PROPUESTA DE UNA ARQUITECTURA DE SISTEMAS DE DETECCIÓN DE INTRUSOS CON CORRELACIÓN

Escola **T**ècnica **S**uperior d'**E**nginyeria
Universidad de Valencia

Ángel Alonso Párrizas
Director: Santiago Felici Castell

7 de noviembre de 2005

A María Dolores.

Agradecimientos

A Santiago, mi director del proyecto, gracias por haber confiado en éste proyecto y haber sido un referente como persona y tutor.

A mi madre María Dolores, por haber conseguido que llegara tan lejos y haberme dado la oportunidad de realizar unos estudios, a ti te lo debo todo.

A todos los compañeros de clase, Ramón, Ángel, Edu, Timoteo, Michel, María, Agustín y todos aquellos que se me olvidan.

A mi familia, mis tíos, mi primo Rafa, a mi abuela Dolores y a mi hermana Alba por haberse interesado en todo momento por mis estudios. Y a la pequeña de la casa, Marina.

Y como no a mi novia María, por la paciencia mostrada durante estos últimos meses y su contribución a este proyecto.

Resumen

Las redes de ordenadores pueden presentar vulnerabilidades difíciles de proteger, dada la heterogeneidad de equipos, sistemas operativos, servicios (aplicaciones) y usuarios.

Las vulnerabilidades conocidas son aprovechas por los atacantes para diferentes fines. Dado que el tipo de ataque utilizado puede ser de lo más diverso, desde denegación de servicio, usurpación/alteración de información, suplantación, etc, en la comunidad científica se han desarrollado y diseñado diferentes implementaciones de sensores o detectores de intrusos (IDSs) con técnicas diversas, con la finalidad de acaparar el máximo número de comportamientos ilícitos.

Como contrapartida a un mayor volumen de información generado por la proliferación de IDSes, un mayor número de alertas son generadas. El efecto colateral que conlleva la diversidad de IDSs, es el incremento de alertas, implicando un aumento de falsos positivos (alertas que no son ataques reales) e indirectamente, un aumento de los falsos negativos (ataques que no son considerados) que impide en cierta manera, el objetivo final de los IDSs, que es la detección de intrusos.

La técnica más utilizada para minimizar este efecto colateral es aplicar correlación sobre las alertas, con el fin de resumir la información que finalmente se le presenta al administrador de la red.

La propuesta realizada en este proyecto ha sido el despliegue de una red de diferentes sensores, los cuales hemos correlado utilizando herramientas de libre distribución.

La idea correlación ha consistido en generar 'metaalertas', producidas tras ciertas secuencias de alertas determinadas de diferentes sistemas de detección de intrusos y herramientas de seguridad.

Es decir, un mismo ataque es capaz de generar diferentes alertas, pero el ataque es uno y único. Si somos capaces de detectar un ataque fiable y real, con intrusión fidedigna y generar una única alarma, habremos cumplido con el objetivo.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Problemática

Debido al desarrollo de las nuevas tecnologías e Internet, tanto el número de ataques a redes como la variedad de éstos ha aumentado.
Los ataques siempre buscan deficiencias en el software o en la configuración del mismo, lo que se conoce como vulnerabilidad, con el fin de ser aprovechadas (o 'explotados') y conseguir algún efecto sobre el sistema.

Firewalls, Proxys inversos, Sistemas de Detección de Intrusos (IDS) son algunas de las herramientas que utilizan los administradores de sistemas y responsables de seguridad para evitar que las vulnerabilidades sean aprovechadas por los atacantes y evitar los ataques. Aunque éstas herramientas no son la solución final sí pueden ser consideradas, en buena medida, una barrera para los intrusos.

Gracias al desarrollo de herramientas de monitorización y detección, como los IDSs, hoy en día se dispone de mucha información para ver qué sucede en un momento determinado en una red o, a posteriori analizar lo que ha sucedido. Dicha información se puede obtener a través de un amplio abanico de herramientas, desde los logs de los propios sistemas operativos hasta herramientas específicas para monitorizar los paquetes que atraviesan una red, pasando por aplicaciones creadas para controlar qué sucede en un host. Pero surgen diversos problemas como consecuencia de la gran cantidad de información disponible, como pueden ser la gestión de tantos eventos producidos por las herramientas de seguridad o el riesgo que existe de obtener demasiadas falsas alarmas. Es aquí donde la correlación juega un papel fundamental: analizar los eventos provenientes de distintos sensores o monitores

con el propósito de reducir el número de falsas alertas, poder reconstruir el hilo de ejecución de un ataque, o ver el impacto real sobre la red de un ataque.

En este proyecto se propone un modelo de correlación basado en distintas fases, empezando por una primera fase de recolección de las alertas, hasta un fase final donde se resumen las consecuencias directas del ataque. El conjunto de herramientas con las que se trabajan varían dependiendo de su función y su uso, siendo los IDS o Sistemas de Detección de Intrusos la principal fuente de información, aunque no la única.

1.2. Motivación

Este proyecto está estrechamente relacionado con el trabajo de investigación Elementos Activos de Seguridad [1] (ELAS) que realiza el Instituto de robótica de la Universitat de València junto con el Centro de Comunicaciones del CSIC [2] (Consejo Superior de Investigaciones Científicas) conocido como Rediris [3]. El objetivo de éste fue la instalación y configuración de distintos detectores de intrusos con el fin de aumentar la fiabilidad de las diferentes alertas.

Por otro lado, se han realizado varios proyectos finales de carrera sobre temática similar en la Universitat de València, como la instalación de un sistema de detección de intrusos en la antigua troncal ATM [4] donde los resultados finales mostraban un número muy elevado de falsos positivos, u otros proyectos de análisis forense sobre sistemas Linux [5] o Windows [6] donde se generaban muchos falsos positivos en los detectores de red.

1.3. Objetivos del proyecto

El objetivo principal es correlar la información de distintos sistemas de seguridad (detectores de intrusos, monitores de red, auditores de red, etc) con el fin de reducir el número de falsas alarmas.

De forma más específica, se pretende en este proyecto:

- Estudiar y evaluar las prestaciones de diferentes sensores y posibles arquitecturas.
- Integrar diferentes sensores de libre distribución.
- Implementar diferentes reglas de correlación.

- Realizar una arquitectura general y abierta de correlación de alertas y contramedidas.
- Incluir distintas herramientas de seguridad: auditores de sistemas y análizadores de factor de riesgo.

Capítulo 2

Estado del Arte

En este capítulo se explicarán los conceptos sobre los que trata el proyecto, los cuales servirán para comprender el funcionamiento de las herramientas que se van usar.

2.1. Sistemas de detección de intrusos (IDS)

Un Sistema de Detección de Intrusos (de ahora en adelante IDS) es una herramienta de seguridad que sirve para monitorizar los intentos de intrusión que ocurren en un sistema informático, entendiendo por intento de intrusión cualquier pretensión de altetar la confidencialidad, integridad o disponibilidad de la información, o evitar los mecanismos de seguridad de una computadora o red. Las intrusiones pueden venir desde el exterior, o desde los usuarios de la propia red, de manera intencionada, o por falta de conocimientos.

Éstos intentos de intrusión, en su mayoría, buscan aprovechar alguna vulnerabilidad (deficiencias en algún software o configuración de algún servicio) con el fin de ser explotada, mediante código programado para ése fin (*exploits*).

Las razones por las cuales es una buena práctica usar IDSs son:

- Prevenir problemas al ser una medida disuasoria.
- Detectar ataques y otras violaciones que herramientas tradicionales no previenen.
- Detectar preámbulos de ataque.

- Documentar el riesgo a la organización.
- Dar información sobre las intrusiones que se está produciendo o se han producido (análisis forense).

2.1.1. Clasificación de los IDS

Los IDS se pueden clasificar de acuerdo con:

2.1.1.1. Fuentes de información

Es posible obtener la información tanto de los paquetes que circulan por la red, como de información local al host (logs del sistema) o software de aplicación.

- **IDSs basados en red (Network IDS)**
  Su funcionamiento se basa en la captura y análisis de paquetes de red. Pueden monitorizar múltiples hosts según la ubicación y configuración del sensor, por ejemplo todos los hosts localizados en un mismo segmento de red. Se pueden configurar de manera que sean totalmente transparentes con el fin de que no sean detectados.
  Las ventajas fundamentales son:
  - El número de hosts a monitorizar puede ser muy elevado, en función de la cantidad de tráfico y la capacidad del host que monitoriza.
  - Son elementos pasivos que no interfieren en el funcionamiento de la red. Por ejemplo, cuando se configuran a través de mirroring (o SPAN) o bien mediante un TAP.

  Como desventajas:
  - Existe el problema de que la cantidad de tráfico sea muy superior a la capacidad de CPU de la máquina que alberga el NIDS.
  - Este tipo de IDSs no capturan el tráfico que va cifrado.
  - A posteriori el IDS no sabe si el ataque tuvo éxito o no.
  - Existen problemas en algunos IDSs que no detectan ataques fragmentados en varios paquetes.
- **IDSs basados en host (Host IDS)**
  Este tipo de IDS se nutren de la información local del host, como pueden ser los ficheros de logs del sistema. Debido a su naturaleza se puede

saber con precisión que usuarios y procesos estuvieron involucrados en el ataque. Una diferencia sustancial con los NIDS es que los HIDS permiten ver el resultado de un ataque: si fue efectivo, las consecuencias sobre el activo, etc.
Las ventajas fundamentales de estos sistemas se pueden categorizar en:

- Son más precisos que los NIDS. Pueden ver ataques que para los NIDS pasarían desapercibidos.
- Para éstos el que el tráfico vaya cifrado es transparente, pues analizan siempre los datos al llegar al host o antes de ser enviados (siempre en texto plano).

Las principales desventajas son:

- Su alcance en cuanto a cantidad de hosts a monitorizar es menor. Además su configuración es más costosa ya que por cada host hay que configurar un IDS.
- Existe el peligro de que el propio host que alberga el IDS y la máquina en producción sea comprometido y se desconfigure el HIDS. Por ejemplo un ataque DoS dejaría el HIDS inutilizado temporalmente.
- Consumen ciclos de CPU, por lo que influyen en el rendimiento del sistema monitorizado.

2.1.1.2. Tipo de análisis

Existen dos vertientes en lo que se refiere al tipo de análisis. La primera, que es en la que se basan casi todos los IDSs, es la detección de abusos. La segunda es la detección de anomalías, en la que aún se está investigando y que muy pocos IDSs usan.

- **Detección de abusos o firmas**
  Este tipo de detectores analizan la actividad del sistema buscando eventos que coincidan con un patrón conocido. Las ventajas de esta tecnología son:

  - Resultan efectivos en la detección de ataques sin generar excesivas falsas alarmas.
  - Permiten diagnosticar el uso de una herramienta o ataque específico.

  Desventajas:

- Sólo detectan aquellos ataques para los que tienen patrones, así que hay que actualizarlos constantemente.
- Si el ataque varía un poco del estándar definido en la firma, el detector no será capaz de detectarlo.

- **Detección de anomalías**
  Este tipo de detección se basa en comportamientos inusuales, entiendo como un comportamiento inusual aquel que difiere de la actividad normal. Este tipo de IDSs construye perfiles que representa el uso o funcionamiento correcto. Estos perfiles pueden ser creados por usuario, host, proceso, conexiones de red, etc. Las medidas y técnicas usadas en este tipo de detección incluyen:

  - Parametrización de un umbral definido, de tal manera que todo atributo que exceda ese umbral será detectado como una anomalía. Por ejemplo, número de ficheros abiertos a la vez, consumo de CPU, intentos de acceso al sistema fallido, etc.
  - Medidas estadísticas, por ejemplo atributos que se perfilan con valores de comportamientos anteriores (histórico).
  - Otras técnicas más complejas incluyen redes neuronales, algoritmos genéticos, etc.

  Las dos primeras son las que actualmente se implementan en los IDSs, quedando para investigación la tercera.
  Las ventajas de este tipo de IDSs son:

  - Permiten detectar comportamientos para los cuales no se tiene conocimiento específico.
  - Permiten producir información que puede ser utilizada para definir firmas en la detección.

  Las desventajas:

  - El número de falsas alarmas es muy alto debido a los comportamientos no predecibles de usuarios y redes.
  - Requieren conjuntos de entrenamiento muy grandes para caracterizar los patrones de comportamiento normal.

2.1.1.3. Tipo de respuesta

Después de la detección del intento de intrusión el IDS reacciona. Según el tipo de respuesta dada hay dos clasificaciones: respuestas activas y pasivas.

- **Respuestas activas** En este tipo de respuestas existe una reacción al intento de intrusión, se clasifican en dos categorías:

  1. Recogida de información adicional: el sensor incrementa su capacidad de recolección de información para el evento con el fin de poder hacer un seguimiento exahustivo de lo que ocurre (por ejemplo capturando todo el tráfico que proviene de la IP atacante).
  2. Cambio del entorno: que consiste básicamente en cortar la conexión. Se puede mandar un segmento TCP con RST a 1, así se cerraría la conexión, o se puede generar una regla DROP en el firewall para esa IP.

- **Respuestas pasivas** En este tipo de respuestas se notifica al responsable de seguridad de la organización, al usuario del sistema atacado o a algún CERT de lo sucedido. También es posible avisar al administrador del sitio desde el cual se produjo el ataque avisándole de lo ocurrido, pero es posible que el atacante monitorice el correo electrónico de esa organización o que haya usado una IP falsa para su ataque.

2.1.2. Herramientas IDS de código libre

Existen gran variedad de herramientas IDS tanto comerciales como de código abierto para diversas plataformas.

2.1.2.1. Snort - Sourcefire

Snort [8] es el sistema de detección de intrusos más utilizado. Es de código abierto y su capacidad de análisis en tiempo real junto con su flexibilidad de integración con otras herramientas de seguridad lo han hecho muy popular entre los administradores de sistemas. Realiza análisis de protocolo, búsqueda y/o comparación del contenido de paquetes y detección de gran variedad de ataques (buffer overflow, port scan, ataques CGI, fingerprinting, detección de virus, etc). Además, la facilidad del lenguaje para definir reglas es otra de las razones por las que es tan utilizado. Permite generar diferentes tipos de salidas a los eventos detectados: texto ASCII, integración con syslog, comunicación por sockets UNIX, almacenamiento en bases de datos (MySQL, ORACLE, PostgresSQL..), etc.

Aunque en la actualidad hay una versión comercial para actualizar las reglas, la comunidad GPL realiza reglas gratuitas en proyectos como *bleeding*

Snort [9] donde diariamente se publican decenas de patrones nuevos.
La optimización de Snort en sus últimas versiones, que aprovecha la capacidad de los procesadores de última generación que trabajan con *threads*, permite obtener unos resultados muy buenos en lo que se refiere a relación entre la cantidad de tráfico a analizar y el consumo de ciclos de reloj.

2.1.2.2. Osiris - Host Integrity

Osiris [54] es un HIDS que comprueba la integridad del sistema de ficheros. Osiris toma periódicamente instantáneas del sistema de ficheros y las almacena en una base de datos, así la siguiente vez, compara los resultados con los almacenados la vez anterior en la base de datos. Si algún se detecta algún cambio, se avisa al administrador. Osiris también puede monitorizar por grupos, usuarios y módulos del kernel.
Osiris funciona en las plataformas: UNIX (incluyendo BSD), Linux, Mac OS X, AIX, IRIX y Windows NT/2000/XP.
Osiris se compone de tres aplicaciones: 'osirismd' que es el proceso de control, 'osirisd' daemon, que controla el estado de cada host reportando los resultados a osirismd, y 'osiris' proceso interactivo, que permite configurar el sistema.

2.1.2.3. Systrace

Este HIDS [16] funciona sólo en Linux y en los UNIX FreeBSD y OpenBSD. Pese a su escasez de integración, es una herramienta extremadamente potente. Su instalación y configuración requiere parchear el kernel y recompilar con opciones específicas. Es un IDS basado en anomalías.

Systrace hace cumplir las políticas en las llamadas al sistema utilizadas por las aplicaciones restringiendo su acceso al sistema. La política se genera interactivamente. Las operaciones no cubiertas por la política activan una advertencia y permiten que un usuario refine la política actualmente configurada. Con Systrace las aplicaciones binarias no confiables pueden ser *enjauladas* (chroot), ya que su acceso al sistema se puede restringir casi arbitrariamente. Además es posible enjaular aplicaciones que sólo están disponibles en forma binaria, sin necesidad de analizar directamente para lo que están diseñadas a realizar. Usando la característica de la elevación de privilegios de Systrace es posible quitar totalmente la necesidad de los binarios que usan setuid o setgid. En lugar de esto, Systrace ejecuta la aplicación sin privilegios y los eleva al nivel deseado cuando éste es requerido.

La principal desventaja es el coste asociado al aprendizaje para la generación de las políticas correctas.

2.1.2.4. Tripwire - Tripwire

Tripwire es un HIDS [17] que detecta ataques comprobando la integridad de ciertos ficheros del sistema. Su funcionamiento es sencillo: calcula el compendio de los ficheros críticos mediante funciones de dispersión MD5 o SHA, almacenando el resultado en un fichero en una base de datos. Periódicamente recalcula la función de dispersión comparando el resultado con el almacenado en la base de datos en busca de cambios.
Puede identificar cambios en otros atributos del sistema, como tamaño de los ficheros, tiempos en que se realizaron los accesos a escritura, etc.

Existen dos versiones de Tripwire: la versión para sistemas Windows, que es comercial, y la versión para UNIX, gratuita. También ha aparecido una versión para los dispositivos Cisco (conmutadores LAN, Routers, etc).

2.1.2.5. Bro

Bro [18] es un NIDS para sistemas UNIX. Esta herramienta monitoriza el tráfico de red y detecta intentos de intrusión basándose en el contenido de los paquetes. Bro detecta intrusiones comparando el tráfico de red con una serie de reglas que describen eventos problemáticos. Esta herramienta puede trabajar con grandes cantidades de tráfico.
Pero quizás su característica principal es que funciona analizando tráfico en el nivel de red (capa 4) o en el nivel de aplicación (capa 7). Si se quiere usar como detector de anomalías en el contexto el análisis se realizará en el nivel de aplicación (sesión telnet, FTP..)

2.1.2.6. Prelude

Esta herramienta [19] Open Source es una herramienta muy completa y está muy de moda actualmente por su capacidad de funcionar de manera distribuida .
Se compone de tres partes bien definidas:

- Sensores: integran tanto NIDS como HIDS, siendo el Prelude-nids su sensor NIDS por excelencia. Prelude-nids funciona con las reglas del Snort. Además integra un HIDS llamado Prelude-lml. Pero no sólo integra sensores propios, si no que puede integrar sensores externos

como Systrace, Bro. Para ello incorpora una librería *libprelude* que convierte cualquier programa en un sensor.

- Manager: recibe, procesa y registra las alertas generadas por los sensores en una base de datos. La comunicación de los sensores con el manager se realiza mediante SSL (Security Socket Layer), bien mediante certificados o autenticación.

- Frontend: interfaz para visualizar las alertas.

Sus características son: es modular, híbrido y distribuido. La salida se puede generar en texto plano, XML, volcar a una base de datos o pasar a otro manager.

2.1.3. Herramientas IDS comerciales

2.1.3.1. Dragon - Enterasys Network

Esta herramienta [10] se divide en tres partes. Las dos primeras son propiamente los sensores y se denominan *Dragon Sensor* y *Dragon Squire*. Su función es la de monitorizar los logs de los firewalls y otros sistemas. La información recolectada es enviada al *Dragon Server* quien la analiza y correla. Los sensores funcionan bajo plataformas: Linux, Solaris (Sparc y x86), FreeBSD, HP-UX, OpenBSD.

2.1.3.2. Cisco Intrusion Detection System - Cisco Systems

Este sistema es la solución ofrecida por Cisco Systems [11]. El software de red se implementa en el propio sistema operativo IOS, que se embebe en un router. Permite analizar paquetes y sesiones comparándolas con una base de datos de reglas que indiquen actividades sospechosas.

Además de detectar cualquier actividad sospechosa, el Cisco IOS IDS reacciona ante cualquier evento de tres maneras posibles:

- Enviar una alerta a un servidor syslog o un interfaz centralizado.

- Desechar el paquete (mediante una regla DROP).

- Cerrar la conexión TCP.

2.1.3.3. NFR - NFR Security

NFR Network Intrusion Detection [12] es una herramienta que reacciona ante cualquier ataque reconfigurando las reglas de los firewalls que haya en la red para detener el ataque.
Esta formada por las siguientes partes:

- Sensor NID: monitoriza todos los eventos que ocurren en la red (comportamientos anómalos, intentos de accesos no autorizados, ataques, etc). La capacidad de los sensores varía según el caudal: 10/100Mbit o Gigabit Ethernet.
- Servidor de Administración Central: su función es la de centralizar la información que viene de los sensores, almacenarla en bases de datos y generar informes con la información existente.
- Interfaz de administración: sirve para gestionar los sensores ID y hacer consultas a la base de datos.

2.1.3.4. CFI LANGuard Security Event Log Monitor - GFI Software

Esta herramienta [13] permite analizar y gestionar de manera automática todos los eventos ocurridos en la red.
Mediante el análisis de registros de sucesos se monitorizan los registros de sucesos de todos los servicios y estaciones de trabajo Windows NT/2000, XP y 2003 y se avisa de posibles intrusiones en tiempo real. Es algo similar al syslog de UNIX, donde se pueden centralizar todas las alertas en una máquina.

2.1.3.5. BlackIce - Internet Security Systems

La herramienta BlacIce [14] es un producto para plataformas Windows de la firma ISS (Internet Security Systems). Lo que hace peculiar a este IDS basado en red es que incorpora un firewall que se reconfigura reaccionando ante eventos detectados por el IDS.
Incorpora firmas específicas para la detección de paquetes con programas destructivos como troyanos o gusanos.

2.1.4. Formatos estándar de alertas

Cuando se habla de sistemas de detección de intrusos distribuidos o de herramientas capaces de integrar sensores de diferentes fabricantes, es im-

portante definir un estándar para la interoperabilidad de las distintas herramientas. Es por esta razón por la que se ha creado el estándar IDMEF o Intrusion Detection Message Exchange Format.

2.1.4.1. IDMEF

Este [21] estándar define qué campos deben crearse cuando se genere una alerta por un sensor. El estándar está definido en XML y entre otros campos contempla [22]:

- Identificador único de la alerta.
- Hora de creación de la alerta.
- Dirección IP origen y destino.
- Puerto y versión del protocolo.
- Bugtraq ID.
- Contenido del paquete que ha disparado la alerta.

2.1.5. Nuevas tendencias

El abaratamiento del hardware y la electrónica de red hace posible que cada vez más se configuren sistemas de detección de intrusos con varios IDSs, con el fin de obtener más información para analizar. Este tipo de arquitectura donde intervienen varios sensores se denomina *Sistema de detección de intrusos distribuido (Distributed IDS)*

2.1.5.1. Distributed IDS (DIDS)

Los DIDS son una combinación de varios IDS, pudiendo convivir sólo varios HIDSs, NIDSs o combinaciones de estos. Lo más habitual es que haya un sistema heterogéneo de IDSs. Una definición formal de DIDS podría ser la de 'aquel sistema de detección capaz de agregar eventos generados por diferentes fuentes, proporcionando así una imagen más amplia y detallada de las actividades maliciosas en un determinado entorno' [7], [60]
Las razones fundamentales para usar DIDS son:

- Tener una visión de la red más amplia. Se puede monitorizar distintos segmentos de red o un mismo segmento de red con varios IDSs.
- Tolerancia a fallos. Puede haber redundancia de IDSs.

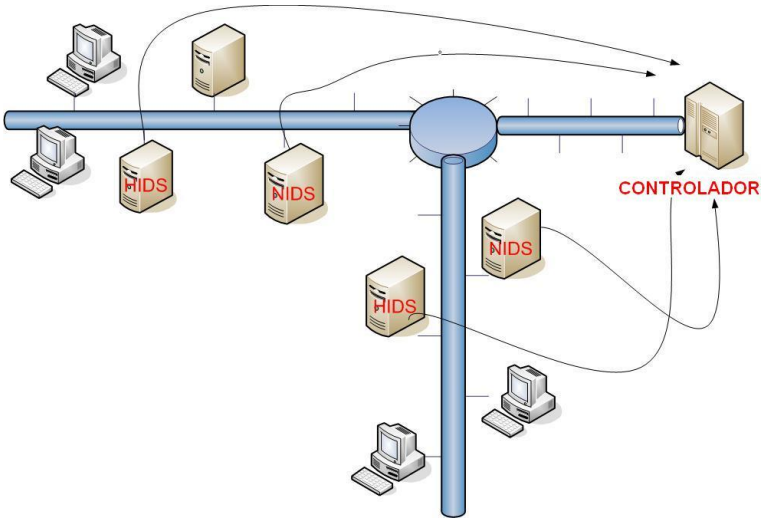

Figura 2.1: Ejemplo de Sistema de Detección de Intrusos Distribuido (DIDS) con HIDS (Host Intrusion Detection System y NIDS (Network Intrusion Detection System

- Puede existir comunicación entre los distintos IDSs y/o bien una centralización de todas las alertas en un punto central

En este tipo de arquitectura distribuida, existe un punto central donde se recogen todas las alertas generadas por los distintos sensores. Un ejemplo de DIDS es el que se muestra en la figura 2.2.

2.2. Análisis de vulnerabilidades

Una de las tareas fundamentales de los analistas de seguridad es la de detectar las posibles vulnerabilidades que los intrusos puedan aprovechar. Para lograr tal objetivo se han desarollado herramientas denominadas auditores de sistemas o escáneres de vulnerabilidades. Estas herramientas varían bastante en cuanto a su función, existiendo herramientas exclusivas para aplicaciones webs, para servicios en los SO, para configuraciones, etc.

Las fases en las que se divide un análisis de vulnerabilidades son estas tres:

1. Detección de máquinas activas: esta primera fase se centra en detectar qué hosts están activos o son visibles. Para ello se puede utilizar la herramienta Nmap [23] o cualquier escaneador de puertos.

2. Detección de servicios activos: el objetivo de esta fase se centra en detectar qué servicios están activos, en qué puertos, que versiones, qué configuración, etc. Se usa también Nmap y SNMP.

3. Test de intrusión: la idea de esta fase es la de aplicar ciertos exploits o scripts con el objetivo de ver hasta que punto son vulnerables los servicios que han sido detectados en la fase dos. Es una fase delicada porque se puede dejar fuera de servicio una máquina o alguno de los servicios que corre en ella (Denial of Service).

Aunque estas son las fases que se usan para una auditoría completa, dependiendo del tipo de auditoría que se quiera realizar habrá otro tipo de análisis. En el caso de una auditoría que se realiza únicamente sobre aplicaciones web que corren sobre un servidor HTTP, únicamente se analizaran las posibles vulnerabilidades existentes en el servidor HTTP, y se deberá centrar más la atención en el análisis de la posibles deficiencias en las propias aplicaciones web.

Estas herramientas son un mecanismo de seguridad para prevenir posibles ataques gracias a su funcionalidad en la detección de errores en las configuraciones o en la falta de actualizaciones.

2.2.1. Herramientas de auditoría

Las herramientas más utilizadas para auditar son:

2.2.1.1. Nmap

Nmap [23] es un escaneador de puertos muy popular entre los administradores de sistemas. Aunque en si no es una herramienta que detecte fallos en algún servicio, si permite averiguar la visibilidad de los puertos abiertos (fase previa a la realización de cualquier auditoría). Dispone de múltiples opciones que lo hacen muy flexible y potente. Desde flags para evitar la detección por los IDSs (-sS), para detectar con precisión la versión del Sistema Operativo (-O), la versión de los servicios que se están ejecutando en la máquina (-sV), etc.

2.2.1.2. Nikto

Nikto [26] es un escaneador de vulnerabilidades en aplicaciones web basado en la librería para perl Libwhisker [25]. Una buena síntesis de la capacidad de esta librería para perl se puede encontrar en un artículo de Securityfocus

[24]. Esta herramienta permite comprobar formularios con distintos métodos como GET y POST probando valores para todos los campos de los formularios CGIs en busca de posibles desbordamientos, inserciones de código SQL, XSS, o cualquier otro posible ataque en aplicaciones web.

2.2.1.3. Nessus Security Scanner

Nessus [27] es la herramienta más popular para auditar sistemas. Actualmente es una herramienta de pago si se quiere tener al día las firmas (o plugins), si no hay que esperar 7 días a poder descargar las firmas.

Nessus es una herramienta formada por dos partes: servidor y cliente. La parte servidora (que corre en linux como un daemon) es la que realiza los ataques y controla las actualizaciones de las firmas, dejando para el cliente la configuración de a quién realizar el análisis, cómo realizarlo, con que plugins, etc. Cabe destacar que la conexión al servidor se puede realizar desde cualquier host, remoto o no (la propia herramienta maneja certificados para más seguridad).
La salida del análisis se puede ver de manera gráfica en el el propi interfaz del cliente o bien generar un fichero que puede ser exportado a HTML, XML, etc.
La característica que lo hace tan potente es el escaneo inteligente que realiza, detectando cualquier servicio en puertos no estándar y probando una batería de diferentes exploits sobre cada servicio detectado. Nessus incorpora además Nikto entre sus plugins, realizando un exhaustivo análisis de todo los CGIs y aplicaciones web. El escaneo para detectar servicios lo puede hacer con Nmap o con sus propios plugins. Como se ve, Nessus es un producto que incorpora otras herramientas y de ahí su potencia. A día de hoy el número de plugins que Nessus tiene para realizar los tests está entorno a los 10.000.

2.3. Otras herramientas de seguridad

Además de las aplicaciones auditoras y de detección de intrusos existen un conjunto de herramientas que sirven a los responsables de seguridad para conseguir su fin, estas herramientas son los monitores.
Los monitores varían en función y en uso, pero básicamente tienen un objetivo común, que es el de controlar o mesurar algún evento o servicio. Los más comunes son los logs que implementan los propios sistemas operativos, como syslog de UNIX, logs de servicios o aplicaciones específicas como logs de Apache. También existen monitores para la red, como es el caso de netstat

de Unix.

2.3.1. Herramientas de red

2.3.1.1. ARPwatch

Esta herramienta de seguridad [37] permite monitorizar a nivel de capa de enlace qué eventos o anomalías ocurren. Entre sus funciones permite mantener una tabla MAC / IP, manteniendo un control de quién es cada uno en una red local. Es útil para evitar ataques de envenenamiento ARP [38] y posibles falsificaciones a nivel 2.

2.3.1.2. Ntop

Ntop [35] permite controlar y sacar estadísticas de todo el tráfico de red que afecte a un host. Se podría decir que tiene las funcionalidades de netstat pero ampliadas, ya que permite ver números de paquetes enviados, número de paquetes recibidos (siempre en cualquier protocolo ICMP, UDP, TCP..).

2.3.2. Herramientas para la detección de sistemas operativos

2.3.2.1. p0f

El objetivo de este monitor [39] es el de averiguar el sistema operativo de un host. Su nivel de precisión es alto y permite localizar incluso la versión del Service Pack que tiene instalado un sistema Windows. Es una versión mejorada de la herramienta QUESO. Puede ser lanzado para que avise de cualquier cambio en el sistema operativo, el cual puede ser útil para controlar suplantaciones a nivel IP.

2.4. Proyecto Elementos Activos de Seguridad (ELAS)

ELAS [1] es un proyecto de investigación financiado por el Ministerio de Ciencia y Tecnología [29], que busca mecanismos de seguridad adaptativos. El proyecto se centra en la detección de actividades sospechosas en la red. Lo que se pretende en este proyecto es reducir el número de falsos positivos y falsos negativos usando distintos tipos de sensores (NIDSs basados en firmas, sensores basados en anomalías y HIDSs) y correlando los eventos.

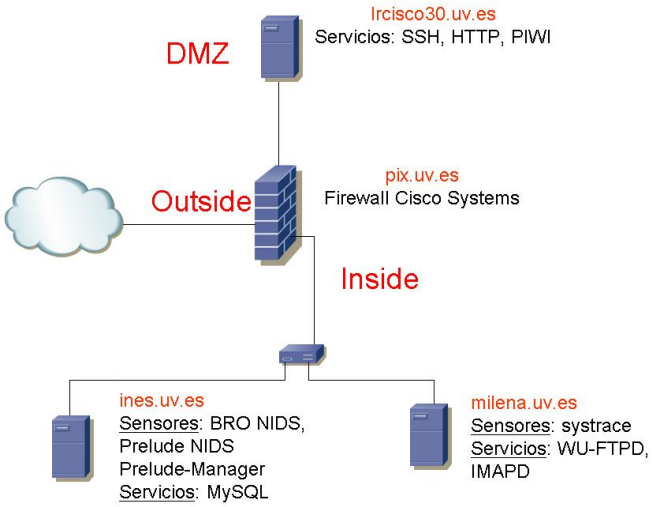

Figura 2.2: Topología Elementos Activos Seguridad (ELAS)

En el proyecto participan el Instituto de Robótica, el centro de Cálculo de la Universidad de Valencia y la red nacional de I+D Rediris.

2.4.1. Descripción de la maqueta desarrollada en ELAS

En la maqueta de pruebas que se muestra en la figura 2.2 se distinguen tres zonas de red, Outside[1] Inside[2], DMZ[3], separadas por un cortafuegos PIX[32] de Cisco System.

Las funciones de cada elemento de la maqueta son:

- ircisco30.uv.es: esta máquina ubicada en la zona DMZ tiene los servicios HTTP y SSH abiertos y desde ella se controla todo el sistema. El servidor Apache que tiene instalado permite monitorizar las alertas generadas a través de un Fronted hecho, en PHP (Piwi)que ataca la base de datos de resultados ubicada en ines.uv.es.
  Desde ésta máquina se lanzarán los ataques a milena. Básicamente se trata de ataques de desbordamiento de pila programados para explotar algunos servicios.

[1] Conexión con internet
[2] Zona de máxima seguridad
[3] Zona desmilitarizada: zona donde están ubicados los servidores que serán accedidos desde Internet

- milena.uv.es: es el servidor al que se ataca durante las pruebas. Tiene habilitados los servicios Telnet (como medio de acceso para modificar alguna configuración), FTP e IMAP, éstos dos últimos con fallos de seguridad y sus respectivos exploits disponibles en ircisco30.uv.es. Dentro de esta máquina se ejecuta un HIDS (systrace) que está configurado para enviar las alertas generadas al manager (centro de recolección de alertas) de Prelude, mediante cifrado SSL.

- ines.uv.es: aquí se encuentran la mayoría de herramientas de detección. Tiene instalado el manager de Prelude que recibirá alertas de prelude-nids (instalado como NIDS basado en firmas) y de BRO (NIDS basado en anomalías). También contiene una base de datos MySQL donde se conecta al manager de Prelude para almacenar las alertas. Un artículo muy interesante de cómo funciona Prelude como DIDS distribuido e híbrido lo podemos encontrar en la referencia [20].

- pix.uv.es: este tipo de firewall funciona siempre definiendo un nivel de seguridad en cada una de las zonas, de tal manera que de una zona de mayor nivel siempre se puede acceder a una zona de menor nivel. En el caso de ésta maqueta los niveles definidos son: 100 zona Inside, 50 zona DMZ y 0 zona Outside, por lo que desde Inside se podrá acceder a Outside o DMZ, y de DMZ sólo hacia Internet. Otra característica de los PIX es que siempre hacen traducción de direcciones (Network Address Translation), así tendremos IPs privadas dentro de cada zona e IPs públicas para referenciar cada máquina desde Internet.

  Las reglas del PIX permiten el tráfico desde un origen 'cualquiera' con destino el servidor ubicado en DMZ ircisco30 al puerto SSH para gestión remota y al puerto 80 para consultar los resultados de las pruebas. Todo el tráfico desde ircisco30 hacia la zona INSIDE se permite, así es posible realizar consultas a la base de datos MySQL ubicada en ines, poder gestionar remotamente las máquinas mediante SSH y poder hacer pruebas de ataques a los servicios IMAP y FTP ubicados en Milena.

En la tabla 2.1 se detalla la función de cada sensor y los servicios instalados.

En el proyecto se han programado distintos *exploits* con el fin de poder evaluar las capacidades de cada sensor y hacer un seguimiento del ataque

Máquina	IDS	Servicios
ircisco30	-	HTTP, SSH
milena	systrace	IMAP, FTP, TELNET
ines	prelude-nids, BRO	SSH, MYSQL

Cuadro 2.1: Servicios e IDSs en Elementos Activos Seguridad.

durante su ejecución. Por ejemplo, para el caso del servidor Wu-ftp existen dos *exploits* distintos, uno aprovecha un fallo de *format string* al ejecutar el comando *site exec* y el otro un *heap overflow* cuando se hace un *List*.

2.5. Resumen

En este capítulo se ha hecho una introducción a los conceptos de seguridad necesarios, así como a los sistemas de detección de intrusos y la tendencia actual de las herramientas IDSs, junto con la de otras herramientas de seguridad, como los auditores de sistemas. También se ha analizado con detalle el proyecto inicial (ELAS) que dio lugar a este proyecto final de carrera.

Capítulo 3

Análisis del sistema

3.1. Análisis del sistema

El objetivo es montar un sistema que minimize el número de alertas, las priorice, permita realizar un análisis en tiempo real de qué ocurre y de qué ha pasado en un instante concreto y reduzca el número de falsos positivos y falsos negativos. Para ello, aplicaremos herramientas auditoras, sistemas de detección de intrusos, monitores de red y/o servicios, y un sistema de correlación de la información.
Tomando como base estos objetivos, a continuacion se analizará la función de cada de herramienta.

3.1.1. Análisis de herramientas

Las diferentes herramientas utilizadas en el análisis son:

Escáner de vulnerabilidades
Un escáner de vulnerabilidades debe realizar las siguientes funciones:

- Ser capaz de reconocer los servicios de una máquina aunque no se estén ejecutando en los puertos por defecto
- Tener capacidad de realizar positivamente muchos tests.
- Flexibilidad para aplicar distintos niveles de dureza de test.
- Mantenerlos al día con las actualizaciones para los últimos bugs.

Detección de intrusos de red
Es necesario instalar un sistema de detección que cumpla con las siguientes características.

- Analizar el tráfico en tiempo real
- Almacenar las alertas y los paquetes que las producen.
- Llevar a cabo continuas actualizaciones de firmas para detectar nuevos ataques.
- Evitar su detección en la red.
- Disponer de interfaz un de gestión y monitorización.

Detección de intrusos de host
El HIDS debe ser capaz de:

- Analizar los eventos que ocurren en un host que delaten la presencia de intrusos.
- Ver que ficheros se han modificado
- Ser adaptable a los servicios a monitorizar

Recoleción de datos
La recoleción de datos se podrá realizar tanto localmente en el propio host como de manera remota, enviando los resultados a otro host. Así, debe ser posible:

- Recibir los datos de distintos sensores de manera segura.
- Monitorizar distintos servicios y ficheros de logs.
- Recolectar de distintos sensores.

3.2. Análisis de correladores

Existen dos tendencias destacadas en los modelos de correlación, modelos basados en algoritmos heurísticos o inteligencia artificial y modelos basados en secuencias de eventos. Quizá el segundo modelo es más sencillo conceptualmente, ya que la idea general es hacer un seguimiento de los distintos eventos que ocurren.

3.2.1. Correlación mediante Algoritmos Heurísticos

La idea básica es detectar situaciones de riesgo mediante funciones heurísticas, para detectar ocurrencias para las cuales no hay patrones existentes. Un acercamiento a este tipo de correladores es el que implementa la herramienta OSSIM [42] la cual será la herramienta usada en este proyecto.

El algoritmo se basa en la acumulación de eventos, obteniendo un indicador númerico del riesgo instantáneo, denominado "Nivel Aucumulado de Riesgo". Es importante tener clara la definición clásica de riesgo, que se puede explicar como: *"la probabilidad de que un ataque aproveche una vulnerabilidad teniendo un impacto sobre los activos"*. Lo que se pretende es ofrecer una visión global rápida de la situación y detectar posibles ataques no caracterizados por patrones.

El algoritmo implementado, también llamado CALM (*Compromise and Attack Level Monitor*), recibe como entradas los eventos que le envían los distintos sensores, y devuelve como salida un valor único que marca el estado del sistema. Se puede correr el algoritmo tanto para un host como para un grupo de máquinas (en un segmento concreto, subred, etc).

Acumulación de eventos
Existen dos variables de estado que sirven como acumuladores: C y A. La variable C define el nivel de compromiso, es decir, la posibilidad de que una máquina se encuentre comprometida. La variable A mide el posible riesgo al que se ve sometido el activo. Dependiendo del tipo de activo y de su localización estás variables tendrán un significado diferente, por ejemplo un activo ubicado en una DMZ tendrá un valor de A muy elevado, en cambio la C será baja.

La asignación de valores a C y A se basará en tres reglas:

1. Un ataque de una máquina 1 a una máquina 2 aumentará la A en la máquina 2 y la C en la máquina 1.

2. En caso de una respuesta a una ataque (attack respone), se incrementará el valor de la C en ambas máquinas.

3. Los eventos internos sólo aumentarán el valor de C.

Acumulación en el tiempo
Este algoritmo está pensado para la monitorización en tiempo real, por lo que la ventana de tiempo será a corto plazo y además se valorarán con un peso

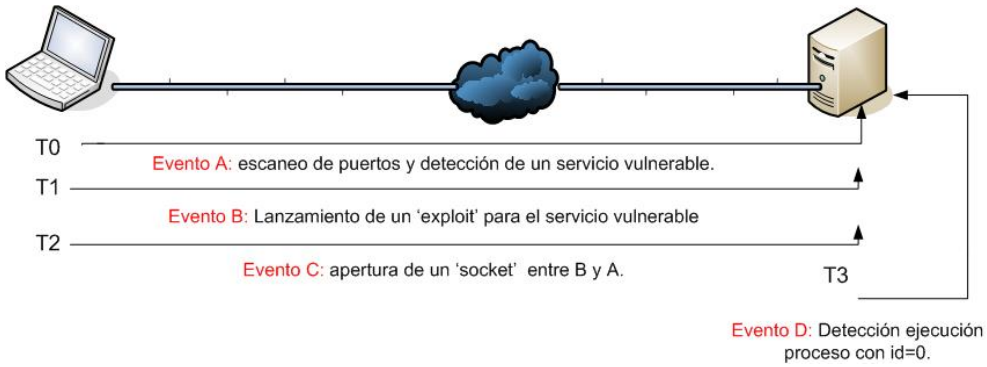

Figura 3.1: Ejemplo de correlación de eventos en diferentes instante de tiempo: T0, T1, T2 y T3

mayor los eventos más recientes frente a eventos más lejanos que tendrán un peso menor.

3.2.2. Correlación mediante secuencia de eventos

La idea de funcionamiento de este tipo de correlación se podría definir como la sucesión de una serie de eventos que concuerdan con unos patrones definidos. Un acercamiento a este tipo de correlación se puede encontrar en [30], [31]

Un ejemplo sencillo sería: "si ocurre un evento A, y luego un evento B, seguido de un evento C, hacer la acción D". Situándose en el contexto de un ataque y siguiendo la secuencia marcada en la figura 3.1 se puede hacer una analogía de los eventos de la siguiente manera:

```
Si 'escaneo' de A a B
    si 'exploit' de A:X a B:Y
        Si 'socket' abierto de B:Y hacia A:X
            Si 'comando detectado' en B.
                entonces MANDAR_ALERTA();
            fsi
        fsi
    fsi
fsi
```

La idea general es relacionar eventos en un espacio de tiempo definido para cada regla, relacionando las IPs[1] origen y destino junto con sus puertos, y

[1]En el ejemplo anterior se ha tomado la descripción A:X como dirección IP A y puerto X

por supuesto las firmas de los patrones que actúan o bien como detectores (IDSs, etc.) o como sensores (conexiones TCP abiertas, comando ejecutados con id=0, etc.)

3.3. Un modelo de correlación general

Existen muchas investigaciones en el campo de la correlación de eventos, con pruebas y tests que demuestran la eficiencia de ciertos enfoques o modelos de correlación. Se puede encontrar más información en: [56], [57], [58], [59].

El modelo en el que se basa este proyecto, que se puede descargar de la página web de uno de los autores [43], define una serie de procesos o pasos a seguir para correlar la información. Este modelo ha sido probado en distintos escenarios publicados en la red, obteniendo una reducción de las alertas en un porcentaje muy elevado. En el caso peor se obtiene una reducción de más de la mitad de las alertas (53 %), siendo en el caso mejor un 99,96 % .

Los autores del artículo se basan en lo que ellos llaman *Meta Alerta*, que no es más que ir fusionando alertas relacionadas entre sí, con el fin de obtener un nivel de abstracción mayor. Las *Meta Alertas* pueden a su vez fusionarse con otras *Meta Alertas* o con simples alertas. El objetivo es obtener una jerarquía en forma de árbol, donde el nodo raíz es la última *Meta Alerta* obtenida.

Por otro lado, para la prueba del diseño de este modelo se utilizan los llamados *Data Set*, que son un conjunto de datos (como tráfico de red) de un ataque guardado para que luego pueda ser utilizando en modo *background* para ser analizado. Aunque evidentemente presenta varias desventajas claras, no se puede comprobar la veracidad de la alerta y que impacto ha ocasionado sobre los activos.

Las fases en las que, según su autor Giovani Vigna[44], se divide el proceso de correlación que se puede observar en la figura 3.2 son las siguientes:

1. Normalización: una vez se reciben las distintas alertas de los sensores hay que normalizarlas de acuerdo a un estándar con el fin de que todos los elementos que componen el proceso puedan entenderlas. Esto es así ya que las alertas producidas por distintos sensores pueden estar codificadas de distinta manera. Al final de este proceso se obtendrán las alertas con sus respectivos atributos en un formato común. En el caso del artículo se utiliza IDMEF como formato.

2. Preproceso: esta fase se podría decir que es una depuración de la ante-

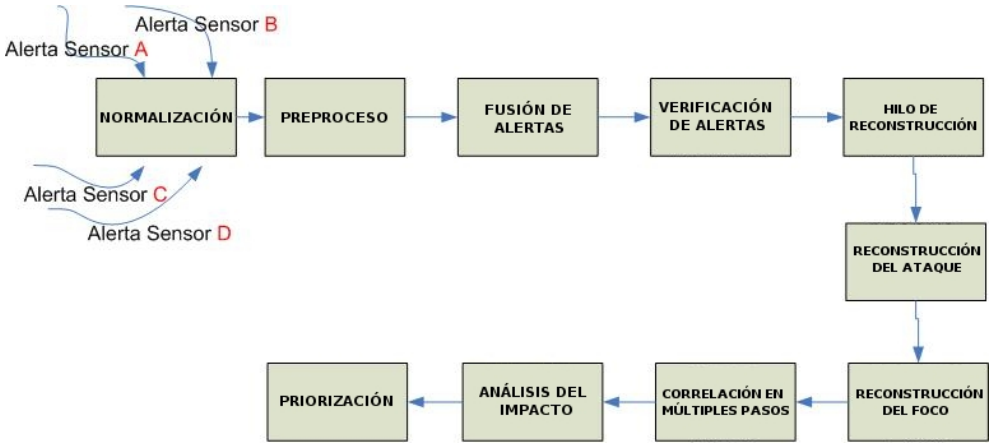

Figura 3.2: Procesos que intervienen en la correlación

rior, ya que lo que se pretende es rellenar aquellos campos que algunos sensores dejan en blanco cuando lanzan alguna alerta. En

3. Fusion de alertas: el objetivo de este elemento es el de combinar alertas de diferentes sensores (por ejemplo dos sensores de red distintos) pero que hacen referencia a un mismo ataque. Aquí juega un papel muy importante el factor tiempo junto con la información de la alerta.

4. Verificación de la alerta: la función aquí es la de desechar las alertas que no sean alertas reales o bien que no hayan conseguido su objetivo (explotar una vulnerabilidad, reventar un servicio mediante un ataque DoS, etc). Se puede obtener tres resultados:

   - El ataque ha sido efectivo. Se está ante un "verdadero positivo".
   - El ataque no ha tenido éxito, y se cataloga la alerta como "no relevante".
   - El sensor ha identificado un evento como ataque que no es un ataque real ("falso positivo").

5. Reconstrucción del hilo de ataque: Es en esta fase donde se realiza un análisis de la sucesión de eventos que tienen como origen una máquina concreta hacia una máquina destino también precisa. En este caso se consideraran las alertas que tienen como origen y destino el mismo en un sensor dado, justo al contrario que ocurre en la fase de fusión donde las alertas a tener en cuenta deben ser en sensores distintos.

6. Reconstrucción de la sesión de ataque: es justo en este punto cuando se relacionan los ataques detectados en un host con los ataques de

lo que se ha tenido constancia a través de la red. A priori no hay una relación directa entre qué IP o puerto puede estar relacionado con un proceso o fichero de una máquina, por lo que la información que permite relacionar estos eventos es la magnitud "tiempo". Por ejemplo, un proceso sospechoso lanzado en una máquina pocos segundos después de que un ataque haya sido detectado en la red, puede ser un ejemplo claro.

7. Reconocimiento del foco del ataque: este elemento es unicamente válido para ataques DDoS *(many2one)*, y para ver si la máquina atacada está siendo usada como pasarela para atacar otras máquinas (*one2many*). El factor a tener en cuenta es otra vez el tiempo, donde se define una ventana de tiempo que nos definirá un umbral, que si no se sobrepasa generará una *metaalerta*.

8. Correlación multipaso: el objetivo es ver la evolución de un atacante, desde la primera fase de un simple escaneo de puertos, hasta la ejecución de comandos como privilegios de administrador. Todo ello comparándose con patrones para cada subfase.

9. Análisis del impacto: ¿qué impacto ha tenido sobre la red o el servidor el ataque?, la respuesta viene definida por está fase del proceso. Por ejemplo, si el servidor tumbado es el DNS de la empresa, o si se ha tumbado NFS que sirve el sistema de ficheros de toda la red corporativa. En esta fase los autores utilizan *heartbeat* para comprobar que los servicios están activos.

10. Priorización de alertas: para esta fase se utiliza la información de la etapa anterior junto con una base de datos de los activos para obtener la importancia de cada elemento en la red. Por ejemplo, un servidor DNS corporativo afecta a los servicios de correo y web. En esta fase el usuario o administrador de la red debe introducir a su modo de ver los valores de sus activos.

3.4. Resumen

En este capítulo se ha introducido las herramientas necesarias para el desarrollo del sistema y las diferentes técnicas de correlación, basadas en algortimos heurísticos y secuencias de eventos. En la última sección se ha explicado un modelo general de correlación de eventos, que ha sido probado en distintos entornos con resultados excelentes. Éste modelo es el que en capítulos posteriores será el elegido como la propuesta del sistema de correlación.

Capítulo 4

Diseño del sistema

Existen dos fases bien diferenciadas en la realización de este proyecto. La primera de ellas consiste en la configuración e integración de todas las herramientas de seguridad necesarias en nuestro sistema, y una segunda en la que se ha desarrollado las propias directivas de correlación.

La necesidad de una herramienta capaz de gestionar distintos IDSs y de correlar los datos que producen éstos, llevó al proyectante a una fácil elección: OSSIM. La razones por las que se decidió usar OSSIM principalmente fueron: su capacidad de integración con la mayoría de IDSs de libre distribución que existen en el mercado, su motor de correlación, su flexibilidad a la hora de configurar y su capacidad de integración con sistemas Linux.

En el punto siguiente se analizan con detalle las características de esta herramienta.

4.1. Open Source Security Information Management (OSSIM)

OSSIM [42] ofrece la posibilidad de obtener una visibilidad de todos los eventos de los sistemas en un punto con un mismo formato, y a través de esta situación privilegiada, relacionar y procesar la información permitiendo aumentar la capacidad de detección, priorizar los eventos según el contexto en que se produzcan, y monitorizar el estado de seguridad de nuestra red.
Permite integrar la información generada por IDSs, detectores de anomalías, Firewalls y monitores varios. Existe la posibilidad de hacer un inventario de todos los activos, definir la topología de red, definir políticas de seguridad,

etc.

Las funciones principales de OSSIM, se podría definir en tres fases:

- **Preproceso**: la detección en si misma, la generación de alertas por los detectores y la consolidación previa al envío de información.
- **Colección**: el envío y recepción de toda la información de estos detectores en un punto central.
- **Postproceso**: el tratamiento que se realizará una vez se disponga de toda la información centralizada.

La fase de Postproceso incluye tres métodos diferentes:

- **Priorización**: se priorizan las alertas que se reciben mediante un proceso de contextualización desarrollado a través de la definición de una Política Topológica de Seguridad y del Inventariado de los sistemas.
- **Valoración de Riesgo**: cada evento será valorado respecto al Riesgo que implique, es decir, de una forma proporcional entre el activo al que se aplica, la amenaza que supone y la probabilidad del evento.
- **Correlación**: se analiza un conjunto de eventos para obtener una información de mayor valor.

Así pues, las alertas ofrecidas por los detectores, después de ser tratadas, pasarán a ser alarmas o no según cada caso. Una alarma es el resultado del proceso de varias alertas y posee un mayor grado de abstracción y de fiabilidad.

4.1.1. Las etapas de OSSIM

Para conocer a fondo OSSIM, a continuación se explicará cada una de las fases en las que se divide todo el proceso.

4.1.1.1. Detección

Los detectores integrados en OSSIM tienen dos orientaciones bien distintas:

- **Detectores de Patrones**: no sólo en el contexto de los sistemas de detección de intrusos, sino en un sentido más amplio, como Firewalls o routers capaces de detectar escaneos de puertos, intentos de Spoofing o ataques de fragmetación. Además muchas herramientas que trabajan a nivel de host permiten monitorizar los eventos al estilo *logger*, como el syslog de los sistemas UNIX.
  El objetivo fundamental de este tipo de detección es el de tener una monitorización total sobre los eventos de toda la red, la visibilidad de la red.

- **Detección de anomalías**: la capacidad de detección de anomalías es más reciente que la de patrones. En este caso al sistema de detección no tenemos que decirle qué es bueno o qué es malo, él es capaz de 'aprender' por sí solo y alertar cuando un comportamiento difiera lo suficiente de lo que ha aprendido como normal. Esta nueva funcionalidad ofrece un punto de vista diferente y complementario sobre detección de patrones pues la naturaleza de los dos procesos es opuesta. La detección de anomalías puede ser especialmente útil para prevenir por ejemplo ataques perimetrales, éstos son en sí una anomalía continua, en la dirección el sentido de las comunicaciones y el camino que definen, en el flujo de datos, el tamaño, el tiempo, el horario, el contenido, etc.
  Esta técnica ofrece una solución de control de accesos de usuarios privilegiados como son los ataques internos, por ejemplo de empleados desleales, los cuales no implican la violación de ninguna política ni la ejecución de ningún exploit. Implican sin embargo una anomalía en el uso y la forma de uso de un servicio.
  Algunos ejemplos de este tipo de anomalías son: el uso en horario anormal, exceso de tráfico, cambio en el sistema operativo, etc.
  El problema fundamental de éste tipo de detección, como se analizó en la sección 2 de esta memoria, es el gran número de falsos positivos, por lo que la información de este tipo de sensores solo se tiene en cuenta como información complementaria a los detectores de patrones.

4.1.1.2. Centralización / normalización

La normalización y centralización (o agregación) tiene como objetivo unificar en una única consola y formato los eventos de seguridad de todos los sistemas críticos de la organización. La normalización implica la existencia de un *parser* o traductor que conozca los tipos y formatos de alertas de los diferentes detectores. Así, será posible tanto almacenar las alertas en un mismo formato en una base de datos.

4.1.1.3. Priorización

La prioridad de una alerta debe ser dependiente de la Topología y el Inventario de sistemas de la organización. Por ejemplo, una alerta de un gusano que ataque los servidores Internet Information Server de Windows 2000, no tendrá prioridad si los servidores web del inventario que monitorizamos son Apache.

4.1.1.4. Valoración del riesgo

La importancia que se debe dar a un evento ha de depender de de estos tres factores:

1. El valor del activo al que el evento se refiere
2. La amenaza que representa el evento
3. La probabilidad de que este evento ocurra

Para realizar tanto la tarea, como la fase de priorización, se dispone de un *framework* (Ossim-Framework) donde se puede configurar:

- Política de Seguridad, o valoración de parejas activo-amenazas según la Topología y flujo de los datos.
- Inventario.
- Valoración de activos.
- Valoración de amenazas (priorización de alertas).
- Valoración de fiabilidad de cada alerta.
- Definición de alarmas.

4.1.1.5. Correlación

El funcionamiento del correlador de OSSIM fue analizado en el capítulo anterior.

4.1.1.6. Monitores

Con el fin de poder controlar las anomalías, OSSIM define tres tipos de monitores:

- Monitor de Uso: ofrece datos generales de la máquina como el número de bytes que transmite al día.
- Monitor de Perfiles: ofrece datos específicos del uso realizado por el usuario y permite establecer un perfil, por ejemplo si usa correo, pop, y http, es un perfil de usuario normal.
- Monitor de Sesiones: permite ver en tiempo real las sesiones que está realizando el usuario. Ofrece una foto instantánea de la actividad de una máquina en la red.

Los monitores que integra OSSIM son herramientas como Ntop, RRD [40], MRTG [41], etc.

4.1.1.7. Consola forense

La Consola forense permite acceder a toda la información recogida y almacenada por el colector. Esta consola es un buscador que ataca a la base de datos de eventos, y permite al administrador analizar a posteriori, y de una forma centralizada, los eventos de seguridad de todos los elementos críticos de la red.

4.1.1.8. Cuadro de mandos

La última de las funcionalidades es el cuadro de mandos, mediante el cual se puede ofrecer una visión de alto nivel de la situación de nuestra red en cuanto a seguridad. El cuadro de mandos monitoriza una serie de indicadores que miden el estado de la organización respecto de seguridad. Permitirá definir una serie de umbrales u objetivos que debe cumplir el sistema. Estos umbrales serán definidos de forma absoluta o relativa como un grado de anomalía. Se podrá asignar el envío de alarmas cuando se superen estos umbrales o la ejecución.

4.1.2. Los procesos de OSSIM

OSSIM se divide en tres aplicaciones o procesos distintos: Ossim-agent, Ossim-server y Ossim-framework.

Ossim-server es el corazón de OSSIM. Se encarga de recibir los eventos que le envían los distintos agentes (Ossim-agents), además de realizar las labores de priorización, correlación y gestión del propio motor de Ossim.

Ossim-agent es un proceso que corre en los hosts donde se quiere monitorizar algún evento. Sus funciones básicas consisten en recoger los eventos de algún detector o sensor, y reportarlos a algún Ossim-server mediante conexiones TCP. Cada Ossim-agent tendrá configurado un conjunto de detectores o monitores, que estarán ejecutándose en el host que controlan. Para cada uno de las fuentes de información de Ossim-agent, existe un parser escrito en Python [45], que se encarga de analizar los logs en el formato estándar del propio sensor, traducirlos a IDMEF y enviarlos al Ossim-server.

Ossim-framework es el gestor de OSSIM. Se podría definir como la capa intermedia entre el propio servidor y el usuario. Entre otras cosas, incorpora un portal el PHP que corre a través del servidor web para facilitar la configuración del sistema.

4.1.3. Las herramientas de OSSIM

OSSIM integra herramientas de software libre, obteniendo una arquitectura abierta y adaptable según las necesidades. Pero además también utiliza los logs de otras herramientas no libres.

Los principales productos que integra OSSIM son: Apache [46], Arpwatch [37], Cisco IDS [11], Firewall PIX de cisco [32], Routers Cisco [47], Firewall one de CheckPoint [48], Iptables [51], Servidor de microsoft Internet Information Server [52], Ntop [35], Ntsyslog [53], Osiris [54], Prelude HIDS [19], Snort [8], Syslog, p0f, tcptrack, snarewindows, realsecure, rrd, opennms, netgear.

4.1.4. Las directivas en OSSIM

Una de las funcionalidades más destacadas de OSSIM es su capacidad de correlar eventos mediante directivas.
En el capítulo anterior se explicó un acercamiento general a este modelo de correlación que ahora se explicará con detalle.

OSSIM identifica a cada monitor y sensor con un identificador único, al igual que los plugins que estos soportan. Así, por ejemplo el sensor Snort tiene un id 1001 y sus plugins que detectan los diferentes ataquen oscilan desde el 1000 hasta el 2000. Sabiendo ésto y teniendo en cuenta que existen las etiquetas 'SRC_IP' y 'DST_IP' para referirse a las IPs origen y destino de un evento concreto, la generación de directivas es sencilla. La idea general es crear setencias, como si se tratarán de "if", pero con formato XML. Un ejemplo sería:

```
<directive id="3" name="correlacion NIDS_HIDS"  priority="5">
   <rule type="detector" name="nids" reliability="1"
       occurrence="1" from="ANY" to="ANY" port_from="ANY"
       port_to="ANY" plugin_id="1001" plugin_sid="ANY" >
       <rules>
           <rule type="detector" name="ALERTA OSIRIS"
           reliability="5"
           occurrence="1" from="ANY" to="ANY" port_from="ANY"
           ort_to="ANY" plugin_id="4001" plugin_sid="ANY"/>
       </rules>
   </rule>
</directive>
```

El nombre de la directiva, será el nombre con el que salte la alarma. El valor prioridad, indica como de peligrosa es la alerta. Las veces que se debe de generar una alerta para que se considere como tal, se indica mediante el valor de *ocurrence*.
De este modo la regla anterior define la correlación entre Snort y el HIDS Osiris, si se da el caso de que salta una alerta Snort seguidamente de una alerta en el HIDS, se fundirá en una única alarma que será la que se verá en el consola.

4.2. Diseño de la arquitectura propuesta

El sistema está formado por una máquina de correlación, que a su vez ejecuta un NIDS (Snort). Esta máquina lleva el propio motor de correlación que busca coincidencias con las directivas. Por otro lado está la máquina de pruebas que ejecutara un HIDS y los monitores de algunos servicios. Ambas máquinas están conectadas a través de un switch Cisco 2950.

Tanto las tramas que van dirigidas a la máquina de pruebas como las que salen de ésta, son replicadas mediante SPAN al puerto donde está conectada

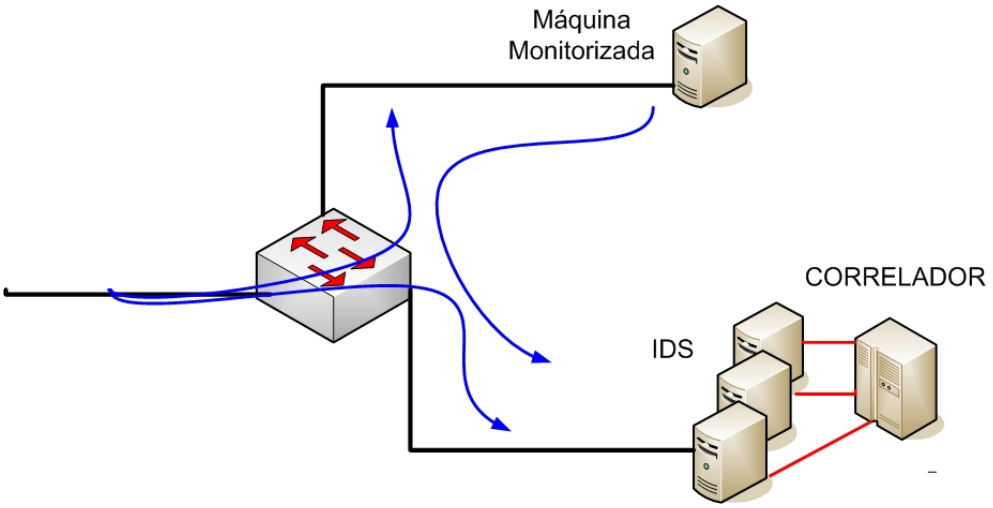

Figura 4.1: Topología de la red, con el conmutador configurado con SPAN (Switch Port Analyzer) haciendo duplicado de paquetes.

la máquina que ejecuta el Snort, así se puede analizar el tráfico en busca de patrones de ataques. Los ataques generan una alertas, que la máquina con correlación se encargará analizar y clasificar. Un dibujo de la arquitectura se puede ver en figura 4.1.

4.2.1. Switch Catalyst Cisco 2950

El conmutador Cisco que permitirá conectar las máquinas a la red de la universidad es un modelo Catalyst Cisco 2950 con dos puertos Gigabit Ethernet. Las características de este switch son:

- 24 puertos Fast ethernet
- 2 puertos Gigabit ethernet.
- Gestión remota mediante los protocolos: SNMP 1, SNMP 2, RMON 1, RMON 2, RMON, Telnet, SNMP 3, SNMP 2c.
- Estándares soportados: IEEE 802.3, IEEE 802.3U, IEEE 802.1D, IEEE 802.1Q, IEEE 802.3ab, IEEE 802.1p, IEEE 802.1w, IEEE 802.1x, IEEE 802.1s.

Esta gama de conmutadores permite configurarse para realizar lo que se conoce como replicado de tráfico o *mirroring*, aunque también se conoce como Switch Port Analyzer (SPAN). La idea es que el conmutador se configura para

que el tráfico de un interfaz se copie a otra, o incluso de varias interfaces se copien a una. Esto es útil para monitorización de tráfico. El conmutador funciona a nivel 2, por lo que se copian tramas de una puerto a otro.

4.2.2. Máquina con correlación - 'ircisco28.uv.es'

La máquina que ejecuta las aplicaciones necesarias como fuente de información de eventos y el sistema de correlación es un ordenador personal con las siguientes características:

- Procesador Pentium III a 1GHz
- 256 MB de memoria RAM
- Disco Duro de 18GB.
- Tarjeta de red Intel Corp. 82540EM Gigabit Ethernet.
- Dirección MAC: 00:0E:0C:34:AA:BE.
- Tarjeta de red 3Com Corporation 3c905C-TX/TX-M.
- Dirección IP: 147.156.222.109 (ircisco28.uv.es).
- Dirección MAC: 00:04:76:A1:61:38.
- Lector CD rom.

Aprovechando las dos tarjetas de red, configuraremos la máquina para capturar el tráfico y los eventos ocurridos en la red.

En relación a las aplicaciones y el software instalado en la máquina, ésta llevará instalado un sistema operativo Debian 3.3 con kernel 2.6.8. Está distribución lleva herramientas para instalar paquetes que la hacen muy amigable y de fácil administración.

Esta máquina no sólo hará la vez de IDS, si no que ejecutará más aplicaciones, con distinto objetivo. La herramienta central de está máquina será OSSIM, que integra desde IDSs, como sensores de red o bases de datos para almacenar los eventos.

4.2.3. Máquina de pruebas - 'labd.uv.es'

Las características hardware de la máquina que servirá como pruebas de los ataques son:

- Procesador Pentium III a 666MHz.
- 128MB de memoria RAM.
- Disco duro de 10GB.
- Tarjeta de red 3COM 3C905C-Tx.
- Dirección IP: 147.156.222.201 (labd.uv.es).
- Direccion MAC: 00:04:75:EF:29:45.

Al igual que en la máquina correladora, instalaremos un sistema operativo Linux Debian. La razón de utilizar esta distribución en la máquina víctima es porque es sencilla su gestión mediante la herramienta 'apt' y porque permite instalar desde los *sources* adecuados el agente 'Ossim-agent' que reportará la información de los eventos del host. Los servicios instalados no tienen fallos de seguridad descubiertos en la fecha de realización de las pruebas, por lo que se decidió instalar una aplicación web en PHP que permitía ejecutar código en la máquina.

En cuanto al software instalado en la máquina, se instala:

- Servidor SSH. OpenSSH 3.8.1p1
- OpenSSL version 0.9.7e.
- Apache 2.0.54-5.
- HIDS Osiris version 4.0.6-1.

Del servidor Apache podremos aprovechar los logs de acceso en busca de peticiones HTTP erróneas y de posibles intentos de ataques web como Cross Site Scripting, SQL injection, LDAP injection, intento de ejecutar CMD.exe, etc. Fundamentalmente buscaremos patrones en los logs de acceso y error (/var/log/apache/access.log y error.log). El *parser* de OSSIM para Apache, integrado en Ossim-agent, traducirá y enviará todos los logs al servidor OSSIM.
Osiris , alertará de las anomalías que ocurran en el host. Las alertas lanzadas

por Osiris indicarán que una intrusión se ha consumado. Osiris se configurará para que cada dos minutos chequee el host, y en caso de haber un cambio envíe una alerta a través del parser de OSSIM para Osiris integrado en Ossim-agent. Además para controlar en todo momento las intrusiones y no depender del propio OSSIM, se enviará un correo con los logs del cambio. Ésto es así para evitar que la máquina atacada sirva de plataforma para atacar otras máquinas.

4.2.4. Acercamiento mediante directivas al modelo general de correlación

El objetivo que se persigue es adaptar de la mejor manera posible el modelo de correlación explicado en el capítulo 3 sección 3.3 a la herramienta OSSIM.

1. Normalización: son los propios *parsers* o traductores de OSSIM los que traducen las alertas generadas por distintos sensores a un formato IDMEF estándar.

2. Preproceso: al igual que la fase de normalización, los *parsers* facilitan la homogeneización mediante el relleno de aquellos campos vacios o el cambio de los campos necesarios.

3. Fusión de alertas: esta fase, aunque el autor no ha podido realizarla por falta de más detectores de red o host, ya que solo se ha dispuesto de uno de cada tipo, sería tan sencillo como generar una directiva anidada donde se comprobarán si ha saltado uno o todos los sensores del mismo tipo, generando una directiva que los fundiría a modo de meta-alerta.

4. Verificación de las alertas: la fase de verificación OSSIM no la implementa como tal, pero si permite realizar mediante la herramienta Nessus un análisis de vulnerabilidades del activo. Así, para cualquier vulnerabilidad detectada, se generaría una regla que involucraría a un sensor (por ejemplo Snort) y a la firma correspondiente para explotar esa vulnerabilidad. Si un ataque coincide con el plugin asociado, automáticamente se generará una alerta.
   A esta correlación OSSIM la llama 'correlación cruzada'.

5. Reconstrucción del hilo de ataque y reconstrucción de la sesión de ataque: estas dos fases, fusionadas en una, serían la propia correlación con otras herramientas de seguridad, por ejemplo los logs de Apache.

Podría también relacionarse con los detectores de host, así se tendría una sucesión de eventos seguidos, que es básicamente lo que se pretende.

6. Reconocimiento del foco de ataque: esta fase se consigue, viendo qué conexiones se abren desde la máquina víctima hacia otras máquinas. Es justo en este punto donde herramientas como Ntop juegan un papel importante. Si la máquina comprometida es usada como plataforma de ataque a otros máquinas, al abrir conexiones TCP, o mandar paquetes ICMP con direcciones IP distintas, se detectará.

7. Correlación multipaso: en esta etapa, lo que principalmente habrá que detectar serán los intentos de escalar privilegios. Actualmente OSSIM no implemente ningún HIDS que compruebe las llamadas al sistema, como por ejemplo systrace, por lo que esta fase se ha obviado.

8. Análisis del impacto y priorización de alertas: esta fase depende mucho del inventario de red y de la valoración de los activos. OSSIM permite configurar a priori, mediante el inventario, el valor de todos los activos. Quizás, esta fase más que técnica, sera labor de consultoría, por ejemplo mediante análisis de riesgos.

   El detalle de las directivas generadas, se puede ver en el apéndice.

4.3. Resumen

Resumiendo el diseño, vamos a tener un sistema atacado que va a ser monitorizado por distintos sensores y monitores, centralizando todos los datos en un punto central y correlándolos. Tendremos una consola donde se verán los resultados de la correlación y un sistema capaz de medir el riesgo al que los activos han sido expuestos.

Capítulo 5

Implantación del sistema

5.1. Preparación de la máquina de control

Una vez decidido las herramientas a utilizar para implantar el sistema de detección de intrusos con correlación, se seguirá describiendo los pasos necesarios para instalar y configurar todo el sistema.

El sistema operativo sobre el que se va a ejecutar las herramientas y la plataforma de correlación será un sistema Linux.
Antes de poner la máquina en producción y recién instalado el SO se configurará el firewall que incorpora el núcleo de Linux, *iptables*, para asegurar la máquina del exterior. Después se instalarán todos los paquetes y aplicaciones necesarios para poder trabajar con el servidor de manera cómoda y para que éste realice todas las operaciones necesarias.

5.1.1. Instalación del sistema operativo

El sistema operativo escogido para trabajar en la máquina es la distribución Linux *Debian 3.1*, con la versión del kernel 2.6.8-2.36 que soporta de manera nativa iptables. El sistema de ficheros está particionado como se muestra en la tabla 5.1.

Partición	**Punto de montaje**	**Tipo**	**Tamaño**
/dev/hda1	/	ext3	4GB
/dev/hda2	swap		600MB
/dev/hda3	/home	ext3	8GB
/dev/hda4	/var	ext3	5,4GB

Cuadro 5.1: Sistema de ficheros de la máquina de correlación

5.1.1.1. Configuración de las tarjetas de red

La máquina que ejecutará Snort tiene dos interfaces de red como se vió en el capítulo anterior. La primera tarjeta de red (eth0) dispondrá de una IP[1] pública con el fin de poder gestionar la máquina de manera remota y que pueda recibir información de los sensores instalados en otras máquinas. La segunda tarjeta de red no dispondrá de IP y únicamente estará levantada el interfaz para poder recibir todo el tráfico del conmutador copiado por SPAN. Snort puede analizar el tráfico de una interfaz sin dirección IP.

1. ifconfig eth0 147.156.222.109 netmask 255.255.254.0
2. ifconfig eth0 up
3. route add default netmask 0.0.0.0 gw 147.156.156.222.1
4. ifconfig eth1 up

5.1.1.2. Configuración del firewall (iptables)

La configuración del firewall permite:

- Cualquier tráfico ICMP.
- El tráfico de la máquina que se monitoriza (labd.uv.es) o que tienen un *agent* ejecutándose con monitores o sensores.
- Cualquier tráfico desde cualquier origen al puerto 22 para gestión remota, 80 para ver los resultados de la correlación por web y 3000 para ver los resultados de un sensor concreto que se explicará más adelante.
- Logear todos los intentos de conexión a cualquier otro puerto y rechazar la conexión.

La configuración detallada de iptables se puede ver en el apéndice B.

5.1.1.3. Securización de la máquina

Como medidas adicionales para asegurar la máquina se denegó el acceso mediante SSH a root. En el fichero de configuración /etc/ssh/sshd-config se añadió la línea:

```
PermitRootLogin no
```

[1]La máquina tiene como nombre de host ircisco28, que corresponde a ircisco28.irobot.uv.es

5.1.2. Configuración del conmutador con Switch Port Analyzer (SPAN)

El conmutador LAN se ha configurado para que haga mirroring de cualquier puerto menos el primero, que va conectado a la roseta de la red de robótica. Se tomó la decisión de hacer SPAN de todos los puertos, con el fin de poder monitorizar más de una máquina. Es tan sencillo como conectar una nueva máquina a algún interfaz libre y poder así ver el tráfico por la boca Gigabit Ethernet 0.
Los comandos ejecutados en la consola del switch son:

```
# monitor session 1 source interface Fa0/2 - 23
# monitor session 1 destination interface Gi0/1
```

5.1.3. Instalación de OSSIM

Para la parte del servidor, se instalará Ossim-server, Ossim-agent y Ossim-framework para gestionar la herramienta mediante interfaz web .

La herramienta OSSIM se puede instalar de diversas maneras, desde el propio código fuente o desde los binarios que hay para Debian y Fedora. Así pues, se instalará mediante la herramienta 'apt' de Debian para el manejo de paquetes.

- Copiar los links de los binarios al fichero de configuración de manejo de paquetes para la herramienta apt.

  deb http://security.debian.org/ sarge/updates main
  deb http://ftp.debian.org/debian/ sarge main
  deb http://www.ossim.net/download/ debian/

- Actualizar la lista de paquetes:

  ```
  # apt-get update
  ```

5.1.3.1. Instalación de la base de datos MySQL

La base de datos MySQL contiene las alertas almacenadas por los distintos sensores, y todos los datos almacenados por OSSIM para la correlación (topologías de red, activos, puertos abiertos en cada host, resultados del Nessus, etc). Para instalar la base de datos y crear las tablas necesarias:

```
# apt-get install ossim-mysql
# mysqladmin -u root password XXXXXXXX
# mysql -u root -p

mysql> create database ossim;
mysql> create database ossim_acl;
mysql> create database snort;
mysql> exit;
```

y para crear las tablas que debe haber en cada base de datos:

```
#   zcat
/usr/share/doc/ossim-mysql/contrib/create_mysql.sql.gz \
/usr/share/doc/ossim-mysql/contrib/ossim_config.sql.gz \
/usr/share/doc/ossim-mysql/contrib/ossim_data.sql.gz \
/usr/share/doc/ossim-mysql/contrib/realsecure.sql.gz | \
mysql -u root ossim -p

# zcat
/usr/share/doc/ossim-mysql/contrib/\
create_snort_tbls_mysql.sql.gz \
/usr/share/doc/ossim-mysql/contrib/
\create_acid_tbls_mysql.sql.gz \
| mysql -u root snort -p
```

Con esto ya se tienen creadas la bases de datos y las tablas de cada una de ellas para que las distintas partes de OSSIM puedan almacenar los datos.

5.1.3.2. Instalación de Ossim-server

El corazón de la aplicación de OSSIM es precisamente la parte que hace de servidor. Ésta controla todos los sensores remotos o locales, realiza la correlación, accede a las bases de datos, y unas cuantas funciones más. Su instalación es sencilla, aunque depende de los sensores con los que se quiera trabajar.

```
# apt-get install ossim-server
```

se introduce cuando se esté instalando el paquete, los parámetros de conexión a la base de datos, como son el host y puerto, nombre de la base de datos, usuario y contraseña. Está configuración siempre se puede modificar a través del fichero de configuración de ossim-server /etc/ossim/server/config.xml.

5.1.3.3. Instalación de Ossim-agent

El proceso de instalación del agent de Ossim se repetirá en cada uno de los hosts que integren un sensor y/o monitor. En función de los sensores y monitores con los que funcione se configura de una u otra manera.

```
# apt-get install ossim-agent
```

5.1.3.4. Instalación de Ossim-framework

La parte framework del OSSIM es un portal escrito en PHP que facilita la configuración de toda la herramienta. Sus funciones son:

- Panel de control de todo el sistema.
- Visualización del riesgo al que ha sido expuesto y el uso del sistema.
- Consola forense.
- Configuración de todos los sensores y monitores.

Para instalar el framework es necesario tener instalado la aplicación phpgacl [33], que sirve para gestionar la lista de usuarios y los permisos para acceder al interfaz web Ossim-framework. De esta manera los pasos para instalar el framework son:

```
# apt-get install phpgacl
# apt-get install ossim-framework
```

Como paquetes adicionales, siendo dependencias del Ossim-framework, se instalan los paquetes Apache, php4, y los módulos para Apache que permiten atacar la base de datos MySQL mediante PHP.
Cuando finaliza la instalación de estos paquetes se puede acceder a gestionar OSSIM mediante el interfaz web en la dirección http://localhost/ossim, un ejemplo del interfaz de acceso se ve en la imagen 5.1.

5.1.3.5. Instalación de OSSIM y sus utilidades

Por último, se instala el paquete propio OSSIM y un conjunto de utilidades que incluyen ciertas funciones para OSSIM.

```
# apt-get install ossim-utils
# apt-get install ossim
```

Figura 5.1: Interfaz de accesso a OSSIM

5.1.3.6. Instalación de los monitores y sensores

OSSIM puede integrar diferentes sensores y monitores, desde productos comerciales hasta herramientas de libre distribución. Además cada una de ellas con una función específica en el conjunto total de la herramienta OSSIM. Las que se van a utilizar son :

SNORT

Snort funciona almacenando las alertas en la base de datos MySQL. Además para facilitar la visualización de las alertas se instalará un frontend escrito en PHP, llamado ACID (Analysis Console for Intrusion Databases [34]). Esto se hace de manera automática al encontrar en la herramienta 'apt' dependencias entre paquetes.

```
apt-get install snort-mysql
```

Ntop

Ntop [35] es un analizador de red que muestra el uso de la red. Es análogo al programa *top* que muestra todo lo relativo a procesos en una máquina UNIX, pero en versión para red. Ntop se basa en la librería *libpcap* [36].
Los usuarios de Ntop pueden usar el navegador para visualizar la información que Ntop reporta, es decir una instantánea de la red. Ntop puede verse como una sonda Rmon con un interfaz web, que funciona bajo un servidor web.

Entre las funciones de Ntop están:

- Ordenar el tráfico en función del tipo de protocolo (ICMP, UDP, TCP, IGMP..).
- Mostrar el tráfico en función de varios criterios.
- Mostrar estadísticas de uso en forma de gráfica.
- Identificar los sistemas operativos y las fuentes del tráfico

En el caso de este proyecto, Ntop es una herramienta que servirá para detectar el tráfico anómalo, por ejemplo si un host ha sido infectado por un gusano el cual hace que abra muchas conexiones (o intentos) a puertos concretos de otras máquinas, o por ejemplo exceso de tráfico en cuanto a cantidad.
La instalación de Ntop se lleva a cabo mediante el siguiente comando:

```
# apt-get install ntop
```

Acto seguido, se debe de crear lanzar la aplicación así:

```
# ntop -u ntop
# /etc/init.d/ntop start
```

Nessus

La herramienta Nessus es un auditor de sistemas que prueba distintos exploits para los servicios que detecta activos en una red. Para instalar Nessus en Debian, lo haremos de la rama *unstable*, ya que sino, no se dispondrá de la última versión. Además, ahora para poder descargarse las firmas o plugins, hay que registrarse en la página web de Nessus y obtener una licencia. Se instalará tanto la parte servidora como el cliente.

```
# apt-get install nessus
# apt-get install nessus
# nessus-update

# nessus-adduser

#/etc/init.d/nesssud start
```

Por último, habrá que añadir un usuario para poder lanzar Nessus.

5.1.3.7. Configuración de OSSIM en el correlador

Una vez instalada toda la parte servidora de OSSIM, se tiene que configurar para que todos los elementos del sistema se integren debidamente. En cuanto al servidor, habrá que configurar el fichero:
/etc/ossim/server/config.xml

- Direción ip del sevidor: 147.156.222.109.
- Interfaz de escucha: eth1.
- Base de datos: usuario, password, nombre de la base de datos, host:puerto y tipo de base de datos.
- Fichero de log donde se almacenarán los eventos: /var/log/ossim/server.log.
- Ficheros de las directivas: generic.xml.

La parte del framework requiere configurar:

- Directorio base de OSSIM: /usr/share/ossim
- Base de datos que usa OSSIM.
- Usuario y directorio de snort.
- Usuario y directorio de Nessus.

Y por último la parte del agent, se debe de configurar en función de los plugins:

- Plugin Snort.
- Plugin Ntop.

5.2. Preparación de la máquina de pruebas

5.2.1. Instalación del sistema operativo

La máquina de pruebas también tendrá un sistema Linux Debian con la misma versión del kernel, siendo la rama elegida la estable de Debian. El sistema de ficheros de está máquina queda configurado de la siguiente manera:

Para configurar la tarjeta de red haremos:

1. ifconfig eth0 147.156.222.201 netmask 255.255.254.0.

Partición	Punto de montaje	Tipo	Tamaño
/dev/hda1	/	ext3	2GB
/dev/hda2	swap		500MB
/dev/hda3	/home	ext3	4GB
/dev/hda4	/var	ext3	3,5GB

Cuadro 5.2: Sistema de ficheros de la máquina de pruebas

2. ifconfig eth0 up.

3. route add default netmask 0.0.0.0 gw 147.156.156.222.1.

5.2.2. Instalación Apache

Para poder tener como fuente de información los logs de Apache se tiene que instalar el servidor web Apache. Se va a instalar la versión 1.3.33, junto con los módulos necesarios para poder instalar aplicaciones en PHP que accedan a bases de datos mysql.

```
# apt-get update
# apt-get mysql
# apt-get install apache php4-mysql libapache-mod-php
```

La razón por la cual se instala una base de datos, es para poder instalar un portal escrito en PHP (php-nuke[50]), el cual es vulnerable a ataques de 'SQL Injection', 'Cross Site Scripting', robo de sesión, etc. con el fin de poder comprobar el funcionamiento de la herramienta de correlación.

Por otro lado, los logs de apache quedan almacenados por defecto en: /var/log/apache/access.log .

5.2.3. Instalación de Ossim-agent

La instalación de Ossim-agent es análoga a la que se ha hecho para la máquina correladora.

```
# apt-get install ossim-agent
```

En este caso, los sensores que se configurarán serán syslog, Apache , y Osiris.

syslog es útil para monitorizar el fichero auth.log, con los intentos fallidos de acceso mediante ssh a la máquina.

5.2.4. Instalación de Osiris

Como se habló en el capítulo anterior, el HIDS que se instalará será Osiris. Osiris monitoriza todos cambios en los ficheros de un host. Además cualquier modificación de los usuarios o grupos del sistema serán detectados por está herramienta.

La herramienta osiris se instalará de ficheros fuente. Una vez descargado y descomprimido el fichero fuente:

```
# ./configure
# make
# make install
```

Para compilar la consola de osiris:

```
# make console
# ./install.sh
```

Y el agente de monitorización:

```
# make console # ./install.sh
```

Una vez instalado las tres partes que forman osiris, hay que configurar la herramienta.
Inicialmente hay que configurar la parte que hará de servidor:

```
# osiris

osiris Osiris Shell Interface - version 4.1.9 unable to load root
certificate for management host:
(/Users/brian/.osiris/osiris_root.pem)
 >>> fetching root certificate from management host (localhost).

The authenticity of host 'localhost' can't be established.

  [ server certificate ]

 subject = /C=US/CN=Osiris Management Osiris Host Integrity System
 issuer  = /C=US/CN=Osiris Management Osiris Host Integrity System



Verify the fingerprint specified above. Are you sure you want to
```

```
continue connecting (yes/no)? yes
 >>> authenticating to (localhost)

User: admin Password:

connected to management console, code version (4.1.9). hello.


osiris-4.1.9

[ edit management host (localhost) ]

  > syslog facility [DAEMON]:
  > control port [2266]:
  > http host name (uses system name by default) []:
  > http control port [2267]:
  > notify email (default for hosts) []:
  > notification smtp host [127.0.0.1]:
  > notification smtp port [25]:

  > authorized hosts:

   127.0.0.1

  Modify authorization list (y/n)? [n] n

[ management config (localhost) ]

syslog_facility = DAEMON control_port = 2266 http_port = 2267
http_host = notify_email = notify_smtp_host = 127.0.0.1
notify_smtp_port = 25 hosts_directory = allow = 127.0.0.1

Is this correct (y/n)? y
 >>> management host configuration has been saved.
osiris-4.1.9:
```

Los campos que se configuran son:

- Syslog facility: funcionará como si se tratara de un daemon como Syslog.
- Control port: puerto de conexión.

- Http control port: puerto de control si se quiere acceder mediante el protocolo HTTP.
- Notify email: dirección de correo si se configura para que envíe correos cuando de detecte alguna anomalía.
- Notification smtp host: servidor SMTP al que conectarse para enviar los correos.
- Notification smtp port: puerto de conexión, normalmente el 25.
- Authorized hosts: Máquinas que se podrán conectar para mandar sus logs.

Cuando se instala Osiris, se generan un par de claves mediante OpenSSL[49], para realizar la conexión segura entre los hosts con Osiris y el servidor.

Por último hay que configurar en Osiris qué configuración afecta a cada host. Para ello desde el la línea de comandos se deberá de ejecutar 'osiris':

```
osiris-4.0.5-release: new-host

[ new host ]

  > name this host []:  labd
  > hostname/IP address []: 127.0.0.1
  > description []: management console agent
  > agent port [2265]:
  > enable log files for this host? (yes/no) [no]:

Scan Databases:

    => keep archives of scan databases?  (yes/no) [no]:

    => auto-accept changes?  (yes/no) [yes]:

    => purge database store? (yes/no)  [yes]:

Notifications:

    => enable admin email notification
    for this host? (yes/no) [no]:
    => send notification on scheduled
```

```
    scans failures? (yes/no) [no]:
    => send scan notification, even when
     no changes detected  (yes/no) [no]:
    => send notification when agent
    has lost session key  (yes/no) [no]:

Scheduling:

  > configure scan scheduling information? (yes/no) [no]: yes

    [ scheduling information for labd ]


    enter scan frequency in minutes: [1440]

  > activate this host? (yes/no) [yes]:


Is this correct (y/n)? y
 >>> new host (labd) has been created.

initialize this host? (yes/no): yes

Initializing a host will push over a configuration, start a scan,
and set the created database to be the trusted database.

Are you sure you want to initialize this host (yes/no): yes

OS Name: Darwin OS Version: 8.0.0b1

use the default configuration for this OS? (yes/no): y
 >>> configuration (default.darwin) has been pushed.
 >>> scanning process was started on host: labd
```

La parte más importante de está configuración, es cuando se aplica la política según el SO. Según ésta, se monitorizará unos ficheros u otros.

5.2.5. Configuración de Ossim-agent

La parte que se refiere al agent de la máquina de pruebas, no requiere configuraciones complejas. Tan sólo se ha de indicar qué plugins deben de

lanzarse, que en éste caso serán tres: syslog, Osiris y Apache. Para cada uno de ellos habrá un fichero de configuración que indicará de donde leer los eventos y cómo se llama el proceso para controlar su estado en todo momento, estos ficheros se encuentran ubicados en el directorio de configuracion de Ossim-agent, /etc/ossim/agent/.

La información relevante a configurar en el fichero syslog.xml es:

- Nombre del proceso cuando se esté ejecutando: syslogd.
- Tipo de sensor: monitor.
- Arranque y parada del monitor: /etc/init.d/syslog start—stop
- Fichero de logs: /var/log/auth.log

Para el fichero de configuración Apache, apache.xml:

- Nombre del proceso cuando se esté ejecutando: apache.
- Tipo de sensor: monitor.
- Arranque y parada del monitor: /etc/init.d/apache start—stop.
- Fichero de logs: /var/log/apache/access.log

Por último, para el fichero de configuración ossim.xml:

- Nombre del proceso cuando se esté ejecutando: osirisd.
- Tipo de sensor: detector.
- Arranque y parada del monitor: /etc/init.d/osirisd start—stop.
- Fichero de logs: /var/log/syslog

Además para todos ellos las opciones por defecto 'start' y 'enabled' deberán estar al valor 'yes', con el fin de que por defecto los sensores estén activos.

5.2.6. Modificación del parser ParserOrisis.py

Los traductores que Ossim instala por defecto para cada sensor no están configurados para que reporten la información correctamente, a excepción del de Snort, Apache y Syslog.
Estos scripts escritos en Python[45], hay que modificarlos adaptándolos a las necesidades de cada caso concreto.

Python es un lenguaje muy útil para interactuar con la shell en sistemas UNIX, además maneja expresiones regulares con la misma facilidad que el lenguaje Perl.

Para el fin que se seguía en este proyecto, se pretendía monitorizar todos los eventos ocurridos en un host, y no sólo la modificación de ficheros que traía por defecto el parser. Así, se tuvo que ver todos los eventos que osiris lanzaba y contemplarlos en el parser.

Incialmente, solo se tenían en cuenta los eventos que osiris etiquetaba como 'cmp' de 'comparation', así se tuvo que añadir que se enviará información para lo otros eventos posibles:

```
if type == "cmp" or type == "missing" or type == "info" or type ==
"error" or type=="warning":
```

5.2.7. Instalación de un portal en PHP vulnerable

Con el fin de poder atraer el interés de los hackers, se instaló una versión antigua de PHP-Nuke, que permite realizar ataques XSS, SQL-Injection. La versión elegida fue la 5.5.
Una vez descargado el paquete PHP-Nuke-5.5.tar.gz, se instaló de la siguiente manera:

```
tar -xvfz PHP-Nuke-5.5.tar.gz #descomprimir
 mysqladmin create nuke #Creación de la base de datos
 mysql nuke < nuke.sql #Creación de las tablas
```

Así, ya se tenía el portal perfectamente instalado y configurado, accesible desde la web http://labd.uv.es

5.3. Configuración y adaptación del sistema

Uno de los problemas con los que el autor se encontró con OSSIM, es la falta de documentación.

Aunque OSSIM es una herramienta Open Source, no existe demasiada información de como configurar la herramienta de manera precisa. Existen unas directrices básicas de como instalar el esqueleto del sistema, siendo insuficientes para un correcto funcionamiento del sistema.

5.3.1. Snort

Snort por defecto configura todas las reglas, así, se genera demasiados falsos positivos. Las reglas que fueron eliminadas, principalmente han sido:
Falsos positivo en el que está implicado tráfico multicast.

```
rules/bad-traffic.rules:#alert ip any any -> any any
(msg:"BAD-TRAFFIC IP Proto 103 PIM"; ip_proto:103;
reference:bugtraq,8211; reference:cve,2003-0567;
classtype:non-standard-protocol; sid:2189; rev:3;)
```

y las reglas de portscan que generaban gran número de falsos positivos:
portscan: TCP Portsweep
portscan: IP Protocol Sweep
portscan: UDP Portsweep

5.3.2. Nessus

Para usar Nessus, integrado con Ossim, se tiene que actualizar las reglas de manera manual y configurar el fichero para lanzar Nessus desde Ossim con los parámetros adecuados.
Para ello, es necesario modificar las variables que indica el login y password en el servidor nessusd, así como el puerto, host y los directorios donde almacenar los resultados:

```
my $nessus = "/usr/bin/nessus";
my $nessus_user = "angel";
my $nessus_pass = "------";
my $nessus_host = "localhost";
my $nessus_port = "1241";
my $nessus_rpt_path = "/usr/share/ossim/www/vulnmeter/";
my $nessus_distributed = "0";
my $nessus_tmp = $nessus_rpt_path . "tmp/";
```

Solo destacar que la variable 'nessus_distributed' indica si se quiere lanzar nessus de manera distribuida.

5.3.3. Configuración a través del framework de OSSIM

La finalidad de este punto, es adaptar Ossim a cada topología de red, para ello desde el framework hay que indicar todo lo relativo a los activos, a la red, a los sensores, monitores y a las reglas de correlación.

Figura 5.2: Menú de políticas de OSSIM

En la figura 5.2 se puede ver el aspecto del menú de configuración de políticas de OSSIM, donde se configura:

- Hosts: Se indica la IP y nombre del host, el valor del activo, si se escaneara con Nessus[27] y los hosts de los sensores que le monitorizaran.
- Networks: para configurar subredes o topologías de red.
- Networks groups: un grupo de redes serán varias redes definidas anteriormente.
- Sensors: sensores existentes. Se indicará el puerto de conexión, su importancia (fiabilidad).
- Ports: puertos importantes a tener en cuenta. Se puede descubrir de manera automática mediante Nmap[23].

Por otro lado la configuración de las opciones principales de OSSIM se hace a través del menú de configuración 'configuration'. Los parámetros a configurar son:

- Usuario y contraseña de acceso al portal OSSIM
- Opciones generales de Snort, MySQL, plugins, etc.

5.4. Resumen

En este capítulo se ha descrito los pasos necesarios para la instalación del sistema diseñado en el capítulo 4, la configuración de los diferentes sensores y la puesta en marcha del sistema.
Se ha descrito las fases de la implantación del sistema, desde una primera fase de instalación del sistema operativo hasta una fase final de integración de las herramientas y aplicaciones que son el esqueleto del sistema propuesto.

Capítulo 6

Resultados

6.1. Pruebas realizadas

Durante el proyecto se han realizado diferentes pruebas, para comprobar el funcionamiento de determinadas partes del sistema. La primeras pruebas se hicieron para ver el funcionamiento correcto de cada sensor.

6.2. Análisis del tráfico con Snort sobre 'glup.uv.es'

El conmutador, al estar monitorizando desde el interfaz 2 al 23, permitía analizar el tráfico de cualquier host que se conectara a una de estas bocas. Antes de conectar la máquina de pruebas, labd.uv.es, se realizaron pruebas sobre el tráfico que se generaba hacia el servidor de robótica 'glup.uv.es'. Este servidor genera tráfico en un día normal de 70GB ó 80GB, ya que no sólo hace de servidor de correo y servidor web para el dominio irobot.uv.es, sino que además tiene otro dominio vinculado, cdlibre.org, que necesita otros servicios, como FTP. Un informe completo del snort se puede ver en la tabla 6.1

Alerta	**Ocurrencias**
(portscan) Open Port	30 %
(portscan) TCP Portscan	25 %
(http_inspect) IIS UNICODE CODEPOINT ENCODING	23 %
(http_inspect) BARE BYTE UNICODE ENCODING	22 %
(http_inspect) DOUBLE DECODING ATTACK	10 %

Cuadro 6.1: Resumen de las alertas de Snort sobre la máquina en producción 'glup.uv.es'

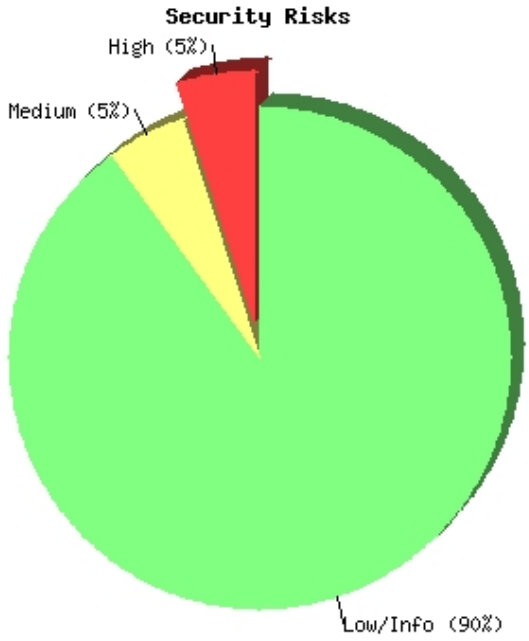

Figura 6.1: Resultados del análisis de vulnerabilidades sobre la máquina en producción 'labd.uv.es'

6.3. Análisis de vulnerabilidades

Para comprobar el correcto funcionamiento de la herramienta Nessus, integrada en OSSIM, se realizaron varios análisis de vulnerabilidades sobre las máquinas labd.uv.es y glup.uv.es.

En el host glup.uv.es, no fueron identificadas ninguna vulnerabilidad grave. La máquina está bien securizada y bastionada (harndering), ya que no tiene activo ningún servicio innecesario y los servicios activos están bien configurados.

En labd.uv.es, se creo un *cgi-bin* (phf) para que saltarán las alertas y la máquina fuera de interés para los atacantes. Nessus, detecto está vulnerabilidad, aunque el CGI había sido habilitado de manera que no hiciera nada, si se detectaba su presencia. Los resultado obtenidos para la máquina labd.uv.es se puede ver en la figura 6.1

Como se ve en la figura 6.1, se tiene un 5 % de riesgo importante, esto es debido principalmente al CGI instalado para ser atractivo a los intrusos.

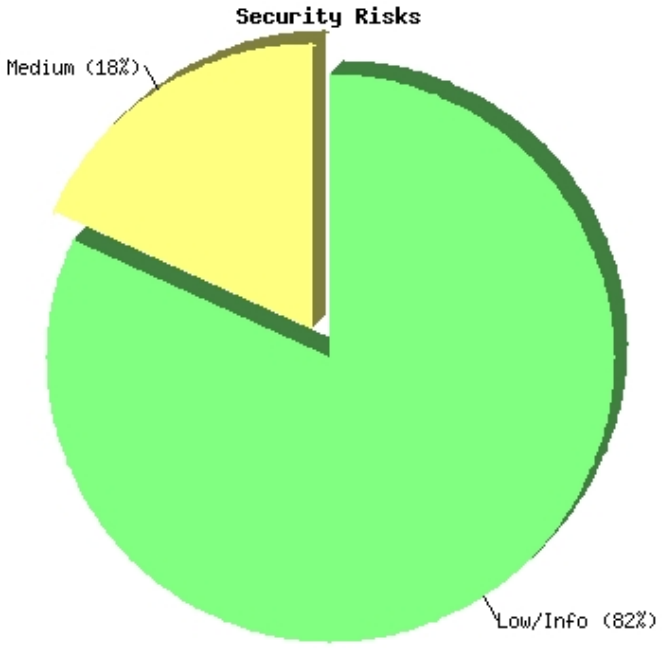

Figura 6.2: Resultados del análisis de vulnerabilidades sobre 'glup.uv.es'

La información que reporta el escáner de vulnerabilidades para ese bug, es la siguiente:

```
Vulnerability found on port www (80/tcp)

      The 'phf' CGI is installed. This CGI has
      a well known security flaw that lets an attacker
      execute arbitrary commands with the privileges of
      the http daemon (usually root or
      nobody).

      Solution : remove it from /cgi-bin.

      Risk factor : High
      CVE : CVE-1999-0067
      BID : 629

      Nessus ID : 10176
```

Rules (Directive 22)

	Name	Priority	Reliability	Time_out	Occurrence	From	To	Port_from	Port_to	Plugin ID	Plugin SID
-	APACHE		1		1	ANY	ANY	ANY	ANY	apache (1501)	ANY
-	LOGS NIDS		5		1	1:SRC_IP	ANY	1:SRC_PORT	ANY	snort (1001)	ANY
	ALERTA OSIRIS		1		1	ANY	ANY	ANY	ANY	osiris (4001)	ANY

Figura 6.3: Directivas en OSSIM

6.4. Pruebas de correlación

Durante el tiempo en que el servidor 'labd.uv.es' (máquina trampa) estuvo funcionando, Snort generó miles de alertas, el día 1 de noviembre de 2005 había un total de unas 10.000 alertas, entre los que había cientos de escaneos de puertos, intentos de peticiones HTTP ilícitos, y de ataques o exploits a servicios como SSH o Apache.
Nuestro sistema de correlación redujo el número a 50, es decir en una relación de 200 a 1. Como se vio en el capítulo 4, OSSIM muestra el resultado de los eventos correlados (alertas) como alarmas, es decir los 10.000 eventos son las alertas y los 50 resultados finales son las alarmas.

Dentro del total de alarmas, existen varias clasificaciones en función de la fiabilidad de éstas y del grado de compromiso del sistema. Se puede ver una captura en la imagen 6.4. Como se aprecia en la imagen, se observa la valoración o fiabilidad de la alerta expresada numéricamente y con color rojo en caso de haber sido comprometidos. Estos valores vienen dados por las directrices de correlación, las que están en verde con número menor de 5, indican que se ha llegado hasta cierto grado de anidamiento de la regla, por ejemplo en el caso de que dos ataques hayan coincido con dos reglas de dos sensores distintos. Se puede ver un ejemplo de como se visualizan las reglas de correlación en OSSIM en la imagen 6.3

Las alertas que indican un valor 10, muestran que han coincido con alguna regla que tienen como fiabilidad el valor máximo 10, que será cuando el HIDS lance algún evento.
Como se instaló un aplicación PHP con ciertos bugs, securizando y bastionando el resto de servicios, se monitorizó todos los eventos que el HIDS lanzará para el directorio /var/www y el usuario www-data. Además, se recibió una alerta sospechosa, que lanzó Osiris, sobre la aparición de cierto fichero sospechoso en /var/www/. El correo enviado por osiris decía lo siguiente:

```
Asunto:  [osiris log][host: labd][1 changes] Para: apan@alumni.uv.es
De:      "Osiris Host Integrity System" <osirismd@labd> Fecha: Wed
```

11	correlacion APACHE-NIDS-HIDS CONEXION VICTIMA A ATACANTE	10	147.156.222.109 127.0.0.1 labd	2005-1
12	correlacion APACHE-HIDS	10		
13	correlacion APACHE-NIDS-HIDS-CONEXION-DESDE_VICTIMA	10	labd 147.156.222.109 127.0.0.1	2005-1
14	correlacion APACHE-NIDS-HIDS CONEXION VICTIMA A ATACANTE	10	labd 147.156.222.109 127.0.0.1	2005-1
15	correlacion APACHE-HIDS	10		
16	correlacion NIDS HIDS	10		
17	correlacion APACHE-HIDS	10		
18	correlacion APACHE-HIDS	10		
19	correlacion NIDS HIDS	10		
20	correlacion APACHE-NIDS-HIDS	3		
21	correlacion APACHE-NIDS	8		
22	correlacion APACHE-NIDS	8		
23	correlacion APACHE-NIDS-HIDS	3		
24	correlacion APACHE-NIDS-HIDS	3		
25	correlacion APACHE-NIDS-HIDS	3		
26	correlacion APACHE-NIDS-HIDS	3		
27	correlacion APACHE-NIDS-HIDS	3		

Figura 6.4: Extracto de las alarmas 11 a la 27 de los resultados de la correlación

```
Oct 26 16:29:04 2005

compare time: Tue Nov  1 16:29:04 2005
             host: labd
      scan config: default.linux (4428ac88)
         log file: 9895
    base database: 75
 compare database: 76

[211][labd][cmp][/var/www][mtime][Mon Oct 01 00:52:55 2005][Wed Oct
26 16:29:04 2005]  [213][labd][cmp][/var/www][ctime][Mon Oct 01
00:52:55 2005][Wed Oct 26 16:29:04 2005]
[203][labd][new][/var/www/color.php]


Change Statistics:
----------------------------------

        checksums: 0
       SUID files: 0
 root-owned files: 1
```

```
 file permissions: 0
              new: 1
          missing: 0

total differences: 1
```

El fichero en cuestión subido, color.php, es una aplicación hecha en PHP que en realidad se llama phpRemoteView y que permite tener control total sobre el sistema de ficheros, aunque solo con privilegios del usuarios www-data. Además, entre alguna de sus funciones, permite lanzar comandos bash.

Una vez se tenía shell, para evitar posibles escaladas de privilegios en la máquina se decidió desactivar el portal en PHP. Además se procedió a quitar el programa PHP que daba acceso al sistema de ficheros.

Dado que el propio HIDS no avisó de ninguna alerta más, no fue necesario realizar el análisis forense del sistema.

Por otro lado, no se ha podido comprobar las reglas de correlación que afectan a más sensores como Ntop, ya que desde la máquina labd.uv.es no se ha realizado ningún intento de ataque a otros hosts.

6.4.1. Métricas de OSSIM

Respecto al valor de las variables A y C que miden el nivel de compromiso y ataque del sistema se puede ver una evolución de las mismas a través de las figuras 6.5, 6.6, 6.7.

En la primera de ellas, se ve que el sistema está el 95 %, luego pasa a estar al 89 %, siendo al final de 44 %. Estos cálculos los ha ido haciendo OSSIM a través del algoritmo CALM. Conforme el sistema recibía ataques, los valores de estas variables se iban incrementando.

El resumen de las gráficas anteriores, se puede ver en la figura 6.8

6.5. Resumen

En este capítulo, se han mostrado las pruebas realizadas con OSSIM y las herramientas que integra, como Snort y Nessus. La herramientas que realiza el test de vulnerabilidades se integra perfectamente en la herramienta OSSIM.

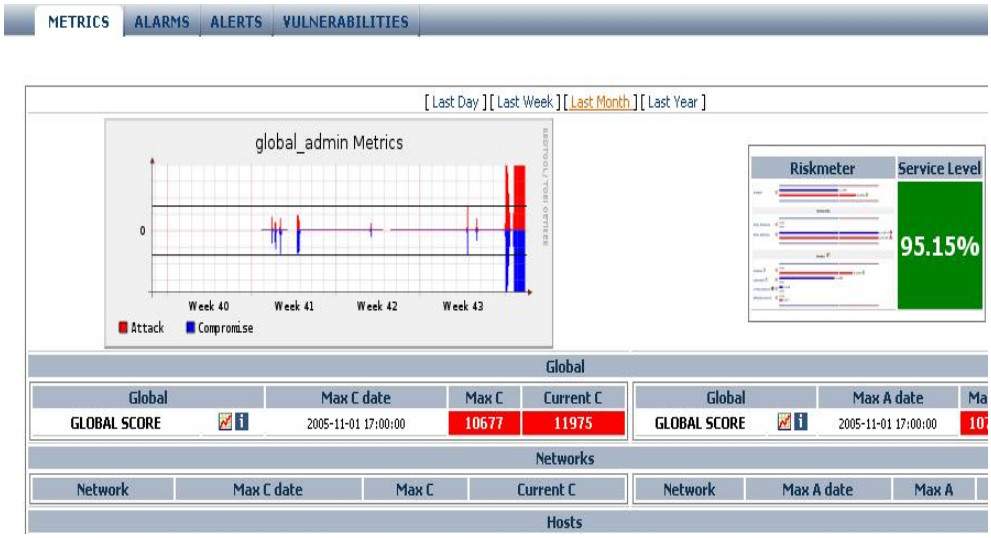

Figura 6.5: Valores del estado y servicio en el mes anterior

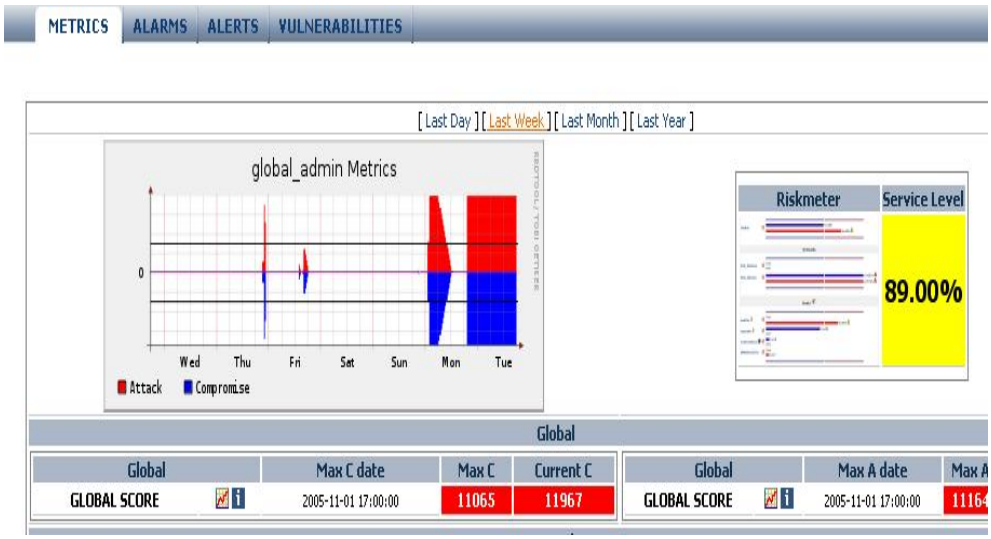

Figura 6.6: Valores del estado y servicio en la semana anterior

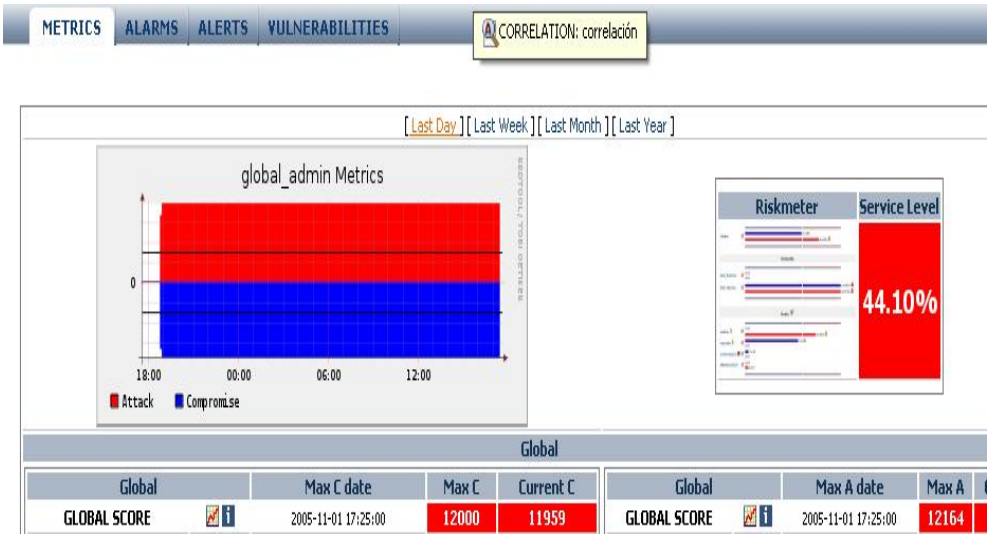

Figura 6.7: Valores del estado y servicio en el día anterior

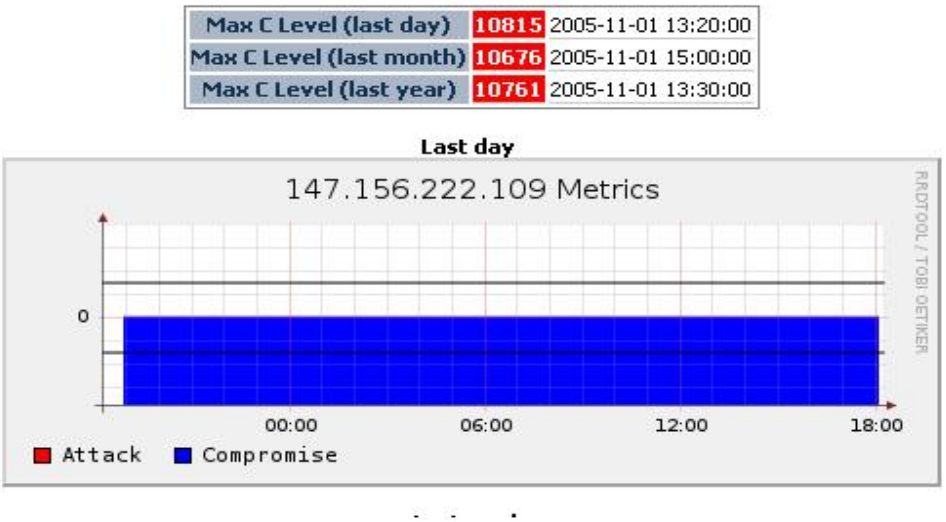

Figura 6.8: Resumen del valor de C o compromiso en OSSIM

Por otro lado la información que proporciona Snort junto con el HIDS Osiris y un tercer elemento, logs de Apache, se correla de manera efectiva obteniendo los resultados esperados. Las alertas son reducidas de manera sustancial en una proporción 1:200.

Capítulo 7

Conclusiones

7.1. Revisión de objetivos

Después de la implementación del sistema y una vez obtenidos los resultados, vamos a revisar en qué medida se han cumplido los objetivos marcados en la sección 1.3:

1. Se ha diseñado un sistema de detección de intrusos con correlación de eventos.

2. Evaluación de diferentes herramientas código libre, integrándolas entre sí, obteniendo una perfecta cohesión entre ellas.

3. Funcionamiento e integración de OSSIM indexOssim [42] tanto en la parte servidora como en los agentes, así como la integración con más herramientas de seguridad.

4. Desarrollo de un modelo de correlación genérico, que relaciona los eventos en la red, con lo que sucede a nivel de aplicación y el impacto real sobre el host, mediante un HIDS.

7.2. Discusión y conclusiones

En vista de los resultados obtenidos, el hecho más importante es que la correlación ha sido efectiva tal y como se vio en la sección 4 del capítulo 6 (con tasa 200:1). Se ha conseguido una reducción de las alertas con simples reglas de relación de eventos. La razón fundamental de esta reducción es la posibilidad de relación entre los eventos que ocurren en la red con lo que sucede a nivel de aplicación y host. Esta relación, por decirlo de alguna

manera, *causa-efecto*, (un ataque detectado en la red tiene como consecuencia un evento en el host) es primordial en los sistemas de correlación. Y si además, se le añade información de cualquier aplicación que se ha visto involucrada, como los logs de algún servicio, las posibilidades de reducción aún son mayores (junto con la fiabilidad).

Cabe destacar, los intentos de aproximación al modelo genérico de correlación explicado en la sección 3.3, que no han logrado emular el modelo en todas sus fases.
No se puede adaptar todas las fases del modelo expuesto, ya que OSSIM no permite realizar algunas de las fases por la propia naturaleza de la herramienta y sus limitaciones. Si bien, si es posible realizar la mayoría de la fases, siendo las fundamentales la normalización y preproceso de las alertas junto con la reconstrucción del ataque. Si se mira la seguridad de un sistema desde el punto de vista del ataque y compromiso, OSSIM permite sin duda, medir estos parámetros e incluso ir más lejos, permite la posibilidad de evaluar si alguna máquina después de comprometida está siendo usada para atacar otros sistemas.

Por otro lado, el manejo de la herramienta OSSIM y su integración con otras herramientas, ha sido en gran medida positiva y se han logrado los objetivos. Aunque, con bastante esfuerzo en cuanto a tiempo.
El problema fundamental con el que se ha encontrado el autor, es la poca documentación existente de la herramienta y la muy pobre información que se aportaba en los foros, teniendo que realizar la mayoría de las pruebas, a base de ir probando con distintas configuraciones.
La razón de esta escasa documentación es fundamentalmente el negocio que envuelve el manejo e instalación de OSSIM por parte de IT-Deusto. Los autores de OSSIM, han creado un producto que no es comercial en sí, pero si comercian con su instalación y configuración en distintos entornos, por lo que la herramienta de serie a penas viene configurada, teniendo que invertirse mucho tiempo en su aprendizaje y manejo.

Una de las cosas más interesantes con las que cuenta OSSIM, es la capacidad de integración de herramientas muy dispares entre si, y aunque las herramientas que se han integrado en este proyecto no eran muchas, si se puede decir que la integración ha sido exitosa, sin tener que hacer grandes adaptaciones.

7.3. Trabajo futuro

La herramienta OSSIM aún está por sus primeras versiones, ya que todavía no se ha publicado una primera versión 1.0 y aún se están desarrollando versiones 0.9, por lo que es mucho el trabajo que se puede hacer en relación a esta herramienta:

- Integración y pruebas de otras herramientas de seguridad. Por ejemplo puede integrarse con los logs de Iptables, PIX o cualquier dispositivo de Cisco Systems.

- Instalación y pruebas en un entorno más hostil, con sistemas heterogéneos, con más usuarios y con un número de máquinas considerable. Con esto se podría valorar las posibilidades de OSSIM en cuanto a rendimiento y funcionamiento con el fin de analizar las necesidades de CPU, memoria y disco que son necesarios para un correcto funcionamiento de OSSIM.

- Mejora de los scripts traductores (parsers) existentes. Aunque OSSIM integra un amplio conjunto de herramientas, los traductores de éstas, hechos en python, están incompletos. Así, es interesante completar los scripts para que permitan tratar la información de manera completa.

- Adaptación a nuevas herramientas. La gran variedad de herramientas de seguridad existentes en el mercado puede ser tenido en cuenta a la hora de trabajar con OSSIM. Es bastante sencillo ver como trabaja una herramienta y adaptar sus logs a OSSIM, mediante la realización del correspondiente script en python.

- Creación de nuevas directivas. Aunque las directivas creadas por el autor son bastante genericas, además de que OSSIM integra muchas directivas particulares para casos como la infección de virus, troyanos, etc, siempre es posible realizar nuevas directivas para ataques que tengan un preámbulo concreto.

- Estudio de la herramienta OSSIM como sensor de anomalías. Es interesante valorar las posibilidades de OSSIM, con las herramientas para anomalías que dispone.

- Integración de cifrado en las comunicaciones de manera nativa. OSSIM por defecto no se comunica mediante SSL ni ningún mecanismo de cifrado, aunque se podría realizar la comunicación por túneles SSH, sería interesante modificar el ossim-agent y ossim-server.

Por otro lado, en lo que se refiere a métodos de correlación se podrían realizar investigaciones en:

- Sistemas de correlación para aplicaciones web basadas en anomalías. Actualmente, uno de los campos donde más se está investigando, es el de las aplicaciones web o *webservices*, ya que son frecuentemente el objetivo de los atacantes por su extensa difusión. Así, trabajar con OSSIM y algún escudo a nivel de aplicación, como *mod security*[55] correlando los eventos con algún detector de anomalías puede dar resultados interesantes.
- Correlación de anomalías. Uno de los puntos que más falta por investigar es la correlación de anomalías, donde se está realizando muchos avances pero donde a día de hoy no hay resultados claros.

7.4. Resumen

En este capítulo se han expuesto las conclusiones del proyecto, así como los resultados obtenidos frente a los objetivos iniciales de este proyecto final de carrera.

También se ha explicado las dos posibles vertientes de trabajo futuro, una estudiando y mejorando las posibilidades de OSSIM y en segundo lugar investigando en otros campos de la correlación.

Capítulo 8

Planificación y presupuesto

8.1. Planificación temporal de las tareas

En este apartado realizaremos una planificación temporal de las tareas a realizar en el proyecto.

- Revisión y puesta al día.
  - Búsqueda bibliográfica.
  - Revisión bibliográfica.
- Desarrollo de la arquitectura.
  - Evaluar herramientas de correlación.
  - Diseño de la red.
  - Instalación de OSSIM.
- Implantación de la máquina víctima
  - Revisar y evaluar IDS.
  - Añadir IDS a la red.
- Puesta en marcha de la red.
- Completar memoria del proyecto.

En la tabla 8.1 se observa la duración de cada fase del proyecto y las dependencias entre éstas.

El diagrama de Gantt de la figura 8.1 resume la planificación de las tareas y

	TAREA	DURACIÓN	DEPENDENCIAS
1	Revisión y puesta al día	60 días	
1.1	Búsqueda bibliográfica	40 días	
1.2	Revisión bibliográfica	20 días	1.1
2	Desarrollo de la arquitectura.	30 días	
2.1	Evaluar herramientas correlación	10 días	1.2
2.2	Diseño de la red	5 días	2.1
2.3	Instalación OSSIM	15 días	2.2
3	Implantación de la máquina víctima	50 días	
3.1	Revisar y evaluar IDS	30 días	1.2
3.2	Añadir IDS en la red	20 días	2.1
4	Puesta en marcha	50 días	2.3, 3.2
5	Completar memoria	40 días	4

Cuadro 8.1: Duración y dependencia de las tareas del proyecto.

los hitos, junto con las fechas estimadas. El tiempo total del proyecto son : 60 días (revisión y puesta al día) + 30 días (desarrollo de la arquitectura) + 50 días (Implantación de la máquina víctima) + 50 días (puesta en marcha) + 40 días (completar memoria) = 230 días.

8.2. Presupuesto del proyecto

Los gastos del proyecto se dividen en dos partes: el coste del personal y el coste de los equipos necesarios.
A partir del coste temporal calculado en el apartado anterior y sabiendo que solo ha intervenido un ingeniero en la realización del mismo, se tiene los siguientes gastos:

- En las fase 1, 2, 3 y 5 del proyecto el trabajo del ingeniero es de 4h/día, con un total de 180 días * 4 horas = 720 horas.
- Durante la fase 4 sólo será necesario invertir 1 hora /día para comprobar que todo funciona correctamente, por lo que el tiempo total es de 50 horas.
- Con un sueldo de 40€/hora, el coste económico del personal es de (720 h + 50 h) * 40 €/hora = 30.800€.

El coste detallado por tareas se puede ver en la figura 8.2

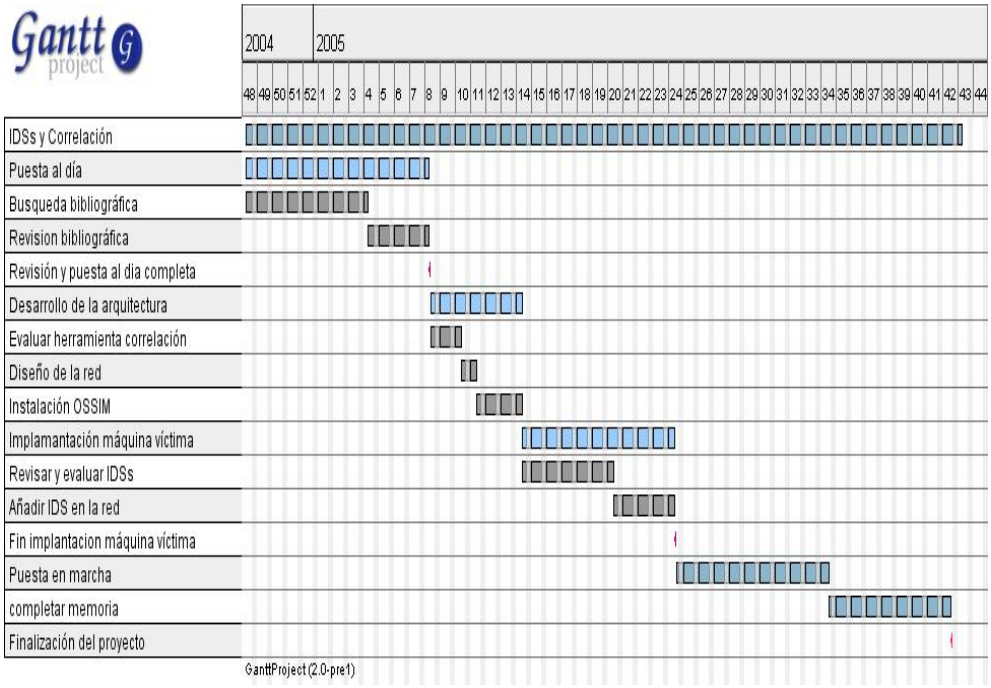

Figura 8.1: Diagrama de Gantt del proyecto

TAREA	COSTE
Revisión y puesta al día	9.600
Búsqueda bibliográfica	6.400
Revisión bibliográfica	3.200
Desarrollo de la arquitectura	4.800
Evaluar herramientas correlación	1.600
Diseño de la red	800
Instalación OSSIM	2.400
Implantación de la máquina víctima	8.000
Revisar y evaluar IDS	4.800
Añadir IDS en la red	3.200
Puesta en marcha	2.000
Completar memoria	6.400

Cuadro 8.2: Coste mano de obra del proyecto.

La parte hardware del proyecto, ya que no habrá que tener en cuenta el coste de licencias del software por ser gratuito, se desglosa en:

- Coste del servidor principal: 1000€.
- Coste de la máquina de pruebas: 1000€.
- Coste del conmutador cisco: 800€.

Como estos equipos tienen un periodo de amortización de 2 años, su coste amortizado se calcula [1]:
(2800 / 720 días) * 230 días = 894,44€

Por lo tanto, el coste económico del proyecto es de:
30.800€+ 894,44€= **31.694,44€**

8.3. Resumen

En este capítulo se ha analizado los costes del proyecto, tanto en recursos hardware como en recursos humanos.

[1] Se supone que dos años con 720 días

Apéndice A

Configuración switch Cisco modelo 2950

A.1. Configuración SPAN (Switch Port Analyzer)

```
interface FastEthernet0/2
 no ip address
 spanning-tree portfast
! interface FastEthernet0/3
 no ip address
 spanning-tree portfast
! interface FastEthernet0/4
 no ip address
 spanning-tree portfast
! interface FastEthernet0/5
 no ip address
 spanning-tree portfast
! interface FastEthernet0/6
 no ip address
 spanning-tree portfast
! interface FastEthernet0/7
 no ip address
 spanning-tree portfast
! interface FastEthernet0/8
 no ip address
```

```
 spanning-tree portfast
! interface FastEthernet0/9
 no ip address
 spanning-tree portfast
! interface FastEthernet0/10
 no ip address
 spanning-tree portfast
! interface FastEthernet0/11
 no ip address
 spanning-tree portfast
! interface FastEthernet0/12
 no ip address
 spanning-tree portfast
! interface FastEthernet0/13
 no ip address
 spanning-tree portfast
! interface FastEthernet0/14
 no ip address
 spanning-tree portfast
! interface FastEthernet0/15
 no ip address
 spanning-tree portfast
! interface FastEthernet0/16
 no ip address
 spanning-tree portfast
! interface FastEthernet0/17
 no ip address
 spanning-tree portfast
! interface FastEthernet0/18
 no ip address
 spanning-tree portfast
! interface FastEthernet0/19
 no ip address
 spanning-tree portfast
! interface FastEthernet0/20
 no ip address
 spanning-tree portfast
! interface FastEthernet0/21
 no ip address
 spanning-tree portfast
! interface FastEthernet0/22
```

```
 no ip address
 spanning-tree portfast
! interface FastEthernet0/23
 no ip address
 spanning-tree portfast
! interface FastEthernet0/24
 no ip address
 spanning-tree portfast

interface GigabitEthernet0/1
 no ip address
! interface GigabitEthernet0/2
 no ip address
! interface Vlan1
 ip address 10.0.0.1 255.255.255.0
 no ip route-cache
! no ip http server ! ! line con 0 line vty 0
 password 7 060303205F001A0E0C031103
 login
line vty 1 4
 login
line vty 5 15
 login
!

monitor session 1 source interface Fa0/2 - 23 monitor session 1
destination interface Gi0/1
```

Apéndice B

Configuraciones servidor de correlación

B.1. Iptables en máquina de correlación

```
#!/bin/bash
#borramos reglas del FW
iptables --flush


iptables -t filter -A INPUT -p icmp  -s 0/0  -d
    147.156.222.109/32  -i eth1 -j ACCEPT

iptables -t filter -A INPUT -p all  -s 147.156.222.65  -d
    147.156.222.109/32  -i eth1 -j ACCEPT
iptables -t filter -A INPUT -p all  -s 147.156.222.111  -d
    147.156.222.109/32  -i eth1 -j ACCEPT
iptables -t filter -A INPUT -p all  -s 147.156.222.201  -d
    147.156.222.109/32  -i eth1 -j ACCEPT


iptables -t filter -A INPUT -p all  -s 0/0
 -d 147.156.222.109/32 -mstate --state ESTABLISHED,RELATED
 -i eth1 -j ACCEPT
iptables -t filter -A INPUT -p tcp
 -s 0/0 -d 147.156.222.109/32
    -m state --state NEW --dport 22  -i eth1 -j ACCEPT
iptables -t filter -A INPUT -p tcp
```

```
-s 0/0 -d 147.156.222.109/32 -m
state --state NEW --dport 3000  -i eth1 -j ACCEPT
iptables -t filter
-A INPUT -p tcp  -s 0/0 -d 147.156.222.109/32
    -m state --state NEW --dport 80  -i eth1 -j ACCEPT

iptables  -t filter -A INPUT -p all  -s 0/0 -d 0/0  -i eth1
    -j REJECT --reject-with icmp-host-prohibited
```

B.2. Configuración OSSIM-agent

/etc/ossim/agent/config.xml

```
 <?xml version="1.0" encoding='UTF-8'
standalone="no" ?>

<!DOCTYPE config [

 <!ENTITY sensor "127.0.0.1" >
 <!ENTITY serverip "127.0.0.1" >

<!-- Log directory --> <!ENTITY logdir "/var/log/ossim" >

<!-- plugins --> <!ENTITY apache SYSTEM
'/etc/ossim/agent/plugins/apache.xml'>
'/etc/ossim/agent/plugins/arpwatch.xml'>
'/etc/ossim/agent/plugins/ca.xml'>
'/etc/ossim/agent/plugins/camonitor.xml'>
'/etc/ossim/agent/plugins/ciscoids.xml'>
'/etc/ossim/agent/plugins/ciscopix.xml'>
'/etc/ossim/agent/plugins/ciscorouter.xml'>
'/etc/ossim/agent/plugins/fw1.xml'>
'/etc/ossim/agent/plugins/iis.xml'>
'/etc/ossim/agent/plugins/iptables.xml'>
'/etc/ossim/agent/plugins/netgear.xml'>
'/etc/ossim/agent/plugins/ntop.xml'>
'/etc/ossim/agent/plugins/ntsyslog.xml'>
'/etc/ossim/agent/plugins/opennms.xml'>
'/etc/ossim/agent/plugins/osiris.xml'>
'/etc/ossim/agent/plugins/p0f.xml'>
```

```
'/etc/ossim/agent/plugins/pads.xml'>
'/etc/ossim/agent/plugins/prelude.xml'>
'/etc/ossim/agent/plugins/realsecure.xml'>
'/etc/ossim/agent/plugins/rrd.xml'>
'/etc/ossim/agent/plugins/snarewindows.xml'>
'/etc/ossim/agent/plugins/snort.xml'>
'/etc/ossim/agent/plugins/syslog.xml'> <!
'/etc/ossim/agent/plugins/tcptrack.xml'> ]>

<config>


<serverip>&serverip;</serverip> <serverport>40001</serverport>

<watchdog enable="yes" interval="30"/> <logdir>&logdir;</logdir>

<!-- restart plugins every hour -->
<plugin-restart enable="yes"
interval="600"/>

<plugins> &ntop; &snort; &syslog; </plugins> </config>
```

B.3. Snort.xml

```
<?xml version="1.0" encoding='UTF-8' ?>

 <plugin id="1001" process="snort" type="detector"
 start="yes"
enable="yes">
    <startup>/etc/init.d/snort start</startup>
    <shutdown>/etc/init.d/snort stop</shutdown>
    <source>fast</source>
    <interface>&interface;</interface>
    <sensor>&sensor;</sensor>
    <location>/var/log/snort/alert</location>
</plugin>
```

B.4. Ntop.xml

```
<?xml version="1.0" encoding='UTF-8' ?>
<plugin id="2005"
process="ntop" type="monitor" start="yes" enable="yes">
    <startup>/etc/init.d/ntop restart</startup>
    <shutdown>/etc/init.d/ntop stop</shutdown>
    <source>socket</source>
    <location>&sensor;:3000</location>
    <interface>&interface;</interface>
    <sensor>&sensor;</sensor>
    <frequency>20</frequency>
</plugin>
```

B.5. Configuración Ossim-framework

```
################################################ #
 nessus_user=angel
 nessus_pass=angel
 nessus_host=localhost
 nessus_port=1241
 nessus_path=/usr/bin/nessus
 nessus_rpt_path=/usr/share/ossim/www/vulnmeter/

sensors (1 = YES, 0 = NO)
 nessus_distributed=0
```

B.6. Directivas correlación

```
<?xml version='1.0' encoding='UTF-8' ?>

<directive id="2" name="correlacion NIDS-APACHE-HIDS"
priority="5">
<rule type="detector" name="nids" reliability="1" occurrence="1"
from="ANY" to="ANY" port_from="ANY" port_to="ANY" plugin_id="1001"
plugin_sid="ANY" >
<rules>
<rule type="detector" name="LOGS APACHE" reliability="3"
occurrence="1" from="1:SRC_IP" to="1:DST_IP" port_from="ANY"
port_to="ANY" plugin_id="1501" plugin_sid="ANY">
```

```
<rules>
<rule type="detector" name="ALERTA OSIRIS" reliability="5"
occurrence="1" from="ANY" to="ANY" port_from="ANY"
port_to="ANY" plugin_id="4001" plugin_sid="ANY"/>
</rules>
</rule>
</rules>
</rule>
</directive>

<directive id="3" name="correlacion NIDS_HIDS"
priority="5"> <rule
type="detector" name="nids" reliability="1"occurrence="1"
from="ANY"
to="ANY" port_from="ANY"
port_to="ANY" plugin_id="1001" plugin_sid="ANY" >
<rules>
<rule type="detector" name="ALERTA OSIRIS" reliability="5"
occurrence="1" from="ANY" to="ANY" port_from="ANY"
port_to="ANY" plugin_id="4001" plugin_sid="ANY"/>
</rules>
</rule>
</directive>

<directive id="4" name="correlacion NIDSaPACHEhIDScONE -
VICTIMA" priority="5"> <rule type="detector" name="nids"
reliability="1" occurrence="1" from="ANY" to="ANY" port_from="ANY
port_to="ANY" plugin_id="1001" plugin_sid="ANY" >
<rules> <rule
type="detector" name="LOGS APACHE" reliability="3" occurrence="1"
from="1:SRC_IP" to="1:DST_IP" port_from="ANY"   port_to="ANY"
plugin_id="1501" plugin_sid="ANY">
<rules>
<rule type="detector" name="ALERTA OSIRIS" reliability="5"
occurrence="1" from="ANY" to="ANY" port_from="ANY"
port_to="ANY" plugin_id="4001" plugin_sid="ANY">
<rules>
<rule type="monitor" name="ALERTA CONEXION RARA"
reliability="10" from="1:DST_IP" to="ANY" port_from="ANY"
port_to="ANY" plugin_id="2005" plugin_sid="ANY"/>
</rules>
```

```
</rule>
</rules>
     </rule>
  </rules>
 </rule>
</directive>

<directive id="5"
name="correlacion NIDS-APACHE-HIDS CONEXION
VICTIMA-ATACANTE" priority="5"> <rule type="detector" name="nids"
reliability="1" occurrence="1" from="ANY" to="ANY" port_from="ANY"
port_to="ANY" plugin_id="1001" plugin_sid="ANY" >
<rules>
<rule type="detector" name="LOGS APACHE" reliability="3"
occurrence="1" from="1:SRC_IP" to="1:DST_IP" port_from="ANY"
port_to="ANY" plugin_id="1501" plugin_sid="ANY">
<rules>
<rule type="detector" name="ALERTA OSIRIS" reliability="5"
occurrence="1" from="ANY" to="ANY" port_from="ANY"
port_to="ANY" plugin_id="4001" plugin_sid="ANY">
<rules>
<rule type="monitor" name="ALERTA CONEXION RARA" reliability="10"
from="1:DST_IP" to="1:SRC_IP" port_from="ANY"
port_to="ANY" plugin_id="2005" plugin_sid="ANY"/>
</rules>
</rule>
</rules>
</rule>
</rules>
</rule>
</directive>

<directive id="7" name="brute force login attempt against
DST_IP" priority="5">
<rule type="detector" name="Authentication failure" reliability="3"
ocurrence="1" from="ANY" to="ANY" port_from="ANY" port_to="ANY"
time_out="10" plugin_id="4002" plugin_sid="1,2,3">
<rules>
<rule type="detector" name="Authentication failure (3 times)"
reliability="+1" occurrence="3" from="1:SRC_IP" to="ANY"
port_from="ANY" time_out="15" port_to="ANY"
```

```
plugin_id="4002" plugin_sid="1,2,3" sticky="true">
<rules>
<rule type="detector" name="Authentication failure (5 times)"
reliability="+2" occurrence="5" from="1:SRC_IP" to="ANY"
port_from="ANY" time_out="20" port_to="ANY"
plugin_id="4002" plugin_sid="1,2,3" sticky="true">
<rules>
<rule type="detector" name="Authentication failure (10 times)"
reliability="+2" occurrence="10" from="1:SRC_IP" to="ANY"
port_from="ANY" time_out="30" port_to="ANY"
plugin_id="4002" plugin_sid="1,2,3" sticky="true">
</rule>
</rules>
</rule>
</rules>
</rule>
</rules>
</rule>
</directive>
```

Apéndice C

Configuraciones máquina de víctima 'labd.uv.es'

C.1. Configuracion iptables

```
iptables --flush

iptables -t filter -A INPUT -p icmp  -s 0/0
 -d 147.156.222.201/3 -i eth0 -j ACCEPT

iptables -t filter -A INPUT -p all  -s 147.156.222.109  -d
    147.156.222.201/32  -i eth0 -j ACCEPT

iptables -t filter -A INPUT -p all  -s 0/0  -d
    147.156.222.201/32  -m state --state ESTABLISHED,
    RELATED -i eth0 -j ACCEPT
iptables -t filter -A INPUT -p tcp  -s 0/0 -d
    147.156.222.201/32 -m state --state NEW --dport
    22  -i eth0 -j ACCEPT
iptables -t filter -A INPUT -p tcp  -s 0/0 -d
    147.156.222.201/32 -m state --state NEW --dport
     80  -i eth0 -j ACCEPT


iptables  -t filter -A INPUT -p all  -s 0/0 -d 0/0
-i eth0 -j REJECT
--reject-with icmp-host-prohibited
```

C.2. Configuración Apache.xml

```
<?xml version="1.0" encoding='UTF-8' ?> <!--
     Apache detector
     location must point to an apache access log
     (for example: /var/log/apache/access.log)
--> <plugin id="1501" process="apache" type="detector" start="yes"
enable="yes">
    <startup>/etc/init.d/apache restart</startup>
    <shutdown>/etc/init.d/apache stop</shutdown>
    <source>common</source>
    <interface>&interface;</interface>
    <sensor>&sensor;</sensor>
    <location>/var/log/apache/access.log</location>
</plugin>
```

C.3. Configuración Syslog.xml

```
<?xml version="1.0" encoding='UTF-8' ?>

<plugin id="4002" process="syslogd" type="detector" start="yes"
enable="yes">


<startup>/etc/init.d/syslog start</startup>
<shutdown>/etc/init.d/syslog stop</shutdown>

<source>common</source>
 <interface>&interface;</interface>

<sensor>&sensor;</sensor>
<location>/var/log/auth.log</location>
</plugin>
```

C.4. Configuración Orisis.xml

```
<?xml version="1.0" encoding='UTF-8' ?>

<plugin id="4001" process="osirisd" type="detector" start="yes"
enable="yes">
```

```
    <startup>/etc/init.d/osirisd restart</startup>
    <shutdown>/etc/init.d/osirisd stop</shutdown>
    <source>syslog</source>
    <interface>&interface;</interface>
    <sensor>&sensor;</sensor>
    <location>/var/log/syslog</location>
    <verbose>yes</verbose>
</plugin>
```

C.5. Configuración de Osiris

El fichero que contiene la información de qué monitorizar en el host, contiene la siguiente información:

```
Recursive   no FollowLinks no

IncludeAll Hash sha

<System>
    Include mod_users
    Include mod_groups
    Include mod_kmods
</System>

<Directory /bin>
    IncludeAll
</Directory>

<Directory /usr/bin>
    IncludeAll
</Directory>

<Directory /usr/local/bin>
    IncludeAll
</Directory>

<Directory /usr/local/sbin>
    IncludeAll
</Directory>

<Directory /sbin>
```

```
    IncludeAll
</Directory>

<Directory /usr/sbin>
    IncludeAll
</Directory>

<Directory /boot>
    IncludeAll
</Directory>

<Directory /var/www>
    IncludeAll
</Directory>
```

Índice alfabético

Bibliografía

[1] ELAS: *Elementos activos de seguridad*. [En línea a 7 de noviembre de 2005].
<http://elas.uv.es>

[2] CSIC: *Consejo Superior de Investigaciones Científicas*. [En línea a 7 de noviembre de 2005].
<http://www.csic.es>

[3] Rediris: *Red Española de I+D*. [En línea a 7 de noviembre de 2005].
<http://www.rediris.es>

[4] : *Implantación de un IDS en la Universitat de Valencia.*. [En línea a 7 de noviembre de 2005].
<http://elas.uv.es/documents/emial.pdf>

[5] : *Detección de ataques y análisis forense en sistemas honeypot en sistema Linux RH-6.2*. [En línea a 7 de noviembre de 2005].
<http://elas.uv.es/documents/jorotren.pdf>

[6] : *Detección de ataques y análisis forense en sistemas honeypot en sistema Windows 2000* . [En línea a 7 de noviembre de 2005].
<http://elas.uv.es/es/resumen-valapont.htm>

[7] *Sistemas de detección de intrusos distribuidos*. Antonio Villalón. Conferencia UCLM, Abril 2004. [En línea a 7 de noviembre de 2005].
<http://andercheran.aiind.upv.es/toni/personal/>

[8] *SNORT*. [En línea a 7 de noviembre de 2005].
<http://www.snort.org/>

[9] *Bleeding Snort Rules*. [En línea a 7 de noviembre de 2005].
<http://www.bleedingsnort.com/>

[10] *Enterasys Intrusion Defense*. [En línea a 7 de noviembre de 2005].
<http://www.enterasys.com/products/ids/>

[11] *Cisco Intrusion Detection System*. [En línea a 7 de noviembre de 2005]. <http://www.cisco.com/warp/public/cc/pd/sqsw/sqidsz/>

[12] *NFR Network Intrusion Detection*. [En línea a 7 de noviembre de 2005]. <http://www.nfr.com/>

[13] *CFI LANGuard Security Event Log Monitor*. [En línea a 7 de noviembre de 2005]. <http://www.surfinsecurity.com/GFI/GFI.htm>

[14] *BlackIce*. [En línea a 7 de noviembre de 2005]. <http://blackice.iss.net/product_pc_protection.php>

[15] *OSIRIS*. [En línea a 7 de noviembre de 2005].<http://www.hostintegrity.com/osiris/>

[16] *Systrace*. [En línea a 7 de noviembre de 2005]. <http://www.citi.umich.edu/u/provos/systrace/>

[17] *Tripwire*. [En línea a 7 de noviembre de 2005]. <http://www.tripwire.com/>

[18] *Bro*. [En línea a 7 de noviembre de 2005]. <http://www.bro-ids.org/>

[19] *Prelude*. [En línea a 7 de noviembre de 2005]. <http://www.prelude-ids.org/>

[20] *Prelude: an Open Source, Hybrid Intrusion Detection System*. [En línea a 7 de noviembre de 2005]. <http://prelude-ids.org/article.php3?id_article=66>

[21] *IDMEF*. [En línea a 7 de noviembre de 2005]. <http://xml.coverpages.org/idmef.html>

[22] *IDMEF en IETF*. [En línea a 7 de noviembre de 2005]. <http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-14.txt>

[23] *Nmap*. [En línea a 7 de noviembre de 2005]. <http://www.insecure.org/nmap/>

[24] *Libwhisker Securityfocus*. [En línea a 7 de noviembre de 2005]. <http://www.securityfocus.com/infocus/1798>

[25] *Libwhisker*. [En línea a 7 de noviembre de 2005]. <http://www.wiretrip.net/rfp/lw.asp>

[26] *Nikto*. [En línea a 7 de noviembre de 2005]. <http://www.cirt.net/code/nikto.shtml>

[27] *Nessus*. [En línea a 7 de noviembre de 2005]. <http://www.nessus.org/>

[28] *SANS*. [En línea a 7 de noviembre de 2005]. <http://www.sans.org/>

[29] *Ministerio de Ciencia y Tecnología*. [En línea a 7 de noviembre de 2005]. <http://www.myct.es>

[30] *Best Known Methods in Security Events Correlation*. [En línea a 7 de noviembre de 2005]. <http://www.mycert.org.my/mycert-sig/mycert-sig-04/slides/security_events(fazil).ppt>

[31] *Event Correlation*. [En línea a 7 de noviembre de 2005]. <http://www.computerworld.com/networkingtopics/networking/management/story/0,10801,83396,00.html>

[32] *PIX*. [En línea a 7 de noviembre de 2005]. <http://www.cisco.com/warp/public/cc/pd/fw/sqfw500/>

[33] *phpgacl*. [En línea a 7 de noviembre de 2005]. <http://phpgacl.sourceforge.net/>

[34] *Acid*. [En línea a 7 de noviembre de 2005]. <http://www.andrew.cmu.edu/user/rdanyliw/snort/snortacid.html>

[35] *Ntop*. [En línea a 7 de noviembre de 2005]. <http://www.ntop.org/ntop.html>

[36] *Libpcap*. [En línea a 7 de noviembre de 2005]. <http://sourceforge.net/projects/libpcap/>

[37] *Arpwatch*. [En línea a 7 de noviembre de 2005]. <http://www.securityfocus.com/tools/142>

[38] *Seguridad en Redes conmutadas*. Angel Alonso Párrizas. Presentación asignatura de redes curso 2003 [En línea a 7 de noviembre de 2005]. <http://www.uv.es/montanan/redes/trabajos/hacking_sw_LAN.pdf>

[39] *P0f*. [En línea a 7 de noviembre de 2005].
<http://www.stearns.org/p0f/>

[40] *Nikto*. [En línea a 7 de noviembre de 2005].
<http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/>

[41] *Nikto*. [En línea a 7 de noviembre de 2005].
<http://people.ee.ethz.ch/~oetiker/webtools/mrtg/>

[42] *OSSIM*. [En línea a 7 de noviembre de 2005].
<http://www.ossim.net>

[43] *A Comprenhensive Approach to Intrusion Detection Alert Correlation*. Giovanni Vigna. 1545-5971/04. IEEE July 2004[En línea a 7 de noviembre de 2005].
<http://www.cs.ucsb.edu/~vigna/pub/2004_valeur_vigna_kruegel_kemmerer_TDSC_Correlation.pdf>

[44] *Vigna*. [En línea a 7 de noviembre de 2005].
<http://www.cs.ucsb.edu/~vigna/>

[45] *Python*. [En línea a 7 de noviembre de 2005].
<http://www.python.org/>

[46] *Apache web server*. [En línea a 7 de noviembre de 2005].
<http://www.apache.org/>

[47] *Cisco*. [En línea a 7 de noviembre de 2005].
<http://www.cisco.com/>

[48] *Firewall One*. [En línea a 7 de noviembre de 2005].
<http://www.checkpoint.com/products/firewall-1/>

[49] *OpenSSL*. [En línea a 7 de noviembre de 2005].
<http://www.openssl.org/>

[50] *PHP-Nuke*. [En línea a 7 de noviembre de 2005].
<http://phpnuke.org/>

[51] *Iptables / Netfilter*. [En línea a 7 de noviembre de 2005].
<http://www.netfilter.org/>

[52] *IIS*. [En línea a7 de noviembre de 2005].
<http://go.microsoft.com/fwlink/?LinkId=7001>

[53] *Ntsyslog*. [En línea a 7 de noviembre de 2005]. <http://ntsyslog.sourceforge.net/>

[54] *Osiris*. [En línea a 7 de noviembre de 2005]. <http://www.hostintegrity.com/osiris/>

[55] *ModSecurity*. [En línea a 7 de noviembre de 2005]. <http://www.modsecurity.org/>

[56] *False Positives: A User's Guide to Making Sense of IDS Alarms* , M. J. Ranum, ICSA Labs IDSC, 2003.

[57] *Strategies to Reduce False Positives and False Negatives in NIDS*, K. Timms, SecurityFocus Article, 2001 http://www.securityfocus.com/infocus/1463

[58] *Tools and Techniques for Analyzing Intrusion Alerts*, P. Ning, Y. Cui, and D. Reeves, in ACM Transactions on Information and System Security, Vol. 7, No. 2, pages 273–318, May 2004.

[59] *EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbance*,P. A. Porras and P.G. Neumann, 20th NISSC, October 9, 1997.

[60] *Seguridad en Unix y Redes. Versión 2.1* . Antonio Villalón, Edicion 2001. [En línea a 7 de noviembre de 2005]. <http://andercheran.aiind.upv.es/toni/personal/>