo que limita la

cantidad de captura de tráfico y las estadísticas presentan pequeñas cantidades de datos.

La recolección de datos contribuyeron a realizar el análisis para obtener el conocimiento de lo que sucede en la red externa además permitieron hacer comparaciones con los resultados que se tiene de herramientas de seguridad implantadas en la red de la UTPL, dando como resultado que coinciden con las alertas emitidas por los equipos de seguridad de la UTPL, lo que valoriza los datos obtenidos por la Honeynet, en otros casos se muestran datos adicionales como el descubrimiento de nuevos dominios, direcciones ip que realizan escaneos a la red los cuales sirven también para ser tomados en cuenta por los administradores de red.

Los resultados obtenidos en la Honeynet UTPL coinciden con las actividades encontradas en los análisis de datos de otros proyectos Honeynet a nivel mundial como el Honeynet.unam.mx de México, Honeynet.Br de Brasil se confirma que los puertos con mas actividad son los de TCP, que los países de origen de los ataques son de Asia y Europa, esta es otra forma de validar los resultados emitidos por la Honeynet.

Conclusiones

- ✓ En base al análisis del tráfico obtenido en la Honeynet se puede concluir:
- ✓ Los objetivos planteados al inicio de este proyecto se han cumplido, se ha implementado la Honeynet en la red de la UTPL y queda como infraestructura base para nuevos proyectos de investigación.
- ✓ Se contó con los recursos físicos e infraestructura necesaria para montar este proyecto debido a los requerimientos mínimos de hardware y software para la implementación.
- ✓ No existe soporte a problemas por parte de otras organizaciones que llevan estos proyectos, para el despliegue de una red trampa y los que existen no proporcionan ayuda real ya que al poner el sistema en producción se experimentaron algunos inconvenientes con las aplicaciones y no se tuvo repuesta inmediata. Quedan documentadas todas las acciones a seguir para solventar estos eventos.
- ✓ Al inicio de este proyecto no se logró establecer relaciones colaborativas con instituciones locales que llevaban el tema de investigación similar, finalmente se tuvieron lagunos intercambios de información.
- ✓ Los datos suministrados por la Honeynet demuestran resultados que aportan a la seguridad a los administradores de red. Los resultados obtenidos se contrastaron con los datos reales generados de los quipos de seguridad de la red de la UTPL resultando comportamientos que coinciden con los del tráfico de la UTPL y otros que deben ser tomados en cuenta para ser analizados por el administrador de la red y tomar acciones de tipo proactivo y preventivo. además de permitir validar su estado de seguridad actual.
- ✓ El tráfico recolectado en la Honeyntet que se ha reportado ha sido en los siguientes protocolos y los que más actividad generaron son: NETBIOS, HTTP, TELNET, DNS, DAYTIME.
- ✓ El análisis de datos demuestra que la mayor cantidad de tráfico ha sido generado desde los puertos Netbios, especialmente el TCP 445, esto ha producido gran cantidad de alertas, esto se debe a los puertos abiertos TCP que por defecto tienen los sistemas y que son los más vulnerables.
- ✓ En sistemas Windows los ataques son más automáticos, y fueron realizados en su totalidad por medio de la propagación de código malicioso, esto se conoce por las repetidas y numerosas cantidades de alertas que se emiten, tal es el caso de las alertas que muestran en la tabla 22 en el anexo 4.

- ✓ No es necesario contar con máquinas potentes para instalar una red trampa. La red Honeynet de la UTPL no utilizó máquinas nuevas y funcionó como se esperaba. Todo lo que necesitamos ya estaba disponible en el campus.
- ✓ El Honeywall es el elemento clave dentro de una arquitectura Honeynet. Hemos cubierto la mayoría de las medidas comunes necesarias para instalar uno de estos dispositivos y los requisitos mínimos de seguridad. Este proyecto ha sido de gran utilidad porque nos permite tener una base de configuración, y un punto de partida para experimentar y aprender en este amplio campo y lograr mayores beneficios a futuro.

RECOMENDACIONES

- ✓ Se recomienda al iniciar este proyecto, virtualizar la Honeynet. Con el fin de hacer fácil el entender el funcionamiento de esta tegnoloigía
- ✓ Se recomienda mantener una estrecha relación con los administradores de la red con la finalidad de informar de las actividades sospechosas que está viendo para tomar las medidas necesarias.
- ✓ Se recomienda seguir los métodos de mantenimiento y plan de pruebas con el fin de mantener la red trampa estable
- ✓ En el plan de mantenimiento, al obtener las imágenes de los discos de los honeypors se recomienda utilizar para este proceso, herramientas forenses digitales que permitan tener la una imagen preparada para un análisis forense posterior.
- ✓ Utilice varias herramientas para el análisis de datos, incluso para conseguir el mismo resultado. Aveces estas herramientas no son capaces de mostrar los resultados en forma adecuada.
- ✓ Desarrollar y personalizar sus propias herramientas, como scripts para automatizar el proceso de análisis de datos.
- ✓ Es necesario cerrar puertos de los honeypots cuando estos generan demasiado tráfico con el fin de evitar el colapso del sistema Honeywall y principalmente evitar distraer la atención para el análisis de otro tipo de tráfico.
- ✓ Análizar los logs, provenientes del servidor como Iptables, Messages. Estudiar especialmente los errores repetidos de acceso a un recurso. Analice también los casos en que se haya rechazado una conexión desde el interior hacia el exterior, esto puede significar que un virus se intenta comunicar con el exterior o que un usuario ha instalado un programa automatizado.
- ✓ No publique el rango de direcciones IP de la red trampa. No hay necesidad de hacer esto. Los atacantes y los gusanos (worms) están escaneando constantemente Internet en busca de máquinas vulnerables. Su red trampa será encontrada y atacada.
- ✓ Dedique todo el tiempo necesario para analizar los datos recogidos por la Honeynet. Estos datos deben ser analizados todos los días. Se recoge mucha información y debe ser analizada para que sea de mucha utilidad. La mayoría de los ataques tardan pocos segundos en comprometer y tomar el control de un sistema vulnerable. Puede llevar semanas analizar y documentar tales ataques.

TRABAJOS FUTUROS

La Honeynet al término de esta etapa deja abiertos algunos temas que se pueden explotar en el campo de la investigación principalmente.

La simulación de servicios reales que existen en los servidores de la UTPL en los Honeypots, es uno de los trabajos que se deben estudiar para obtener estadísticas de intentos de ataques a los servicios reales de la Universidad.

La virtualización de la Honeynet de la UTPL es otro tema a desarrollar para acumular información que permita realizar comparaciones con los datos existentes.

El proceso de aplicación de una metodología de análisis forense a los recursos de la Honeynet que permitirán concluir sobre hallazgos que se puedan encontrar.

La universidad cuenta ya con el estudio de un CERT C*omputer Emergency Response Team* es un equipo de personas dedicado a dar respuesta a incidentes y de gestión de seguridad, surge la necesidad de fortalecer este grupo, de retomar y actualizar sus metodologías para dar ayuda a los nuevas acometidas de los hackers.

La creación del grupo de Etichal Hacking que servirá para realizar actividades de acometidas a los equipos con fines de investigación. Este grupo es necesario para cumplir tareas de intrusión a vulnerabilidades de los equipos con el fin de contrastar con los datos encontrados en la Honeynet para validar las acometidas de los intrsuso y dar a conocer a los administradores y al grupo de seguridades la confirmación de técnicas, herramiestas y métodos confirmados de ataques.

Formar parte del proyecto Honeynet Distribuido que existe a nivel mundial con el fin de fortalecer la Honeynet UTPL e intercambiar experiencia obtenida con aquellas que tienen años en este campo y unificar el conocimiento de patrones de atques en la red universal.

Las redes trampa pueden ser una herramienta muy poderosa cuando se establecen en un entorno académico. No sólo pueden ser utilizadas para asegurar los recursos de producción actuando como una solución de detección fiable, sino que también pueden ser utilizadas para fomentar la investigación de la actividad maliciosa existente en las redes y una gran variedad de proyectos de investigación como el análisis de tráfico

para el comportamiento de incidentes como Spam, la creación de Honeynets Wi-Fi para analizar ataques realizados a redes inalámbricas, malware y ya que se ha iniciado el tema de IP V6, el estudio de implementación de una Honeynet con un direccionamiento de este tipo.

Al finalizar este proyecto se cuenta formalmente con una relación-convenio académica con el Proyecto Honeynet-UNAM de la Universidad Nacional Autónoma de México con los cuales se trabajará en conjunto para realizar el afinamiento de la Honeynet-UTPL e iniciar una colaboración mutua entre instituciones y se formará parte de uno de sus principales proyectos PSTM Plan de Sensores de Tráfico Malicioso.

BIBLIOGRAFÍA.

Referencias

[1] Internet Usage Statistics for the Americas (2009). Disponible en: http://www.internetworldstats.com/stats2.htm

[2] Black hat (2009). Disponible en: http://en.wikipedia.org/wiki/Black_hat

[3] Utimaco Data Encryption Software (2006). Disponible en: http://americas.utimaco.com/encryption/fbi_csi_2006_p3.html

[4] Honeynet Ecuador (2009). Disponible en: www.honeynet.ec

[5] SPITZNER, L. (2003): *Honeypots: Tracking Hackers*. Boston: Ed. Addison Wesley.

[6] INCO, A. (2007): *Honeypots*. Grupo de Seguridad Informática.

[7] Resultados y tendencias por parte del Honeynet Project. (2005). Disponible en: http://www.honeynet.unam.mx/docs/Intro.pdf

[8] HERNÁNDEZ Y LÓPEZ, M.J. y LERMA RESÉNDEZ, C.F.: *Aplicaciones Prácticas de los Honeypots en la Protección y Monitoreo de Redes de Información*.

[9] *SPECTER Intrusion Detection System* (2009). Disponible en: http://www.specter.com/

[10] Developments of the Honeyd Virtual Honeypot (2008). Disponible en: http://www.honeyd.org/

[11] KFSensor, Advanced Windows Honeypot System (2003). Disponible en: http://www.keyfocus.net/kfsensor/

[12] PatriotBox (Honey-Pot Server for Windows (2006). Disponible en: http://www.alkasis.com/?fuseaction=products.info&id=20

[13] Know Your Enemy: Honeynets (2006). Disponible en: http://old.honeynet.org/papers/honeynet/

[14] Symantec ManTrap (2004). Disponible en: https://www.arabtrust.com/partner/symantec/mantrap.html

[15] CARVAJAL, A. (2007): *Introducción a las técnicas de ataque e investigación Forense, un enfoque pragmático*. Colombia.

[16] VERDEJO, G. (2004): *Capítulo 4: Seguridad en Redes IP: Honeypots y Honeynets*.

[17] Ley de comercio electrónico, firmas electrónicas y mensajes de datos (2008). Disponible en: http://www.conatel.gov.ec/site_conatel/index.php?option=com_content&view=article&catid=48%3Anormas-del-sector&id=98%3Aley-de-comercio-electronico-firmas-electronicas-y-mensajes-de-datos&Itemid=103&limitstart=17

[18] The Deception Toolkit Home Page and Mailing List. Disponible en: http://www.all.net/dtk/dtk.html

[19] LaBrea: "Sticky" Honeypot and IDS. Disponible en: http://labrea.sourceforge.net/labrea-info.html

[20] Conoce a tu enemigo: Honeynets GenII (2003). Disponible en: http://his.sourceforge.net/honeynet/papers/gen2/

[21] RAJAN, S.: *Intrusion detection and the use of deception systems*. San Marcos. Texas.

[22] Snort Official Documentation (2009). Disponible en: http://www.snort.org/docs

[23] Conoce a tu enemigo: Honeywall CDROM (2004). Disponible en: http://his.sourceforge.net/honeynet/papers/cdrom/

[24] The new p0f: 2.0.8 (2006). Disponible en: http://lcamtuf.coredump.cx/pOf.shtml

[25] Swatch (2009). Disponible en: http://sourceforge.net/projects/swatch/

[26] Welcome to the Sebek project site. Disponible en: https://projects.honeynet.org/sebek

[27] The Honeynet Project. (2003): *Know your enemy: Sebek. A kernel based data capture tool*.

[28] BALAS, E. y VIECCO C. (2002): *Towards a Third Generation Data Capture Architecture for Honeynets*.

[29] Walleye. Tracking the attackers (2008). Disponible en: http://www.securityfocus.com/infocus/1855/2

[30] Wireshark User's Guide (2008). Disponible en: http://www.wireshark.org/docs/wsug_html_chunked/

[31] The Honeynet Project. Honeysnap. Disponible en: https://projects.honeynet.org/honeysnap/

[32] Niels Provos; Thorsten Holz, Virtual Honeypots: From Botnet Tracking to intrusión Detection, Addison Wesley professional, 2007

[33] Fingerprinting. Disponible en: http://nmap.org/nmap-fingerprinting-article-mx.html

ANEXOS.

ANEXO A
Archivo de configuración *rc.firewall*

```
#!/bin/bash
#
##########################################
#
# Copyright (C) <2005> <The Honeynet Project>
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or (at
# your option) any later version.
#
# This program is distributed in the hope that it will be useful, but
# WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
# General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
# USA
#
##########################################

#
# $Id: rc.firewall 5190 2007-03-13 17:54:40Z esammons $
#
# PURPOSE: To use the honeywall variables defined by the user and
#          configure the iptables firewall accordingly.  It has the following
#          possible arguments:
#          initial set default DROP rules and allow localhost access
#          start   configure the firewall according to honeywall.conf
#          stop    flush firewall and set forward default to drop.
#                  You should still be able to access the management
#                  interface.
#          restart stop then start
#
# chkconfig: 2345 20 95
# description: Honeywall iptables script, responsible for setting
#              default rules, white/black lists, etc.
# Comments: This is based on the same rc.firewall posted on
#           http://www.honeynet.org/papers/honeynet/tools/
# Modified: 12/21/04 by Allen Harper to incorporate -m physdev --physdev xxxx
#           syntax for linux kernel 2.6.4
# Modified: 2/15/05 by Allen Harper to incorporate a fence list concept to bound
#           outbound connections from hitting a list of IPs.
#
PATH="/sbin:/usr/sbin:/usr/local/sbin:/bin:/usr/bin"
. /etc/rc.d/init.d/functions
. /etc/rc.d/init.d/hwfuncs.sub

hw_setvars

# Note that load_modules() gets called each time the firewall is
# started/stopped.  In reality, modules only need to be loaded once,
# not each time this script is run, so it doesn't make sense to do
# this here. All of this should be done in a separate startup script
# that is run only once, very early in the boot process, and not
# here.
#
# In addition, there is a check to see if queueing is configured in
```

```
# order to determine if the queue module needs to be loaded.  This
# should probably just be done regardless, as the loading of the
# module when not used should not be an issue.
#
# These changes won't be made now, however, as this needs to be
# thought out and done carefully.  Attempting to reload the modules
# won't hurt anything.

load_modules() {
  #########
  # Load all required IPTables modules
  #

  lsmod | grep ipchain
  IPCHAINS=$?

  if [ "$IPCHAINS" = 0 ]; then
     echo ""
     echo "Dooh, IPChains is currently running! IPTables is required by"
     echo "the rc.firewall script. IPChains will be unloaded to allow"
     echo "IPTables to run.  It is recommened that you permanently"
     echo "disable IPChains in the /etc/rc.d startup scripts and enable"
     echo "IPTables instead."
     ipchains -F
     rmmod ipchains
  fi

  ### Add iptables target LOG.
  modprobe ipt_LOG

  ### Add iptables QUEUE support (Experimental)
  # Check this variable setting using the API to make sure it works
  # on a non-configured system.
  if [ "$(hw_get HwQUEUE)" = "yes" ]; then
     # Insert kernel mod
     modprobe ip_queue

     # check to see if it worked, if not exit with error
     lsmod | grep -q ip_queue
     IPQUEUE=$?

     if [ "$IPQUEUE" = 1 ]; then
        echo ""
        echo "It appears you do not have the ip_queue kernel module compiled"
        echo "for your kernel.  This module is required for Snort-Inline and"
        echo "QUEUE capabilities.  You either have to disable QUEUE, or compile"
        echo "the ip_queue kernel module for your kernel.  This module is part"
        echo "of the kernel source."
        exit
     fi


     # These are commented out, since the script used to dump all output
     # to /dev/null anyway.  (That had the negative side-effect of preventing
     # the "Starting firewall" note to come through, which was needed.)
     #echo "Enabling Snort-Inline capabilities, make sure Snort-Inline is"
     #echo "running in -Q mode, or all outbound traffic will be blocked"
  fi

  ### Support for connection tracking of FTP and IRC.
  modprobe ip_conntrack_ftp
  modprobe ip_conntrack_irc
}

# Fake a PID file in /var/run to support hwctl and Makefile.hwctl
create_pidfile() {
  echo ENABLED > /var/run/rc.firewall.pid
```

```
   #touch /var/lock/subsys/rc.firewall
}

# Make sure the fake PID file in /var/run is cleaned up.
delete_pidfile() {
   rm -f /var/run/rc.firewall.pid
   #rm -f /var/lock/subsys/rc.firewall
}


isactive() {
   if [ -f /var/run/rc.firewall.pid ]; then
      echo 1
      return 1
   fi
   echo 0
   return 0
}

# This function simply flushes all tables.  (Is this the right thing
# to do for restart?  Shouldn't tables be flushed individually right
# before they are populated?  Whatever is least disruptive.)

flush() {
   #########
   # Flush rules
   #
   iptables -F
   iptables -F -t nat
   iptables -F -t mangle
   iptables -X
}

# This function sets up the chains we will need for later use

create_chains() {


   if [ -n "${HwFWBLACK}" ] && [ -e ${HwFWBLACK} ] &&
      [ "${HwBWLIST_ENABLE}" == "yes" ]; then
      # BlackList chain
      iptables -N BlackList
   fi

   if [ -n "${HwFWWHITE}" ] && [ -e ${HwFWWHITE} ] &&
      [ "${HwBWLIST_ENABLE}" == "yes" ]; then
      # WhiteList chain
      iptables -N WhiteList
   fi

   if [ -n "${HwFWFENCE}" ] && [ -e ${HwFWFENCE} ] &&
      [ "${HwFENCELIST_ENABLE}" == "yes" ]; then
      # FenceList is the file listing of IPS that should not be reachable from
      # honeypots (outbound)
      iptables -N FenceList

      # FenceLogDrop is a table used to log/drop packets that bounce off
      # the fence. This is done so we don't need to write a log entry
      # for every ip in table.
      iptables -N FenceLogDrop
   fi

   if [ -n $HwTCPRATE ] && [ $HwTCPRATE -gt 0 ]; then
      # Create TCP handling chain
      iptables -N tcpHandler
   fi
```

```
  if [ -n $HwUDPRATE ] && [ $HwUDPRATE -gt 0 ]; then
    # Create UDP handling chain
    iptables -N udpHandler
  fi

  if [ -n $HwICMPRATE ] && [ $HwICMPRATE -gt 0 ]; then
    # Create ICMP handling chain
    iptables -N icmpHandler
  fi

  if [ -n $HwOTHERRATE ] && [ $HwOTHERRATE -gt 0 ]; then
    # Create other protocol handling chain
    iptables -N otherHandler
  fi

}

# Lockdown Policy
lockdown_policy () {
  iptables -P FORWARD DROP
}

# Set default policy to drop all.  Typically, this is done before
# anything else.  (This may need some logic to prevent temporarily
# dropping packets when this script is called to just restart the
# firewall rules when they are updated.)
default_policy() {
  iptables -P FORWARD DROP
  iptables -P INPUT DROP
  iptables -P OUTPUT DROP
}

# Set firewall rules allowing localhost access.
localhost_policy() {
  iptables -A INPUT -i lo -j ACCEPT
  iptables -A OUTPUT -o lo -j ACCEPT
}

# Set firewall rules allowing access to the management interface.
management_policy() {
  ##############################
  # MANAGEMENT INTERFACE RULES #
  ##############################
  if [ "${HwMANAGE_IFACE}" != "none" ] && [ -n "${HwMANAGE_IFACE}" ]; then
     # Make sure HwSSHD_PORT is in the list of allowed ports, and that we
     # don't have any duplicate ports.
     local portlist=`(for i in ${HwALLOWED_TCP_IN} ${HwSSHD_PORT}; do echo $i; done) | sort |uniq`
     for port in ${portlist}; do
        if [ "${HwMANAGER}" = "any" ]; then
           #iptables -A INPUT -m physdev --physdev-in ${HwMANAGE_IFACE} -p tcp --dport $port \
                 #-m state --state NEW -j LOG --log-level debug \
                 #--log-prefix "MANAGE port:$port=>"

              iptables -A INPUT -i ${HwMANAGE_IFACE} \
                    -p tcp -s 0.0.0.0/0 --dport $port -m state \
                   --state NEW -j ACCEPT

        else
           for ips in ${HwMANAGER}; do
              #iptables -A INPUT -m physdev --physdev-in ${HwMANAGE_IFACE} -p tcp -s $ips \
                    #--dport $port -m state --state NEW -j LOG --log-level debug \
                    #--log-prefix "MANAGE port:$port=>"
              iptables -A INPUT -i ${HwMANAGE_IFACE} \
                     -p tcp -s $ips  --dport $port -m state \
                     --state NEW -j ACCEPT

                     done
```

```
        fi
      done
      #handle outbound packets from management interface
      for port in ${HwALLOWED_TCP_OUT}; do
        iptables -A OUTPUT -o ${HwMANAGE_IFACE} -p tcp --dport $port \
        -m state --state NEW -j ACCEPT
      done

      for port in ${HwALLOWED_UDP_OUT}; do
        iptables -A OUTPUT -o ${HwMANAGE_IFACE} -p udp --dport $port \
        -j ACCEPT
      done
      #handle return tcp and udp from roo outbound after 3way handshake.
      iptables -A OUTPUT -o ${HwMANAGE_IFACE} -p tcp -m state \
            --state RELATED,ESTABLISHED -j ACCEPT
      iptables -A OUTPUT -o ${HwMANAGE_IFACE} -p udp -m state \
            --state RELATED -j ACCEPT
      # Allows return traffic
      iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
    fi
}

lockdown ()
{
    load_modules
    flush
    lockdown_policy
    localhost_policy
    management_policy

    create_pidfile
}
initial ()
{
    load_modules
    flush
    default_policy
    localhost_policy
    management_policy

    create_pidfile
}

start ()
{
    load_modules
    flush
    create_chains
    default_policy
    localhost_policy
    management_policy

    ##########
    # Check for blacklist variable and file.  This will drop and not log
    #   any ip within the file identified by the HwFWBLACK variable
    if [ -n "${HwFWBLACK}" ] && [ -e ${HwFWBLACK} ] &&
      [ "${HwBWLIST_ENABLE}" == "yes" ]; then

      iptables -A FORWARD -j BlackList

      logger -p local0.info "rc.firewall: loading blacklist from ${HwFWBLACK}"

      IPS=$(cat ${HwFWBLACK} | grep -v ^# | sed 's/[\r|\n]//')
      for ip in $IPS; do
        iptables -A BlackList -s "${ip}" -j DROP
        iptables -A BlackList -d "${ip}" -j DROP
      done
```

```
  # Return to the calling chain.
  iptables -A BlackList -j RETURN
fi

##########
# Check for whitelist variable and file.  This will accept and not log
#   any ip within the file identified by the HwFWWHITE variable
if [ -n "${HwFWWHITE}" ] && [ -e ${HwFWWHITE} ] &&
  [ "${HwBWLIST_ENABLE}" == "yes" ]; then

  iptables -A FORWARD -j WhiteList

  logger -p local0.info "rc.firewall: loading whitelist from ${HwFWWHITE}"

  IPS=$(cat ${HwFWWHITE} | grep -v ^# | sed 's/[\r|\n]//')
  for ip in $IPS; do
    iptables -A WhiteList -s "${ip}" -j ACCEPT
    iptables -A WhiteList -d "${ip}" -j ACCEPT
  done

  # Return to the calling chain.
  iptables -A WhiteList -j RETURN
fi

##########
# Check for fencelist variable and file.  This will drop AND log
#  any outbound dest ip within the file identified by the HwFWFENCE variable
if [ -n "${HwFWFENCE}" ] && [ -e ${HwFWFENCE} ] &&
  [ "${HwFENCELIST_ENABLE}" == "yes" ]; then

  # Only forward outbound packets from honeypots
  iptables -A FORWARD -m physdev --physdev-in ${HwLAN_IFACE} -j FenceList

  logger -p local0.info "rc.firewall: loading fencelist from ${HwFWFENCE}"

  IPS=$(cat ${HwFWFENCE} | grep -v ^# | sed 's/[\r|\n]//')
  for ip in $IPS; do
    iptables -A FenceList -d "${ip}" -j FenceLogDrop
  done

  iptables -A FenceLogDrop  -j LOG --log-level debug \
     --log-prefix "BOUNCED OFF Fence: "
  iptables -A FenceLogDrop  -j DROP


  # Return to the calling tables.
  iptables -A FenceList -j RETURN
  iptables -A FenceLogDrop -j RETURN
fi

##########
# Let's setup the firewall according to the Mode selected: bridge or nat
# Note: by default the firewall will be in bridge mode unless there is
# a value in the HwHPOT_PRIV_IP_FOR_NAT variable.

if [ -n "${HwHPOT_PRIV_IP_FOR_NAT}" ]; then    #NAT Mode
    logger -p local0.info "rc.firewall: enabling NAT mode"
    #Let's bring up our internal interface
    ifconfig ${HwLAN_IFACE} ${HwHPOT_PRIV_IP_FOR_NAT} \
      netmask ${HwALIAS_MASK_FOR_NAT} up

    i=0
    z=1
    tempPub=( ${HwHPOT_PUBLIC_IP} )

    for host in ${HwHPOT_PRIV_IP_FOR_NAT}; do
       if [ ${i} = "0" ]; then
```

```
          #This is the first honeypot. Let's attach it to our nic
          ifconfig ${HwINET_IFACE} ${tempPub[$i]} \
             netmask ${HwALIAS_MASK_FOR_NAT} up
       else

          # Bring up eth aliases
          ifconfig ${HwINET_IFACE}:${z} ${tempPub[$i]} \
             netmask ${HwALIAS_MASK_FOR_NAT} up
          let "z += 1"
       fi

       # Ensure proper NATing is performed for all honeypots
       iptables -t nat -A POSTROUTING -m physdev \
          --physdev-out ${HwINET_IFACE} -s ${host} \
          -j SNAT --to-source ${tempPub[$i]}
       iptables -t nat -A PREROUTING -m physdev \
          --physdev-in ${HwINET_IFACE} -d ${tempPub[$i]} \
          -j DNAT --to-destination ${host}
       let "i += 1"
    done
else
    logger -p local0.info "rc.firewall: enabling bridged mode"
fi

# Let's figure out dns
if [ -z "${HwDNS_HOST}" ]; then
    if [ -z "${HwHPOT_PRIV_IP_FOR_NAT}" ]; then   #bridge mode (default)
       HwDNS_HOST="${HwHPOT_PUBLIC_IP}"
    else                                          #nat mode
       HwDNS_HOST="${HwHPOT_PRIV_IP_FOR_NAT}"
    fi
fi


### Enable ip_forward
echo "1" > /proc/sys/net/ipv4/ip_forward

# Forward Chain:
#     Some of these rules may look redundant, but they allow us to catch
#    'other' protocols.


# Internet -> honeypot  -
#     This logs all inbound new connections and we must
#     specifically allow all inbound traffic because
#     the default policy for forwarding traffic
#     will be drop.  This will ensure if something
#     goes wrong with outbound connections, we
#     default to drop.
#
# Also, in case we have something listening to the QUEUE, we
#     will send all packets via the QUEUE.


# Since this is a bridge, we want to allow broadcast.  By default,
# we allow all inbound traffic (including broadcast).  We also want
# to allow outbound broadcast (such as NetBIOS) but we do not want
# to count it as an outbound session.  So we allow it here *before*
# we begin counting outbound connections

#iptables -A FORWARD -m physdev \
#    --physdev-in ${HwLAN_IFACE} -d ${HwLAN_BCAST_ADDRESS} \
#    -j LOG --log-level debug \
#    --log-prefix "Legal Broadcast: "
iptables -A FORWARD -d ${HwLAN_BCAST_ADDRESS} -j ACCEPT

#iptables -A FORWARD -m physdev --physdev-in ${HwLAN_IFACE} \
```

```
  #    -d 255.255.255.255 -j LOG --log-level debug \
  #    --log-prefix "Legal Broadcast: "

  iptables -A FORWARD -d 255.255.255.255 -j ACCEPT


  ### Inbound TCP
  iptables -A FORWARD -m physdev --physdev-in ${HwINET_IFACE} \
     -p tcp -m state --state NEW -j LOG --log-level debug \
     --log-prefix "INBOUND TCP: "
  iptables -A FORWARD -m physdev --physdev-in ${HwINET_IFACE} \
     -p tcp -m state --state NEW -j ACCEPT

  ### Inbound UDP
  iptables -A FORWARD -m physdev --physdev-in ${HwINET_IFACE} \
     -p udp -m state --state NEW -j LOG --log-level debug \
     --log-prefix "INBOUND UDP: "
  iptables -A FORWARD -m physdev --physdev-in ${HwINET_IFACE} \
     -p udp -m state --state NEW -j ACCEPT

  ### Inbound ICMP
  iptables -A FORWARD -m physdev --physdev-in ${HwINET_IFACE} \
     -p icmp -m state --state NEW -j LOG --log-level debug \
     --log-prefix "INBOUND ICMP: "
  iptables -A FORWARD -m physdev --physdev-in ${HwINET_IFACE} \
     -p icmp -m state --state NEW -j ACCEPT

  ### Inbound anything else
  iptables -A FORWARD -m physdev --physdev-in ${HwINET_IFACE} \
     -m state --state NEW -j LOG --log-level debug \
     --log-prefix "INBOUND OTHER: "
  iptables -A FORWARD -m physdev --physdev-in ${HwINET_IFACE} \
     -m state --state NEW -j ACCEPT

  # The remainder of established connections will be ACCEPTED.  The rules
  # above are required in order to log new inbound connections.
  iptables -A FORWARD -m physdev --physdev-in ${HwINET_IFACE} -j ACCEPT

  # This rule is for use with sebek.  If sebek is used, and we don't want
  # the logs filled by SPOOFED SOURCE entries because sebek uses spoofed
  # IPs, we should drop all traffic in the sebek ip range.
  if [ ${HwSEBEK} = "yes" ]; then
     if [ ${HwSEBEK_LOG} = "yes" ]; then
        iptables -A FORWARD -m physdev --physdev-in ${HwLAN_IFACE} \
           -p udp -d ${HwSEBEK_DST_IP} \
           --dport ${HwSEBEK_DST_PORT} -j LOG --log-level debug \
           --log-prefix "SEBEK"
     fi
     iptables -A FORWARD -m physdev --physdev-in ${HwLAN_IFACE} \
           -p udp -d ${HwSEBEK_DST_IP} \
           --dport ${HwSEBEK_DST_PORT} -j ${HwSEBEK_FATE}
  fi


  ### DNS / NTP Perhaps one of your honeypots needs consistent
  ### outbound access to provide internal service.

  # If we did not identify a specific destination dns server, let's go ahead
  # and allow any.
  if [ -z "${HwDNS_SVRS}" ]; then
     HwDNS_SVRS="0.0.0.0/0"
  fi

###########
# 1st ROACHMOTEL Feature Check (see other check below)
# Following test is to see if we are running in roach motel mode, where attackers
# can check into the motel (honeypots) but they cant send packets out.
```

```
############
if [ "${HwROACHMOTEL_ENABLE}" = "no" ]; then
  # Egress filtering, don't want to let our compromised honeypot send
  # spoofed packets.  Stops most outbound DoS attacks.  However, we might
  # want to allow our honeypots to use dhcp to get an ip while in
  # bridge mode.
  if [ -z  "${HwHPOT_PRIV_IP_FOR_NAT}" ]; then  # bridge mode
     iptables -A FORWARD -m physdev --physdev-in ${HwLAN_IFACE} \
        -p udp --sport 68 \
        -d 255.255.255.255 --dport 67 -j LOG --log-level debug \
        --log-prefix "DHCP OUT REQUEST: "

     iptables -A FORWARD -m physdev --physdev-in ${HwLAN_IFACE} \
        -p udp --sport 68 \
        -d 255.255.255.255 --dport 67 -j ACCEPT
  fi
  #now handle dns packets from honeypots
  for srvr in ${HwDNS_SVRS}; do
     for host in ${HwDNS_HOST}; do
        iptables -A FORWARD -p udp -m physdev --physdev-in ${HwLAN_IFACE} \
           -s ${host} -d ${srvr} \
           --dport 53 -j LOG --log-level debug \
           --log-prefix "Legal DNS: "
        iptables -A FORWARD -p tcp -m physdev --physdev-in ${HwLAN_IFACE} \
           -s ${host} -d ${srvr} \
           --dport 53 -j LOG --log-level debug \
           --log-prefix "Legal DNS: "

        iptables -A FORWARD -p udp -m physdev --physdev-in ${HwLAN_IFACE} \
           -s ${host} -d ${srvr} \
           --dport 53 -j ACCEPT
        iptables -A FORWARD -p tcp -m physdev --physdev-in ${HwLAN_IFACE} \
           -s ${host} -d ${srvr} \
           --dport 53 -j ACCEPT
     done
  done
# end 1st check for roach motel
fi
  if [ -n "${HwHPOT_PRIV_IP_FOR_NAT}" ]; then
     LIMIT_IP="${HwHPOT_PRIV_IP_FOR_NAT}"
  else
     LIMIT_IP="${HwHPOT_PUBLIC_IP}"
  fi

  ### Count and limit all other outbound connections

  # This will ensure we don't restrict Honeypots talking to each other, and
  # we don't log them as outbound connections (in bridge mode, the
  # firewall sees all packets; therefore, we have to make sure it doesn't
  # log packets incorrectly and give false positives).
  # If you do not want to see this log, comment out the logging rule.
  # You will still need the ACCEPT rule to ensure they honeypots can talk
  # to eachother freely.
  iptables -A FORWARD -m physdev --physdev-in ${HwLAN_IFACE} \
     --physdev-out ${HwLAN_IFACE} -j LOG --log-level debug \
     --log-prefix "Honeypot -> Honeypot: "

  iptables -A FORWARD -m physdev --physdev-in ${HwLAN_IFACE} \
     --physdev-out ${HwLAN_IFACE} -j ACCEPT

  # moved this section out of fencelist check, need it all the time (AAH).
  # This portion of the script will ensure that established or related
  # connections that were allowed, continue to work.  If these lines
  # are not here, only the first packet of each connection that hasn't
  # reached the limit will be allowed in because we are dropping
  # all outbound connections by default.
  if [ ${HwQUEUE} = "yes" ]; then
```

```
    iptables -A FORWARD -m physdev --physdev-in ${HwLAN_IFACE} \
       -m state --state RELATED,ESTABLISHED \
       -j QUEUE
  else
  iptables -A FORWARD -m physdev --physdev-in ${HwLAN_IFACE} \
     -m state --state RELATED,ESTABLISHED \
     -j ACCEPT
  fi

############
# 2nd ROACHMOTEL Feature Check (see other check in forward section above)
# Following test is to see if we are running in roach motel mode, where attackers
# can check into the motel (honeypots) but they cant send packets out.
############
if [ "${HwROACHMOTEL_ENABLE}" = "no" ]; then


  # Moved the following block to this location, should be subject to ROACHMOTEL mode
  # If we selected to restrict the firewall, lets implement it here.
  if [ ${HwRESTRICT} = "yes" ]; then
     for port in ${HwALLOWED_TCP_OUT}; do
        iptables -A OUTPUT -p tcp --dport $port -m state \
              --state NEW,ESTABLISHED,RELATED -j ACCEPT
     done

     for port in ${HwALLOWED_UDP_OUT}; do
        iptables -A OUTPUT -p udp --dport $port -m state \
              --state NEW,ESTABLISHED,RELATED -j ACCEPT
     done
  else
    # Change default to accept all outbound packets
    iptables -P OUTPUT ACCEPT
  fi



  for host in ${LIMIT_IP}; do

  # TCP:
  #   This next rule is the connection limiter.  If it has not exceeded
  #   the limit, the packet will be sent to the tcpHandler.  The
  #   tcpHandler will log and either QUEUE or ACCEPT depending on
  #   the Architecture selected.
  #
  #   NOTE: The purpose of the drop rule is to ensure we can catch 'other'
  #   protocols that enter our network.  If this statement is not here
  #   we will get false log entries stating "Drop other > xxx connections."

     if [ $HwTCPRATE -gt 0 ]; then

       iptables -A FORWARD -p tcp -m physdev --physdev-in ${HwLAN_IFACE} \
          -m state --state NEW \
          -m limit --limit ${HwTCPRATE}/${HwSCALE} \
          --limit-burst ${HwTCPRATE} -s ${host} -j tcpHandler

       iptables -A FORWARD -p tcp -m physdev --physdev-in ${HwLAN_IFACE} \
          -m state --state NEW \
          -m limit --limit 1/${HwSCALE} --limit-burst 1  -s ${host} \
          -j LOG --log-level debug \
          --log-prefix "Drop TCP > ${HwTCPRATE} attempts"

       iptables -A FORWARD -p tcp -m physdev --physdev-in ${HwLAN_IFACE} \
          -m state --state NEW \
          -s ${host} -j DROP

       # This rule is for Mike Clark in order to give him RELATED
       # information.  For example, this will tell him the data channel
```

```
      # related to an ftp command channel of a connection.
      iptables -A FORWARD -p tcp -m physdev --physdev-in ${HwLAN_IFACE} \
         -m state --state RELATED \
         -s ${host} -j tcpHandler
   fi

#
# UDP - see TCP comments above.
#
   if [ $HwUDPRATE -gt 0 ]; then

      iptables -A FORWARD -p udp -m physdev --physdev-in ${HwLAN_IFACE} \
         -m state --state NEW \
         -m limit --limit ${HwUDPRATE}/${HwSCALE} \
         --limit-burst ${HwUDPRATE} -s ${host} -j udpHandler

      iptables -A FORWARD -p udp -m physdev --physdev-in ${HwLAN_IFACE} \
         -m state --state NEW \
         -m limit --limit 1/${HwSCALE} --limit-burst 1 -s ${host} \
         -j LOG --log-level debug \
         --log-prefix "Drop udp > ${HwUDPRATE} attempts"

      iptables -A FORWARD -p udp -m physdev --physdev-in ${HwLAN_IFACE} \
         -m state --state NEW \
         -s ${host} -j DROP
   fi


#
# ICMP - see TCP comments above.
#
   if [ $HwICMPRATE -gt 0 ]; then

      iptables -A FORWARD -p icmp -m physdev --physdev-in ${HwLAN_IFACE} \
         -m state --state NEW \
         -m limit --limit ${HwICMPRATE}/${HwSCALE} \
         --limit-burst ${HwICMPRATE} -s ${host} -j icmpHandler

      iptables -A FORWARD -p icmp -m physdev --physdev-in ${HwLAN_IFACE} \
         -m state --state NEW \
         -m limit --limit 1/${HwSCALE} --limit-burst 1 -s ${host} \
         -j LOG --log-level debug \
         --log-prefix "Drop icmp > ${HwICMPRATE} attempts"

      iptables -A FORWARD -p icmp -m physdev --physdev-in ${HwLAN_IFACE} \
         -m state --state NEW \
         -s ${host} -j DROP
   fi
#
# EVERYTHING ELSE - see TCP comments above.
#
   if [ $HwOTHERRATE -gt 0 ]; then

      iptables -A FORWARD -m physdev --physdev-in ${HwLAN_IFACE} \
         -m state --state NEW -m limit \
         --limit ${HwOTHERRATE}/${HwSCALE} --limit-burst ${HwOTHERRATE} \
         -s ${host} -j otherHandler

      iptables -A FORWARD -m physdev --physdev-in ${HwLAN_IFACE} \
         -m state --state NEW -m limit \
         --limit 1/${HwSCALE} --limit-burst 1 -s ${host} \
         -j LOG --log-level debug \
         --log-prefix "Drop other > ${HwOTHERRATE} attempts"
   fi

done
```

```
    ###  These define the handlers that actually limit outbound connection.
    #
    # tcpHandler - The only packets that should make it into these chains
    #              are new connections, as long as the host has not
    #              exceeded their limit.
    #
    #
    iptables -A tcpHandler -j LOG --log-level debug \
        --log-prefix "OUTBOUND TCP: "
    if [ ${HwQUEUE} = "yes" ]; then
        iptables -A tcpHandler -j QUEUE
    fi
    iptables -A tcpHandler -j ACCEPT


    #
    # udpHandler - see tcpHandler comments above.
    #
    iptables -A udpHandler -j LOG --log-level debug \
        --log-prefix "OUTBOUND UDP: "
    if [ ${HwQUEUE} = "yes" ]; then
        iptables -A udpHandler -j QUEUE
    fi
    iptables -A udpHandler -j ACCEPT


    #
    # icmpHandler - see tcpHandler comments above.
    #
    iptables -A icmpHandler -j LOG --log-level debug \
        --log-prefix "OUTBOUND ICMP: "
    if [ ${HwQUEUE} = "yes" ]; then
        iptables -A icmpHandler -j QUEUE
    fi
    iptables -A icmpHandler -j ACCEPT


    #
    # otherHandler - see tcpHandler comments above.
    #
    iptables -A otherHandler -j LOG --log-level debug \
        --log-prefix "OUTBOUND OTHER: "
    if [ ${HwQUEUE} = "yes" ]; then
       iptables -A otherHandler -j QUEUE
    fi
    iptables -A otherHandler -j ACCEPT

################################
#end the roach motel mode check
fi
################################


    create_pidfile
    action "Starting up Firewall: " /bin/true
    return 0
}

stop ()
{
    flush
    default_policy
    localhost_policy
    ### Disable ip_forward
    echo "0" > /proc/sys/net/ipv4/ip_forward
    delete_pidfile
    action "Stopping Firewall: " /bin/true
```

```
   return 0
}

checkconfigured() {
    if [ $(hw_isconfigured) -eq 0 ]; then
        echo 1
        return 1
    else
        echo 0
        return 0
    fi
}

# Begin main body


case "$1" in
    start)
        if [ "$(checkconfigured)" -eq 0 ]; then
           start
        fi
        ;;
    stop)
        if [ "$(checkconfigured)" -eq 0 ]; then
           stop
        fi
        ;;
    restart)
        if [ $(isactive) -eq 1 ]; then
           stop
        fi
        start
        ;;
    initial)
        initial
        action "Loading initial firewall rules: " /bin/true
        ;;
  lockdown)
             lockdown
             action "Loading lockdown firewall rules: " /bin/true
             ;;
    status)
        if [ $(isactive) -eq 1 ]; then
            echo "$0 is active"
            exit 0
        else
            echo "$0 is not active"
            exit 1
        fi
        ;;
    *)
        echo "Usage: $0 {start|stop|restart|initial|status}"
        exit 1
        ;;
esac

# Exit with a return value
exit 0
```

ANEXO B

Archivo de configuración *snort.conf*

```
#--------------------------------------------------
#   http://www.snort.org     Snort 2.6.1.5 Ruleset
#     Contact: snort-sigs@lists.sourceforge.net
#--------------------------------------------------
# $Id$
#
##################################################
# This file contains a sample snort configuration.
# You can take the following steps to create your own custom configuration:
#
#  1) Set the variables for your network
#  2) Configure dynamic loaded libraries
#  3) Configure preprocessors
#  4) Configure output plugins
#  5) Add any runtime config directives
#  6) Customize your rule set
#
##################################################
# Step #1: Set the network variables:
#
# You must change the following variables to reflect your local network. The
# variable is currently setup for an RFC 1918 address space.
#
# You can specify it explicitly as:
#
# var HOME_NET 10.1.1.0/24
#
# or use global variable $<interfacename>_ADDRESS which will be always
# initialized to IP address and netmask of the network interface which you run
# snort at.  Under Windows, this must be specified as
# $(<interfacename>_ADDRESS), such as:
# $(\Device\Packet_{12345678-90AB-CDEF-1234567890AB}_ADDRESS)
#
# var HOME_NET $eth0_ADDRESS
#
# You can specify lists of IP addresses for HOME_NET
# by separating the IPs with commas like this:
#
# var HOME_NET [10.1.1.0/24,192.168.1.0/24]
#
# MAKE SURE YOU DON'T PLACE ANY SPACES IN YOUR LIST!
#
# or you can specify the variable to be any IP address
# like this:

var HOME_NET any

# Set up the external network addresses as well.  A good start may be "any"
var EXTERNAL_NET any

# Configure your server lists.  This allows snort to only look for attacks to
# systems that have a service up.  Why look for HTTP attacks if you are not
# running a web server?  This allows quick filtering based on IP addresses
# These configurations MUST follow the same configuration scheme as defined
# above for $HOME_NET.

# List of DNS servers on your network
var DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
```

```
var SMTP_SERVERS $HOME_NET

# List of web servers on your network
var HTTP_SERVERS $HOME_NET

# List of sql servers on your network
var SQL_SERVERS $HOME_NET

# List of telnet servers on your network
var TELNET_SERVERS $HOME_NET

# List of snmp servers on your network
var SNMP_SERVERS $HOME_NET

# Configure your service ports.  This allows snort to look for attacks destined
# to a specific application only on the ports that application runs on.  For
# example, if you run a web server on port 8081, set your HTTP_PORTS variable
# like this:
#
# var HTTP_PORTS 8081
#
# Port lists must either be continuous [eg 80:8080], or a single port [eg 80].
# We will adding support for a real list of ports in the future.

# Ports you run web servers on
#
# Please note:  [80,8080] does not work.
# If you wish to define multiple HTTP ports, use the following convention
# when customizing your rule set (as part of Step #6 below).  This should
# not be done here, as the rules files may depend on the classifications
# and/or references, which are included below.
#
## var HTTP_PORTS 80
## include somefile.rules
## var HTTP_PORTS 8080
## include somefile.rules
var HTTP_PORTS 80

# Ports you want to look for SHELLCODE on.
var SHELLCODE_PORTS !80

# Ports you do oracle attacks on
var ORACLE_PORTS 1521

# other variables
#
# AIM servers.  AOL has a habit of adding new AIM servers, so instead of
# modifying the signatures when they do, we add them to this list of servers.
var                                                                    AIM_SERVERS
[64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.188.3.0/24,205.188.5.0/2
4,205.188.7.0/24,205.188.9.0/24,205.188.153.0/24,205.188.179.0/24,205.188.248.0/24]

# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH /etc/snort/rules

# Configure the snort decoder
# ============================
#
# Snort's decoder will alert on lots of things such as header
# truncation or options of unusual length or infrequently used tcp options
#
#
# Stop generic decode events:
#
# config disable_decode_alerts
```

```
#
# Stop Alerts on experimental TCP options
#
# config disable_tcpopt_experimental_alerts
#
# Stop Alerts on obsolete TCP options
#
# config disable_tcpopt_obsolete_alerts
#
# Stop Alerts on T/TCP alerts
#
# In snort 2.0.1 and above, this only alerts when a TCP option is detected
# that shows T/TCP being actively used on the network.  If this is normal
# behavior for your network, disable the next option.
#
# config disable_tcpopt_ttcp_alerts
#
# Stop Alerts on all other TCPOption type events:
#
# config disable_tcpopt_alerts
#
# Stop Alerts on invalid ip options
#
# config disable_ipopt_alerts
#
# Alert if value in length field (IP, TCP, UDP) is greater than the
# actual length of the captured portion of the packet that the length
# is supposed to represent:
#
# config enable_decode_oversized_alerts
#
# Same as above, but drop packet if in Inline mode -
# enable_decode_oversized_alerts must be enabled for this to work:
#
# config enable_decode_oversized_drops
#

# Configure the detection engine
# ==============================
#
# Use a different pattern matcher in case you have a machine with very limited
# resources:
#
# config detection: search-method lowmem

# Configure Inline Resets
# =======================
#
# If running an iptables firewall with snort in InlineMode() we can now
# perform resets via a physical device. We grab the indev from iptables
# and use this for the interface on which to send resets. This config
# option takes an argument for the src mac address you want to use in the
# reset packet.  This way the bridge can remain stealthy. If the src mac
# option is not set we use the mac address of the indev device. If we
# don't set this option we will default to sending resets via raw socket,
# which needs an ipaddress to be assigned to the int.
#
# config layer2resets: 00:06:76:DD:5F:E3

##################################################
# Step #2: Configure dynamic loaded libraries
#
# If snort was configured to use dynamically loaded libraries,
# those libraries can be loaded here.
#
# Each of the following configuration options can be done via
# the command line as well.
```

```
#
# Load all dynamic preprocessors from the install path
# (same as command line option --dynamic-preprocessor-lib-dir)
#
dynamicpreprocessor directory /usr/lib/snort-2.6.1.5_dynamicpreprocessor/
#
# Load a specific dynamic preprocessor library from the install path
# (same as command line option --dynamic-preprocessor-lib)
#
# dynamicpreprocessor file /usr/local/lib/snort_dynamicpreprocessor/libdynamicexample.so
#
# Load a dynamic engine from the install path
# (same as command line option --dynamic-engine-lib)
#
dynamicengine /usr/lib/snort-2.6.1.5_dynamicengine/libsf_engine.so
#
# Load all dynamic rules libraries from the install path
# (same as command line option --dynamic-detection-lib-dir)
#
# dynamicdetection directory /usr/local/lib/snort_dynamicrule/
#
# Load a specific dynamic rule library from the install path
# (same as command line option --dynamic-detection-lib)
#
# dynamicdetection file /usr/local/lib/snort_dynamicrule/libdynamicexamplerule.so
#

##################################################
# Step #3: Configure preprocessors
#
# General configuration for preprocessors is of
# the form
# preprocessor <name_of_processor>: <configuration_options>

# Configure Flow tracking module
# ------------------------------
#
# The Flow tracking module is meant to start unifying the state keeping
# mechanisms of snort into a single place. Right now, only a portscan detector
# is implemented but in the long term,  many of the stateful subsystems of
# snort will be migrated over to becoming flow plugins. This must be enabled
# for flow-portscan to work correctly.
#
# See README.flow for additional information
#
preprocessor flow: stats_interval 0 hash 2

# frag2: IP defragmentation support
# ------------------------------
# This preprocessor performs IP defragmentation.  This plugin will also detect
# people launching fragmentation attacks (usually DoS) against hosts.  No
# arguments loads the default configuration of the preprocessor, which is a 60
# second timeout and a 4MB fragment buffer.

# The following (comma delimited) options are available for frag2
#    timeout [seconds] - sets the number of [seconds] that an unfinished
#                        fragment will be kept around waiting for completion,
#                        if this time expires the fragment will be flushed
#    memcap [bytes] - limit frag2 memory usage to [number] bytes
#                     (default:  4194304)
#
#    min_ttl [number] - minimum ttl to accept
#
#    ttl_limit [number] - difference of ttl to accept without alerting
#                         will cause false positves with router flap
#
# Frag2 uses Generator ID 113 and uses the following SIDS
```

```
# for that GID:
#  SID     Event description
# -----   -------------------
#   1       Oversized fragment (reassembled frag > 64k bytes)
#   2       Teardrop-type attack

#preprocessor frag2

# frag3: Target-based IP defragmentation
# --------------------------------------
#
# Frag3 is a brand new IP defragmentation preprocessor that is capable of
# performing "target-based" processing of IP fragments.  Check out the
# README.frag3 file in the doc directory for more background and configuration
# information.
#
# Frag3 configuration is a two step process, a global initialization phase
# followed by the definition of a set of defragmentation engines.
#
# Global configuration defines the number of fragmented packets that Snort can
# track at the same time and gives you options regarding the memory cap for the
# subsystem or, optionally, allows you to preallocate all the memory for the
# entire frag3 system.
#
# frag3_global options:
#   max_frags: Maximum number of frag trackers that may be active at once.
#              Default value is 8192.
#   memcap: Maximum amount of memory that frag3 may access at any given time.
#           Default value is 4MB.
#   prealloc_frags: Maximum number of individual fragments that may be processed
#                   at once.  This is instead of the memcap system, uses static
#                   allocation to increase performance.  No default value.  Each
#                   preallocated fragment eats ~1550 bytes.
#
# Target-based behavior is attached to an engine as a "policy" for handling
# overlaps and retransmissions as enumerated in the Paxson paper.  There are
# currently five policy types available: "BSD", "BSD-right", "First", "Linux"
# and "Last".  Engines can be bound to standard Snort CIDR blocks or
# IP lists.
#
# frag3_engine options:
#   timeout: Amount of time a fragmented packet may be active before expiring.
#            Default value is 60 seconds.
#   ttl_limit: Limit of delta allowable for TTLs of packets in the fragments.
#              Based on the initial received fragment TTL.
#   min_ttl: Minimum acceptable TTL for a fragment, frags with TTLs below this
#            value will be discarded.  Default value is 0.
#   detect_anomalies: Activates frag3's anomaly detection mechanisms.
#   policy: Target-based policy to assign to this engine.  Default is BSD.
#   bind_to: IP address set to bind this engine to.  Default is all hosts.
#
# Frag3 configuration example:
#preprocessor frag3_global: max_frags 65536 prealloc_frags 262144
#preprocessor frag3_engine: policy linux \
#                           bind_to [10.1.1.12/32,10.1.1.13/32] \
#                           detect_anomalies
#preprocessor frag3_engine: policy first \
#                           bind_to 10.2.1.0/24 \
#                           detect_anomalies
#preprocessor frag3_engine: policy last \
#                           bind_to 10.3.1.0/24
#preprocessor frag3_engine: policy bsd

preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy first detect_anomalies
```

```
# stream4: stateful inspection/stream reassembly for Snort
#--------------------------------------------------------------------
# Use in concert with the -z [all|est] command line switch to defeat stick/snot
# against TCP rules.  Also performs full TCP stream reassembly, stateful
# inspection of TCP streams, etc.  Can statefully detect various portscan
# types, fingerprinting, ECN, etc.

# stateful inspection directive
# no arguments loads the defaults (timeout 30, memcap 8388608)
# options (options are comma delimited):
#   detect_scans - stream4 will detect stealth portscans and generate alerts
#                  when it sees them when this option is set
#   detect_state_problems - detect TCP state problems, this tends to be very
#                           noisy because there are a lot of crappy ip stack
#                           implementations out there
#
#   disable_evasion_alerts - turn off the possibly noisy mitigation of
#                            overlapping sequences.
#
#   ttl_limit [number]     - differential of the initial ttl on a session versus
#                            the normal that someone may be playing games.
#                            Routing flap may cause lots of false positives.
#
#   keepstats [machine|binary] - keep session statistics, add "machine" to
#                           get them in a flat format for machine reading, add
#                           "binary" to get them in a unified binary output
#                           format
#   noinspect - turn off stateful inspection only
#   timeout [number] - set the session timeout counter to [number] seconds,
#                      default is 30 seconds
#   max_sessions [number] - limit the number of sessions stream4 keeps
#                           track of
#   memcap [number] - limit stream4 memory usage to [number] bytes (does
#                     not include session tracking, which is set by the
#                     max_sessions option)
#   log_flushed_streams - if an event is detected on a stream this option will
#                           cause all packets that are stored in the stream4
#                           packet buffers to be flushed to disk.  This only
#                           works when logging in pcap mode!
#   server_inspect_limit [bytes] - Byte limit on server side inspection.
#   enable_udp_sessions - turn on tracking of "sessions" over UDP.  Requires
#                           configure --enable-stream4udp.  UDP sessions are
#                           only created when there is a rule for the sender or
#                           responder that has a flow or flowbits keyword.
#   max_udp_sessions [number] - limit the number of simultaneous UDP sessions
#                               to track
#   udp_ignore_any - Do not inspect UDP packets unless there is a port specific
#                    rule for a given port.  This is a performance improvement
#                    and turns off inspection for udp xxx any -> xxx any rules
#   cache_clean_sessions [number] - Cleanup the session cache by number sessions
#                                   at a time.  The larger the value, the
#                                   more sessions are purged from the cache when
#                                   the session limit or memcap is reached.
#                                   Defaults to 5.
#
#
#
# Stream4 uses Generator ID 111 and uses the following SIDS
# for that GID:
#  SID     Event description
# -----   -------------------
#   1       Stealth activity
#   2       Evasive RST packet
#   3       Evasive TCP packet retransmission
#   4       TCP Window violation
#   5       Data on SYN packet
#   6       Stealth scan: full XMAS
```

```
#   7       Stealth scan: SYN-ACK-PSH-URG
#   8       Stealth scan: FIN scan
#   9       Stealth scan: NULL scan
#   10      Stealth scan: NMAP XMAS scan
#   11      Stealth scan: Vecna scan
#   12      Stealth scan: NMAP fingerprint scan stateful detect
#   13      Stealth scan: SYN-FIN scan
#   14      TCP forward overlap

preprocessor stream4: disable_evasion_alerts

# tcp stream reassembly directive
# no arguments loads the default configuration
#   Only reassemble the client,
#   Only reassemble the default list of ports (See below),
#   Give alerts for "bad" streams
#
# Available options (comma delimited):
#   clientonly - reassemble traffic for the client side of a connection only
#   serveronly - reassemble traffic for the server side of a connection only
#   both - reassemble both sides of a session
#   noalerts - turn off alerts from the stream reassembly stage of stream4
#   ports [list] - use the space separated list of ports in [list], "all"
#                  will turn on reassembly for all ports, "default" will turn
#                  on reassembly for ports 21, 23, 25, 42, 53, 80, 110,
#                  111, 135, 136, 137, 139, 143, 445, 513, 1433, 1521,
#                  and 3306
#   favor_old - favor an old segment (based on sequence number) over a new one.
#               This is the default.
#   favor_new - favor an new segment (based on sequence number) over an old one.
#   overlap_limit [number] - limit on overlaping segments for a session.
#   flush_on_alert - flushes stream when an alert is generated for a session.
#   flush_behavior [mode] -
#           default      - use old static flushpoints (default)
#           large_window - use new larger static flushpoints
#           random       - use random flushpoints defined by flush_base,
#                          flush_seed and flush_range
#   flush_base [number] - lowest allowed random flushpoint (512 by default)
#   flush_range [number] - number is the space within which random flushpoints
#                          are generated (default 1213)
#   flush_seed [number] - seed for the random number generator, defaults to
#                         Snort PID + time
#
# Using the default random flushpoints, the smallest flushpoint is 512,
# and the largest is 1725 bytes.
preprocessor stream4_reassemble

# stream5: Target Based stateful inspection/stream reassembly for Snort
# ---------------------------------------------------------------------
# EXPERIMENTAL CODE!!!
#
# THIS CODE IS STILL EXPERIMENTAL AND MAY OR MAY NOT BE STABLE!
# USE AT YOUR OWN RISK!  DO NOT USE IN PRODUCTION ENVIRONMENTS.
# YOU HAVE BEEN WARNED.
#
# Stream5 is a target-based stream engine for Snort.  Its functionality
# replaces that of Stream4.  Consequently, BOTH Stream4 and Stream5
# cannot be used simultaneously.  Comment out the stream4 configurations
# above to use Stream5.
#
# See README.stream for details on the configuration options.
#
# Example config (that emulates Stream4 with UDP support compiled in)
# preprocessor stream5_global: max_tcp 8192, track_tcp yes, \
#                              track_udp yes
# preprocessor stream5_tcp: policy first, use_static_footprint_sizes
# preprocessor stream5_udp: ignore_any_rules
```

```
# Performance Statistics
# ----------------------
# Documentation for this is provided in the Snort Manual.  You should read it.
# It is included in the release distribution as doc/snort_manual.pdf
#
# preprocessor perfmonitor: time 300 file /var/snort/snort.stats pktcnt 10000

# http_inspect: normalize and detect HTTP traffic and protocol anomalies
#
# lots of options available here. See doc/README.http_inspect.
# unicode.map should be wherever your snort.conf lives, or given
# a full path to where snort can find it.
preprocessor http_inspect: global \
    iis_unicode_map unicode.map 1252

preprocessor http_inspect_server: server default \
    profile all ports { 80 8080 8180 } oversize_dir_length 500

#
#  Example unique server configuration
#
#preprocessor http_inspect_server: server 1.1.1.1 \
#    ports { 80 3128 8080 } \
#    flow_depth 0 \
#    ascii no \
#    double_decode yes \
#    non_rfc_char { 0x00 } \
#    chunk_length 500000 \
#    non_strict \
#    oversize_dir_length 300 \
#    no_alerts


# rpc_decode: normalize RPC traffic
# ---------------------------------
# RPC may be sent in alternate encodings besides the usual 4-byte encoding
# that is used by default. This plugin takes the port numbers that RPC
# services are running on as arguments - it is assumed that the given ports
# are actually running this type of service. If not, change the ports or turn
# it off.
# The RPC decode preprocessor uses generator ID 106
#
# arguments: space separated list
# alert_fragments - alert on any rpc fragmented TCP data
# no_alert_multiple_requests - don't alert when >1 rpc query is in a packet
# no_alert_large_fragments - don't alert when the fragmented
#                            sizes exceed the current packet size
# no_alert_incomplete - don't alert when a single segment
#                       exceeds the current packet size

preprocessor rpc_decode: 111 32771

# bo: Back Orifice detector
# -------------------------
# Detects Back Orifice traffic on the network.
#
# arguments:
#   syntax:
#     preprocessor bo: noalert { client | server | general | snort_attack } \
#                      drop    { client | server | general | snort_attack }
#   example:
#     preprocessor bo: noalert { general server } drop { snort_attack }

#
# The Back Orifice detector uses Generator ID 105 and uses the
# following SIDS for that GID:
#  SID     Event description
```

```
# -----   -------------------
#   1       Back Orifice traffic detected
#   2       Back Orifice Client Traffic Detected
#   3       Back Orifice Server Traffic Detected
#   4       Back Orifice Snort Buffer Attack

preprocessor bo

# telnet_decode: Telnet negotiation string normalizer
# ---------------------------------------------------
# This preprocessor "normalizes" telnet negotiation strings from telnet and ftp
# traffic.  It works in much the same way as the http_decode preprocessor,
# searching for traffic that breaks up the normal data stream of a protocol and
# replacing it with a normalized representation of that traffic so that the
# "content" pattern matching keyword can work without requiring modifications.
# This preprocessor requires no arguments.
#
# DEPRECATED in favor of ftp_telnet dynamic preprocessor
#preprocessor telnet_decode
#
# ftp_telnet: FTP & Telnet normalizer, protocol enforcement and buff overflow
# ---------------------------------------------------------------------------
# This preprocessor normalizes telnet negotiation strings from telnet and
# ftp traffic.  It looks for traffic that breaks the normal data stream
# of the protocol, replacing it with a normalized representation of that
# traffic so that the "content" pattern matching keyword can work without
# requiring modifications.
#
# It also performs protocol correctness checks for the FTP command channel,
# and identifies open FTP data transfers.
#
# FTPTelnet has numerous options available, please read
# README.ftptelnet for help configuring the options for the global
# telnet, ftp server, and ftp client sections for the protocol.

#####
# Per Step #2, set the following to load the ftptelnet preprocessor
# dynamicpreprocessor <full path to libsf_ftptelnet_preproc.so>
# or use commandline option
# --dynamic-preprocessor-lib <full path to libsf_ftptelnet_preproc.so>

preprocessor ftp_telnet: global \
   encrypted_traffic yes \
   inspection_type stateful

preprocessor ftp_telnet_protocol: telnet \
   normalize \
   ayt_attack_thresh 200

# This is consistent with the FTP rules as of 18 Sept 2004.
# CWD can have param length of 200
# MODE has an additional mode of Z (compressed)
# Check for string formats in USER & PASS commands
# Check nDTM commands that set modification time on the file.
preprocessor ftp_telnet_protocol: ftp server default \
   def_max_param_len 100 \
   alt_max_param_len 200 { CWD } \
   cmd_validity MODE < char ASBCZ > \
   cmd_validity MDTM < [ date nnnnnnnnnnnnn[.n[n[n]]] ] string > \
   chk_str_fmt { USER PASS RNFR RNTO SITE MKD } \
   telnet_cmds yes \
   data_chan
   preprocessor ftp_telnet_protocol: ftp client default \
   max_resp_len 256 \
   bounce yes \
   telnet_cmds yes
```

```
# smtp: SMTP normalizer, protocol enforcement and buffer overflow
# ------------------------------------------------------------------------
# This preprocessor normalizes SMTP commands by removing extraneous spaces.
# It looks for overly long command lines, response lines, and data header lines.
# It can alert on invalid commands, or specific valid commands.  It can optionally
# ignore mail data, and can ignore TLS encrypted data.
#
# SMTP has numerous options available, please read README.SMTP for help
# configuring options.

#####
# Per Step #2, set the following to load the smtp preprocessor
# dynamicpreprocessor <full path to libsf_smtp_preproc.so>
# or use commandline option
# --dynamic-preprocessor-lib <full path to libsf_smtp_preproc.so>

preprocessor smtp: \
  ports { 25 } \
  inspection_type stateful \
  normalize cmds \
  normalize_cmds { EXPN VRFY RCPT } \
  alt_max_command_line_len 260 { MAIL } \
  alt_max_command_line_len 300 { RCPT } \
  alt_max_command_line_len 500 { HELP HELO ETRN } \
  alt_max_command_line_len 255 { EXPN VRFY }

# sfPortscan
# ----------
# Portscan detection module.  Detects various types of portscans and
# portsweeps.  For more information on detection philosophy, alert types,
# and detailed portscan information, please refer to the README.sfportscan.
#
# -configuration options-
#     proto { tcp udp icmp ip all }
#       The arguments to the proto option are the types of protocol scans that
#       the user wants to detect.  Arguments should be separated by spaces and
#       not commas.
#     scan_type { portscan portsweep decoy_portscan distributed_portscan all }
#       The arguments to the scan_type option are the scan types that the
#       user wants to detect.  Arguments should be separated by spaces and not
#       commas.
#     sense_level { low|medium|high }
#       There is only one argument to this option and it is the level of
#       sensitivity in which to detect portscans.  The 'low' sensitivity
#       detects scans by the common method of looking for response errors, such
#       as TCP RSTs or ICMP unreachables.  This level requires the least
#       tuning.  The 'medium' sensitivity level detects portscans and
#       filtered portscans (portscans that receive no response).  This
#       sensitivity level usually requires tuning out scan events from NATed
#       IPs, DNS cache servers, etc.  The 'high' sensitivity level has
#       lower thresholds for portscan detection and a longer time window than
#       the 'medium' sensitivity level.  Requires more tuning and may be noisy
#       on very active networks.  However, this sensitivity levels catches the
#       most scans.
#     memcap { positive integer }
#       The maximum number of bytes to allocate for portscan detection.  The
#       higher this number the more nodes that can be tracked.
#     logfile { filename }
#       This option specifies the file to log portscan and detailed portscan
#       values to.  If there is not a leading /, then snort logs to the
#       configured log directory.  Refer to README.sfportscan for details on
#       the logged values in the logfile.
#     watch_ip { Snort IP List }
#     ignore_scanners { Snort IP List }
#     ignore_scanned { Snort IP List }
#       These options take a snort IP list as the argument.  The 'watch_ip'
#       option specifies the IP(s) to watch for portscan.  The
```

```
#       'ignore_scanners' option specifies the IP(s) to ignore as scanners.
#       Note that these hosts are still watched as scanned hosts.  The
#       'ignore_scanners' option is used to tune alerts from very active
#       hosts such as NAT, nessus hosts, etc.  The 'ignore_scanned' option
#       specifies the IP(s) to ignore as scanned hosts.  Note that these hosts
#       are still watched as scanner hosts.  The 'ignore_scanned' option is
#       used to tune alerts from very active hosts such as syslog servers, etc.
#     detect_ack_scans
#       This option will include sessions picked up in midstream by the stream
#       module, which is necessary to detect ACK scans.  However, this can lead to
#       false alerts, especially under heavy load with dropped packets; which is why
#       the option is off by default.
#
preprocessor sfportscan: proto  { all } \
                    memcap { 10000000 } \
                    sense_level { low }

# arpspoof
#----------------------------------------
# Experimental ARP detection code from Jeff Nathan, detects ARP attacks,
# unicast ARP requests, and specific ARP mapping monitoring.  To make use of
# this preprocessor you must specify the IP and hardware address of hosts on
# the same layer 2 segment as you.  Specify one host IP MAC combo per line.
# Also takes a "-unicast" option to turn on unicast ARP request detection.
# Arpspoof uses Generator ID 112 and uses the following SIDS for that GID:

#  SID     Event description
# -----   -------------------
#   1       Unicast ARP request
#   2       Etherframe ARP mismatch (src)
#   3       Etherframe ARP mismatch (dst)
#   4       ARP cache overwrite attack

#preprocessor arpspoof
#preprocessor arpspoof_detect_host: 192.168.40.1 f0:0f:00:f0:0f:00

# ssh
#----------------------------------------
# EXPERIMENTAL CODE!!!
#
# THIS CODE IS STILL EXPERIMENTAL AND MAY OR MAY NOT BE STABLE!
# USE AT YOUR OWN RISK!  DO NOT USE IN PRODUCTION ENVIRONMENTS.
# YOU HAVE BEEN WARNED.
#
# The SSH preprocessor detects the following exploits: Gobbles, CRC 32,
# Secure CRT, and the Protocol Mismatch exploit.
#
# Both Gobbles and CRC 32 attacks occur after the key exchange, and are
# therefore encrypted.  Both attacks involve sending a large payload
# (20kb+) to the server immediately after the authentication challenge.
# To detect the attacks, the SSH preprocessor counts the number of bytes
# transmitted to the server.  If those bytes exceed a pre-defined limit
# within a pre-define number of packets, an alert is generated.  Since
# Gobbles only effects SSHv2 and CRC 32 only effects SSHv1, the SSH
# version string exchange is used to distinguish the attacks.
#
# The Secure CRT and protocol mismatch exploits are observable before
# the key exchange.
#
# SSH has numerous options available, please read README.ssh for help
# configuring options.

#####
# Per Step #2, set the following to load the ssh preprocessor
# dynamicpreprocessor <full path to libsf_ssh_preproc.so>
# or use commandline option
# --dynamic-preprocessor-lib <full path to libsf_ssh_preproc.so>
```

```
#
#preprocessor ssh: server_ports { 22 } \
#                 max_client_bytes 19600 \
#                 max_encrypted_packets 20

# DCE/RPC
#----------------------------------------
#
# The dcerpc preprocessor detects and decodes SMB and DCE/RPC traffic.
# It is primarily interested in DCE/RPC data, and only decodes SMB
# to get at the DCE/RPC data carried by the SMB layer.
#
# Currently, the preprocessor only handles reassembly of fragmentation
# at both the SMB and DCE/RPC layer.  Snort rules can be evaded by
# using both types of fragmentation; with the preprocessor enabled
# the rules are given a buffer with a reassembled SMB or DCE/RPC
# packet to examine.
#
# At the SMB layer, only fragmentation using WriteAndX is currently
# reassembled.  Other methods will be handled in future versions of
# the preprocessor.
#
# Autodetection of SMB is done by looking for "\xFFSMB" at the start of
# the SMB data, as well as checking the NetBIOS header (which is always
# present for SMB) for the type "SMB Session".
#
# Autodetection of DCE/RPC is not as reliable.  Currently, two bytes are
# checked in the packet.  Assuming that the data is a DCE/RPC header,
# one byte is checked for DCE/RPC version (5) and another for the type
# "DCE/RPC Request".  If both match, the preprocessor proceeds with that
# assumption that it is looking at DCE/RPC data.  If subsequent checks
# are nonsensical, it ends processing.
#
# DCERPC has numerous options available, please read README.dcerpc for help
# configuring options.

#####
# Per Step #2, set the following to load the dcerpc preprocessor
# dynamicpreprocessor <full path to libsf_dcerpc_preproc.so>
# or use commandline option
# --dynamic-preprocessor-lib <full path to libsf_dcerpc_preproc.so>
preprocessor dcerpc: \
    autodetect \
    max_frag_size 3000 \
    memcap 100000

# DNS
#----------------------------------------
# The dns preprocessor (currently) decodes DNS Response traffic
# and detects a few vulnerabilities.
#
# DNS has a few options available, please read README.dns for
# help configuring options.

#####
# Per Step #2, set the following to load the dns preprocessor
# dynamicpreprocessor <full path to libsf_dns_preproc.so>
# or use commandline option
# --dynamic-preprocessor-lib <full path to libsf_dns_preproc.so>

preprocessor dns: \
    ports { 53 } \
    enable_rdata_overflow

##################################################################
# Step #4: Configure output plugins
#
```

```
# Uncomment and configure the output plugins you decide to use.  General
# configuration for output plugins is of the form:
#
# output <name_of_plugin>: <configuration_options>
#
# alert_syslog: log alerts to syslog
# ----------------------------------
# Use one or more syslog facilities as arguments.  Win32 can also optionally
# specify a particular hostname/port.  Under Win32, the default hostname is
# '127.0.0.1', and the default port is 514.
#
# [Unix flavours should use this format...]
# output alert_syslog: LOG_AUTH LOG_ALERT
#
# [Win32 can use any of these formats...]
# output alert_syslog: LOG_AUTH LOG_ALERT
# output alert_syslog: host=hostname, LOG_AUTH LOG_ALERT
# output alert_syslog: host=hostname:port, LOG_AUTH LOG_ALERT

# log_tcpdump: log packets in binary tcpdump format
# -------------------------------------------------
# The only argument is the output file name.
#
# output log_tcpdump: tcpdump.log

# database: log to a variety of databases
# ---------------------------------------
# See the README.database file for more information about configuring
# and using this plugin.
#
# output database: log, mysql, user=root password=test dbname=db host=localhost
# output database: alert, postgresql, user=snort dbname=snort
# output database: log, odbc, user=snort dbname=snort
# output database: log, mssql, dbname=snort user=snort password=test
# output database: log, oracle, dbname=snort user=snort password=test

# unified: Snort unified binary format alerting and logging
# -------------------------------------------------------------------
# The unified output plugin provides two new formats for logging and generating
# alerts from Snort, the "unified" format.  The unified format is a straight
# binary format for logging data out of Snort that is designed to be fast and
# efficient.  Used with barnyard (the new alert/log processor), most of the
# overhead for logging and alerting to various slow storage mechanisms such as
# databases or the network can now be avoided.
#
# Check out the spo_unified.h file for the data formats.
#
# Two arguments are supported.
#    filename - base filename to write to (current time_t is appended)
#    limit    - maximum size of spool file in MB (default: 128)
#
# output alert_unified: filename snort.alert, limit 128
# output log_unified: filename snort.log, limit 128


# prelude: log to the Prelude Hybrid IDS system
# ---------------------------------------------
#
# profile = Name of the Prelude profile to use (default is snort).
#
# Snort priority to IDMEF severity mappings:
# high < medium < low < info
#
# These are the default mapped from classification.config:
# info   = 4
# low    = 3
# medium = 2
```

```
# high   = anything below medium
#
# output alert_prelude
# output alert_prelude: profile=snort-profile-name


# You can optionally define new rule types and associate one or more output
# plugins specifically to that type.
#
# This example will create a type that will log to just tcpdump.
# ruletype suspicious
# {
#   type log
#   output log_tcpdump: suspicious.log
# }
#
# EXAMPLE RULE FOR SUSPICIOUS RULETYPE:
# suspicious tcp $HOME_NET any -> $HOME_NET 6667 (msg:"Internal IRC Server";)
#
# This example will create a rule type that will log to syslog and a mysql
# database:
# ruletype redalert
# {
#   type alert
#   output alert_syslog: LOG_AUTH LOG_ALERT
#   output database: log, mysql, user=snort dbname=snort host=localhost
# }
#
# EXAMPLE RULE FOR REDALERT RULETYPE:
# redalert tcp $HOME_NET any -> $EXTERNAL_NET 31337 \
#   (msg:"Someone is being LEET"; flags:A+;)

#
# Include classification & priority settings
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\etc\classification.config
#

include classification.config

#
# Include reference systems
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\etc\reference.config
#

include reference.config

##################################################################
# Step #5: Configure snort with config statements
#
# See the snort manual for a full set of configuration references
#
# config flowbits_size: 64
#
# New global ignore_ports config option from Andy Mullican
#
# config ignore_ports: <tcp|udp> <list of ports separated by whitespace>
# config ignore_ports: tcp 21 6667:6671 1356
# config ignore_ports: udp 1:17 53


##################################################################
# Step #6: Customize your rule set
#
# Up to date snort rules are available at http://www.snort.org
#
```

```
# The snort web site has documentation about how to write your own custom snort
# rules.

#========================================
# Include all relevant rulesets here
#
# The following rulesets are disabled by default:
#
#   web-attacks, backdoor, shellcode, policy, porn, info, icmp-info, virus,
#   chat, multimedia, and p2p
#
# These rules are either site policy specific or require tuning in order to not
# generate false positive alerts in most enviornments.
#
# Please read the specific include file for more information and
# README.alert_order for how rule ordering affects how alerts are triggered.
#========================================

include $RULE_PATH/local.rules
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/scan.rules
include $RULE_PATH/finger.rules
include $RULE_PATH/ftp.rules
include $RULE_PATH/telnet.rules
include $RULE_PATH/rpc.rules
include $RULE_PATH/rservices.rules
include $RULE_PATH/dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/tftp.rules

include $RULE_PATH/web-cgi.rules
include $RULE_PATH/web-coldfusion.rules
include $RULE_PATH/web-iis.rules
include $RULE_PATH/web-frontpage.rules
include $RULE_PATH/web-misc.rules
include $RULE_PATH/web-client.rules
include $RULE_PATH/web-php.rules

include $RULE_PATH/sql.rules
include $RULE_PATH/x11.rules
include $RULE_PATH/icmp.rules
include $RULE_PATH/netbios.rules
include $RULE_PATH/misc.rules
include $RULE_PATH/attack-responses.rules
include $RULE_PATH/oracle.rules
include $RULE_PATH/mysql.rules
include $RULE_PATH/snmp.rules

include $RULE_PATH/smtp.rules
include $RULE_PATH/imap.rules
include $RULE_PATH/pop2.rules
include $RULE_PATH/pop3.rules

include $RULE_PATH/nntp.rules
include $RULE_PATH/other-ids.rules
# include $RULE_PATH/web-attacks.rules
# include $RULE_PATH/backdoor.rules
# include $RULE_PATH/shellcode.rules
# include $RULE_PATH/policy.rules
# include $RULE_PATH/porn.rules
# include $RULE_PATH/info.rules
# include $RULE_PATH/icmp-info.rules
# include $RULE_PATH/virus.rules
# include $RULE_PATH/chat.rules
# include $RULE_PATH/multimedia.rules
```

```
# include $RULE_PATH/p2p.rules
# include $RULE_PATH/spyware-put.rules
include $RULE_PATH/experimental.rules

# Include any thresholding or suppression commands. See threshold.conf in the
# <snort src>/etc directory for details. Commands don't necessarily need to be
# contained in this conf, but a separate conf makes it easier to maintain them.
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\etc\threshold.conf
# Uncomment if needed.
      # include threshold.conf
```

ANEXO C

Manual de Sebek

1. INSTALACIÓN DE SEBEK EN PLATAFORMAS WINDOWS.

Iniciar el programa de instalación de Sebek, esto le permitirá instalar el núcleo conductor en el sistema cliente.

Figura 1. Pantalla de inicio de instalación de Sebek. *Clic en* **Next**.

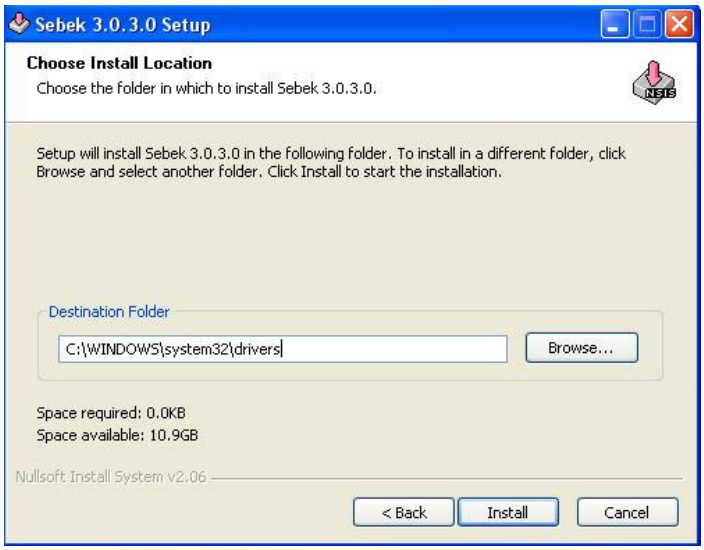


Figura 2. Directorio donde se instalará Sebek. *Clic en* **Install**.

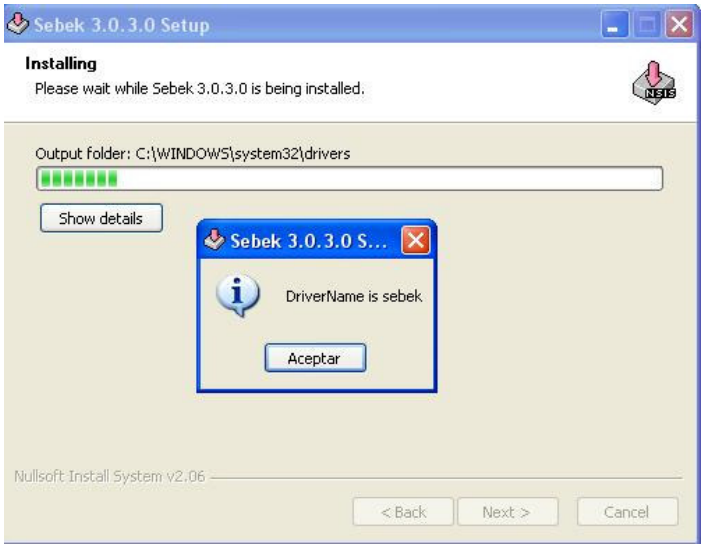


Figura 3. DriverName. *Clic en* **Aceptar**.

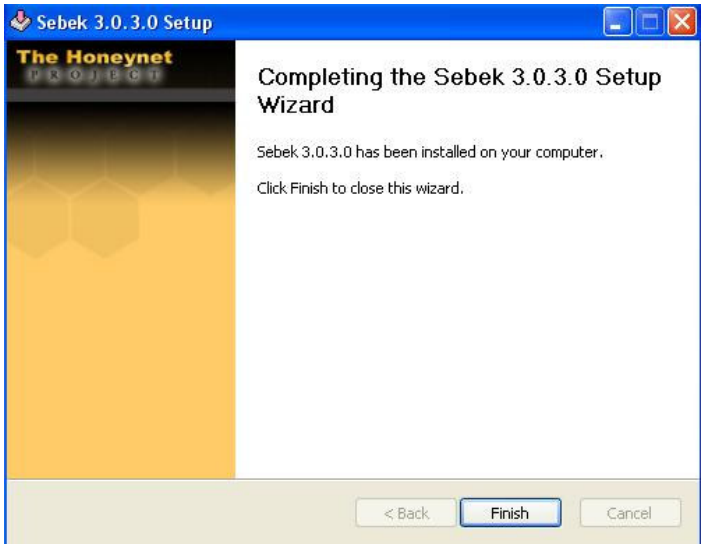


Figura 4. Instalación completa de Sebek.
Clic en **Finish**.

2. CONFIGURACIÓN DE LAS VARIABLES.

Ahora hay que configurar Sebek antes de reiniciar el equipo de lo contrario no funcionará correctamente.

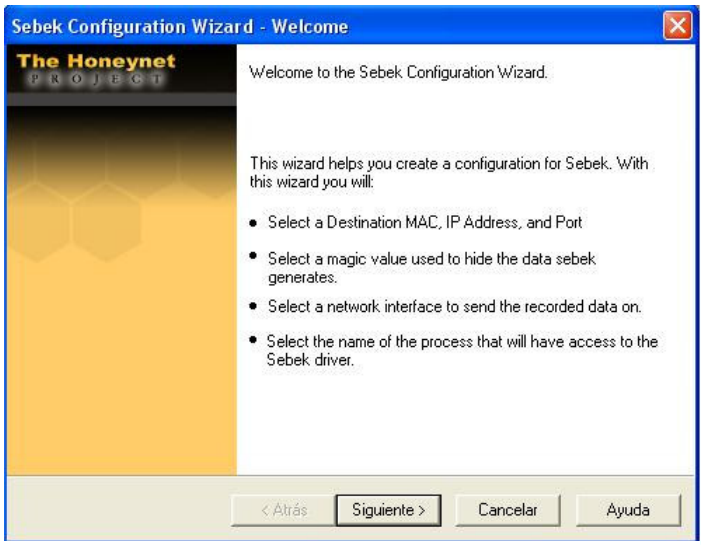


Figura 5. Pantalla de inicio de configuración de Sebek.
Clic en **Siguiente**.

2.1 Elegir la ubicación del driver Sebek, por defecto se ubica en la ruta:
c:\WINDOWS\system32\drivers\SEBEK.SYS

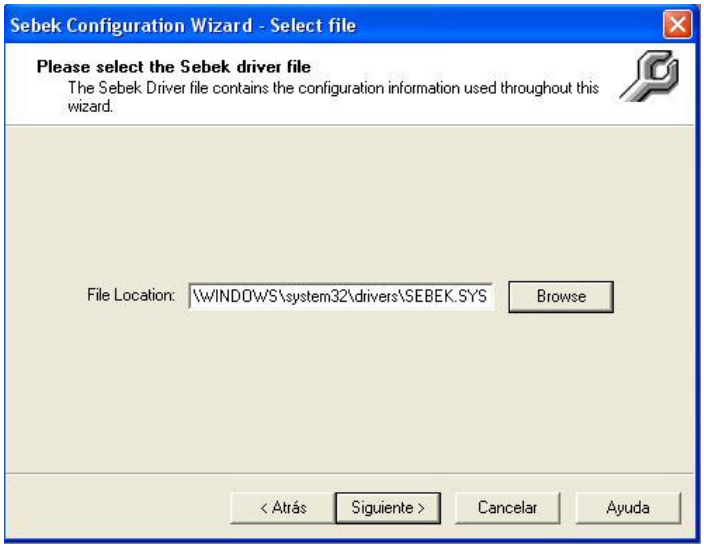


Figura 6. Ubicación del driver Sebek. *Clic en* **Siguiente**.

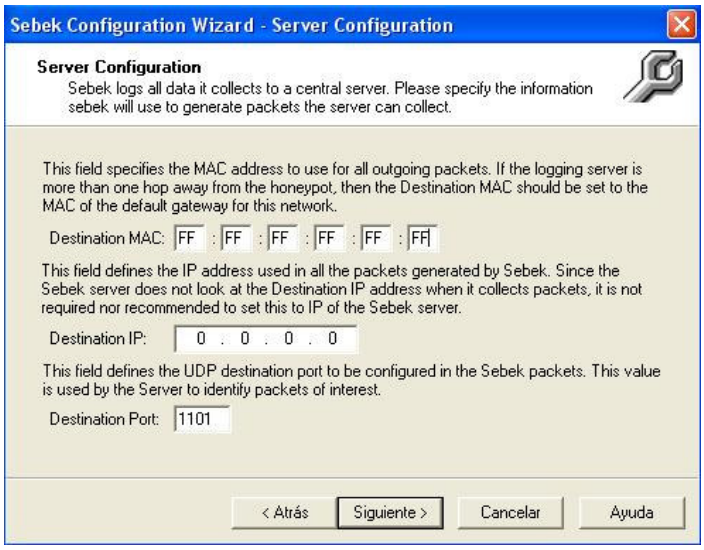


Figura 7. Configuración de puerto y direcciones de destino – MAC, IP –. *Clic en* **Siguiente**.

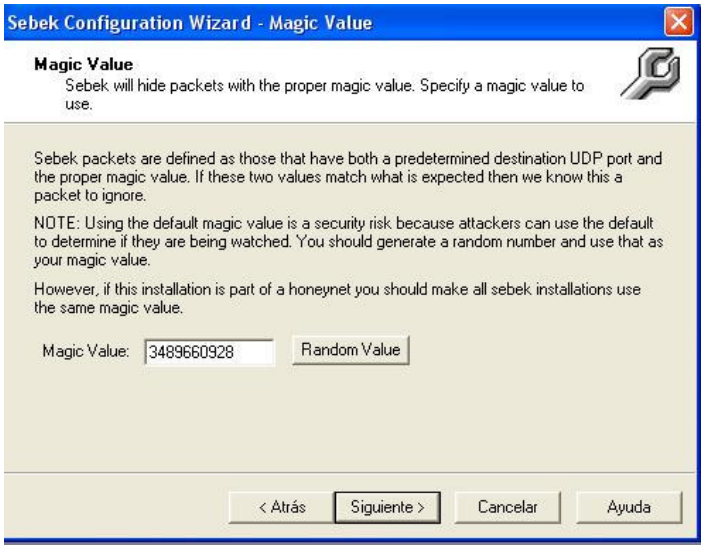


Figura 8. Valor mágico.
Obtenerlo con **Random Value**, *clic en* **Siguiente**.

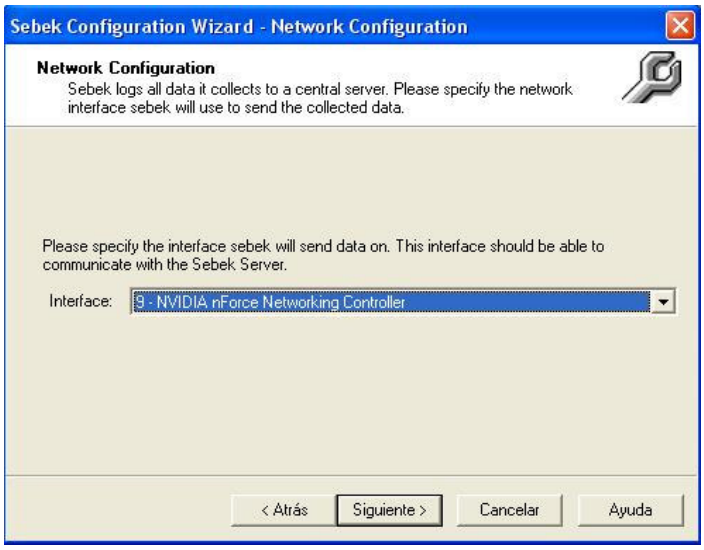


Figura 9. Configuración de la red.
Seleccione la interface, *clic en* **Siguiente**.

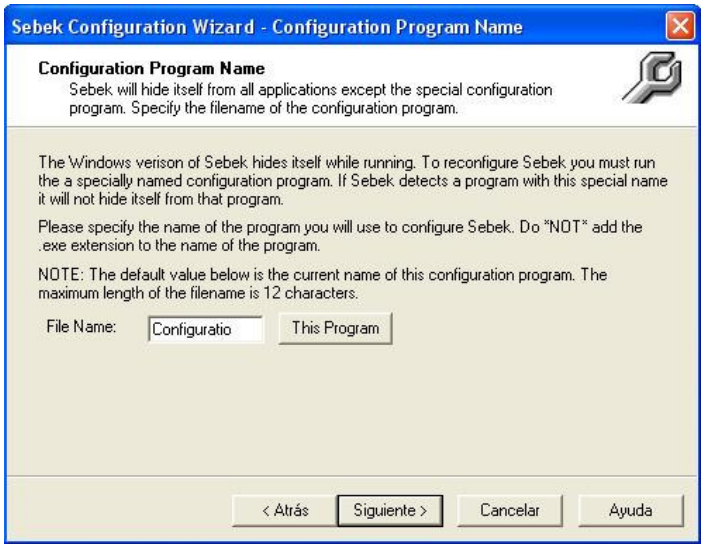


Figura 10. Nombre del programa de Configuración. *Clic en* **Siguiente**.

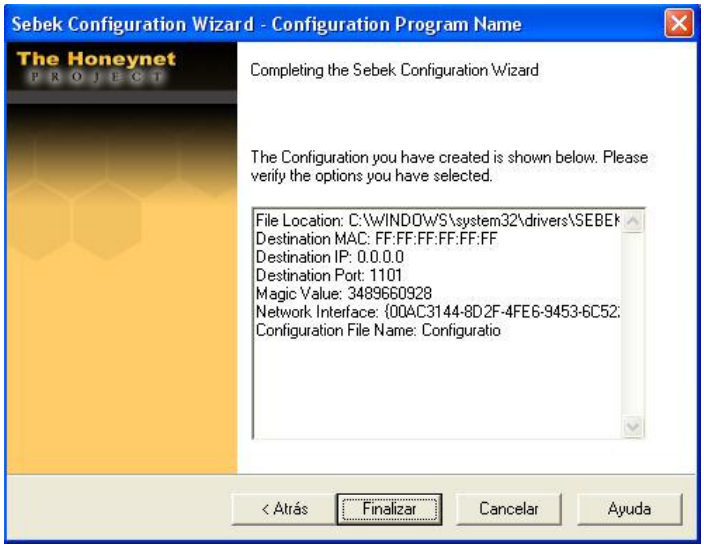


Figura 11. Verificación de datos ingresados. *Clic en* **Finalizar**.

3. DESINSTALACIÓN DE SEBEK.

Debido a que el instalador no se registra en el sistema se deberá desinstalar manualmente.

Los siguientes pasos son necesarios para eliminar Sebek de un PC:

1. Inicie el equipo con el CD de instalación del sistema operativo.
2. Seleccione la consola de recuperación pulsando "R".
3. Si está ejecutando XP puede ignorar este paso. Si está ejecutando en Windows 2000, después presione ***C*** para abrir la consola de recuperación.
4. Seleccione la instalación de Windows que desea modificar.
5. Proporcionar la contraseña de administrador.
6. Una vez ya en el símbolo del sistema, ir a ***C:\WINDOWS\System32\drivers*** y escriba *'delete Sebek*" sin las comillas.
7. Una vez que el comando se completa reiniciar la máquina escribiendo ***Exit***.
8. Elimine Sebek situado en la clave del Registro

   HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Sebek
9. Y finalmente quite el programa de configuración si está en la máquina.

ANEXO D

Manual de ROO

1. INTRODUCCIÓN.

El propósito de este manual es dar a conocer al Administrador del Proyecto Honeynet, las características técnicas de instalación y configuración del Honeywall CDROM Roo-1.4.hw-2009.

Se espera que el manual sirva de apoyo para modificar valores y parámetros de las variables existentes cuando sea necesario.

2. INSTALACIÓN DEL HONEYWALL.

2.1 Cambiar la secuencia de arranque del computador para que inicie con el CD.

2.2 Inserte el CD del Honeywall.

2.3 Presione Enter.

Figura 1. Pantalla de inicio de instalación del Honeywall.

2.4 Seguidamente se inicia con el formateo y copiado de archivos necesarios.

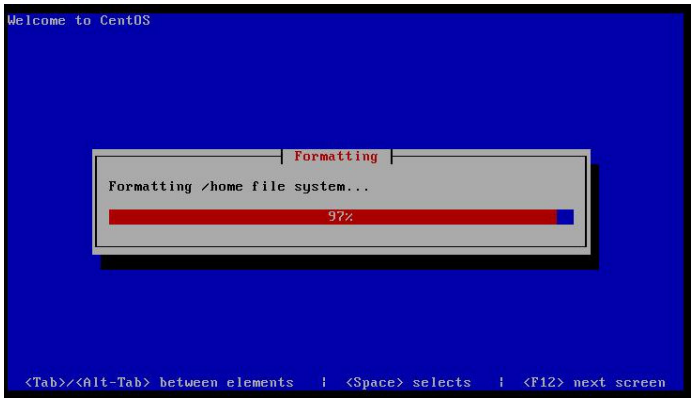


Figura 2. Progreso de instalación - Formateo

Figura 3. Progreso de instalación – Transferencia de archivos.

2.5 Inicio del proceso de instalación.

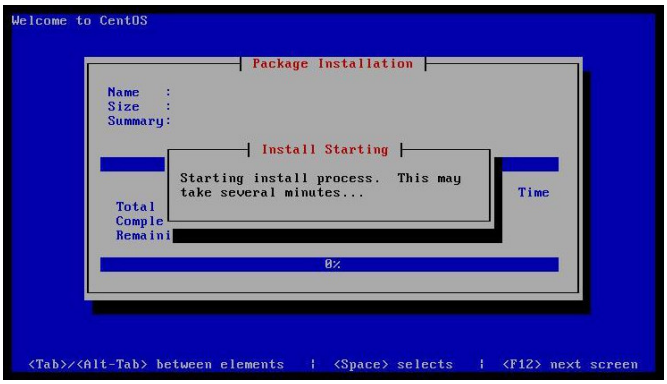


Figura 4. Inicio de instalación.

2.6 Luego de finalizar la instalación retire el CD y presione Enter.

3. ACCESO AL SISTEMA.

Para ingresar a la administración como usuario, ***root***, obligadamente se debe iniciar antes como usuario ***roo***, la contraseña para estos usuarios es ***honey*** por defecto.

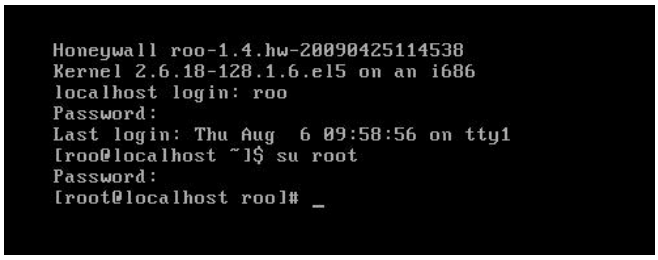


Figura 5. Accediendo al sistema

4. CONFIGURACIÓN DE LAS VARIABLES.

4.1 Iniciar la configuración ingresando al directorio ***dlg*** y ejecutando la aplicación ***./dialogmenu.sh***.

4.2 En el menú principal, seleccionar la opción 4: Honeywall Configuration, para iniciar las configuraciones.

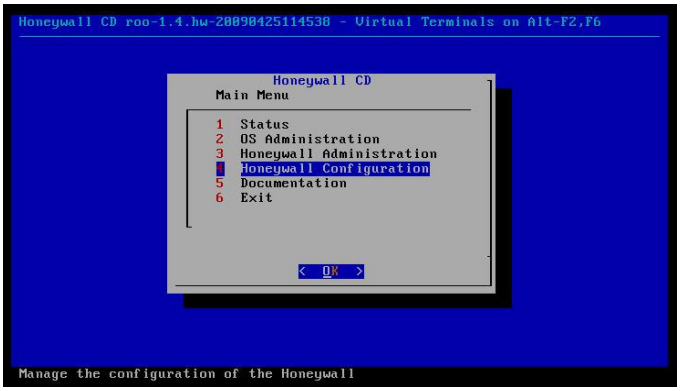


Figura 6. Menú principal: *Configuración Honeywall*

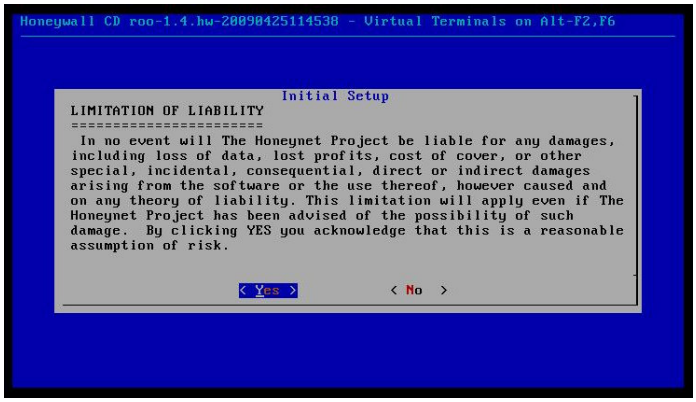


Figura 7. Configuración inicial.

4.3 En el método de configuración inicial se tienen tres opciones:

FLOPPY: El Honeywall permite cargar configuraciones existentes almacenadas en un disquete.

DEFAULTS: Se cargan las configuraciones por defecto.

INTERVIEW: Elegir si se va a configurar por primera vez.

Figura 8. Método de configuración inicial.

4.4 Se recomienda leer el documento *"Know your enemy: Honeynets".*

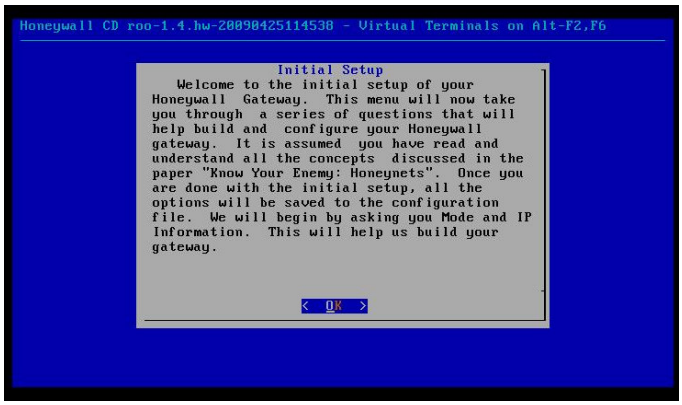


Figura 9. Mensaje de inicio de configuración inicial.

4.5 Ingrese las direcciones IP's de todos los Honeypots, separadas por un espacio.

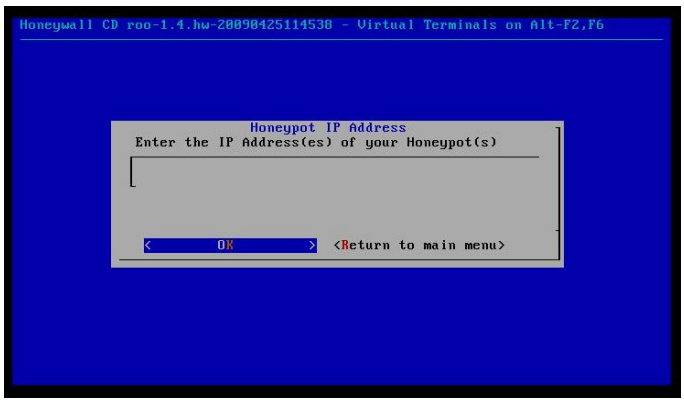


Figura 10. Ingreso de direcciones IP.

4.6 Ingrese la Red que se va a utilizar en la Honeynet.

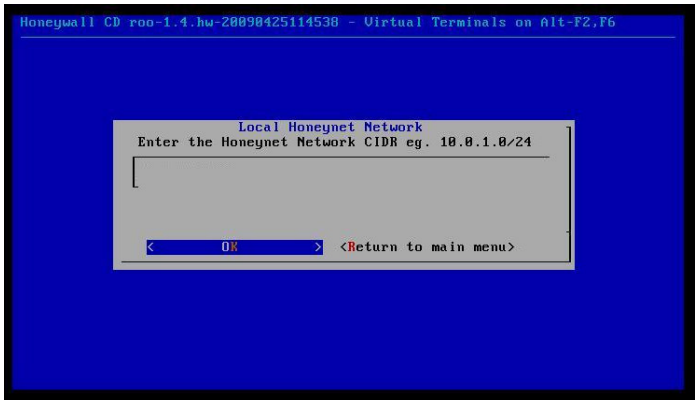


Figura 11. Ingreso de la red.

4.7 Ingrese la dirección de Broadcast correspondiente a la red de la Honeynet.

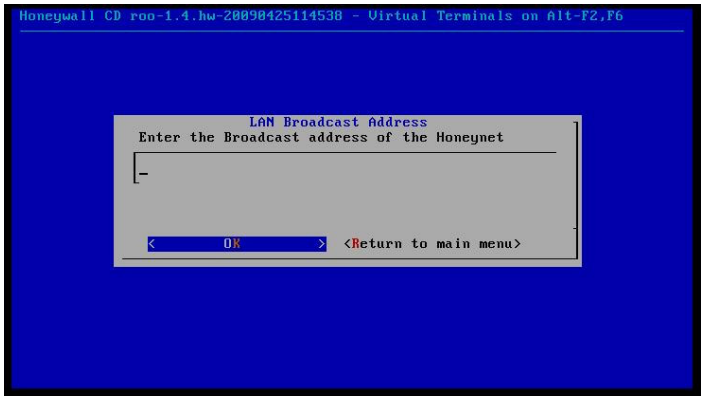


Figura 12. Ingreso de la dirección de Broadcast.

4.8 Configuración de la interfaz de administración remota.

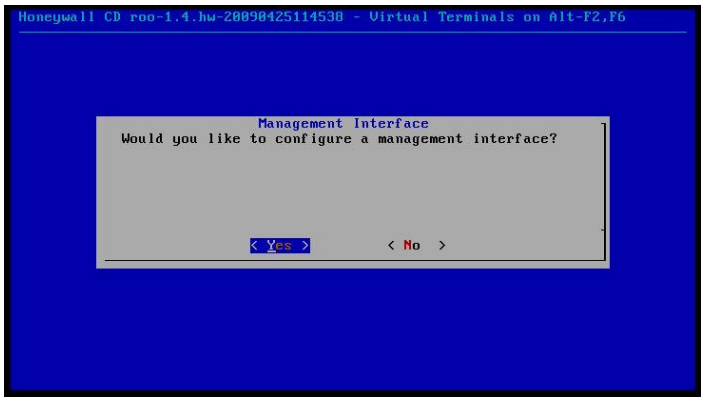


Figura 13. Configuración de interfaz de administración.

4.9 Ingresar el nombre de la interfaz, por defecto se elige la ***eth2***.

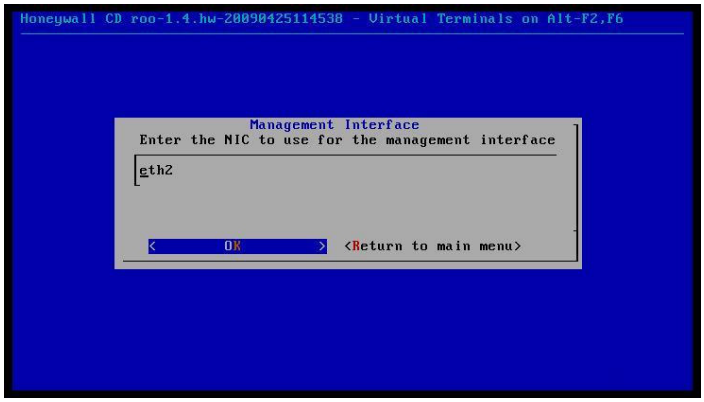


Figura 14. Ingreso del nombre de la interfaz.

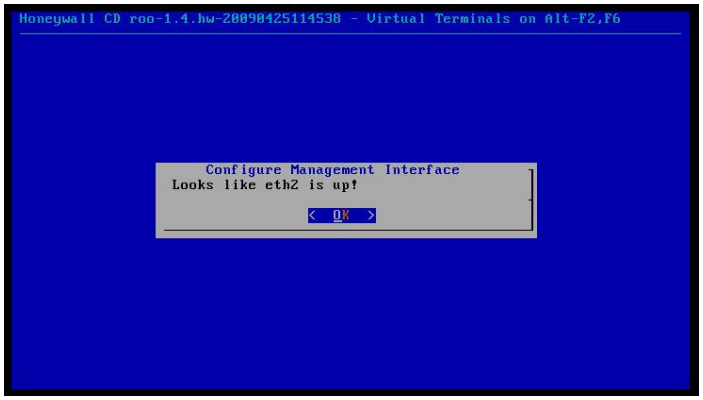


Figura 15. Confirmación de interfaz.

4.10 Ingrese la dirección IP para la interfaz de administración, esta dirección se utiliza para conectarse remotamente y tener acceso a la interfaz web que servirá para monitorear la activad de la Honeynet.

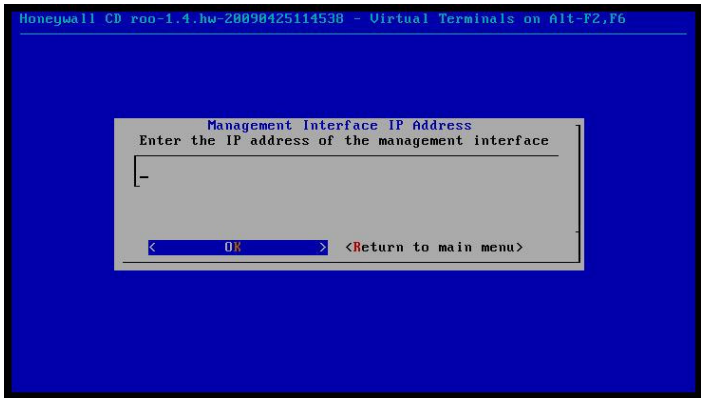


Figura 16. Ingreso de dirección IP – Interfaz de administración.

4.11 Ingrese la máscara de red para la interfaz de administración.

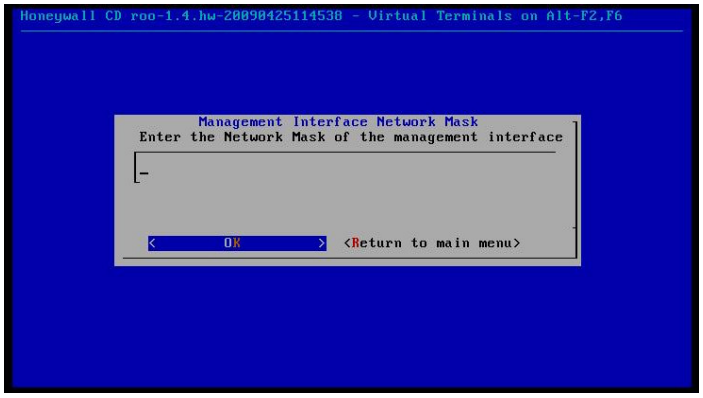


Figura 17. Ingreso de máscara de red – Interfaz de administración.

4.12 Ingrese la dirección Gateway para la interfaz de administración.

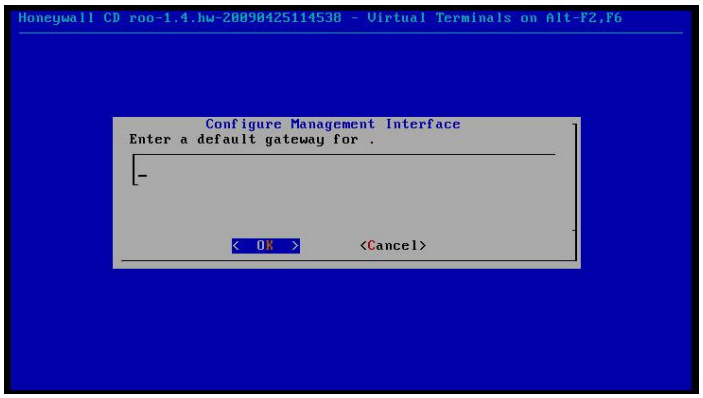


Figura 18. Ingreso de Gateway – Interfaz de administración.

4.13 Elija un nombre para su sistema.

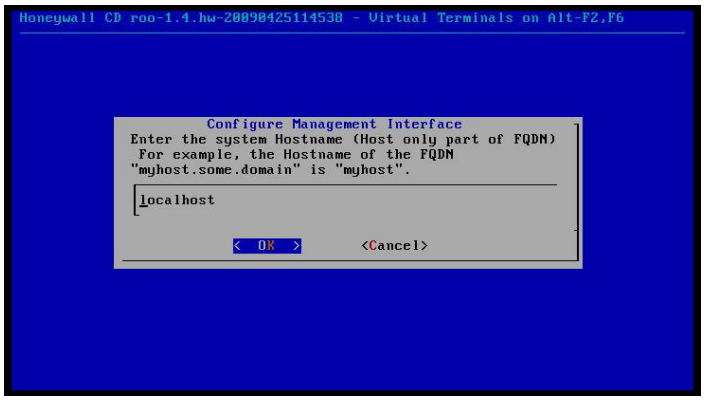


Figura 19. Ingreso del *hostname* – Interfaz de administración.

4.14 Ingrese la dirección del servidor DNS; si posee más de una dirección debe escribirlos separados por un espacio.

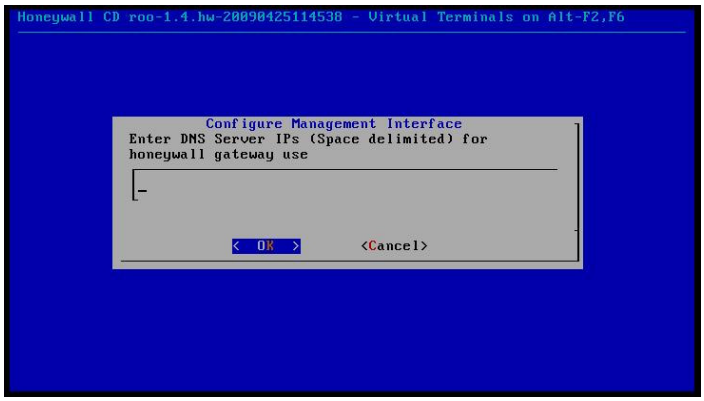


Figura 20. Ingreso de IP(s) del servidor DNS – Interfaz de administración.

4.15 Confirme la activación de la interface de administración.

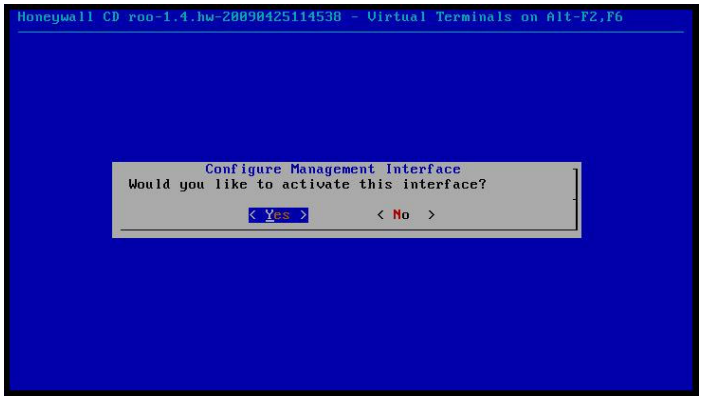


Figura 21. Confirmación de interface – Interfaz de administración.

5. CONFIGURACIÓN SSH.

5.1 Confirme el mensaje de que desea configurar SSH.

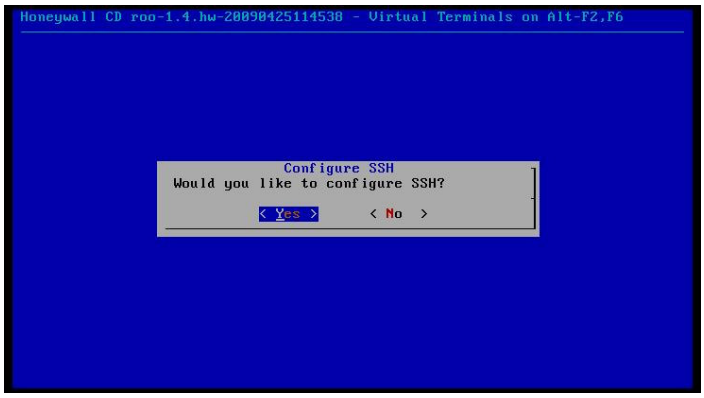


Figura 22. Confirmación para la configuración de SSH.

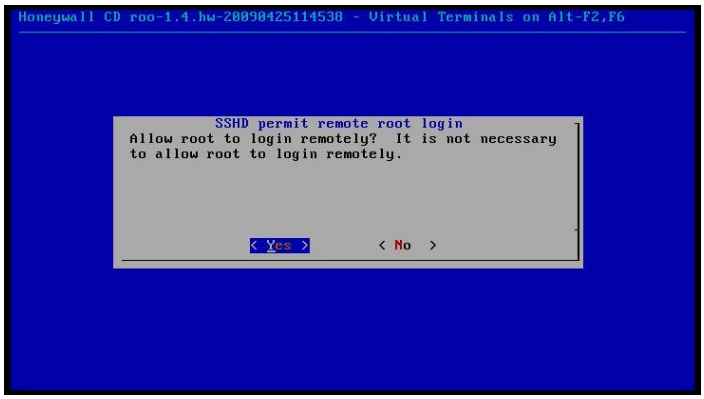


Figura 23. Confirmación para permiso remoto.

6. CAMBIAR EL PASSWORD DE LOS USUARIOS.

6.1 Es necesario cambiar las contraseñas que trae por defecto el sistema Honeywall, el administrador debe elegir una contraseña más segura.

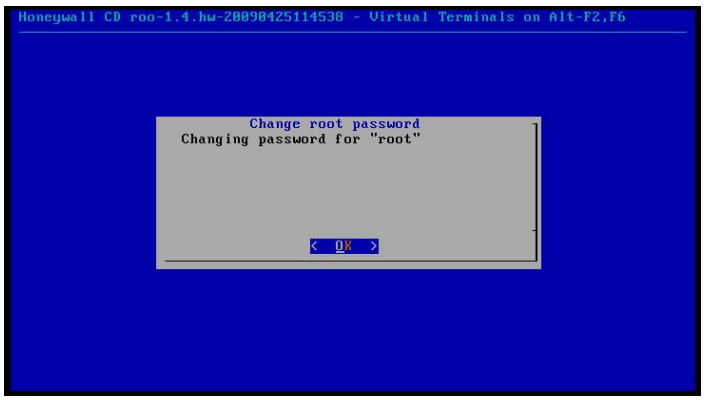


Figura 24. Confirmación de cambio de password.

Figura 25. Ingreso de nuevo password.

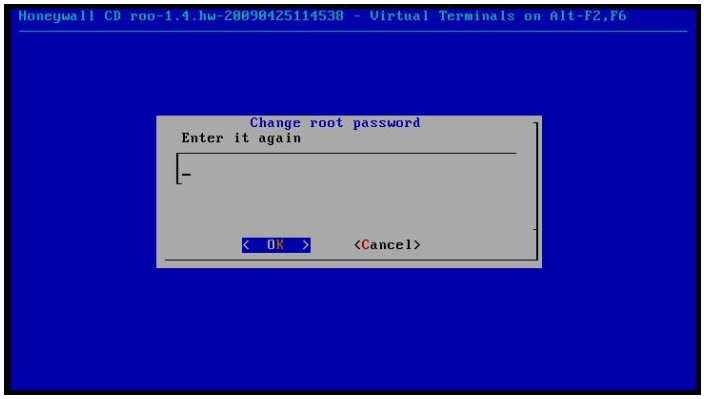


Figura 26. Confirme el password ingresado.

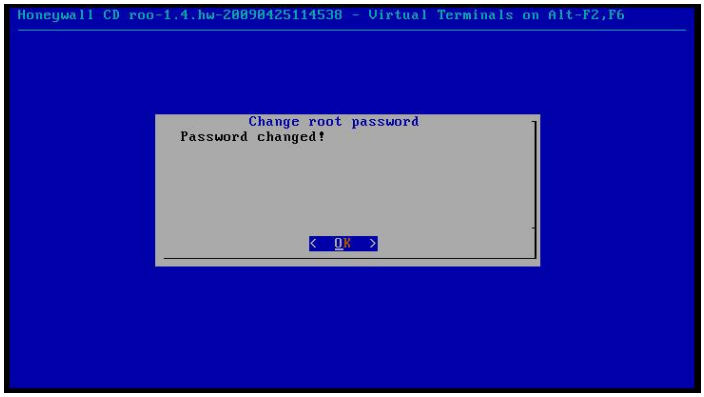


Figura 27. Confirmación de password cambiado.

6.2 Ingrese una lista de puertos TCP permitidos para la interfaz de administración, por defecto está incluido SSH.

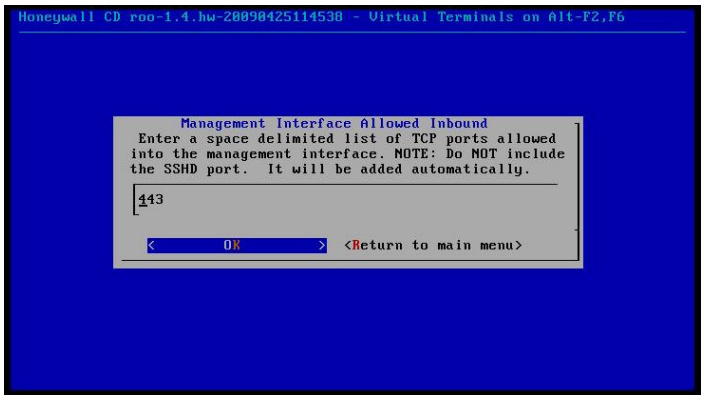


Figura 28. Ingreso de puertos TCP.

6.3 Ingrese el rango de direcciones IP que pueden tener acceso a la administración.

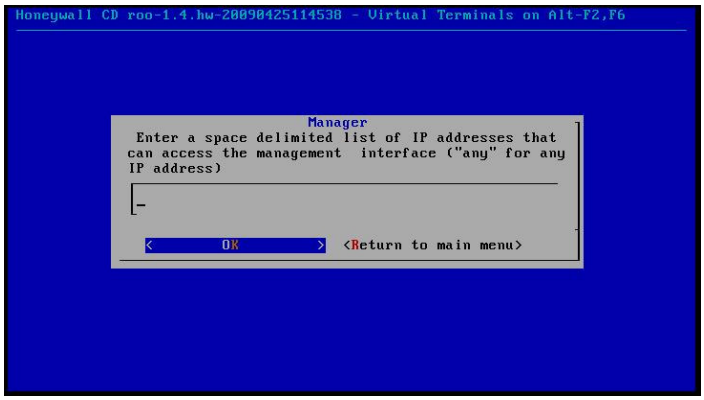


Figura 29. Ingreso de rango de direcciones IP.

6.4 Elija ***yes*** para habilitar la interfaz web de administración.

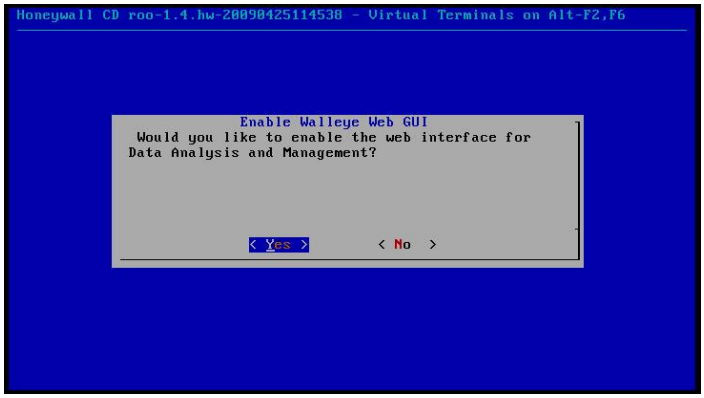


Figura 30. Confirmación para habilitar interfaz web.

6.5 Confirme las restricciones al firewall.

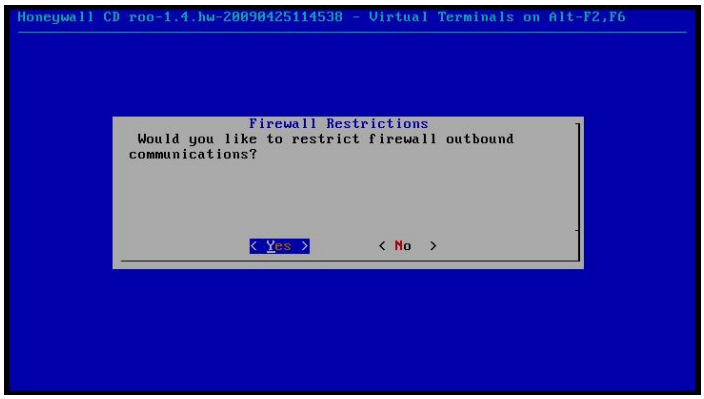


Figura 31. Confirmación para restricciones de firewall.

6.6 Ingrese la lista de puertos TCP necesarios de salida.

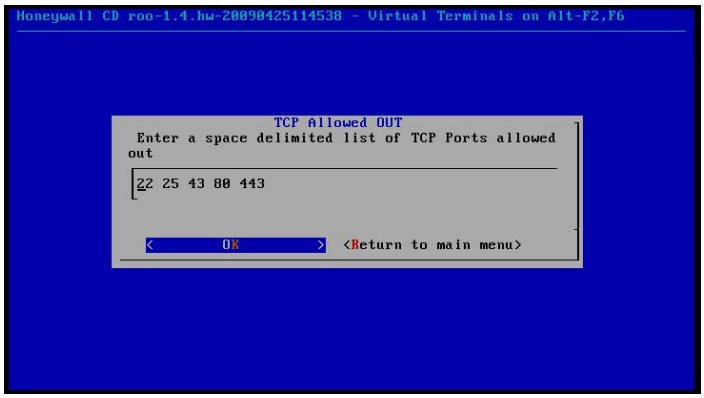


Figura 32. Ingreso de puertos permitidos TCP para salida.

6.7 Ingrese la lista de puertos UDP necesarios de salida.

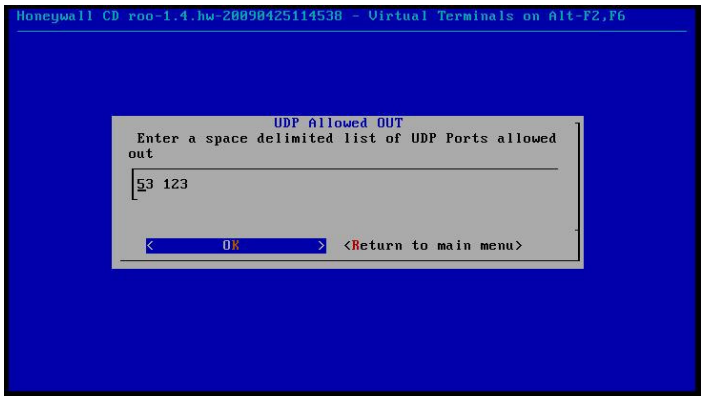


Figura 33. Ingreso de puertos permitidos UDP para salida.

6.8 Ha finalizado la 2da sección de la configuración del sistema Honeywall.

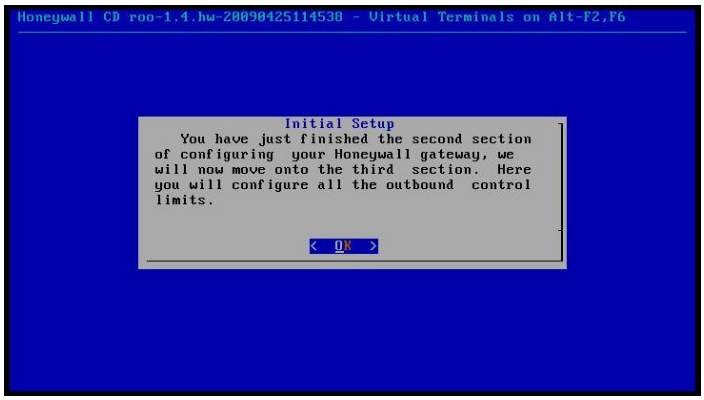


Figura 34. Confirmación de finalizar la 2da sección de configuraciones.

7. CONFIGURACIÓN DEL LÍMITE DE CONEXIONES PERMITIDAS.

7.1 Se especifica el límite de conexiones por unidad de tiempo (segundo, minuto, hora, día y mes) y también por protocolos.

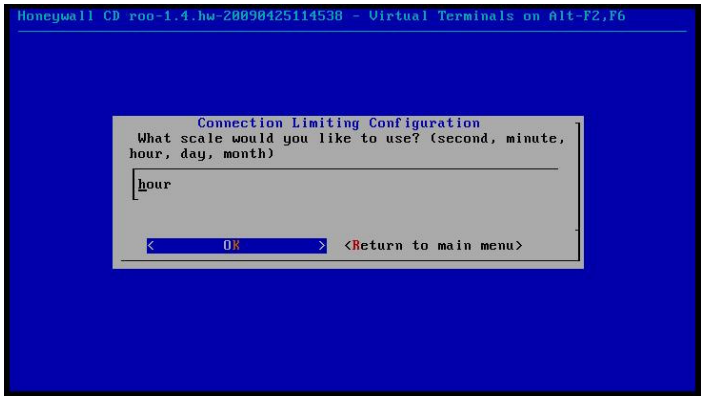


Figura 35. Especificación del límite de conexiones.

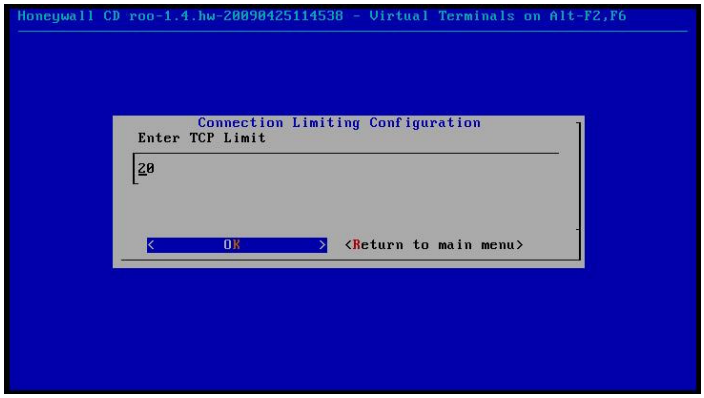


Figura 36. Especificación del límite de conexiones: *TCP*.

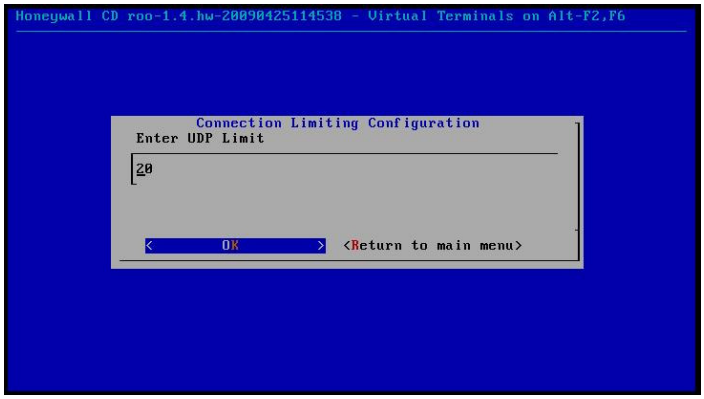


Figura 37. Especificación del límite de conexiones: *UDP*.

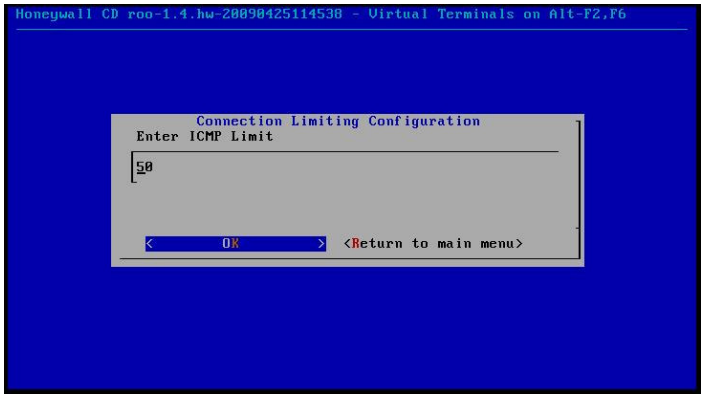


Figura 38. Especificación del límite de conexiones: *ICMP*.

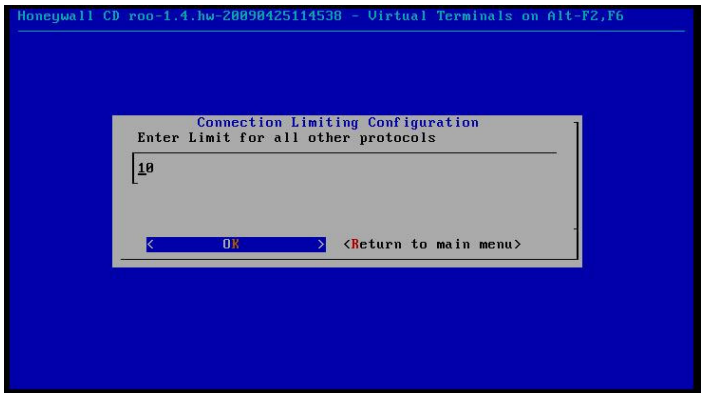


Figura 39. Especificación del límite de conexiones para otros protocolos.

7.2 Elija ***yes*** para habilitar la aplicación ***snort_inline***.

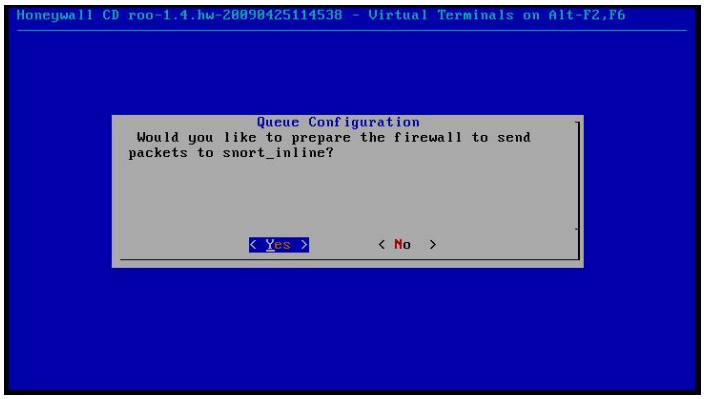


Figura 40. Confirmación para habilitar *snort_inline*.

7.3 Indicar el archivo que contiene la lista negra '*blacklist*' (direcciones IPs que generan Spam). Elija ***OK*** y presione ***Enter***.

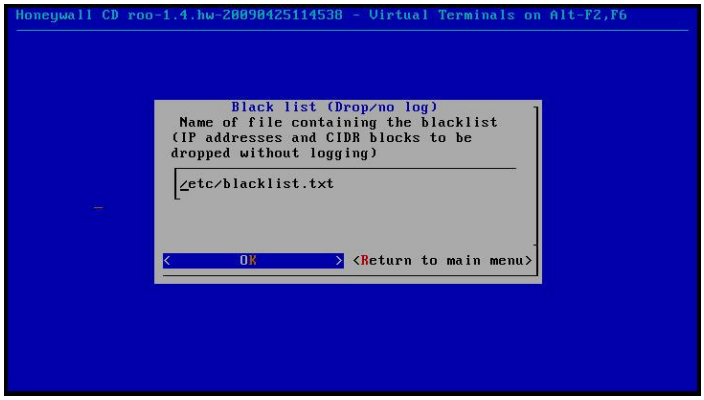


Figura 41. Archivo Blacklist.

7.4 Indicar el archivo que contiene las direcciones IPs, fijadas por el usuario, que no deben ser detectadas ni consideradas como Spam. Elija ***OK*** y presione ***Enter***.

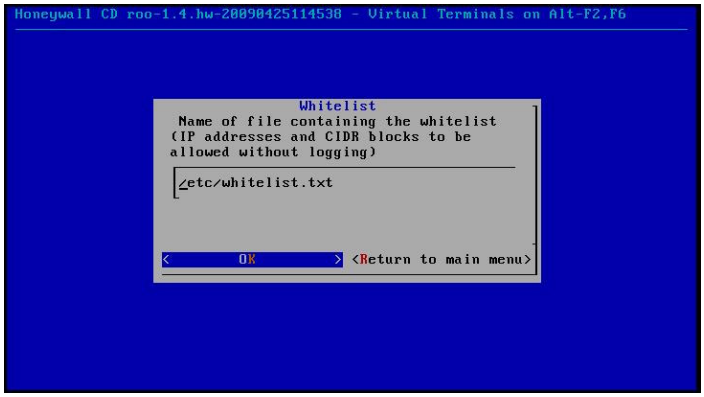


Figura 42. Archivo Whitelist.

7.5 Habilitar el filtrado para las listas (Black – White). Elija ***yes*** y presione ***Enter***.

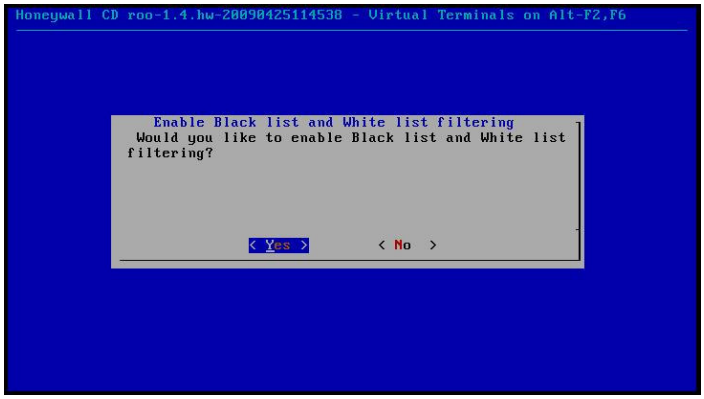


Figura 43. Filtrado listas (Black - White).

7.6 Elegir la opción ***no*** para habilitar el filtrado seguro durante la captura de paquetes.

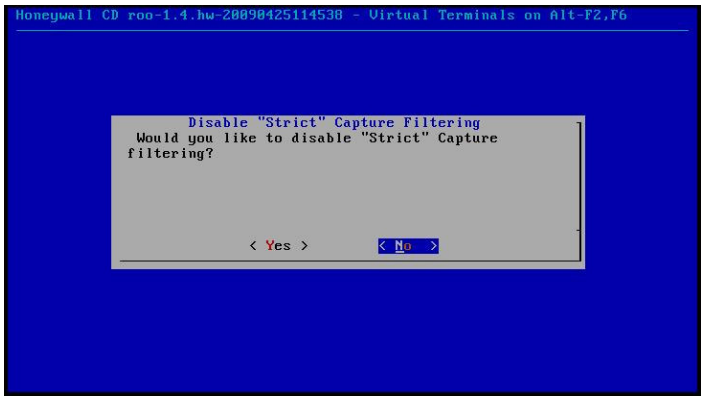


Figura 44. Filtrado seguro.

8. CONFIGURACIÓN DE FENCELIST.

8.1 ***FENCELIST:*** La finalidad de este fichero es para configurar IPTABLES para registrar y bloquear tráfico de salida hacia otros equipos o redes. Indique la ruta del directorio donde se encuentra el fichero y elija ***OK***.

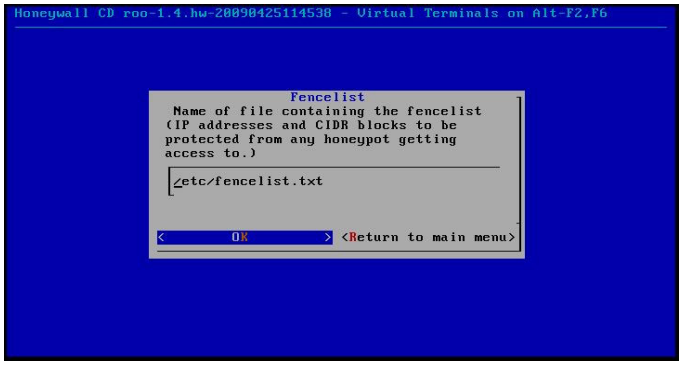


Figura 45. Fencelist.

8.2 Es necesario habilitar el filtrado ***Fencelist***, elija Yes.

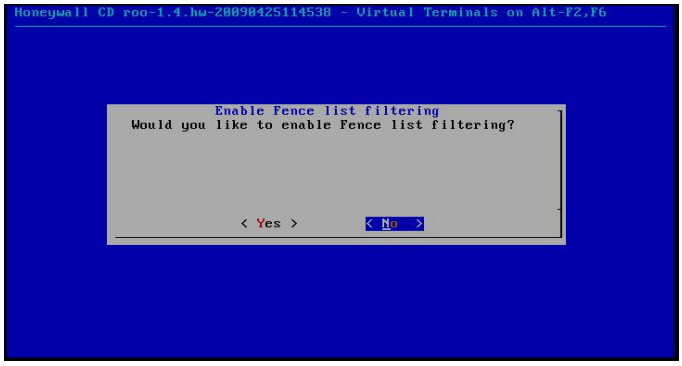


Figura 46. Filtrado Fencelist.

8.3 ***ROACH MOTEL:*** A través de este modo, un atacante podría detectar fácilmente el Honeywall y atacarlo posteriormente, es por ello que esta opción debe ser deshabilitada. Elija la opción ***no***.

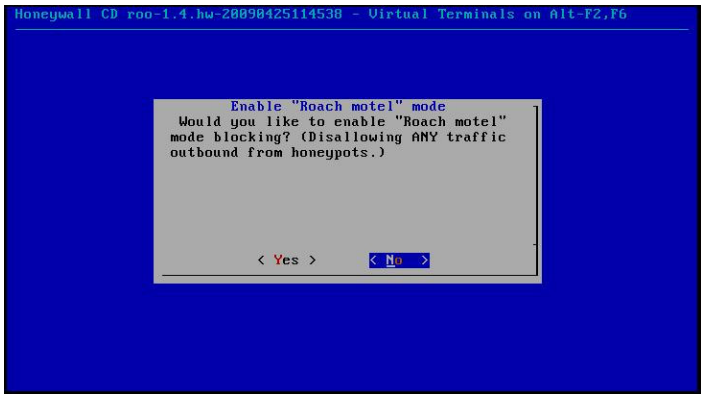


Figura 47. Modo Roach motel.

8.4 Confirme el mensaje de haber finalizado la 3ra sección de la configuración del sistema Honeywall.

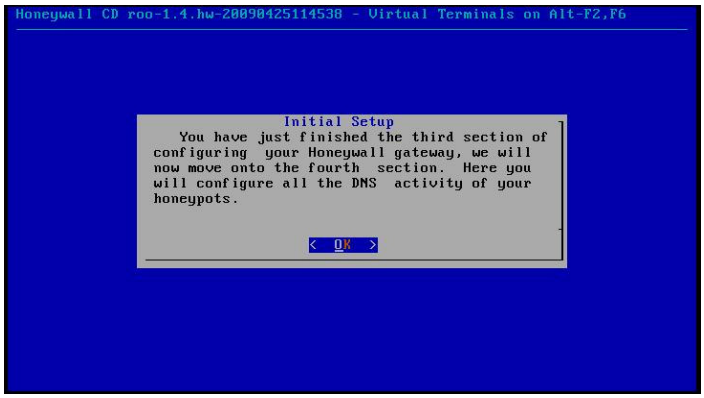


Figura 48. Finalización de la 3ra sección de configuraciones.

9. CONFIGURACIÓN DE DNS.

9.1 Habilitar los accesos ilimitados de los Honeypots al servidor DNS.

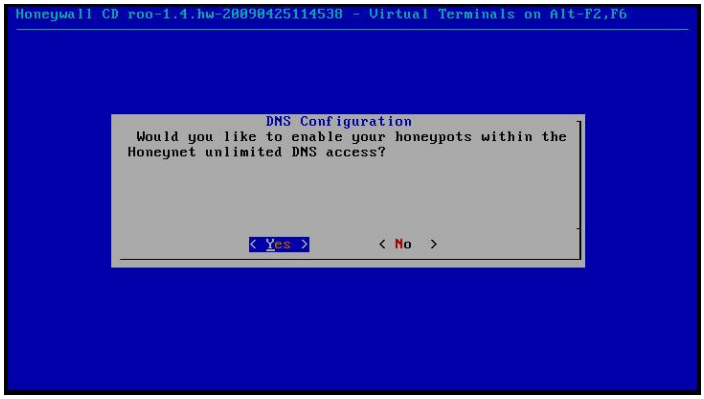


Figura 49. Accesos DNS ilimitados de los *Honeypots.*

9.2 Habilitar la restricción de accesos ilimitados de los Honeypots hacia servidores DNS externos.

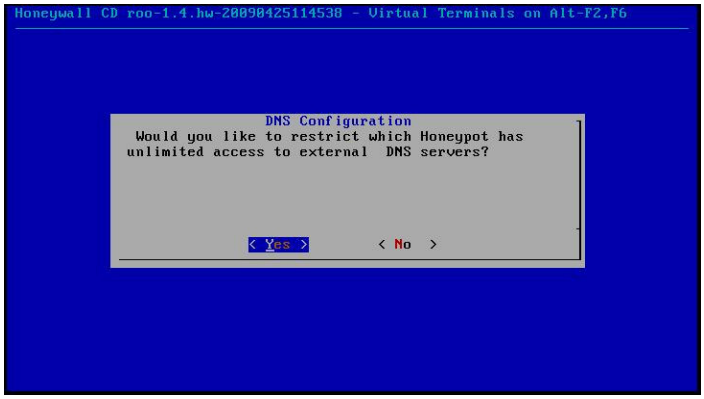


Figura 50. Accesos DNS ilimitados. *Honeypot – Servidores externos.*

9.3 Ingrese los Honeypots que accederán a servidores DNS externos.

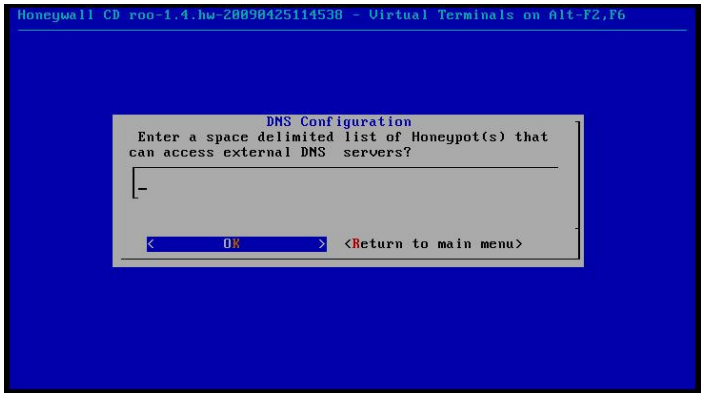


Figura 51. Honeypots que pueden acceder a servidores DNS externos.

9.4 Habilitar la restricción del servidor DNS a ser utilizado para accesos ilimitados.

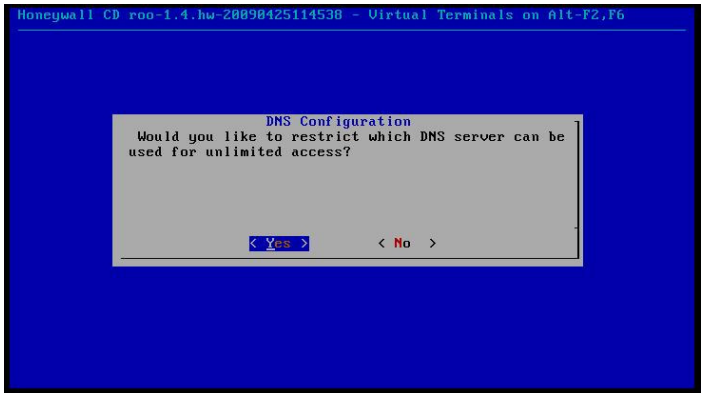


Figura 52. Restringir servidor DNS.

9.5 Ingrese la dirección IP del servidor DNS.

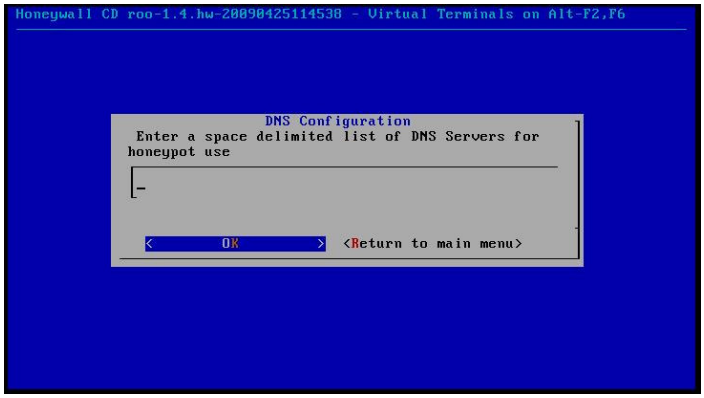


Figura 53. Ingreso de dirección del servidor DNS.

10. CONFIGURACIÓN DE ALERTAS.

10.1 Habilite la entrega de alertas por Email.

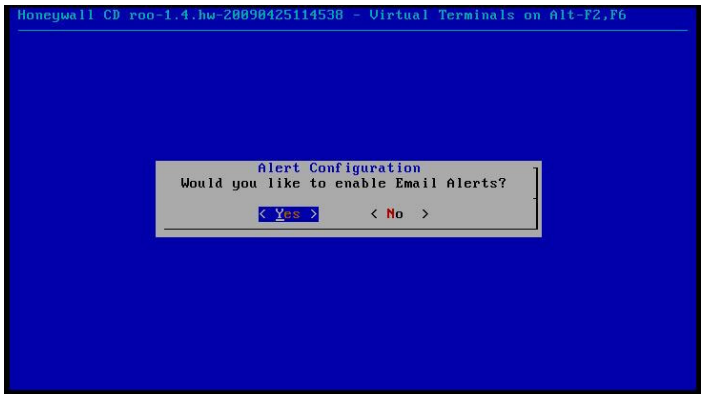


Figura 54. Configuración de alertas por Email.

10.2 Ingrese la dirección de correo electrónico donde se deben enviar las alertas.

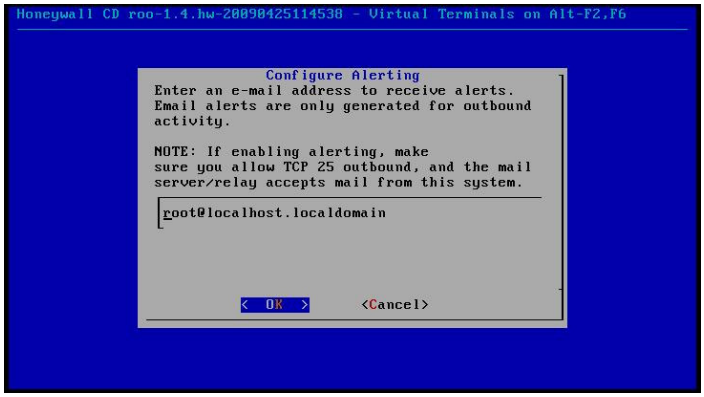


Figura 55. Ingreso de dirección de correo.

11. CONFIGURACIÓN DE LAS VARIABLES DE SEBEK.

11.1 Elija ***yes*** para iniciar la configuración de las variables de Sebek.

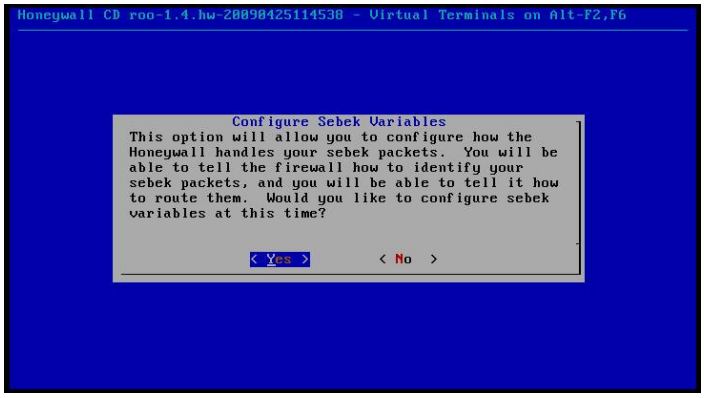


Figura 56. Inicio de configuración de Sebek.

11.2 Ingrese la dirección IP de destino de los paquetes Sebek, por defecto ***0.0.0.0***.

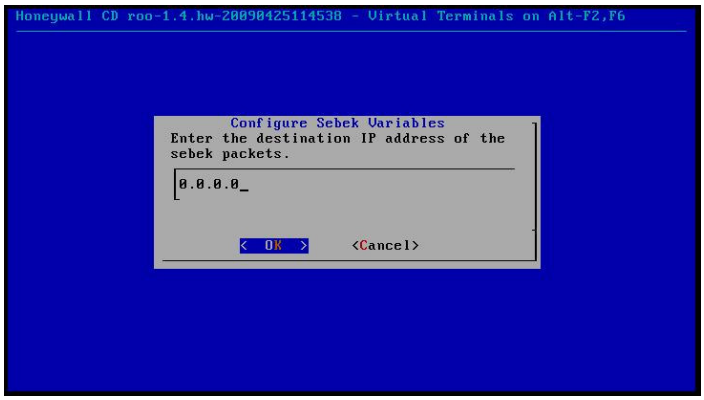


Figura 57. Dirección IP de destino – 0.0.0.0.

11.3 Ingrese el puerto UDP de destino de los paquetes Sebek, por defecto ***1101***.

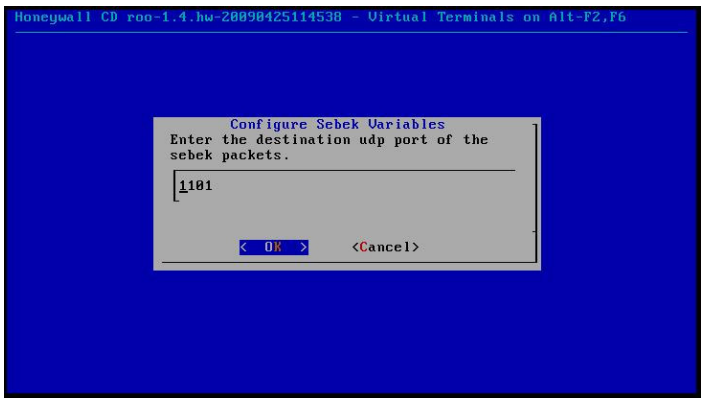


Figura 58. Puerto UDP de Sebek – 1101.

11.4 Del menú de opciones de registro, elegir el literal 4.

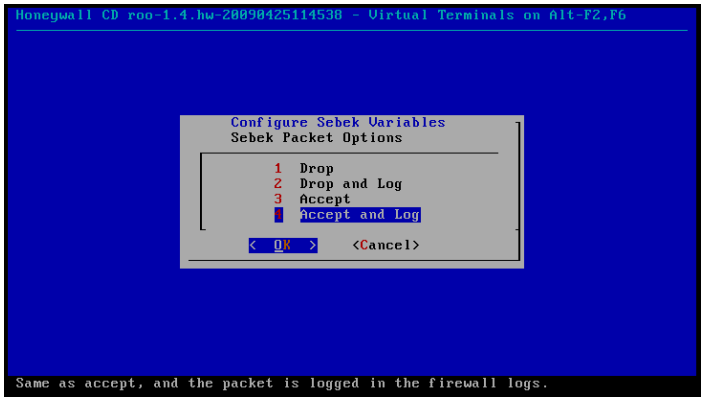


Figura 59. Opciones de registro del paquete Sebek.

11.5 El proceso de instalación y configuración ha terminado.

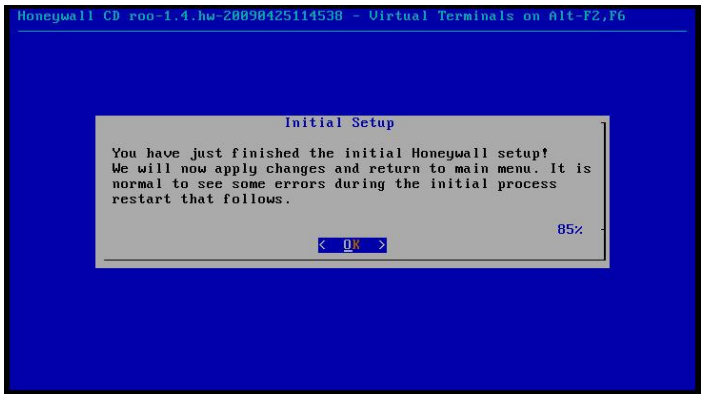


Figura 60. Pantalla de finalización del proceso.

ANEXO G
Análisis de datos

ETAPA 1

Reporte de los datos obtenidos durante los meses de Febrero y Marzo

Se utilizó la herramienta Wireshark para realizar el análisis de los archivos ***pcap*** capturados.

Los datos obtenidos durante estos meses están divididos de la siguiente manera:

1. Paquetes por Protocolos TCP – UDP – ICMP.

Tabla 1. Paquetes por Protocolo

Protocolo	Num. Paquetes
TCP	1570489
UDP	728654
ICMP	10755
TOTAL	2309898

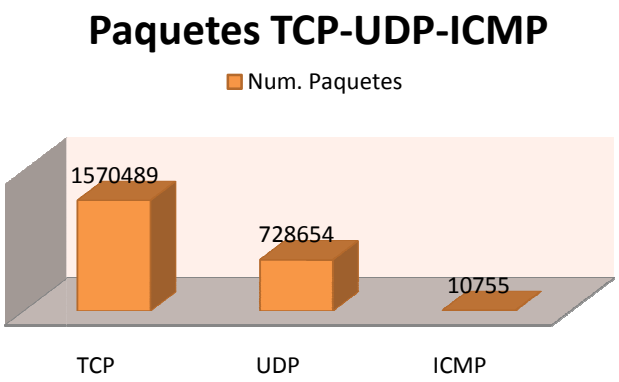


Figura 1. Muestras por paquetes hacia los protocolos durante la etapa 1

2. Cantidad de paquetes por Protocolo TCP y UDP.

Conexiones a través de los diferentes puertos TCP

Tabla 2. Conexiones a puertos TCP

Puerto	Total
TCP-6667	49064
microsoft- ds (445)	6587
epmap (135)	5771
ident (113)	1530
netbios-ssn (139)	1031
http-alt (8080)	686
csd-monitor (3072)	531
1024	510
X11 (6000)	195
ms-sql-s (1433)	122
ssc-agent (2967)	102
radmin-port (4899)	72
5900	50
instl_boots (1067)	12

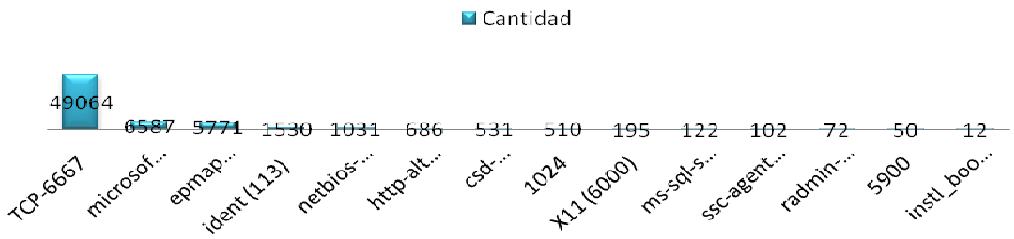


Figura 2. Número de conexiones a los puertos TCP

Conexiones a través de los diferentes puertos UDP

Tabla 3. Conexiones a puertos UDP

Puerto	Total
netbios - ns (137)	262
ms-sql-m (1434)	57
cap (1026)	54

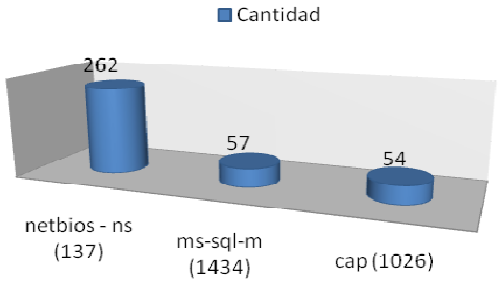


Figura 3. Número de conexiones a los puertos UDP

Febrero y Marzo son los primeros meses en los que se comenzó a recolectar los datos para su posterior análisis, los datos recogidos no son de todo el mes de febrero y marzo, sino, solamente de algunos días, sin embargo, la cantidad de datos es bastante numerosa pudiendo destacar que:

- ✓ Los puertos por los cuáles se genera mayor cantidad de tráfico es por el protocolo TCP, especialmente por los puertos 6667 corresponde a IRC, seguido de los puertos correspondientes al NetBios (445, 135, etc.).
- ✓ En cuanto a la cantidad de accesos por los puertos UDP, se puede destacar que también corresponden a los puertos NetBios.

Durante estos primeros meses no se pudo obtener información más detallada en cuanto a las alertas generadas, ni en cuanto al tráfico por puertos, los datos obtenidos fueron obtenidos de los archivos ***pcap*** capturados durante este corto lapso de tiempo.

3. Direcciones IP de origen.

Las direcciones listadas a continuación, son algunas las direcciones origen hacia los Honeypots, están ordenadas de acuerdo al número de paquetes transmitidos hacia los Honeypots.

El tipo de conexión utilizado es vía DSL.

Tabla 4. Direcciones IP que han accedido a los Honeypots

Dirección origen	Paquetes	Organización	País
91.189.88.40	175483	drescher.canonical.com	Reino Unido
91.189.88.31	43832	leningradskaya.canonical.com	Reino Unido
201.224.79.231	34031	Cable & Wireless Panamá	Panamá
200.0.29.66	26611	Universidad Técnica Particular de Loja	Ecuador
200.24.16.20	24080	nat20.udea.edu.co / Uni. de Antioquia	Colombia
66.232.143.55	22762	Hostway-KR	Corea
80.82.113.61	22656	ns.magicserve.co.uk	Reino Unido
204.160.100.124	17754	Level 3 Communications	EEUU
219.150.232.245	14198	CHINANET henan province network	China
200.7.104.38	13348	38-104-7-200.static-aceu.genesisbci.net / Eurocentro	Venezuela
200.0.29.66	10698	Universidad Técnica Particular de Loja	Ecuador
141.28.131.113	8096	Hochschule Furtwangen	Alemania
200.7.198.162	7558	mail.mscecuador.com / OTECEL	Ecuador
217.23.239.37	7502	Derwentside District Council	Reino Unido
66.221.203.235	5629	fikr.propagation.net / C I HOST	EEUU
190.95.134.34	4873	host-190-95-134-34.nettplus.net/ Telconet S.A	Ecuador
79.118.32.186	4728	Romania Data Systems	Rumania
161.53.178.240	2623	zagreb.hr.eu.undernet.org / CARNet backbone	Croacia
85.185.146.4	2455	Information Technology Company (ITC)	Irán
194.109.20.90	2425	undernet.xs4all.nl / XS4ALL Servers	Países Bajos
78.24.53.211	2095	MEDIANET (ALIT SRL)	Rep. de Moldova
116.7.255.86	2037	ChinaNet Guangdong Province Network	China
96.17.104.147	1839	AKAMAI TECHNOLOGIES	EEUU
195.197.175.21	1476	Saunalahti Group Oyj / irc2.saunalahti.fi	Finlandia
200.27.141.130	1375	Bata S.A.C	Chile
194.109.20.90	1342	XS4ALL Servers	Países Bajos
161.53.178.240	1271	CARNet backbone	Croacia
208.113.108.18	1227	208.113.108.18.servepath.com	EEUU
217.168.95.245	1136	undernet.underworld.no / NTEBB IRC Servernet	Noruega
58.213.155.162	1039	CHINANET jiangsu province network	China
199.93.41.124	977	Level 3 Communications	EEUU
195.18.164.194	938	BaneTele AS /oslo.no.eu.undernet.org	Noruega

208.83.20.130	798	tampa.fl.us.undernet.org / tampa.fl.us.undernet.org	EEUU
218.8.52.7	722	CNCGROUP Heilongjiang province network	China
81.195.21.45	603	ppp21-45.pppoe.mtu-net.ru / ZAO MTU-Intel	Rusia
69.16.172.34	595	mesa.az.us.undernet.org /Easynews	EEUU
69.16.172.40	550	irc3.easynews.com /Easynews	EEUU
221.231.104.134	546	CHINANET jiangsu province network	China
85.185.146.4	538	ISDP / Information Technology Company (ITC)	Irán

4. Reporte Honeysnap.

Durante el mes de marzo se analizó los archivos ***pcap*** del mes de febrero con una nueva herramienta encontrada para realizar el análisis de datos llamada ***Honeysnap***, esta herramienta permite obtener un resumen general de la cantidad de paquetes por protocolo, y permite ver un resumen de datos Http, IRC, entre otros.

Durante este tiempo la cantidad de tráfico dirigida al puerto TCP 6667 (IRC) es mucho mayor en cuanto al tráfico generado por los otros puertos, el resumen obtenido por Honeysnap es el siguiente:

Análisis de Tráfico IRC que contienen PRIVMSG para el Honeypot 200.0.30.51

Tabla 5. Resumen del tráfico IRC encontrado

Pattern	Source	Sport	Destino	Dport	Cant
Privmsg	200.0.30.51	37619	195.197.175.21	6667	1
Privmsg	195.18.164.194	6667	200.0.30.51	6667	72
Privmsg	195.197.175.21	6667	200.0.30.51	37619	71
Privmsg	194.109.20.90	6667	200.0.30.51	47355	128
Privmsg	217.168.95.245	6667	200.0.30.51	60307	40
Privmsg	200.0.30.51	6667	217.168.95.245	6667	1
Privmsg	161.53.178.51	6667	200.0.30.51	58058	36

De acuerdo a IRC, servidores utilizados

Tabla 6. Resumen del tráfico IRC encontrado

Canal	Cantidad
Diemen.NL.EU.Undernet.Org	1039
Oslo2.NO.EU.undernet.org	545
Oslo2.NO.EU.undernet.org	541
Zagreb.HR.Eu.UnderNet.org	529
Trondheim.NO.EU.Undernet.org	522
Tampa.FL.US.Undernet.org	161
Borning!~Lordx@Borning.users.undernet.org	133
Born2Love!~born@Profi.users.undernet.org	97

Algunos de los nicks utilizados son:

- ✓ AUTH: 3240
- ✓ #NewHacks: 1207
- ✓ RaydeN: 1206
- ✓ Lordx: 922
- ✓ Born2love: 886
- ✓ Distrus: 813
- ✓ RaydeN: 485
- ✓ Lordx_: 461
- ✓ _orn2love: 455

5. Inconvenientes.

Durante este tiempo, la interfaz gráfica Walleye estuvo sin funcionar, esto debido a inconvenientes en cuanto a las versiones de software y hardware utilizadas. El resto de archivos ***pcap*** capturados, no fueron analizados debido a que estos no registraban datos Sebek, es decir no se registraba actividad desde los Honeypots hacia afuera.

ETAPA 2

Reporte de los datos obtenidos desde el mes de Mayo a Julio

Desde el día 6 de mayo la Honeynet está más estable, la interfaz gráfica Walleye funciona, desde esta interfaz se ha podido obtener información más detallada acerca de las alertas emitidas por el Snort, se tienen datos más precisos en cuanto a la cantidad de accesos desde los diferentes protocolos.

Consolidado total de los datos obtenidos durante los meses de Mayo, Junio y Julio.

Tabla 7. Tráfico por paquetes TCP – UDP – ICMP

Tráfico	Mayo	Junio	Julio	Total
TCP	152935	811708	21184113	22148756
ICMP	3134	19032	491496	513662
UDP	47705	82149	7795	137649
TOTAL				**22800067**

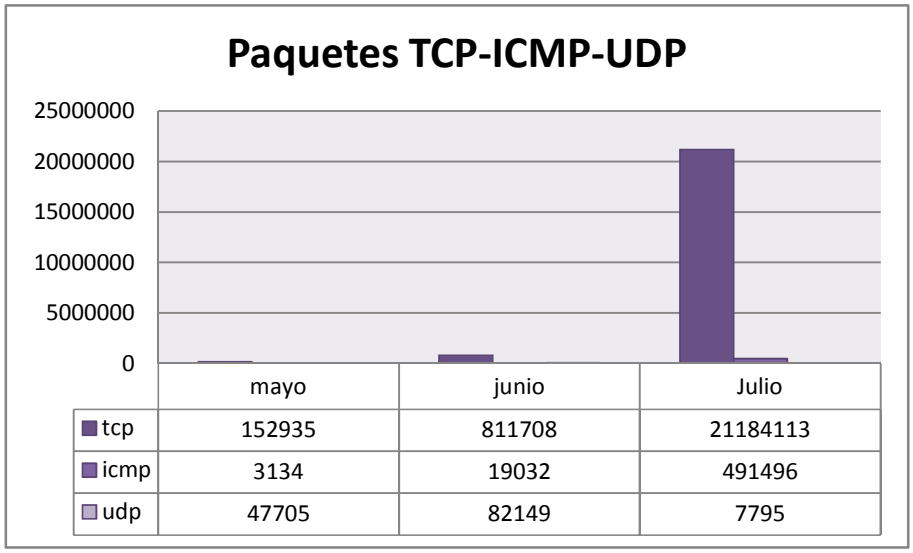


Figura 4. Muestras por paquetes hacia los protocolos durante la etapa 2

Tabla 8. Número de accesos por Protocolo TCP

Puertos TCP	Mayo	Junio	julio	Total
445 (microsoft-ds)	7661	21470	1363190	1392321
daytime	0	0	21298	21298
80 (htp)	334	13773	110	14217
tcpmux	0	0	9458	9458
ident (113)	3302	0	0	3302
22 (ssh)	568	1149	0	1717
1433(ms-sql-s)	183	691	82	956
135(epmap)	200	459	49	708
2967 (ssc-agent)	38	636	0	674
23 (telnet)	116	366	0	482
netbios- ssn(139)	0	0	59	59
25 (smtp)	0	0	43	43

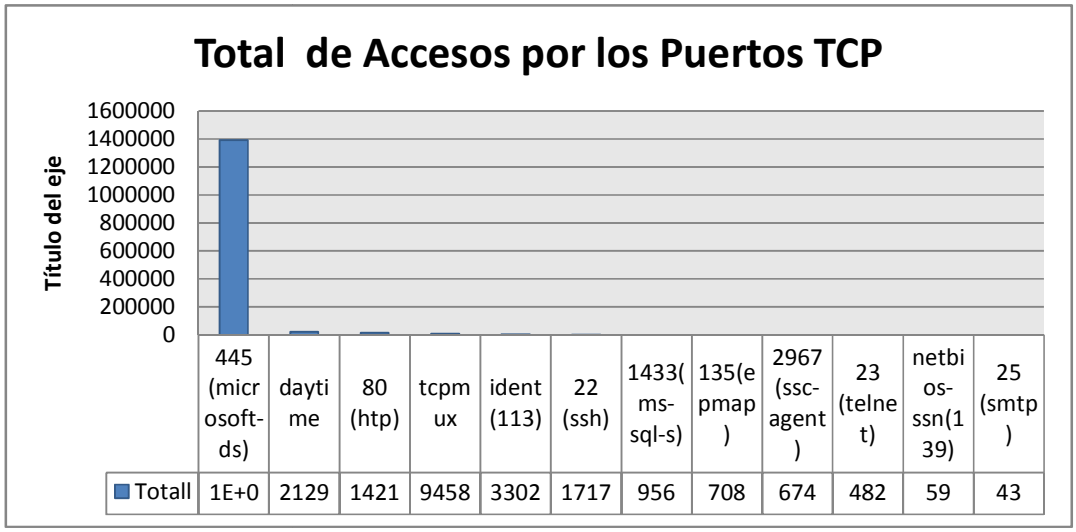


Figura 5. Muestras de accesos hacia los puertos TCP durante la etapa 2

Tabla 9. Número de accesos por Protocolo UDP

Puertos UDP	Mayo	Junio	Julio	Total
138 (netbios-dgm)	1770	6862	94	8726
53(domain)	1135	3218	0	4353
137 (netbios-ns)	138	1667	46	1851
1434 (ms-sql-m)	155	303	28	486

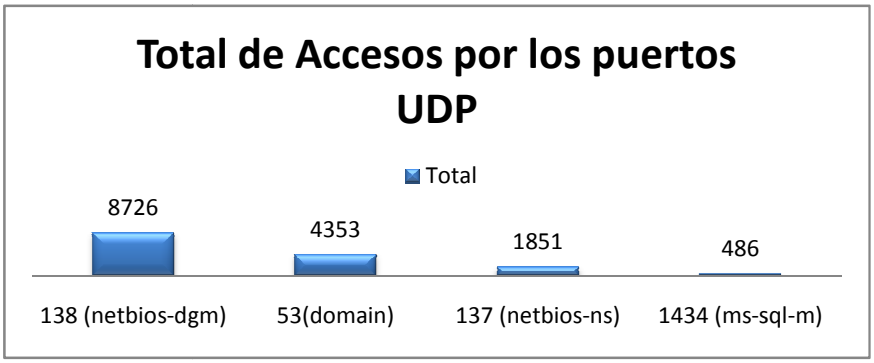


Figura 6. Muestras de accesos hacia los puertos UDP durante la etapa 2

Las alertas obtenidas durante estos meses suman un total de 29069 alertas, estas se han dividido en dos partes, el primer grupo de alertas corresponde a:

Tabla 10. Alertas ICMP recibidas – 1er bloque

Alertas ICMP				
Descripción	**Mayo**	**Junio**	**Julio**	**Total**
ICMP Destination Unreachable Communication Administratively Prohibited (1)	0	0	21110	**21110**
ICMP PING CyberKit 2.2 Windows (2)	1934	1097	56	**3031**
ICMP redirect host (3)	0	0	1284	**1284**
ICMP redirect net (4)	0	0	808	**808**
ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited (5)	0	0	328	**328**
ICMP PING NMAP	39	26	4	**65**
ICMP webtrends scanner	13		0	**13**
ICMP Destination Unreachable Communication with Destination Network is Administratively Prohibited	0	0	11	**11**
ICMP PING speedera	0	4	0	**4**
Total				**26654**

A continuación se explica el funcionamiento de cada alerta (con respecto al total de alertas reportadas durante este tiempo), se hace referencia a continuación solamente a las alertas subrayadas, esto debido a que el número de estas alertas es considerable.

La mayoría de las alertas a las que se hace referencia son alertas de tipo ICMP, las mismas que corresponden a un tipo y un código establecidos por este protocolo.

Alertas 1 y 5:

- ✓ **ICMP Destination Unreachable Communication Administratively Prohibited (1)**
- ✓ **ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited (5)**

Estas alertas son de tipo 3 y códigos 13 y 10 respectivamente, el código 3 (*Destination Unreachable*) es un tipo de error que se presenta cuando un host, recibe el mensaje de un enrutador intermedio, lo que significa que este router considera el destino inalcanzable o el servicio al que quiere acceder no está disponible, si este mensaje es recibido desde el host destino quiere decir que el protocolo al que se intentó acceder no está activo en aquel momento.

El código 13 (*Communication Administratively Prohibited*) se genera si un router no puede avanzar debido a que un paquete es filtrado administrativamente, es decir cuando el sistema destino está configurado para rechazar el envío de datagramas del sistema, este tipo de error se utiliza cuando los datagramas, basados en algún tipo de criterio están filtrados por un router, u otras medidas de seguridad.

Los datos obtenidos son los siguientes, se debe resaltar que en este lapso de tiempo ésta alerta mantiene la misma estructura.

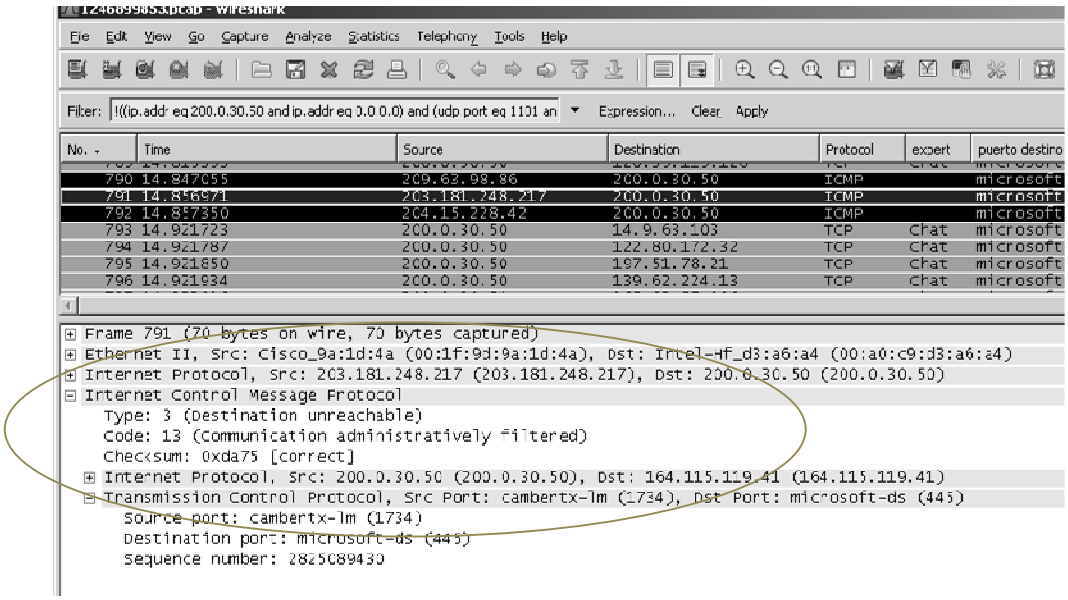


Figura 7. Data Wireshark – Alertas ICMP

Lo mismo sucede con el error generado por el código 10 (*Communication with Destination Host is Administratively Prohibited*), cuando este error es generado en el host destino significa que la dirección IP no puede enviar el datagrama debido a que el puerto del proceso indicado no está activo.

Alerta 2:

- ✓ **ICMP PING CyberKit 2.2 Windows (2)**

Esta alerta indica una mayor investigación sobre la exploración de posibles escaneos de host infectados con el worm ***Welchia/nachi***.

De acuerdo a Snort ésta alerta no es más que una solicitud normal de eco ICMP procedente de la herramienta de Windows CyberKit.

La alerta que da el Snort es de prioridad 2, en los archivos revisados no hay indicios de algún virus, lo que lleva a la conclusión de que es un falso positivo y que tiene que ver con una solicitud normal de eco ICMP.

El paquete obtenido es:

```
07/23-12:44:29.950798 0:1F:9D:9A:1D:4A -> 0:A0:C9:D3:A6:A4 type:0x800 len:0x4E
201.57.19.11 -> 200.0.30.50 ICMP TTL:116 TOS:0x0 ID:27006 IpLen:20 DgmLen:64
Type:8  Code:0  ID:512   Seq:10300  ECHO
AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA  ................
AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA  ................
AA AA AA AA                                      ....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Figura 8. Paquete Decodificado Snort – Alerta ICMP PING Cyberkit 2.2 Windows

Esta alerta es una alerta ICMP de tipo 8 y de código 0, el mismo que es utilizado para determinar si una dirección IP de destino está o no activa, esto solamente enviando una solicitud de eco ICMP (tipo 8) y se espera una respuesta de ICMP (tipo 0), si se recibe una respuesta la dirección IP destino esta activa.

Datos proporcionados por el Wireshar:

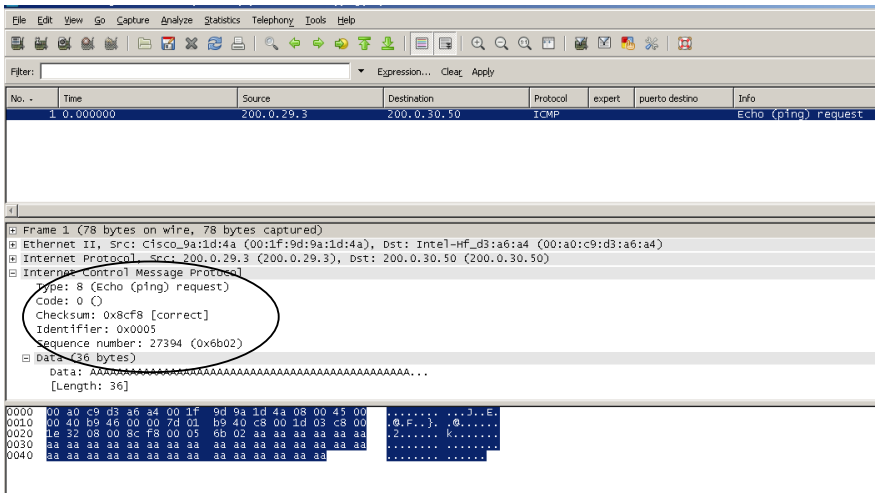


Figura 9. Data Wireshark – Alertas ICMP

Alerta 3 y 4.

- ✓ **ICMP redirect host (3)**
- ✓ **ICMP redirect net (4)**

Estas alertas son parte de ICMP, son de código 5, este tipo de mensaje es utilizado por un router / firewall para informar a la fuente sobre una ruta favorita hacia el host de destino seleccionado.

Funcionamiento: El router envía un paquete al destino y un ICMP lo redirecciona a la fuente que contiene la entrada alternativa, lo que provoca un cambio en la tabla de routing de la fuente.

El router que lanza el ICMP redirect tiene que estar en la misma subred que la fuente y la nueva entrada.

Código 1: *Redirect host (Redireccionar el datagrama para el host)*, está catalogado como potencial tráfico malicioso, tiene prioridad 2 y hace referencia a la vulnerabilidad CVE – 1999-0265.

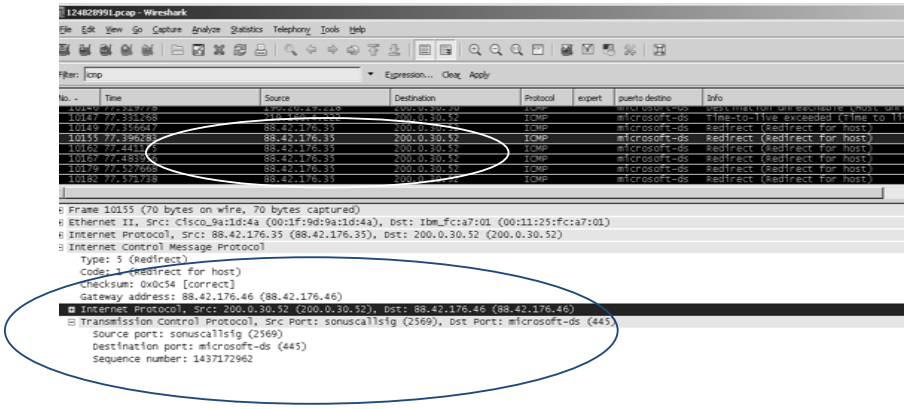


Figura 10. Data Wireshark – Alerta ICMP Redirect host

Código 0: *Redirect net (Redirecciones el datagrama para la red)*, significa que indica al host que debe cambiar la dirección del enrutador que le fue predeterminado, o indica que debe cambiar la dirección a otra subred, al igual que la alerta anterior hace referencia a la vulnerabilidad CVE – 1999-0265.

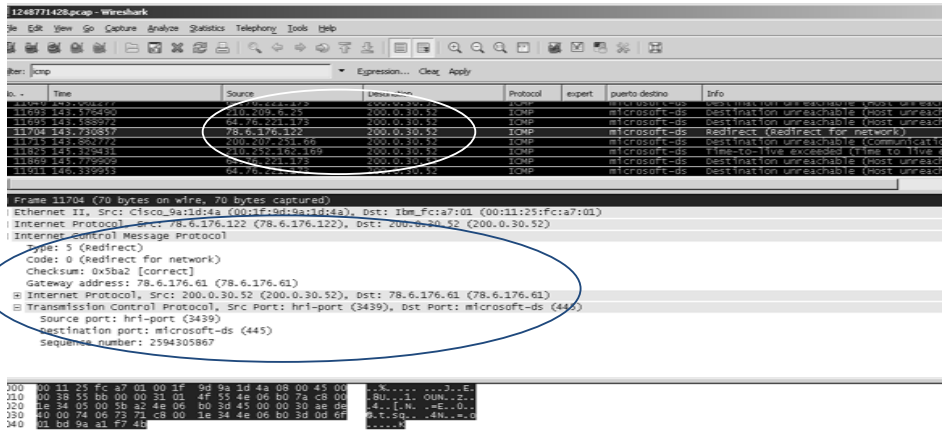


Figura 11. Data Wireshark – Alerta ICMP Redirect net

El posible abuso de estas alertas es que un usuario con malas intenciones pueda cambiar la tabla de routing de un host para redireccionar el tráfico y así cortar la comunicación.

Los puertos origen de las alertas presentadas en el primer grupo son el *TCP 445, daytime y tcpmux*, lo que se puede contrastar con las estadísticas referentes al total de accesos por los puertos TCP.

La mayor parte de estas alertas son catalogadas como falsos positivos, la solución consiste en revisar las alertas configuradas por defecto en el Snort.

La mayor parte de las alertas presentadas en el primer grupo son alertas ICMP, este tipo de alertas son enviadas en varias situaciones, como las que se han presentado a continuación, es decir cuando un datagrama no puede alcanzar su destino y cuando el gateway puede dirigir al host para enviar el tráfico por una ruta más corta.

Segundo bloque de alertas

El segundo bloque de alertas corresponde a:

Tabla 11. Alertas ICMP recibidas – 2do bloque

Alerta	Junio	Mayo	Julio	Total
MS-SQL Worm propagation attempt ; MS-SQL Worm propagation attempt OUTBOUND ; MS-SQL version overflow attempt (1)	740	456	18	**1196**
NETBIOS SMB-DS IPC$ unicode share access (2)	0	0	631	**631**
NETBIOS SMB-DS srvsvc NetrPathCanonicalize unicode little endian overflow attempt (3)	0	0	538	**538**
WEB-IIS view source via translate header (4)	0	0	19	**19**
BAD-TRAFFIC tcp port 0 traffic	3	16	4	**16**
SNMP request tcp	6		0	**6**
SNMP AgentX/tcp request	5		0	**5**
SNMP trap tcp	4		0	**4**
Total				**2415**

Alerta 1:

El primer bloque de alertas (1), de acuerdo a los datos de la tabla tiene mayor incidencia desde el mes de mayo, luego del análisis de los datos correspondientes se determinó que a través del tráfico dirigido al puerto UDP 1434 se infectó a los Honeypots con el *Worm Slammer*, explotando la vulnerabilidad CVE 2002-0649. Ver Anexo4: Vulnerabilidades Encontradas y Anexo 5: Código Malicioso encontrado

Las direcciones que han accedido a los Honeypots con el puerto UDP 1434 durante estos meses son:

Tabla 12. Direcciones IP que han accedido a los Honeypots con el puerto UDP 1434

Dirección	Dominio	País	Cantidad
61.145.123.141	ChinaNet Guangdong Province Network	China	38
60.161.78.144	CHINANET Yunnan province network	China	27
218.22.244.45	CHINANET Anhui province network	China	25
218.75.199.50	CHINANET-HN Zhuzhou node network	China	21
211.99.122.18	Jinan Dadu Hotel	China	17
61.139.54.94	Beelink Information Science &	China	17
59.80.95.35	Beelink Information Science &	china	12
222.82.249.235	WLMQ-NONGJITUIGUANG	China	11
61.145.123.141	ChinaNet Guangdong Province Network	China	10
58.243.161.51	CNC Group AnHui province network	China	10
202.101.180.165	Jinhua Telecommunication Co.ltd	China	9
218.4.149.124	Kuatang Primary School of Suzhou Industrial Park	China	7
61.131.151.83	CHINANET Jiangxi province network	China	7
65.145.123.141	Qwest Communications	EEUU	6
89.34.153.157	SC UNDERNET SRL	Rumania	6
61.184.255.175	CHINANET Hubei province network	China	5
218.204.137.156	China Mobile Communications Corporation - jiangxi	China	5
210.217.174.37	KRNIC	Corea	5

Alertas 2 y 3:

- ✓ **NETBIOS SMB-DS IPC$ unicode share access (2)**
- ✓ **NETBIOS SMB-DS srvsvc NetrPathCanonicalize unicode little endian overflow attempt (3)**

La alerta 3 está catalogada como alerta de Prioridad 1, hace referencia a la vulnerabilidad CVE 2006-3439 y al boletín MS06-040. Ver Anexo 4: Vulnerabilidades Encontradas

El boletín MS06-040 corresponde a una vulnerabilidad en el servicio de servidor, el impacto la ejecución de código remoto, se recomienda bloquear los puertos Netbios.

Alerta 4 (WEB-IIS view source via translate header):

Esta alerta de acuerdo a Snort ésta clasificada como Acceso a una aplicación web potencialmente vulnerable, tiene una prioridad de 2. Ver Anexo 4: Vulnerabilidades encontradas

Data Sebek:

Son varias las direcciones que desde el Honeypot realizan conexiones hacia afuera con el puerto TCP 445 como destino, algunas de las direcciones encontradas en Data Sebek son:

192.26.232.5; 101.37.188.33; 196.119.92.100; 96.125.203.40; 42.35.35.104; 51.80.56.93; 33.62.72.80.

Realizando un filtrado en el Wireshark en el que se pide que todas las direcciones que tengan como origen la dirección 200.0.30.50 y como destino 0.0.0.0 y el puerto de origen sea el UDP 1101 se obtiene los siguientes datos:

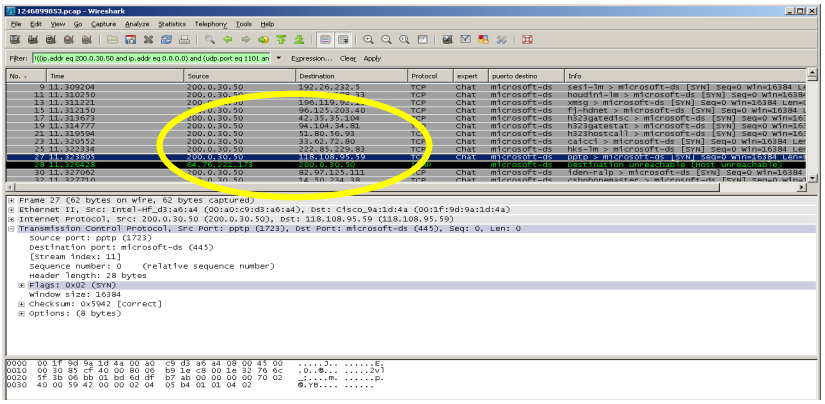
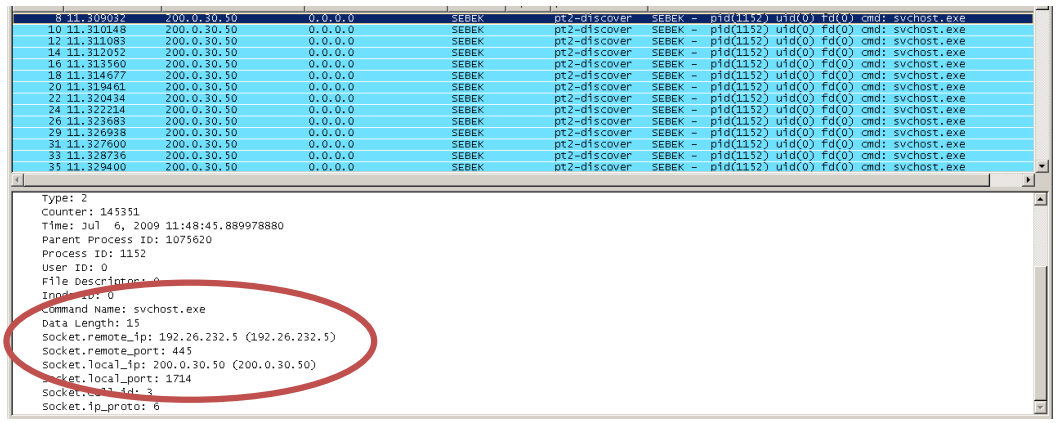


Figura 12. Data Wireshark de los paquetes Sebek encontrados

Lo que confirma que el puerto utilizado como destino desde el Honeypot hacia afuera es el TCP 445, desde los Honeypots se ha generado una cantidad considerable de alertas especialmente de tipo ICMP.

Desde el Honeypot 200.0.30.50 se ingresa al host 192.26.232.5 con el puerto TCP 445 y con el comando svchost.exe.



Para verificar si la dirección 192.26.232.5 que aparece en Data Sebek, consta entre las direcciones de origen o destino en los Honeypots se hace un filtrado de esa dirección y se obtienen los datos que se observan en la Figura 14.

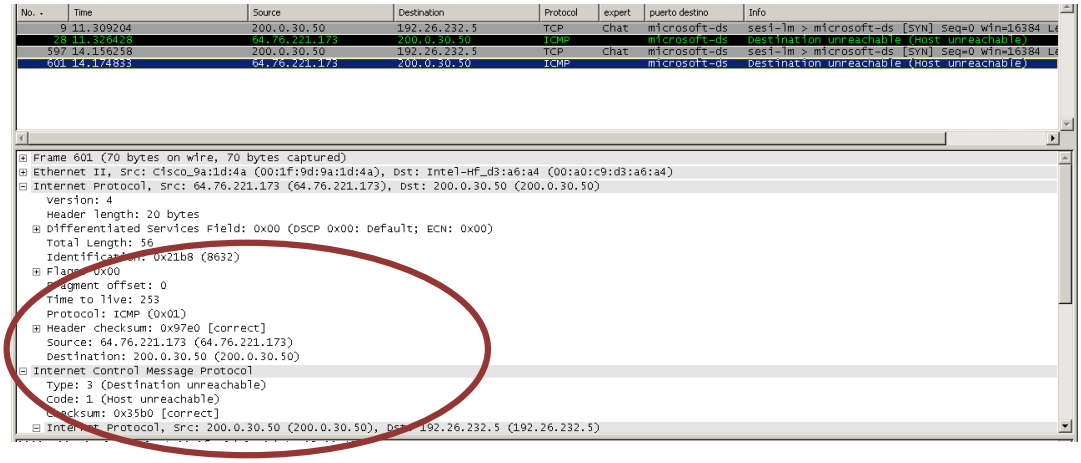


Figura 14. Data Wireshark de los paquetes Sebek encontrados

Lo que significa que desde el Honeypot se accede a esa dirección a través del puerto TCP 445, a su vez se muestra que con el comando ICMP a través de la dirección 64.76.221.73 se accede al Honeypot 200.0.30.50 lo que da como resultado una alerta ICMP tipo 3 (*Destination Unreachable*) de código 1 (*Host Unreacheable*).

ETAPA 3

Reporte de los datos obtenidos en el mes de Agosto

La cantidad de paquetes correspondientes a los puertos TCP-UDP e ICMP se detallan a continuación:

Tabla 13. Paquetes por protocolos TCP – UDP – ICMP

Tráfico	Semana 1	Semana 2	Total
TCP	8801251	1590566	**10391817**
ICMP	281756	56630	**338386**
UDP	845290	774125	**1619415**

Tomando en cuenta la cantidad de datos dirigida específicamente del puerto TCP 445 del mes anterior, el día 3 de agosto se bloqueó el tráfico dirigido a este puerto en el Honeypot 200.0.30.52, tomada esa acción, la cantidad de alertas ICMP y el tráfico al puerto 445 disminuyó considerablemente, sin embargo, la cantidad de tráfico a este puerto sigue encabezando la lista debido a que ahora la mayoría de tráfico está dirigido hacia el Honeypot 200.0.30.51.

En la Figura 15 se presenta la cantidad total de paquetes TCP, ICMP y UDP del mes de agosto.

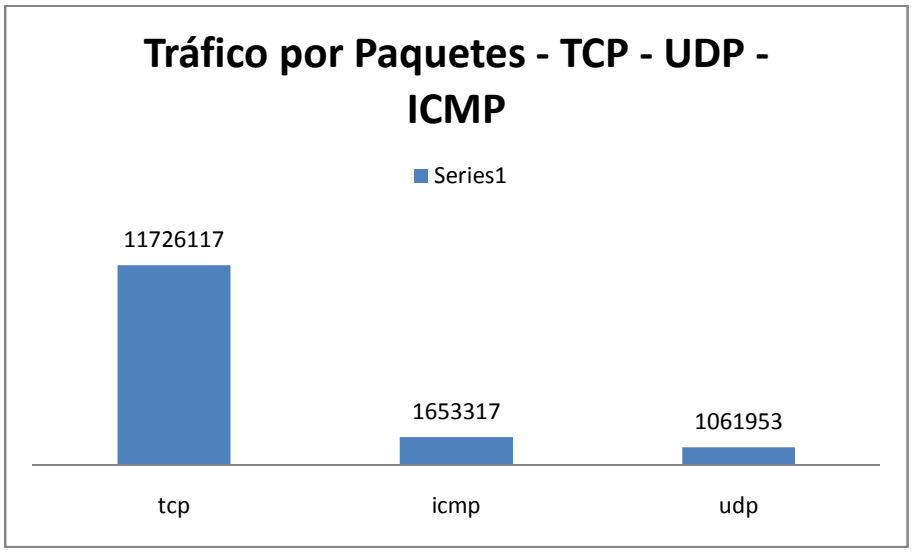


Figura 15. Paquetes TCP – ICMP – UDP

La cantidad de accesos por los diferentes protocolos se detalla a continuación:

Tabla 14. Cantidad de accesos por puertos TCP

Puertos TCP	Semana 1	Semana 2	Semana 3	Semana 4	Total
445 (microsoft-ds)	2156832	2632	2632	61580	2223676
daytime	14884	19	19	405	15327
tcpmux	10423	11	11	322	10767
netbios-ssn(139)	819	48	48	66	981
80 (htp)	479	80	80	107	746
135(epmap)	127	99	99	57	382
25 (smtp)	98	108	108	18	332
1433(ms-sql-s)	105	85	85	29	304

Como se mencionó anteriormente, encabeza la lista el puerto 445, la mayor cantidad de tráfico tanto de origen como destino se genera en su mayoría desde y hacia el equipo 200.0.30.50.

Como se puede observar en la Figura 16, le siguen al puerto 445 los puertos *daytime* y *tcpmux*, los mismos que están relacionados con las alertas ICMP, NetBios e IIS.

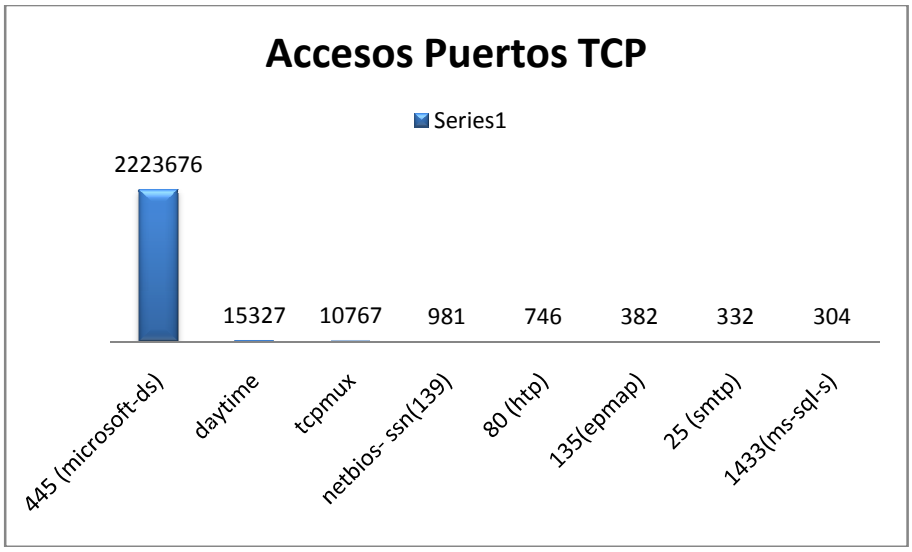


Figura 16. Cantidad de accesos por los puertos TCP

Tabla 15. Cantidad de accesos por puertos UDP

Puertos UDP	Semana 1	Semana 2	Semana 3	Semana 4	Total
138 (netbios-dgm)	87	0	0	246	333
1434 (ms-sql-m)	39	21	21	5	86
137 (netbios-ns)	55	3	3	4	65

De acuerdo a la Tabla 15, aparte del puerto 138 se registra actividad en el puerto UDP 1434, el que está relacionado con el *worm Slammer*, el mismo que se ha detallado en los meses anteriores.

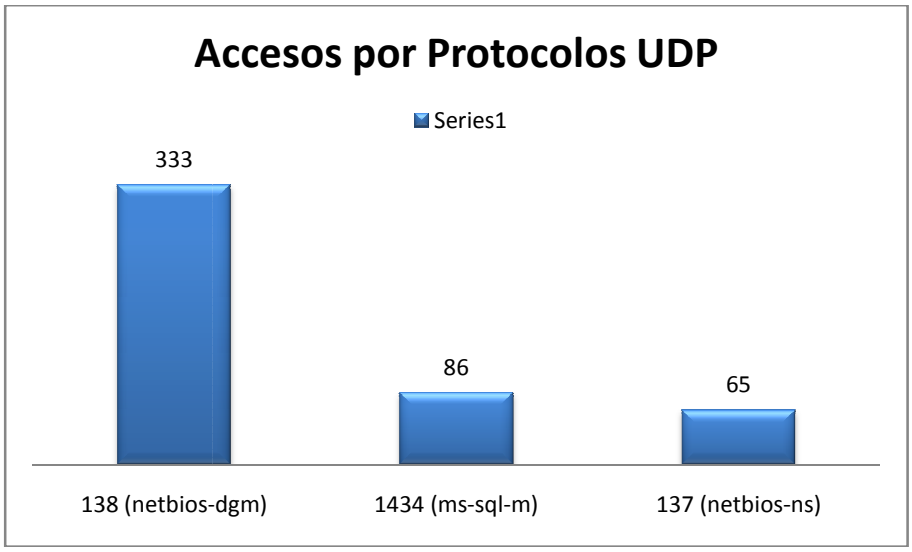


Figura 17. Accesos por protocolos UDP

1. Alertas recibidas.

Las alertas recibidas durante este mes se detallan a continuación:

Tabla 16. Alertas recibidas

Alerta	Semana 1	Semana 2	Semana 3	Semana 4	Total
ICMP Destination Unreachable Communication Administratively Prohibited	16237	47	47	515	16284
ICMP redirect host	1059	0	0	0	1059
ICMP PING CyberKit 2.2 Windows	289	327	0	167	616
NETBIOS SMB-DS IPC$ unicode share access	565	0	0	0	565
NETBIOS SMB-DS srvsvc NetrPathCanonicalize unicode little endian overflow attempt	535	0	0	0	535
ICMP redirect net	479	0	0		479
ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited	332	0	0	21	332
MS-SQL Worm propagation attempt ; MS-SQL Worm propagation attempt OUTBOUND ; MS-SQL version overflow attempt	96	63	63	21	243
ICMP Destination Unreachable Communication with Destination Network is Administratively Prohibited	58	58	58	4	178

WEB-IIS view source via translate header	30	0	0	12	42
ICMP PING NMAP	5	0	0	3	8
bad trafico port 0	1	1	1	0	3
ICMP PATH MTU denial of service	1	0	0	0	1
ICMP Source Quench	1	1	1	0	3

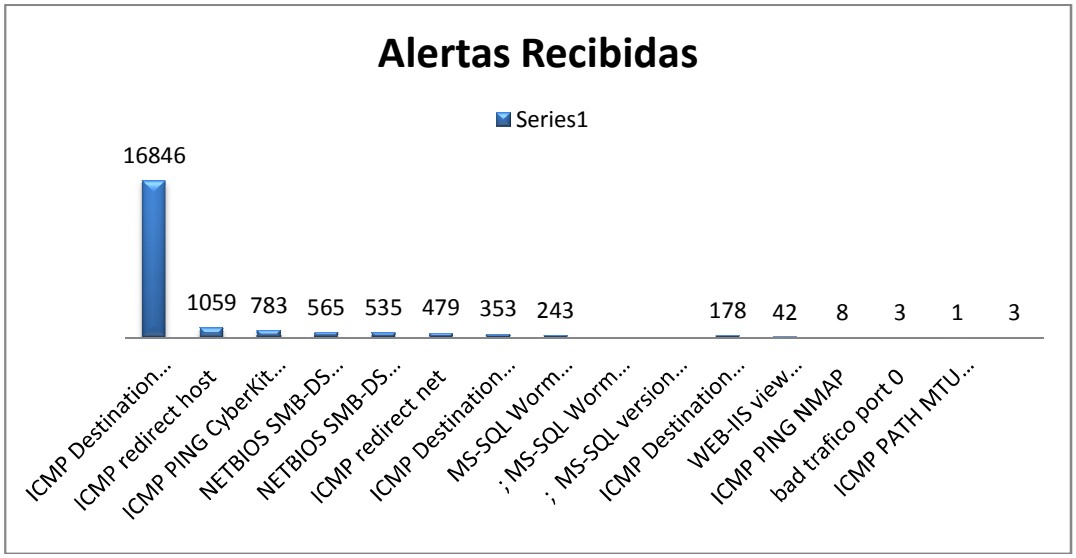


Figura 18. Alertas recibidas

Las dos primeras alertas tipo ICMP han sido explicadas anteriormente, en esta semana al igual que en los meses anteriores existe un alto número de alertas ICMP Ping Cyberkit 2.2 Windows, y en general el tipo de alertas se mantiene desde los meses anteriores.

Las direcciones que han accedido a los Honeypots con el puerto UDP 1434 son las siguientes:

Tabla 17. Direcciones que han accedido a los Honeypots con el puerto UDP 1434

Dirección Origen	Dominio	País	Cantidad
61.145.123.141	ChinaNet Guangdong Province Network	China	10
202.101.180.165	Jinhua Telecommunication Co.ltd	China	9
58.243.161.51	CNC Group AnHui province network	China	9
218.75.199.50	CHINANET-HN Zhuzhou node network	China	5
218.204.137.156	China Mobile Communications Corporation - jiangxi	China	3
58.42.234.135	CHINANET Guizhou province network	China	3

61.139.54.94	Beelink Information Science &	China	3
132.248.171.172	Universidad Nacional Autonoma de Mexico	México	1
200.233.146.143	Companhia de Telecomunicacoes do Brasil Central	Brasil	1
119.85.73.8	CHINANET Chongqing province network	China	1
124.173.184.18	World Crossing Telecom(GuangZhou) Ltd.	China	1
128.97.152.187	University of California, Los Angeles	EEUU	1
218.65.14.6	CHINANET Jiangxi province network	China	1

Data Sebek

Los datos de la Figura 19 indican que desde los Honeypots se ha utilizado el comando ***netstat.exe***

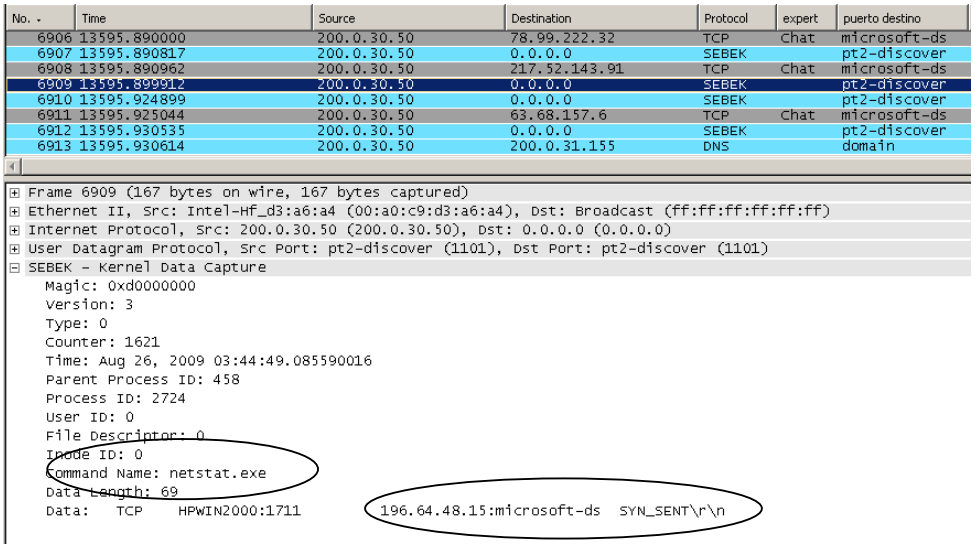


Figura 19. Data Wireshark de los paquetes Sebek encontrados

Analizando estos datos en el Honeysnap, se tiene como resultado:

- ✓ Se verifica que exista conectividad desde el Honeypot haciendo una petición de ping, tanto al Honeypot como a la puerta de enlace.

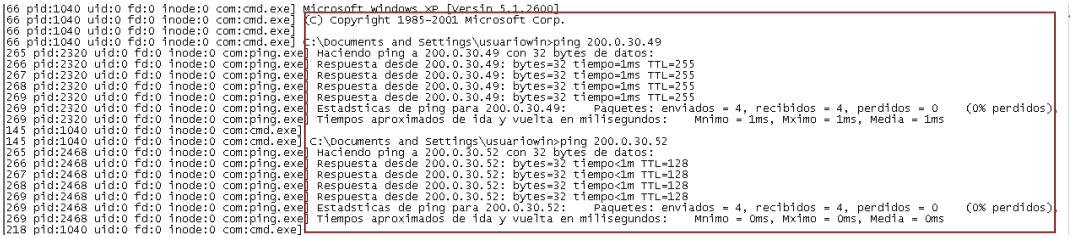


Figura 20. Resumen obtenido de la interfaz Honeysnap

- ✓ Luego con el comando ***netstat – a*** se muestran todas las conexiones y los puertos de escucha, por ejemplo la dirección 196.164.48.25 con el puerto TCP 445.

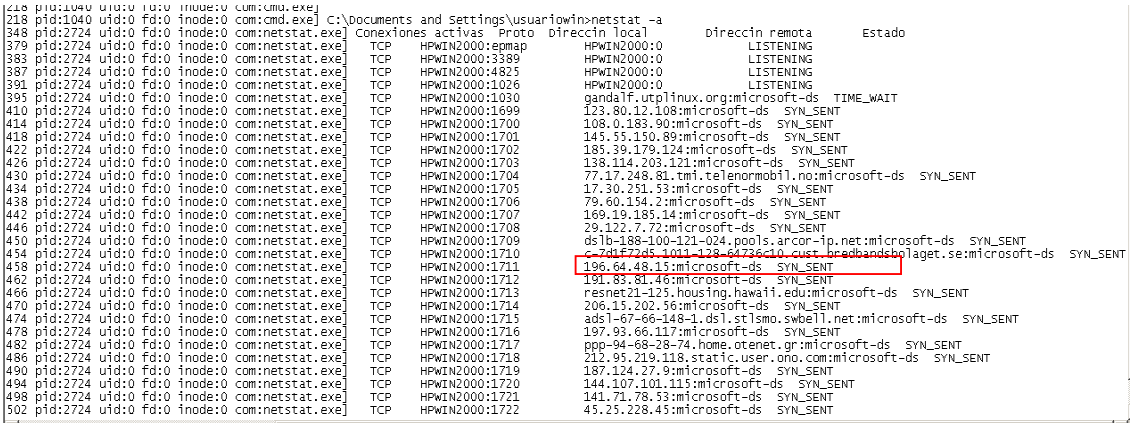


Figura 21. Resumen obtenido de la interfaz Honeysnap

Finalmente revisando esta dirección en el Wireshark encontramos que, desde el Honeypot se accedió a esta dirección (196.64.48.15) con el puerto TCP 445.

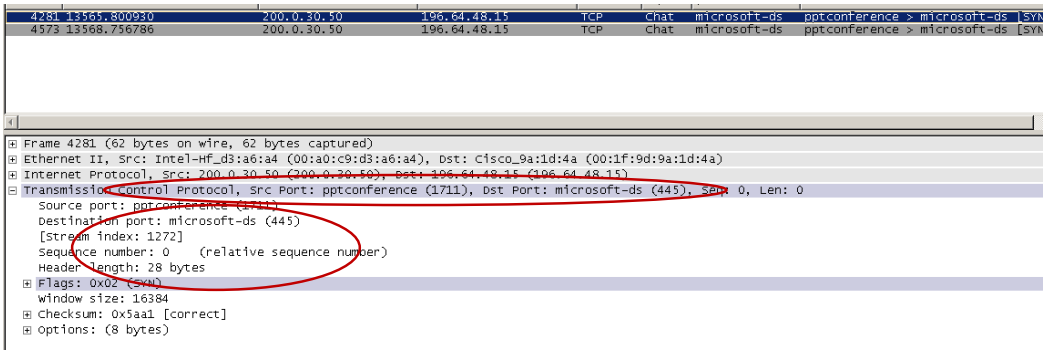


Figura 22. Data Wireshark de la actividad registrada

A más del comando ***netstat.exe*** se ha encontrado también actividad con el comando ***svchost.exe***, y coincide con la actividad indicada en las semanas anteriores.

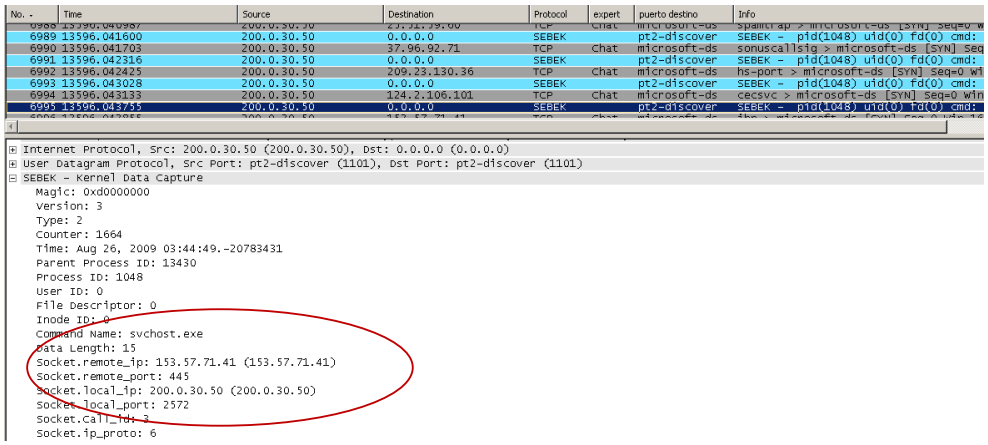


Figura 23. Data Wireshark de la actividad registrada

ETAPA 4

Consolidado Final

A continuación se presenta un resumen consolidado de los datos obtenidos.

Tabla 18. Paquetes TCP – ICMP – UDP

Tráfico	Mayo	Junio	Julio	Agosto	Total
TCP	152935	811708	21184113	11726117	33874873
ICMP	3134	19032	491496	1653317	2166979
UDP	47705	82149	7795	52906	190555

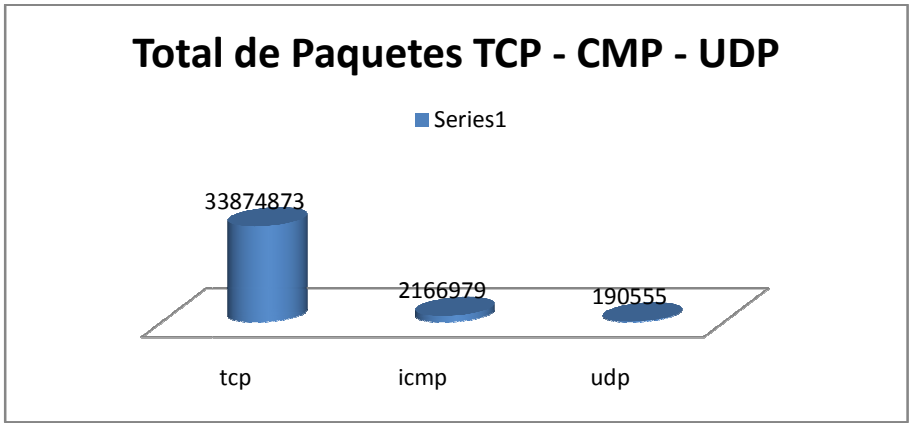


Figura 24. Muestras por paquetes hacia los protocolos

Tabla 19. Cantidad de accesos a puertos UDP

Puertos UDP	Mayo	Junio	Julio	Agosto	Total
138 (netbios-dgm)	1770	6862	94	333	9059
53(domain)	1135	3218	0	0	4353
137 (netbios-ns)	138	1667	46	65	1916
1434 (ms-sql-m)	155	303	28	86	572

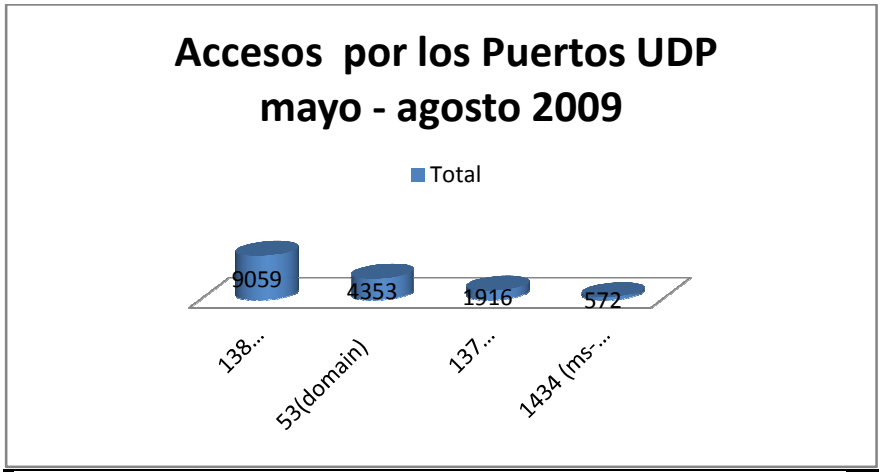
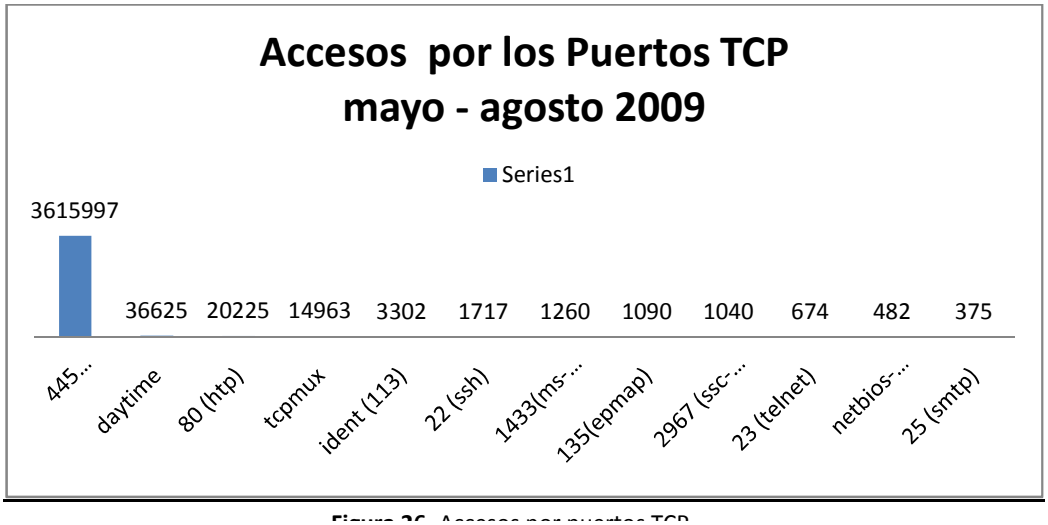


Figura 25. Accesos por puertos UDP

Tabla 20. Cantidad de accesos a puertos TCP

Puertos TCP	Mayo	Junio	Julio	Agosto	Total
445 (microsoft-ds)	7661	21470	1363190	2223676	3615997
daytime	0	0	21298	15327	36625
80 (htp)	0	0	9458	10767	20225
tcpmux	334	13773	110	746	14963
ident (113)	3302	0	0	0	3302
22 (ssh)	568	1149	0	0	1717
1433(ms-sql-s)	183	691	82	304	1260
135(epmap)	200	459	49	382	1090
2967 (ssc-agent)	0	0	59	981	1040
23 (telnet)	38	636	0	0	674
netbios- ssn(139)	116	366	0	0	482
25 (smtp)	0	0	43	332	375



1. Alertas recibidas.

El total de alertas recibidas ha sido de 50254, están divididas en dos grupos, las alertas tipo ICMP y las alertas tipo MS-SQL, NetBios, etc.

Tabla 21. Otras alertas

Alertas	Total
MS-SQL Worm propagation attempt ; MS-SQL Worm propagation attempt OUTBOUND ; MS-SQL version overflow attempt	1457
NETBIOS SMB-DS IPC$ unicode share access	1196
NETBIOS SMB-DS srvsvc NetrPathCanonicalize unicode little endian overflow attempt	1073
WEB-IIS view source via translate header	61
BAD-TRAFFIC tcp port 0 traffic	26
SNMP request tcp	6
SNMP AgentX/tcp request	5
SNMP trap tcp	4
Total	**3828**

Tabla 22. Alertas ICMP

Alertas	Total
ICMP Destination Unreachable Communication Administratively Prohibited	37956
ICMP PING CyberKit 2.2 Windows	3870
ICMP redirect host	2343
ICMP redirect net	1287
ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited	681
ICMP PING NMAP	77
ICMP webtrends scanner	13
ICMP Destination Unreachable Communication with Destination Network is Administratively Prohibited	189
ICMP PING speedera	4
ICMP Source Quench	**3**
ICMP PATH MTU denial of service	3
Total	**46420**

2. Direcciones IP que han accedido a los Honeypots.

Las direcciones IP señaladas con rojo, están reportadas en listas negras de Spam, la manera en la que han accedido es vía DLS y Dial Up.

Las direcciones de la UTPL que han accedido a los Honeypots son:

- ✓ 200.0.29.83
- ✓ 200.0.30.40
- ✓ 200.0.29.66
- ✓ 200.0.29.3

Existen algunas páginas que permiten conocer a quien pertenece una determinada dirección IP, las páginas que se han utilizado son:

- ✓ ARIN WHOIS Database Search
  https://ws.arin.net/whois/?queryinput=218.63.198.93

- ✓ LACNIC - Latin American and Caribbean Internet Addresses Registry
  http://lacnic.net/cgi-bin/lacnic/whois?lg=EN

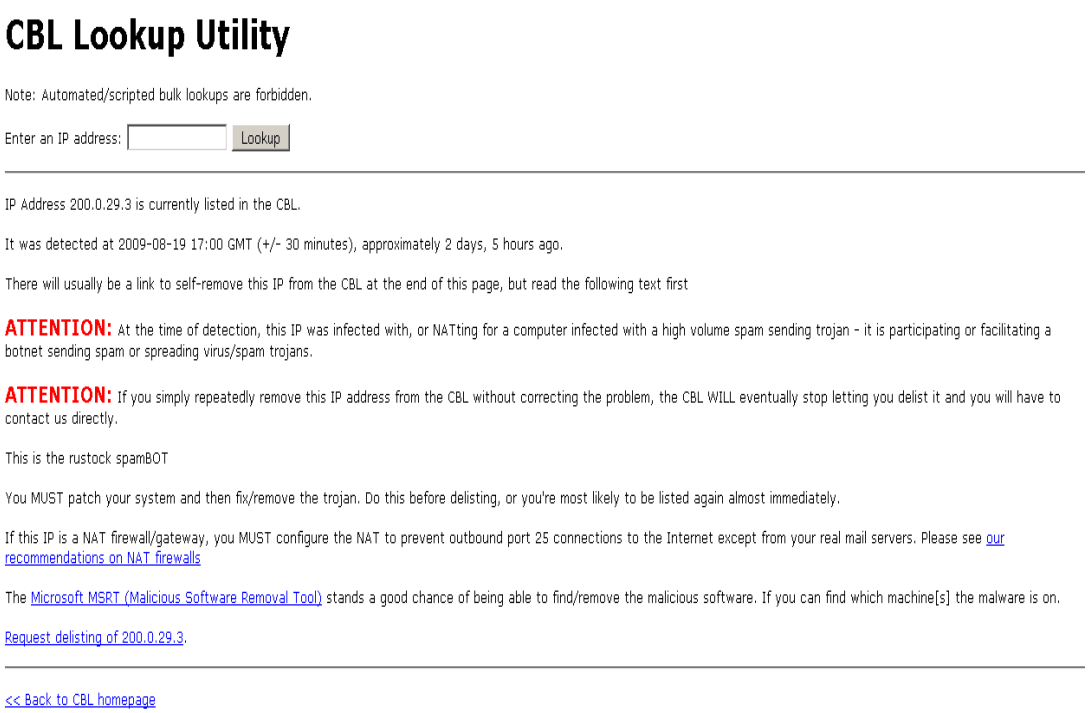
- ✓ APNIC - Query the APNIC Whois Database
  http://wq.apnic.net/apnic-bin/whois.pl

- ✓ What is my ip address
  http://whatismyipaddress.com/staticpages/index.php/lookup-results

Para saber si las direcciones se encuentran en las listas negras en la página ***What is my ip address*** hay una sección en la que se presenta si la dirección está reportada en listas negras.

La dirección **200.0.29.3** de acuerdo a esta página está reportada en las siguientes listas negras:

- ✓ nsbl.sorbs.net
- ✓ cbl.abuseat.org
- ✓ xbl.spamhaus.org
- ✓ web.dnsbl.sorbs.net



CBL Lookup Utility

Note: Automated/scripted bulk lookups are forbidden.

Enter an IP address: Lookup

IP Address 200.0.29.3 is currently listed in the CBL.

It was detected at 2009-08-19 17:00 GMT (+/- 30 minutes), approximately 2 days, 5 hours ago.

There will usually be a link to self-remove this IP from the CBL at the end of this page, but read the following text first

ATTENTION: At the time of detection, this IP was infected with, or NATting for a computer infected with a high volume spam sending trojan - it is participating or facilitating a botnet sending spam or spreading virus/spam trojans.

ATTENTION: If you simply repeatedly remove this IP address from the CBL without correcting the problem, the CBL WILL eventually stop letting you delist it and you will have to contact us directly.

This is the rustock spamBOT

You MUST patch your system and then fix/remove the trojan. Do this before delisting, or you're most likely to be listed again almost immediately.

If this IP is a NAT firewall/gateway, you MUST configure the NAT to prevent outbound port 25 connections to the Internet except from your real mail servers. Please see our recommendations on NAT firewalls

The Microsoft MSRT (Malicious Software Removal Tool) stands a good chance of being able to find/remove the malicious software. If you can find which machine[s] the malware is on.

Request delisting of 200.0.29.3.

<< Back to CBL homepage

Figura 27. Búsqueda de direcciones IP registradas en listas negras

Tabla 23. Direcciones IP registradas en el Honeywall

Direcciones	Organización	País	Conex. Diarias
200.179.95.132	Embratel	Brasil	1269
64.210.19.210	Global Crossing	EEUU	901
85.185.146.4	ISDP	Irán	809
194.109.21.230	XS4ALL Servers	Países Bajos	768
64.18.128.86	RackVibe LLCProxy:None detected	EEUU	644
89.118.98.51	FIAT S.P.A.	Italia	444
85.42.132.98	ADI S.P.A.	Italia	441
217.168.95.245	NTEBB IRC Servernet	Noruega	419
161.53.178.240	CARNet backbone	Croacia	415
195.18.164.194	BaneTele AS	Noruega	412
194.109.20.90	XS4ALL Servers	Países Bajos	408
202.32.71.80	MURATA MACHINERY, LTD.	Japón	394
195.197.175.21	Saunalahti Group Oyj	Finlandia	391
217.31.48.106	Ignum s.r.o	Rep. Checa	385
85.205.24.234	Vodafone Group Services	Reino unido	381
194.42.125.194	TheCloud	Reino unido	342
81.17.16.18	ISP Relians	Rusia	333
208.83.20.130	Desync Networks	EEUU	319
199.109.4.34	NYSERNET	EEUU	301
198.108.92.131	MichNet	EEUU	290

84.74.121.36	Cablecom GmbH	Zúrich	270
200.103.48.147	Brasil Telecom S/A - Filial Distrito Federal	Brasil	263
203.181.248.217	Asia Pacific Advanced Network - Japan	Japón	261
148.74.3.3	SPACENET		260
207.231.245.131	Pacific Wave	EEUU	258
198.124.216.209	Energy Sciences Network	EEUU	257
209.124.191.163	Pacific Northwest Gigapop	EEUU	241
199.109.4.14	NYSERNET	EEUU	230
217.204.60.82	Easynet	Reino Unido	230
216.27.100.74	MAGPI c/o University of Pennsylvania	EEUU	226
4.68.63.21	Level 3 Communications	EEUU	213
4.68.63.17	Level 3 Communications	EEUU	208
217.204.60.90	Easynet	Reino Unido	195
4.68.110.17	Level 3 Communications	EEUU	184
69.16.172.40	Easynews	EEUU	181
69.16.172.34	Easynews	EEUU	178
81.17.16.18	ISP Relians	Rusia	172
4.71.4.181	Level 3 Communications	EEUU	149
82.196.213.250	oslo.no.undernet.org	Noruega	146
4.58.108.154	Level 3 Communications	EEUU	143
133.24.225.161	Yamagata University	Japón	137
64.76.221.173	ImpSat Ecuador	Ecuador	134
189.3.174.136	Embratel	Brasil	124
220.189.222.68	Cixi jinlun Company	China	123
61.191.191.73	CHINANET Anhui province network	china	120
94.80.175.219	CAR DIESEL SPA	Italia	116
64.152.218.5	Level 3 Communications	EEUU	115
131.144.101.38	State Board of Regents	EEUU	114
194.70.98.234	Demon Internet	Reino unido	114
88.42.235.110	Interbusiness	Italia	114
193.52.8.195	Metropolitan Network Toulouse City	Francia	113
59.185.97.134	Mahanagar Telephone Nigam Ltd.	India	112
81.216.5.12	Sjofartsverket	Suecia	107
89.43.216.97	SC Bluelink SRL	Rumania	105
85.35.136.166	Telecom Italia SpA	Italia	104
61.151.254.122	Beijing Arctic Ice Sci-Tech Developing Co.,Ltd.	China	98
142.92.10.113	Communications Research Centre of Canada	Canadá	98
87.85.236.226	VicorpGr	Reino Unido	98
142.66.16.9	University of Lethbridge	Canadá	97
201.78.134.100	Tele Norte Leste Participações S.A.	Brasil	95
202.211.0.240	Tohoku Open Internet Community	Japón	93
192.148.242.195	OARnet	EEUU	91

201.57.19.11	Embratel	Brasil	91
213.200.84.1	Tiscali International Network B.V.	Alemania	91
199.109.11.14	NYSERNET	EEUU	88
200.179.95.132	Embratel	Brasil	87
193.55.215.29	Academic metropolitan network of Lyons	Francia	83
144.223.241.77	Sprint/United Information Service	EEUU	81
200.23.60.109	Corporacion Universitaria para el Desarrollo de In	México	79
202.112.36.54	CERNET super computer center	China	79
201.48.217.144	COMPANHIA DE TELECOM. DO BRASIL CENTRAL	Brasil	77
133.81.127.250	Fukuoka University of Education	Japón	73
71.118.223.26	Verizon Internet Services	EEUU	72
152.76.0.194	RPAH	Australia	71
163.9.15.2	Campus CNRS d'Orleans	Francia	71
122.224.54.82	CHINANET-ZJ Shaoxing node network	China	64
206.196.178.11	University of Maryland Network Operations Center	EEUU	60
195.111.97.242	HBONE ATP PVCs	Hungría	59
207.166.50.26	Sacramento County Office of Education	EEUU	59
137.164.26.10	ISP:CENICOrganization:CENICProxy:None detectedType:CorporateBlacklist:	EEUU	58
217.35.211.106	BT-DATADIAL	Reino Unido	58
188.1.231.214	IP networking on DFN's Wissenschaftsnetz X-WiN	Alemania	57
198.32.166.118	EP.NET, LLC.	EEUU	56
137.164.27.138	ISP:CENICOrganization:CENICProxy:None detectedType:CorporateBlacklist:	EEUU	55
200.249.200.4	Intertrade Despachos Aduaneiros e Serv. Mar. Ltda	Brasil	55
66.97.23.26	ORANO	Canadá	55
198.32.252.250	EP.NET, LLC.	EEUU	54
200.35.206.198	Supercable	Venezuela	54
150.99.187.221	fukuoka-dc-rm-ge-7-1-0-103.sinet.ad.jp	Japón	52
172.27.80.18	dir. Privada		52
220.233.2.45	Exetel	Australia	52
83.221.128.41	Customer point-to-point addresses - not for genera	Dinamarca	52
83.206.56.129	PRODUCTEUR BOULANGERIE INDUSTRIELLE	Francia	51
163.5.255.254	Ecole Pour l'Informatique et les Techniques Avance	Francia	50
200.90.64.201	CANTV Servicios, Venezuela	Venezuela	49
213.144.181.26	TI Sparkle Seabone Palermo POP		49
208.44.132.97	Qwest Communications	EEUU	47
200.252.154.3	PK Decorações Ltda.	Brasil	45

209.124.181.149	Pacific Northwest Gigapop	EEUU	44
131.144.101.17	State Board of Regents	EEUU	44
210.173.83.5	GS Information Network Co.,Ltd.	Japón	44
216.11.250.2	Oakland Schools	EEUU	43
200.202.201.196	Micropic Ltda	Brasil	42
207.61.39.242	Corp. of the City of Toronto	Canadá	42
200.45.119.201	Colsecor Ltda.	Argentina	41
62.40.124.158	DANTE Ltd.	Reino Unido	41
130.227.0.82	Tele2 A/S	Dinamarca	40
188.1.230.242	IP networking on DFN's Wissenschaftsnetz X-WiN	Alemania	40
192.88.115.24	Pittsburgh Supercomputing Center	EEUU	40
201.12.62.11	Intelig Telecomunicações Ltda.	Brasil	40
218.61.79.34	CNCGROUP Liaoning province network	China	39
87.237.114.38	JSC TKT	Rusia	39
142.13.148.41	Brandon University	Canadá	37
138.86.3.1	University of Northern Colorado	EEUU	35
74.208.125.21	1&1 Internet	EEUU	35
92.46.181.169	JSC Kazakhtelecom, Pavlodar Affiliate	Kazakstán	35
92.46.182.42	JSC Kazakhtelecom, Pavlodar Affiliate	Kazakstán	34
74.213.177.181	Momentum Advanced Solution	Canadá	33
200.231.59.9	FC Consultoria Informática e Comércio Ltda	Brasil	33
125.89.77.122	ChinaNet Guangdong Province Network	China	32
198.104.184.58	Verio Web Hosting	EEUU	32
67.38.100.97	Chicago Admin	EEUU	32
221.201.148.165	CNCGROUP Liaoning province network	China	31
195.2.12.148	Cable & Wireless Telecommunication Services GmbH	Europa	30
205.213.118.5	WiscNet	EEUU	30
66.109.25.148	Galaxyvisions	EEUU	30
74.3.205.220	74.3.205.220.reverse.gogrid.com		30
91.124.19.15	Ukrtelecom IP access network	Ucrania	30
200.90.65.145	CANTV Servicios, Venezuela	Venezuela	29
200.71.170.213	TELCEL - BANDA ANCHA- 1XRTT	Venezuela	28
147.46.254.110	Seoul National University	Corea	27
200.106.221.145	SUPERCABLE TELECOMUNICACIONES	Colombia	26
192.5.89.102	Harvard University	EEUU	26
192.5.89.22	Harvard University	EEUU	26
199.109.4.74	NYSERNET	EEUU	26
124.160.44.218	CNC Group Zhejiang province network	China	25
192.42.152.97	University of Minnesota	EEUU	25
200.75.249.189	Cable Onda	Panamá	25

61.141.8.5	ChinaNet Guangdong Province Network	China	25
66.97.28.66	ORANO	Canadá	25
83.110.67.225	Emirates Telecommunications Corporation	Emiratos Arb.	25
83.149.193.129	Network of the Center of Scientific	Rusia	24
150.99.188.153	osaka-dc-rm-ge-5-2-0-109.sinet.ad.jp	Japón	23
210.7.32.25	Research and Educationa Advanced Network New Zeala	N. Zelanda	23
198.22.64.34	Lodden Technology	EEUU	22
200.35.205.107	Supercable	Venezuela	22
58.9.26.165	True Internet Co.	Tailandia	22
189.105.165.137	Tele Norte Leste Participações S.A.	Brasil	21
189.68.165.219	TELECOMUNICACOES DE SAO PAULO S.A. -	Brasil	21
200.32.116.217	Coop. de Emp. Multiples Sudecor	Argentina	21
217.165.18.97	Emirates Telecommunications Corporation	Unión Emiratos Árabes	21
199.109.9.46	NYSERNET	Reino Unido	20
200.0.25.60	Ministerio de Cultura, CUBARTE	Cuba	20
74.213.177.75	Momentum Advanced Solution	Canadá	20
207.210.143.186	Harvard/NOX Router to Router /30's	EEUU	19
207.68.183.120	Microsoft Corp	EEUU	19
95.58.139.75	JSC Kazakhtelecom, Pavlodar Affiliate	Kazakstán	19
61.151.254.122	Beijing Arctic Ice Sci-Tech Developing Co.,Ltd.	China	18
218.108.238.140	WASU TV & Communication Holding Co.,Ltd.	China	18
200.74.140.142	Telmex Colombia S.A.	Colombia	18
201.30.152.196	Embratel	Brasil	18
200.112.138.242	Broadbandtech S. A.	Argentina	17
200.90.77.212	CANTV Servicios, Venezuela	Venezuela	17
92.46.178.97	JSC Kazakhtelecom, Pavlodar Affiliate	Kazakstán	17
95.58.143.103	JSC Kazakhtelecom, Pavlodar Affiliate	Argentina	17
209.210.249.181	Integra Telecom	EEUU	16
200.41.59.147	roemmers	Argentina	16
200.49.17.15	Plug and play Net S.A.	Chile	16
38.104.58.6	Performance Systems International	EEUU	16
82.138.56.170	Network for OOO Teleservice Internet Service Provi	Rusia	16
92.46.182.155	JSC Kazakhtelecom, Pavlodar Affiliate	Kazakstán	16
95.58.138.85	JSC Kazakhtelecom, Pavlodar Affiliate	Kazakstán	16
200.112.148.113	Broadbandtech S. A.	Argentina	15
202.170.127.165	Proen Internet, Internet Service Provider, Bangkok	Tailandia	15
196.218.29.69	Ramsis-Zone-DSL	Egipto	15
200.119.18.182	:ETB - Colombia	Colombia	15

200.123.124.66	Techtel LMDS Comunicaciones Interactivas S.A.	Argentina	15
200.123.69.151	Techtel LMDS Comunicaciones Interactivas S.A.	Argentina	15
200.172.87.52	Embratel	Brasil	15
200.87.167.67	Entel S.A. - EntelNet	Bolivia	15
62.132.250.2	Gene it Solutions	Países Bajos	15
91.124.9.42	Ukrtelecom IP access network	Ucrania	15
200.123.122.214	Techtel LMDS Comunicaciones Interactivas S.A.	Argentina	14
194.46.1.114	Belfast LAN Infrastructure	Reino unido	14
200.172.34.67	Embratel	Brasil	14
200.51.81.128	Advance Telecomunicaciones S.A.	Argentina	14
200.90.64.144	CANTV Servicios, Venezuela	Venezuela	14
201.78.143.122	Tele Norte Leste Participações S.A.	Brasil	14
58.221.34.126	CHINANET jiangsu province network	China	14
60.28.2.77	Sina Limited company	China	14
119.146.75.132	ChinaNet Guangdong Province Network	China	13
119.167.246.138	CNCGROUP Shandong province network	China	12
116.71.10.52	PTCL Triple Play Project	Pakistán	12
173.45.76.138	eNET	EEUU	12
200.202.12.39	MATRIX INTERNET S.A.	Brasil	12
200.85.34.46	Telecel S.A.	Paraguay	12
218.111.235.25	Telekom Malaysia Berhad	Malasia	12
200.193.48.42	Brasil Telecom S/A - Filial Distrito Federal	Brasil	11
93.157.184.159	Prov.RU	Rusia	10
200.255.88.134	AGORA CTVM S/A	Brasil	10
58.9.30.174	True Internet Co.	Tailandia	10
74.86.203.123	uniaohost	EEUU	10
200.150.146.27	Dominio BR Consultoria em Informatica LTDA	Brasil	9
200.180.54.130	DATA CENTER INTERNET LTDA	Brasil	2

- Las direcciones subrayadas, se encuentran en listas negras antispam.

ANEXO E

Archivo de configuración *honeywall.conf*

```
###########################################################################
#
# $Id: honeywall.conf 4552 2006-10-17 01:06:51Z esammons $
#
###########################################
#
# Copyright (C) <2005> <The Honeynet Project>
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or (at
# your option) any later version.
#
# This program is distributed in the hope that it will be useful, but
# WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
# General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
# USA
#
###########################################

#
# This file is the Honeywall import file (aka "honeywall.conf").
# It is a list of VARIABLE=VALUE tuples (including comments as
# necessary, # such as this) and whitespace lines.
#
# note: DO NOT surround values in quotation marks
#
```

```
##############################################################################

############################
# Site variables that are  #
# global to all honeywalls #
# at a site.               #
############################

# Specify the IP address(es) and/or networks that are allowed to connect
# to the management interface.  Specify any to allow unrestricted access.
# [Valid argument: IP address(es) | IP network(s) in CIDR notation | any]
HwMANAGER=172.16.0.0/16

# Specify the port on which SSHD will listen
# NOTE: Automatically aded to the list of TCP ports allowed in by IPTables
# [Valid argument: TCP (port 0 - 65535)]
HwSSHD_PORT=22

# Specify whether or not root can login remotely over SSH
# [Valid argument: yes | no]
HwSSHD_REMOTE_ROOT_LOGIN=yes

# NTP Time server(s)
# [Valid argument: IP address]
HwTIME_SVR=


############################
# Local variables that are #
# specific to each         #
# honeywall at a site.     #
############################

# Specify the system hostname
# [Valid argument: string ]
HwHOSTNAME=localhost

# Specify the system DNS domain
# [Valid argument: string ]
HwDOMAIN=localdomain

#Start the Honeywall on boot
# [Valid argument: yes | no]
HwHONEYWALL_RUN=yes

# To use a headless system.
# [Valid argument: yes | no]
HwHEADLESS=no


# This Honeywall's public IP address(es)
# [Valid argument: IP address | space delimited IP addresses]
HwHPOT_PUBLIC_IP=200.0.30.50 200.0.30.51 200.0.30.52

# DNS servers honeypots are allowed to communicate with
# [Valid argument: IP address | space delimited IP addresses]
HwDNS_SVRS=200.0.31.155

# To restrict DNS access to a specific honeypot or group of honeypots, list
# them here, otherwise leave this variable blank
# [Valid argument: IP address | space delimited IP addresses | blank]
HwDNS_HOST=200.0.30.50 200.0.30.51 200.0.30.52

# The name of the externally facing network interface
# [Valid argument: eth* | br* | ppp*]
HwINET_IFACE=eth0
```

```
# The name of the internally facing network interface
# [Valid argument: eth* | br* | ppp*]
HwLAN_IFACE=eth1

# The IP internal connected to the internally facing interface
# [Valid argument: IP network in CIDR notation]
HwLAN_IP_RANGE=200.0.30.48/29

# The IP broadcast address for internal network
# [Valid argument: IP broadcast address]
HwLAN_BCAST_ADDRESS=200.0.30.255

# Enable QUEUE support to integrate with Snort-Inline filtering
# [Valid argument: yes | no]
HwQUEUE=yes

# The unit of measure for setting oubtbound connection limits
# [Valid argument: second, minute, hour, day, week, month, year]
HwSCALE=hour

# The number of TCP connections per unit of measure (HwScale)
# [Valid argument: integer]
HwTCPRATE=20

# The number of UDP connections per unit of measure (HwSCALE)
# [Valid argument: integer]
HwUDPRATE=20

# The number of ICMP connections per unit of measure (HwSCALE)
# [Valid argument: integer]
HwICMPRATE=50

# The number of other IP connections per unit of measure (HwSCALE)
# [Valid argument: integer]
HwOTHERRATE=10

# Enable the SEBEK collector which delivers keystroke and files
# to a remote system even if an attacker replaces daemons such as sshd
# [Valid argument: yes | no]
HwSEBEK=yes

# Enable the Walleye Web interface.
#[Valid argument: yes | no]
HwWALLEYE=yes

# Specify whether whether to drop SEBEK packets or allow them to be sent
# outside of the Honeynet.
# [Valid argument: ACCEPT | DROP]
HwSEBEK_FATE=ACCEPT

# Specify the SEBEK destination host IP address
# [Valid argument: IP address]
HwSEBEK_DST_IP=0.0.0.0

# Specify the SEBEK destination port
# [Valid argument: port]
HwSEBEK_DST_PORT=1101

# Enable SEBEK logging in the Honeywall firewall logs
# [Valid argument: yes | no]
HwSEBEK_LOG=yes


# Specify whether the dialog menu is to be started on login to TTY1
# [Valid argument: yes | no ]
HwMANAGE_DIALOG=yes
```

```
# Specify whether management port is to be activated on start or not.
# [Valid argument: yes | no ]
HwMANAGE_STARTUP=yes

# Specy the network interface for remote management.  If set to br0, it will
# assign MANAGE_IP to the logical bridge interface and allow its use as a
# management interface.  Set to none to disable the management interface.
# [Valid argument: eth* | br* | ppp* | none]
HwMANAGE_IFACE=eth2

# IP of management Interface
# [Valid argument: IP address]
HwMANAGE_IP=172.16.71.222

# Netmask of management Interface
# [Valid argument: IP netmask]
HwMANAGE_NETMASK=255.255.255.0

# Default Gateway of management Interface
# [Valid argument: IP address]
HwMANAGE_GATEWAY=172.16.71.10

# DNS Servers of management Interface
# [Valid argument: space delimited IP addresses]
HwMANAGE_DNS=200.0.31.155

# TCP ports allowed into the management interface.
# Do NOT include the SSHD port.  It will automatically be included
# [Valid argument: space delimited list of TCP ports]
HwALLOWED_TCP_IN=443

# Specify whether or not the Honeywall will restrict outbound network
# connections to specific destination ports.  When bridge mode is utilized,
# a management interface is required to restrict outbound network connections.
# [Valid argument: yes | no]
HwRESTRICT=yes

# Specity the TCP destination ports Honeypots can send network traffic to.
# [Valid argument: space delimited list of UDP ports]
HwALLOWED_TCP_OUT=22 23 25 43 80 443

# Specity the UDP destination ports Honeypots can send network traffic to.
# [Valid argument: space delimited list of UDP ports]
HwALLOWED_UDP_OUT=53 123

# Specify whether or not to start swatch and email alerting.
# [Valid argument: yes | no]
HwALERT=yes

# Specify email address to use for email alerting.
# [Valid argument: any email address]
HwALERT_EMAIL=hendryfer@gmail.com

# NIC Module List - Set this to the number and order you wish
# to load NIC drivers, such that you get the order you want
# for eth0, eth1, eth2, etc.
# [Valid argument: list of strings]
#
# Example: eepro100 8139too
HwNICMODLIST=

# Blacklist, Whitelist, and Fencelist features.
# [Valid argument: string ]
HwFWBLACK=/etc/blacklist.txt

# [Valid argument: string ]
HwFWWHITE=/etc/whitelist.txt
```

```
# [Valid argument: string ]
HwFWFENCE=/etc/fencelist.txt

# [Valid argument: yes | no]
HwBWLIST_ENABLE=no

# [Valid argument: yes | no]
HwFENCELIST_ENABLE=no

# The following feature allows the roo to allow attackers into the
# honeypots but they can't send packets out...
# [Valid argument: yes | no]
HwROACHMOTEL_ENABLE=no

# Disables BPF filtering based on the contents of HwHPOT_PUBLIC_IP
# and the black and white list contained within HwFWBLACK and HwFWWHITE
# if the HwBWLIST_ENABLE is on.  Other wise, it just filters based on
# the contents of HwHPOT_PUBLIC_IP
# [Valid argument: yes | no]
HwBPF_DISABLE=no

# This capability is not yet implemented in roo.  The variable
# has been commented out for this reason. dittrich - 02/08/05
# Options for hard drive tuning (if needed).
# [Valid argument: string ]
# Example: -c 1 -m 16 -d
HwHWPARMOPTS=

# Should we swap capslock and control keys?
HwSWAP_CAPSLOCK_CONTROL=no

##############################################################################
# Snort Rule Update Variables
##############################################################################
# Enable or disable automatic snort rule updates
# [Valid argument: yes | no]
HwRULE_ENABLE=no

# Automatically restart snort and snort_inline when automatic updates are
# applied and when calls to update IDS or IPs rules?
# [Valid argument: yes | no]
HwSNORT_RESTART=no

# Oink Code - Required by Oinkmaster to retrieve VRT rule updates
# See: /hw/docs/README.snortrules or
#      http://www.honeynet.org/tools/cdrom/roo/manual/
# for instructions on how to obtain it (Free registration).
# [Valid argument: ~40 char alphanum string]
HwOINKCODE=

# Day automatic snort rule updates should be retrieved (for weekly updates)
# For daily updates, set this to ""
# [Valid argument: sun | mon | tue | wed | thu | fri | sat]
HwRULE_DAY=sat

# Hour of day snort rules updates should be retrieved
# [Valid argument: 0 | 1 | 2 | ... | 23] (0 is Midnight, 12 is noon, 23 is 11PM)
HwRULE_HOUR=3

##############################################################################
# Pcap and DB data retention settings
# Currenrly ONLY used when Pcap/DB purge scripts are called
# Pcap/DB data *is NOT* automatically purged
##############################################################################
# Days to retain Pcap data.  This will be used *IF* /dlg/config/purgePcap.pl
# is called with NO arguments.
```

```
# NOTE: Override this by supplying the number of days as an argument ala:
# /dlg/config/purgePcap.pl <days>
HwPCAPDAYS=45

# Days to retain DB data.  This will be used *IF* /dlg/config/purgeDB.pl
# is called with NO arguments.
# NOTE: Override this by supplying the number of days as an argument ala:
# /dlg/config/purgeDB.pl <days>
HwDBDAYS=180

##############################################################################
# NAT mode is no longer supported.
# Don't mess with anything below here unless you know what you're
# doing! Don't say we didn't warn you, and don't try logging a bugzilla
# request to clean up the mess!
##############################################################################

# Space delimited list of Honeypot ips
# NOTE: MUST HAVE SAME NUMBER OF IPS AS PUBLIC_IP VARIABLE.
# [Valid argument: IP address]
#HwHPOT_PRIV_IP_FOR_NAT=

# Specify the IP address of the honeywall's internal (i.e. gateway
# IP for NAT) IP address.  This is only used in NAT mode.
# [Valid argument: IP address ex: 192.168.10.1]
#HwPRIV_IP_FOR_NAT=


# Specify the IP netmask for interface alises.  One aliases will be created
# on the external interface for each Honeypot when in NAT mode only.
# [Valid argument: IP netmask]
#HwALIAS_MASK_FOR_NAT=255.255.255.0



# End of honeywall.conf parameters

#
# Newly defined variables as of Wed Jul 22 16:39:16 GMT 2009
#
HwHFLOW_DB=3
HwSENSOR_ID=945782425
```

ANEXO F
Imagen del Honeywall

```
#! /bin/bash

hostname >> imagen
date -R >> imagen
fdisk -l >> imagen
df -Th >> imagen
ps aux >> imagen
route -nee >> imagen
cp /var/log/messages paso | cat paso >> imagen
cp /var/log/iptables paso | cat paso >> imagen
cp /etc/hosts.deny paso | cat paso >> imagen
cp /etc/hosts.allow paso | cat paso >> imagen
```