Proyecto

Honeypots

Fecha: 18/04/2008
Versión: 6.0

Docentes

Responsable: Dr. Ing. Gustavo Betarte
Tutor: Ing. Alejandro Blanco
Cotutor: Ing. Marcelo Rodriguez

Integrantes

Fernando Cócaro
Mauricio García
Maria Jose Rouiller

Grupo de Seguridad Informática — Instituto de Computación — Facultad de Ingeniería — Universidad de la República

RESUMEN

El presente documento es el informe final correspondiente al proyecto de grado P2007_0028, "*Diseño e Implementación de un Honeypot*" del año 2007. El proyecto consistió en desarrollar un estudio del arte de la tecnología de *honeypots* y en la implantación de un *honeypot* en la Facultad de Ingeniería de la UdelaR.

Un *honeypot* es un señuelo implantado en una red con múltiples propósitos. Por ejemplo, distraer la atención de los atacantes (evitando que comprometan sistemas de valor), proveer alertas tempranas sobre las tendencias de ataques o exploits (programas o técnicas que aprovechan una vulnerabilidad de una aplicación) utilizados o estudiar las técnicas y herramientas utilizadas por los atacantes. Se clasifican principalmente por la veracidad de los servicios que dispone, denominado nivel de interacción. Dependiendo de sus características la implantación puede ser compleja.

En el contexto del proyecto de grado, el grupo se avocó a la creación de una red de *honeypots.* Dichas arquitecturas son denominadas *honeynets*. La particularidad de la red construida reside en la facilidad de despliegue, la posibilidad de integrar a la red implementaciones de *honeypots* con distintas capacidades de captura de datos y niveles de interacción, minimización del impacto de agregar más *honeypots* a la red ya implantada y en la centralización de la información obtenida. Además incluye un sistema de alarmas para notificar a los administradores de la actividad en la red y un sistema de monitoreo para trabajar con los datos recolectados por los *honeypots*. De esta forma se diseña así una solución flexible y escalable que permite desplegar en forma sencilla una *honeynet*.

Para la implementación de la solución se hace necesario llevar los datos obtenidos a una representación unificada, por lo que se estudia y utiliza el modelo de datos *IDMEF* [NWGp07] para realizar dicha normalización. También se analizan las ventajas de distribuir los componentes de la solución en formato de *livecd* para potenciar y facilitar el despliegue de la arquitectura. Adicionalmente se determina la necesidad de virtualizar los componentes de la solución, facilitando su movilidad y despliegue.

Luego de su implantación, la solución se mantiene operativa recolectando la información de los ataques recibidos. A partir de ello se realiza un análisis de los datos obtenidos, el cual brinda información sobre la actividad de los atacantes. Se identifican diferentes ataques y las regiones desde donde se originan los mismos.

Palabras claves: *honeypot, honeynet, virtualización, livecd, IDMEF*.

Índice

Resumen....3
Agradecimientos....11
Capítulo 1 Introducción....13
1.1 Problema....14
1.2 Motivación....15
1.3 Objetivos....15
1.4 Organización....16
1.5 Alcance del proyecto....16
1.6 Resultados obtenidos....17
1.7 Estructura del documento....17
Capítulo 2 Estado del Arte de Honeypots....19
2.1 Introducción....19
2.2 Aprendiendo del enemigo....20
2.2.1 Objetivos....21
2.2.2 Desafíos....21
2.2.3 Clasificación....22
2.2.3.1 Funcionalidad....22
2.2.3.2 Nivel de interacción....22
2.2.3.3 Nivel de implantación....23
2.2.4 Captura de datos....23
2.2.4.1 Honeypots de bajo nivel de interacción....23
2.2.4.2 Honeypots de alto nivel de interacción....24
2.3 Honeynets....24
2.3.1 Arquitectura de una Honeynet....25
2.3.1.1 Generación I....25
2.3.1.2 Generación II....26
2.3.1.3 Generación III....27
2.3.1.4 Generación IV....28
2.4 Implementaciones de honeypots....28
2.4.1 Deception Toolkit (DTK)....28
2.4.2 Back Officer Friendly (BOF)....29
2.4.3 Specter....29
2.4.4 Symantec Decoy Server....30
2.4.5 LaBrea Tarpit....30
2.4.6 Honeytrap....31
2.4.7 Honeyd....32
2.4.8 PHP.Hop....33
2.4.9 The Google Hack Honeypot (GHH)....33
2.5 Conclusión....35
Capítulo 3 Análisis y Diseño....37
3.1 Puntos a tener en cuenta....37
3.2 Arquitectura básica....39

3.3 Normalización....40
3.3.1 Intrusion Detection Message Exchange Format....42
3.3.1.1 Aplicación de IDMEF....44
3.4 Centralización de datos....45
3.4.1 Repositorio de datos....48
3.5 Monitoreo y Análisis....48
3.6 Alarmas....49
3.7 Seguridad....50
3.8 Integración de los componentes....52
3.9 Conclusión....53
Capítulo 4 Implementación....55
4.1 Captura de datos....55
4.2 Normalización y Centralización....56
4.3 Monitoreo de datos....59
4.4 Generación de alarmas....60
4.5 Integración de componentes....61
4.6 Despliegue de la honeynet....61
4.6.1 Despliegue de los componentes de la honeynet....63
Capítulo 5 Implantación....65
5.1 Host base....65
5.2 HoneyCell....70
5.2.1 Captura de Datos con Honeytrap....71
5.2.2 Captura de Datos con Honeyd....72
5.2.2.1 Otras configuraciones....73
5.2.3 Proceso de Normalización....73
5.3 HoneyCenter....74
5.4 Honeywall....78
5.5 Terminal de Administración....79
5.6 Conclusión....80
Capítulo 6 Resultados obtenidos....81
6.1 Capa de datos de HoneyCell....81
6.1.1 Ataques recibidos....81
6.1.2 Objetivos de los ataques....84
6.1.3 Origen de los ataques....87
6.2 Capa de datos de Honeywall....87
6.2.1 Ataques registrados....87
6.2.2 Análisis de tráfico de ataques....88
6.3 Capa de datos de HoneyCenter....91
6.4 Conclusión....91
Capítulo 7 Gestión del proyecto....93
7.1 Organización....93
Capítulo 8 Conclusiones....97
8.1 Conclusiones obtenidas....97
8.2 Problemas encontrados....99
8.3 Lecciones aprendidas....101

Capítulo 9 Trabajos a futuro....103
9.1 Estudio de comportamiento de intrusos....103
9.2 Extender la centralización de datos....103
9.3 Normalización y Centralización de datos capturados....104
9.4 Presentación de datos refinada....104
9.5 Refinamiento del Sistema de alarmas....104
9.6 Incorporar nuevas implementaciones de honeypots....105
9.7 Implementación de honeynet de generación IV....105
9.8 Crear otras honeynets en Facultad....105
9.9 Ataques DoS....106
9.10 Wireless honeypots....106
9.11 Otros trabajos a futuro....107
Anexo A: Glosario....109
Referencias....117

Índice de imágenes

Imagen 2.1: Evolucion de herramientas y disminución del conocimiento requerido para realizar ataques.....20
Imagen 2.2: Esquema de una honeynet básica.....25
Imagen 2.3: Arquitectura Honeynet Gen I.....26
Imagen 2.4: Arquitectura Honeynet Gen II.....27
Imagen 2.5: Comunicación entre Sebek Server y Sebek Client.....27
Imagen 2.6: Esquema de una honeynet distribuida.....28
Imagen 2.7: Back Officer Friendly.....29
Imagen 2.8: Esquema de virtualización con SDS.....30
Imagen 2.9: Esquema de una Honeynet virtualizada con Honeyd.....33
Imagen 2.10: Esquema de sitio Web usando GHH.....34
Imagen 3.1: Propuesta de diseño.....38
Imagen 3.2: Arquitectura basica de la solución.....39
Imagen 3.3: Arquitectura donde cada componente se comunica con el repositorio.....40
Imagen 3.4: Esquema de normalización indirecta de logs generados.....41
Imagen 3.5: Estructura definida para la normalización.....42
Imagen 3.6:Principales clases del modelo IDMEF 43
Imagen 3.7: Esquema de transmisión de los datos capturados al repositorio.....46
Imagen 3.8: Componente: Sistema de alarmas.....50
Imagen 3.9: Honeywall. Modalidad de filtrado por conexión limitada. 51
Imagen 3.10: Honeywall. Control de datos utilizando el modo drop. 52
Imagen 3.11: Honeywall. Control de datos utilizando el modo de reemplazo. 52
Imagen 3.12: Arquitectura de la solución.....54
Imagen 4.1: Esquema envío de alertas mediante Syslog.....56
Imagen 4.2: Arquitectura de Prelude-IDS.....57
Imagen 4.3: Esquema envío de alertas mediante Prelude-IDS.....59
Imagen 4.4: Integración de los componentes seleccionados.....62
Imagen 4.5: Arquitectura de la solución virtualizada.....63
Imagen 5.1: Ubicación conceptual del servidor.....66
Imagen 5.2: Arquitectura virtual implantada.....67
Imagen 5.3: Parámetros de reporte “Gráfica – Evolución de Ataques a un Puerto”.....76
Imagen 5.4: Gráfica de evolución de ataques al puerto 1433.....76
Imagen 6.1: Ataques distribuidos en el período de análisis.....82
Imagen 6.2: Gráfica de los 10 puertos más atacados.....85
Imagen 6.3: Gráfica de evolución de ataques al puerto 1433.....86
Imagen 6.4: Gráfica de evolución de ataques al puerto 139.....86
Imagen 6.5: Gráfica de evolución de ataques al puerto 25.....86
Imagen 6.6: Zonas de origen de los ataques recibidos.....87
Imagen 7.1: Detalle de tareas de investigación.....93
Imagen 7.2: Detalle de tareas de fase de armado de propuesta y fase de Implantación.....95
Imagen 7.3: Detalle de tareas de documentación.....95

Índice de tablas

Tabla 2.1: Comparación entre niveles de interacción de honeypots.....23
Tabla 6.1: Lista de los 10 días con más ataques capturados.....82
Tabla 6.2: Distribución puertos objetivos de ataques correspondientes al 29/01/2008.....83
Tabla 6.3: Distribución de ataques por IP's de origen correspondientes en la fecha 29/01/2008.....83
Tabla 6.4: Distribución de puertos objetivos de ataques correspondientes al 05/01/2008.....83
Tabla 6.5: Origen y cantidad de acceso de ataques del día 29/01/2008.....83
Tabla 6.6: Origen y cantidad de acceso de ataques del día 01/01/2008.....83
Tabla 6.7: Origen y cantidad de acceso de ataques del día.....84
Tabla 6.8: Origen de los ataques según al 29/01/2008.....84
Tabla 6.9: Lista de los 10 puertos mas atacados.....85
Tabla 6.10: Cantidad de ataques por país.....88

Índice de fragmentos

Fragmento 2.1: Sección del archivo de log de Honeytrap....24
Fragmento 2.2: Cadena para buscar servidores SquerrelMail....34
Fragmento 2.3: Cadena para buscar logs de DrWatson....34
Fragmento 3.1: uso de IDMEF para reportar un ataque de port-scanning....44
Fragmento 4.1: Definición de la personalidad WinXP con un servidor Web en el puerto 80....56
Fragmento 4.2: Definición de una regla para Prelude-LML....58
Fragmento 5.1: Integridad del sistema, lista de sumas MD5....68
Fragmento 5.2: Iptables, registro de una conexión al puerto 22....69
Fragmento 5.3: Regla de SEC para identificar una conexión via SSH....70
Fragmento 5.4: Cuerpo del mensaje enviado al administrador....70
Fragmento 5.5: Registro del servicio de alarmas en el host base....70
Fragmento 5.6: Ejecución de Honeytrap....71
Fragmento 5.7: Honeytrap. Conexión al puerto TCP 139....72
Fragmento 5.8: Honeytrap. Transferencia de datos al puerto 22....72
Fragmento 5.9: Definición de la personalidad WinXP con un servidor Web en el puerto 80....72
Fragmento 5.10: Honeyd. Registro de una conexión establecida....73
Fragmento 5.11: Honeyd. Registro de una conexión finalizada....73
Fragmento 5.12: Honeyd. Registro de un paquete sin conexión establecida....73
Fragmento 5.13: Definición de la normalización con Prelude-LML....74
Fragmento 5.14: Configuración utilizada por SEC para el envío de alarmas....77
Fragmento 5.15: Regla de conexión vía SSH para la alarma local del repositorio....77
Fragmento 5.16: Parte de un archivo de configuración de Honeywall....78
Fragmento 5.17: Regla de iptables para limitar las conexiones entrantes desde un host....79
Fragmento 6.1: Ejemplo de string de conexión transmitido (1)....88
Fragmento 6.2: Ejemplo de string de conexión transmitido (2)....89
Fragmento 6.3: Ejemplo de string de conexión transmitido (3)....89
Fragmento 6.4: Ejemplo de string de conexión transmitido (4)....89
Fragmento 6.5: Ejemplo de transmisión al puerto TCP 139 (1)....90
Fragmento 6.6: Ejemplo de transmisión al puerto TCP 139 (2)....90
Fragmento 6.7: Ejemplo de transmisión al puerto TCP 25 (1)....90
Fragmento 6.8: Ejemplo de transmisión al puerto TCP 25 (2)....90

AGRADECIMIENTOS

A nuestros profesores, por el apoyo brindado, y por permitirnos desarrollar un proyecto más ambicioso al planteado inicialmente.

Este proyecto no hubiera sido posible sin la generosa colaboración de nuestros allegados, a quienes deseamos expresar nuestro agradecimiento.

A mis padres y hermanos que me apoyaron a lo largo de la carrera. Y muy especialmente a Verónica, por su comprensión durante el tiempo que le dediqué al proyecto. ¡Gracias!

Fernando Cócaro Mendoza

A Laura, a mis padres y hermanos que estuvieron a mi lado en toda la carrera brindándome su apoyo. Y en especial a mi hermano Rodrigo quien estuvo y estará por siempre junto a mí ...

Mauricio Fernando García Friguglietti

A mis amigos por su paciencia y comprensión de mi ausencia durante el proyecto. Y sobre todo a mi madre por su apoyo en cada uno de los pasos de mi carrera.

María José Rouiller Tagliani

CAPÍTULO 1 INTRODUCCIÓN

Internet es sin lugar a dudas uno de los medios de comunicación más utilizados y el que más expansión ha tenido en estos tiempos. Se le han dado distintas aplicaciones, que van por ejemplo desde la lectura de correos electrónicos hasta transacciones financieras. Sin embargo, este escenario no escapa a las entidades mal intencionadas cuyo fin es atentar contra los usuarios u organizaciones que hacen uso de este medio, por ejemplo realizando el robo de información o la destrucción de sistemas informáticos.

Con el objetivo de asegurar sus recursos y tratar de evitar ser blanco de ataques, usuarios y empresas invierten tiempo y dinero en distintas herramientas de seguridad, como por ejemplo firewalls, sistemas de detección de intrusos, mecanismos de cifrado, antivirus, etc. Éstas se basan en el enfoque de realizar acciones defensivas y asegurar la integridad de los sistemas, para mitigar vulnerabilidades que puedan ser explotadas por un atacante.

Un paradigma diametralmente opuesto es implantar un señuelo, un sistema "*vulnerable*" en el ambiente bajo estudio para que pueda ser atacado libremente. Así se engaña a los atacantes y se aprende de los ataques recibidos por estos.

El primer registro que se tiene de este modelo data aproximadamente del año 400 A.C., donde el general chino Sun Tzu en su libro "*El Arte de la guerra*" [SunT07] hace mención a una frase que 2400 años más tarde fuera tomada como estandarte de la tecnología, "*conoce a tu enemigo*", bajo este título se publicaron una serie de documentos en los que enseñan las herramientas, tácticas y motivos de los atacantes [KyEn00].

No fue hasta 1991 que se publican dos papers, "*The Cuckoos's Egg*" de Clifford Stoll [Clif95] y "*An Evening with Berferd*" de Bill Cheswick [Ches92], en los cuales se explican y aplican los conceptos de la tecnología de *honeypots*, aunque no se utiliza el término en sí.

"*Un honeypot es un recurso que simula ser un objetivo real, el cual se espera que sea atacado o comprometido. Los principales objetivos son el distraer a los atacantes y obtener información sobre el ataque y el atacant*e" [BaPl02].

"*El valor de un honeypot está en ser escaneado, atacado o comprometido*" [Spit01].

Aprender de ataques y atacantes permite mejorar la seguridad de una infraestructura informática, siendo una de las finalidades más conocidas que se le atribuyen. También permiten distraer al atacante de los servicios en producción y proveer información de que tan segura se encuentra la infraestructura. Algunas implementaciones de *honeypots* son capaces de hacer más lento un ataque [LbrT03] incluso hasta detenerlo.

El uso del término "*honeypot*" inicia próximo al año 2000, siendo en el correr del año 2002 cuando se reconoce la importancia de estas herramientas cuando un *honeypot* implantado en un sistema operativo Solaris captura el primer ataque conocido (en Internet) a una vulnerabilidad al servicio `dtspcd`[1][IISS06]. Desde entonces ha demostrado, a pesar de su relativa corta existencia en el

1 Se conocía la existencia de la vulnerabilidad, pero no se había visto un ataque en Internet aún.

ámbito informático, su valor como herramienta de investigación en el área de seguridad. Cada vez más investigadores y organizaciones hacen uso de esta tecnología para aprender las técnicas y procedimientos que utilizan los atacantes para irrumpir en los sistemas, buscando reforzar o mejorar las herramientas tradicionales de seguridad utilizadas.

Los *honeypots* no garantizan seguridad en la red que se encuentren, tampoco son mecanismos de defensa, sino que son herramientas diseñadas para obtener información que permitan reforzar la seguridad o en el mejor de los casos distraer a los atacantes de los sistemas en producción. Al recibir un ataque, un *honeypot* obtiene información que ayuda en la toma de decisiones para solucionar las carencias que se tengan en la red. Esta información luego puede ser utilizada para mitigar esa clase de ataque.

Según el tipo y uso del *honeypot*, depende la cantidad y granularidad de la información que se obtiene, pero se debe tener en cuenta que adquirir información más detallada puede llegar a introducir riesgos en la red donde se implante, ya que al atacante se le brinda un mayor nivel de interacción sobre el *honeypot*.

Como aplicación directa de la tecnología se diseñan redes de *honeypots* conocidas como *honeynets* [HoPr02]. Estas tienen la característica de ser altamente controladas para contener y analizar a los atacantes que ingresan a ella, y no tener valor de producción.

En lo restante del capítulo 1 se presenta el problema a resolver, se establece la motivación que impulsa al grupo para su realización, se enuncian los objetivos y metas buscadas así como también la secuencia de pasos que el grupo siguió para llevarlo a cabo. Se da una mirada rápida de los datos recolectados y las conclusiones obtenidas.

1.1 Problema

En base al objetivo inicial del proyecto [GSI06], se identifican tres problemas a tratar:

- Estudio del "*Estado del arte de honeypots*".
- Implantación de un *honeypot.*
- Mecanismos de monitoreo y generación de alertas.

En cuanto a la implantación, el grupo desea instalar más de un tipo de *honeypot*, se piensa entonces en desplegarlos en una red de *honeypots*. Teniendo esto en cuenta, se converge en conformar una *honeynet*. Por lo que se adicionan los siguientes problemas a resolver,

- Implantación de una *honeynet.*
- Centralización y normalización de los datos recolectados.

El principal activo de los *honeypots* son los datos recolectados durante un ataque. En la actualidad se tienen muchas implementaciones de *honeypots* disponibles, cada una con su forma particular de registrar los datos obtenidos. Además la información recolectada del ataque depende enteramente del nivel de interacción que se le permita al atacante. Dichos datos en general se almacenan en archivos de *log* (archivos de texto plano). Llevar esto a una *honeynet* implica analizar tantos archivos como *honeypots* se tengan en la red, a menos que se utilicen mecanismos para su

centralización.

Este problema se acrecienta cuando en la red se tienen *honeypots* que generan *logs* completamente heterogéneos. Analizar estos archivos consume más tiempo y los mecanismos utilizados resultan cada vez más complejos o difíciles de mantener, al igual que las herramientas utilizadas para el análisis.

1.2 Motivación

A continuación se citan los puntos que el grupo identifica como factores motivadores para la realización del proyecto:

- La tecnología de *honeypots* es una tecnología relativamente nueva.
- Es un área de la seguridad informática en pleno auge y en creciente desarrollo.
- Es factible realizar algún aporte a la comunidad, como por ejemplo descubrir un nuevo ataque o "*exploit*".
- Obtener registros de la zona en lo que a intrusiones se refiere.
- Con estas herramientas es posible aprender cómo se realizan ataques, con lo que se puede profundizar en el área de seguridad.
- No se encuentran registros públicos de *honeypots* en Uruguay, por lo tanto este proyecto es la primera implantación conocida de esta tecnología en un ambiente de producción en el país.

1.3 Objetivos

Los objetivos determinados para el proyecto abarcan los problemas descriptos anteriormente, en particular estudiar el "*estado del arte*" de la tecnología y el diseño e implantación de un *honeypot*. A continuación se describen los objetivos del grupo ante cada problema planteado.

Para realizar el "*estudio del arte de honeypots*", se investiga la evolución de la tecnología, sus características, aplicaciones, clasificaciones, ventajas y desventajas, así como se instalan y prueban las implementaciones de *honeypots* más conocidas y accesibles.

El objetivo principal del proyecto es la implantación de un *honeypot* en Facultad. El grupo desea llevar esto más allá e implantar una *honeynet* a mediano plazo. Para esta actividad es necesario realizar una propuesta de diseño e implantación que permita llevar a cabo dicho objetivo.

La solución se enfoca en la recolección y centralización de los datos obtenidos, brindando mecanismos estandarizados para el reporte de intrusiones, haciendo que el impacto de agregar un nuevo tipo de *honeypot* a la red sea mínimo. Adicionalmente se busca permitir implantar un *honeypot* de forma rápida y sencilla o de igual modo desplegar una *honeynet*, en este caso también se busca que sea fácil de extender.

La normalización de los datos recolectados es un punto primordial en este proyecto ya que se desean mantener integradas distintas implementaciones de *honeypots* y no necesariamente con las mismas características, ni con las mismas capacidades de captura de datos. Para poder implementarla de forma correcta y de modo de lograr la mayor generalidad posible a la hora de realizar la centralización de información, todos los datos deben ser llevados a una representación única, en lo posible un estándar.

Se busca brindar mecanismos de monitoreo en tiempo real de los datos obtenidos, con posibilidades de realizar reportes estadísticos y creación de gráficas que apoyen su análisis.

Complementando el sistema de monitoreo, se busca implementar un sistema de generación de alertas para notificar a los administradores ante nuevas intrusiones a los *honeypots* que conforman la solución planteada.

1.4 Organización

Se divide el desarrollo del proyecto en cinco fases: investigación, diseño, implementación, implantación en el ambiente objetivo y pruebas al producto final.

Durante la investigación en el área se adquieren los conocimientos sobre la tecnología con los cuales se realiza el trabajo de "*Estado del arte de honeypots*" [EsAr08]. Se adquieren de nociones de seguridad informática, se reconocen los tipos de ataques más comunes y formas en que se llevan a cabo. Se estudian, instalan y prueban las distintas implementaciones del mercado que se basan en la tecnología de *honeypots* de bajo nivel de interacción, como *Honeyd* [Hond05] y *Honeytrap* [Hont05], además se estudian otros como *Back Officer Friendly* [BOF08], DTK [Cohe97], *Google Hack Honeypot* [GHH05], etc.

Obtenidos los conocimientos necesarios se crea una *propuesta de diseño* [Prop08], solución al problema planteado, en la cual se aplican los conceptos adquiridos. Se propone la solución del proyecto y mejora de aquellos problemas de los que adolece la tecnología de *honeypots*, definiendo una línea de desarrollo a seguir.

La solución se implanta en Facultad y se realizan pruebas para comprobar su correcto funcionamiento. Una vez realizado esto se comienza con la captura de datos y se sigue con su posterior análisis.

1.5 Alcance del proyecto

Dado el problema y los objetivos que se establecen para este proyecto, es posible abarcar un amplio campo dentro de la tecnología de *honeypots*. Con el interés de cumplir todos los puntos establecidos y no exceder de manera considerable el tiempo de entrega del proyecto se define el alcance del mismo.

Debido a que se trata de un sistema que adquiere valor al ser atacado, se debe resolver como realizar la captura de datos sin comprometer su integridad. Se desea centralizar los datos capturados aprovechando al máximo las características que brinda una *honeynet*, debiendo resolver para ello diferentes problemáticas, como por ejemplo el proceso de normalización de datos, la comunicación de los *honeypots* para la transferencia de la información obtenida, etc.

Implantar la arquitectura de la solución planteada [ArqI08], haciendo uso para ello del equipo dedicado para el proyecto que fue cedido por el GSI (*Grupo de Seguridad Informática*) [GSI06].

Se proporciona un sistema de monitoreo y análisis de datos, así como también un sistema de alarmas que notifica a los administradores ante nuevos ataques perpetrados contra los elementos de la red (*honeypots*, repositorio, sistema base, etc.).

Esta solución normaliza los datos capturados por los diferentes tipos de *honeypots* según el formato *IDMEF* (*Intrusion Detection Message Exchange Format*) [NWGp07]. Se permite agregar nuevos *honeypots* de forma sencilla, transparente y sin esfuerzo adicional. Con un sistema de alarmas que notifica vía SMTP ante nuevas intrusiones, las que pueden ser analizadas con el sistema monitoreo y análisis estadístico.

La solución se implanta definitivamente en Facultad, capturando ataques provenientes de varias partes del mundo, permitiendo elaborar un completo análisis sobre los mismos, estableciendo a partir de ellos diversas estadísticas, para luego comprobar si se alinean con estadísticas de otras *honeynets* atacadas como ser estadísticas del *Proyecto Honeynet* de la UNAM [UNAM06] y de la *Brazilian Honeypots Alliance*.

1.6 Resultados obtenidos

Durante el período de puesta en producción de la solución (5 de diciembre de 2007 al 13 de febrero de 2008) se realiza la captura de ataques hacia los distintos *honeypots*. Estos ataques consistieron en intentos de conexión a puertos determinados, siendo los tres más atacados el TCP 1433 asociado al servicio del DBMS *MSSQL Server*, el TCP 139 asociado el servicio NetBIOS y el TCP 25 correspondiente al servicio SMTP. También se registraron ataques de fuerza bruta para intentar tomar el control del sistema vía SSH.

Fue posible identificar ataques dirigidos al TCP 1433 y 139. Para el primero se tiene indicios de que se trata de ataques *MSSQL Hello Buffer Overflow* [Benj03]. En el segundo caso se investigo el patrón del ataque, correspondiendo este a la secuencia de escaneo previa a la infección del gusano *W32. Sasser.Worm* [MiTN04] [LkLg06]. Para este último se observa el papel de los *honeypot* retrasando el ataque automático, ya que se insistió una y otra vez con el escaneo hacia el puerto.

Por último como dato estadístico se tiene que el país desde donde se registra la mayor cantidad de ataques es India, seguido por Portugal, Estados Unidos y China.

1.7 Estructura del documento

Este documento está compuesto de diez capítulos que abarcan todas las etapas cubiertas durante la realización del trabajo. Además lo acompaña un DVD el cual contiene documentos anexos donde profundiza en detalle en algunos temas investigados y herramientas utilizadas.

En el capítulo 2 se presenta el "*estado del arte de honeypots*", donde se hace un racconto del documento del mismo nombre [EsAr08]. Se definen los conceptos necesarios para comprender el resto del trabajo. Se describen los *honeypots* más conocidos del mercado que fueron estudiados para conocer las características y funcionalidades de herramientas basadas en esta tecnología.

En el capítulo 3 se establece la solución planteada [Prop08] al problema y se explican las decisiones de diseño tomadas al momento de establecer la arquitectura de la solución. Se abordan los temas de captura, normalización y centralización de datos. Se presenta y discute el modelo de datos *IDMEF* [NWGp07], utilizado para llevar a cabo la normalización. Se discuten las ventajas y desventajas de realizar el monitoreo en forma centralizada, así como también el disparo de alarmas desde un único punto. Por último se describen los componentes participantes de la solución.

En el capítulo 4 se presentan las herramientas utilizadas para la implementación de la solución obtenida en la etapa de diseño, justificando las decisiones tomadas en cada paso.

El capítulo 5 se describe la implantación, se trata cada componente de la arquitectura, detallando su funcionalidad y configuración. Dichos componentes son el *Honeywall*, encargado del control y captura de datos, los *honeypots* (*Honeyd* y *Honeytrap*) integrados en un único dispositivo denominado *HoneyCell*, el repositorio central de datos, desde donde se disparan las alarmas y se realiza el análisis y monitoreo estadístico, conocido como *HoneyCenter* y la terminal de administración.

En el capítulo 6 se realiza el análisis de los resultados obtenidos luego de mantener la solución operativa durante más de dos meses. Para dicho análisis se utilizan los datos obtenidos en el período comprendido entre el 5 de diciembre de 2007 y el 13 de febrero de 2008.

El capítulo 7 trata temas referentes a la gestión y puesta en marcha del proyecto.

En el capítulo 8 se enumeran las conclusiones obtenidas en base a los estudios e investigaciones realizadas, teniendo presente los resultados obtenidos durante el período de captura de datos.

Por último, el capítulo 9 presenta los trabajos a futuro que el grupo identificó como relevantes durante el transcurso de este proyecto de grado.

CAPÍTULO 2 ESTADO DEL ARTE DE HONEYPOTS

En este capítulo se resume el estudio realizado sobre el "*estado del arte*" referente a la tecnología de *honeypots* y se presentan los conceptos básicos necesarios para comprender el resto del trabajo. El estudio completo realizado por el grupo sobre el tema se encuentra disponible en el documento "*Estado del Arte de honeypots*" [EsAr08].

2.1 Introducción

La masiva expansión de las computadoras en todos los ámbitos de la sociedad, así como la acelerada evolución de las tecnologías de las comunicaciones, han sido los pilares del crecimiento de las redes de computadoras. El ejemplo más famoso es la mayor red de computadoras existente: Internet.

En la actualidad millones de personas de todo el mundo, día a día utilizan Internet como parte de su trabajo y ocio. El carácter universal que ha tomado la red de redes permite la conectividad global y permanente entre actores de todo el planeta, en forma económica y rápida, lo que ha impulsado la participación de las empresas de forma de aumentar el alcance de estas en la sociedad.

Sin embargo, a medida que se interconectan más equipos han avanzado los distintos incidentes de seguridad, que ponen en peligro la integridad de los sistemas y seguridad de la información. La Imagen 2.1 exhibe un gráfico donde se ve que al pasar los años los ataques son cada vez más sofisticados y dado la disponibilidad de diferentes herramientas, los atacantes no necesariamente deben tener conocimientos muy amplios para poder llevarlos a cabo [LoBro08].

Desde los primeros incidentes reportados, las tecnologías de seguridad intentan bloquear los ataques (firewalls) o detectarlos antes de que penetren e infecten la red (IDS's). Ambas tecnologías son críticas, pero tienen limitaciones. Lo que suele ocurrir es que las herramientas utilizadas generan mucha información, tienen altas tasas de falsos negativos, y por ello no proporcionan datos a tiempo para asegurar los sistemas vulnerables y mitigar así los efectos de las intrusiones.

¿Por qué no aplicar una técnica milenaria, utilizada en las guerras?

"*El arte de la guerra se basa en el engaño*", Sun Tzu. [SunT07].

La utilización adecuada del engaño puede proporcionar tiempo adicional, obteniendo información para preparar una buena defensa y prevenir el ataque. La idea es poner en funcionamiento una de las estrategias más utilizadas durante toda la historia, para la defensa de las redes y los sistemas. Es decir, aprender de los ataques hacia los distintos sistemas de una organización, identificando distintas vulnerabilidades y posibles problemas de seguridad pasados por alto, aprendiendo las técnicas usadas por los atacantes. Este objetivo es el que se encuentra detrás de la tecnología de *honeypots*.

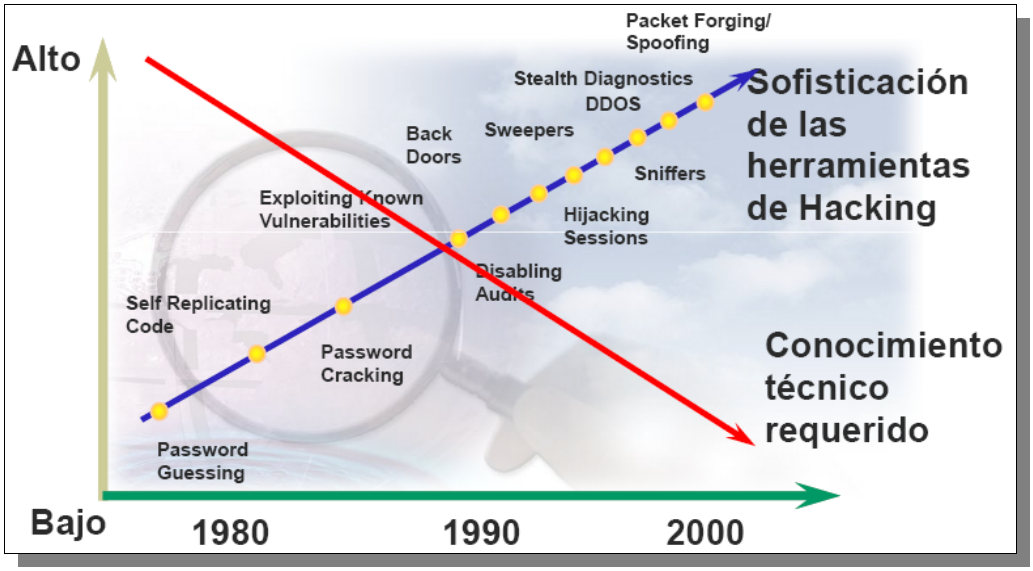


Imagen 2.1: Evolucion de herramientas y disminución del conocimiento requerido para realizar ataques.

2.2 Aprendiendo del enemigo

El concepto de *honeypots* surge hace más de 10 años y existen varias interpretaciones del término. Lance Spitzner [Spit01] lo define de la siguiente manera:

"Un honeypot es un recurso cuyo valor está en ser atacado o comprometido. Se espera de un honeypot que sea probado, atacado y potencialmente explotado. Los honeypots no arreglan nada. Proporcionan información adicional y valiosa." [Spit01]

Reto Baumann [BaPl02] en su paper propone una definición un poco diferente, centrando su definición en la distracción del atacante:

"Un honeypot es un recurso que pretende ser un objetivo real. Se espera de un honeypot que sea atacado o comprometido. Su meta principal es la distracción de los atacantes y obtener información sobre los ataques y atacantes." [BaPl02]

Como mencionan los autores, un *honeypot* en ningún momento persigue la finalidad de ser una solución de seguridad, sino que es un complemento a los sistemas de seguridad existentes. Se encarga de proporcionar información sobre los atacantes que intentan comprometerlo, antes de que comprometan otros sistemas de la red en la que se encuentra. Es un recurso que busca ser comprometido, el cual no contiene valor de producción. Por consiguiente, nadie (equipo, usuario ni aplicación) debería interactuar con ellos y cualquier comunicación establecida se considera como no autorizada. Por lo tanto, no se tienen casos de falso positivo y los datos recolectados son altamente valiosos al ser siempre producto de intrusiones realizadas. Con estos datos se puede averiguar los medios que utiliza el atacante y, quizás sus motivos, sin el costo de analizar un sistema de producción que haya sido comprometido.

Los *honeypots* constituyen una tecnología altamente flexible y adaptable a cualquier ambiente. El recurso para ser atacado puede ser de diversa índole como computadoras con diferentes sistemas operativos (Windows, Linux, Unix, etc.), un equipo de red (router, firewall, etc.), un

servicio (correo electrónico, página Web, base de datos), etc.

2.2.1 Objetivos

Los *honeypots* no resuelven problemas de seguridad, ni detienen a los atacantes. Sus objetivos principales son la captura de ataques a vulnerabilidades conocidas y desconocidas (identificación de nuevas vulnerabilidades) y descubrimiento de riesgo en los sistemas.

Tradicionalmente, la detección de ataques ha sido una tarea extremadamente difícil de llevar a cabo. Las tecnologías clásicas como los sistemas de detección de intrusos (IDS) han sido deficientes en el punto de que generan información en cantidades excesivas, con una alta tasa de falsos positivos (se reporta como ataque algo que en realidad constituye tráfico normal y autorizado) y no cuentan con la habilidad de detectar nuevos ataques (0day attacks[2]).

Es importante destacar que los *honeypots* pueden capturar ataques desconocidos. Diariamente aparecen nuevas versiones y formas de malware, con la captura de estos es posible obtener una firma y así actualizar los sistemas de seguridad.

Por definición los *honeypots* no contienen sistemas con valor de producción. Con lo cual todo tráfico hacia ellos es considerado producto de un ataque o intrusión. Es por esta razón que no generan falsos positivos y solo capturan pequeñas cantidades de datos en comparación con los sistemas de detección de intrusos y firewalls.

Por otro lado, al detectar una intrusión en sistemas de producción comprometidos, deben ser desconectados para ser analizados. A su vez la cantidad de información que se genera es considerablemente extensa, de manera que es muy difícil determinar las acciones realizadas por el atacante dentro del sistema. Los *honeypots* son excelentes para ayudar en el análisis de incidencias. Se pueden sacar de la red de forma fácil y rápida para realizar un análisis forense completo sin causar impacto en las operaciones diarias de la organización.

Otra característica de importancia de los *honeypots* es la del engaño o la disuasión. La idea de esta contramedida es confundir al atacante y hacerle perder tiempo y recursos mientras interactúa con el *honeypot*, lo que en definitiva se transformará en el desistimiento por parte del atacante. Mientras ese proceso se lleva a cabo, se puede detectar la actividad del atacante y se tiene tiempo para reaccionar y detener el ataque. En esta área se ha discutido el valor de los *honeypots*, ya que hoy en día muchos de los ataques son producto de herramientas automatizadas que atacan a los *honeypots* y a los sistemas en producción de forma muy rápida.

2.2.2 Desafíos

Como en cualquier otra tecnología, los *honeypots* también tienen sus debilidades. Una de ellas es su limitado campo de visión, pues sólo detectan los ataques que interactúan con ellos y no aquellos que son dirigidos a otros recursos. Esta propiedad es conocida como "*visión limitada*".

El uso de *honeypots* puede introducir riesgo adicional al ambiente donde éste se instale, ya que al ser comprometido puede ser usado para causar daños dentro de la red, puede utilizarse para lanzar ataques a otros recursos, tanto internos como externos, que contengan información o valor de producción.

2 El el contexto de la seguridad informática, se denomina "*0day attack*" a los ataques que intentan explotar alguna vulnerabilidad hasta el momento desconocida.

No obstante, una buena planificación, implantación y control de un sistema *honeypot* permite minimizar el riesgo y obtener información valiosa y detallada acerca de las herramientas, tácticas y motivos que utiliza un atacante para violar la seguridad de los sistemas.

2.2.3 Clasificación

Se pueden clasificar de 3 formas, por funcionalidad (dependientes del objetivo perseguido), según el nivel de interacción que se permita o según la plataforma de instalación, real o virtual. Es claro que estas formas de clasificación no son disjuntas entre las categorías ya que por ejemplo se puede tener un *honeypot* de bajo nivel de interacción y virtualizado (como es el caso del proyecto).

2.2.3.1 Funcionalidad

Clasificar un *honeypot* por funcionalidad se refiere a cuál será el fin de la operativa del mismo. Esta clasificación se divide en dos categorías:

- **Honeypot de producción**: Son utilizados para mitigar el riesgo en una organización distrayendo o haciendo más lento un ataque. Al detectar una intrusión se toman las medidas de contingencia oportunas (denegación de acceso a un origen determinado, limitación de capacidades de servicios, etc.).

- **Honeypot de investigación:** El principal objetivo es el de recoger información sobre los ataques librados sobre el mismo, de forma de obtener comportamientos y técnicas utilizadas por los atacantes. Se utilizan generalmente en organizaciones o universidades, interesadas en aprender sobre las amenazas, como por ejemplo para fabricar un antivirus.

2.2.3.2 Nivel de interacción

Por interacción se entiende el grado de interacción que el atacante tiene con el sistema. Se definen tres niveles bajo, medio y alto. A medida que se escala en el nivel de interacción el riesgo de la red en la que se encuentra implantado también aumenta.

- **Bajo nivel de interacción (*Low involvement*):** Este tipo de *honeypot,* simplemente simula la existencia de servicios como `HTTP`, `FTP`, `TELNET`, etc.; escuchando peticiones y almacenándolas en *logs*, pero sin responderlas. Esto determina un sistema totalmente pasivo, que solo registra peticiones.

- **Nivel medio de interacción (*Medium involvement*):** Este tipo de *honeypot*, brinda servicios más sofisticados, de modo de captar una mayor atención. Para ello aumenta el grado de interacción con el atacante, respondiendo a las peticiones según un script que simula el comportamiento del puerto objetivo. Esto permite un análisis más minucioso de la actividad del atacante.

- **Alto nivel de interacción (*High involvement*):** Este tipo de *honeypot* no simula ningún servicio, sino que presta servicios reales. Resultan muy atractivos para los atacantes, y también son los que más datos proporcionan sobre ellos y pueden poner en evidencia las técnicas utilizadas. A la vez son los más riesgosos, ya que un atacante podría penetrar el sistema apoderándose de él, y podría utilizarlo para atacar otros recursos con valor real.

La Tabla 2.1 establece una comparación de las características de los *honeypots* según la clasificación por nivel de interacción.

Nivel de interacción	Bajo	Medio	Alto
Servicio real	No	No	Si
Nivel de riesgo	Bajo	Medio	Alto
Información obtenida	Conexiones	Peticiones	Todo
Riesgo - Compromiso	No	No	Si
Conocimiento del funcionamiento	Bajo	Bajo	Alto
Conocimiento del desarrollo	Bajo	Alto	Mediano a alto
Tipo de mantenimiento	Bajo	Bajo	Muy Alto

Tabla 2.1: Comparación entre niveles de interacción de honeypots

2.2.3.3 Nivel de implantación

Una tercera clasificación posible es de acuerdo al nivel de implantación:

- **Física**: es cuando se utiliza un equipo real para soportar la implantación de un *honeypot.*
- **Virtual**: es cuando se utiliza software de virtualización para implantar el *honeypot.*

2.2.4 Captura de datos

Los datos proporcionados por cualquier sistema de seguridad son los que dan una idea real de la salud de los sistemas, contienen información sobre quién, qué, cuándo y por qué ocurre un evento.

El principal cometido de los *honeypots* es la captura de datos. Dichos datos son el resultado de registrar los ataques recibidos. La información capturada depende del nivel de interacción con el que fue diseñado.

2.2.4.1 Honeypots de bajo nivel de interacción

En un *honeypot* de bajo nivel de interacción, todos los servicios brindados son emulados, y por definición se debe limitar estos a no interactuar con el atacante. Por este motivo la captura de datos apunta a registrar los intentos de conexión a los diferentes servicios, obteniendo los datos que se envían a un puerto atacado.

Los datos capturados son procesados por el *honeypot* y volcados a un archivo de *log*. Cada implementación de *honeypots* define su propia estructura de archivos *logs*, pero existen datos que son básicos y deben estar en estos archivos como lo son la `IP` del atacante, puerto atacado, servicio, etc.

El Fragmento 2.1 es una parte de un archivo de *log* de *Honeytrap* [Hont05], donde se muestra la información obtenida ante ataques al *honeypot*.

```
honeytrap v1.0.0 Copyright (C) 2005-2007 Tillmann Werner <tillmann.werner@gmx.de>
[2008-01-24 13:27:55] ---- Trapping attacks on eth0 via PCAP. ----
[2008-01-24 13:28:48]  * 139/tcp          209 bytes attack string from 194.109.225.85:2615.
[2008-01-24 13:28:49]    139/tcp          Connection from 194.109.225.85:2868 accepted.
[2008-01-24 13:29:01]  * 139/tcp          209 bytes attack string from 194.109.225.85:2868.
[2008-01-24 13:29:02]    139/tcp          Connection from 194.109.225.85:3291 accepted.
[2008-01-24 13:29:14]  * 139/tcp          209 bytes attack string from 194.109.225.85:3291.
[2008-01-24 13:29:15]    139/tcp          Connection from 194.109.225.85:3590 accepted.
[2008-01-24 13:29:28]  * 139/tcp          209 bytes attack string from 194.109.225.85:3590.
[2008-01-24 13:29:29]    139/tcp          Connection from 194.109.225.85:3862 accepted.
[2008-01-24 13:29:41]  * 139/tcp          209 bytes attack string from 194.109.225.85:3862.
[2008-01-24 13:29:42]    139/tcp          Connection from 194.109.225.85:4625 accepted.
[2008-01-24 13:29:55]  * 139/tcp          209 bytes attack string from 194.109.225.85:4625.
[2008-01-25 12:36:38]    4899/tcp         Connection from 75.108.157.107:2038 accepted.
[2008-01-25 12:36:39]  * 4899/tcp         10 bytes attack string from 75.108.157.107:2038.
[2008-01-26 15:20:51]    1433/tcp         Connection from 204.15.73.118:19136 accepted.
[2008-01-26 15:20:53]  * 1433/tcp         158 bytes attack string from 204.15.73.118:19136.
```

Fragmento 2.1: Sección del archivo de log de Honeytrap

2.2.4.2 Honeypots de alto nivel de interacción

Un *honeypot* de alto nivel de interacción ofrece servicios reales con los cuales el atacante puede interactuar, permitiendo así recopilar mayor cantidad de información relativa al ataque.

¿Cómo capturar las acciones del intruso sin que lo note? El enfoque tradicional se basa en capturar las actividades registrando sus datos en la red, sin embargo no es posible cuando el atacante utiliza un canal cifrado. Una solución a este problema es realizar la captura a nivel de kernel, donde los datos no están cifrados. Por ejemplo con la herramienta *Sebek* [HPSe05] se evita el cifrado al capturar la actividad en el espacio del kernel. Esta herramienta puede capturar pulsaciones de teclas, recuperar contraseñas, y monitorizar cualquier comunicación.

Es importante garantizar la integridad de los archivos de *log* y más aún en este tipo de *honeypots,* ya que se tiene un alto riesgo de ser comprometido por un atacante. Si esto sucede, el atacante tiene control total del sistema y podría alterar los archivos. Una de las mejores medidas es realizar el registro en un sistema remoto, si bien es una solución sencilla es necesario considerar aspectos tales como el mecanismo de volcado utilizado, seguridad del canal de comunicación y del equipo que almacena los *logs*.

2.3 Honeynets

Es una red de *honeypots* que puede incluir otros componentes para la captura de datos y está diseñada principalmente para la investigación. Se debe destacar que es una arquitectura no un producto concreto. El objetivo principal es obtener la mayor cantidad de información de los ataques a una red real. Se permite mayor interacción ya que se interactúa con varios elementos de una red (firewall, router, switch, etc.).

En la Imagen 2.2 se muestra un esquema de una *honeynet* básica, compuesta por *honeypots*, un mecanismo de control y captura de datos (en este caso un firewall) y un repositorio que almacena los datos obtenidos de los ataques recibidos por los distintos componentes.

Un factor clave es que la red ha sido diseñada para ser comprometida. Por lo que proporciona un gran atractivo para el atacante. Esto a su vez obliga a tomar medidas para asegurar que ésta no

sea utilizada para desplegar ataques hacia otras redes.

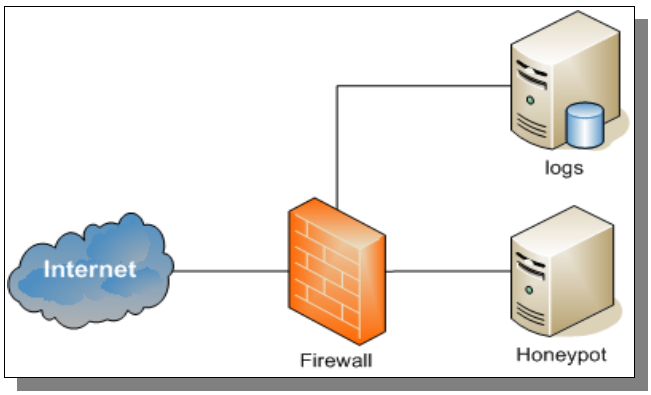


Imagen 2.2: Esquema de una honeynet básica.

Como todo el tráfico que entra a una *honeynet* se considera un ataque, cualquier tráfico saliente es indicador de que la *honeynet* fue comprometida y puede tratarse de un ataque a sistemas o computadores externos.

Una *honeynet* presenta 2 requisitos fundamentales,

- **Control del flujo de datos**: Al ingresar un atacante a la *honeynet,* se controla el flujo de datos, de manera que los elementos de la red (*honeypots*) no sean usados para generar ataques a otros sistemas. Aquí se debe llegar a un equilibrio entre nivel de interacción que se va a permitir a los atacantes y los riesgos a que se van a exponer los sistemas. Para ello se pueden aplicar distintas estrategias, como por ejemplo, limitar el tráfico y conexiones salientes, limitar el ancho de banda, limitar rangos de direcciones IP entrantes. Estas estrategias disminuyen el riesgo sin llegar a eliminarlo por completo.

- **Captura de datos:** El gran reto es lograr la captura de tantos datos como sea posible, sin que el atacante lo perciba. Esto se logra realizando modificaciones sobre los elementos de la red. Los datos capturados no deben guardarse localmente en el *honeypot*, deben ser almacenados de forma remota, ya que pueden ser detectados por el atacante, y como consecuencia de esto pueden resultar alertados. Además, la captura de datos se debe realizar por capas, es decir realizar la captura de información haciendo uso de distintos recursos, cada uno aportando distinta información de una misma conexión, al combinar estas capas se obtiene el panorama completo de lo que ha realizado el atacante.

2.3.1 Arquitectura de una Honeynet

Con el pasar del tiempo las *honeynets* han ido mejorando progresivamente, dando lugar a lo que los autores llaman generaciones de *honeynets*, cada una incorporando nuevas características y funcionalidades.

2.3.1.1 Generación I

En sus principios las *honeynet* sólo implementaban mecanismos básicos de control y captura de datos. El objetivo de este tipo de redes es capturar la máxima cantidad de información de un intruso, simulando una red real. En la Imagen 2.3 se presenta la arquitectura utilizada para el

despliegue de una *honeynet* de generación I por parte del "*Honeynet Project*" [HoPr02] en el período comprendido entre 1999 y 2002.

Este grupo ha escrito una serie de documentos llamados "*Know your enemy*" [KyEn00] donde se abarca desde la construcción de una *honeynet* hasta el análisis de los datos recolectados, además recopilan experiencias, información de interés y ejemplos.

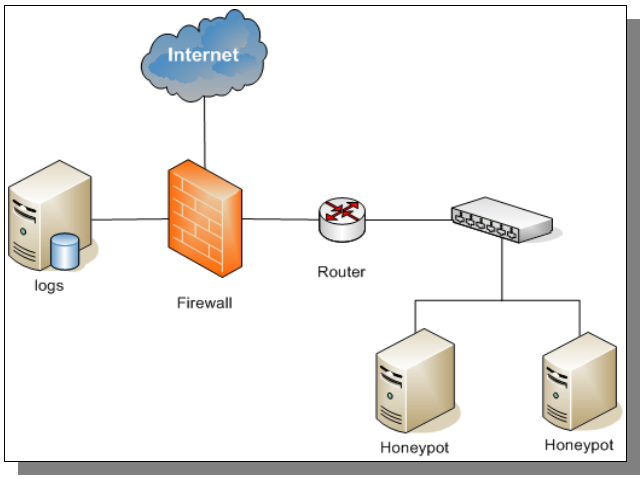


Imagen 2.3: Arquitectura Honeynet Gen I

La arquitectura dispone un único punto de acceso que se encarga de controlar el tráfico de entrada y salida. Los datos, al ser capturados, se almacenan en servidores de *logs* remotos, evitando exponerlos en caso de que la *honeynet* sea comprometida.

Como se observa en la Imagen 2.3, el firewall se encuentra delante de un router que gestiona la *honeynet*. De esta forma se logra ocultar el firewall o al menos disimularlo obteniendo un agente de control a la misma vez. En caso de que la *honeynet* sea comprometida, existen 2 puntos de control, el firewall y el router. Los *logs*, se depositan en un sitio seguro, por fuera de la *honeynet*.

2.3.1.2 Generación II

Esta generación combina todas las características de control y captura de datos de la generación anterior en un dispositivo único, denominado *Honeywall* [HPHW07], facilitando la instalación de una *honeynet* y haciendo difícil su detección.

El *Honeywall*, creado por el *Honeynet Project* en el 2002, es un dispositivo a modo de *gateway* que trabaja a nivel de capa 2 (en modo bridge) por lo que resulta invisible para un atacante. En el dispositivo se recopilan varias herramientas necesarias para administrar una *honeynet* de segunda generación, como por ejemplo las que controlan el tráfico de entrada y salida a la red. El grupo realizó un análisis más detallado de este conjunto de herramientas, el cual, puede consultar en el documento "*Honeywall*" [Howl08].

La Imagen 2.4 muestra la arquitectura de una *honeynet* de generación II, en esta arquitectura el router y el firewall se fusionan para centralizar el control y captura de datos en un componente denominado *Honeywall*.

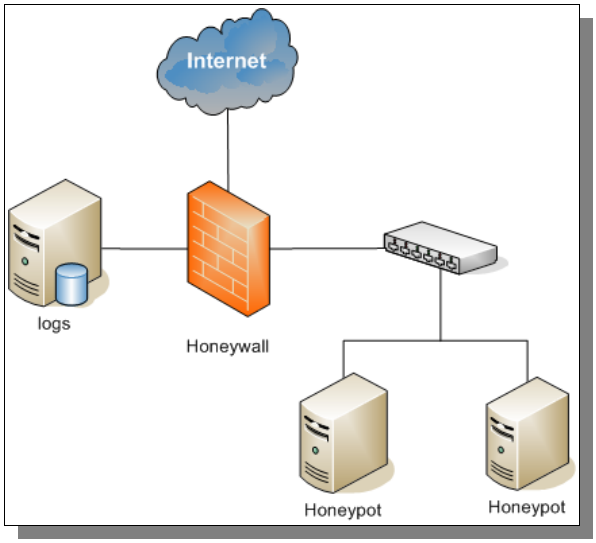


Imagen 2.4: Arquitectura Honeynet Gen II

2.3.1.3 Generación III

Las *honeynets* de tercera generación permiten la recolección de información a más bajo nivel que los anteriores. Esta generación hace fundamental hincapié en los datos recolectados de las acciones de un atacante directamente sobre el *honeypot*. Para ello se idearon implementaciones, como se verá a continuación, a nivel del kernel (en el equipo atacado), de forma de capturar todas las interacciones que mantiene un atacante.

Sebek [HPSe05] es una herramienta diseñada para la captura de datos, intenta capturar la mayor cantidad posible de datos de la actividad en el *honeypot* sin que el atacante lo note. Envía los datos obtenidos a un servidor-sebek. La idea tras esta herramienta es lograr la captura de pulsaciones de teclado, archivos transmitidos, contraseñas ingresadas, chats abiertos en el sistema, y todas las comunicaciones encriptadas (`SSH`, `IPSec`, `SSL`). Es utilizada principalmente en *honeypots* de alto nivel de interacción. La Imagen 2.5 es un diagrama que muestra como realiza la herramienta la captura de datos cuando un atacante realiza una conexión vía `SSH` para tomar el control del *honeypot*.

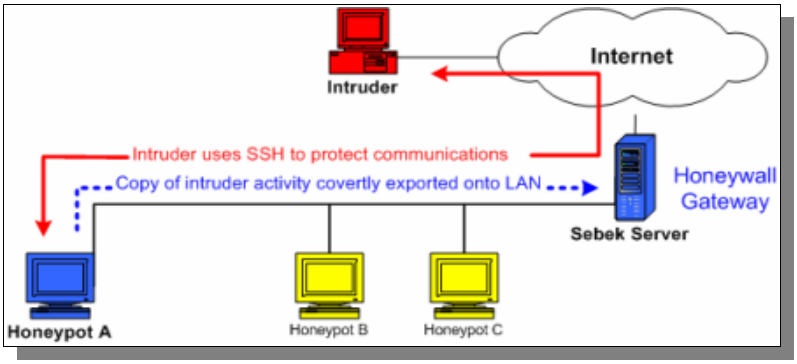


Imagen 2.5: Comunicación entre Sebek Server y Sebek Client

2.3.1.4 Generación IV

La cuarta generación de *honeynets* refiere a *honeynets* distribuidas [HPDi07]. La idea tras esto es integrar distintas *honeynets* y compartir la información recolectada.

Como es de esperar, esto no es algo trivial ya que se necesitan variados elementos de seguridad y compatibilidad para poder lograr un vuelco exitoso de datos. Se debe tener en cuenta que no siempre puede ser posible compartir toda la información entre los miembros (como por ejemplo IP's atacadas, direcciones de MAC, información personal, etc.), de modo que se hace necesario realizar un filtro a la información antes de compartirla.

Un esquema de una *honeynet* distribuida se presenta en la Imagen 2.6.

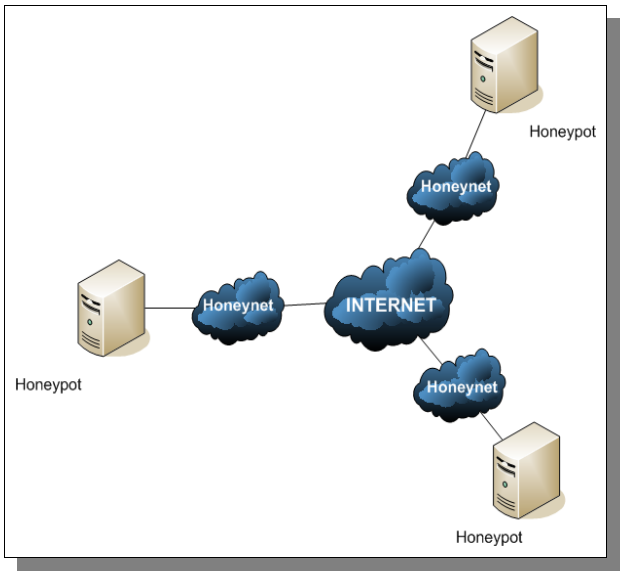


Imagen 2.6: Esquema de una honeynet distribuida.

Como consecuencia inmediata de la centralización de información, es posible estudiar los datos realizando un análisis completo con gran cantidad de información aunque, esto pueda ser difícil de manejar.

2.4 Implementaciones de honeypots

A continuación se describen las implementaciones más populares de *honeypots* vistos, junto con un resumen de sus principales características.

2.4.1 Deception Toolkit (DTK)

Es un *honeypot* de bajo nivel de interacción, open source, el cual se basa en simular servicios. Básicamente se trata de un conjunto de scripts *Perl* [Perl02] diseñados para emular una variedad de vulnerabilidades conocidas, las cuales pueden ser configuradas para dar respuestas falsas (archivos, peticiones, etc.).

Un objetivo peculiar de *DTK* [Cohe97] es que sus desarrolladores intentan hacer que este sea fácil de detectar. Esto lo hacen porque la idea del sistema se basa en el supuesto de que si un

atacante identifica la instalación en la red que desea comprometer, desistirá de su intento y buscará otro blanco (decepción). Está orientado a desahuciar al atacante, intentando vencerlo psicológicamente. Este objetivo no es posible cumplirlo si se trata de un ataque automatizado.

2.4.2 Back Officer Friendly (BOF)

Es un *honeypot* de bajo nivel de interacción, desarrollado en 1998 por Marcus Ranum que corre bajo plataforma Windows. Su desarrollo fue pensado para responder al ataque específico del troyano BackOrifice [CDCo08].

Es una herramienta que fue diseñada pensando en el usuario final, por eso su característica principal es la facilidad de uso. Si bien su distribución es libre actualmente es difícil de encontrar en el mercado, ya que se encuentra en desuso. Es posible exportar el *log* a un archivo `txt` de forma manual.

En la Imagen 2.7 se muestra la ejecución del programa, y se ven los posibles puertos a emular. Por más información sobre esta herramienta puede consultar [BOF08], un documento realizado por el grupo donde se estudian las principales características del *honeypot*.

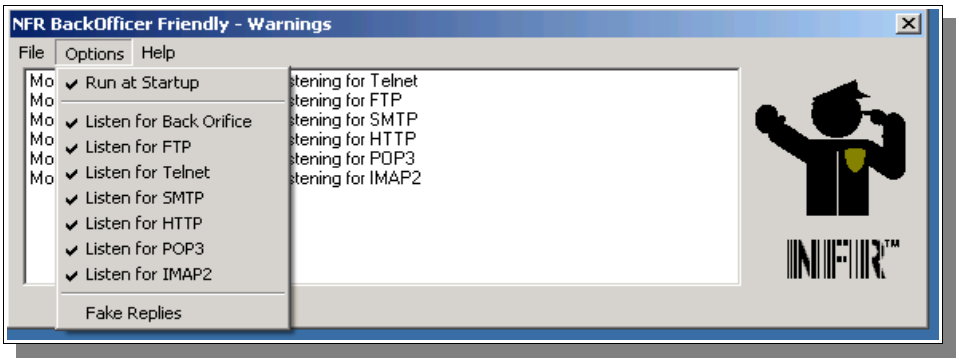


Imagen 2.7: Back Officer Friendly

2.4.3 Specter

Es un *honeypot* comercial de bajo nivel de interacción creado por NetSec [Spec97], una compañía Suiza de seguridad de redes.

Puede emular diversos servicios con sus respectivas vulnerabilidades, y también puede escuchar en determinados puertos para detectar cualquier actividad.

Ofrece la emulación de varios sistemas operativos, limitado por sistema anfitrión a los sistemas Windows. El stack `IP` no se emula, con lo cual el *honeypot* corre el riesgo de ser detectado cuando se emula un sistema diferente a Windows.

Si desea profundizar sobre este *honeypot* se puede referir al estudio realizado por el grupo [Spec08].

2.4.4 Symantec Decoy Server

Permite virtualizar, hasta cuatro sistemas operativos sobre un único sistema operativo base. En este producto la virtualización es realizada por la herramienta en sí y no por software de virtualización de terceros.

Cada sistema instalado es completamente funcional, tiene su propio sistemas de archivos, binarios, bibliotecas, procesos y tarjeta de red. En la Imagen 2.8 se bosqueja la virtualización de sistemas con este *honeypot*.

Imagen 2.8: Esquema de virtualización con SDS

2.4.5 LaBrea Tarpit

LaBrea Tarpit [LbrT03] es un *honeypot* open source creado por Tom Liston, diseñado para enlentecer o detener los ataques, a este tipo de *honeypots* se los denomina "*stinky honeypot*" (o *honeypot pegajoso)*. Para ello introduce el concepto de *Tarpit,* un servicio que intenta frenar a los emisores de *correo spam*[3].y *worms*[4]. enlenteciendo las conexiones TCP, llegando incluso a paralizarlas pero sin cortar la conexión.

Su funcionamiento se basa en tomar direcciones IP no utilizadas en la red en que se encuentra, instalando y configurando en ellas servidores virtuales que brindan servicios comunes, como por ejemplo, HTTP, POP3, etc. Cuando un atacante realiza una conexión intenta mantener esta abierta (*IDLE)* el mayor tiempo posible. Para lograr ese objetivo cuenta con los siguientes mecanismos:

- **Regulación (throtlling):** este mecanismo consiste en aceptar nuevas conexiones, pero utilizando una ventana pequeña para la recepción de datos. De esa forma indica a la entidad emisora no enviar más datos por paquete de lo soportado, obteniendo con ello lentitud en la conexión.

- **Captura persistente (persistent capture):** establece que la ventana de recepción TCP tiene tamaño 0, por lo que indica a los emisores a realizar esperas antes del envío de más información. Para que la conexión no sea cancelada, periódicamente vuelve a dar un tamaño mayor para que el emisor vuelva a trasmitir obteniendo así una alta degradación de la velocidad de la conexión.

3 Se denomina *correo spam* a correos recibidos no solicitados, en general con objetivo publicitario.
4 El término *worms* (*gusano)* refiere a programas con la capacidad de replicarse en forma automática en otros PC's conectados mediante una red.

Con esto logra mantener ocupado al atacante retrasando su búsqueda de nuevos objetivos. *LaBrea* es utilizado principalmente para el bloqueo de ataques de automatizados.

2.4.6 Honeytrap

Honeytrap [Hont05] es un *honeypot* de bajo nivel de interacción que se especializa en la captura de malware. Creado por Tillmann Werner, la característica más sobresaliente de esta herramienta es la posibilidad de capturar ataques conocidos como no conocidos.

Realiza una estricta distinción entre la captura de datos y su análisis. Los servicios destinados a la captura de datos se encuentran en la distribución principal de la herramienta, posibilitando al instalarla contar inmediatamente con la posibilidad de realizar la captura de cualquier ataque.

Además se tiene la capacidad de incluir *plugins* en tiempo de ejecución para realizar análisis de las capturas realizadas. Esto permite estudiar nuevos ataques simplemente creando *plugins* para la nueva vulnerabilidad sin necesidad de recompilar la herramienta y responder así en forma rápida a amenazas desconocidas.

El funcionamiento de *Honeytrap* es muy sencillo, al detectar una petición de conexión a un puerto, inicia inmediatamente un servicio el cual abre dicho puerto y establece un manejador para los datos recibidos. De esta manera no es necesario contar con miles de puertos abiertos para asegurar la captura de un nuevo ataque. Para ello utiliza lo que denomina monitores de conexión, éstos son de dos tipos:

- **Sniffers de red:** son programas basados en la librería de captura de paquetes de red `libpcap` [TCPD07], los cuales buscan paquetes `RST` (reset)[5]. con secuencia 0 que es generada por el host local. Estos paquetes indican un rechazo de conexión a un puerto porque se encuentra cerrado. Al encontrar uno de estos paquetes, se abre el puerto y se le asigna un manejador por parte de *Honeytrap* para las conexiones entrantes.

- **Sistema Linux:** utiliza los sistemas `netfilter` o `iptables` del kernel de Linux para la intercepción de respuestas de conexiones. Se debe crear una regla en `iptables` para derivar por ejemplo los paquetes `SYN`[6]. a *Honeytrap.*

La información recopilada por *Honeytrap,* puede ser muy básica, o muy específica (si se utilizan plugins). Todas las conexiones capturadas son registradas en un archivo de *logs*, en el que se indica:

- `IP` de origen del ataque
- Puerto de origen
- Puerto atacado
- Protocolos utilizados

5 Especifica el reinicio de la conexión entre las entidades receptora y la emisora.
6 Se utiliza para la sincronización de números de secuencia entre las entidades.

2.4.7 Honeyd

Honeyd [Hond05] es quizás el software de tecnología *honeypots* con más adeptos en Internet. Es mucho más que un *honeypot* ya que permite virtualizar una *honeynet* relativamente compleja con un grado de realismo muy elevado sin necesidad de utilizar costosos recursos de hardware. En otras palabras ha llevado el grado de emulación hasta simular redes enteras de *honeypots*. Permite ampliar su funcionalidad mediante la incorporación de nuevos módulos de emulación, lo que lo convierte en una herramienta extremadamente flexible.

Honeyd está disponible para las principales plataformas UNIX y actualmente para Windows [WiHo04], además de:

- Linux
- BSD
- Solaris

Como *honeypot*, permite emular servicios, la principal particularidad es que emula los sistemas operativos a nivel de stack TCP/IP, simulando servidores, estaciones de trabajo o elementos de interconexión como routers. Herramientas como `nmap` [Nmap08] ejecutan de forma rápida y sencilla ataques que permiten identificar el sistema operativo de la víctima, utilizando patrones de comportamiento de los protocolos de nivel de red y de transporte. *Honeyd* utiliza estrategias de emulación imitando precisamente estos patrones con lo cual los equipos virtualizados presentan exactamente el mismo comportamiento que un equipo real con el sistema operativo que emula. Para formar estos patrones de respuesta hace uso de las bases de estos programas utilizados para los ataques, con lo que se pueden simular nuevos sistemas agregando el patrón de comportamiento a la base. Entre los sistemas operativos emulados se encuentran:

- Windows 95, 98, NT, 2000, Me, XP
- Linux
- Mac OS
- Cisco IOS

Honeyd utiliza direcciones IP no asignadas en la red mediante un procedimiento denominado "*IP takeover*". De esta forma simula la presencia de un conjunto de máquinas en un determinado rango de direcciones IP. El comportamiento de cada terminal o personalidad se establece por separado y depende del sistema operativo y de los servicios de red activos. Otros parámetros que se pueden emular son ratio de pérdida de datagramas y la latencia de comunicación, lo que aporta una alta cuota de realismo a la red creada.

La Imagen 2.9 muestra como *Honeyd* virtualiza una red con cuatro sistemas, cada uno con una personalidad distinta (Linux con kernel 2.6, Windows NT, FreeBSD y Windows XP Home).

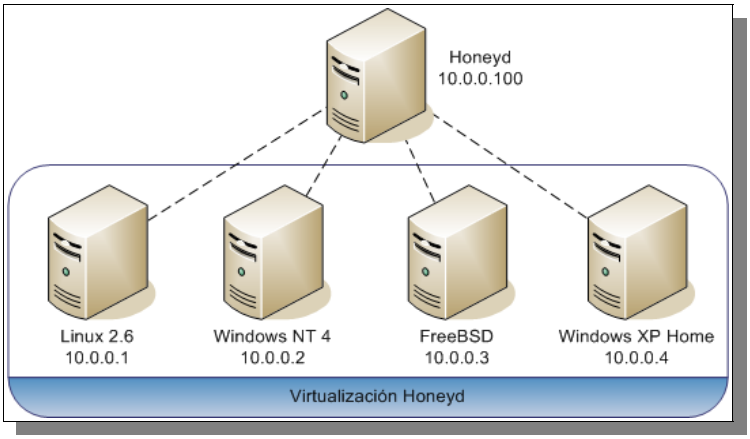


Imagen 2.9: Esquema de una Honeynet virtualizada con Honeyd

2.4.8 PHP.Hop

PHP.Hop [PHP06] es un *honeypot* de bajo nivel de interacción que provee una interfaz Web falsa y así detectar una gran variedad de ataques Web. A diferencia de los *honeypots* vistos hasta ahora, este está dedicado enteramente al servicio Web. Según su autor Laurent Oudot, *PHP.Hop* permite identificar un gran rango de herramientas automáticas que buscan vulnerabilidades basadas en los servicios, las cuales son explotadas mediante worms.

Una característica fundamental es la capacidad de agregar módulos específicos como *HipHop* [PHP06], *PhpMyAdmin* [PhpD98], etc.; los cuales son simulaciones de herramientas Web específicas con ciertas vulnerabilidades. Así un atacante puede creer que está atacando *PhpMyAdmin,* un famoso front-end Web de la base de datos *MySQL*.

2.4.9 The Google Hack Honeypot (GHH)

GHH [GHH05] es un *honeypot* diseñado para detectar a los atacantes que utilizan los motores de búsqueda como herramienta de *hacking* contra diferentes recursos. El extenso número de páginas localizadas por el motor de búsqueda de *Google* [GOOG08] y el incremento del uso de aplicaciones Web, tales como message boards y herramientas de administración remota, han dado lugar a un aumento en el número de aplicaciones Web mal configuradas y vulnerables disponibles en Internet.

"*Google Hacking*" consiste en explotar la gran capacidad de almacenamiento de información de *Google*, buscando información específica que ha sido añadida a las bases de datos del buscador. Si las búsquedas son orientadas a ciertas palabras claves, es posible que ayuden a encontrar información o puntos de entrada sensibles a posibles ataques. En otras palabras, *Google Hacking* es buscar en *Google* información sensible, generalmente, con fines maliciosos.

Por ejemplo una búsqueda como la del Fragmento 2.2 permitirá encontrar diferentes servidores de *SquirrelMail* [SMail04] indexados por el buscador:

```
"SquirrelMail version 1.4.4" inurl:src ext:php
```

Fragmento 2.2: Cadena para buscar servidores SquerrelMail

De esta manera un atacante que posea un exploit para *SquirrelMail* obtendrá de forma rápida una gran cantidad de servidores que lo contenga, para hacer uso de dicho exploit.

O para buscar *logs* de alguna herramienta, como el caso de *DrWatson* [Msft07] como lo muestra el Fragmento 2.3,

```
"Microsoft (R) Windows * (TM) Version * DrWtsn32 Copyright (C)" ext:log
```

Fragmento 2.3: Cadena para buscar logs de DrWatson

El software inseguro, combinado con el poder de un motor de búsqueda, da lugar a un gran vector de ataque para los usuarios maliciosos. *GHH* es una herramienta que permite combatir esta amenaza, se encuentra escrito en PHP y se distribuye bajo la *GNU Public License* [GNUP07].

El funcionamiento de *Google Hack Honeypot* es muy sencillo, emula vulnerabilidades de aplicaciones Web y permite que sus páginas sean indexadas por los motores de búsqueda. Estas no son visibles a usuarios comunes de la Web, sino que están diseñadas para ser encontradas por motores de búsqueda mediante los llamados "*enlaces transparentes*". Estos permiten reducir falsos positivos y evitan además dejar la huella digital del señuelo. Este enlace se coloca en páginas Web ya existentes ya que no afecta su funcionamiento.

El *honeypot* se conecta a un archivo de configuración, él cual establece diferentes características y registra en un archivo de *log* los datos del acceso, dirección IP, etc. *GHH* utiliza un mínimo de 3 archivos, incluido el que implementa el *honeypot*, el archivo de configuración, y el *log*.

En la Imagen 2.10 se diagrama un sitio Web, se muestra como se le integra *GHH* y como se relacionan cada uno de los archivos que lo conforman.

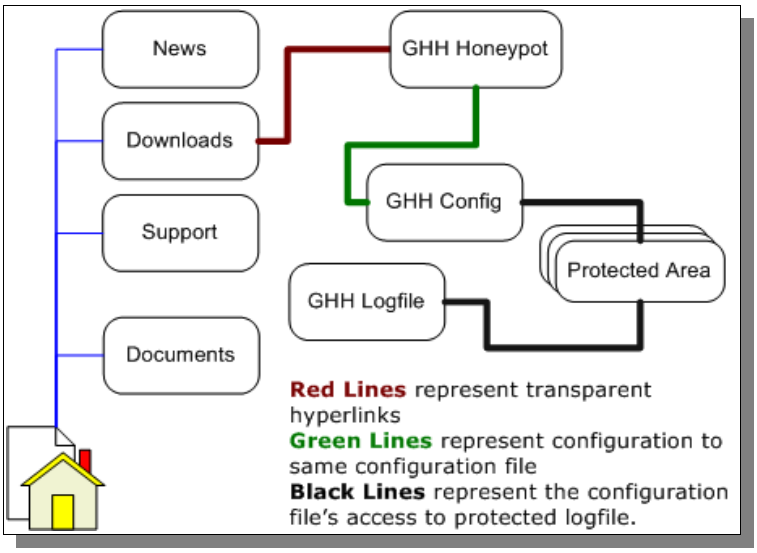


Imagen 2.10: Esquema de sitio Web usando GHH

2.5 Conclusión

En sus inicios las comunidades de seguridad se mostraron cautas a los *honeypots*. Debido a que no se llegaba a un acuerdo de su definición. Había interpretaciones que definían los *honeypots* como dispositivos para atraer y engañar atacantes; otros sostenían que esta tecnología estaba diseñada para detectar ataques. A principio del siglo XXI estas concepciones se unificaron, ganando día a día más adeptos.

Poco a poco los *honeypots* comenzaron a evolucionar, limando diferentes características. Uno de los grandes retos de las tecnologías de *honeypots,* es la configuración. Dado que son sistemas que se utilizan para atraer atacantes, pueden convertirse fácilmente en herramientas para propiciar ataques. Un punto de inflexión importante fue el surgimiento de la herramienta *Honeywall*. Esta propició una forma relativamente fácil de configurar de forma segura una *honeynet* para que no sea utilizada como parte de un ataque. Mostró como una configuración empaquetada permite el rápido despliegue de *honeynets,* facilitando el trabajo de los administradores de seguridad de las organizaciones.

Se deben tener presentes muchas variables al momento de implantar un *honeypot,* por ejemplo, su nivel de interacción, su comportamiento, sus características, sistema de implantación: sistema Windows, Linux, etc. Una vez que el sistema operativo ha sido elegido, se deben establecer los servicios que se deseen ejecutar: `SMTP`, `HTTP`, `FTP` o conexión a una base de datos. Si se comete un error en la configuración de los servicios, se podrían perder ataques, pruebas o escaneos de gran valor.

El tratamiento y almacenamiento de los registros obtenidos de un ataque es un punto que no se ha logrado resolver completamente en esta tecnología. Todas las implementaciones que se investigaron vuelcan sus registros en archivos de *logs*, en forma simple a un texto plano, aunque lo hacen para ganar tiempo y no descuidar al atacante. El procesamiento de la información obtenida se encuentra en una etapa muy primitiva todavía. Al detectar un ataque, por ejemplo, se podrían enviar los datos a un sistema remoto para almacenarlo y/o enviar alertas a los administradores.

Se ha visto en el desarrollo del proyecto los beneficios que brindan los *honeypots* a la hora de "*conocer a nuestros enemigos*". En escenarios de producción la posibilidad de conocer ataques nuevos e inesperados representan un gran valor dentro de la seguridad de una organización y en el ámbito de investigación y académico. A pesar de ello las organizaciones comerciales actualmente prefieren mantener alejados a los atacantes que aprender de ellos. Los ámbitos que si han adoptado su uso son las universidades, gobiernos, e incluso organizaciones que se encuentran vinculadas a la seguridad. Esta tendencia poco a poco está cambiando a medida que va evolucionando esta tecnología, y mostrando su potencial.

El futuro de los *honeypots* es muy prometedor. Un escenario del cual pueden formar parte es en la captura de atacantes expertos de los cuales se tiene poca información y cuyos objetivos son de alto valor. Por ejemplo se podrían emular sistemas de e-banking, e-commerce, etc.; e incluso tarjetas de créditos como *honeypots*, que luego puedan ser rastreadas.

Las *honeynet* Gen IV prometen llevar la obtención y él análisis de datos a su máximo extremo, recopilando datos para su posterior análisis de varias *honeynets* desplegadas por toda Internet. Este proyecto tiene como principal patrocinador a la *Honeynet Research Alliance* [HPRA05], la cual

cuenta con miembros de todas partes del mundo (por el momento Uruguay no es uno de ellos).

CAPÍTULO 3 ANÁLISIS Y DISEÑO

Luego del estudio del arte de la tecnología de *honeypots* [EsAr08], se presentan varias alternativas de caminos a seguir para llevar a cabo el diseño e implementación de un *honeypot.*

De forma de cumplir con los objetivos planteados, el grupo realiza un análisis del problema y elabora una propuesta de diseño que plasma en el documento "Propuesta de diseño" [Prop08]. En la primera sección se resumen de dicho documento los puntos más importantes que se han tenido en cuenta a la hora del desarrollo del proyecto.

En el resto del capítulo se presentan las decisiones tomadas para el diseño de la solución, se abordan los temas de captura, normalización y centralización de datos. Se presenta el modelo de datos *IDMEF* [NWGp07], utilizado para llevar a cabo la normalización. Se discuten las ventajas y desventajas de realizar el monitoreo en forma centralizada, así como también el disparo de alarmas desde un único punto. Por último se describen los componentes participantes de la solución.

3.1 Puntos a tener en cuenta

Para cumplir con los objetivos del proyecto, el diseño de la solución debe contemplar una serie de pautas que se enumeran a continuación,

- **Implantación de una honeynet.** El objetivo principal del proyecto de grado es la implantación de un *honeypot* en Facultad de Ingeniería[7]. La implantación de una red de *honeypots* es un objetivo más ambicioso que el planteado originalmente por los docentes [GSI06]. A partir de esto surgen otros puntos a tener en cuenta para escalar la solución.
  - **Arquitectura flexible.** Durante el estudio realizado se pudo observar dificultades en las implementaciones actuales de *honeynets* a la hora de agregar nuevos *honeypots* a la red ya implantada. Por tal motivo se quiere minimizar el impacto de agregar un *honeypot,* permitiendo que esta tarea sea fácil y transparente, sin necesidad de reconfigurar toda la *honeynet.*
  - **Escalabilidad de la arquitectura.** Se desea proporcionar una solución en la que sea posible agregar distintas implementaciones de *honeypots* y con distintos niveles de interacción, sin impactar el diseño ni el funcionamiento de la *honeynet* una vez implantada.
  - **Maximizar la captura de datos.** Dado que el objetivo de la *honeynet* es la recolección de información a partir de los ataques recibidos, es necesario que todo componente que la integre, de ser posible, sea capaz de registrar la intrusión. Esto permite tener información desde distintas perspectivas, según la visión de cada dispositivo que registre el ataque, lo que posibilita realizar un análisis más completo de los datos obtenidos.

7 Se toma como objetivo secundario el estudio del arte de dicha tecnología.

- **Normalización de los datos capturados.** Una red de computadores por lo general está compuesta por routers, firewalls, switches, nodos, etc. Como se dijo anteriormente se desea que todo componente que integre la *honeynet* sea capaz de aportar información. Teniendo presente la heterogeneidad de los componentes que conforman la *honeynet*, se pretende unificar la forma en la que se representan los datos obtenidos a partir de los ataques recibidos.

- **Centralización de la información.** Cuando un atacante toma el control de un *honeypot* (solo en caso que sea de alto nivel de interacción), puede alterar o eliminar la información obtenida del ataque. Es por ese motivo que no puede ser almacenada en el equipo comprometido. Teniendo esto presente se desea realizar la centralización de los datos obtenidos de toda la *honeynet* en un sistema externo a ella. Tener la información centralizada permite realizar un mejor análisis de datos, ya que se tiene toda la información disponible en un único punto.

- **Sistema de alarmas.** Teniendo presente que la *honeynet* no tiene valor de producción, todo acceso es considerado un ataque. Con tal motivo se desea estar informado de cada nuevo suceso, para ello se debe implementar un sistema de alarmas que notifique a los administradores ante cada nueva intrusión. El formato de la notificación debe poder ser extensible (enviar un correo, un mensaje de texto, una salida en la consola, etc.), así como la información en ella reportada.

- **Sistema de monitoreo.** La información obtenida debe poder ser accesible en cualquier momento por personas autorizadas. Se debe proveer un sistema capaz de generar reportes y estadísticas de los datos obtenidos.

- En la Imagen 3.1 se observa un esquema de alto de nivel de la solución buscada, teniendo en cuenta los objetivos antes mencionados. Se presentan varios *honeypots* que posiblemente tengan diferentes implementaciones y distintos niveles de interacción. Todos ellos pertenecientes a la misma red, envían sus datos a un servidor donde se centraliza toda la información recolectada en la *honeynet*. Cuando se detecta actividad, se activa el sistema de alarmas que envía una notificación a los administradores. Por último se dispone el sistema de reporte y visualización de datos, que permite analizar los datos recolectados.

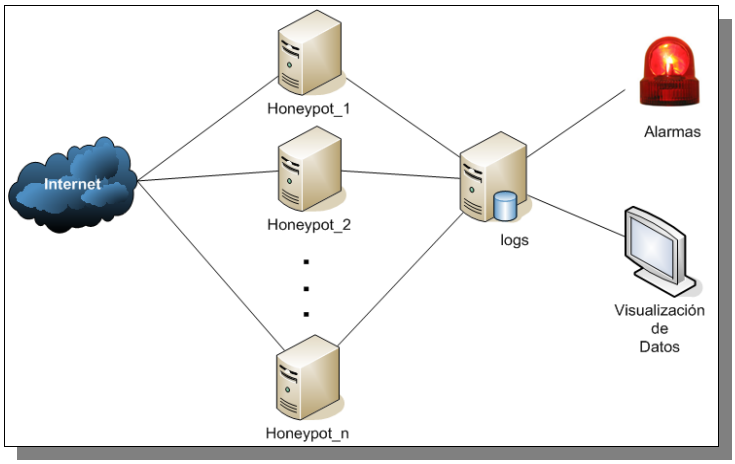


Imagen 3.1: Propuesta de diseño

3.2 Arquitectura básica

De acuerdo a la propuesta de diseño planteada [Prop08], los componentes principales de la arquitectura son:

- **Firewall**, este componente tiene por finalidad bloquear los ataques entrantes o salientes que deban ser mitigados para preservar la integridad de la red (bloqueando por ejemplo ataques tipo DoS o evitando ataques a objetivos externos desde la *honeynet*). Además de esto registra, sin ser detectado, todo el tráfico que se dirige desde y hacia la *honeynet*.

- **Honeypot**, es el encargado de capturar y registrar los ataques. En particular para este proyecto, dado que se trata con *honeypots* de bajo nivel de interacción, la captura de datos se realiza simulando servicios. Pero extendiendo la solución es posible integrar *honeypots* de alto nivel de interacción, con lo que la captura se realiza exponiendo un sistema real.

- **Repositorio**, es el responsable de la centralización de información recolectada, ya sea por los *honeypots* o por otro componente con la capacidad de registrar un ataque (como por ejemplo el firewall). En este componente se encuentran los sistemas responsables del despliegue de datos, generación de estadísticas y disparo de alarmas.

En la Imagen 3.2 se presenta la arquitectura ideada inicialmente donde el firewall es el único punto de acceso a la red que permite el ingreso de conexiones a los *honeypots* desde Internet. Conectado a él, se encuentra el repositorio para disponer allí la información recolectada.

Se realiza una modificación de este esquema para llevarlo a una *honeynet* de GenII creada por el *Honeynet Project*. Para ello se sustituye el firewall por la herramienta *Honeywall* como *gateway*. Además se busca elevar el nivel de modularización de los componentes, desacoplando el firewall del repositorio. De esta forma cada componente de la solución se comunica directamente con el repositorio para almacenar la información recolectada durante un ataque.

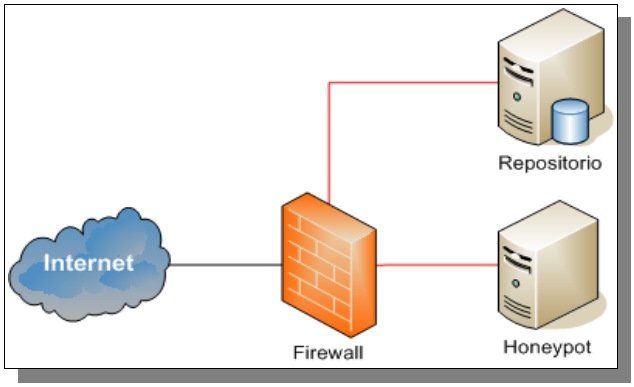


Imagen 3.2: Arquitectura basica de la solución.

Tal como muestra la Imagen 3.3, existe una vía de comunicación directa entre cada componente de la *honeynet* y el repositorio. Esta arquitectura establece comunicaciones separadas para cada componente, de forma que si uno de ellos es comprometido, los demás seguirán

registrando sus datos.

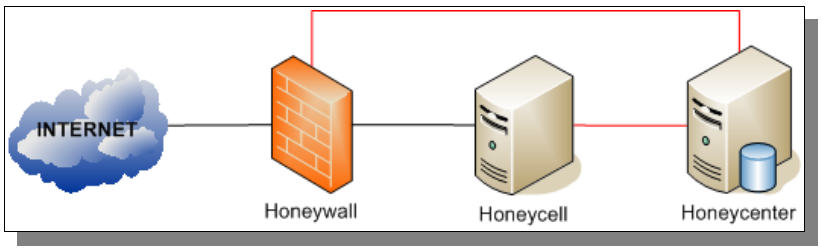


Imagen 3.3: Arquitectura donde cada componente se comunica con el repositorio.

De cara a *honeypots* de bajo nivel de interacción, esta arquitectura le oculta al atacante todo el tráfico entre el *honeypot* y el repositorio, por lo que no podrá atacar la comunicación y sus acciones podrán ser monitoreadas y registradas. Pero en sistemas de alto nivel de interacción se deben buscar mecanismos para ocultar dicho tráfico, de lo contrario un intruso puede atacar el repositorio y alterar los registros o tomar el control del componente.

Se deben tener en cuenta métodos para evitar ataques de suplantación de identidad, adicionalmente es necesario buscar mecanismos para registrar y validar las entidades que se conectan al repositorio.

3.3 Normalización

Uno de los problemas actuales de las infraestructuras de seguridad informática, es que se necesita obtener información refinada para controlar su rendimiento y correcto funcionamiento. Los *logs* son usados para registrar datos o información sobre quién, qué, cuándo, dónde y por qué (*who*, *what*, *when*, *where* and *why* [Alle01]) ocurre un incidente de seguridad. El problema principal radica en que cada componente utilizado para asegurar la red genera sus archivos de *log* con un formato independiente, que por lo general difiere del resto. En una infraestructura relativamente compleja se mantienen sistemas de diversos fabricantes y con diferentes tecnologías, cada uno con un formato propio de captura y almacenamiento de datos. Esto hace más difícil el análisis automático de los datos obtenidos.

Para facilitar el manejo de la información, es necesario organizarla y representarla de manera única, evitando así manejar diferentes formatos y simplificando el proceso de consolidación de datos provenientes de distintas fuentes (en el caso del proyecto las fuentes son, *honeypots*, *Honeywall*, etc.). El proceso de transformación a una representación única es denominado, normalización. Para llevar a cabo dicho proceso es necesario definir un modelo de datos que sea utilizado como estándar para la transformación.

La representación de datos de una manera normalizada, permite una mayor escalabilidad en la arquitectura de la solución. Esto se debe a que, el agregar un componente (*honeypot*) a la red no implica una modificación al modelo de datos seleccionado (en caso de que sea bien definido); sino que implica simplemente definir un proceso de transformación de los datos recolectados, a la forma normal. Luego de esto, los datos pueden ser centralizados y procesados sin mayores dificultades, como se realizaba antes de agregar el componente.

La forma en que se lleva a cabo la normalización es independiente del modelo de datos seleccionado. Se plantean dos modos de realizar esto, a saber:

- **Normalización directa.** Cada componente (o herramienta) de la solución se encarga de generar los datos ya normalizados. Seguramente sea lo más eficiente teniendo como principal objetivo un rápido procesamiento de datos. Esta opción puede implicar una modificación a los programas, con lo que se nota una desventaja al momento de agregar un nuevo componente.

- **Normalización indirecta.** Como se dijo anteriormente, por lo general las herramientas utilizadas para la captura de datos y registro de conexiones almacenan la información obtenida en archivos de texto plano. Con la forma de normalización indirecta se busca mantener el comportamiento estándar de la herramienta, generando su salida de la manera original, sin modificar el formato. Una vez que se escribe el registro en el archivo, se procesa y normaliza por un proceso independiente, de forma transparente para la herramienta. En la Imagen 3.4 se ejemplifica el proceso; se escribe el registro en el archivo de *log* (independientemente de la herramienta que lo genere), y luego es leído por un proceso que se encarga de la normalización y posterior disposición de los datos.

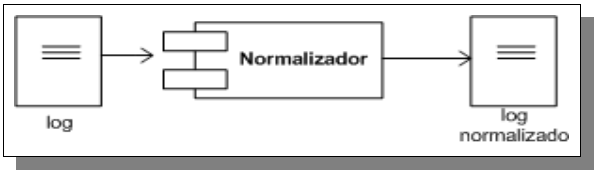


Imagen 3.4: Esquema de normalización indirecta de logs generados

Dado que el modo de "*normalización indirecta*" es independiente de la herramienta, ya que se basa en los datos, se puede aplicar para *honeypots* de cualquier nivel de interacción (siempre y cuando escriban sus datos en un archivo). Es una solución sencilla, modularizada y que no implica cambios en las implementaciones de *honeypots* usadas, permitiendo extender con facilidad la arquitectura.

Para poder llevar a cabo la normalización es necesario definir un modelo de datos. Para ello inicialmente se plantea un modelo como el que se presenta en la Imagen 3.5. Se deben tener en cuenta los datos capturados por todos los componentes de la red. En la estructura se definen dos entidades que representan los componentes: los que capturan datos de un ataque (Sensores) y los datos obtenidos (Alertas). Este modelo está enfocado a *honeypots* de bajo nivel de interacción, con lo que es muy limitado y se aplica solo a una realidad muy particular.

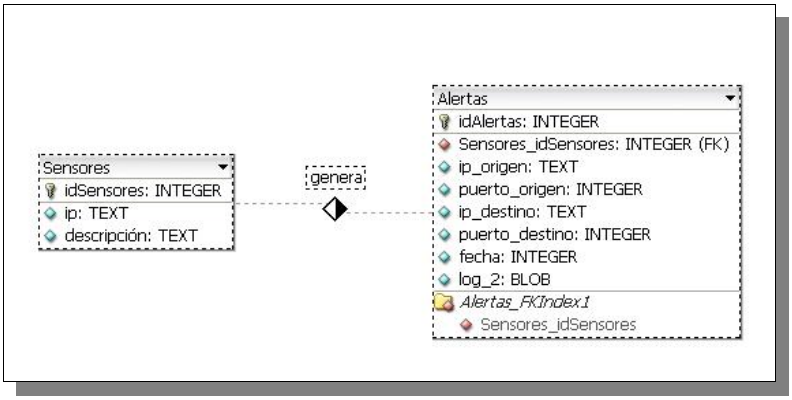


Imagen 3.5: Estructura definida para la normalización

Para completar o complementar el modelo definido se investigan los modelos de datos existentes para el reporte de incidencias de seguridad de CSIRTs (*Computer Security Incident Response Team*) [CSIR08].

De esta investigación surge la idea de utilizar un modelo de datos llamado *IDMEF* (*Intrusion Detection Message Exchange Format*) [NWGp07] (actualmente se encuentra en estado de RFC). El modelo fue propuesto por el grupo *IDWG* (*Intrusion Detection Working Group*) [IDWG05] de la *IETF* (*Internet Engineering Task Force*) [IETF08].

Entre los cometidos del *IDWG* se encuentra la redacción de un documento que especifique los requisitos de comunicación entre IDS's y entre un IDS y el sistema de gestión; además de la definición de un formato estándar para la representación de los mensajes intercambiados entre las partes. Se busca permitir la interoperabilidad entre distintos productos de detección de intrusos. El resultado del trabajo realizado por el *IDWG* es el modelo de datos *IDMEF* antes mencionado.

El modelo define el formato y la semántica de los mensajes de alertas, mediante una representación orientada a objetos. Aún no es un estándar oficial, su definición comenzó en el año 2000 y ha pasado por diferentes actualizaciones (Internet-Draft) hasta convertirse en el RFC 4765 en noviembre del 2006. Su última actualización se realizó en marzo 2007 [NWGp07].

3.3.1 Intrusion Detection Message Exchange Format

El formato *IDMEF* es una propuesta para el intercambio de mensajes entre los distintos componentes y dispositivos de seguridad, tales como sistemas de detección de intrusos, firewalls, etc. Es una especificación basada en XML para un formato de alertas de intrusión. El formato del mensaje es independiente del protocolo de comunicación, aunque el *IDWG* recomienda el protocolo IDXP (*Intrusion Detection eXchange Protocol*) [NWGX07] (no es obligatorio utilizarlo).

Un mensaje *IDMEF* se organiza de la siguiente manera: la clase de nivel superior para todo mensaje *IDMEF* es *IDMEF-Message*, y cada tipo de mensaje es una subclase de ésta. Actualmente, hay definidos dos tipos de mensajes: *Alerts* y *Heartbeats*. Dentro de cada mensaje se utilizan subclases para proporcionar información detallada.

Cada vez que un componente de captura de datos (analizador) detecta un evento para el que ha sido configurado, envía un mensaje de alerta (*Alerts*) a la consola de administración[8]

8 Puede haber más de un manejador por analizador.

(manejador). Dependiendo del analizador, un mensaje Alert puede corresponder a un único evento detectado, o a múltiples eventos. Los mensajes de este tipo contienen información (subclases) sobre los ataques, tales como: fecha de detección de la intrusión (Detect Time), quien informa del ataque (Analyzer), información del atacante (Source), destino del ataque (Target), información de lo que ha sucedido (Classification), fecha y hora de creación de la alerta (Create Time), fecha y hora del analizador (Analyzer Time), evaluación de la alerta por parte del analizador (Assesment) e información adicional que permite realizar su extensión (Additional Data).

El analizador usa mensajes *Heartbeat* para indicar su estado actual al manejador.

Las relaciones entre los componentes principales del modelo de datos se muestran en la Imagen 3.6.

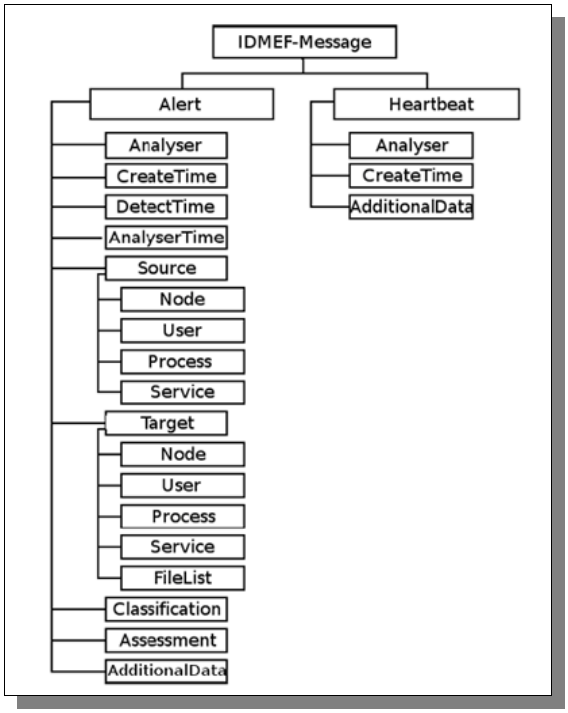


Imagen 3.6:Principales clases del modelo IDMEF

La clase Source contiene información referente a las fuentes del ataque, almacenando información sobre la máquina o dispositivo (dirección IP, red, etc.), usuario, servicio (puerto, protocolo, etc.) y proceso que provoca el ataque. Mientras que la clase Target contiene el mismo tipo de información que la clase Source, pero con referencia al objetivo del ataque. Además contiene información sobre los archivos creados, modificados o borrados en la máquina atacada.

Como se mencionó anteriormente, el modelo cuenta con más clases de las que se muestran en el diagrama. Por ejemplo, se cuenta con la subclase ToolAlert que contiene información sobre la herramienta utilizada para el ataque. La subclase OverflowAlert que contiene información referente a ataques de *overflow* [Dona02], donde se puede almacenar el programa desde el que se ejecuta el ataque (no es el programa atacado), el tamaño y los datos del desbordamiento (el buffer parcial o total). Por más información sobre el modelo puede referirse al documento "*Modelo de datos*" [MoDa08] proporcionado por el grupo.

Para concluir con la presentación del modelo se presenta un ejemplo, que se muestra en el Fragmento 3.1, de un ataque de port scanning a los puertos 5-25,37,42,43,53,69-119,123-514 reportado en formato XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<idmef:IDMEF-Message version="1.0" xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="hq-dmz-analyzer62">
      <idmef:Node category="dns">
        <idmef:location>Headquarters Web Server</idmef:location>
        <idmef:name>analyzer62.example.com</idmef:name>
      </idmef:Node>
    </idmef:Analyzer>
    <idmef:CreateTime ntpstamp="0xbc72b2b4.0x00000000">
      2000-03-09T15:31:00-08:00
    </idmef:CreateTime>
    <idmef:Source ident="abc01">
      <idmef:Node ident="abc01-01">
        <idmef:Address ident="abc01-02" category="ipv4-addr">
          <idmef:address>192.0.2.200</idmef:address>
        </idmef:Address>
      </idmef:Node>
    </idmef:Source>
    <idmef:Target ident="def01">
      <idmef:Node ident="def01-01" category="dns">
        <idmef:name>www.example.com</idmef:name>
        <idmef:Address ident="def01-02" category="ipv4-addr">
          <idmef:address>192.0.2.50</idmef:address>
        </idmef:Address>
      </idmef:Node>
      <idmef:Service ident="def01-03">
        <idmef:portlist>5-25,37,42,43,53,69-119,123-514
        </idmef:portlist>
      </idmef:Service>
    </idmef:Target>
    <idmef:Classification text="simple portscan">
      <idmef:Reference origin="vendor-specific">
        <idmef:name>portscan</idmef:name>
        <idmef:url>http://www.vendor.com/portscan</idmef:url>
      </idmef:Reference>
    </idmef:Classification>
  </idmef:Alert>
</idmef:IDMEF-Message>
```

Fragmento 3.1: uso de IDMEF para reportar un ataque de port-scanning

3.3.1.1 Aplicación de IDMEF

El modelo de datos presentado es muy potente, principalmente debido a que fue desarrollado teniendo como objetivo primario la intercomunicación entre distintos sistemas de detección de intrusos. Esto implica que en algunos escenarios, como es el caso de este proyecto, las herramientas utilizadas para la captura de datos (*honeypots*) no obtengan toda la información necesaria para hacer un uso extensivo del mismo y sacarle el mayor provecho. Al aumentar el nivel de interacción de los *honeypots*, la cantidad y granularidad de la información obtenida de los ataques también aumenta, aprovechándose mejor las capacidades del modelo y justificando su utilización.

En *honeypots* de bajo nivel de interacción la información que es posible obtener de un ataque es:

- *Honeypot* atacado
- IP y puerto origen del ataque (fuente)
- IP y puerto atacados (destino)

- Fecha y hora de realización del ataque

- En caso que el servicio explotado esté siendo emulado al momento del ataque, se puede obtener información más específica sobre lo que en realidad hizo el atacante. Por ejemplo se puede obtener usuario y password utilizados para acceder a una cuenta, URL introducida para generar el ataque, etc.

Esta información es modelada por las subclases Analizer, Source, Target, AnalizerTime y AdditionalData respectivamente. Adicionalmente se puede definir un valor para la severidad del ataque recibido (clase Assessment) y una clasificación del mismo (clase Classification[9]).

La decisión de adoptar el modelo *IDMEF* se basa en los siguientes 3 puntos,

- La oportunidad de utilizar un RFC, que está en camino de convertirse en un estándar oficial.

- Es un modelo extensible que se puede utilizar para cualquier reporte de incidencias, y es adaptable en caso de utilizar *honeypots* de mayor nivel de interacción.

- La existencia de un framework construido en base a este modelo, solucionando la normalización de datos (además de cubrir otras características que se detallan en el siguiente capítulo).

3.4 Centralización de datos

Los datos obtenidos una vez normalizados, se almacenan de forma centralizada en una única locación. El almacenamiento de la información no se realiza dentro de los *honeypots*, ya que no sería una solución extensible, y la información corre peligro cuando se compromete el sistema; sino que se concentra en un terminal independiente de la red de captura de datos (repositrio) al que los atacantes no tienen acceso (o no deberían tener la posibilidad de acceder a él). En las Imágenes 3.2 y 3.3 se presenta el repositorio en una subred entre los *honeypots* y el *Honeywall*.

Con este esquema se pretende obtener mayor seguridad frente a un ataque a la *honeynet*, almacenando la información obtenida fuera de los *honeypots*, proveyendo una manera ágil de organizar, asegurar y controlar los *logs*, además de establecer una única fuente de datos para el análisis.

Se recuerda que todo componente de la red implantada (en el caso particular de este proyecto son los *honeypots* y el *Honeywall*) es capaz de obtener información sobre ataques perpetrados a la *honeynet*. La centralización de datos alcanza a cada uno de esos componentes.

En la Imagen 3.7, se presenta un esquema donde se despliega el *Honeywall* y un *honeypot* que transmiten los datos obtenidos al repositorio donde se centraliza la información.

9 El modelo no especifica como debe ser clasificada una intrusión, sino que determina el formato que debe tener la alerta generada a raíz de ésta.

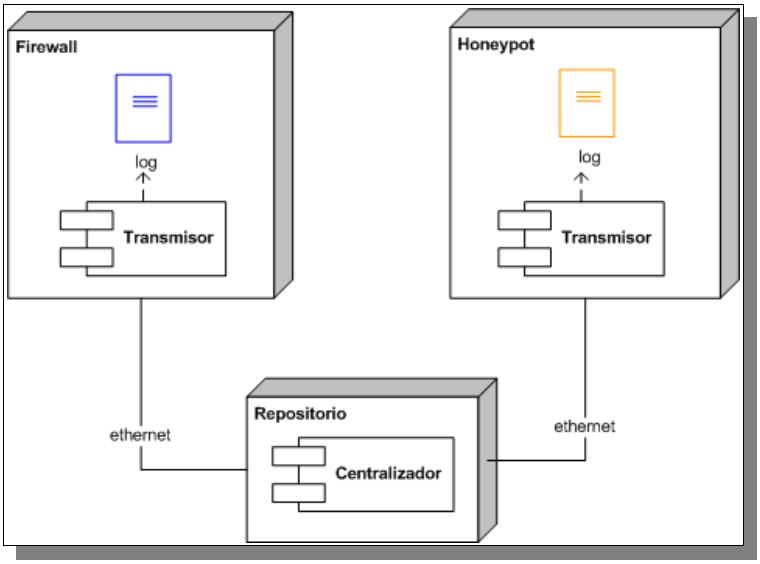


Imagen 3.7: Esquema de transmisión de los datos capturados al repositorio.

Al centralizar los datos en otro equipo se introducen nuevos problemas a tener en cuenta:

- Formato en que los datos deben ser enviados. Dada la cantidad de eventos que se pueden generar, enviar los datos en texto plano puede consumir mucho ancho de banda. Por lo que la performance de la red puede disminuir considerablemente, pudiendo incluso llegar al caso de congestión y pérdida de datos. Se debe entonces, buscar mecanismos para solventar esto. Una solución posible es realizar la compresión del mensaje antes de su envío, esto trae como contrapartida un consumo extra de CPU por los cálculos necesarios para llevar a cabo esta tarea (lo que lleva a una demora en el envío del dato).

- Determinación del responsable de transmisión de los datos. Para definir un responsable para la transmisión de datos se debe establecer un sentido en la comunicación entre las partes. Para definir esto se tienen dos vías posibles, es el repositorio quien consulta los componentes de captura (*honeypots* y *Honeywall*) por nuevos datos o son los componentes quienes los envían al repositorio.

  - En el primer caso es necesario que el repositorio conozca a todos los componentes para poder consultarlos. Por lo que se debe establecer la frecuencia y modo de selección del siguiente elemento a consultar (se debe establecer una planificación por ejemplo, round robin, secuencial, etc.). En este caso al integrar *honeypots* de alto nivel de interacción pueden surgir problemas, ya que cuando sea comprometido se intentará acceder a él, poniendo al atacante alerta o dándole otro posible blanco.

  - En el otro caso, es cada componente el responsable del envío de la información. Cada uno de ellos sabe que tiene nuevos datos disponibles para enviar al repositorio. El problema radica nuevamente con *honeypots* de alto nivel de interacción, donde es necesario ocultar los mecanismos de envío de datos para que el atacante no se percate que sus acciones están siendo registradas.

- Frecuencia de transmisión de los datos. La frecuencia con la que los datos son actualizados es dependiente del responsable de transmisión y puede llegar a ser un

problema. Se identifican tres modos:

- Bajo demanda, en este caso es el centralizador quien realiza la solicitud. La idea es que el *honeypot* (o dispositivo de captura de datos) ante una petición específica del repositorio, transmita los últimos registros desde el último pedido. El problema en este enfoque es que ante cada nuevo componente se debe analizar la forma en que se deben realizar las peticiones. Pero las peticiones a ese nuevo componente dependen de sus capacidades de configuración (por ejemplo los routers por lo general pueden enviar sus datos por *Syslog* a otro equipo).

- Periódico. La mayoría de los componentes permiten el envío periódico de sus registros a un equipo o servicio particular. En este caso se debe establecer si el intervalo es de tiempo o de cantidad, por ejemplo, enviando los registros capturados en los últimos minutos o las últimas 10 capturas. Para esta alternativa, los sistemas de software deben ser modificados o crear tareas que deben ser ejecutadas mediante un temporizador (o planificador). El problema en este caso es si se desean utilizar *honeypots* de alto nivel de interacción, ya que los registros deberían ser almacenados temporalmente de forma local y esta alternativa no es viable ya que en caso de que sea comprometido un atacante puede darse cuenta del mecanismo y alterarlo.

- En tiempo real. El envío de datos en tiempo real parece la opción más adecuada de cara a la disponibilidad de la información, se puede obtener un registro en el mismo momento en que ocurre un ataque, independientemente del nivel de interacción del *honeypot*, así mismo routers, firewalls y demás componentes tienen la posibilidad de distribuir la información de esta manera. Un problema a tener en cuenta en este caso es la carga en el tráfico de red que se genera, ya que en el caso de un ataque masivo cabe la posibilidad que se congestione la red y se pierdan paquetes (problema que suele ocurrirles a los NIDS), por lo que se deben buscar mecanismos para que esto no ocurra.

- Asegurar la transmisión. Los datos obtenidos son el activo más importante generado por los componentes de la arquitectura. Se debe contar con mecanismos para evitar que un atacante comprometa esta información. Se pueden tener problemas de pérdida, inyección de *logs*, o modificación de los datos. Para solucionar este problema se piensa en una comunicación SSL/TLS entre los componentes y el repositorio, evitando la modificación de datos en el canal y ataques tipo "*man in the middle*". [10] Adicionalmente se debe contar con mecanismos para validar y autenticar las entidades que se conectan al repositorio.

En base a los casos vistos anteriormente, el modo de centralización seleccionado para este proyecto tiene como responsables de la transmisión de información a los componentes que realizan la captura de datos. Se entiende que de esta forma se libera de responsabilidad al repositorio (ganando cohesión ya que se encarga principalmente de la centralización); y vemos que agregar nuevos componentes de captura no implica modificaciones en la forma de transmisión de datos de la red ya implantada. En base al modelo de datos seleccionado, los datos tendrán un formato XML (como lo define *IDMEF*).

10 Cabe recordar que la comunicación entre el repositorio y los componentes es mediante una interfaz de red secundaria y en teoría otro segmento de red distinto a por cual se realiza el ataque, por este motivo un atacante no debería ser capaz de comprometer este canal de comunicaciones.

En cada componente se realiza la compresión de la información a enviar. Se entiende que el consumo de CPU es mínimo, teniendo en cuenta el tipo y tamaño de la información a transmitir, y se gana en un mejor aprovechamiento de la red para la transferencia de datos. El envío al centralizador se realiza en tiempo real, se implementa de este modo para hacer disponible la información en el mismo momento. Se tiene presente que hay ataques que generan mucha carga a la red, como por ejemplo port scannings o ataques tipo DoS, pero esto se intenta mitigar al comprimir los datos enviados (generando paquetes más pequeños). Por último la comunicación entre los componentes de captura de datos y el s donde se centraliza la información, es encriptada usando el protocolo SSL/TLS [TLSe07].

3.4.1 Repositorio de datos

En cuanto al repositorio, se debe tener en cuenta el modo de almacenamiento de la información obtenida. Dado que ésta es enviada en forma normalizada y en tiempo real, se decide que sea accesible en el mismo instante en que es recibida.

Este esquema es una alternativa al utilizado por "*The honeynet .BR Project*"[HPBr02], en el cual los datos son procesados desde un archivo de *log* para poder obtener la información a desplegar y mediante scripts se generan estadísticas deseadas.

A pesar de la existencia de herramientas para el manejo de estos archivos de *log,* se presentan varias limitaciones a la hora de su procesamiento, teniendo en cuenta el uso que se les pretende dar y que se desean generar consultas complejas para analizar los resultados. Cuestiones como el tamaño del archivo de *logs,* la performance y complejidad al analizar muchos datos, el acceso concurrente al archivo para la escritura y generación de estadísticas, la escritura atómica de registros que puede no dar la información completa en el momento deseado, son razones por las que no se desea utilizar estos archivos para almacenar los registros.

Se ve entonces la necesidad de utilizar una arquitectura más robusta para el almacenamiento de los *logs*, como un manejador de base de datos (DBMS). Esto trae ventajas al momento de procesar los datos, ya que en vez de utilizar programas específicos para obtener la información almacenada se utiliza el lenguaje SQL. Además se facilita el manejo de grandes volúmenes de información, se aumenta la velocidad de respuesta a las consultas y se independiza la visualización y tratamiento de los datos almacenados. Con esto se gana una mejor performance en la solución.

3.5 Monitoreo y Análisis

Al contar con los datos normalizados y centralizados, se puede realizar un monitoreo y análisis más completo sobre los mismos. Al hacer uso de un DBMS, se tiene la posibilidad de usar muchas herramientas para realizar el análisis de datos.

Para realizar el análisis y monitoreo de datos se evalúan diferentes posibilidades:

- Contar con un componente que realice el monitoreo de datos en forma externa al repositorio. Para implementar esto se puede:
  - Permitir el acceso al DBMS desde fuera del equipo. Esto implica que se debe modificar la arquitectura para evitar ataques al DBMS. Se deben definir políticas de acceso a la información, controlar, registrar y monitorear dichos accesos. Esta opción hace más

compleja la arquitectura de la solución y dado los tiempos y alcance del proyecto se opta por no implementarla.

- Consultas via SNMP[11] [SNMP90] (*Simple Network Management Protocol*). Este protocolo define cuatro entidades, administradores, agentes, base de información y consola de administración. A grandes rasgos, en este esquema, los agentes envían información sobre los recursos que monitorean a los administradores, ya sea por "*iniciativa propia*" o porque un administrador así lo solicita (polling). Implementar este modo de monitoreo implica crear los agentes en el repositorio para que consulten la información que se encuentra en la base de datos. A modo de consola de administración se pueden utilizar herramientas open source, como por ejemplo *Cacti* [Cacti04] o *JFFNMS* [JFFN06]. La principal desventaja es que posee un nivel de funcionalidad limitada con datos orientados a objetos o basados en XML. Esta opción resulta poco extensible en cuanto al nivel de datos presentados, por lo que se opta por no implementar el sistema de monitoreo de este modo.

- Montar un sitio Web en el servidor (*HoneyCenter)* para realizar el monitoreo de datos en forma remota. La implantación de este sistema se basa en la configuración de herramientas open source con diferentes propósitos. Por ejemplo, se puede utilizar el motor de aplicaciones *Tomcat* [Tomc07] para montar el sitio y *Open Reports* [OpRe08], [OpnR08], *Pentaho* [Pent08] u otro sistema para la generación de reportes (que corran sobre *Tomcat*) para permitir el acceso a los datos. Adicionalmente las herramientas de reporting permiten obtener gráficas y estadísticas, lo que potencia su utilización. De esta manera no se agrega mayor complejidad a la solución y es posible monitorear el sistema haciendo uso de un navegador Web.

En base a las opciones vistas anteriormente se decide implementar un sistema de monitoreo y análisis montado sobre un sitio Web. La ventaja principal de esta opción radica en el conocimiento de las tecnologías mencionadas[12] (y de las herramientas *Tomcat*, *Open Reports*, *Pentaho*, etc.), lo que implica una rápida puesta en producción del sitio Web, esta opción es la mejor teniendo en cuenta el alcance planificado. Además con esta solución se puede implementar un sistema de monitoreo más potente, por ejemplo creando Web Services para consultar la información (similar a como se trabaja con el protocolo SNMP). La seguridad en este caso, se implementa configurando el firewall del repositorio, permitiendo solo tráfico por el puerto HTTP (puede implementarse un sitio seguro bajo HTTPS para garantizar la confidencialidad de los datos).

3.6 Alarmas

Se recuerda que una *honeynet* no tiene componentes con valor de producción, con lo que no debería haber tráfico hacia la misma. Cuando se detecta actividad se registra la información obtenida centralizándola (como se vio anteriormente) e inmediatamente se dispara una alarma a los administradores informando lo ocurrido.

11 SNMP es un protocolo a nivel de capa de aplicación para monitorear y administrar dispositivos de red.
12 En la etapa de diseño se plantea el uso de las herramientas, pero no se ha evaluado cual se usará efectivamente; en la etapa de implementación se analizan y decide cual utilizar.

Para implementar el sistema de alarmas se definen dos componentes, encargados de:

- **Identificar eventos**. El primer componente del sistema es el encargado de identificar eventos. Identificar un evento que genere una alarma puede depender de varios factores. Por ejemplo puede ser necesario enviar una alarma cada 10 ataques desde una misma IP o cada cierta cantidad de conexiones a un puerto, o una por cada conexión detectada. Este componente es el encargado de procesar los eventos, definir la frecuencia de disparo y generar el contenido de la alarma.

- **Envío de alarma.** Una vez identificado el evento y generada la alarma, debe ser "*disparada*". El segundo componente es el encargado de realizar dicho disparo. Por ejemplo puede ser enviada por correo vía SMTP, vía SMS, realizar un `insert` en una base de datos[13], enviar un mensaje a la consola, etc.

El sistema así modularizado gana flexibilidad, se independiza el mecanismo de identificación de eventos y generación de la alarma, de su envío concreto. En la Imagen 3.8 se muestra un diagrama donde se identifican los módulos.

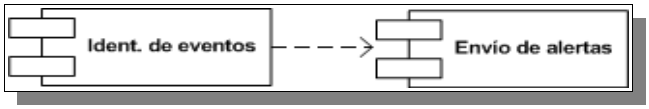


Imagen 3.8: Componente: Sistema de alarmas.

Para este proyecto se decide utilizar un sistema de alarmas sencillo, en el que ante cada evento registrado genere una alarma y se envíen los datos de la intrusión por correo (vía SMTP). Esta opción tiene como desventaja que si se realiza un port scanning o un ataque DoS a algún elemento de la *honeynet,* se genera un correo electrónico por cada conexión. Estos ataques generan miles de conexiones, con lo que se estarían generando miles de correos electrónicos. La solución así planteada puede llegar a generar un ataque DoS al servidor de correos. En particular el grupo desea estar notificado ante cada acceso o ataque recibido, por lo que, en caso de llegar a ocurrir esto se deberá desactivar el sistema de alarmas hasta que haya finalizado[14].

Se opta por establecer el sistema de alarmas en el repositorio para simplificar el diseño, además en caso de trabajar con *honeypots* de alto nivel de interacción no es viable tenerlo en el *honeypot*, ya que si el atacante logra comprometer el equipo, el sistema de alarmas queda expuesto.

3.7 Seguridad

Contrariamente a otro sistema informático, un *honeypot* adquiere valor cuando es atacado o comprometido. Uno de los aspectos principales a tener en cuenta es el control del flujo de datos, de manera de evitar que al ser comprometido sea utilizado para atacar otros sistemas. En el caso de una *honeynet*, para realizar el control del tráfico entrante y saliente de la red, el mecanismo utilizado es denominado *Honeywall* [Howl08].

13 En el caso de alarmas complejas es deseable mantenerlas en una base de datos para poder visualizarlas y mantener un registro de todos los eventos disparadores de la misma, con esto se puede realizar análisis estadístico en base a los eventos registrados.

14 En particular durante las pruebas realizadas con *honeypots* de bajo nivel de interacción, no se recibieron ataques de este tipo. Salvo las generadas por el grupo.

Honeywall es un dispositivo en modo de *gateway* que trabaja a nivel de capa 2 (en modo bridge) por lo que resulta invisible para un atacante. En el se recopilan varias herramientas necesarias para administrar una *honeynet* de segunda generación. En particular las que controlan el tráfico de entrada y salida a la red. Para realizar esto se cuenta con dos métodos de control de datos: limitación de conexiones salientes basada en `iptables` y el sistema de prevención de intrusiones.

La limitación de conexiones puede ser configurada para establecer límites sobre la cantidad de datos que pueden ser enviados desde cada sistema por unidad de tiempo, eliminando los datos luego de pasado el umbral fijado. Pueden controlarse las conexiones salientes para `TCP` o el número de paquetes cruzados en `UDP`, `ICMP` y `OTHER` (una categoría que detecta si el atacante está haciendo uso de un protocolo diferente para su comunicación).

La unidad de tiempo puede indicarse en segundos, minutos, horas o días. En la Imagen 3.9 se muestra el camino que realiza un paquete al momento de salir de *Honeywall*, en este caso se utiliza el primer filtro a cargo de las `iptables`.

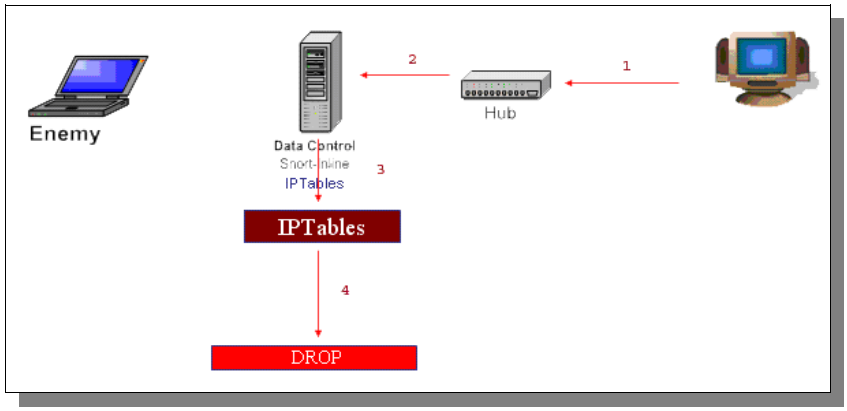


Imagen 3.9: Honeywall. Modalidad de filtrado por conexión limitada.

Una vez que el paquete es "*autorizado*" para su salida por `iptables`, es enviado a *Snort-inline* [SnIl08]. Éste es un sistema de prevención de intrusiones basado en *Snort* [Snor08], el cual es capaz de tomar acciones contra el tráfico saliente que tenga ataques conocidos. Las acciones que permite son descartar el paquete (*drop*), rechazarlo (*reject*) o reemplazarlo por un paquete no dañino.

En las imágenes 3.10 y 3.11 se muestran dos intervenciones de Snort-Inline. La primera en modo drop, descartando el paquete al ser un ataque reconocido. Snort-inline lee los paquetes en el espacio del kernel y lo compara contra una base de firmas y si coincide, se descarta.

En la segunda Imagen, se realiza el reemplazo de la respuesta por una no dañina. Snort-inline lee los paquetes y los compara contra una base de firmas, si coincide, el paquete se altera y sigue su curso, no se descarta. La configuración de estos parámetros dependen de cada implementación de red.

Si bien este dispositivo forma parte de la arquitectura básica de una *honeynet,* el objetivo adicional del mismo es aumentar la escalabilidad de la arquitectura, ya que de aumentar el nivel de interacción de los *honeypots* se debe tener un nivel mayor de seguridad (principalmente en lo que al tráfico saliente se refiere).

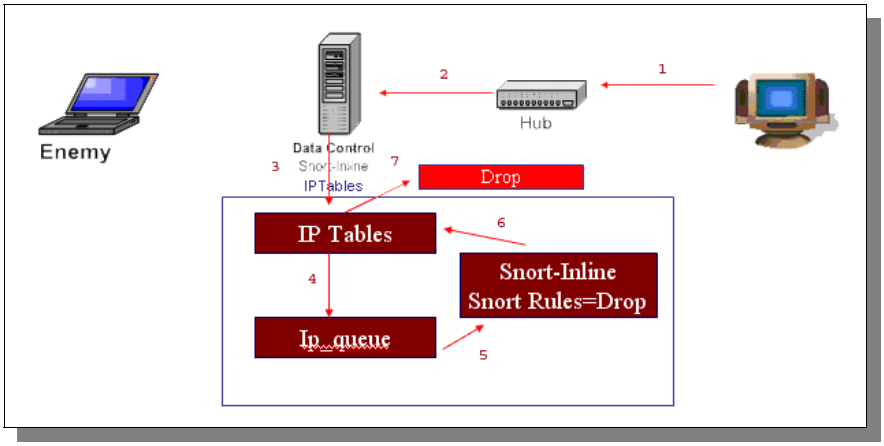


Imagen 3.10: Honeywall. Control de datos utilizando el modo drop.

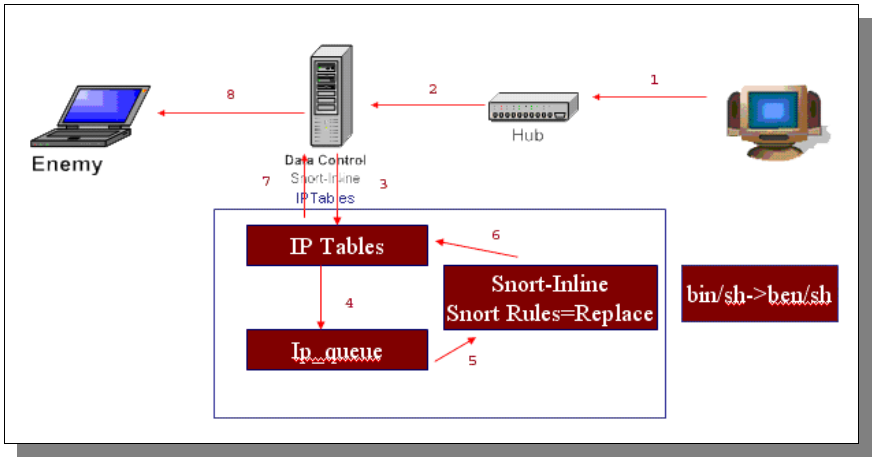


Imagen 3.11: Honeywall. Control de datos utilizando el modo de reemplazo.

Para una red de *honeypots* de bajo nivel de interacción no puede ocurrir que un atacante tome el control de sus objetivos. En todo momento se emulan los servicios brindados y el atacante no interactúa con un sistema real, a menos que el *honeypot* esté mal configurado o implantado (dando la posibilidad de que el atacante cumpla su objetivo).

3.8 Integración de los componentes

La arquitectura definida para la solución, como se vio en la Imagen 3.2, se compone de 3 componentes básicos:

- **Honeywall**: Como se describió anteriormente, es el encargado de regular el tráfico entrante y saliente de la *honeynet* de forma que si esta es comprometida no se utilice para realizar ataques hacia otros objetivos fuera de la misma. Dado que todo paquete que ingrese a la red es monitoreado por este componente, proporciona una capa complementaria de recolección de información. El tráfico registrado es normalizado y enviado al componente centralizador.

- **HoneyCell**: Se denomina *HoneyCell* al dispositivo que agrupa al componente de

captura de datos (alguna implementación de *honeypot* de bajo nivel de interacción como *Honeyd* o *Honeytrap*) y el componente para la normalización de los datos capturados. Además realiza el envío, en tiempo real, de los datos obtenidos al *HoneyCenter* vía SSL/TLS.

- **HoneyCenter**: El repositorio junto con el sistema de alarmas y monitoreo, descriptos anteriormente, es denominado *HoneyCenter*. En este componente se almacenan los datos capturados en forma centralizada, utilizando para ello un DBMS. Se mantiene aquí el sistema de monitoreo y análisis y el sistema de generación de alarmas. Se incluye además un sistema de control de tráfico y un firewall que filtra y registra los accesos al equipo.

El cuarto componente es una terminal de administración, un equipo destinado a controlar la solución.

En la Imagen 3.12 se presenta la arquitectura de la solución integrando todos los dispositivos antes mencionados. Además se identifican dos agentes, el administrador y el atacante. El atacante solo tiene acceso a los *HoneyCells* que se encuentren en la *honeynet*, pero para llegar a ellos debe pasar a través del *Honeywall*[15]. Este dispositivo con su componente de "*Control y registro de tráfico*" analiza, controla y registra toda comunicación entrante y saliente de la *honeynet*. El registro obtenido es normalizado, y luego transmitido al *HoneyCenter* para su almacenamiento. Por su parte el atacante, que intenta comprometer el *HoneyCell*, interactúa con un *honeypot* de bajo nivel de interacción allí implantado. El *honeypot* registra cada acción del atacante, mientras que el componente de normalización transforma al formato *IDMEF* cada registro obtenido, para luego transmitirlo al *HoneyCenter*. El *HoneyCenter*, haciendo uso de su componente de centralización recibe los datos ya normalizados y los almacena en la base de datos. El componente de identificación de eventos detecta que hay un nuevo registro, y en caso que así lo determine, genera una nueva alarma que es enviada a los administradores por el componente disparador de alarmas.

3.9 Conclusión

La selección de un modelo para el reporte de incidencias para sistemas de detección de intrusos no es una tarea sencilla, cada sistema tiene sus particularidades y necesidades específicas, por lo que se debe trabajar sobre un modelo de datos lo más general posible, que permita aumentar las capacidades del sistema con el menor impacto sobre éste. Se piensa que *IDMEF* es un buen ejemplo de un modelo altamente genérico, que se adapta en particular a las necesidades del proyecto, pero puede ser aplicado en distintos ámbitos dentro de un sistema de seguridad.

La posibilidad de centralizar toda la información obtenida le otorga un gran potencial al sistema desarrollado, ya que se permite un tratamiento de los datos "*online*" y en tiempo real. Pero se debe tener en cuenta que en *honeynets* distribuidas, donde cada *honeypot* puede estar ubicado en distintas regiones geográficas, subredes, etc., la solución implementada puede no ser la más adecuada.

15 Se recuerda que este dispositivo trabaja a modo de gateway a nivel de capa dos, por lo que resulta invisible al atacante.

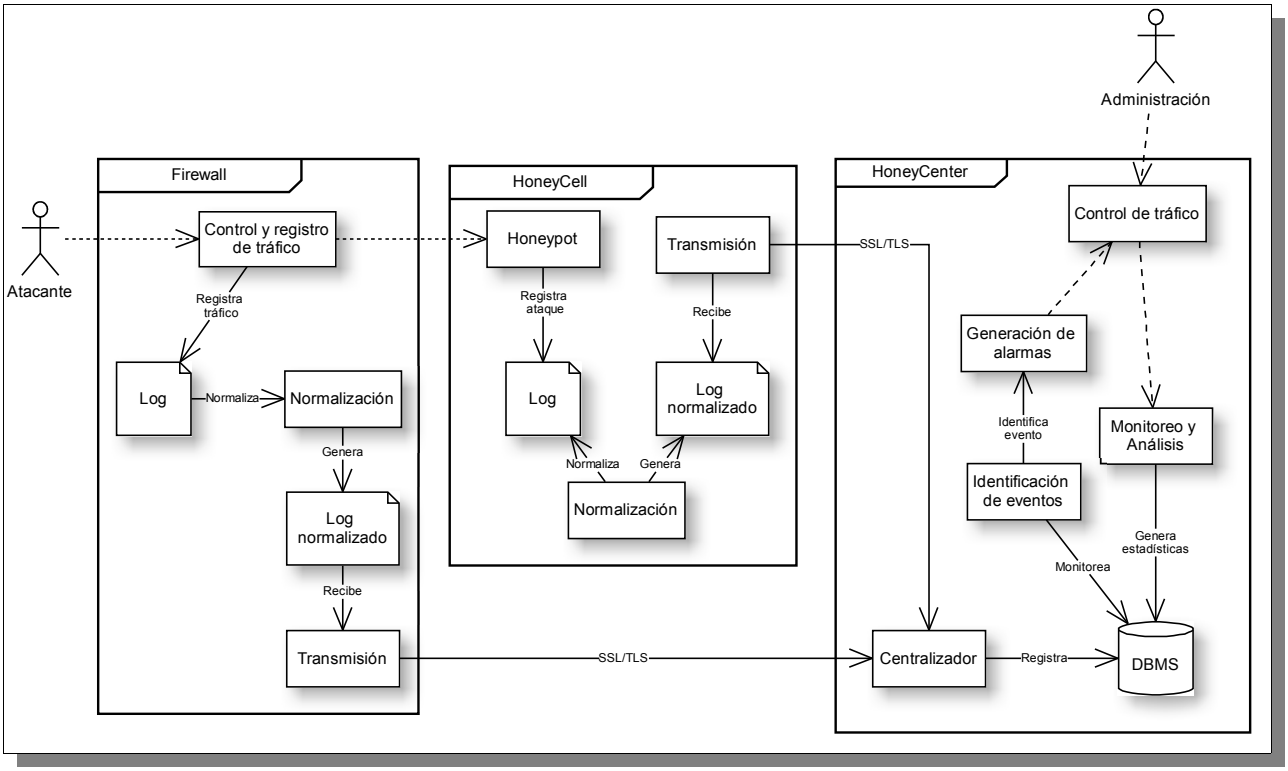


Imagen 3.12: Arquitectura de la solución

CAPÍTULO 4 IMPLEMENTACIÓN

En este capítulo se presentan las herramientas utilizadas para la implementación de la solución vista en el capítulo de diseño, justificando las decisiones tomadas en cada paso.

Dado que se busca facilitar la implantación y el rápido despliegue, los componentes son distribuidos en formato de *livecds*. Se discuten ventajas y desventajas de este modo de distribución. Se introduce a *Prelude IDS framework* [Prel04], [Prel08], un conjunto de herramientas creadas en base al modelo *IDMEF*, que ayudan a la centralización de datos.

4.1 Captura de datos

Para la captura de los datos se utilizan dos implementaciones muy conocidas y distintas de *honeypots* de bajo nivel de interacción, *Honeytrap* [Hont05] y *Honeyd* [Hond05].

Honeytrap, haciendo uso del sistema operativo base donde se encuentra instalado, se especializa en la captura de malware. Al ejecutarlo queda a la espera de una petición de conexión, la cual al producirse, si el puerto no se encuentra en uso lo abre y acepta petición procesando la conexión generando así el *log* correspondiente. Permite registrar dos tipos de ataques:

- Conexiones establecidas.
- Conexiones con envío de datos.

Honeyd por su parte se enfoca en la virtualización de sistemas y redes virtuales para la captura de datos. Según su configuración, los terminales virtualizados responden de distinto modo a los paquetes de fingerprint de *Nmap* [Nmap08] y *Xprobe* [Xpro08], este modo de respuesta es denominado personalidad. En el Fragmento 4.1 se define la personalidad llamada WinXP, que emula un sistema operativo *Windows XP Home Edition* [WinX08] con un servidor Web atendiendo en el puerto 80.

Honeyd permite registrar tres tipos de conexiones:

- Conexiones establecidas.
- Conexiones finalizadas.
- Paquetes que no pertenecen a ninguna conexión.

```
#ejemplo de una personalidad de Honeyd
create WinXP
set WinXP personality "Microsoft Windows XP Home Edition"
set WinXP default tcp action reset
set WinXP default udp action reset
set WinXP default icmp action open
add WinXP tcp port 80 "sh scripts/web.sh"
set WinXP uptime 234512
```

Fragmento 4.1: Definición de la personalidad WinXP con un servidor Web en el puerto 80.

4.2 Normalización y Centralización

Para llevar a cabo el proceso de normalización y centralización de datos obtenidos se evaluaron diferentes herramientas. El interés en las mismas se centró en que permitieran:

- Captar una nueva entrada en el archivo de *logs*.
- Verificar si aplica una determinada condición (regla) sobre la nueva entrada.
- En caso de aplicar una regla, permitir ejecutar una acción devolviendo una salida.

Algunas de las herramientas evaluadas son:

- *Logwatch* [LWAT07].
- *Swatch* [Atki95].
- *SEC* [SEC08], [Vaar08].
- *Prelude-IDS framework* [Prel08].

Como una primera alternativa se utilizó *SEC*[16], dado que éste permite de manera sencilla analizar archivos de *logs* especificando expresiones regulares (reglas), que identifican y parsean cada entrada del archivo. Una vez obtenidos los datos, se reestructuran generando el contenido de una alerta *IDMEF* en formato XML.

Una vez obtenido el contenido de la alerta, debe ser enviada al *HoneyCenter* para centralizarla. Para realizar esta tarea se pensó en utilizar *Syslog*, el cual sería el encargado de enviarla desde *HoneyCell* y recibirla en *HoneyCenter.* Luego la alerta sería obtenida nuevamente por *SEC* quien se encargaría de procesarla para centralizarla. Este esquema se observa en la Imagen 4.1.

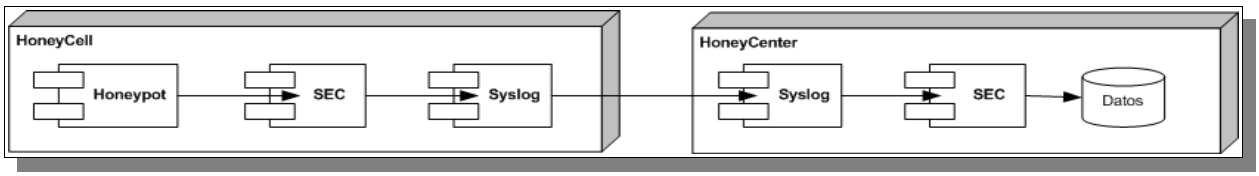


Imagen 4.1: Esquema envío de alertas mediante Syslog.

Una desventaja que se presenta es que se debe contar con otro componente que realice la reestructura y el armado de la alerta, ya que *SEC* solo se remite a la entrega de los datos parseados,

16 Dadas las características y versatilidad de la herramienta, el grupo realizó un estudio en profundidad de la misma. El documento denominado "*SEC, Simple Event Correlator*" [SEC08], es el resultado del estudio realizado.

la misma carencia la presentan *Logwatch* y *Swatch*.

Pero la desventaja principal refiere al mecanismo para el envío de la alerta al repositorio. Este posee varias limitantes al aplicarse esta solución:

- Los mensajes son enviados en texto plano, con lo que un atacante de interceptarlos puede fácilmente ver la información transmitida. Como solución a este problema se puede encriptar el *log* antes de entregárselo a *Syslog* y hacer el camino inverso al recibirlo en el centralizador.

- La configuración es poco flexible. A medida que se incrementa la cantidad de sensores se debe configurar el servicio para que realice el envío, además se deben tener métodos para identificar cada transmisor.

- No registra el origen de la fuente. Cada host que propaga un mensaje modifica la dirección IP registrada como origen del mensaje.

- Dado que se utiliza *Syslog*, este admite el reenvío mediante el protocolo UDP, lo que no garantiza que el destinatario reciba el paquete.

La tercera herramienta evaluada *Prelude-IDS framework* presenta un escenario totalmente diferente ya que:

- Es un framework para sistemas de detección de intrusos
- Su implementación se basa en el modelo de alertas *IDMEF*
- Permite unificar diferentes tipos de *logs* de diferentes herramientas

Prelude-IDS se basa en *sensores* que generan alertas y *managers* que se encargan de procesarlas. Cuando un sensor registra actividad envía una alerta en formato *IDMEF* al manager, permitiendo redirigir la salida a un archivo o una base de datos. En la figura 4.2 se observa la arquitectura del framework.

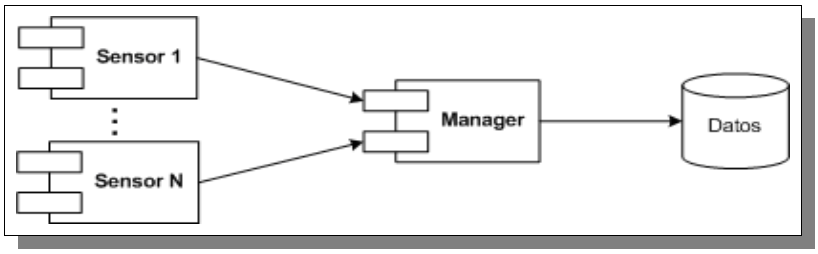


Imagen 4.2: Arquitectura de Prelude-IDS

El proyecto *Prelude* escribió su propia implementación siguiendo la especificación del modelo *IDMEF*, usando una estructura propia y preservando los tipos de datos originales. El framework se encuentra conformado por *Prelude Library, Prelude-LML, Prelude Manager, PreludeDB Library*.

Prelude Library es el corazón del framework, proporcionando una *API* (Application Programming Interface) para la creación de mensajes *IDMEF* y el manejo de las comunicaciones entre los sensores y sus managers. Además provee:

- **Comunicación segura:** la comunicación entre cada sensor y el manager, se establece mediante un protocolo SSL, encriptando la comunicación entre ambas partes para que no

sea intervenida. La autenticación de los sensores contra el manager se realiza mediante intercambio de certificados.

- **Eventos asincrónicos**: la generación de los eventos ocurridos se realiza de manera asincrónica, de forma de no afectar el envío de los mismos.

El componente *Prelude-LML* (*Prelude Log Monitoring Lackey*), monitorea los archivos de *logs*, para lo que cuenta con un sistema de plugins, como por ejemplo Single, un motor de expresiones regulares impulsado por la biblioteca *PCRE* [PCRE08].

Cada regla denominada *ruleset* especifica una expresión regular que identifica un *log* particular. Cuando el patrón de una alerta coincide con una expresión regular de una regla los datos son enviados al manager.

Dichas reglas se establecen en un archivo dentro de `/usr/local/etc/prelude-lml/ruleset/`. El Fragmento 4.2 muestra la regla para *Prelude-LML* que permite reconocer una entrada de *log* de *Honeytrap*, para el caso en que un atacante establece una conexión al *honeypot*.

```
#LOG:[2007-05-26 16:48:09]  * 22       No bytes received from X.X.X.X:YYYY.
regex=\* (\d+)\s+No bytes received from ([\d\.]+):(\d+).; \
classification.text=Reconnaissance Probe at port $1; \
id=40000; \
revision=1; \
analyzer(0).name=honeytrap; \
analyzer(0).manufacturer=http://honeytrap.mwcollect.org; \
analyzer(0).class=Honeypot-id; \
source(0).node.address(0).category=ipv4-addr; \
source(0).node.address(0).address=$2; \
source(0).service.port=$3; \
#target(0).node.address(0).category=ipv4-addr; \
#target(0).node.address(0).address=$2; \
target(0).service.port=$1; \
#assessment.impact.completion=succeeded; \
assessment.impact.type=recon; \
assessment.impact.severity=low; \
assessment.impact.description=A connection to honeytrap has been established.; \
last
```

Fragmento 4.2: Definición de una regla para Prelude-LML.

Prelude Manager es el encargado de recibir los datos desde los sensores y guardarlos en el medio especificado. Para aceptar datos primero se deben registrar los sensores. Además puede establecer comunicación con otros managers, de forma de replicar los datos recibidos.

PreludeDB Library permite comunicar el framework con diferentes DBMS, simplificando el acceso a los mismos evitando el manejo de sentencias SQL. Esta comunicación se realiza mediante la instalación de plugins. Las bases de datos soportadas son: *MySQL* [MySq95], *Postgres* [Posg08], *SQLite* [SQLi08].

Por la ventajas comparativas que presenta esta herramienta, se decide utilizar el *Prelude framework* para la normalización y la centralización de los datos capturados. Mientras que para el almacenamiento de los registros en forma centralizada se opta por utilizar el manejador de base de datos *MySQL*. Esta elección se debe a que el DBMS es soportado por *PreludeDB Library*, y es un motor ya conocido y utilizado anteriormente por los integrantes del grupo.

En la Imagen 4.3 se muestra (a grandes rasgos) el camino seguido desde la captura de datos en el *HoneyCell* (utilizando para ello una implementación de un *honeypot* de bajo nivel de interacción) hasta su centralización en el *HoneyCenter*.

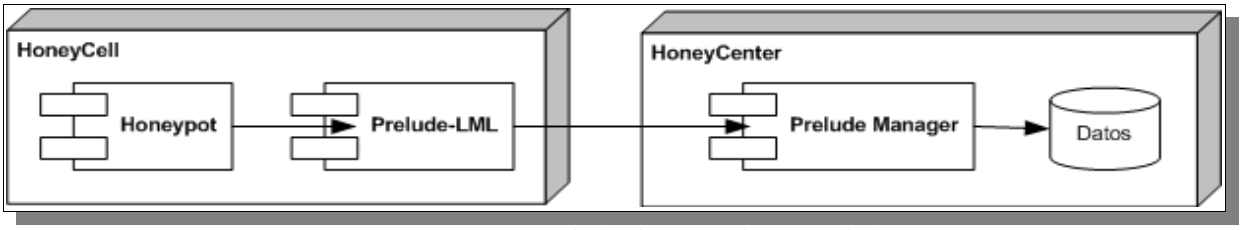


Imagen 4.3: Esquema envío de alertas mediante Prelude-IDS.

4.3 Monitoreo de datos

Un *log* de un dispositivo puede llegar a no revelar cierta información, pero al relacionarlo con registros de otros dispositivos se puede encontrar información valiosa. Por esa razón los *logs* generados sobre los ataques perpetrados a la *honeynet* desplegada, se encuentran normalizados y almacenados en un único punto (centralizados).

Se utiliza la herramienta *OpenReports*[17] [OpRe08] para la construcción de reportes, la cual provee una interfaz Web para su uso y administración. Cuenta con dos mecanismos de ejecución de reportes: programación (*scheduler*) para tomar una foto de los datos para su posterior consulta, y ejecución a demanda en la cual se obtienen los datos del momento. Este último mecanismo es el que se utiliza en la solución presentada.

Esta herramienta provee una manera fácil de creación de reportes, pudiendo ser de dos tipos, listados (ReportQuery) y gráficos (ChartQuery). En ambos casos, la manera de crearlo es indicando una consulta SQL. Todo los reportes permiten utilizar *parameters* los cuales son valores que se especifican al momento de ejecución. En la mayoría de los reportes se cuenta con los filtros del período de tiempo de análisis, indicando la fecha mínima y máxima.

Se implementa un conjunto de reportes para analizar la información almacenada en el repositorio. Se proveen reportes que muestran los puertos más atacados (según la cantidad de ataques recibos o según la cantidad de IP's origen de estos ataques). Diferenciando ataques masivos (desde una sola fuente) a un puerto, o ataques a un puerto por varias fuentes.

Para conocer las fuentes de ataques se cuenta con reportes que identifican las IP's que han realizado más ataques. Por ejemplo, se cuenta con un reporte que identifica todas las IP's por las cuales se ha recibido ataques, indicando la cantidad que han realizado, en cuantos días y cuantos puertos distintos han atacado. Otro ejemplo es un reporte que visualiza los ataques provenientes de IP's que han realizado ataques en más de un día.

Para tener más información sobre la procedencia de los ataques se asocia el número de IP al país de pertenencia. Pudiendo así identificar los países que han realizado más ataques, con cuantas cantidades de IP's distintas y la cantidad de días en que han realizado los ataques. También se

17 Por más información de *Open Reports* puede dirigirse al documento, "OpenReports" [OpnR08].

puede conocer todos los ataques o simplemente las IP's de ataque correspondientes a un país determinado.

En muchos casos después de conocer las tendencias de los ataques es necesario conocer que ataques componen esos grandes números. Para eso se tienen reportes que muestran a nivel de ataque todos aquellos que se realizan en un determinado período de tiempo, o todos aquellos realizados a un determinado puerto, o ataques a una determinada IP objetivo, o los realizados por una misma IP fuente. Obteniendo así más conocimiento de lo que ha sucedido, por ejemplo, identificando las secuencias de puertos accedidos por una determinada IP fuente.

Puede dirigirse al documento "Reportes Estadísticos" [ReEs08], para una lista competa de los reportes creados por el grupo (donde se especifica el motivo y la manera de la creación del mismo).

4.4 Generación de alarmas

El sistema de alarmas, según el diseño establecido, debe generar una alarma ante la detección de una intrusión.

Una desventaja de este esquema es que la cantidad de alarmas generadas en el caso un port scanning (identificado como múltiples intentos de conexión desde una IP a distintos puertos de la máquina objetivo) es de una alarma por cada conexión detectada a la *honeynet*.

Para ello se evalúan diferentes implementaciones:

- Implementar un triger en la base de datos, que se ejecute al detectar el registro de una intrusión.
- Implementar una aplicación que permita monitorear la base de datos ante nuevos registros de intrusiones.
- Al recibir la información a centralizar, disponerla en un archivo de *log* además de darla de alta en *MySQL*. Para luego realizar el monitoreo del archivo.

Se utiliza la última opción por dos motivos. Primero, la herramienta utilizada para realizar la centralización (*Prelude Manager*), permite generar un *log* de las capturas volcadas a la base de datos. Segundo, ya se tiene experiencia en procesamiento de archivos de *log*, dado que se realizó un análisis para solucionar la normalización y centralización de datos.

Además el esquema de monitoreo de *log*, permite utilizar el sistema de alarmas en otros escenarios con solo redefinir las reglas a utilizar.

El problema se reduce entonces a:

- Captar una nueva entrada en el archivo de *logs*.
- Verificar si aplica regla sobre la nueva entrada.
- Enviar una alerta de acuerdo a la regla aplicada.

Para implementar el mecanismo de alarmas, se utiliza SEC [Vaar08] en conjunto con un programa escrito en C++ para el envío de correos electrónicos, basado en la biblioteca lsmtp++

[Timo03], el cual envía un correo electrónico por cada instrucción reportada.

Si bien existen alternativas como *Swatch* [Atki95], la cual es una herramienta que cumple con el objetivo sin la necesidad de utilizar ningún componente extra, la inclinación hacia SEC se debe a que permite la correlación de eventos. Esto posibilita establecer reglas mucho más complejas y precisas para reconocer los ataques admitiendo así generar alarmas más específicas.

Por ejemplo es posible correlacionar los ataques desde un mismo objetivo, estableciendo intervalos de tiempo, puerto objetivos, etc. Con ello sería posible enviar una alerta indicando la posibilidad de que se está en presencia de un ataque de port scanning o bien de una denegación de servicio.

De esta manera se logra un sistema sencillo de alarmas, flexible, extensible y reusable. La desventaja de este sistema es que la información obtenida se duplica (en el archivo de *log* y en el DBMS). Si bien esta solución consume espacio de disco extra (que es un recurso de bajo costo) no compromete la performance del sistema.

4.5 Integración de componentes

En la Imagen 4.4 se presenta la integración final de componentes utilizados para la implementación de la solución al proyecto. En ella, el *HoneyCell* cuenta con una implementación de *honeypot* de bajo nivel de interacción (pudiendo ser *Honeyd* o *Honeytrap*), que registra los ataques recibidos en un archivo de *log*. Este archivo es monitoreado por *Prelude-LML* que, al detectar nuevas entradas en el *log*, genera la versión normalizada de los datos (denominada alerta *IDMEF*). Dicha alerta es enviada a través de una conexión SSL/TLS al componente de centralización *Prelude Manager* que reside en *HoneyCenter*. *Prelude Manager* al recibir la información, la almacena en el DBMS y en un archivo de *log*. Por otro lado, SEC monitorea dicho archivo en busca de nuevas entradas para generar la alarma de actividad que se envía a los administradores.

Adicionalmente los accesos al *HoneyCenter* son filtrados y registrados en un archivo de *log*, por un firewall implementado con `iptables`. Se mantiene una instancia de *Prelude-LML* analizando dicho archivo, normalizando y enviando al *Prelude Manager* cada entrada registrada para ser centralizada.

4.6 Despliegue de la honeynet

El diseño de la solución es concebido teniendo en cuenta la escalabilidad de la misma, lo que permite colocar tantos *sensores* como recursos se tengan disponibles, y en forma transparente. Este punto no debe pasar desapercibido a la hora de desplegar una *honeynet* con esta solución.

Específicamente se virtualiza la solución como una *honeynet virtual Auto-Contenida* en un único medio físico.

Las ventajas que se obtienen son:

- **Movilidad**: las *honeynets virtuales* pueden ser instaladas en un PC portátil y llevadas a donde se necesite.
- **Menos hardware:** al tratarse de un solo sistema físico, el hardware requerido es menor

en relación a si utilizara una PC por cada componente. Como contrapartida se debe contar con hardware más potente.

- **Menos espacio**: por el mismo motivo anterior el requerimiento de espacio físico disminuye notablemente.

Como software de virtualización se utiliza *Vmware Server* [Vmwa08], ya que es una herramienta muy difundida y permite tratar la máquina virtual como un archivo. La ventaja que se obtiene es una fácil movilidad de la máquina virtual, ya que se tratan simplemente de archivos. Además permite iniciar varias instancias con la misma configuración, ya que basta con copiar los archivos que la definen.

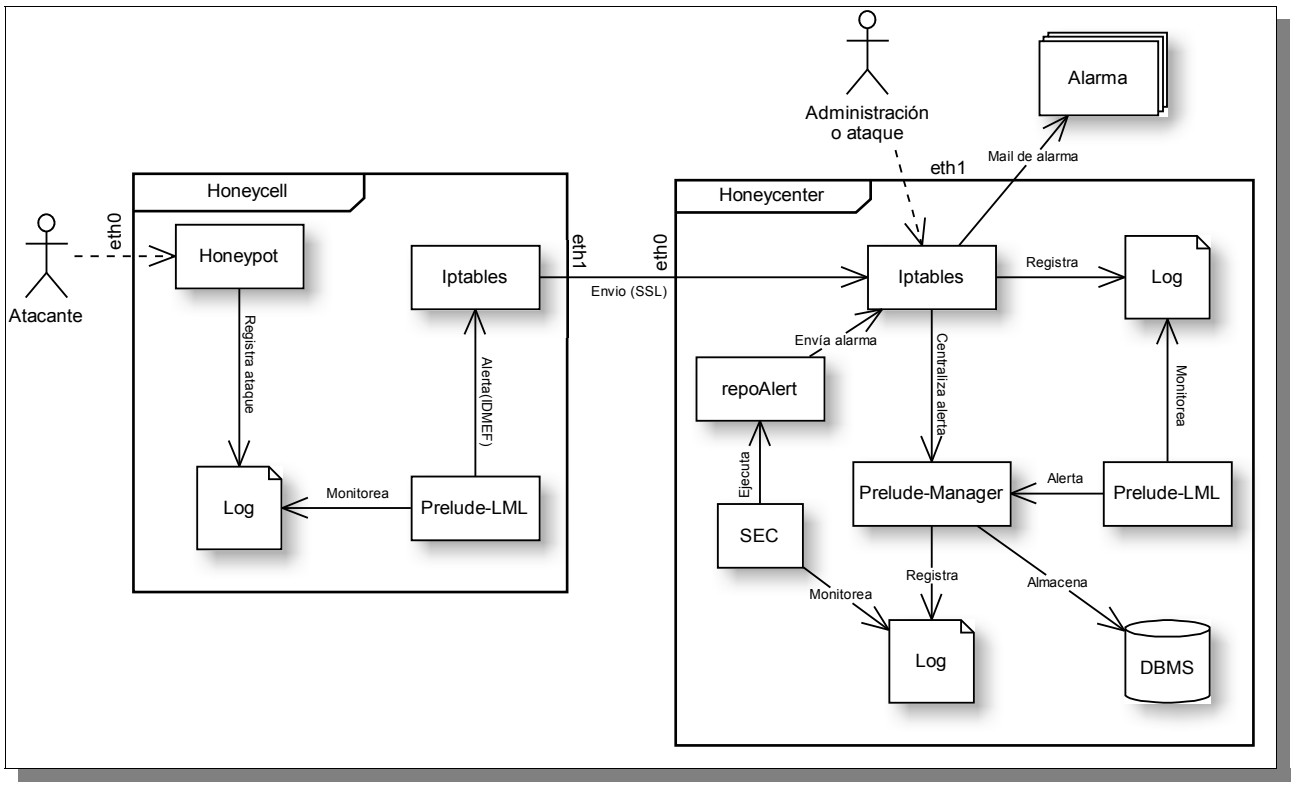


Imagen 4.4: Integración de los componentes seleccionados.

Al utilizar este software se aumenta notablemente el grado de flexibilidad de la arquitectura, ya que se pueden tratar las máquinas virtuales simplemente como archivos de gran tamaño, permitiendo el transporte de los mismos a distintos ambientes (desarrollo, testeo, producción).

VMware Server permite implementar conexiones (denominadas *VMnets*) y redes virtuales, a las que se conectan las máquinas virtuales. Se tienen cuatro tipos de interconexiones virtuales: host-only, bridged, custom y NAT. Haciendo uso de estas herramientas se implanta la *honeynet virtual*.

En la Imagen 4.5 se observa la arquitectura de la solución virtualizada.

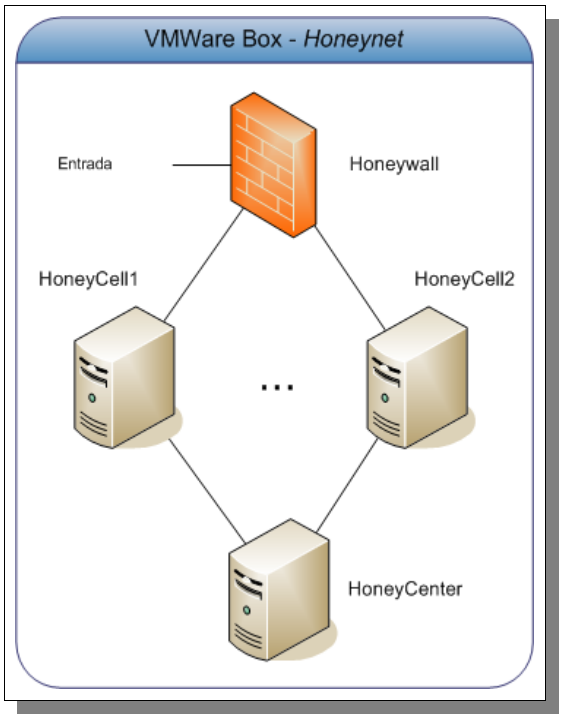


Imagen 4.5: Arquitectura de la solución virtualizada

4.6.1 Despliegue de los componentes de la honeynet

Con el objetivo de brindar portabilidad y uniformizar los despliegues de los componentes de la solución se utiliza el formato *livecd* para generar las distribuciones de los componentes.

Un *livecd* es un sistema operativo y un conjunto de herramientas que se encuentran almacenados y pueden ser ejecutados desde un medio extraíble (como puede ser un CD, un DVD o un pendrive). En este tipo de distribuciones se crea un disco virtual haciendo uso de la memoria RAM del equipo en el cual es ejecutado.

Una de sus ventajas es que en general no se efectúan cambios en la máquina (o disco duro) real utilizada a menos que se deseen guardar las preferencias de ejecución (y la distribución esté creada para hacerlo).

Al utilizar este tipo de tecnologías no es necesario realizar instalaciones en la máquina donde se ejecuta el *livecd*. Se tienen las herramientas instaladas y preconfiguradas, haciendo que la puesta en producción del sistema sea en forma inmediata. Facilitan la portabilidad del sistema ya que reconocen el hardware sobre el que se está ejecutando.

Hay que tener en cuenta que al reiniciar la máquina en la que se encuentra ejecutando, se vuelve a su estado inicial, por lo que si se instalan nuevas herramientas o se cambian configuraciones se pierden los cambios realizados.

Por más información sobre la creación y configuración utilizada para generar las imágenes con la tecnología de *livecd*, puede referirse a la documentación creada por el grupo, "*LivecdTools*" [Live08].

CAPÍTULO 5 IMPLANTACIÓN

En este capítulo se describe la implantación de la solución al presente proyecto de grado, realizada en la Facultad de Ingeniería de la Universidad de la República Oriental del Uruguay [FING08].

Se comienza por el sistema que soporta la infraestructura de virtualización, luego se detalla la configuración de cada uno de los componentes que conforman la arquitectura la solución: H*oneywall*, H*oneyCell* y *HoneyCenter*; especificando el hardware utilizado, configuración de herramientas, etc.

Para finalizar se trata la terminal de administración, explicando como conectarse a la infraestructura para poder trabajar en forma remota con ella.

5.1 Host base

Para desplegar la solución en el ambiente de Facultad, se puso a disposición del grupo un servidor (cuyas características se detallan más adelante, en el cual se implanta la *honeynet* virtual) y tres direcciones IP. Estas direcciones son utilizadas por los componentes que necesitan acceso directo a Internet. Una de las direcciones IP es utilizada para la administración remota del host base y las restantes son utilizadas por los *HoneyCells* (para que los *honeypots* realicen la captura de datos).

El servidor cuenta con dos interfaces de red conectadas a la DMZ de Facultad. Como se dijo anteriormente, a una de las interfaces de red se le asigna una IP para realizar la administración remota de la solución. A la otra interfaz de red no se le asigna IP[18]. Ésta interfaz es utilizada por *VMware Server* para la conexión en modo bridge de sus maquinas virtuales. De esta manera se le asignan las dos IP's restantes a cada una de las instalaciones del *HoneyCell*.

Los administradores de red del InCo (*Instituto de Computación*) [INCO08] permiten el libre tráfico a Internet desde y hacia las dos direcciones usadas por lo *HoneyCells* mientras que restringen el tráfico a la IP de administración donde solo se permite el acceso a través de "*lulu*" a los puertos TCP 23 (SSH), TCP 443 (HTTPS), TCP 904 (*VMware Server Console*), TCP 8080 (*Tomcat*) y TCP 9443 (utilizado para realizar un port forwarding al puerto 443 de la interfaz de administración de *HoneyWall*); por lo que, para poder realizar una administración remota de los sistemas (desde fuera de la DMZ) se implementa un *tunneling* a través de *"lulu"*.

El resultado de la configuración antes mencionada, conceptualmente resulta como lo muestra la Imagen 5.1.; donde el servidor se encuentra conectado a Internet por la interfaz sin IP (eth0) y a la DMZ (controlada por los administradores del InCo) por la otra interfaz (eth1).

18 Carecer de dirección IP reduce el riesgo de accesos no autorizados al servidor (por dicha interfaz), evitandose ataques. Las aplicaciones de capa 3, de este modo, no reciben paquetes porque el sistema no es el destino de ningún mensaje.

Como se dijo anteriormente, la función del servidor es soportar toda la infraestructura virtual sobre la que se monta la red solución. Este cuenta con las siguientes características de hardware:

- PC Pentium IV
- 1 GB de memoria RAM
- 80 GB de disco duro
- 2 tarjetas de red.

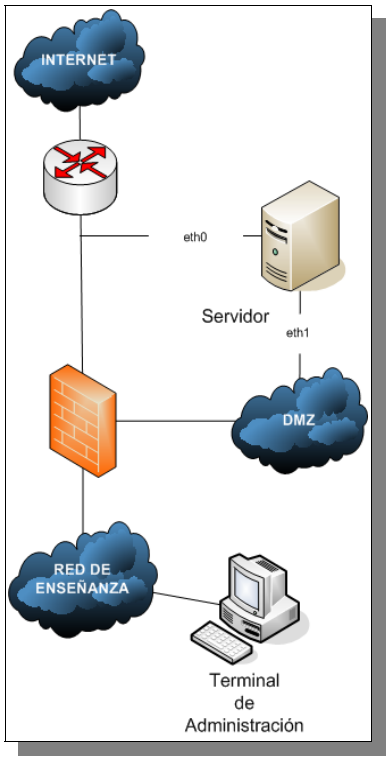


Imagen 5.1: Ubicación conceptual del servidor

Como sistema operativo para el servidor se selecciona la distribución de Linux *CentOS 5* (*Comunity Enterprise Operative System*) [Cent07], una recopilación libre y gratuita de *RedHat Enterprise Linux* (*RHEL*) [RedH08].

Entre las características principales de *CentOS* se destacan,

- Compatibilidad 100% con *RHEL 5*, a nivel de binarios. Toda aplicación compatible con *RHEL* también lo es con *CentOS*.
- Rápido tiempo de respuesta a incidentes de seguridad. Si se detecta alguna vulnerabilidad hay un lapso muy corto hasta que se publica el parche correspondiente, por lo general de 24 a 48 horas luego de reportada la incidencia.
- Los ciclos de vida del producto son de aproximadamente 5 años, lo que ofrece mayor estabilidad.
- Cuenta con el apoyo y soporte de la comunidad de usuarios Linux.

Teniendo presente la funcionalidad del servidor, se incluyen solamente los componentes mínimos necesarios para asegurar un correcto funcionamiento del sistema, sin descuidar los requisitos de administración y de instalación de *VMware Server*. Esto es en pro de tener la cantidad mínima de servicios y aplicaciones corriendo en el servidor, maximizando los recursos del sistema (procesador, memoria, disco, etc.).

Para la implantación de la *honeynet* se crean 3 redes virtuales,

- `VMNet1`: comunica los *HoneyCell* con el *HoneyCenter* para el reporte de los ataques.
- `VMNet3`: subred administrativa por la cual se accede a *Honeywall, HoneyCell y HoneyCenter* desde el host base.
- `VMNet4`: es la subred por donde *Honeywall* redirige los ataques a los *HoneyCells.*

En la Imagen 5.2 se muestra el despliegue de la arquitectura virtual implantada.

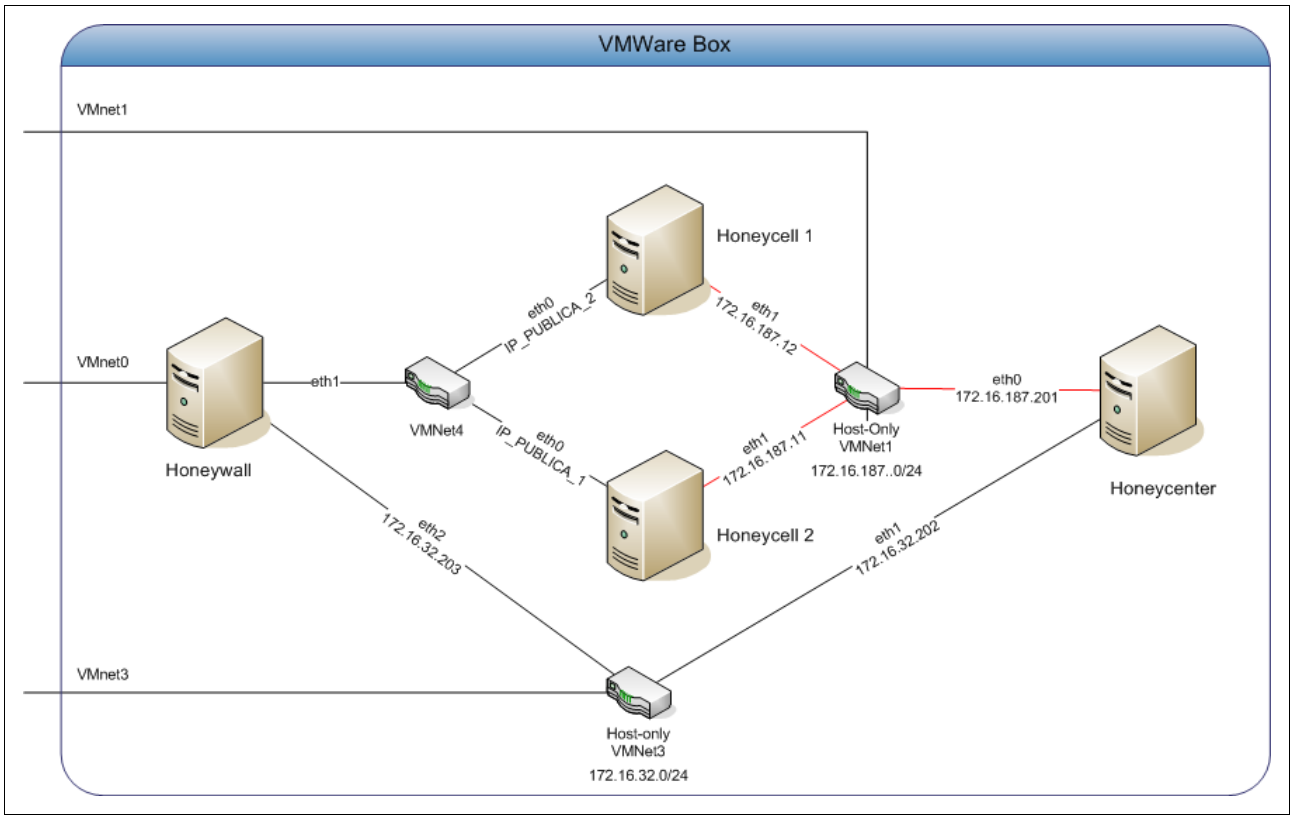


Imagen 5.2: Arquitectura virtual implantada

El acceso a las `VMNets` desde el host base para la administración de los componentes de la solución se realiza a través de la interfaz virtual `vmnet3` con lo que se establece una red punto a punto entre el servidor y la máquina virtual. Esto a la vez puede generar riesgo, ya que de comprometer un *HoneyCell*, el atacante podría dirigir su ataque hacia el host base.

Como forma de elevar el nivel de seguridad del host base, se configuran reglas de `iptables` a todas las interfaces menos la eth0 para:

- Bloquear ataques del tipo `SYN` flood.
- Realizar una validación del origen de los paquetes.
- Ignorar los paquetes de tipo `ICMP` enviados a la dirección de broadcast.
- Redireccionar los paquetes solo si provienen del *gateway* por defecto.
- Registrar todo el tráfico que llegue con direcciones erróneas (martian sources).
- Permitir conexiones `TCP` a los puertos `SSH`, `HTTP`, `HTTPS`, 904, 8333 y 9443.

De manera de posibilitar el acceso en forma remota a los componentes de la solución, se crearon reglas de port forwarding. Mediante estas se redirige el tráfico de los puertos mencionados anteriormente hacia los componentes, permitiendo ejecutar aplicaciones en ellos a través del host base como intermediario. En este caso quien realiza el port forwarding es el host, cuando recibe una petición al puerto expuesto lo redirige a la `IP` establecida en las reglas de `iptables`.

Como forma de comprobar la integridad del sistema se incluye el script `/host/respaldo-md5.sh`, implementado por el grupo, que lleva a cabo la suma de comprobación *md5*[19] [MD508] de los principales archivos del sistema operativo y lo guarda en un archivo (especificado como parámetro al script). Este listado es útil para comprobar periódicamente los archivos críticos y asegurar así que el servidor no ha sido comprometido. Se debe tener la precaución de no dejar el archivo generado por el script en la máquina donde se ejecuta, ya que de ser comprometida, el archivo puede ser modificado por el atacante; por ello el grupo lo mantiene en un medio extraíble. Puede acceder a las sumas *md5* realizados en los sistemas, consultando el material complementario en el DVD que acompaña esta documentación.

El Fragmento 5.1 muestra los primeros cuatro elementos del archivo "*respaldo*", resultado de la ejecución del script antes mencionado.

```
[bash]# sh respaldo-md5.sh respaldo
[bash]# head -n 4 respaldo
15f5473f7b71266dc576e546a9221ea3 /bin/chown
92deae0a5945bfeb6eb7cf7b302b3bb2 /bin/setserial
5e169b70c7a55d9edefe584ebf4e53cc /bin/mount
ad68cb12e23ff3171788328204902e32 /bin/echo
```

Fragmento 5.1: Integridad del sistema, lista de sumas MD5.

En el caso que la suma *md5* para algún binario difiera, implica una modificación de éste y se puede llegar a asumir que alguien logró una penetración al sistema sin que pudiese ser detectado, estudiado o detenido.

Cabe aclarar que existen herramientas para chequear la integridad de los sistemas, como por ejemplo *Tripwire* [Trip08]. Se evaluaron las funcionalidades prestadas por esta en conjunto con los requerimientos de integridad del host base, de los *HoneyCells* y del *HoneyCenter*, teniendo en

19 MD5 es un algoritmo de reducción criptográfico de 128 bits desarrollado por Ronald Rivest. Este algoritmo supone computacionalmente imposible generar dos mensajes distintos (x e y) con el mismo hash (MD5(x) = MD5(y)), resistencia fuerte a colisiones. (En 1996, Hans Dobbertin descubre dos cadenas distintas x e y, tales que generan el mismo hash MD5(x) = MD5(y)).

cuenta además el tiempo necesario para poner en funcionamiento la herramienta en cada sistema. La evaluación mostró la necesidad de instalar varios paquetes en los sistemas y configurar cada instalación para aprovechar las funcionalidades de la herramienta.

Se concluyó que *Tripwire* es una herramienta mucho más potente que la necesitada para el proyecto, la idea es tener un mecanismo rápido de comprobación de la integridad del sistema. Se debe tener en cuenta que, un atacante no debería ser capaz de comprometer el host base directamente (como se vio antes, se tiene una interfaz de red sin IP, los accesos a la otra interfaz de red son controlados por `iptables` y además los accesos desde Internet son controlados por las políticas del firewall del InCo). Adicionalmente, las implementaciones de *honeypots* en los *HoneyCells* son de bajo nivel de interacción (con lo que un atacante no interactúa con el sistema directamente, sino que se emulan los servicios prestados) por lo que tampoco sería posible que éstos sean comprometidos. Por otra parte, para comprometer el *HoneyCenter* es necesario haber comprometido un *HoneyCell* o el host base, lo cual se dijo anteriormente es poco factible. Viendo esto, se opta por desarrollar un script sencillo para realizar las sumas de comprobación de los archivos y realizar los chequeos de integridad manualmente.

Se incorpora al host base un sistema de alarmas (independiente del sistema de alarmas, que se verá más adelante, de *HoneyCenter*) para notificar cuando se intenta acceder al servidor en los puertos TCP destinados para la administración remota. Para ello se crearon reglas en las `iptables` para que se registren las conexiones (esto lo hace en el archivo `/var/log/messages`). Ante cada intento de acceso se registran los datos de IP origen y destino de la conexión, puertos involucrados, banderas del paquete, interfase contactada, entre otra información que puede resultar interesante como por ejemplo el largo del mensaje. En el Fragmento 5.2 se muestra el registro realizado por `iptables` de una conexión al puerto 22.

```
Oct 16 09:14:08 host kernel: ssh - IN=eth1 OUT= MAC=00:0c:29:59:5b:d6:00:50:56:c0:00:03:08:00
SRC=10.10.10.1 DST=10.10.10.128 LEN=48 TOS=0x00 PREC=0x00 TTL=64 ID=20601 DF PROTO=TCP SPT=1234
DPT=22 WINDOW=65535 RES=0x00 SYN URGP=0
```

Fragmento 5.2: Iptables, registro de una conexión al puerto 22.

Se definen seis prefijos para identificar el tipo de conexión realizada:

- `ssh` si la conexión es al puerto TCP 22,
- `http` si es al puerto TCP 8080,
- `http` si se realiza una conexión al puerto TCP 443,
- `vmw` cuando se intente conectar al puerto de administración de *VMware Server Console*,
- `mui` cuando se quiera tener acceso a la administración remota vía Web.
- `DROP` es utilizado cuando se descarta el mensaje.

Para implantar este sistema de alarmas se configura *SEC* [Vaar08] para analizar el contenido del archivo `/var/log/messages` en busca de los prefijos antes mencionados. El Fragmento 5.3 es la regla utilizada para detectar una conexión al servicio SSH como la registrada en el Fragmento 5.2.

```
type=single
desc = Conexion via SSH
ptype=regexp
pattern=([^ ]*) kernel: ssh - IN=(.*) OUT=(.*) SRC=(.*) DST=(.*) LEN=(.*) PROTO=(.*) SPT=(.*)
DPT=(.*) WINDOW=(.*)
action = shellcmd /host/centosAlert -ssh "Inicio de conexion via SSH ($7) desde $4:$8 a $2 en
$5:$9"
```

Fragmento 5.3: Regla de SEC para identificar una conexión via SSH.

Cuando coincide el patrón buscado *SEC* ejecuta la aplicación *centosAlert* (programa realizado por el grupo, escrito en C++ que se apoya en la librería *libsmtp*++ [Timo03]), que se encarga de enviar por correo electrónico el registro de la conexión. El cuerpo del mensaje en este caso es el que se detalla en el Fragmento 5.4.

```
Inicio de conexion via SSH (TCP) desde 10.10.10.1: 1234 a eht1 en 10.10.10.128:22
```

Fragmento 5.4: Cuerpo del mensaje enviado al administrador.

Para que este *sistema de alertas* esté siempre activo se lo agrega como servicio a ejecutarse en los modos 3 y 5 de linux, como se muestra en el Fragmento 5.5.

```
[bash]# chkconfig --add /etc/init.d/init-sec
[bash]# chkconfig --level 35 /etc/init.d/init-sec on
```

Fragmento 5.5: Registro del servicio de alarmas en el host base.

Este sistema de alarmas fue implantado para notificar cuando algún usuario intenta conectarse al host base. Y así llevar un registro de los accesos realizados, en caso de conexiones no autorizados tomar las medidas necesarias para incrementar la seguridad del sistema.

Los eventos reportados por `iptables` y el tráfico registrado en la interfaz sin `IP` (`eth0`) , pueden ser incorporados al repositorio de centralización. Para llevar esto a cabo, es posible utilizar *Tcpdump* [Tcpd07] para registrar todos los paquetes que llegan a la `eth0`. Agregando así una capa adicional de registro de datos. Esto no se llevó a cabo para no descuidar el alcance establecido.

5.2 HoneyCell

Las máquinas virtuales en la que se instalan los *HoneyCell* tienen las siguientes características:

- 2 tarjetas de red
- lector de `CD`
- 128Mb de `RAM`[20]
- Disco virtual de 8Gb[21]

La interfaz `eth1` es utilizada para la administración del componente y la `eth0` es utilizada por los *honeypots*, emulando servicios en la `IP` asociada a dicha interfaz.

20 32 Mb de RAM son suficientes para operar el sistema, pero visto que por razones de disponibilidad de IPs solo se instalan 2 máquinas virtuales, la memoria real restante es utilizada para disminuir los accesos a disco necesarios para la operativa.

21 Dado que los componentes de la red son distribuidos en formato de LiveCD, no es necesario (a menos de querer instalarlo), el uso de disco duro.

En *honeypots* de bajo nivel de interacción un atacante no interactúa con el sistema base, por lo que la selección de sistema operativo no es lo fundamental en este caso, por ello queda a libre elección. Se decide utilizar Linux *Fedora 7* [Fedo07] porque el grupo ha trabajado con esta distribución de Linux con anterioridad.

Es necesario asegurar el componente *HoneyCell*, para ello se configuran `iptables`, para filtrar y registrar toda conexión a la interfaz de administración `eth1`; excepto al puerto `SSH`, y al puerto 4690 `TCP`, utilizado por *Prelude-LML* para el envío de alertas *IDMEF* al *Prelude Manager* (residente en el *HoneyCenter*). Dado que los *honeypots* asumen el control de la interfaz `eth0`, no se configuran las `iptables` para dicha interfaz, permitiendo todo el tráfico entrante y saliente a la misma.

5.2.1 Captura de Datos con Honeytrap

Honeytrap corre directamente sobre el sistema operativo base y se especializa en la captura de malware. Para facilitar su utilización se creó el script `/sensor/honeytrap-simple.sh` que crea, antes de la inicialización del *honeypot,* un archivo de configuración basado en los valores recibidos como parámetros. En el Fragmento 5.6 se muestra la ejecución de *Honeytrap* con dicho script.

```
[bash]# sh honeytrap-simple.sh xxx.xxx.xxx.xxx d

honeytrap v1.0.0 - Initializing.
  Port 21/tcp is configured to be handled in normal mode.
  Port 22/tcp is configured to be handled in normal mode.
  Port 25/tcp is configured to be handled in normal mode.
  Port 80/tcp is configured to be handled in normal mode.
  Port 8080/tcp is configured to be handled in normal mode.
  Port 21/udp is configured to be handled in normal mode.
  Port 22/udp is configured to be handled in normal mode.
  Port 25/udp is configured to be handled in normal mode.
  Port 80/udp is configured to be handled in normal mode.
  Port 8080/udp is configured to be handled in normal mode.
  Servers will run as user nobody (99).
  Servers will run as group nobody (99).
  Loading default responses.
  Connections will be handled in normal mode by default.
  Logging to /sensor/logs/honeytrap.log.
  Initialization complete.

honeytrap v1.0.0 Copyright (C) 2005-2007 Tillmann Werner <tillmann.werner@gmx.de>
[2007-11-18 00:49:57] ---- Trapping attacks on eth0 via PCAP. ----
```

Fragmento 5.6: Ejecución de Honeytrap.

En el script de configuración se definen los puertos 21, 22, 25, 80 y 8080, para que tengan interacción con el atacante, ya que en el directorio `/sensor/honeytrap/responses`, se mantienen respuestas simulando las brindadas por los servicios reales. Esto no implica que otros puertos no sean atendidos, sino que no van a presentar una pantalla de bienvenida como si fuese un servicio real. Luego de esto queda habilitado para comenzar a capturar los ataques.

En base a la configuración realizada, *Honeytrap* una vez en ejecución queda a la espera de una petición de conexión. Cuando ésta se produce si el puerto no se encuentra en uso lo abre y acepta petición procesando la conexión, generando así los *logs.* Permite registrar 2 tipos de ataques:

- conexiones establecidas
- conexiones con envío de datos

La traza del *log* para el caso de una conexión establecida al puerto 139 TCP se puede ver en el Fragmento 5.7. Mientras que el Fragmento 5.8 muestra el caso de una transferencia de datos al puerto 22 TCP.

```
[2008-01-27 13:29:45] 139/tcp              Connection from  194.169.225.85:3862 acepted
```

Fragmento 5.7: Honeytrap. Conexión al puerto TCP 139.

```
[2008-01-27 16:49:23]        *22/tcp       724 bytes attack string from 157.100.50.58:47537
```

Fragmento 5.8: Honeytrap. Transferencia de datos al puerto 22.

Cabe resaltar que ambos casos son conexiones, pero en la primera no se produce ningún tráfico extra, mientras que la segunda indica que se estableció una conexión seguida de una transferencia de datos a través de la misma.

5.2.2 Captura de Datos con Honeyd

Se proveen scripts para que *Honeyd* se pueda configurar de distintas maneras. Para trabajar con un solo *honeypot virtualizado*, una *honeynet virtualizada* o en modo *híbrido* (es cuando se tiene una *honeynet* virtualizada y un *honeypot* no simulado por *Honeyd*, pero que lógicamente se encuentra dentro de la *honeynet*). En el proyecto, dado que está limitada la cantidad de direcciones IP a utilizar, se trabajó virtualizando solo un *honeypot.*

Para establecer la personalidad[22] del sistema operativo a virtualizar, *Honeyd* utiliza el archivo de configuración (`/sensor/honeyd/conf/simple`) que se muestra en el Fragmento 5.9. Bajo dicha configuración se crea un terminal, al que se le asigna la personalidad de un sistema operativo "*Microsoft Windows XP Home Edition*" [WinX08]. El sistema responde como si tuviese cerrado todos los puertos, menos el 80 TCP, donde atiende un script que simula ser un servidor Web.

```
#ejemplo de una personalidad de honeyd
create WinXP
set WinXP personality "Microsoft Windows XP Home Edition"
set WinXP default tcp action reset
set WinXP default udp action reset
set WinXP default icmp action open
add WinXP tcp port 80 "sh scripts/web.sh"
set WinXP uptime 234512
```

Fragmento 5.9: Definición de la personalidad WinXP con un servidor Web en el puerto 80.

Además se provee del script `/sensor/honeyd-simple.sh`, que se encarga de crear el archivo de configuración y además de inicializar *Honeyd* con los parámetros correctos.

En base a la configuración, *Honeyd* permite registrar 3 tipos de conexiones:

- Conexiones establecidas (Fragmento 5.10).
- Conexiones finalizadas (Fragmento 5.11).
- Paquetes que no pertenecen a ninguna conexión (Fragmento 5.12).

22 Se recuerda que la personalidad de un *honeypot* virtualizado con *Honeyd* se refiere al comportamiento del sistema base que simula tener ante ataques de fingerprint por herramientas como Nmap [Nmap08] y Xprobe [Xpro08].

```
2006-08-18-12:21:12.1239 icmp(1) S 11.11.11.11 22.22.22.22: 8(0): 84 [SunOS 4.1 ]
```

Fragmento 5.10: Honeyd. Registro de una conexión establecida.

```
2006-08-18-12:21:12.1239 tcp(6) E 11.11.11.11 53952 22.22.22.22 10078: 44 S [Linux 2.6 ]
```

Fragmento 5.11: Honeyd. Registro de una conexión finalizada.

```
2006-03-22-13:13:27.4148 tcp(6) - 128.39.44.11 54265 128.39.44.40 80: 40 R [Linux 2.6 ]
```

Fragmento 5.12: Honeyd. Registro de un paquete sin conexión establecida.

5.2.2.1 Otras configuraciones

Cabe resaltar que también se proveen scripts para inicializar *Honeyd* para trabajar como una *honeynet* virtualizada, y también integrando un *honeypot* híbrido. Los mismos no fueron utilizados en el proyecto dado que no se contaba con la cantidad de direcciones `IP` necesarias (al menos 3 para asignarle a los *honeypots*).

La *honeynet* virtualizada consta de un router y cuatro terminales de distinto fingerprint (personalidades) conectados a éste. Para montar la red, se ejecuta el script `/sensor/honeyd-network.sh`, con esto se crea dinámicamente el archivo de configuración `/sensor/honeyd/conf/network`, que es leido por *Honeyd*. Este script requiere que se le asignen las `IP's` de cada uno de los *honeypots* virtualizados y del router (punto de acceso a la red virtualizada).

Para definir el sensor híbrido, se utiliza una configuración adicional, que permite integrar a la topología de red virtualizada terminales "*reales*" (o terminales no creadas por *Honeyd*). De este modo se indica que el terminal es alcanzado a través de una de las interfaces de red de la maquina anfitriona. Como a nivel lógico dicha `IP` se encuentra dentro de la subred virtualizada, acceder a esta `IP` atravesaría toda la red hasta llegar al equipo requerido.

Para montar esta *honeynet* se utiliza el script `/sensor/honeyd-hibirido.sh`, ésta topología es igual a la provista por el script `/sensor/honeyd-network.sh`, salvo que permite agregar un terminal externo a la virtualización por la interface `eth0` (de la maquina virtual de *VMware Server*).

5.2.3 Proceso de Normalización

Los datos obtenidos por los *honeypots* se normalizan al formato establecido por el modelo *IDMEF*. Para realizar la normalización *Prelude-LML* utiliza el archivo `/usr/local/etc/prelude-lml/ruleset/` en el cual se especifican las reglas a aplicar a las entradas de los archivos de *log*. En el Fragmento 6.13, se presenta la normalización realizada con *Prelude-LML* tomando como base el registro de un ataque a *Honeytrap*.

Luego de la expresión regular utilizada para realizar el "*pattern matching*" se encuenta un bloque (pintado de gris en el Fragmento 5.13) de líneas que inicia en `classification.text` y termina en `last`. Cada línea del bloque es la representación de las clases del modelo *IDMEF*. Donde, el separador punto "." se utiliza para acceder a una atributo de la misma y el número entre paréntesis luego del nombre de la clase, establece que se trata de una colección (y se está accediendo al elemento indizado con ese número). Por ejemplo `analizer(0).name` es el nombre de la instancia cero de la clase *Analizer*.

En la clase *Analizer* se cargan los atributos *name* y *manufacturer* con el nombre de la herramienta (en este caso *Honeytrap* y *Honeyd*) y la url del desarrollador. En el atributo *class* se carga la constante `honeypot` (ya que todos los analizadores son de este tipo).

En la clase *Target* se cargan los valores referentes al objetivo de los ataques. En el atributo *address* de la clase *Node.Address* se guarda la dirección IP, y en el atributo *port* de la subclase *Service* se almacena el puerto atacado. Lo mismo se realiza con la clase *Source*, salvo que la IP y puerto registrados son del atacante.

Un campo interesante a ser llenado es *Additional_data.* Actualmente está configurado para almacenar la entrada del archivo de *log* monitoreado por *Prelude-LML,* que coincidió con el patrón buscado. En este campo se pueden llegar a almacenar otro tipo de datos, como por ejemplo archivos descargados al *honeypot*[23].

De este modo *Prelude-LML* establece los valores para las clases mencionadas. Y haciendo uso de *PreludeLibrary* realiza el envío de los datos normalizados al *Prelude Manager* residente en el *HoneyCenter* de manera segura a través de una conexión SSL.

```
#LOG:[2007-05-26 16:49:23]  * 22/udp       724 bytes attack string from 157.100.50.58:47537.
 regex=\* (\d+)/udp\s+(\d+) bytes attack string from ([\d\.]+):(\d+).; \
 classification.text=Transferencia de datos al puerto $1 ; \
 id=40002; \
 revision=1; \
 analyzer(0).name=honeytrap; \
 analyzer(0).manufacturer=http://honeytrap.mwcollect.org; \
 analyzer(0).class=Honeypot; \
 source(0).node.address(0).category=ipv4-addr; \
 source(0).node.address(0).address=$4; \
 source(0).service.protocol=UDP; \
 source(0).service.port=$5; \
 target(0).node.address(0).category=ipv4-addr; \
 target(0).node.address(0).address=164.73.36.70; \
 target(0).service.port=$1; \
 assessment.impact.completion=succeeded; \
 assessment.impact.type=recon; \
 assessment.impact.severity=medium; \
 assessment.impact.description=Transferencia de datos.; \
 additional_data(0).type=integer; \
 additional_data(0).meaning=Size; \
 additional_data(0).data=$2; \
 last
```

Fragmento 5.13: Definición de la normalización con Prelude-LML.

5.3 HoneyCenter

Para su instalación se creo una máquina virtual con las siguientes características:

- 448 Mb. de RAM
- Disco virtual de hasta 10 Gb.
- 2 tarjetas de red

23 *Prelude-LML* no permite este tipo de acciones, pero es posible crear un sensor que reporte la alerta con dicha capacidad.

La interfaz eth0 se utiliza para comunicarse con los *HoneyCells*, permitiendo el registro y el envío de los datos capturados por estos. La interfaz eth1 es utilizada para conectarse para administración remota del equipo.

Como sistema operativo base para este componente se utiliza Linux *Fedora* 7 [Fedo07], creando una distribución minimal que contiene solo las herramientas necesarias para realizar la centralización, administración y mantenimiento del sistema. De igual forma que los *HoneyCells*, el sistema operativo seleccionado no es determinante en la solución y se elige en base al conocimiento de la distribución.

En relación al mecanismo de seguridad, se habilitan iptables configurando el filtrado para todo tipo de acceso salvo los siguientes casos:

- Registro de "*sensores*". *Prelude Manager* necesita autenticar a todos los dispositivos capaces de transmitirle datos. Por lo que para registrar los *Prelude-LML* (residentes en los *HoneyCells*) se habilita el puerto TCP 5554.
- Para la recepción de datos del pool de sensores *Prelude-LML*, el puerto TCP 4690.
- El puerto de administración remota SSH.
- El puerto para monitoreo HTTP.

Añadiendo un nivel adicional de seguridad, se realiza un backup (en un medio extraíble) de las sumas md5 de los principales archivos y programas, para un posterior chequeo de diferencias.

El motor encargado de la recepción de alertas *IDMEF* y su centralización es *Prelude Manager*. Por tal motivo se instalan los componentes *PreludeLibrary, PreludeDB Library y Prelude Manager* del framework *Prelude-IDS* [Prel04]. La creación del esquema del modelo *IDMEF* implementado por *Prelude* se realiza mediante un script que proporciona *PreludeDB Library* para el DBMS seleccionado (se deben crear usuarios con permisos de acceso al esquema). Para este proyecto, como se indicara con anterioridad, el DBMS utilizado es *MySQL* [MySq95]. Por más información sobre el *Prelude IDS framework*, se puede consultar el documento con ese mismo nombre [Prel08] realizado por el grupo.

En forma adicional se instala el componente *Prelude-LML*. Esto se realiza para capturar los intentos de accesos no autorizados al *HoneyCenter,* centralizándolos como se hace con la información obtenida de los *honeypots*, para este fin se registran los *logs* de iptables. Si bien se basa en que *HoneyCenter* se encuentra asegurado, se opta por agregar esta capa de captura de datos para detectar cualquier conexión no debida.

En lo que refiere el sistema de *monitoreo y análisis* se instala *OpenReports* [OpRe08]. Para su funcionamiento se debe instalar la *JDK(Java Development Kit),* un servidor Web con soporte para servlets como *Tomcat* [Tomc07] y un DBMS. *OpenReport* proporciona los scripts de creación del esquema de la base de datos para el DBMS seleccionado[24] (para almacenar los reportes, usuarios, grupos de usuarios, etc).

24 Como se vio anteriormente se dispone de *MySQL* para almacenar la información de *Prelude Manager*, por lo tanto se hace uso de dicho DBMS.

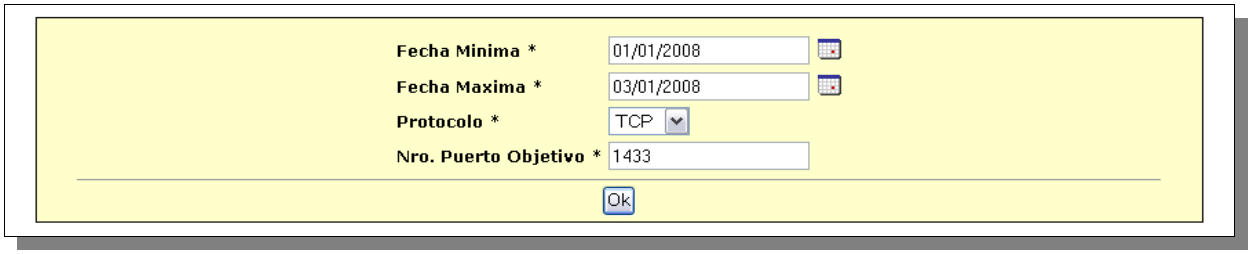
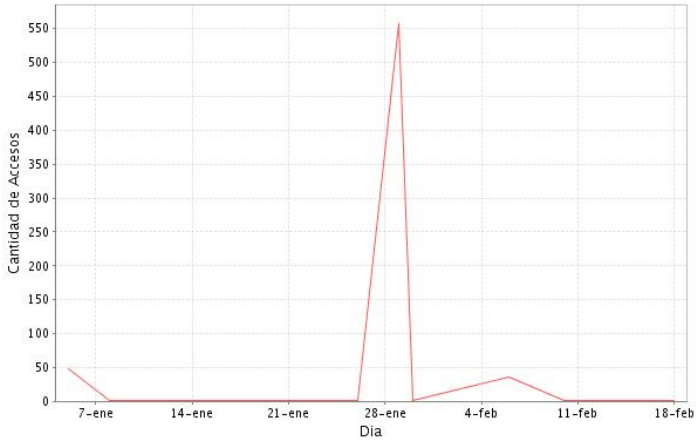


Imagen 5.3: Parámetros de reporte "Gráfica – Evolución de Ataques a un Puerto"

Una vez en funcionamiento *OpenReports*, se crean reportes para visualizar y analizar la información obtenida por los *honeypots*. Como ejemplo se toma el reporte "*Grafica – Evolución Ataques a un Puerto*" que permite analizar los ataques realizados a un puerto dado en un determinado periodo de tiempo. Los parámetros del reporte se indican, como lo muestra la Imagen 6.3, por filtros dinámicos en el momento de la ejecución. La Imagen 5.4 muestra el resultado del reporte con los parámetros de la Imagen 5.3.



Dia	Cantidad de Accesos
01/05/2008 12:00 AM	49,00
01/08/2008 12:00 AM	1,00
01/09/2008 12:00 AM	1,00
01/10/2008 12:00 AM	1,00
01/11/2008 12:00 AM	1,00
01/16/2008 12:00 AM	1,00
01/26/2008 12:00 AM	1,00
01/29/2008 12:00 AM	558,00
01/30/2008 12:00 AM	2,00
02/06/2008 12:00 AM	37,00
02/10/2008 12:00 AM	1,00
02/11/2008 12:00 AM	2,00
02/16/2008 12:00 AM	1,00

Imagen 5.4: Gráfica de evolución de ataques al puerto 1433.

Los reportes implementados fueron organizados de acuerdo al enfoque de la información que brindan siendo estos el detalle de ataques, frecuencias de ataques, objetivos de ataques, origen de ataques y por *honeypot* atacado.

El *HoneyCenter* cuenta con dos sistemas de alarmas, uno para notificar de nuevos ataques a la *honeynet* y el otro para notificar de ataques dirigidos hacia el. Para implantarlos se utiliza *SEC* (como se vió en el sistema de alarmas diseñado en el host base). El Fragmento 5.14 muestra la configuración utilizada en el caso de ataques a la *honeynet* (que se encuentra en el archivo `/repositorio/alertas.conf`). Cuando un *honeypot* detecta un ataque envía una alerta *IDMEF* al servidor; de ese lado *Prelude-Manager* la recibe, la integra a su base de datos y escribe en el archivo `/var/log/prelude-manager.log` una copia de la alerta recibida. *SEC* detecta la nueva entrada en el archivo de *log*, la procesa y al identificar la expresión regular (`regexp`), ejecuta el programa de envío de mails con los datos de la intrusión.

```
# Configuracion de sec para monitorear el log de prelude-manager enviando las alarmas
type=Single
ptype=RegExp
pattern=Original Log: (\S+)
desc=Actividad en el honeypot
action = shellcmd /repositorio/repoAlert -ataque "$0"
```

Fragmento 5.14: Configuración utilizada por SEC para el envío de alarmas.

Las alarmas locales son generadas cuando algún usuario intenta conectarse a un puerto de administración, los cuales son registrados por las `iptables` en `/var/log/messages`. *SEC* nuevamente detecta la nueva entrada, la procesa, y al identificarla mediante la expresión regular (`regexp`), ejecuta el programa de envío de correos electrónicos con los datos de la conexión. El Fragmento 5.15 muestra una sección del archivo de configuración utilizado en este caso (`/repositorio/repo-sec.conf`).

```
type=single
desc = Conexion via SSH
ptype=regexp
pattern=([^ ]*) kernel: ssh - IN=(.*) OUT=(.*) SRC=(.*) DST=(.*) LEN=(.*) PROTO=(.*) SPT=(.*)
DPT=(.*) WINDOW=(.*)
action = shellcmd /repositorio/repoAlert -ssh "Inicio de conexion via SSH ($7) desde $4:$8 a $2 en
$5:$9"
```

Fragmento 5.15: Regla de conexión vía `SSH` para la alarma local del repositorio.

Se mantiene una instancia de *SEC* para cada tipo de sistema de alarmas, iniciándose ambas mediante un único script en `/etc/init.d/init-sec`. Se optó por esta solución, separando el sistema de alarma del repositorio del de los ataques capturados, para no agregarle complejidad a la configuración de *SEC*; lo que podría llevar a demoras en el procesamiento de los archivos.

El programa para el envío de los datos vía correo electrónico, se basa en la librería *libSMTP* [Timo03] y fue implementado por el grupo[25]. En él se incluye información para el armado de los mensajes, así como los datos de la cuenta mediante la que se envían correos electrónicos. Cabe destacar que como destinatario se colocó un "*alias*" proporcionado por los docentes del proyecto. Luego en este alias se especifican los destinatarios definitivos.

25 La poca experiencia con los sistemas Linux fue el motivo por el que no se usara el comando "*mail*" en vez de crear un programa para el envío.

5.4 Honeywall

En la solución se utiliza el *Honeywall* [HPHW07], un conjunto de herramientas open source, configuradas por el "*Honeynet Project*" [HoPr02]. Permite implantar una pasarela o *gateway* de generación II para una *honeynet*, de manera sencilla y segura. La versión actual (conocida como roo-1.2) está disponible para la descarga desde su sitio Web [HPHW07]. Por más información sobre esta herramienta puede dirigirse al documento "*Honeywall*" [Howl08], realizado por el grupo, donde se encontrará información más detallada de su instalación y configuración.

Para la instalación de este componente se crea una máquina virtual con las siguientes características:

- 128Mb de RAM
- disco duro IDE de hasta 10 Gb.
- 3 interfaces de red

Dos de sus tarjetas de red hacen de puente de capa 2 (`eth0` y `eth1`) y la tercera es la interfase de administración (`eth2`) del *Honeywall*.

Luego de realizada la instalación [Howl08], se configura de acuerdo a las características de la *honeynet.* En el Fragmento 5.16 se presentan algunos parámetros de configuración.

```
# This Honeywall's public IP address(es)
# [Valid argument: IP address | space delimited IP addresses]
HwHPOT_PUBLIC_IP=XXX.XXX.XXX.XXX# This Honeywall's public IP address(es)

# [Valid argument: IP address | space delimited IP addresses]
HwHPOT_PUBLIC_IP=XXX.XXX.XXX.XXX,  XXX.XXX.XXX.XXX
# The unit of measure for setting oubtbound connection limits
# [Valid argument: second, minute, hour, day, week, month, year]
HwSCALE=hour

# The number of TCP connections per unit of measure (HwScale)
# [Valid argument: integer]
HwTCPRATE=20

# The number of UDP connections per unit of measure (HwSCALE)
# [Valid argument: integer]
HwUDPRATE=20

# The number of ICMP connections per unit of measure (HwSCALE)
# [Valid argument: integer]
HwICMPRATE=50

# The number of other IP connections per unit of measure (HwSCALE)
# [Valid argument: integer]
HwOTHERRATE=10
```

Fragmento 5.16: Parte de un archivo de configuración de Honeywall..

Se realizaron pruebas para modificar la herramienta de forma que los datos obtenidos fuesen normalizados y centralizados, al igual que el resto de los componentes de la solución (como se dijo en el capítulo 4), pero éstas no fueron exitosas. Por lo que se decidió recortar el alcance y no llevar a cabo la modificación. Dentro de las opciones que el grupo planteó para llevar a dicha tarea se encuentra:

- Instalar *Prelude-LML* y realizar el volcado de datos como en los otros componente de la

red. Esta tarea no se pudo completar debido a que es necesario instalar y actualizar los paquetes del sistema, y no se logró llevar a cabo. La paquetes requeridos tienen fallas de dependencias, y no es posible utilizar `yum` para actualizarlos, porque no se logró configurar correctamente esa herramienta en el *Honeywall*.

- Dado que la información se encuentra en una base de datos, se discutió la posibilidad de realizar un volcado periódico de dicha base y cargarlo en el repositorio, haciendo uso de *Prelude-LML* o con algún script (esta solución no se implantó por falta de tiempo).

Honeywall permite el ingreso de todo el tráfico a la *honeynet* por lo que, para intentar controlar ataques de denegación de servicio contra algún objetivo de la red interna, se hace necesario modificar las `iptables` para establecer restricciones en base a unidades de tiempo y carga. Para realizarlo se establece la regla que se ve en el Fragmento 5.17, que limita la cantidad de conexiones entrantes de un mismo host.

```
#Se limitan las conexiones para evitar DOS
iptables -I INPUT -p tcp --dport 80 -m connlimit --connlimit-above CANT_CON -j REJECT --reject
-with tcp-reset
```

Fragmento 5.17: Regla de iptables para limitar las conexiones entrantes desde un host

El problema radica en la selección del valor `CANT_CON`, ya que un valor muy pequeño no proporcionaría información relevante y un valor muy grande puede llegar a hacer caer los servicios atacados. Con esta regla se está controlando una cantidad de conexiones desde un mismo host, por lo que no soluciona ataques DDoS contra los componentes de la *honeynet*. Si bien se considera en el diseño de la solución, este tipo de casos no se incluye dentro del alcance ya que es un tema muy complejo de solucionar, por lo que puede ser encarado como un trabajo a futuro.

5.5 Terminal de Administración

El sistema operativo que se utiliza para este componente es *Fedora* 7 [Fedo07] con el entorno de escritorio *Gnome* [Tgno99], siendo la elección del sistema operativo no determinante en la solución. Como se dijo anteriormente, se elige en base al conocimiento de la distribución. Se instalan las herramientas necesarias para la administración del sistema implantado:

- *VMware Server Console* para el monitoreo y administración de las máquinas virtuales.
- El navegador *Firefox* [Mozi02] para la administración vía Web de: *OpenReports* y *Wallaye*.
- El paquete `ssh-client` para la comunicación con los servidores a través del comando `SSH`.

Para garantizar la seguridad de este sistema y dado su uso, se tienen configuradas las `iptables` para permitir solamente el tráfico saliente por los puertos de administración y ningún tráfico entrante más que el de respuesta a una conexión iniciada en la terminal.

Es necesario configurar las interfaces de red porque por defecto se inician desactivadas, para ello se provee el script `/root/network.sh` que realiza esta acción.

5.6 Conclusión

Durante la etapa de implantación se encontraron dificultades al momento de configurar herramientas utilizadas, como por ejemplo *VMware Server*. Esta habilita por defecto un servicio *DHCP* para sus redes virtuales, el desconocimiento de este hecho causó problemas en la red de Facultad.

En lo que a virtualización con *VMware* refiere, la posibilidad de pausar las máquinas virtuales es una muy buena característica para detener y analizar tranquilamente un ataque.

Sin embargo, una vez solucionados los inconvenientes anteriores, la solución implantada dispone de un fácil despliegue, cumpliendo con los objetivos planteados. La distribución de los componentes en formato de *livecds* es un buen método para lograr este objetivo.

El monitoreo en formato full Web, y la posibilidad de administrar todo el sistema en forma remota mediante "*ssh tunneling*" es una característica importante de la solución desarrollada.

CAPÍTULO 6 RESULTADOS OBTENIDOS

En este capítulo se presenta el análisis realizado en base a los datos recolectados por la *honeynet*. El análisis abarca el período comprendido desde el 5 de Diciembre de 2007 al 13 de Febrero de 2008.

6.1 Capa de datos de HoneyCell

En esta sección se analizan los datos de los ataques capturados por los *honeypots* instalados: *Honeytrap* y *Honeyd*. Para ello se utilizan las siguientes métricas:

- Frecuencia de ataques
  - Cantidad por mes y día, para poder identificar picos de actividad.
  - Cantidad según la hora y día de la semana, para detectar tendencias de los ataques y poder enfocar los esfuerzos de monitoreo.
- Objetivos de los ataques, para conocer los servicios más atacados y sobre que implementación de *honeypot* fue realizada.
- Origen de los ataques, de modo de tener conocimiento sobre el país de origen de los mismos.

Por tratarse de una primera puesta en producción de la solución, todo tipo de ataque resulta de interés; de esta forma de tener una lectura general de la actividad de los atacantes hacia la red de la Facultad de Ingeniería.

6.1.1 Ataques recibidos

Durante el período analizado, *Honeytrap* estuvo activo por los primeros 62 días y luego *Honeyd* por los 7 restantes. El motivo por el cual no fue posible tener funcionando en paralelo ambos *honeypots* fue debido a una limitación con las `IP's` disponibles.

A lo largo de los 69 días del período de análisis se obtuvo una cantidad constante de ataques por día, con leves incrementos a excepción de 2 grandes picos correspondientes a los días 01/01/2008 con 328 y 29/01/2008 con 564 ataques detectados respectivamente. Un tercer pico menos significativo en cantidad de ataques se registra el día 05/01/2008 con 52 ataques. En la Imagen 6.1 se puede visualizar la gráfica que representa la distribución en que se produjeron los ataques en el período establecido.

Se toman los ataques ocurridos los días picos como puntos interesantes a analizar, ya que en ellos se concentraron la mayoría de los ataques a la *honeynet*.

En la Tabla 6.1 se observa que los ataques realizados el día 01/01/2008 no se concentraron en un puerto en particular, sino que se atacó a un rango equivalente en número a la cantidad de ataques desde una misma IP. Esto hace pensar en forma inmediata de que se trato de un port scanning a un rango de puertos.

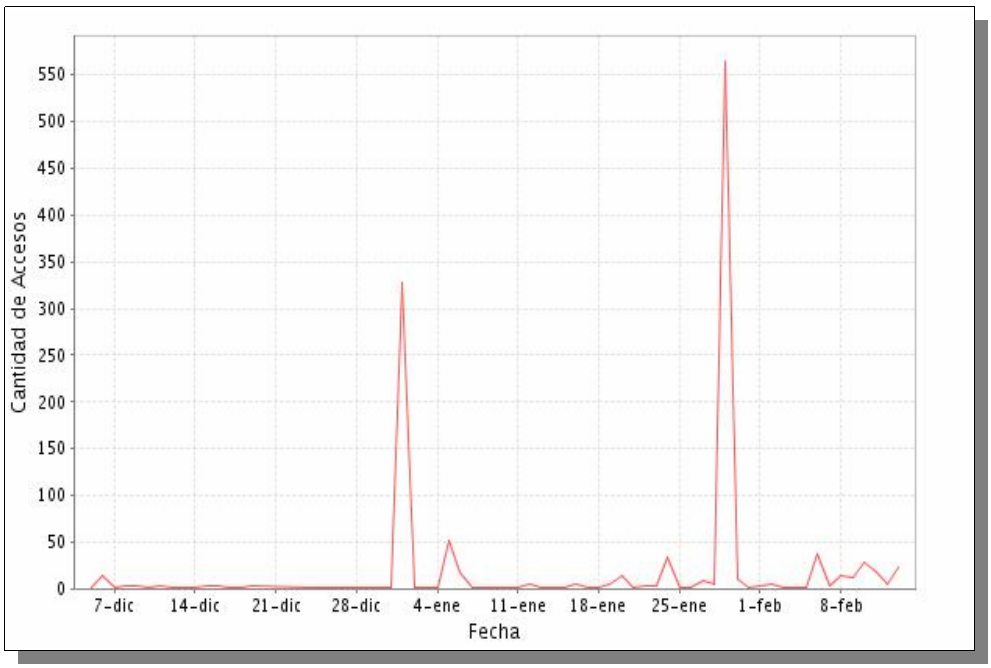


Imagen 6.1: Ataques distribuidos en el período de análisis.

Día	Cantidad ataques	IP distintas	Cantidad puertos
29/01/2008	564	4	3
01/01/2008	328	1	328
05/01/2008	51	3	3
06/02/2008	38	2	2
24/01/2008	34	3	3
10/02/2008	29	11	12
13/02/2008	24	6	18
11/02/2008	20	5	7
06/01/2008	18	3	3
06/12/2007	15	10	8

Tabla 6.1: Lista de los 10 días con más ataques capturados

En cambio los ataques registrados el 29/01/2008 de un total de 564 , 558 tienen como objetivo el puerto TCP 1433 donde atiende el servicio del DBMS *Microsoft SQL Server* y 5 de ellos tienen como objetivo el puerto TCP 139 referente al servicio NetBIOS. También se observa un ataque al puerto TCP 32000 asociado al servicio del troyano *BDDT* [BDDT07], el cual permite establecer una conexión remota. En este caso queda en evidencia un ataque directo a los puertos mencionados en la Tabla 6.2.

Protocolo	Puerto objetivo	Cantidad de accesos	Cantidad de IP's
TCP	1433	558	2
TCP	139	5	1
TCP	32000	1	1

Tabla 6.2: Distribución puertos objetivos de ataques correspondientes al 29/01/2008

Observando la Tabla 6.3 se puede ver que los ataques al puerto TCP 1433 se realizaron 557 con una misma IP y 1 vez con otra distinta.

IP origen	Cantidad de accesos
164.100.24.207	557
196.35.44.65	1
74.132.207.62	1
88.231.126.100	5

Tabla 6.3: Distribución de ataques por IP's de origen correspondientes en la fecha 29/01/2008

Para los ataques registrados el 05/01/2008, como se puede observar en la Tabla 6.3 de un total de 51 ataques, 49 fueron realizados por la misma dirección IP y al puerto TCP 1433, 1 ataque al puerto TCP 2100 el cual se asocia comúnmente a la herramienta *phpSymon* [FlaR95], y 1 ataque al puerto TCP 5900 el cual corresponde al servicio de la aplicación *VNC* [ReaV02].

Protocolo	Puerto objetivo	Cantidad de accesos	Cantidad de IP's
TCP	1433	49	1
TCP	2100	1	1
TCP	5900	1	1

Tabla 6.4: Distribución de puertos objetivos de ataques correspondientes al 05/01/2008

En las Tablas 6.5, 6.6 y 6.7 se puede observar la procedencia, la IP asociada y la cantidad de accesos efectuados en los días pico.

País Origen	IP origen	Cantidad de accesos
India	164.100.24.207	557
Turkey	88.231.126.100	5
United States	74.132.207.62	1
South Africa	196.35.44.65	1

Tabla 6.5: Origen y cantidad de acceso de ataques del día 29/01/2008

País Origen	IP origen	Cantidad de accesos
Portugal	213.13.113.105	328

Tabla 6.6: Origen y cantidad de acceso de ataques del día 01/01/2008

País Origen	IP origen	Cantidad de accesos
Portugal	213.58.174.38	49
Hong Kong	210.3.9.231	1
Netherlands	82.72.185.14	1

Tabla 6.7: Origen y cantidad de acceso de ataques del día

Al investigar en la Web sobre cada IP, se pudo obtener la ubicación de la IP, el ISP que la proporciono, así como su dominio. En la Tabla 6.8 se muestra la información obtenida.

IP origen	Información encontrada
164.100.24.207	**Ubicación**: DELHI, INDIA **ISP**: NICNET INDIA **Dominio**: ALPHA.NIC.IN **Info extra**: Es un equipo que contiene un servidor IIS con una aplicación para la publicación de documentos online: List Of Bussiness Input Module.
88.231.126.100	**Ubicación**:- **ISP**: TT ADSL-NEC DYNAMIC_ULUS **Dominio**: - **Info extra**: -
74.132.207.62	**Ubicación**:INDIANA, UNITED STATES **ISP**: INSIGHT COMMUNICATIONS COMPANY L.P **Dominio**: INSIGHTBB.COM **Info extra**: -
196.35.44.65	**Ubicación**: GAUTENG,SOUTH AFRICA **ISP**: AFRINIC **Dominio**: PODFARM.CO.ZA **Info extra**: Servidor Web: Wesgro is the official Trade and Investment Promotion Agency for the Western Cape
213.13.113.105	**Ubicación**: PORTO, PORTUGAL **ISP**: TELEPAC - COMUNICACOES INTERACTIVAS SA **Dominio**: - **Info extra**: -
213.58.174.38	**Ubicación**: AZORES, PORTUGAL **ISP**: VIA OCEANICA MARKETING E PUBLICIDADE LDA **Dominio**: - **Info extra**: Se encuentra registrado como servidor pirata del juego Counter-Strike y otros
210.3.9.231	**Ubicación**: HONG KONG (SAR) **ISP**: HUTCHISON GLOBAL COMMUNICATIONS **Dominio**: HUTCHCITY.COM **Info extra**: -
82.72.185.14	**Ubicación**: AMSTERDAM, NETHERLANDS **ISP**: @HOME ALMELO HEADEND BLOCK **Dominio**: HOME.NL **Info extra**: -

Tabla 6.8: Origen de los ataques según al 29/01/2008

Un dato interesante es que solo la dirección IP 213.13.113.105 registro ataques en otra oportunidad, a diferencia del resto que solo lo realizaron en la fechas listadas en las tablas anteriores.

6.1.2 Objetivos de los ataques

De los datos obtenidos en el período de recolección se desprende que los tres puertos más atacados son: el TCP 1433 recibiendo 658 intrusiones, seguido por el TCP 139 con 58 ataques y por último el TCP 25 correspondiente al servicio SMTP con 26 ataques como se muestra en la Imagen 6.2.

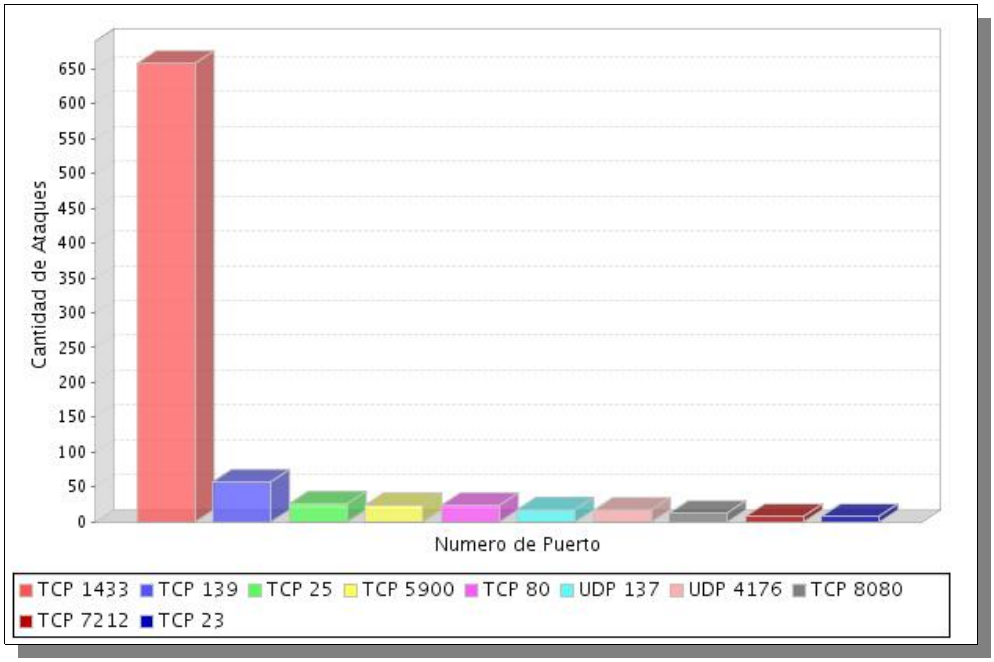


Imagen 6.2: Gráfica de los 10 puertos más atacados.

Protocolo	Puerto	Servicio	Cant. Accesos
TCP	1433	MSSQL Server	658
TCP	139	Net BIOS	58
TCP	25	SMTP	26
TCP	5900	VNC	24
TCP	80	HTTP	24
UDP	137	NetBios	18
UDP	4176		16
TCP	8080	Serv. de Aplic.	12
TCP	7212		7
TCP	23	TELNET	7

Tabla 6.9: Lista de los 10 puertos mas atacados

La frecuencia de ataques al puerto TCP 1433 puede ser causado por un gusano como por ejemplo *W32/Cblade.Worm* [W32C07] o *SQLSpida* [Digi07] que se propagan rápidamente a través de servidores comprometidos. Estos ataques son automatizados y se caracterizan por repetidas solicitudes al puerto del servicio.

Si bien todo parece indicar que se trata del mencionado gusano, no se cuenta con suficientes elementos para asegurarlo. Podría tratarse de un simple intruso intentando utilizar algún exploit, ya que la captura no brinda otro dato significativo.

A modo ilustrativo en la gráficas de las Imágenes 6.3, 6.4 y 6.5 se muestra como se distribuyeron los ataques en el período analizado a los puertos discutidos anteriormente 1433 TCP, 139 TCP y 25 TCP.

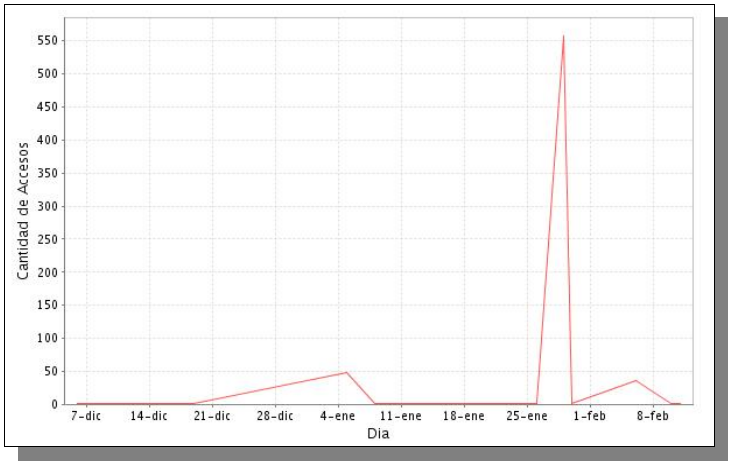


Imagen 6.3: Gráfica de evolución de ataques al puerto 1433

Imagen 6.4: Gráfica de evolución de ataques al puerto 139

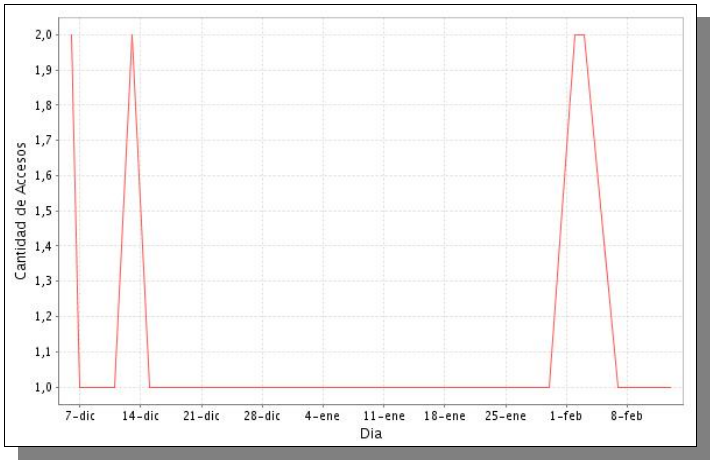


Imagen 6.5: Gráfica de evolución de ataques al puerto 25.

6.1.3 Origen de los ataques

Los ataques registrados se originan en varias partes del mundo. En la Tabla 6.10 se puede observar la cantidad de ataques por país, además de la cantidad de IP's distintas desde donde se se efectuaron.

El país con más ataques es India, seguido por Portugal y Estados Unidos. En general se observa que la mayoría de los ataques provienen de países asiáticos, correspondiéndose con las tendencias generales de incidencias [HPRA05], [UNAM06], [HoPr02], [HPRA05].

En la Imagen 6.6 se presentan, en color rojo, los países desde los que se registraron más de 4 ataques.

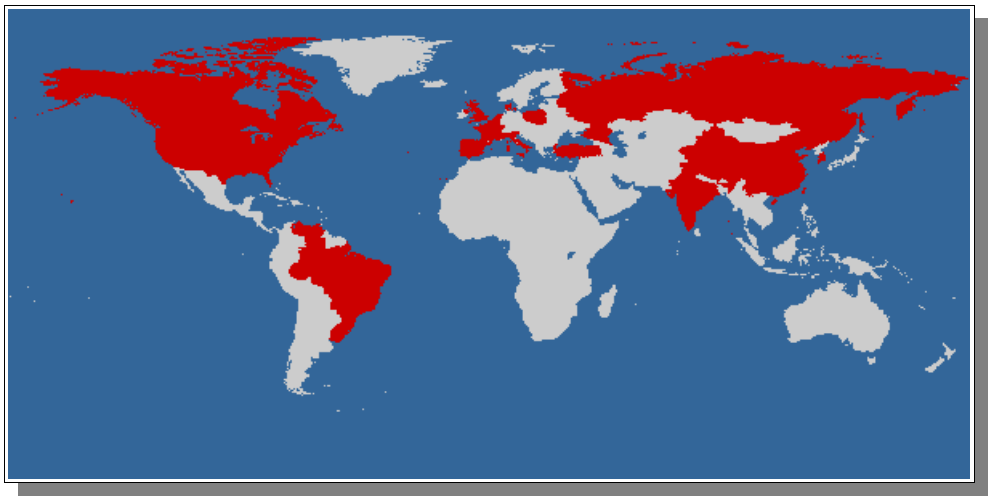


Imagen 6.6: Zonas de origen de los ataques recibidos.

Los ataques representados en la Imagen 6.6 se detallan en la Tabla 6.10 , donde para cada país se muestra la cantidad de direcciones IP's distintas que participaron y el total de accesos registrados.

6.2 Capa de datos de Honeywall

Honeywall, al utilizarse a modo de *gateway*, tiene una visión global de las conexiones y ataques que se realizan hacia la *honeynet*.

6.2.1 Ataques registrados

El número de ataques registrados por *Honeywall* es superior al registrado por los *honeypots*. Esto evidencia deficiencias en la configuración utilizada. Esta situación permitió observar que los *honeypots* no estaban capturando los ataques al servicio SSH. Esto se debió a que se tenía el verdadero servicio (en el puerto TCP 22) utilizado para la administración de los *HoneyCells*. Esto motivo una reconfiguración del *HoneyCell*, la cual consistió en el cambio de puerto del servicio SSH real, y el bloqueo del mismo mediante el firewall. Observando las tendencias de ataques al puerto TCP 22 y se reconfiguraron los *honeypots* para emular este servicio.

Por otro lado *Honeywall* muestra registros de tráfico ICMP, tráfico que *Honeytrap* no detecta. Esto muestra claramente una limitación de esta implementación.

Pais	Cant. Accesos	Cantidad de IP's
India	558	2
Portugal	378	2
United States	76	31
China	45	17
Netherlands	32	3
Taiwan	25	19
Denmark	24	3
Russian Federation	19	6
Spain	18	7
Uruguay	13	5
Belgium	13	1
Italy	10	5
Canada	9	5
Poland	9	5
Brazil	6	3
France	5	3
Venezuela	5	2
Turkey	5	1
Republic of Korea	4	3
United Kingdom	4	4

Tabla 6.10: Cantidad de ataques por país

6.2.2 Análisis de tráfico de ataques

A continuación se analiza la actividad registrada hacia los tres puertos más atacados: TCP 1433 *MSSQL Server*, 139 NetBIOS y 25 SMTP.

En la actividad registrada en el puerto TCP 1433, se observa gran cantidad de intentos de conexión al servicio *MSSQL Server*. Decodificando los paquetes transmitidos se puede observar en casi todos los casos el constante envío de strings de conexión.

En los Fragmentos 6.1 a 6.4 se muestran ejemplos de strings de conexión decodificados. Estos siempre conservan la dirección IP, un string S.2.s.a y un string que sufre pequeñas variaciones en cada ejemplo.

```
12/24-12:32:44.349370 0:E0:4C:8D:7:1C -> 0:C:29:52:A8:F4 type:0x800 len:0xC0
213.58.174.38:4278 -> XXX.XXX.XXX.XXX:1433 TCP TTL:103 TOS:0x0 ID:10819 IpLen:20 DgmLen:178 DF
***AP*** Seq: 0x6F4A3707  Ack: 0x1A262F5B  Win: 0xFFDA  TcpLen: 20
10 01 00 8A 00 00 01 00 82 00 00 00 01 00 00 71  ...............q
00 00 00 00 00 00 00 07 BC 19 00 00 00 00 00 00  ................
E0 03 00 00 3C 00 00 00 16 08 00 00 56 00 03 00  ....<.......V...
5C 00 02 00 60 00 00 00 00 60 00 00 00 60 00 0D 00  \...`....`...`...
00 00 00 00 7A 00 04 00 82 00 00 00 82 00 00 00  ....z...........
00 0C 6E 20 C5 2A 00 00 00 00 82 00 00 00 4E 00  ..n .*........N.
53 00 32 00 73 00 61 00 xx xx xx xx xx xx xx xx  S.2.s.a.x.x.x...
xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx  x.x...x.x...x.x.
xx xx xx xx xx xx xx xx xx xx                    2.O.D.B.C.
```

Fragmento 6.1: Ejemplo de string de conexión transmitido (1).

```
12/24-12:32:47.438954 0:E0:4C:8D:7:1C -> 0:C:29:52:A8:F4 type:0x800 len:0xC4
213.58.174.38:4331 -> XXX.XXX.XXX.XXX:1433 TCP TTL:103 TOS:0x0 ID:10933 IpLen:20 DgmLen:182 DF
***AP*** Seq: 0x2A6609C2  Ack: 0x1D0B599E  Win: 0xFFDA  TcpLen: 20
10 01 00 8E 00 00 01 00 86 00 00 00 01 00 00 71  ...............q
00 00 00 00 00 00 00 07 BC 19 00 00 00 00 00 00  ................
E0 03 00 00 3C 00 00 00 16 08 00 00 56 00 03 00  ....<.......V...
5C 00 02 00 60 00 02 00 64 00 00 00 64 00 0D 00  \...`...d...d...
00 00 00 00 7E 00 04 00 86 00 00 00 86 00 00 00  ....~...........
00 0C 6E 20 C5 2A 00 00 00 00 86 00 00 00 4E 00  ..n .*........N.
53 00 32 00 73 00 61 00 92 A5 B3 A5 xx xx xx xx  S.2.s.a.....x.x.
xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx  x...x.x...x.x...
xx xx xx xx xx xx xx xx xx xx xx xx xx           x.x.x.O.D.B.C.                  2.O.D.B.C.
```

Fragmento 6.2: Ejemplo de string de conexión transmitido (2).

```
12/24-12:33:00.630267 0:E0:4C:8D:7:1C -> 0:C:29:52:A8:F4 type:0x800 len:0xCC
213.58.174.38:4532 -> XXX.XX.XX.XXX:1433 TCP TTL:103 TOS:0x0 ID:11452 IpLen:20 DgmLen:190 DF
**AP** Seq: 0x89154FC8 Ack: 0x2988EA46 Win: 0xFFDA TcpLen: 20
10 01 00 96 00 00 01 00 8E 00 00 00 01 00 00 71 ...............q
00 00 00 00 00 00 00 07 BC 19 00 00 00 00 00 00 ................
E0 03 00 00 3C 00 00 00 16 08 00 00 56 00 03 00 ....<.......V...
5C 00 02 00 60 00 06 00 6C 00 00 00 6C 00 0D 00 \...`...l...l...
00 00 00 00 86 00 04 00 8E 00 00 00 8E 00 00 00 ................
00 0C 6E 20 C5 2A 00 00 00 00 8E 00 00 00 4E 00 ..n .*........N.
53 00 32 00 73 00 61 00 92 A5 B3 A5 B6 A5 86 A5 S.2.s.a.........
xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx ....x.x.x...x.x.
xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx ..x.x...x.x.x.O.
xx xx xx xx xx xx                               D.B.C.
```

Fragmento 6.3: Ejemplo de string de conexión transmitido (3).

```
12/24-12:33:06.863838 0:E0:4C:8D:7:1C -> 0:C:29:52:A8:F4 type:0x800 len:0xD0
213.58.174.38:4626 -> XXX.XX.XX.XXX:1433 TCP TTL:103 TOS:0x0 ID:11668 IpLen:20 DgmLen:194 DF
**AP** Seq: 0x6487EB2D Ack: 0x2F0A9B15 Win: 0xFFDA TcpLen: 20
10 01 00 9A 00 00 01 00 92 00 00 00 01 00 00 71 ...............q
00 00 00 00 00 00 00 07 BC 19 00 00 00 00 00 00 ................
E0 03 00 00 3C 00 00 00 16 08 00 00 56 00 03 00 ....<.......V...
5C 00 02 00 60 00 08 00 70 00 00 00 70 00 0D 00 \...`...p...p...
00 00 00 00 8A 00 04 00 92 00 00 00 92 00 00 00 ................
00 0C 6E 20 C5 2A 00 00 00 00 92 00 00 00 4E 00 ..n .*........N.
53 00 32 00 73 00 61 00 92 A5 B3 A5 xx xx xx xx S.2.s.a.........
xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx ........x.x.x...
xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx x.x...x.x...x.x.
xx xx xx xx xx xx xx xx xx                      x.O.D.B.C.
```

Fragmento 6.4: Ejemplo de string de conexión transmitido (4).

Recabando información sobre el ataque, se tiene indicios de que se esta en presencia de un ataque de *MSSQL Hello Buffer Overflow* [Benj03]. Este se basa en alterar el contenido del primer paquete enviado por el cliente, explotando una condición de *stack buffer overflow*; lo que posibilita la ejecución de código arbitrario por parte del atacante.

En este caso si estamos en condiciones de asegurar que los ataques vistos se tratan de un ataque de *Hello Buffer*, despejando la duda planteada en la sección 7.1.2 ya que se cuenta con datos más concretos.

En cuanto a la actividad en el puerto TCP 139, se tienen transmisiones de strings que en principio no son muy claros. Esto se puede observar en los Fragmento 6.5 y 6.6.

```
01/12-10:02:39.245734 0:E0:4C:8D:7:1C -> 0:C:29:52:A8:F4 type:0x800 len:0x7E
194.109.225.85:3262 -> XXX.XXX.XXX.XXX:139 TCP TTL:103 TOS:0x0 ID:40384 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0x90015933  Ack: 0xE4AACDD2  Win: 0x4470  TcpLen: 20
81 00 00 44 20 43 4B 46 44 45 4E 45 43 46 44 45  ...D CKFDENECFDE
46 46 43 46 47 45 46 46 43 43 41 43 41 43 41 43  FFCFGEFFCCACACAC
41 43 41 43 41 00 20 45 46 46 43 45 4A 45 44 43  ACACA. EFFCEJEDC
41 43 41 43 41 43 41 43 41 43 41 43 41 43 41 43  ACACACACACACACAC
41 43 41 43 41 41 41 00                          ACACAAA.
```

Fragmento 6.5: Ejemplo de transmisión al puerto TCP 139 (1).

```
01/12-10:02:40.481175 0:E0:4C:8D:7:1C -> 0:C:29:52:A8:F4 type:0x800 len:0xBF
194.109.225.85:3262 -> XXX.XXX.XXX.XXX:139 TCP TTL:103 TOS:0x0 ID:40472 IpLen:20 DgmLen:177 DF
***AP*** Seq: 0x9001597B  Ack: 0xE4AACDD6  Win: 0x446C  TcpLen: 20
00 00 00 85 FF 53 4D 42 72 00 00 00 00 18 53 C8  .....SMBr.....S.
00 00 00 00 00 00 00 00 00 00 00 00 00 FF FE  ................
00 00 00 00 00 62 00 02 50 43 20 4E 45 54 57 4F  .....b..PC NETWO
52 4B 20 50 52 4F 47 52 41 4D 20 31 2E 30 00 02  RK PROGRAM 1.0..
4C 41 4E 4D 41 4E 31 2E 30 00 02 57 69 6E 64 6F  LANMAN1.0..Windo
77 73 20 66 6F 72 20 57 6F 72 6B 67 72 6F 75 70  ws for Workgroup
73 20 33 2E 31 61 00 02 4C 4D 31 2E 32 58 30 30  s 3.1a..LM1.2X00
32 00 02 4C 41 4E 4D 41 4E 32 2E 31 00 02 4E 54  2..LANMAN2.1..NT
20 4C 4D 20 30 2E 31 32 00                        LM 0.12.
```

Fragmento 6.6: Ejemplo de transmisión al puerto TCP 139 (2).

Investigado el patrón del ataque, este corresponde a la secuencia de escaneo previa a la infección del gusano *W32. Sasser.Worm* [MiTN04] [LkLg06] el cual accede mediante el puerto TCP 139. El gusano detecta el puerto abierto intentando establecer la comunicación una y otra vez, pero dado que no se brinda respuesta al intento no se concreta la infección.

Aquí puede observarse el papel de los *honeypot* retrasando el ataque automático, ya que se insistió una y otra vez con el escaneo hacia el puerto.

Por último la actividad referente al puerto TCP 25, permitió registrar los strings mostrados en los Fragmentos 6.7 y 6.8.

```
12/07-15:36:37.727917 0:C:29:52:A8:EA -> 0:E0:4C:8D:7:1C type:0x800 len:0x52
XXX.XXX.XXX.XXX:25 -> 220.141.37.20:3213 TCP TTL:64 TOS:0x0 ID:54829 IpLen:20 DgmLen:68 DF
***AP*** Seq: 0x6BA67C3C  Ack: 0x2093A74F  Win: 0x5B4  TcpLen: 20
32 35 30 20 6C 6F 63 61 6C 68 6F 73 74 20 45 53  250 localhost ES
4D 54 50 20 50 6F 73 74 66 69 78 0A               MTP Postfix.
```

Fragmento 6.7: Ejemplo de transmisión al puerto TCP 25 (1).

```
03/14-20:41:50.574936 0:E0:4C:8D:7:1C -> 0:C:29:9E:15:88 type:0x800 len:0x3C
190.0.129.33:20064 -> XXX.XXX.XXX.XXX:25 TCP TTL:118 TOS:0x0 ID:22709 IpLen:20 DgmLen:46 DF
***AP*** Seq: 0x13A87A21  Ack: 0x32EF4ED3  Win: 0x4454  TcpLen: 20
51 55 49 54 0D 0A                                 QUIT..
```

Fragmento 6.8: Ejemplo de transmisión al puerto TCP 25 (2).

A partir de este registro no fue posible hallar un patrón que permitiera identifica los ataques a este puerto.

6.3 Capa de datos de HoneyCenter

Aunque *HoneyCenter* se encuentra oculto y con mecanismos de protección, no se encuentra exento de recibir ataques.

Analizando los datos obtenidos por el *HoneyCenter*, no se encontraron indicios de ataques al mismo.

6.4 Conclusión

Se observa el beneficio de la centralización de los datos, ya que permite contemplar los datos de ambos *honeypot* a la vez y relacionarlos. A su vez se hace notar la falta de centralización de datos de *Honeywall*.

El incluir herramientas de análisis y estadísticas como *Open Reports* permite el proceso de datos en forma más dinámica y fácil. Igualmente, para un mejor análisis se debe personalizar más la herramienta. Se incluyen reportes que permiten observar ataques de todas partes del mundo, siendo el país con más ataques registrados India.

Se confirma como puerto más atacado el TCP 1433 correspondiente al servicio de *Microsoft SQL Server*, lo cual se alinea con la tendencia mundial, siendo seguido por el puerto TCP 139 NetBIOS en segundo lugar y el TCP 25 en tercer lugar.

Los datos registrados por *Honeywall* muestran que la cantidad de ataques es superior a la registrada por los *honeypots,* evidenciando deficiencias y/o limitaciones en la captura de determinado tipo de ataques así como en la configuración de los mismos.

También es importante recalcar que la información capturada por *Honeywall* es complementaria a los datos capturados por *HoneyCell*, con lo cual afirma la necesidad de normalizarlos y almacenarlos en el repositorio para posteriormente analizar los datos en conjunto.

CAPÍTULO 7 GESTIÓN DEL PROYECTO

El presente capítulo trata sobre la organización que define y guía al grupo a lo largo de todo el desarrollo del proyecto. Se describen las etapas definidas y los objetivos planteados para cada una de ellas. Se enuncian también los problemas ocurridos que llevaron a un atraso en la ejecución del plan.

7.1 Organización

En una primera instancia se realizó una estimación inicial a grandes rasgos con granularidad mensual. Avanzado el primer mes de investigación se realizó una segunda estimación más detallada de las tareas del proyecto.

En el mes de Agosto del 2007, luego de finalizada la etapa de investigación, se realizó una replanificación de las tareas que se debían realizar y una estimación del tiempo que llevaría realizarlas. En este nuevo punto de planificación, se distinguen las fases del proyecto: la fase de investigación de la realidad planteada, el diseño de la solución y la fase de implementación.

En la imagen 7.1 se muestra la ejecución de la fase de investigación.

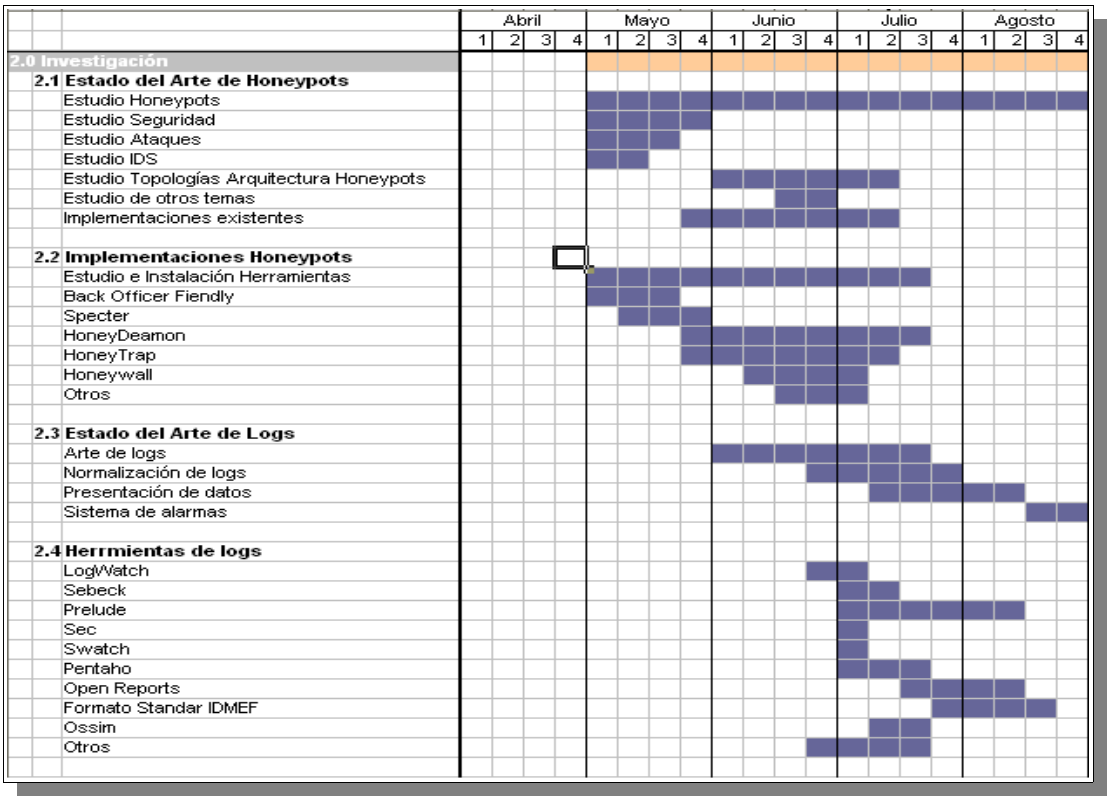


Imagen 7.1: Detalle de tareas de investigación

Para la planificación del proyecto se crean Diagramas de Gantt, como el de la Imagen 7.1, donde se muestra la planificación para la fase de investigación.

Según la propuesta realizada [Prop08], se desprende que la fase de implementación se divide en las siguientes subfases:

- **1 Arquitectura básica:** instalación y configuración de los distintos componentes.
- **2 Arquitectura con 2 *honeypots* iguales:** despliegue de 2 *honeypots.*
- **3 Arquitectura con 2 *honeypots* distintos:** despliegue de 2 *honeypots* de distintas herramientas.
- **4 Análisis y presentación de los datos: i**mplantación del sistema de análisis de datos.
- **5 Arquitectura distribuida:** extensión a una *honeynet* generación IV.
- **6 Análisis de los datos capturados.**

Para la fase de implantación fue recomendado acortar el alcance de la solución, excluyendo de la misma lo referente a la implantación de una arquitectura distribuida (subfase 5). Esto se apoya en que el trabajo de esta fase agrega complejidad a la solución, traduciendo esto en la extensión de plazos del proyecto. Además se debe tener en cuenta, que realizando las demás fases se cumple tanto con los objetivos iniciales del proyecto, como los que se incluyeron en la fase de investigación.

Ocurrieron desvíos en la planificación de actividades. Los mismos son atribuidos principalmente a la puesta en producción y recolección de datos. De todas maneras se continuó con las etapas faltantes. Esta acción permitió neutralizar el desvío manteniendo los plazos de la fase de implantación, pero no los tiempos por cada subfase. La consecuencia fundamental es que, no se pudo desplegar la arquitectura básica en producción a principios de Septiembre, tal como se proponía, pero permitió desplegar la arquitectura completa a fines de Octubre del 2007 como estaba planificado.

En la imagen 7.2 se muestra la planificación de la ejecución de la fase de diseño e implantación.

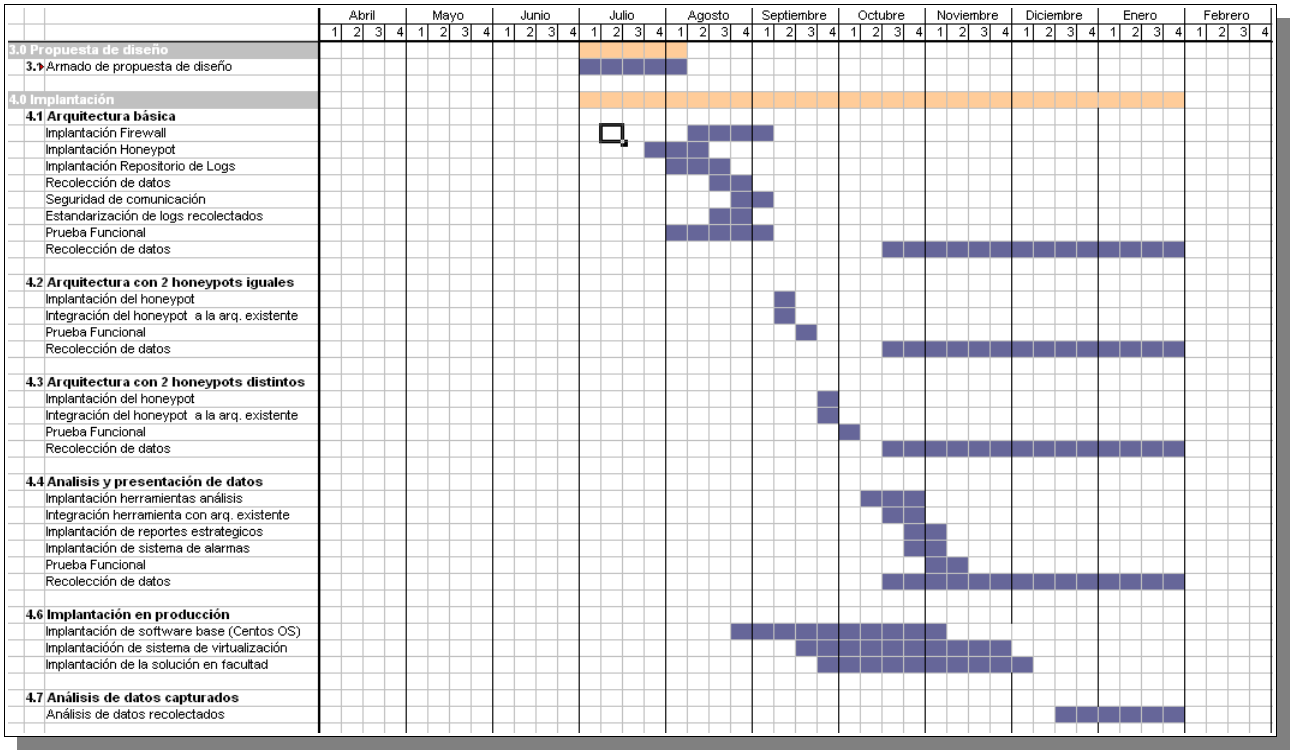


Imagen 7.2: Detalle de tareas de fase de armado de propuesta y fase de Implantación.

Un problema que no pudo mitigarse fue la demora en el avance de la documentación, principalmente durante la fase de implantación, por lo cual fue necesario recuperar el tiempo al final de la puesta en producción. La gran cantidad de información a documentar y los problemas de redacción del grupo afectan directamente en el avance por lo cual se modificaron las fechas previstas extendiendo los plazos iniciales del proyecto.

En la imagen 7.3 se observa la ejecución del desarrollo de la documentación.

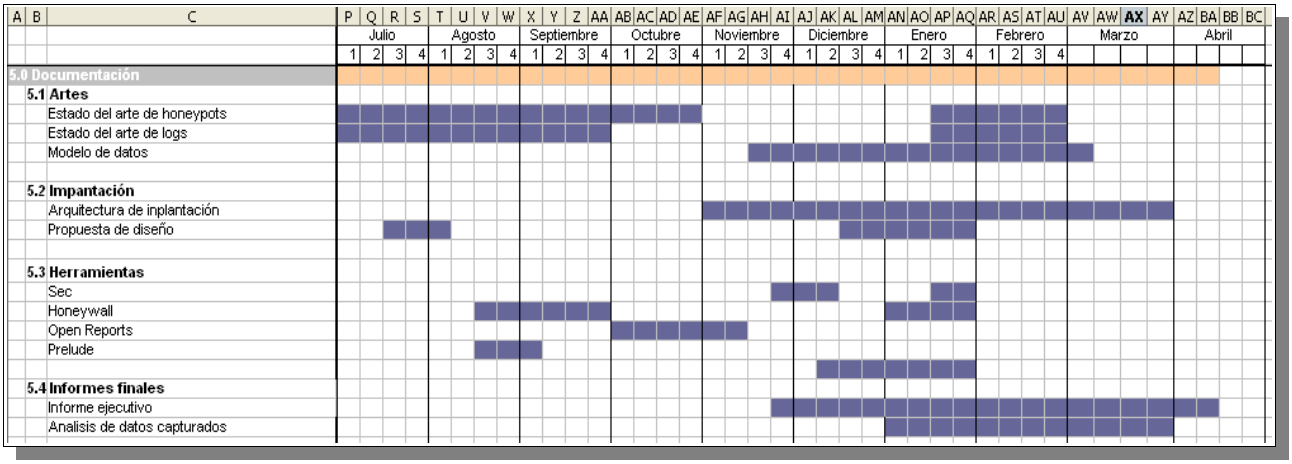


Imagen 7.3: Detalle de tareas de documentación.

CAPÍTULO 8 CONCLUSIONES

Luego del estudio realizado para poder llevar a cabo el proyecto y después de más de dos meses de recolección de datos con la solución implementada, se realizó el análisis de la información obtenida. Desde los inicios del análisis y a lo largo de todo el proceso de desarrollo, se han obtenido diversas conclusiones que el grupo ha plasmado a lo largo del documento. En el presente capítulo se realiza la conclusión global a todo el proyecto de grado "*Diseño e Implementación de un honeypot*".

8.1 Conclusiones obtenidas

Tradicionalmente, las tecnologías de seguridad de la información se basan en intentar bloquear los ataques o detectarlos antes de que penetren e irrumpan en los sistemas (firewalls, sistemas de detección de intrusos, encriptación, antivirus, etc.). En cambio la tecnología *honeypots* presenta una alternativa distinta, enfocada en la exposición a los ataques. Desplegando un sistema que resulte atractivo para el atacante, pero sin valor de producción, que registre todas las acciones que se realicen con él. Buscando así aprender de los ataques y métodos utilizados por los atacantes para realizarlos. En algunos casos se intenta además enlentecer o detener los ataques; como es el caso de la implementación llamada *Labrea Tarpit*. Durante la investigación sobre el "*estado del arte*" de la tecnología, se pudo establecer que, si bien se basa en técnicas y conceptos milenarios (descriptas por el general chino Sun Tzu en su libro "*El arte de la guerra*"), se trata de una rama de investigación relativamente nueva en el área de la seguridad informática. Cliff Stoll (en su libro "*The Cuckoo's Egg*") y Bill Cheswick (con "*An Evening with Berferd*") fueron los primeros investigadores en utilizar el concepto de exponer sistemas para su estudio. Desde entonces los *honeypots* han ido evolucionando, igualando conceptos y objetivos; las definiciones de Lance Spitzner y de Reto Baumann son las más conocidas y aceptadas. Actualmente, el *Honeynet Project*, una organización internacional sin fines de lucro (fundada en 1999), se dedica a la investigación y creación de nuevas herramientas y dispositivos para apoyar el crecimiento de la tecnología de *honeypots*. En particular su enfoque está dirigido a las redes de *honeypots (honeynets),* arquitecturas que permiten un mayor nivel de interacción con los atacantes (simulan redes con valor de producción), obteniendo así información más detallada sobre atacantes, ataques y modalidades de intrusión. Sin embargo la implantación y mantenimiento de una *honeynet* puede resultar una tarea altamente compleja. Se deben tener en cuenta varios factores, como por ejemplo el control y la captura de datos para asegurar la integridad de la red. Ya que una falla en la configuración puede poner en riesgo sistemas externos, permitiendo que un atacante que tome el control de la *honeynet* la utilice para realizar ataques. No obstante, un gran avance en el uso y la implantación de *honeynets* se produce a partir de la liberación (por parte del *Honeynet Project*) de la herramienta *Honeywall*, agrupando en un solo dispositivo gran cantidad de herramientas integradas entre sí y configuradas para facilitar el problema de control y captura de datos.

El interés del grupo en esta tecnología, que se encuentra en creciente desarrollo, permitió extender los objetivos y alcance planteados inicialmente. Para este proyecto se realizó la implantación de dos *honeypots* de bajo nivel de interacción (*Honeyd* y *Honeytrap*). En particular se

deseaba integrar ambas implantaciones en una *honeynet*. En base a los conocimientos adquiridos y teniendo presente las dificultades de este tipo de arquitecturas, el grupo se plantea diseñar una solución que permita integrar, de manera sencilla, distintas implementaciones de *honeypots* con diversas características y niveles de interacción. Se diseña e implementa una solución flexible y extensible, orientada al rápido despliegue de una *honeynet*; basada en la centralización en tiempo real de la información obtenida por los *honeypots* y demás componentes que conforman la red; posibilitando así un análisis y monitoreo en conjunto de todos los datos obtenidos.

Para realizar la centralización de información, es necesario llevar los datos a una representación normalizada. Esto garantiza que los datos obtenidos de un ataque se expresarán siempre de la misma forma, sin importar las características específicas del componente que lo capture. Para realizar la normalización, se hace uso del modelo de datos denominado *Intrusion Detection Exchange Format* (*IDMEF* - RFC 4765), un RFC que va en camino de estandarizarse. Este modelo tiene como propósito definir el formato de los datos (en lenguaje XML) y los procedimientos de intercambio para compartir información de interés en sistemas de detección de intrusos (IDS) y sistemas de respuesta a incidentes de seguridad (CSIRTs). Este modelo resulta extensible y aplicable a otros contextos, como por ejemplo reportes de datos de *honeypots*. Sin embargo, la centralización de datos en tiempo real implica costos extras que deben contemplarse. En este caso, los componentes de la *honeynet* deben estar comunicados con el repositorio central, con lo que se deben establecer medidas para autenticar los *honeypots* con el repositorio y es necesario asegurar la comunicación de los datos.

Teniendo presente los objetivos, características y dificultades de las *honeynets*, el grupo genera un par de herramientas denominadas *HoneyCell* y *HoneyCenter*. Enfocados en la captura de información y en la centralización de datos respectivamente. Estos componentes se distribuyen en formato de *livecd*, facilitando su despliegue, instalación y configuración. Si bien la creación de estas distribuciones resulta costosa (es necesario instalar y configurar todas las herramientas de modo genérico), se tiene como ventaja la reusabilidad y disponibilidad de los sistemas. Además, gracias a este formato, es posible desplegar un componente en cualquier equipo sin necesidad de instalar ningún software. Como se dijo anteriormente, el componente llamado *HoneyCell* es el encargado de la captura de datos, y contiene las implementaciones de los *honeypots* de bajo nivel de interacción *Honeytrap y Honeyd* ya pre-configurados; además de los mecanismos de normalización y envío de datos al *HoneyCenter*. Por su parte el *HoneyCenter*, además de centralizar la información, cuenta con herramientas para el monitoreo y análisis de datos y un sistema de alarmas para notificar las intrusiones detectadas por los componentes de la *honeynet*. Ambas soluciones son totalmente extensibles, permitiendo agregar nuevas implementaciones de *honeypots,* nuevas herramientas de análisis, monitoreo, etc.

Para realizar la implantación se dispone de un equipo cedido por el *Grupo de Seguridad Informática*, que soporta la infraestructura de virtualización utilizando *VMWare Server*. Cada dispositivo de la solución es instalado en una máquina virtual independiente, interconectadas entre sí mediante interfaces y redes virtuales. El uso de tecnologías de virtualización, en particular *VMware Server*, es de gran ayuda a la hora de realizar pruebas de implantación.

Un aspecto a destacar, es la posibilidad de restaurar el sistema a un estado anterior de la maquina virtual. Un estado "*estable*", es definido por medio de un "*snapshot*", es como una fotografía del sistema en el momento en que es tomada (es una imagen del disco y la BIOS de la maquina virtual). Esto es de gran utilidad ya que en caso de fallas de configuración, borrado

accidental de archivos, compromiso del sistema, etc., se puede volver al estado del "*snapshot*" (antes de que ocurriese el problema).Otro punto favorable a tener en cuenta, es la capacidad de realizar una administración remota de las máquinas virtuales; lo que permitió llevar a cabo varias configuraciones a la solución desde fuera de Facultad.

Para la puesta en producción de la solución, se utilizan dos direcciones IP's pertenecientes al rango de la DMZ de Facultad de Ingeniería. Estas direcciones no estaban asignadas a ningún sistema de producción, y no fueron hechas públicas promoviendo algún servicio en ellas, por lo que es de esperar que no les llegue tráfico. Pero la realidad demuestra lo contrario, ya que durante una serie de pruebas a la arquitectura implantada, se registraron los primeros ataques. De esta manera se establece que, el solo hecho de estar conectado a Internet convierte rápidamente un sistema en objetivo para los atacantes.

La solución se comportó como era esperado, con algunas carencias como por ejemplo en lo que al sistema de alarmas refiere, ya que debería trabajarse para mejorar el reconocimiento de ataques. El sistema se mantuvo operativo hasta los primeros días del mes de Abril de 2008, cuando se decidió detenerlo ya que no podía ser monitoreado. Como resultado de la operativa, se realizó el análisis de los datos obtenidos por la solución, en el lapso de tiempo comprendido entre el día 5 de Diciembre de 2007 y el día 13 de Febrero de 2008. En ese plazo se detectaron ataques provenientes de todas partes del mundo. A pesar de haberse utilizado *honeypots* de bajo nivel de interacción, la cantidad de ataques fue importante y más teniendo en cuenta que (como se dijo anteriormente), las IP's no fueron publicadas. De los ataques recibidos, el que resalta notoriamente es un ataque al puerto TCP 1433 donde se obtuvieron más de 500 intentos de conexión en un solo día. Analizando a fondo el origen y tipo de ataque se pudo determinar que se trataba de un ataque a una vulnerabilidad conocida como "*MSSQL Hello Buffer Overflow*". En orden de precedencia por cantidad de ataques, se encuentran los ataques realizados al puerto TCP 139 por el gusano "*W32.Sasser.Worm*".

Por último, se puede indicar que se concluye un proyecto por de más interesante, que cautivó al grupo día a día durante todo su desarrollo. Cada tema investigado referente a esta tecnología motivó el trabajo realizado.

8.2 Problemas encontrados

Al iniciar el proyecto se comenzó a transitar un largo camino en el cual se encontraron algunos obstáculos. A continuación se enumeran los más relevantes:

- El grupo contaba con conocimientos básicos de plataformas Linux, de su administración, instalación y configuración. Esto afectó en las pruebas iniciales ya que se debía adquirir un cierto nivel de conocimiento y uso de las herramientas del sistema a nivel de línea de comando. Este problema pudo ser sobrellevado rápidamente.

- Planificación de horarios y actividades. Los integrantes del grupo disponían de horarios dispares, esto incidía en la coordinación de reuniones. La manera de mitigar este problema fue la de realizar reuniones semanales, principalmente los días Sábado. Allí realizar un plan de acción para la semana en curso, y en caso de disponibilidad, reuniones durante la semana.

- Poca documentación disponible de algunas herramientas estudiadas. Como se dijo anteriormente esta es una tecnología en creciente desarrollo, lo que se transmite a las herramientas que se utilizaron. *Honeywall*, *Honeytrap* y *Fedora* tuvieron actualizaciones, y dentro de lo posible se incluyeron en el desarrollo. En cuanto a documentación *Honeytrap* y *Open Reports* fueron de las herramientas en las que el grupo invirtió más tiempo investigando y realizando su documentación. En el caso de *Open Reports* casi llegando al final del proyecto se hace público el manual de usuario, disponible en [OpRe08].

- Se disponían de varias líneas a seguir y cada una de ellas era de interés para el grupo. La definición de una propuesta que abarcara la mayor cantidad de dichas líneas y a su vez no descuidar los alcances y tiempos establecidos, fue un problema que llevó mas tiempo del estimado.

- Dificultades con *VMware Server*. Se tuvieron dificultades a la hora de instalar la herramienta en el *host base*. Se recuerda que el sistema operativo utilizado es una versión minimal de *CentOS 5*, con lo que se omitieron gran cantidad de paquetes, esto trajo consigo varias fallas por chequeos de dependencias. Una vez identificados los paquetes faltantes se prosiguió con la instalación.

- Configuración de redes virtuales. La mala configuración de estas causó problemas en la red de Facultad. *VMware Server* crea un servidor `DHCP` para asignarle `IP's` a las máquinas virtuales que se conecten a la `VMnet`. Esto implicó que a varios usuarios de la red, en la que se encontraba en ese momento el *host base*, se le diera una `IP` de la red virtual. Esto fue rápidamente solucionado deshabilitando el servidor `DHCP`.

- Creación de *livecds*. Para la creación de los *livecd* es necesario contar con un repositorio desde donde el instalador descarga los paquetes a instalar. Para llevar esto a cabo las primeras pruebas descargaban todos los paquetes necesarios desde Internet, con lo que se demoraba mucho tiempo entre pruebas. Luego se aprendió a configurar repositorios locales para que sean utilizados por *LiveCDTools*. Además, para disponer de los componentes en formato de *livecds*, se requirió una configuración generica de los sistemas.

- Se presentaron dificultades al modificar la distribución de *Honeywall*. Principalmente para transmitir los datos capturados en este al repositorio de centralización. Se encontró que la herramienta está creada de una manera muy hermética, y por más que en la documentación se dice que es fácil extenderla, el grupo no opina lo mismo. Se estuvo un tiempo considerable intentando actualizar los paquetes de la distribución sin lograrlo, por lo tanto este objetivo se dejó de lado para cumplir con el resto.

- Configuración de los *honeypots*. Las implementaciones de *honeypots* de bajo nivel de interacción son de las más fáciles en implantar y mantener. No obstante el grupo experimentó problemas a la hora de su configuración. Luego de varios ajustes derivaría en la configuración que se tiene hoy en los *HoneyCells*.

8.3 Lecciones aprendidas

El interés del grupo en el área, es un factor importante ya que despierta inquietudes por adquirir conocimientos en ella. Otro factor que cabe resaltar es el alto nivel de compromiso asumido para la realización de este proyecto. Todo el desarrollo del proyecto fue de constante aprendizaje, pero el grupo desea resaltar los siguientes puntos:

- Se profundizaron los conocimientos de seguridad informática. Se analizaron herramientas utilizadas para realizar ataques y se estudiaron algunos de los ataques más conocidos. Esto fue principalmente durante el trabajo realizado para el desarrollo del "*Estado del arte de honeypots*", pero también durante la implantación de la solución y durante el análisis de datos recolectados por los *honeypots*.

- Inicialmente la instalación de paquetes fue realizada vía `rpm`, luego se conocería la herramienta `yum`, que simplifica enormemente los tiempos de instalación al descargar las dependencias necesarias para la instalación.

- Se descubrió que no se puede ejecutar una máquina virtual dentro de otra utilizando *VMware Server*. Esto se vio cuando se quiso replicar el ambiente de Facultad en nuestras casas.

CAPÍTULO 9 TRABAJOS A FUTURO

A medida que se avanzaba en la realización del proyecto se encontraban nuevas líneas para proseguir la investigación y el desarrollo. A continuación se describen varios de los trabajos a futuro que pueden realizarse, para profundizar sobre la tecnología de *honeypots* o para enriquecer la solución implantada.

9.1 Estudio de comportamiento de intrusos

La solución implantada ha capturado datos desde los primeros días de Diciembre a la fecha, brindando un panorama de lo que está pasando en la red. A medida que incrementa la cantidad de datos obtenidos, se hacen notorias las modificaciones a la implementación actual de modo de maximizar la captura de datos.

Realizar un estudio del comportamiento y realizar un seguimiento de la actividad del atacante. Para ello se deben considerar varios puntos:

- Agregar nuevas implementaciones de *honeypots* a la *honeynet*. De esta manera tener varios servicios que simulan ser una red real de producción. Por ejemplo integrar un servidor Web e instalar *GoogleHack Honeypot.*
- Integrar a la red nuevos elementos que emulen diferentes sistemas operativos (esto puede llevarse a cabo con *Honeyd* o con nuevos *HoneyCells* generados con otros sistemas base).
- Emular nuevos servicios para la captura de ataques.
- Mejorar o extender el comportamiento de los scripts que emulan los servicios, mejorando la credibilidad sobre el *honeypot*. Esto llevará a un atacante a esmerarse más por atacarlo.
- Emular servicios específicos para capturar *malware*.

9.2 Extender la centralización de datos

Uno de lo temas que el proyecto no pudo completar por diferentes razones, es el de lograr la centralización completa de los datos recolectados por los componentes de la *honeynet.* Específicamente, una de la metas del proyecto era la centralización de los datos capturados por el componente *Honeywall*. Con ello es posible llevar a cabo un análisis más meticuloso, permitiendo correlacionar los datos capturados.

Para lograr este objetivo se deben instalar en el *Honeywall,* las herramientas utilizadas para la normalización de datos.

La centralización puede hacerse extensible a otros componentes que integren la solución. Por ejemplo si se instala *Sebek* [HPSe05], es deseable integrar la información obtenida por esta

herramienta al motor de centralización.

9.3 Normalización y Centralización de datos capturados

La solución implantada realiza la normalización de datos de intrusiones tomando como referencia el modelo de datos *IDMEF*. Este modelo es un RFC que establece un formato de datos estándar para el reporte de intrusiones.

Se propone seguir una línea de investigación que permita extender y aplicar este modelo en la tecnología de *honeypots*. De manera de potenciar la normalización de datos, así como la centralización de los mismos, interoperando con otros componentes como ser Firewall, IDS, etc.

Específicamente, se debe considerar características como:

- uso de *honeypots* de medio/alto nivel de interacción.
- registro de captura de malware.
- registros de alertas de otros componentes de seguridad

En la solución se utilizó como implementación del modelo el *Prelude IDS Framework*, pero puede resultar interesante desarrollar una implementación propia.

9.4 Presentación de datos refinada

Actualmente se cuenta con un conjunto de reportes realizados con *Open Reports* para el monitoreo y análisis de los datos. Este conjunto de reportes brinda la posibilidad de analizar los datos por frecuencias, objetivos y fuentes de ataques. La herramienta fue utilizada por sus ventajas a la hora de generar nuevos reportes (ya que permite crearlos de forma rápida y sencilla) y por su interfaz full Web. Pero es una herramienta muy general de reporting, por lo que debería ser personalizada para el monitoreo de datos referidos a incidentes de seguridad. O realizar el desarrollo de una herramienta específica para este tipo de infraestructura, que permitan el monitoreo de intrusiones, así como la presentación de los datos capturados apuntando a presentación en tiempo real de intrusiones. Incluso se pueden incorporar sistemas de análisis multidimensional y sistemas de Data Mining para mejorar la visualización de datos.

9.5 Refinamiento del Sistema de alarmas

Actualmente el sistema de alarmas realiza el envío de un correo electrónico por cada intrusión detectada, comunicándose al administrador de forma inmediata de la actividad en la red. Este sistema resulta muy básico e ineficiente, pero fue concebido para ser extendido de manera de establecer reglas que permitan identificar patrones de ataques, como por ejemplo port scannings, ataques DoS, ataques a un puerto en particular, etc. Esto permite el envío de alertas personalizadas de acuerdo al tipo de intrusión detectada.

Se propone refinar el sistema de alarmas permitiendo mejorar las siguientes características:

- Correlación de eventos permitiendo identificar los patrones de ataques y de este modo enviar alarmas más precisas.

- Identificación de ataques críticos.
- Identificación de varios ataques de una fuente en tiempos cortos.
- Enviar la alarma por otro medio, como por ejemplo, en una consola, en la Web de monitoreo, vía SMS, etc.

9.6 Incorporar nuevas implementaciones de honeypots

Dadas las características de la solución propuesta y con el objetivo de recabar la mayor cantidad de información posible, se propone la extensión de la solución incorporando nuevas implementaciones de *honeypots.*

Aquí es posible introducir el estudio de otros tipos de *honeypots* centrados en ataques Web, como ser el envío de correo *spam*, etc. Una incorporación ambiciosa, pero contemplada en el análisis y diseño de la solución, sería la de un *honeypot* de alto nivel de interacción.

9.7 Implementación de honeynet de generación IV

Resulta muy interesante extender la solución para alcanzar una *honeynet* de generación IV: *honeynet* distribuida, cruzando información con otras *honeynets* implementadas.

La solución planteada por el presente proyecto permite transmitir de repositorio a repositorio los datos obtenidos en la *honeynet* a la que pertenece cada uno de ellos. Pero esto depende de varios factores técnicos como políticos a resolver. Por ejemplo:

- Confidencialidad de datos reportados, no es deseable para una organización brindar información interna.
- Intercomunicación de redes, para comunicar organizaciones se deben sortear firewalls, ruteos, etc. para intercomunicar redes. Al igual que el punto anterior no es deseable brindar información interna.
- Frecuencia de recibo/envío de datos entre los diferentes repositorios.

Una de las ideas que manejó el grupo, fue la posibilidad de realizar una interconexión con la *Honeynet Project de Brasil* [HPBr02]; o más aun se pensó en iniciar trámites para iniciar un capítulo en Uruguay del *Honeynet Project* [HoPr02].

9.8 Crear otras honeynets en Facultad

Con el objetivo de obtener una medida de la actividad de atacantes en torno a la red de Facultad, se propone la implantación de nuevas *honeynets* en diferentes puntos de la red (como por ejemplo la red de enseñanza). Para ello se deben tener en cuenta varias puntualizaciones:

- Ubicación de la *honeynet.*
- Se deben establecer los servicios a emular, enfocándose al perfil de los atacantes. Esto depende de la ubicación de la *honeynet*.

- Medidas de seguridad, lo cual va de la mano con los servicios emulados y la ubicación.

Como se mencionó en el capítulo de implantación, la solución desarrollada se llevo a cabo en la DMZ de la red de Facultad. Esta disposición perseguía específicamente atraer ataques externos.

9.9 Ataques DoS

Una característica de la que adolece cualquier infraestructura de seguridad es la vulnerabilidad de las mismas a los ataques DoS, por lo que una línea interesante a seguir es el estudio de ataques DoS haciendo uso de *honeypots*.

La utilización de *honeypots* por si solos podría resultar insuficiente para estudiar este tipo de ataques ya que solo recolectan información del mismo. Esto llevaría a la utilización de otros componentes en conjunto, para lograr:

- Identificar que se esta bajo un ataque DoS
- Recolectar información sensible del mismo
- Reaccionar para que el ataque no tenga efecto

El punto crucial a tener en cuenta es como hacer atractivo el *honeypot* para ataques DoS y lograr un equilibrio entre recolectar información relevante de este ataque sin ser afectado por el mismo.

9.10 Wireless honeypots

Cada vez son más comunes los dispositivos Wireless en las distintas organizaciones gracias a las prestaciones que brindan, como por ejemplo la conexión sin necesidad de cables, menores costos y tiempos de implantación, movilidad de los dispositivos conectados, etc.

Estas son redes interesantes de investigar, ya que un individuo no necesita estar conectado a Internet para lanzar un ataque. Con solo tener un PC con un dispositivo Wireless que detecte la presencia de una red de este tipo le basta para atacarla o usar una red de acceso gratuito para realizar un ataque a otros objetivos.

Se proponen 2 líneas de investigación a seguir: *Wireless Honeynets* y la solución que se puede denominar *HoneyWAP*.

Una *Wireless Honeynet* es una *honeynet* proyectada para la captura e investigación de actividad Wireless existente, pudiendo detectar técnicas y estrategias de los atacantes en las mismas. Este tipo de redes son *clónicas* respecto a las redes tradicionales, simplemente es necesaria una tarjeta de red WiFi para cada elemento de la red y un Hub, Router o WAP que la cohesione.

Un *HoneyWAP* (*Honey Wireless Access Point*) emula un dispositivo WAP, permitiendo recabar información sobre estrategias y técnicas de ataques a los mismos. WAP (*Wireless Access Point*) es el dispositivo que permite interconectar dispositivos de comunicación inalámbrica para formar una red inalámbrica.

9.11 Otros trabajos a futuro

Como otros trabajos a futuro se desean plantear:

- La creación de una DMZ para la red de enseñanza. La escasez de direcciones IP's libres en el rango de la DMZ de Facultad, incidió en que no se pudiera desplegar la *honeynet* como se tenía previsto.

- Estudio de "*active honeypots*". A lo largo del proyecto el enfoque hacia los *honeypots* fue como el de entidades pasivas, de visión limitada (solo ven los ataques dirigidos a ellos) y que obtienen información del ataque y del atacante. Un enfoque distinto al tradicional es que las entidades no sean pasivas. Es decir, además de distraer al atacante de los sistemas de producción, el *honeypot* obtenga la mayor cantidad de información adicional posible. Por ejemplo haciendo uso de herramientas como finger, tracer, whois, DNS, etc. Dichas entidades son conocidas como *active honeypots*, ya que reaccionan ante un ataque intentando obtener toda la información posible del atacante [DiJo04].

- Investigar más a fondo la implantación de granja de *honeypots* y las herramientas disponibles con tal fin.

- Metalenguaje para generar configuraciones de *honeypots* de bajo nivel de interacción. Las implementaciones de *honeypots* actuales utilizan archivos de configuración para establecer su comportamiento. La idea es establecer un metalenguaje y un generador de archivos de configuración y de este modo abstraerse de la implementación.

- Sistema para administrar una red de *honeypots* de bajo nivel de interacción. Desarrollar un sistema de monitoreo y configuración en tiempo real de la red y de cada *honeypot*. Desde donde se pueda definir el comportamiento, puertos abiertos, respuestas, topología de la red, etc., sin necesidad de interactuar directamente con los componentes.

ANEXO A: GLOSARIO

0day attack: Ataques que no han sido descubiertos.

API: Application Program Interface. Conjunto de rutinas, protocolos, y herramientas para la construcción de aplicaciones de software.

Auditoria forense: Es el proceso de identificar, preservar, analizar y presentar la evidencia digital de una manera legalmente aceptable.

BackOrifice: Es un programa de control remoto de equipos, que trabaja con una arquitectura servidor-cliente.

Bridge: Dispositivo que permite la interconexión de dos redes con igual o distintos interfaces o pila de protocolos.

BOF: Back Officer Friendly. *Honeypot* de bajo nivel de interacción.

BSD: Berkeley Software Distribution. Sistema operativo derivado de Unix nacido a partir de las aportaciones realizadas a ese sistema por la Universidad de California en Berkeley.

cDc: Cult Of the Dead Cow. Grupo de Blackhats que desarrollaron BackOrifice.

CentOS: Comunity Enterprise Operative System. Es un sistema operativo clon de *RedHat Enterprise Linux*.

CERT: Computer Emergency Response Team. Fue creado por DARPA en 1988, con el objetivo de trabajar junto a la comunidad Internet para facilitar su respuesta a problemas de seguridad informática que afecten a los sistemas centrales de Internet.

chkconfig: Comando de algunas distribuciones de Linux para registrar servicios.

Correlación de Eventos: Refiere a la capacidad de agrupar y asociar eventos generados por los distintos dispositivos, con el objetivo de facilitar la labor del análisis de los eventos, especificando el máximo detalle posible de los mismos.

CSIRT: Computer Security Incident Response Team. Es una organización que es responsable de recibir, revisar y responder a informes y actividad sobre incidentes de seguridad. Un CSIRT puede ser un equipo formalizado o un equipo *ad hoc*. Un equipo formalizado realiza un trabajo de respuesta a incidentes como su función principal. A un equipo *ad hoc* se lo convoca durante un incidente de seguridad que esté ocurriendo en ese momento o para responder a un incidente cuando surge la necesidad.

DDoS: Distributed Denial of Service. Estrategia de ataque que coordina la acción de múltiples sistemas para saturar a la víctima con información inútil para detener los servicios que ofrece. Los sistemas utilizados para el ataque suelen haber sido previamente comprometidos, pasando a ser controlados por el atacante mediante un cliente DDoS.

DIDS: Distributed IDS. Sistema basado en la arquitectura cliente-servidor que está compuesto por múltiples IDS que actúan como sensores centralizando la información de posibles ataques en una unidad central.

DoS: Denial of Service. Estrategia de ataque que consiste en saturar de información a la víctima con información inútil para detener los servicios que ofrece.

DTK: Deception Tool Kit. *Honeypot* de bajo nivel de interacción.

dtspcd: CDE Subprocess Control Service, es un demonio de red que acepta requerimientos de clientes para ejecutar comandos y lanzar aplicaciones remotamente.

Emular: Imitar el comportamiento o las acciones de otro, procurando igualarlas o superarlas. En el ámbito del proyecto se utiliza el término para imitar comportamientos de servicios y sistemas operativos.

Exploit: Programa malicioso que intenta forzar alguna vulnerabilidad de otro programa con el fin de inhabilitación del sistema objetivo o la destrucción misma.

Falso positivo: Es un error, donde se detecta una falla o malfuncionamiento de otro individuo cuando en realidad no lo hay. En caso de un antivirus, existe un falso positivo cuando se detecta un virus en un archivo que no lo tiene.

Falsos negativo: Es un error donde no se detecta el error cuando en realidad existe. En caso de un antivirus, existe un falso negativo cuando un archivo infectado no es detectado.

Fedora: Es una distribución de Linux, desarrollada por la comunidad que soporta al *Proyecto Fedora* y es patrocinado por *Red Hat.* Es un completo sistema operativo de propósito general, que contiene sólo software libre y de código abierto.

Fingerprinting: En el contexto de la seguridad informática en general, refiere a las de diferentes técnicas mediante las cuales es posible identificar con mayor o menor grado de certeza, un dispositivo, host, sistema operativo, etc.

GGHH: *Google Hack Honeypot.* Es un *honeypot* diseñado para detectar a los atacantes que utilizan los motores de búsqueda como herramienta de hacking contra diferentes recursos.

GPL: *GNU GPL* es una *Licencia Pública General*, licencia creada por la Free Software Fundation a mediados del 80, orientada principalmente a proteger la libre distribución, modificación y uso de software.

HIDS: Host IDS. IDS que vigila un solo host.

HoneyCell: Producto entregable del proyecto de grado. Contiene el componente de captura de datos (Honeyd o Honeytrap), componente para la normalización de los datos capturados y realiza el envío, en tiempo real, de los datos obtenidos al *HoneyCell* vía SSL/TLS.

HoneCenter: Producto entregable del proyecto de grado. Los datos capturados por los *HoneyCell* son almacenados de forma centralizada por *HoneyCenter.* Además contiene un sistema de monitoreo de datos, un sistema e alarmas, un sistema de control de tráfico, y un firewall que filtra y registra los accesos al dispositivo.

Honeyd: Es un *honeypot* de bajo nivel de interacción, permite virtualizar una *honeynet* relativamente compleja con un grado de realismo muy elevado sin necesidad de utilizar costosos recursos de hardware.

Honeynet: Es una red de *honeypots* que puede incluir otros componentes para la captura de datos, diseñada principalmente para la investigación. Se debe destacar que es una arquitectura no un producto concreto. El objetivo principal es obtener la mayor cantidad de información de los ataques a una red real.

Honeynet Project: Es una organización internacional fundada en 1999, sin fines de lucro dedicada a la investigación para la mejora de la seguridad de Internet.

Honeypot: Un *honeypot* es un recurso que pretende ser un objetivo real. Se espera de un *honeypot* que sea atacado o comprometido. Su meta principal es la distracción de los atacantes y obtener información sobre los ataques y atacantes."

Honeytrap: Es un *honeypot* de bajo nivel de interacción.

Honeywall: Es un conjunto de herramientas Open Source, configuradas por el *Honeynet Project*, que permiten implantar un *gateway* para una *honeynet* de generación II, de manera sencilla y segura. Su distribución es por medio de un CD configurable que contiene un Linux minimal.

host-only: Es un modo de conexión de una maquina virtual sobre *VMWare* y el pc real en el que se crea una red privada entre ambos.

IDMEF: Intrusion Detection Message Exchange Format. Estándar de formato de mensajes de alertas de sistemas de detección de intrusos. Actualmente se encuentra en estado RFC.

IDXP : Intrusion Detection eXchange Protocol. Protocolo de intercambio de mensajes de alertas de IDS, definido por *IDWG*.

IDS: Intrusion Detection System. Sistema que detecta manipulaciones no deseadas en un sistema o dispositivo, que no pueden ser detectados por un firewall convencional. Están compuestos por sensores, sistema de monitoreo de eventos, sistema de alertas y un sistema de almacenamiento de eventos que es utilizado para generar las alertas de seguridad.

IDWG: Intrusion Detection Working Group. Grupo de trabajo perteneciente a *IETF*.

IETF: The Internet Engineering Task Force. Es una organización sin fines de lucro fundada en 1992 para proporcionar liderazgo en relación a estándares, educación y políticas referente a Internet.

Intrusión: Es cuando un usuario no autorizado ingresa a un sistema ya sea en forma local o remota.

Iptables: Es un sistema de firewall que esta integrado con el kernel de linux. Los comandos iptables se utilizan para crear las reglas de aceptación, denegación, etc. de paquetes.

LaBrea Tarpit: *Honeypot* open source creado por Tom Liston, diseñado para enlentecer o detener los ataques.

Libprelude: Es la biblioteca sobre la que se basa el framework de *Prelude-IDS*, para normalizar las alertas y la comunicación en el estándar *IDMEF*.

libsmtp++: Librería escrita en C++ que proporciona funcionalidades para el envío de mail sobre protocolo SMTP.

LiveCD: Es un sistema operativo y un conjunto de herramientas que se encuentran almacenados y pueden ser ejecutados desde un medio extraíble (CD, DVD, etc.). En este tipo de distribuciones se crea un disco virtual haciendo uso de la memoria RAM del equipo en el cual es ejecutado, una de sus ventajas es que en general no se efectúan cambios en la maquina (o disco duro) real utilizada a menos que se deseen guardar las preferencias de ejecución.

Log: Es un registro oficial durante un periodo de tiempo en particular de eventos generados por sistemas y dispositivos. Para los profesionales en seguridad informática un *log* es usado para registrar datos o información sobre quien, que, cuando, donde y por que (*who*, *what*, *when*, *where* y *why*, W5) ocurre un evento.

Malware: Del ingles malicious software, también llamado badware o software malicioso, es un software que tiene como objetivo infiltrarse en o dañar una computadora sin el conocimiento de su dueño y con finalidades muy diversas ya que en esta categoría encontramos desde un troyano hasta un spyware. Y si bien puede instalarse a través de algún correo electrónico, lo puede hacer a través de una página falsa que se enlaza desde el buscador.

Máquina virtual: Es un software que crea un entorno virtual entre el sistema operativo de una computadora y el usuario, permitiendo la ejecución de software arbitrario en este, emula el comportamiento de una máquina real.

Martian source: Se produce un "martian source" cuando el kernel recibe un paquete erróneo en lo referente al enrutamiento, por ejemplo una IP destino o una IP de origen, que es incorrecta, por lo que no debe ser retransmitida.

Md5: Es un algoritmo de criptografía muy difundido, desarrollado como reemplazo del algoritmo MD4. Sus aplicaciones varían desde la encriptación de archivo, la firma de archivos, etc.

MySQL: Sistema de gestión de base de datos relacional, bajo licencia GNU GPL.

NIDS: (Network IDS, IDS que se basa en la red, detectando ataques que se hacen en la misma.

nmap: Programa que sirve para efectuar port scanning. Esta orientado a la identificación de puertos abiertos en una computadora objetivo, determinando que servicio/s esta ejecutando la misma, e intenta determinar que sistema operativo utiliza dicha computadora.

Open Reports: Es una herramienta Web de reportes, es fácil de usar. Soporta una variedad de motores de reportes tales como JasperReports, JFreeReport, JXLS, y Eclipse BIRT. OpenReports provee una interfase Web para la visualización de los reportes, así como la creación de nuevos reportes, edición de reportes existentes y administración de usuarios, conexiones a base de datos, filtros dinámicos, etc.

Open source: El software open source se define por la licencia que lo acompaña, que garantiza a cualquier persona el derecho de usar, modificar y redistribuir el código libremente.

OSSIM: Open Source Security Information Management. Es una distribución de productos open source integrados para construir una infraestructura de monitoreo de seguridad. Su objetivo es ofrecer un marco para centralizar, organizar y mejorar las capacidades de detección y visibilidad en el monitoreo de eventos de seguridad de la red.

PHP.Hop: *Honeypot* de bajo nivel de interacción que provee una interfaz Web falsa y así detectar una gran variedad de ataques Web.

Port forwarding: Es el reenvío de puertos de red de un dispositivo de red a otro.

Port scanning: Método de intrusión. Analiza el estado de los puertos de un host conectado a una red y detecta si un puerto está abierto, cerrado, o protegido por un firewall. Se utiliza para detectar posibles vulnerabilidades de seguridad.

Postgres: Es un servidor de base de datos relacional orientada a objetos de software libre, liberado bajo la licencia BSD.

Prelude IDS Framework: Framework basado en *IDMEF*, para la extensión de un IDS híbrido.

Prelude Manager: Es el corazón del *framework Prelude-IDS*, proporciona estructuras de almacenamiento para la información recolectada de la red compuesta por sensores heterogéneos. Recibe los mensajes a través de conexiones SSL/TLS, texto plano o conexiones Unix, para posteriormente procesarlos.

Prelude-LML: *Prelude Log Monitoring Lackey*. Es el componente del framework de *Prelude-IDS* que se ocupa de los aspectos de la detección de intrusión a nivel de host. Una de sus funciones es monitorear archivos creados por *Syslogs* provenientes de host de diferentes plataformas (sistemas unix, switches, routers, firewalls, impresoras, etc), u otro tipo de *logs* del estilo single-event, pudiendo también simular un servidor *syslog* por su cuenta.

RFC: Request for Comments. Son documentos publicados de forma gratuita que se especifican, entre otros los protocolos oficiales de la IESG (Internet Engieneering Stering Group), la IAB (Interne Architecture Board) y la comunidad de Internet.

Roo: Nombre de la versión actual del *Honeywall*.

rpm: Red Hat Package Manager. Es una herramienta de administración de paquetes originalmente desarrollado para *Red Hat* Linux. En la actualidad es usada por muchas distribuciones como ser *Fedora*, *Mandriva*, *SuSE* y *Conectiva*.

SDS: Symantec Decoy Server. *Honeypot* que permite virtualizar, hasta cuatro sistemas operativos

Sebek: Es una herramienta para capturar actividad en el espacio del kernel, puede capturar pulsaciones de teclas, recuperar contraseñas, y monitorizar cualquier comunicación.

SEC: Simple Event Correlador. Herramienta correlación de eventos que utiliza reglas para el procesamiento de eventos. Realiza monitoreo de archivos de *logs* en tiempo real y en el caso de que un evento cumpla con el patrón de regla, se especifica una lista de acciones a realizarse.

Sensor: Es una entidad que detecta eventos de seguridad.

Sniffer: Escuchan y capturan el tráfico de una red.

Snort: Es una aplicación de escucha de paquetes de una red (sniffer) permitiendo la detección de intrusos en la misma.

Snort_inline: Es una modificación de Snort la cual permite modificar los paquetes capturados a partir de un conjunto de reglas.

Specter: *Honeypot* comercial de bajo nivel de interacción creado por NetSec. Funciona en sistemas Windows.

MSSQL Server: Sistema de gestión de base de datos relacional de Microsoft.

Swatch: Es una herramienta orientada al monitoreo de archivos de *logs* de sistemas Unix.

Syslog: Es un sistema de *logs* que se encarga capturar *logs* generados por eventos del sistema, y realizar el almacenamiento o reenvío de estos.

Syslog-ng: Es una sustitución del código fuente abierto de `syslogd`, incorporando nuevas funcionalidades tales como la admisión de trafico `TCP`, mejoría en el sistema de filtrado, etc.

Tomcat: Es un servidor Web con soporte de servlets y JSPs.

Troyano: Se denomina troyano (o caballo de Troya, traducción fiel del inglés Trojan horse) a un programa malicioso capaz de alojarse en computadoras y permitir el acceso a usuarios externos, a través de una red local o de Internet, con el fin de recabar información o controlar remotamente a la máquina anfitriona sin ser advertido, normalmente bajo una apariencia inocua. Al contrario que un virus, que es un huésped destructivo, el troyano no necesariamente provoca daños porque no es su objetivo.

Tunneling: Es una tecnología que permite enviar datos en una red mediante otras conexiones de la red. El tunneling funciona encapsulando el protocolo de red dentro de paquetes transportados por la segunda red.

UML: User Mode Linux. Modificación del kernel del sistema operativo para que trabaje sobre su propia interfaz de llamadas al sistema. Mediante este sistema un kernel puede operar sobre otro como un proceso de usuario. Es un modo de virtualización.

Virtualización: Se refiere a la abstracción de los recursos de una computadora. Se puede dividir en dos categorías: Virtualización de plataforma que involucra la simulación de máquinas virtuales y virtualización de recursos que involucra la simulación de recursos combinados, fragmentados o simples.

VMnet: Se denomina así a una red totalmente virtualizada por la aplicación *VMWare Server*.

VMUI: VMware Management User Interface, consola de administración para la infraestructura *VMware Server* mediante una interfaz Web.

VMware: Es una firma dedicada al desarrollo de máquinas virtuales, se trata de una tecnología que permite correr varios sistemas operativos al mismo tiempo sobre una misma computadora.

VMware Server: Producto de software de la empresa VMware Inc. Su primera versión fue lanzada el 12 de julio de 2006. *VMware Server* puede crear, editar y ejecutar máquinas virtuales. Emplea un modelo cliente/servidor, permitiendo así el acceso remoto a máquinas virtuales creadas por otros productos de *VMware* o de terceros.

VMware Server Console: Cliente instalado en forma local que provee mecanismos de administración remota sobre el host donde se alojan las maquinas virtuales mediante la aplicación *VMware Server*.

Walleye: Interfaz Web que permite el análisis de datos registrados por Honeywall en forma rápida y sencilla.

yum: Yellow dog Updater, Modified. Es una herramienta opensource de gestión de paquetes para sistemas Linux basados en RPM. sistemas Linux basados en RPM.

REFERENCIAS

[ArqI08] CÓCARO, Fernando y GARCÍA, Mauricio y ROUILLER, María José. Arquitectura de Implantación. [DVD]. 2.0. Montevideo. 19 de Enero de 2008 .

[Alle01] ALLEN, Stewart. Importance of Understanding Logs from an Information Security Standpoint. SANS Institute. [online]. 2001. Disponible en: http://www.sans.org/rr/papers/33/200.pdf . [17 de enero de 2008 última fecha de acceso]

[Atki95] E. Todd Atkins. Swatch. [online]. 13 de Diciembre de 1995. Disponible en: http://www.cs.vassar.edu/cgi-bin/man2html?swatch+8#lbAC. [19 de enero de 2008 última fecha de acceso]

[BaPl02] BAUMANN, Reto y PLATTNER, Christian. White Paper: Honeypots. [online]. Disponible en: http://www.megasecurity.org/papers/whitepaper.pdf. 26 de Febrero de 2002. [15 de Diciembre de 2007 última fecha de acceso]

[BDDT07] bddt.exe - bddt - BDDT - Spyware Check. [online]. 2007. Disponible en: http://spyware.adslnet.es/genera.php?processfile=bddt.exe&dir=b&pag=28 [17 de Abril de 2008 ultima fecha de acceso].

[Benj03] BENJURRY. Optimize NFR.Part 1-MSSQL Hello Buffer Overflow. XFocus Team. [online]. 10 de Octubre de 2003. Disponible en: http://www.xfocus.org/documents/200308/3.html. [1ro. de Abril de 2008 última fecha de acceso]

[BOF08] CÓCARO, Fernando y GARCÍA, Mauricio y ROUILLER, María José. Back Officer Friendly. [DVD]. 2.0. Montevideo. 26 de Enero de 2008.

[Cacti04] CACTI. [online]. 2004. Disponible en: http://www.cacti.net/. [30 de Noviembre de 2007 última fecha de acceso]

[CDCo08] Cult of the Dead Cow. [online]. 2008. Disponible en: http://www.cultdeadcow.com/. [26 de Eenero de 2007 última fecha de acceso]

[Cent07] CENTOS Project. CentOS. [online]. 5.0. [s.l.]. Disponible en: http://mirror.centos.org/centos/5/isos/. [30 de Noviembre de 2007 última fecha de acceso]

[Ches94] CHESWICK, Bill. An Evening with Berferd, In Which a Cracker is Lured, Endured, and Studied. [online]. 1994. Disponible en: http://www.cheswick.com/ches/papers/berferd.pdf. [3 de Marzo de 2008]

[CIDF99] Common Intrusion Detection Framework. [online]. 10 de Septiembre de 1999. Disponible en: http://gost.isi.edu/cidf/. [26 de Eenero de 2007 última fecha de acceso]

[Cisc08] Cisco. [online]. 2008. Disponible en: http://www.cisco.com/. [26 de Eenero de 2007 última fecha de acceso]

[Clif95] STOLL, Clifford. The Cuckoo's Egg, Tracking a Spy Through the Maze of Computer Espionage. [s.l.]. Pocket Books. 1995. ISBN: 0671726889

[Cohe97] COHEN, Fred. Deception Toolkit. [online]. 18 de Agosto de 1999. Disponible en: http://all.net/dtk/v1999-08-18.html [17 de Abril de 2008 última fecha de acceso]

[CSIR08] CSIRT Development. CERT. [online]. 2008. http://www.cert.org/csirts/. [26 de Eenero de 2007 última fecha de acceso]

[Digi07] Symantec. Digispid.B.Worm. [online]. 13 de Febrero de 2007. Disponible en: http://www.symantec.com/security_response/writeup.jsp?docid=2002-052108-5430-99

[DiJo04] Dieter Joho. Active Honeypots. [online]. 16 de Diciembre de 2004. Disponible en: http://www.megasecurity.org/papers/Joho_Dieter.pdf [18 de Abril de 2008 última fecha de acceso].

[Dona02] DONALDSON, Mark E. Inside the Buffer Overflow Attack: Mechanism, Method, & Prevention. [online]. 3 de Abril de 2002. 1.3. Disponible en: http://www.sans.org/reading_room/whitepapers/securecode/386.php. [29 de Marzo de 2008 última fecha de acceso]

[EALo08] CÓCARO, Fernando y GARCÍA, Mauricio y ROUILLER, María José. Estado del Arte de Logs. [DVD]. 2.0. Montevideo. 26 de Enero de 2008.

[EsAr08] CÓCARO, Fernando y GARCÍA, Mauricio y ROUILLER, María José. Estado del Arte. [DVD]. 2.0. Montevideo. 21 de Enero de 2008.

[Fedo07] Fedora. Red Hat Inc. [online]. 7.0. [s.l.]. Disponible en: http://fedoraproject.org. [29 de Marzo de 2008 última fecha de acceso]

[Feie99] FEIERTAG, Rich [et al]. A Common Intrusion Specification Language (CISL).[online]. 11 de Junio de 1999. [30 de Marzo de 2008 última fecha de acceso]. Disponible en: http://gost.isi.edu/cidf/drafts/language.txt

[FING08] Facultad de Ingenería, Universidad de la República, Montevideo - Uruguay. [online]. 2008. Disponible en: http://www.fing.edu.uy/. [14 de Abril de 2008 última fecha de acceso]

[FlaR95] Flannery,Ryan. PhpSymon. [online]. 1995. Disponible en: http://www.ryanflannery.net/works/phpsymon/. [30 de Julio de 2007 última fecha de acceso]

[GHH05] The Google Hack Honeypot. [online]. 2005. Disponible en: http://ghh.sourceforge.net/. [29 de Marzo de 2008 última fecha de acceso]

[GNUP07] Free Software Foundation, Inc. GNU General Public License. [online]. 3. 29 de Enero del 2007. Disponible en: http://tools.ietf.org/html/rfc4765. [3 de Marzo de 2008 última fecha de acceso]

[Goog08] Google. 2008. [online]. Disponible en: http://www.google.com. [3 de Marzo de 2008 última fecha de acceso]

[GSI06] GSI. [online]. 2006. Disponible en: http://www.fing.edu.uy/inco/grupos/gsi/. [14 de abril de 2008 última fecha de acceso]

[Hond05] Developments of the Honeyd Virtual Honeypot. Niels Provos. [online]. 2005. Disponible en: http://www.honeyd.org. [3 de Marzo de 2008 última fecha de acceso]

[Hont05] Honeytrap. mwcollect Alliance. [online]. 2005. Disponible en: http://honeytrap.mwcollect.org/. [14 de Abril de 2008 última fecha de acceso]

[HoPr02] The Honeynet Project. [online]. 2002. Disponible en: http://www.honeynet.org. [14 de Abril de 2008 última fecha de acceso]

[Howl08] CÓCARO, Fernando y GARCÍA, Mauricio y ROUILLER, María José. Honeywall. [DVD]. 2.0. Montevideo. 21 de Enero de 2008.

[HPBr02] The Honeynet.BR. [online]. 2002. Disponible en: http://www.honeynet.org.br/. [14 de Abril de 2008 última fecha de acceso]

[HPDi07] Distributed Honeynets Project. [online]. 2007. Disponoble en: http://distributed.honeynets.org/main.ph. [14 de Abril de 2008 última fecha de acceso]

[HPHW07] HONEYNET Project. Honeywall CDROM. [online]. Roo 1.2. [s.l.]. Disponible en: http://www.honeynet.org/tools/cdrom/roo/download.html. [14 de Abril de 2008 última fecha de acceso]

[HPRA05] Reseach Alliance. [online]. 2005. Disponible en: http://www.honeynet.org/alliance/. [14 de Abril de 2008 última fecha de acceso]

[HPSe05] HONEYNET PROJECT. Sebek. [online]. 3.2. Disponible en: http://www.honeynet.org/tools/sebek/

[ICSA04] ICSA LABS IDS CONSORTIUM. ICSA Labs IDS Consortium Announces Network Intrusion Detection System Alert Specification Format. Business Wire. [online]. 23 de Febrero de 2004. E.E.U.U . [30 de Marzo de 2008]. Disponible en: http://www.allbu

[ICSA07] ICSA Labs. [online]. 2007. Disponible en: http://www.icsalabs.com/icsa/icsahome.php

[IDWG05] Intrusion Detection Exchange Format (idwg). [online]. 26 de Enero de 2005. Disponible en: http://www.ietf.org/html.charters/OLD/idwg-charter.html. [3 de Marzo de 2008 última fecha de acceso]

[IETF08] The Internet Engineering Task Force. [online]. 2088. Disponible en: http://www.ietf.org/. [3 de Marzo de 2008 última fecha de acceso]

[IISS07] IBM Internet Securuty System. [online]. 1ro. Octubre de 2007. Disponible en: http://xforce.iss.net/xforce/alerts/id/advise101. [3 de Marzo de 2008 última fecha de acceso]

[INCO08] Instituto de Computación. [online]. 2008. Disponible en: http://www.fing.edu.uy/inco/pm/field.php?n=Main.HomePage. [14 de Abril de 2008 última fecha de acceso]

[JFFN06] JFFNMS: Meet your network. [online]. 2006. Disponible en: http://www.jffnms.org/. [3 de Marzo de 2008 última fecha de acceso]

[KyEn00] HONEYNET Project. Know Your Enemy. [online]. Disponible en: http://www.honeynet.org/papers/kye.html. [3 de Marzo de 2008 última fecha de acceso]

[LbrT03] LORGOR - LISTON, Tom. LaBrea. [online]. 2003. Disponible en: http://labrea.sourceforge.net/. [3 de

Marzo de 2008 última fecha de acceso]

[LiCD04] The LiveCD List. [online]. 2004. Disponible en: http://www.livecdlist.com/. [1ro. de Febrero de 2008 última fecha de acceso]

[Live08] CÓCARO, Fernando y GARCÍA, Mauricio y ROUILLER, María José. LiveCDTools. [DVD]. 2.0. Montevideo. 21 de Enero de 2008.

[LkLg06] LINKLOGGER. PortPeeker Capture of Sasser Scan and Infection Attempt. [online]. 2006.. Disponible en: http://www.linklogger.com/sasser.htm. [1ro. de Abril de 2008 última fecha de acceso]

[LoBr08] LÓPEZ Bravo, Cristina. Seguridad en Internet. GRIS: Grupo de Redes e Ingeniería del Software. [online]. 2008. [30 de Marzo de 2008 última fecha de acceso]. Disponible en: http://www-gris.det.uvigo.es/wiki/pub/Main/PaginaNST/seguridad-Clase1-NSTx.pdf

[LWAT07] LogWatch.org. LogWatch. [online]. Febrero de 2007. Disponible en: http://www.logwatch.org/. [17 de enero de 2008 última fecha de acceso]

[MD508] MD5 Information. [online]. 2008. Disponible en: http://www.bullzip.com/md5/md5.htm. [5. de Diciembre de 2008 última fecha de acceso]

[MiTN04] MICROSOFT TechNet. Alerta de Seguridad- Nuevo Gusano Sasser. [online]. 1ro. de Mayo de 2004. Disponible en: http://www.microsoft.com/spain/technet/seguridad/alertas/sasser.mspx. [1ro. de Abril de 2008 última fecha de acceso]

[MoDa08] CÓCARO, Fernando y GARCÍA, Mauricio y ROUILLER, María José. Modelo de Datos. [DVD]. 2.0. Montevideo. 26 de Enero de 2008.

[Mozi02] Mozilla. Firefox. [online]. 2004. Disponible en:http://www.mozilla.com/en-US/firefox/. [01 de Febrero de 2008 última fecha de acceso]

[Msft07] MICROSOFT. Descripción de la herramienta Dr. Watson (Drwatson.exe). [online]. 29 de Octubre de 2007. 1.2. [29 de Marzo dfe 2008]. Disponible en: http://support.microsoft.com/kb/308538/es

[MySq95] MySQL AB. MySql. [online]. 1995. Disponible en: http://mysql.com/ [20 de Diciembre de 2008 última fecha de acceso]

[Netf99] WELTE, Harald y NEIRA, Pablo. Netfilter. [online]. 1999. Disponible en: http://www.netfilter.org/. [30 de Marzo de 2008 última fecha de acceso]

[Nmap08] Nmap Free Security Scanner. [online]. 2008. Disponible en: http://nmap.org/. [14 de Abril de 2008 última fecha de acceso]

[NWGp07] Network Working Group. The Intrusion Detection Message Exchange Format (IDMEF). [online]. Marzo del 2007. Disponible en: http://tools.ietf.org/html/rfc4765. [23 de enero de 2008 última fecha de acceso]

[NWGr90] Network Working Group. A Simple Network Management Protocol (SNMP). Disponible en: http://tools.ietf.org/html/rfc1157. [online]. Mayo de 1990. [30 de Marzo de 2008 última fecha de acceso]

[NWGX07] Network Working Group. The Intrusion Detection Exchange Protocol (IDXP). [online]. Marzo de 2007. Disponible en: http://www.ietf.org/rfc/rfc4767.txt?number=4767. [23 de Enero de 2008 última fecha de acceso]

[OpnR08] CÓCARO, Fernando y GARCÍA, Mauricio y ROUILLER, María José. OpenReports. [DVD]. 2.0. Montevideo. 21 de Enero de 2008.

[OpRe08] Open Reports. [online]. 2008. Disponible en: http://oreports.com/. [14 de Abril de 2008 última fecha de acceso]

[PCRE08] PCRE - Perl Compatible Regular Expressions. [online]. 2008. Disponible en: http://www.pcre.org/. [14 de Abril de 2008 última fecha de acceso]

[Pent08] Pentaho open source business Intelligence. [online]. 2008. Disponible en: http://www.pentaho.com/. [29 de Febrero de 2008 última fecha de acceso]

[Perl02] Perl FoundationSite Information and Contacts. Perl. [online]. 2002. Disponible en: http://www.perl.org/. [29 de Febrero de 2008 última fecha de acceso]

[PHP06] PHP.Hop - PHP Honeypot Project. [online]. 2006. Disponible en: http://www.rstack.org/phphop/. [29 de Febrero de 2008 última fecha de acceso].

[PhpD98] PhpMyAdmin Devel Team. PhpMyAdmin. [online]. 1998. Disponible en: http://es.wikipedia.org/wiki/PhpMyAdmin. [29 de Febrero de 2008 última fecha de acceso]

[Posg08] PosgreSQL. [online]. 2008. Disponible en: http://www.postgresql.org/. [29 de Febrero de 2008 última fecha de acceso]

[Prel04] Prelude IDS Framework. Prelude. [online]. 2004. Disponible en: http://www.prelude-ids.org. [29 de Febrero de 2008 última fecha de acceso]

[Prel08] CÓCARO, Fernando y GARCÍA, Mauricio y ROUILLER, María José. Prelude. [DVD]. 2.0. Montevideo 17 de Enero de 2008.

[Prop08] CÓCARO, Fernando y GARCÍA, Mauricio y ROUILLER, María José. Propuesta. [DVD]. 2.0. Montevideo. 26 de Enero de 2008.

[ReaV02] RealVNC. VNC. [online]. 2002. Disponible en: http://www.realvnc.com/index.html. [03 de Agosto de 2007 última fecha de acceso]

[RedH08] RED Hat Inc. Red Hat Enterprise Linux. 5. [s.l.]. Disponible en: http://www.europe.redhat.com/rhel/

[ReEs08] CÓCARO, Fernando y GARCÍA, Mauricio y ROUILLER, María José. Resportes Estadisticos. [DVD]. 2.0. Montevideo. 21 de Enero de 2008.

[SEC08] CÓCARO, Fernando y GARCÍA, Mauricio y ROUILLER, María José. Simple Event Correlator. [DVD]. 2.0. Montevideo. 17 de Enero de 2008.

[Smail04] Secunia Stay Secure. SquirrelMail Multiple Vulnerabilities. [online]. 4 de Abril de 2004. Disponible en:

http://secunia.com/advisories/18985/

[SnIl08] Snort InLine. [online]. 2008. Disponible en: http://snort-inline.sourceforge.net/. [29 de Febrero de 2008 última fecha de acceso]

[Snor08] Snort.Org. [online]. 2008. Disponible en: http://www.snort.org/. [29 de Febrero de 2008 última fecha de acceso]

[Spec08] CÓCARO, Fernando y GARCÍA, Mauricio y ROUILLER, María José. Specter. [DVD]. 2.0. Montevideo. 21 de Enero de 2008.

[Spec97] Specter Intrusion Detection System. [online]. 1997. Disponible en: http://www.specter.com. [1ro. de Febrero de 2008 última fecha de acceso]

[Spit01] SPITZNER, Lance. The Value of Honeypots, Part One: Definitions and Values of Honeypots. [online]. 10 de Octubre de 2001. [30 de Marzo de 2008 última fecha de acceso]. Disponible en: http://www.securityfocus.com/infocus/1492

[SQLi08] SQLite. [online]. 2008. Disponible en: http://www.sqlite.org/. [29 de Febrero de 2008 última fecha de acceso]

[SRMS08] Security Risk Management Solutions: Net Vigilance. [online]. 2008. Disponible en: http://www.netvigilance.com/winhoneyd. [29 de Febrero de 2008 última fecha de acceso]

[SunT07] TZU, Sun. El arte de la guerra. 6ta. Edición. Madrid. Editorial Trotta. 2007. ISBN 8481644927.

[Syma95] Symantec Corporation. [online]. 1995. Disponible en: http://www.symantec.com. [29 de Febrero de 2008 última fecha de acceso]

[TCPD07] Tcpdump/libpcap. [online]. 2007. Disponible en: http://www.tcpdump.org/. [29 de Febrero de 2008 última fecha de acceso]

[Timo03] libsmtp++. Timo Benk. [online]. 0.1. [s.l.]. Disponible en: http://sourceforge.net/project/showfiles.php?group_id=71829&package_id=98957

[Tgno99] The GNOME Project. Gnome. [online].1999. Disponible en: http://www.gnome.org/. [30 de Octubre de 2008 última fecha de acceso]

[TLSe07] Transport Layer Security (tls). [online]. 1ro. de Octubre de 2007. Disponible en: http://www.ietf.org/html.charters/tls-charter.html. [29 de Febrero de 2008 última fecha de acceso]

[Tomc07] Apache Tomcat. [online]. 2007. Disponible en: http://tomcat.apache.org/. [29 de Febrero de 2008 última fecha de acceso]

[Trip08] Tripwire Configuration Audit & Control. [online]. 2008. Disponible en: http://www.tripwire.com/. [29 de Febrero de 2008 última fecha de acceso]

[UNAM06] Proeycto Honeynet UNAM. Estadísticas del Proyecto Honeynet de la UNAM. [online]. 2006. Disponible en: http://www.honeynet.unam.mx/es/stats.pl. [29 de Febrero de 2008 última fecha de acceso]

[Vaar08] VAARANDI, Risto. SEC – Simple Event Correlator. [online] Disponible en: http://simple-evcorr.sourceforge.net. [19 de enero de 2008 última fecha de acceso].

[VMSC08] VMWARE Inc. VMware GSX Server 3.2. [online]. 2008. [30 de Marzo de 2008 última fecha de acceso]. Disponible en: http://www.vmware.com/support/gsx3/doc/manage_console_upgrade_gsx.html

[Vmwa08] VMWare. [online]. 2008. Disponible en: http://www.vmware.com/. [29 de Febrero de 2008 última fecha de acceso]

[W32C07] Symantec. W32.Cblade.Worm. [online]. 13 de Febrero de 2007. Disponible en: http://www.symantec.com/security_response/writeup.jsp?docid=2001-112113-4951-99&tabid=1. [29 de Febrero de 2008 última fecha de acceso]

[WinX08] Windows XP. [online]. 2008. http://www.microsoft.com/spain/windowsxp/home/default.mspx. [29 de Febrero de 2008 última fecha de acceso]

[Xpro08] Xprobe Active OS Fingerprinting too 2. [online]. 2008. Disponible en: http://xprobe.sourceforge.net/. [29 de Febrero de 2008 última fecha de acceso]

[Xtra08] Symantec. POP3 Xtramail Multiple DoS. [online]. 2008. Disponible en: http://www.symantec.com/avcenter/attack_sigs/s21108.html. [3 de Marzo de 2008 última fecha de acceso]