Universidad de las Ciencias Informáticas

Facultad 2

Sistema para el registro y control centralizado de eventos y sucesos de la red de un centro de datos.

Trabajo de diploma por el título de Ingeniero en Ciencias Informáticas.

Autor:

Yordan Velez Rodríguez

Tutor:

Ing. Tte. Norge Fajardo Vega.

Ciudad de La Habana, julio 2008

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo a la Unidad para la Compatibilización, Integración y Desarrollo (UCID) y a la Facultad 2 de la universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los _____ días del mes _____ del año 2008.

_____	_____
Yordan Velez Rodríguez	Ing. Tte. Norge Fajardo Vega
Autor	Tutor

Datos de contacto

Ing. Tte. Norge Fajardo Vega.

Graduado en el 2007 de Ingeniero en Ciencias Informáticas de la Universidad de las Ciencias Informáticas (UCI). Especialista general del bloque 8 de la Infraestructura Productiva. Actualmente desempeña las funciones de administrador de la red de dicho bloque. Impartió la asignatura de práctica profesional en el primer semestre como profesor de la facultad 4.

e-mail: norge@uci.cu
teléfono: 07 837 3611
.

Agradecimientos

A la ***Revolución*** por hacer obras tan bellas e importantes como esta universidad y darle la posibilidad a todo el pueblo de acceder a ella.

A las ***Fuerzas Armadas Revolucionarias*** y a la ***Universidad de las Ciencias Informáticas*** por la formación y la educación profesional que me ofrecieron.

A mis ***padres***, por la incansable labor de velar por el correcto desarrollo de esta tesis y mis estudios.

A mi ***hermano***, por ser la persona que más admiro.

A mis ***uci-hermanos*** Darvis y Alain por el apoyo y la amistad que quedó para toda la vida.

A todos los ***amigos*** que encontré en Venezuela, por el apoyo incondicional que me brindaron.

A todos los ***amigos*** que de una forma u otra me han ayudado.

A todas esas personas que con sus consejos oportunos hicieron de mí una mejor persona.

Gracias a todos.

Dedicatoria

En especial a mi abuelo Enrique que siempre ha deseado vivir lo suficiente como para ver a todos sus nietos graduados.

A mis padres y hermano por todo su amor.

A los hijos que todavía no tengo, por ser la fuerza de mi empeño.

Resumen

Las Fuerzas Armadas Revolucionarias (FAR), apostando por un *software libre* que se adecua a los principios socialistas, no se ha quedado atrás en el desarrollo e implementación de las ciencias informáticas en pro de la defensa del país. El aumento de los diferentes servicios a brindar por sus centros de datos, así como el crecimiento notable de los usuarios que hacen empleo de éstos, hace evidente la necesidad de gestionar, con una mayor eficacia, calidad y seguridad, todos los eventos y sucesos relacionados con la red, computadoras y dispositivos implicados en el adecuado funcionamiento de dichos servicios. La diversidad de herramientas que se emplean para tales efectos acarrea un mal mayor: insuficiencia de tiempo para garantizar la calidad de la labor que se realiza. En aras de cambiar esta situación; la presente tesis que responde a la temática de Seguridad Informática, propone la implantación de un sistema que de manera centralizada permite una eficiente gestión y control de los eventos y sucesos en las redes de datos.

Índice

Índice

Índice de figuras.

Índice de tablas.

Introducción

En la actualidad, los centros de datos se han convertido en uno de los principales objetivos de ataques informáticos debido al gran cúmulo e importancia de la información que procesan.

Sin dudas, un centro de datos oferta disímiles de servicios que deben cumplir con los principios básicos de la seguridad informática: **disponibilidad**, **integridad** y **confidencialidad**.

Para prevenir y resolver cualquier intento de alteración de los sistemas informáticos bajo su responsabilidad, los administradores de red implementan varias técnicas que pueden ser sistemas o *software* para tales objetivos, además de la revisión periódica de los log (registro de hechos y eventos en el transcurso del tiempo) que generan cada uno de los sistemas empleados para brindar los servicios. Esta última un tanto compleja, a pesar de la existencia de "analizadores de log", debido a la diversidad del formato de salida de los *log* generados por las aplicaciones de cada centro de datos en particular.

Varios administradores de red no reconocen la importancia que revista la administración y control de los log y por eso descuidan esta tarea que es muy útil para prevenir ataques, mal funcionamiento de aplicaciones, así como tener todo un registro de los eventos y sucesos ocurridos en la red.

Para una buena administración de la red, es necesario conocer todo lo que ocurre en la red, desde el funcionamiento de los servicios que se ofertan hasta el tráfico de datos. Esto último es también muy importante porque avizora sobre cualquier exceso de carga y/o peticiones a determinado servicio o dispositivo. Existen aplicaciones que permiten registrar estos acontecimientos así como hacer todo un análisis de éstos.

Un administrador de red consta de varias aplicaciones para realizar su labor diaria, pero éstas, generalmente, residen en diferentes ordenadores o dispositivos, son muy diversas y no tienen prácticamente ninguna relación entre ellas. Todo esto conlleva a que el trabajo del administrador de la red sea harto complejo.

Cabe preguntar, ¿cómo lograr una eficiente gestión y control centralizado de los eventos y sucesos de la red en un centro de datos?

Este trabajo tiene como **objetivo general** proponer e implantar un sistema que permita la gestión y control centralizado de los eventos y sucesos en un centro de datos de manera que facilite y agilice el trabajo de los administradores de red. Esto implica que el **objeto de estudio** sean los procesos de administración de red y el **campo de acción** es el proceso de gestión de la administración y control de eventos y sucesos de la red en un centro de datos militar.

A continuación las tareas de investigación:

- Realizar la fundamentación teórica que sustenta el estudio de la administración y control de eventos y sucesos de una red de datos.
- Analizar las herramientas existentes para el análisis, control y reporte de ficheros *log*.
- Analizar las herramientas para la detección de intrusos.
- Proponer e implantar una solución integral para la gestión del control centralizado del registro de eventos y sucesos de la red de datos

PALABRAS CLAVES
Seguridad informática
Administración
Control
Log
Análisis
Reporte
Red
Detección de intrusos
Administración de redes

Capítulo 1. Fundamentación teórica.

1.1. Introducción.

Con el decursar de los días, va aumentando el número de los servicios a brindar por los centros de datos, lo que hace más complejo la gestión de los registros de eventos y sucesos generados por las aplicaciones y dispositivos empleados para garantizar la disponibilidad de estas prestaciones. El monitoreo y control de redes y sistemas en un centro de datos tiene muchas ventajas pero la diversidad de los formatos de los log, y de las herramientas existentes para el análisis del flujo de datos a través de la red hacen complejo dicho monitoreo y control.

Este capítulo aborda lo referente al estado del arte relacionado con la gestión y control de los eventos y sucesos de la red en un centro de datos; así como los diferentes conceptos y características relacionadas con las tendencias, metodologías y tecnologías actuales, con el objetivo de encontrar las más adecuadas para la realización de esta tesis.

1.2. Centro de datos.

Según el diccionario en línea, *Dictionary.com* (1), un centro de datos (*DC*, por sus siglas en inglés) es una lugar con equipamiento necesario y/o una o más computadoras conectadas, que se emplean para el procesamiento y/o trasmisión de datos.

Para procesar y trasmitir datos, además de los equipos, se necesita aplicaciones, sistemas informáticos que vinculados a dichos equipos hacen que esto sea posible.

Los centros de datos, básicamente, siempre prestan servicios como portales web, proxy para el acceso a internet, sistemas de gestión, servidores de documentos, bases de datos; además centralizan y protegen toda la información referente al lugar, empresa o institución a la que pertenecen.

Según el volumen y tipo de información que procesen, así será el tipo de tecnología, personal y seguridad que requieran. En la actualidad, las variantes de centro de datos existentes, y los servicios que ofertan varían de acuerdo a las necesidades de las empresas.

Es necesario esclarecer que no siempre cuando se refiere al término “centro de datos”, significa un único lugar geofísico, sino que de acuerdo a las complejidades y necesidades pueden estar distribuidos en más de un “sub-centro”.

1.2.1 Centro de datos militares.

Un centro de datos militar no dista mucho de los centro de datos comunes, pero debido a la información que manipulan, requiere mayores niveles de seguridad, lo que implica que el proceso de administración sea más complejo y sistemático.

1.3 Administración de redes.

Actualmente, el número de personas que pueden acceder a las redes, ha crecido mucho; y pueden hacerlo con diferentes propósitos e intereses. Estas redes tan complejas ha creado la necesidad de tener una gran seguridad en los sistemas que dan acceso a la información y sobre el contenido que se trasmite a través de las redes de comunicación.

Esto hace que el área de administración y seguridad en redes cobre una importancia que anteriormente no tenía y justifica ampliamente que se desarrollen metodologías para la implementación de sistemas de administración y seguridad en redes, e inclusive que se desarrolle investigación en estas áreas.

Gestionar y controlar una o un grupo de redes, en pocas palabras, es realizar una serie de actividades con cierta sistematización mediante las cuales se garanticen el correcto funcionamiento de los equipos y aplicaciones que se están empleando para brindar los servicios. Entre dichas tareas, se encuentran el mantenimiento y soporte a los servicios, actualizaciones de seguridad, pruebas de vulnerabilidades antes ataques informáticos, y el chequeo y monitoreo de los log de seguridad, entre otros.

1.4 Seguridad informática.

Este concepto varía de uno a otro experto en la materia. No obstante todo se basa en tres conceptos elementales: (2)

Confidencialidad: la información o los activos informáticos son accedidos solo por las personas autorizadas.

Disponibilidad: los activos informáticos son accedidos por las personas autorizadas en el momento requerido.

Integridad: los activos informáticos o la información solo pueden ser modificadas por las personas autorizadas y de la forma autorizada.

Existen algunos principios y estrategias a seguir para mantener adecuadamente la seguridad informática de una institución. A continuación se mencionan estos principios. (2)

Mínimo privilegio: se deben otorgar los permisos estrictamente necesarios para efectuar las acciones que se requieran, ni más ni menos de lo solicitado.

Eslabón más débil: la seguridad de un sistema es tan fuerte como su parte más débil. Un atacante primero analiza cual es el punto más débil del sistema y concentra sus esfuerzos en ese lugar.
Proporcionalidad: las medidas de seguridad deben estar en correspondencia con lo que se protege y con el nivel de riesgo existente. No sería lógico proteger con múltiples recursos un activo informático que no posee valor o que la probabilidad de ocurrencia de un ataque sobre el mismo es muy baja.
Dinamismo: la seguridad informática no es un producto, es un proceso. No se termina con la implementación de los medios tecnológicos, se requiere permanentemente monitoreo y mantenimiento.
Participación universal: la gestión de la seguridad informática necesita de la participación de todo el personal de una institución. La seguridad que puede ser alcanzada mediante medios técnicos es limitada y debiera ser apoyada por una gestión y procedimientos adecuados, que involucren a todos los individuos.

1.5 Cortafuegos (*firewall*).

La forma más efectiva de aislar una computadora es hacerlo físicamente, es decir, desconectarla de cuanto dispositivo pueda introducir información; aún así no estaría del todo segura. Sucede que la necesidad demuestra que los usuarios necesitan compartir archivos, ficheros, información entre ellos, de forma tal que solo los autorizados puedan acceder a las cosas compartidas.
“Un cortafuego, es un sistema o grupo de sistemas que hace cumplir una política de control de acceso entre dos redes” (3). Esencialmente, es crear un punto de control del tráfico de la red. Esto contribuye, notablemente, a ralentizar cualquier ataque dirigido a la red, sin embargo, por lo general provoca una creencia de que ya la red está protegida, y eso conlleva a descuidar otras actividades.

1.6 Servidor web.

Es un programa que implementa el protocolo *HTTP* (*hypertext transfer protocol*). Este protocolo está diseñado para transferir lo que se llama hipertextos, páginas web o páginas *HTML*: textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.
Un servidor web se encarga de mantenerse a la espera de peticiones *HTTP* llevadas a cabo por un cliente *HTTP*, normalmente, un navegador. El navegador realiza una petición al servidor y éste le responde con el contenido que el cliente solicita.

1.7 *Proxy*.

"Es un programa, que permite o niega el acceso a una aplicación determinada entre dos redes. Los clientes se comunican sólo con los servidores *proxy*, que autorizan las peticiones y las envían a los servidores reales, o las deniegan y las devuelven a quien las solicitó." (3)

1.8 *Log.*

Log: Registro oficial de eventos durante un período de tiempo en particular (4). Para los profesionales en seguridad informática un log es usado para registrar datos o información sobre quién, qué, cuándo, dónde y por qué (*who, what, when, where* y *why*, W5) un evento ocurre para un dispositivo en particular o aplicación (5).

La mayoría de los log se almacenan en texto plano, es decir, entendibles por cualquier persona, lo que permite que cualquier otro dispositivo o aplicación pueda hacer uso de éstos.

1.8.1 Propósito de los *log*.

Si un log tiene la capacidad de almacenar los eventos que ocurren en cierto dispositivo o aplicación, entonces el objetivo de un log es proveer a los profesionales de la seguridad informática la habilidad de monitorear las actividades realizadas por quien haya generado el fichero. Revisando los *log*, se puede prevenir ataques informáticos y tomar las decisiones necesarias para corregir problemas de funcionamiento.

1.8.2 Consolidación de *log*.

La consolidación de *log* permite una vía rápida y muy útil de organizar, asegurar y controlar dichos archivos.

Se debe considerar dos escenarios a la hora de consolidar ficheros de *log*, ambos sobre un área centralizada de almacenaje.

ESCENARIO # 1

En este caso se propone un único servidor central para la administración de *log*. La información está disponible *24x7* (veinticuatro horas al día los siete días de la semana) y utilizada para cualquier propósito de los administradores de red.

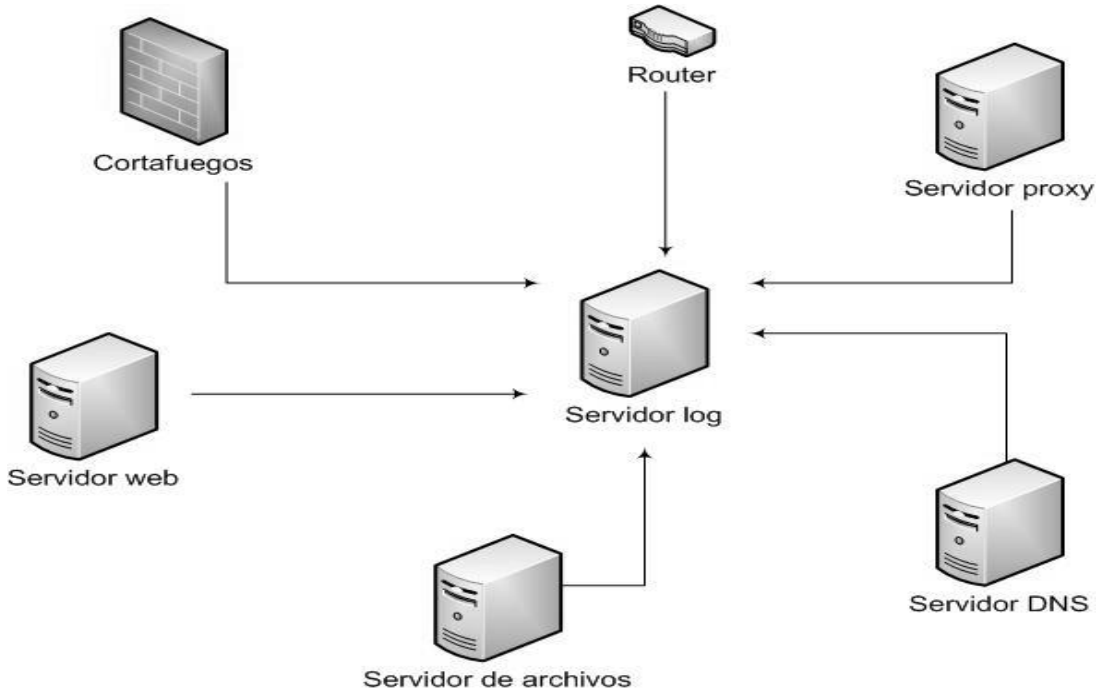

Figura # 1. Escenario 1: Un único servidor central.

Esta configuración requiere de buena capacidad de almacenamiento. El principal riesgo que presenta es que al ser un único servidor de log, se convierte en un único punto de ataque. El sistema completo dependería de la disponibilidad de este punto e igualmente para un atacante la tarea de "limpiar" su actividad sería mucho más sencilla. Una nueva variante del *ESCENARIO 1* radica en centralizar los log en un solo servidor, pero en vez de hacerlo con los ficheros, los datos de los archivos son insertados en un Sistema Gestor de Bases de Datos (SGBD).

ESCENARIO# 2

Esta arquitectura plantea un almacenaje de log de acuerdo a su clasificación. Cada servidor es utilizado para cada clase de fuente de log. Los log están centralizados en un ambiente de control *24x7*. En este caso si un atacante desea borrar los registros de las actividades que hizo, tendrá mucho más trabajo. El problema esencial de esta configuración es el costo que implica montar la infraestructura que la soporte.

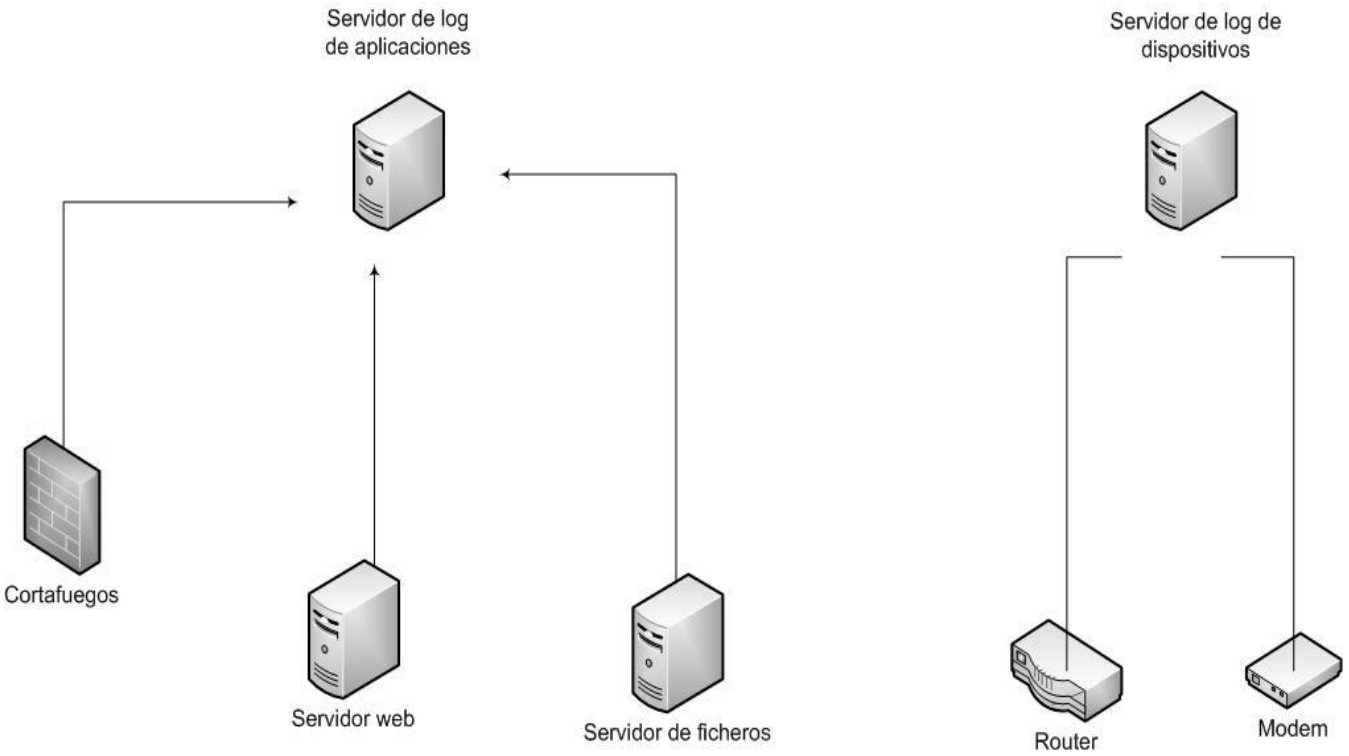

Figura # 2. Escenario 2: Varios servidores para la centralización de *logs*.

1.8.3 Rotación de *log*

Los archivos de log pueden consumir gran cantidad de capacidad de almacenamiento, una técnica para limitar este volumen es la rotación de log, la cual consiste en moverlos, cada cierto período de tiempo, para una nueva ubicación y renombrarlos de forma que se refleje la rotación. Debido a la dependencia del tiempo, es necesario que los dispositivos y servidores estén sincronizados.

Existen aplicaciones que realizan esta función de forma automática, solo hay que configurarles las opciones de tiempo y los lugares a los cuales se moverán los archivos.

1.9 Analizadores de *log.*

Existen diferentes aplicaciones en las distintas plataformas que analizan los ficheros de log generados por los sistemas y dispositivos. Unas son muy básicas, limitándose a leer el archivo y a reportar cierta información elemental. Sin embargo existen otros muy potentes, pues además de ser capaces de interpretar los log de diferentes aplicaciones de más de un servicio, generan informes completos del análisis que realizan.

Varios de estos analizadores se enfocan, o bien a todas las aplicaciones de un servicio determinado o simplemente a una aplicación de cada servicio.

A continuación se muestra una breve descripción de los analizadores de archivos de log más utilizados.

Webalizer (6)

Es un programa rápido y libre para el análisis de ficheros de log de servidores web. Su salida es altamente detallada y fácilmente configurable para crear los reportes en formato *HTML*, que permite ser visto por cualquier navegador web.

Está escrito en *C*, extremadamente rápido y altamente portable, sin necesidad de máquinas con características muy avanzadas de cálculo. Soporta el formato estándar de ficheros *log* generados por los servidores. También soporta, nativamente, los formatos de archivos *log* que producen las aplicaciones *wu-ftpd, xferlog FTP* y *squid*. Los reportes generados pueden ser configurados desde una consola o desde los ficheros de configuración. Soporta múltiples lenguajes. Trabaja con ficheros log de diferentes tamaño, permitiendo rotarlos según las necesidades y eliminando aquellos que llevan mucho tiempo en el sistema. Distribuido bajo la licencia *GNU General Public License*.

AWFFull (7)

Herramienta para el análisis de ficheros log de servidores web. Principalmente usado para obtener reportes simples. Genera simples páginas *HTML* que pueden ser visualizadas por cualquier navegador. Permite, en una mirada, identificar las áreas con problemas. Está basado en *Webalizer* (6) y necesita adicionalmente librerías *GD Graphics, PCRE, libpng* y *zlib.*

Calamaris (8)

Capaz de interactuar con una gran variedad de ficheros log generados por varios servidores proxy web y a partir de ahí produce reportes basados en métodos de peticiones, reportes de estado de peticiones de entrada/salida, tipos de contenido y desempeño. Está escrito en *Perl*, independiente de sistema operativo, distribuido bajo la licencia *GNU General Public License*.

Logwatch (9)

Es un sistema configurable para el análisis de ficheros *log*. Dado un período de tiempo revisa los log del sistema y genera reportes de acuerdo a las áreas especificada por el usuario y tan detalladas como éste las desee. Es fácil de usar y puede trabajar con múltiples máquinas. Está distribuido bajo la licencia *GNU General Public License.*

Iptqlog (10)

Significa *IPTables Queue Logger*. Presenta de manera clara y distintivamente (mediante colores, opcional), los registros del objetivo *QUEUE* del *IPTables*, por lo que está hecho para plataformas *GNU/Linux* que estén usando *kernel* mayor o igual a 2.4.5. Está escrito en *Perl* y es distribuido bajo la licencia *GNU General Public License*.

SNARE (11)

El significado de sus siglas es *System iNtrusion Analysis & Reporting Enviroment* (Entorno para el análisis y reporte de intrusiones en sistemas). Está dividido en tres componentes claves: Cambios del *kernel* (The *Kernel changes)* para controlar los cambios del sistema*,* Auditor de demonios *(The Snare Audit Deamon)* es la interfaz entre el *kernel* de *GNU/Linux* y el administrador*,* Servidor Micro-Web *(The Snare Micro-Web Server)* para controlar la aplicación desde el navegador web. Está pre-compilado para la mayoría de las "distribuciones de *GNU/Linux*" y es distribuido bajo la licencia *GNU General Public License.* También está disponible para plataforma *MS Windows*. Es una herramienta muy completa, pero como uno de sus componentes está inter-relacionado con el *kernel* de *GNU/Linux* y dicha aplicación tiene poco soporte, se ha quedado obsoleta.

Sawmill (12)

Es un "sistema propietario (licencia privativa)" que se puede ejecutar en la mayoría de las plataformas. Se emplea para el análisis de ficheros log. Esencialmente fue diseñado para trabajar con ficheros log de servidores web, sin embargo es capaz de entender alrededor de ochocientos nueve formatos de archivos *log*. Es fácil de usar con una extensiva documentación. Tiene un paquete de herramientas para un fuerte análisis de log y genera gráficos y reportes. Utilización de base de datos. Fácil de instalar y altamente configurable.

Phpsyslog-ng (13)

Es una aplicación basada en interfaz web, que interactúa con una base de datos en *MySQL,* donde son almacenados los log generados por el demonio *syslog-ng*. Puede trabajar con log provenientes de múltiples máquinas. Presenta características de búsqueda avanzada, permitiendo hacer filtrado por diferentes criterios. Está escrito en *PHP* y es distribuido bajo la licencia *GNU General Public License*.

1.10 Sistemas de detección de intrusos (*IDS*, siglas en inglés).

Es un sistema o aplicación para detectar el acceso no autorizado a una computadora o red. Mediante sensores virtuales obtienen los datos del tráfico en la red. Dicho tráfico se somete a un análisis que usa técnicas de inteligencia artificial para compararlo con ataques conocidos, comportamientos sospechosos como pueden ser: paquetes malformados, escaneo de puertos. No sólo analiza qué tipo de tráfico es, sino que también revisa el contenido y el comportamiento.

Existen, esencialmente, tres tipos de sistemas de detección de intrusos, clasificados de acuerdo a la forma en que se desempeñan:

- *HIDS*(*HostIDS*): es un *IDS* vigilando una única computadora. La ventaja radica en que la carga de procesamiento es bajo.
- *NIDS*(*NetworkIDS*): detecta ataque en segmentos de red pues analiza todo el tráfico de la red.
- *DIDS*(*DistributedIDS*): basado en la arquitectura cliente-servidor, compuesto por una serie de *NIDS*, que actúan como sensores enviando la información a una unidad central, de la cual se puede además recuperar datos.

1.10.1 Ejemplos de IDS.

SNORT: implementa un motor de detección de ataques y barrido de puertos que permite registrar, alertar y responder ante cualquier anomalía previamente definida como patrones. Todo esto es en tiempo real. Está disponible bajo licencia *GNU General Public License*, gratuito y funciona bajo plataformas *MS Windows* y *UNIX/Linux*. Implementa un lenguaje de creación de reglas flexibles, potente y sencillo. Durante su instalación ya nos provee de cientos de filtros o reglas para *backdoor*, *ddos, finger, ftp*, ataques web, *CGI,* escaneos *nmap,* y más.

SAMHAIN: es un sistema de detección de intrusos multiplataforma, de código abierto basado en host. Proporciona comprobación de la integridad de los archivos, así como la detección de *rootkit*, monitoreo de puertos, y los procesos ocultos. Ha sido diseñado para controlar potencialmente múltiples hosts con diferentes sistemas operativos y el mantenimiento centralizado, aunque también puede ser utilizado como aplicación independiente en un único *host*. Tiene monitoreo centralizado, se puede administrar desde un sitio *web* y tiene facilidades de guardar los *log* en diferentes gestores de bases de datos (*MySQL, PostgreSQL, Oracle*, otros.) Es distribuido bajo la licencia *GNU General Public License*.

OSSEC: es un detector de intrusos basado *host* que proporciona análisis de *logs*, verificación de integridad de ficheros, monitorización del registro de *MS Windows*, detección de *rootkits* en tiempo real, y que permite configurar respuestas activas. Funciona en multitud de sistemas operativos como

GNU/Linux, *OpenBSD, FreeBSD, MacOS, Solaris* y *MS Windows*. Es gratuito y distribuido bajo la licencia *GNU General Public License*.

OSSIM: Significa *Open Source Security Information Management*. Su objetivo es proporcionar una recopilación completa de las herramientas que, al trabajar juntas, permiten al administrador de red, obtener una vista detallada sobre todos y cada uno de los aspectos de la red, ordenadores, dispositivos de acceso, servidores, y otros componentes. Proporciona detallados niveles de visualización de las interfaces, así como la presentación de informes y herramientas de gestión de incidentes. Toda esta información puede ser limitada por la red o el sensor, a fin de ofrecer sólo la información necesaria a usuarios específicos. Permite gestionar y saber el nivel de seguridad (métrica) que tiene nuestra empresa. Se trata de un proyecto *Open Source*, con lo que todo el mundo puede disfrutar de él sin ningún coste, además de poder colaborar en su código para formar parte de su evolución. Engloba más de veintidós herramientas de seguridad, entre ellas: *IDS* (*snort*), detector de vulnerabilidades (*nessus*), *firewall* (i*ptables*), detector de sistema operativo (*p0f*), monitorización en tiempo real y estadísticas (*ntop*), detector de anomalías (*RRD*), escaneadores (*nmap*), y otros.

1.11 Sistemas gestores de bases de datos

Son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos y el usuario, las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Tienen como propósito manejar de manera clara, sencilla y ordenada un conjunto de información.

1.11.1 *Oracle*

Es un sistema de administración de bases de datos fabricado por *Oracle Corporation*. Se considera como uno de los sistemas de bases de datos más completos, destacando su soporte de transacciones, estabilidad y escalabilidad. Además, es multiplataforma. Su mayor defecto es su enorme precio, que es de varios miles de euros (según versiones y licencias).

1.11.2 *Microsoft SQL Server*

Es un sistema de gestión de bases de datos relacionales basado en el lenguaje *Transac-SQL*, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea. Soporta transacciones y procedimientos almacenados. Es escalable, estable y seguro.

1.11.3 *MySQL.*

Es uno de los Sistemas Gestores de Bases de Datos (SGBD) más populares desarrolladas bajo la filosofía de código abierto. Entre las características disponibles en las últimas versiones están: amplio subconjunto del lenguaje *SQL*, disponibilidad en gran cantidad de plataformas y sistemas, transacciones y claves foráneas, conectividad segura, replicación, búsqueda e indexación de campos de texto.

1.11.4 PostgreSQL

Es un servidor de bases de datos relacional libre, ofrecido bajo la licencia *BSD*. Sus principales características son: llaves ajenas o llaves foráneas, disparadores, vistas, integridad transaccional, acceso concurrente multiversión (no se bloquean las tablas, ni siquiera las filas, cuando un proceso escribe), capacidad de albergar programas en el servidor en varios lenguajes. Herencia de tablas, tipos de datos y operaciones geométricas.

1.12 Lenguajes de programación

Un lenguaje de programación es una poderosa herramienta utilizada para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

1.12.1 C++

Es un lenguaje de programación, diseñado a mediados de los años 1980, por *Bjarne Stroustrup*, como extensión del lenguaje de programación *C*. Se puede decir que *C++* abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos. Actualmente existe un estándar, denominado *ISO C++,* al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Las principales características del *C++* son las facilidades que proporciona para la programación orientada a objetos y para el uso de plantillas o programación genérica.

1.12.2 Perl

Lenguaje Práctico para la Extracción e Informe. Perl toma características del *C*, del lenguaje interpretado *Shell, AWK, Sed, Lisp* y, en un grado inferior, de muchos otros lenguajes de programación. Estructuralmente, está basado en un estilo de bloques como los del *C* o *AWK*, y fue ampliamente adoptado por su destreza en el procesado de texto y no tener ninguna de las limitaciones de los otros lenguajes de *script*.

1.12.3 *Python*

Es un lenguaje de programación interpretado e interactivo, capaz de ejecutarse en una gran cantidad de plataformas. Es multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación estructurada, programación funcional y programación orientada a aspectos. Otros muchos paradigmas más están soportados mediante el uso de extensiones. Usa tipado dinámico de datos y *reference counting* para el manejo de memoria. Una característica importante del *Python* es la resolución dinámica de nombres, lo que enlaza un método y un nombre de variable durante la ejecución del programa.

1.12.4 *HTML.*

Acrónimo inglés de ***H**yper**t**ext **M**arkup **L**anguage* (lenguaje de formato de documentos de hipertexto), es un lenguaje de marcas diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. Gracias a Internet, el *HTML* se ha convertido en uno de los formatos más populares que existen para la construcción de documentos.

1.12.5 PHP

Acrónimo recursivo de "***P**HP: **H**ypertext **P**reprocessor*". El fácil uso y la similaridad con los lenguajes más comunes de programación estructurada, permiten a la mayoría de los programadores experimentados crear aplicaciones complejas con una curva de aprendizaje muy suave. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones y prácticas.

1.13 Licencias de software

Una licencia de *software* es la autorización o permiso concedido por el titular del derecho de autor, en cualquier forma contractual, al usuario de un programa informático, para utilizar éste en una forma determinada y de conformidad con unas condiciones convenidas.

La licencia, que puede ser gratuita u onerosa, especifica los derechos (de uso, modificación o redistribución) concedidos a la persona autorizada y sus límites. Además, puede señalar el plazo de duración, el territorio de aplicación y todas las demás cláusulas que el titular del derecho de autor establezca.

1.13.1 No libres

Se protege contra uso, modificación o redistribución (copia).Este tipo de licencias no permiten que el software sea modificado, desensamblado, copiado o distribuido de forma ilegal (piratería de *software*), así como regular el número de copias que pueden ser instaladas en un equipo de computo. La mayoría de estas licencias deslindan de toda garantía que pudiese tener el programa.

Algunos ejemplos de este tipo de licencias son las llamadas **CLUF**s: **C**ontrato de **L**icencia para **U**suario **F**inal o **EULA**s: ***E**nd **U**ser **L**icense **A**greement*, por sus siglas en Inglés.

1.13.2 Libres

Existen dos grupos de este tipo de licencia:

- *Sin protección heredada*: se puede crear una obra derivada sin que ésta tenga obligación de protección alguna.
- *Protección heredada*: algunas restricciones se aplican a las obras derivadas.

1.13.2.1 GNU General Public License (GNU/GPL)

Entre las de protección heredada, se encuentra la *GNU General Public License.* Creada por la *Free Software Foundation* (*FSF*), está orientada principalmente a proteger la libre distribución, modificación y uso de *software*. Su propósito es declarar que el software cubierto por esta licencia es *software* libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios. Actualmente va por la versión 3.

1.14 GNU/Linux

GNU/Linux es la denominación defendida por *Richard Stallman* y otros para el sistema operativo que utiliza el *kernel Linux* en conjunto con las aplicaciones de sistema creadas por el proyecto *GNU*. Comúnmente este sistema operativo es denominado simplemente *Linux*, al hacerlo se comete un error no solo técnico sino ético.

1.14.1 Proyecto *Linux*

Linux es el núcleo o *kernel* del sistema operativo libre denominado *GNU/Linux* (también llamado Linux). Lanzado bajo la licencia *GNU/GPL* y desarrollado gracias a las contribuciones de todo el mundo. Linux es uno de los mejores ejemplos de *software open source* cuyos desarrolladores originales siguieron la filosofía del movimiento *open source*. *Linux* fue creado por *Linus Torvalds* en 1991. Muy pronto, la comunidad de *Minix* (un clon del sistema operativo *Unix*) contribuyó en el código y en ideas para el *kernel Linux*.

1.14.2 Proyecto *GNU*

GNU (GNU's Not Unix : *GNU* No es *Unix*) es un acrónimo recursivo que identifica al sistema operativo creado por *Richard Stallman* en cooperación con otros integrantes de la *Free Software Fundation*, cuyo objetivo es reemplazar a cada uno de los programas que integran al sistema operativo *Unix* por uno equivalente que sea libre, creándose así un sistema operativo igual a *Unix* en cuanto a funcionamiento e interfaz, pero que *no es Unix*, sino otro sistema operativo totalmente libre.

1.14.3 Distribuciones

Conjunto de aplicaciones que unidas a una versión del *kernel* conforman un sistema operativo completamente funcional. Normalmente redefinen temas de interfaz, modos de usabilidad y los perfiles de configuración.

1.14.3.1 Debian

Es una asociación o comunidad conformada por desarrolladores y usuarios que pretende crear y mantener un sistema operativo *GNU* basado en *software libre* pre-compilado y empaquetado en un formato sencillo en múltiples arquitecturas y en varios núcleos. Nace como una apuesta por separar en sus versiones el *software* libre del *software* no libre. El modelo de desarrollo del proyecto es ajeno a motivos empresariales o comerciales, siendo llevado adelante por los propios usuarios, aunque cuenta con el apoyo de varias empresas en forma de infraestructuras. *Debian* no vende directamente su software, lo pone a disposición de cualquiera en Internet, aunque sí permite a personas o empresas distribuir comercialmente este software mientras se respete su licencia.

1.14.3.2 Ubuntu

Ubuntu es una distribución *GNU/Linux* que ofrece un sistema operativo predominantemente enfocado a ordenadores de escritorio aunque también proporciona soporte para servidores. Basada en *Debian GNU/Linux*, Ubuntu concentra su objetivo en la facilidad de uso, la libertad de uso, los lanzamientos regulares y la facilidad en la instalación. Es patrocinado por *Canonical Ltd*., una empresa privada fundada y financiada por el empresario sudafricano *Mark Shuttleworth*.

El nombre de la distribución proviene del concepto *zulú* y *xhosa* de *ubuntu*, que significa "humanidad hacia otros" o "yo soy porque nosotros somos". El eslogan de Ubuntu – "Linux para seres humanos" (en inglés "*Linux for Human Beings*") – resume una de sus metas principales: hacer de *GNU/Linux* un sistema operativo más accesible y fácil de usar.

1.14.3.3 FreeBSD

Es un sistema operativo libre, multiusuario para ordenadores personales basado en las *CPU* de arquitectura *Intel*. Capaz de efectuar multitarea con apropiación y multiproceso en plataformas compatibles con múltiples procesadores; el funcionamiento de *FreeBSD* está inspirado en la variante 4.4 *BSD-Lite* de *UNIX*.

1.14.3.4 Gentoo

Gentoo Linux es una distribución *GNU/Linux* orientada a usuarios con cierta experiencia en este sistema operativo, fue fundada por *Daniel Robbins*, basada en la inactiva distribución llamada *Enoch Linux*. El nombre *Gentoo* proviene del nombre en inglés del pingüino papúa.

1.14.3.5 NovaLnx

Es una distribución hecha en la Universidad de las Ciencias Informáticas (UCI). Está basada en la distribución *Gentoo Linux.* Es utilizada en diferentes áreas de la universidad, ya sea de docencia, producción o trabajo. Las Fuerzas Armadas Revolucionarias han apostado por el uso de esta distribución, y en su migración a *software* libre, esta fue la distribución *GNU/Linux* escogida.

1.15 Conclusiones

Por la prioridad que tiene para el mundo en general y para Cuba en particular**, la seguridad informática;** se hace necesario utilizar herramientas cada ves más técnicamente perfeccionadas que posibiliten el monitoreo, control y registro centralizado de los eventos y sucesos de la red de datos.

La utilización de software libre permite mayor disponibilidad, integridad y confidencialidad en la redes de datos y/o centros de cálculo.

Capítulo 2. Descripción de la plataforma *OSSIM*.

2.1 Introducción.

En el presente capítulo se hace una descripción detallada del funcionamiento interno, flujo de datos, algoritmos de procesamiento y funcionalidades del *framework* de *OSSIM*. Además, se describen las herramientas más relevantes que integra *OSSIM* para garantizar la fiabilidad de la gestión que hace. También se hace mención a las ventajas y limitaciones del empleo de esta plataforma centralizada de control de eventos y sucesos de una red de datos.

2.2 Plataforma *OSSIM*.

Es sorprendente que con el gran desarrollo tecnológico producido en los últimos años, del cual provienen herramientas con capacidades como la de los IDS, sea tan complejo desde el punto de vista de seguridad, obtener una foto de una red u obtener cierta información con un grado de abstracción que permita una revisión práctica y asumible.

Algo falla en la detección de anomalías en las redes corporativas, aparentemente existe la tecnología apropiada, a través de diferentes sistemas se detectan los eventos más concretos, sin embargo es muy complejo revisar todas las alertas que estos envían debido a dos razones:

- la cantidad
- la poca fiabilidad

En otras palabras, son demasiadas alertas y estas no son fiables, es decir, demasiados falsos positivos. Así mismo, se obtiene información muy detallada, pero atómica, parcial y sin capacidad de abstracción, por lo que no se puede detectar ataques definidos por comportamientos más complejos, lo que proporciona un segundo problema que son los falsos negativos.

OSSIM es una distribución de productos *open source* integrados para construir una infraestructura de monitorización de seguridad. Su objetivo es ofrecer un marco para centralizar, organizar y mejorar las capacidades de detección y visibilidad en la monitorización de eventos de seguridad de una organización.

Surge con el propósito de suplir una de las grandes necesidades que los profesionales del mundo de la seguridad informática tienen día a día. Para mejorar esta situación, *OSSIM* tiene como base el concepto de CORRELACIÓN, lo que se traduce en la posibilidad de obtener una visibilidad de todos los eventos de los sistemas en un punto y con un mismo formato, y a través de esta situación

privilegiada relacionar y procesar la información para detectar, monitorizar, priorizar y organizar el estado de la seguridad de nuestra red a través de los registros generados por los distintas aplicaciones.

La idea de correlación está también implícita en *OSSIM* en el sentido de agregación e integración de productos, pues incluye un número de magníficos productos desarrollados en estos años en un *Framework* general que permite nuevas posibilidades al interrelacionar todas sus funcionalidades. El grado de precisión de *OSSIM* es muy elevado debido a la VALORACIÓN DE RIESGOS, capacidad que tiene para decidir en cada caso cuando un evento que afecta a un activo y representa una amenaza debido a su procedencia, y ha sido detectada a través de sensores fiables, implica la necesidad de ejecutar una acción u otra.

Sin lugar a dudas, *OSSIM* es un sistema altamente complejo pues es capaz de implementar una Política de Seguridad, un Inventario de la Red, Inventario de Hardware, nos ofrece un Monitor de Riesgos en tiempo Real, Control de *logs* de seguridad, todo ello configurado y gestionado desde un *Framework* sin dejar de cumplir con el objetivo de la integración de productos.

2.3 Conceptos y definiciones previas.

Parte fundamental de la funcionalidad de *OSSIM* es aumentar la capacidad de detección. Para poder comprender esto, así como su funcionamiento y estructura, se da una breve introducción a ciertos términos relacionados con el tema.

2.3.1 Detectores

Se define como detector a cualquier programa capaz de procesar información en tiempo real, información normalmente a bajo nivel como tráfico o eventos de sistema y lanzar alertas ante la localización de situaciones previamente definidas. La definición de estas situaciones se puede hacer de dos formas:

- A través de patrones, o reglas definidas por el usuario
- A través de grados de anomalía

2.3.2 Capacidad de detección

La capacidad de detección ha aumentado enormemente en los últimos años, si se piensa, por ejemplo en su máximo exponente los *IDS*, capaces de detectar patrones al nivel más bajo de detalle. Para hablar sobre la capacidad de un detector se define mediante dos variables:

- Sensibilidad: capacidad de análisis, en profundidad y complejidad, que posee el detector a la hora de localizar una posible anomalía.
- Fiabilidad: es el grado de certeza que ofrece el detector ante el aviso de un posible evento.

2.3.3 Incapacidad de detección

Se comprueba que pese al desarrollo "en la profundidad de detección" de estos sistemas, dista de que su capacidad sea aceptable. De la incapacidad de los detectores de afrontar estas dos propiedades provienen los dos principales problemas de la actualidad:

- Falsos Positivos. La falta de fiabilidad de los detectores es el causante de los falsos positivos, es decir alertas que realmente no corresponden con ataques reales.
- Falsos Negativos. La incapacidad de detección implicaría que un ataque es pasado por alto.

Se define entonces, como "Proceso de Detección" al proceso global realizado por el sistema, incluyendo tanto los diferentes detectores y monitores de la organización como los realizados por el *framework* para procesar esta información. Implica normalmente tres fases bien distinguidas:

- Pre-proceso: 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.
- Post-proceso: El tratamiento que realizaremos una vez tenemos toda la información centralizada

2.3.4 Post-proceso

El pre-proceso y la colección son capacidades ya clásicas y no aportan nada nuevo, pero en el post-proceso, una vez obtenida toda la información en un mismo punto, *OSSIM* implementa mecanismos para poder mejorar la sensibilidad y fiabilidad de la detección. Aumenta la complejidad del tratamiento incluyendo métodos que se encargan de descartar falsos positivos o al contrario priorizar o descubrir patrones más complejos que los detectores han pasado por alto. *OSSIM* desarrolla tres métodos en el post-proceso:

- Priorización: se priorizan las alertas recibidas mediante un proceso de contextualización desarrollado a través de la definición de una Política Topológica de Seguridad y el Inventariado de los sistemas.

- Valoración de Riesgo: Cada evento es valorado respecto del Riesgo que implica, es decir, de una forma proporcional entre el activo al que aplica, la amenaza que supone y la probabilidad del evento.
- Correlación: se analizan un conjunto de eventos para obtener una información de mayor valor.

Método	Efecto
Priorización	Aumenta la Fiabilidad
Valoración de Riesgo	Aumenta la Fiabilidad
Correlación	Aumenta la Fiabilidad, Sensibilidad, y Abstracción.

Tabla # 1. Correlación método-efecto.

Por lo tanto, *OSSIM* se nutre de "alertas" contenidas en los registros de seguridad ofrecidas por los detectores y produce, tras el tratamiento de las mismas las "alarmas".

Una alarma es normalmente el resultado del proceso de varias alertas, tiene un mayor grado de abstracción permitiendo localizar patrones más complejos, lo que aumenta el grado de fiabilidad.

2.4 Arquitectura.

En la siguiente figura se muestra la arquitectura general según los procesos realizados, donde se pueden percibir tres bases de datos y las funcionalidades principales, en las cuales son detallas más adelante.

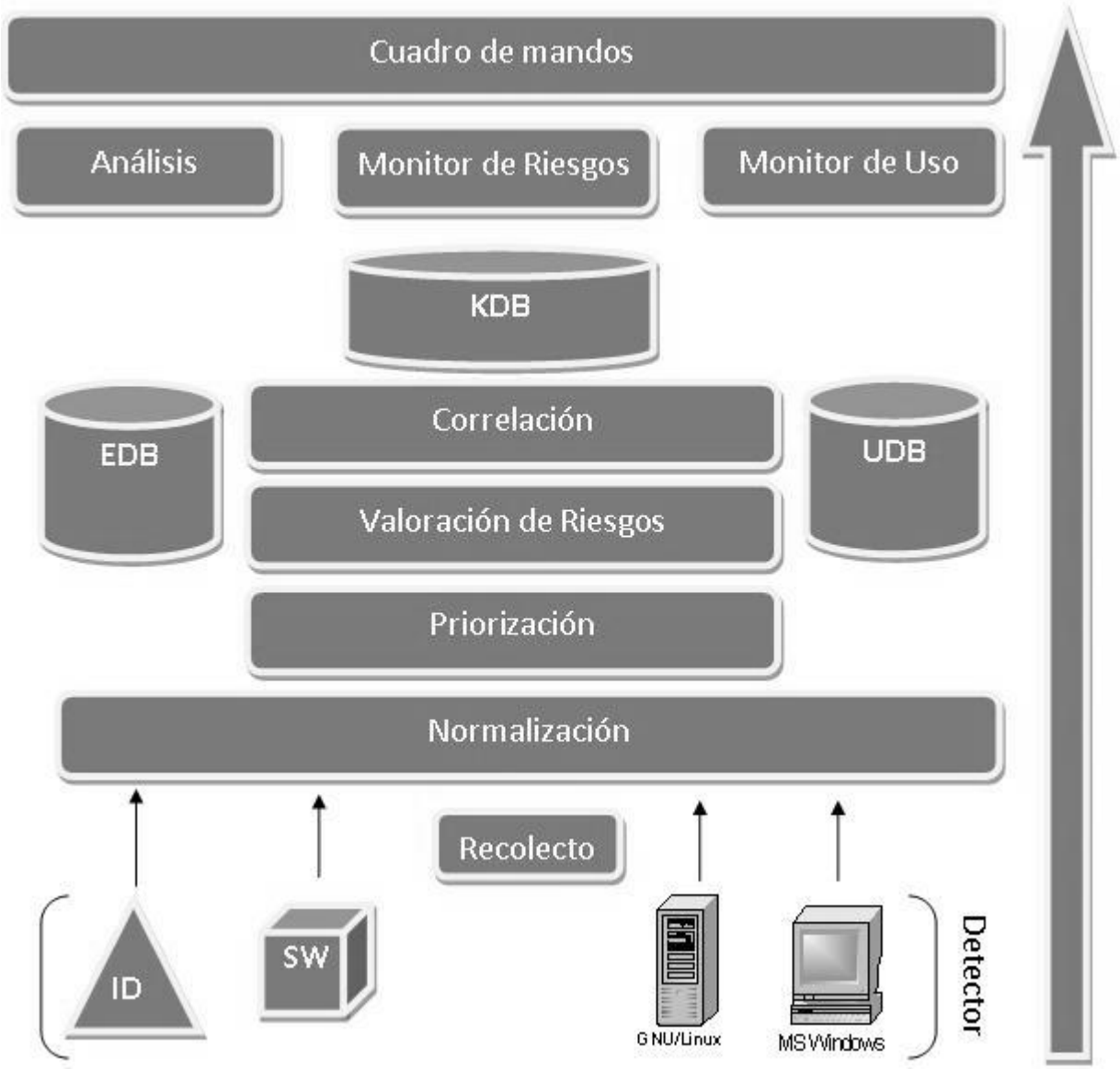

Figura # 3. Arquitectura general según los procesos.

- *EDB*, la base de datos eventos, la más voluminosa pues aloja todos los eventos individuales recibidos de los detectores.
- *KDB*, la base de datos del *Framework*, en la cual parametriza el sistema para que conozca la red y define la política de seguridad.
- *UDB*, la base de datos de perfiles, que almacena todos los datos aprendidos por el Monitor de Perfiles

2.4.1 Arquitectura de la distribución

OSSIM está definido como una "distribución" en vez de un producto, esto significa que el objetivo es integrar antes que desarrollar, es decir, lograr integrar productos y que se "hablen" entre ellos. Para ello se han definido dos niveles:

2.4.1.1 Núcleo

Lo que una distribución típica llamaría el *Núcleo*, desarrollado en el proyecto *GNU* de *OSSIM* y donde se desarrollan las siguientes tareas:

- Se define la estructura de datos
- Se ofrece las interfaces para hablar con los diferentes productos
- Se realiza lo que anteriormente se ha definido como "post-proceso"
- Se ofrece el la primera capa de administración a través de un *Framework*, que enlaza con los diferentes sistemas de administración.
- Se implementa el Cuadro de Mandos.

2.4.1.1 Productos terceros

Productos terceros no desarrollados por el grupo de desarrollo del proyecto *OSSIM* pero integrados en la solución. De este tipo de productos hay dos posibilidades:

- Productos *Open Source*. Que dependiendo de los casos son modificados y/o parcheados y la mayoría de las veces están incluidos en la distribución.
- Productos de pago. Que lógicamente no están incluidos en la distribución ni son parcheados ni modificados.

2.5 Funcionalidad.

Mediante los siguientes nueve niveles se puede tener una apreciación gráfica y simplificada de la funcionalidad de *OSSIM*:

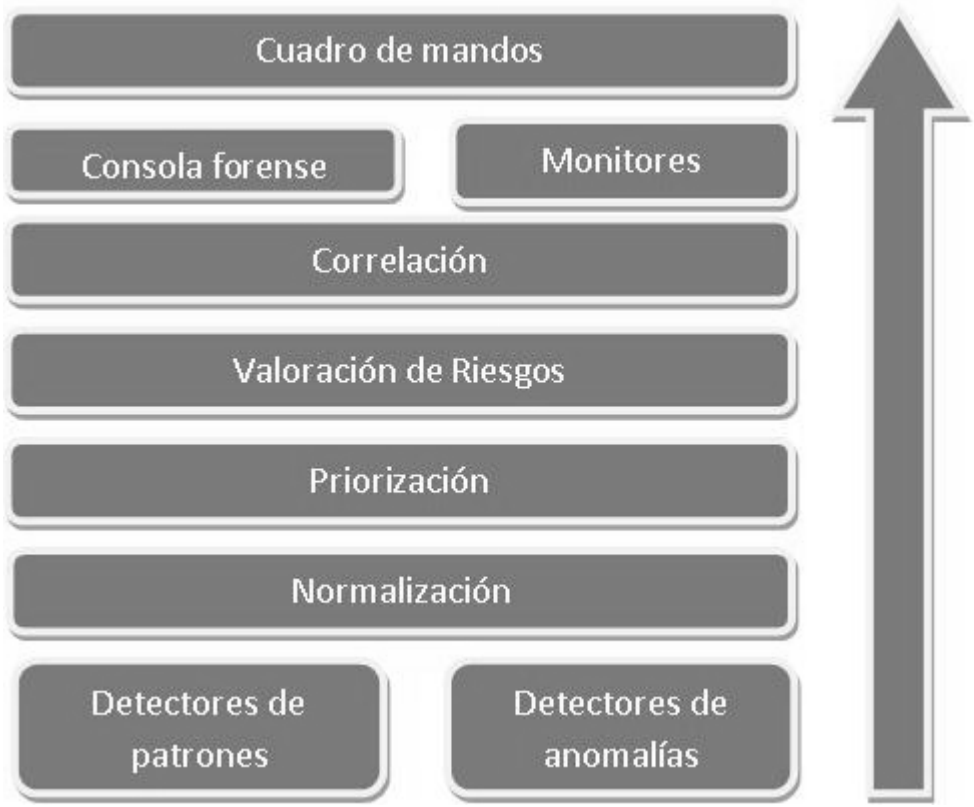

Figura # 4. Funcionalidades del sistema *OSSIM.*

2.5.1 Detectores de patrones

La mayoría de los detectores clásicos funcionan con patrones, el ejemplo más claro es el *IDS* o sistema de detección de intrusos, sistema capaz de detectar patrones definidos a través de firmas o reglas. Existen otra serie de detectores de red incluidos en la mayoría de los dispositivos como *routers* o *Firewalls*, capaces de detectar por ejemplo escaneo de puertos, intentos de *spoofing*, o posibles ataques por fragmentación. Existe además, detectores para los eventos de seguridad de un sistema operativo, casi todos ellos incluyen su propio *logger* como el de *GNU/Linux* llamado *syslog*, siendo capaces de alertar de posibles problemas de seguridad.

En definitiva cualquier elemento de la red, como un *router*, un puesto de trabajo, el *Firewall*, u otro incluye la capacidad de detección en mayor o menor medida; *OSSIM* se interesa en recibir los eventos

de todos los sistemas críticos para de esta forma cumplir con uno de los objetivos principales: *"la Visibilidad de la red"*.

2.5.2 Detectores de anomalías

La capacidad de detección de anomalías es más reciente que la de patrones. En este caso no hay que especificar en el sistema de detección 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 complementa la detección de patrones pues la naturaleza de los dos procesos no es diferente sino opuesta.

Este tipo de detección es especialmente útil para prevenir actividades perimetrales, estas 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, todo.

Otros de los casos en los que estos detectores son útiles:

- Un nuevo ataque del cual no existen todavía firmas podría traspasar los sistemas de detección de patrones pero producirá una anomalía clara.
- Las actividades maliciosas internas, por parte de cualquier persona dentro de nuestra red, no implican la violación de ninguna política ni la ejecución de ningún *exploit*. Pero sí una anomalía en el uso y la forma de uso de un servicio.
- Un gusano que se ha introducido en la organización, un ataque de *spamming*, o el mismo uso de programas *P2P*, generarían un número de conexiones anómalas fácilmente detectable.
- Así mismo se puede detectar:
    - Usos de servicios anormales por origen y destino
    - Usos en horario anormal
    - Exceso en el uso de tráfico o conexiones
    - Copia anormal de ficheros en la red interna
    - Cambios en el sistema operativo de una máquina

Como efecto negativo estos detectores generan un número de nuevas alertas, amplificando la señal y empeorando el problema (el objetivo es limitar el número de alertas), sin embargo tomándose como información adicional que acompaña a las clásicas alertas de patrones permite cualificar y por lo tanto diferenciar aquellas que puedan implicar una situación de mayor de riesgo.

2.5.3 Normalización

La normalización y centralización o agregación tiene como objetivo unificar en única consola y formato los eventos de seguridad de todos los sistemas críticos de la organización.

Todos los productos de seguridad poseen normalmente la capacidad de gestión centralizada a través de protocolos estándar, la agregación es por lo tanto sencilla utilizando estos protocolos. En las más reciente versiones de *OSSIM* se tiene en cuenta una arquitectura de cifrado e identificación mediante firma la cual aumenta la confidencialidad y autenticación para entornos que lo requieren.

La normalización implica la existencia de un *"parser"* o traductor que conozca los tipos y formatos de alertas de los diferentes detectores. De esta forma se puede observar en la misma pantalla y con un mismo formato los eventos de seguridad de un determinado momento, ya sean del *router*, el *Firewall*, del *IDS*, o del servidor *GNU/Linux*.

Al tener centralizados en la misma base de datos todos los eventos de la red se obtiene una gran "visibilidad" de lo que ocurre en ella, lo que posibilita desarrollar procesos que permiten detectar patrones más complejos y distribuidos.

2.5.4 Priorización

La prioridad de una alerta debe ser dependiente de la Topología y el Inventario de sistemas de la organización, las razones son bastante claras como muestran estos ejemplos:

- Si una alerta que se refiere a un ataque al servicio *IIS* de *Microsoft* llega de una máquina con sistema operativo *GNU/Linux* y servidor *Apache*, la alerta debe ser desprioriza da.
- Si existe una conexión sospechosa de un usuario sobre un servidor, el sistema debe:
  - Darle máxima prioridad si el usuario es externo y ataca a la base de datos de clientes.
  - Darle prioridad baja si el usuario es interno y ataca a una impresora de red.
  - Descartarla pues es un usuario que normalmente hace pruebas contra un servidor de desarrollo.

OSSIM permite realizar estas tareas debido a su proceso de priorización el cual contextualiza, es decir, la evalúa de la importancia de una alerta respecto al escenario de la organización. Este escenario está descrito en una base de conocimiento sobre la red del cliente formada por:

- Inventario de Máquinas y Redes (identificadores, sist. operativo, servicios)
- Política de Accesos (desde donde a donde está permitido o prohibido)

Para efectuar esto está el *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.

2.5.5 Valoración de riesgos

La importancia que *OSSIM* le da a un evento debe ser dependiente de los siguientes tres factores:

- El valor del Activo al que el evento se refiere.
- La Amenaza que representa el evento.
- La Probabilidad de que este evento ocurra.

2.5.5.1 Riesgo intrínseco

Con los factores mencionados anteriormente se construye la definición clásica de riesgo: el valor del posible impacto de una amenaza sobre un activo ponderado con la probabilidad de que este ocurra.

La valoración de riesgos se ha referido clásicamente a riesgos intrínsecos, o riesgos latentes, es decir riesgos que soporta una organización derivados del hecho de "ser" (los activos que posee para desarrollar su negocio) y "estar" (las amenazas circunstanciales que existen sobre estos activos).

2.5.5.2 Riesgo instantáneo

OSSIM, debido a la capacidad de medir en tiempo real, mide el riesgo asociado a la situación actual, en términos instantáneos. En este caso el riesgo es medido como la medida ponderada del daño que produce y la probabilidad de que este ocurriendo en el momento. Esta probabilidad, derivada de la imperfección de los sensores, no será más que el grado de fiabilidad de estos en la detección de la posible anomalía en curso.

En fin, "riesgo instantáneo" es la situación de riesgo producida por la recepción de una alerta, valorada de forma instantánea como la medida ponderada entre el daño que produce la anomalía y la fiabilidad del detector que lo reporta.

OSSIM calcula el "riesgo instantáneo" de cada evento recibido que es la medida objetiva que utiliza para valorar la importancia que un evento puede implicar en términos de seguridad, sólo a través de esta medida se valora la necesidad de actuar.

2.5.6 Correlación

Se define una función de correlación como un algoritmo que realiza una operación a través de unos datos de entrada y ofrece un dato de salida.

Si se analiza la información recogida por los detectores y monitores como información específica pero parcial, se dibujan pequeñas zonas del espectro de toda la información que es de interés tener. Utilizar la capacidad de correlación es aprovechar estos sistemas y a través de una nueva capa de proceso llenar otras zonas de ese espectro infinito de toda la información que podría existir de una red.

En contra de esta idea podía intentar instalarse un sistema único con un detector capaz de localizar toda la información posible de la red, pero para ello se necesitaría una visibilidad total desde un punto único y una capacidad de almacenamiento y de memoria casi ilimitada.

Los sistemas de correlación son por tanto artificios que suplen la falta de sensibilidad, fiabilidad y la visibilidad limitada de los detectores.

2.5.6.1 Modelo de correlación

OSSIM desarrolla un modelo de correlación con un amplio espectro para poder:

- Desarrollar patrones específicos para detectar lo conocido y detectable.
- Desarrollar patrones ambiguos para detectar lo desconocido o no detectable.
- Poseer una máquina de inferencia configurable a través e reglas relacionadas entre sí capaz de describir patrones más complejos.
- Permitir enlazar Detectores y Monitores de forma recursiva para crear cada vez objetos más abstractos y capaces.
- Desarrollar algoritmos que ofrezcan una visión general de la Situación de Seguridad.

2.5.6.2 Métodos de correlación

Para lograr estos objetivos utiliza dos métodos de correlación diferentes:

- Correlación mediante Secuencias de Eventos. Focalizado en las anomalías conocidas y detectables, relaciona a través de reglas que implementarán una máquina de estados los patrones y comportamientos conocidos que definen una anomalía.
- Correlación mediante Algoritmos Heurísticos. Tomando una aproximación opuesta implementa algoritmos que mediante funciones heurísticas detectan situaciones de riesgo. Este método detecta situaciones sin conocer ni ofrecer detalle de las mismas, intenta suplir pues la incapacidad de los anteriores métodos y es útil para detectar anomalías.

2.5.7 Monitores

Son simples monitores de procesos pero como tal son funcionalidades que vale la pena destacar debido al hecho de que muestran información valiosa y lo hacen de una manera sencilla y fácilmente entendible por las personas. Básicamente, la información que se muestra a través de estos monitores es mediante gráficos, *flash* y tablas que consolidan los estados de diversas herramientas, *plugins*, dispositivos entre otros.

2.5.7.1 Monitor de riesgos.

OSSIM posee un monitor de riesgos que dibuja los valores producidos por el algoritmo *CALM*, valores que miden el nivel de riesgo de compromiso (C) y de ataque (A) derivados de la recepción de alertas que indican la posibilidad de que una máquina ha sido comprometida, o está siendo atacada.

2.5.7.2 Monitor de Uso, Sesiones y Perfiles.

OSSIM da mucha importancia a la monitorización detallada de cada máquina y perfil. Tiene tres tipos de monitorización para ello:

- 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: 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 esta máquina en la red.

Cualquiera de estos tres son imprescindibles para un sistema de seguridad, en caso contrario, el administrador de seguridad estará ciego ante eventos pasados, no podrá valorar lo normal de lo anormal y no será capaz de “observar” su red.

2.5.7.3 Monitor de caminos

Este monitor es capaz de dibujar en tiempo real los caminos trazados en la red entre las diferentes máquinas que realizan conversaciones o enlaces entre ellas. El dibujo se realiza en un intervalo de tiempo creando un grafo cuyas ramas caducan en el tiempo.

El monitor obtiene sus datos de otros dos monitores: el de sesiones donde están localizados cada uno de los enlaces del momento, y del monitor de riesgo de donde obtiene el nivel de riesgo de cada máquina para dibujar cada una con un color diferente y calcular el riesgo agregado de cada uno de estos grafos. La monitorización de enlaces tiene a su vez dos métodos:

- Análisis de conexión persistente (*TCP*): Mediante el cual se dibuja únicamente sesiones *TCP* persistentes.
- Análisis de conexiones débiles: Mediante el cual se dibuja todos los enlaces percibidos en la red, tanto *UDP* como *TCP* como *ICMP* lo cual puede implicar en muchos casos un mapa de red caótico.

2.5.8 Consola forense

La Consola Forense permite acceder a toda la información recogida y almacenada por el colector. Esta consola es un buscador que accede a la base de batos 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.

Al contrario que el Monitor de Riesgos referido anteriormente, esta consola permite profundizar al máximo detalle sobre cada uno de los eventos ocurridos en el sistema.

2.5.9 Cuadro de mandos

La última de las funcionalidades es el Cuadro de Mandos, mediante la cual se obtiene una visión de alto nivel de la situación de la red respecto a seguridad.

El cuadro de mandos monitoriza una serie de indicadores que miden el estado de la organización respecto de seguridad.

Permite definir una serie de umbrales u objetivos que debe cumplir la organización. Estos umbrales serán definidos de forma absoluta o relativa como un grado de anomalía. Así mismo, es configurable el envío de alarmas cuando se superen estos umbrales o la ejecución de cualquier procedimiento automático.

El cuadro de mandos es completamente personalizado y hecho a medida. Es como un termómetro general de todo lo que ocurre en la red. A través de él se enlazan cada una de las herramientas de monitorización para profundizar sobre cualquier problema localizado.

2.6 Flujo de datos.

Es necesario, para una mejor comprensión el recorrido del flujo de datos desde la generación de un evento:

- Los eventos son procesados por los detectores hasta que, bien por la localización de un patrón o una anomalía se produce una alerta.

- Las alertas son procesadas en caso de ser necesario por los recolectores antes de ser enviadas. Estos se encargarán de enviar la información agrupada para ocupar el mínimo ancho de banda.
- Las alertas son recibidas a través de diferentes protocolos abiertos de comunicación.
- El *"parser"* se encarga de normalizarlas y guardarlas si procede en base de datos de eventos.
- El *"parser"* se encarga así mismo de cualificarlas determinando su prioridad según la política de seguridad definida en el *framework* y los datos sobre el sistema atacado localizados en el Inventario de Sistemas.
- El *"parser"* valora el riesgo instantáneo que implica la alerta y en caso de ser necesario envía una alarma al Cuadro de Mandos.
- Las alertas cualificadas son enviadas a cada uno de los procesos de correlación que actualizan sus variables de estado y eventualmente lanzan nuevas alertas con una información más completa o fiable. Estas alertas son enviadas nuevamente al *"parser"* para su almacenamiento, priorización, valoración del riesgo, y las otras operaciones que normalmente se realizan.
- El monitor de riesgos visualiza periódicamente la situación de cada uno de los índices de riesgo según han sido calculados por el algoritmo *CALM*.
- El cuadro de mandos muestra las alarmas recientes, actualiza el estado de cada uno de los índices los comparará respecto de los umbrales, y lanza nuevas alarmas o realiza las acciones correspondientes en caso de ser necesario.
- El administrador puede desde el cuadro de mandos enlazar y visualizar a través de la consola forense todos los eventos ocurridos en el momento de la alerta.
- Puede además, comprobar el estado instantáneo de la máquina a través de los monitores de uso, perfiles, y sesiones.

La siguiente figura muestra lo que se ha descrito anteriormente:

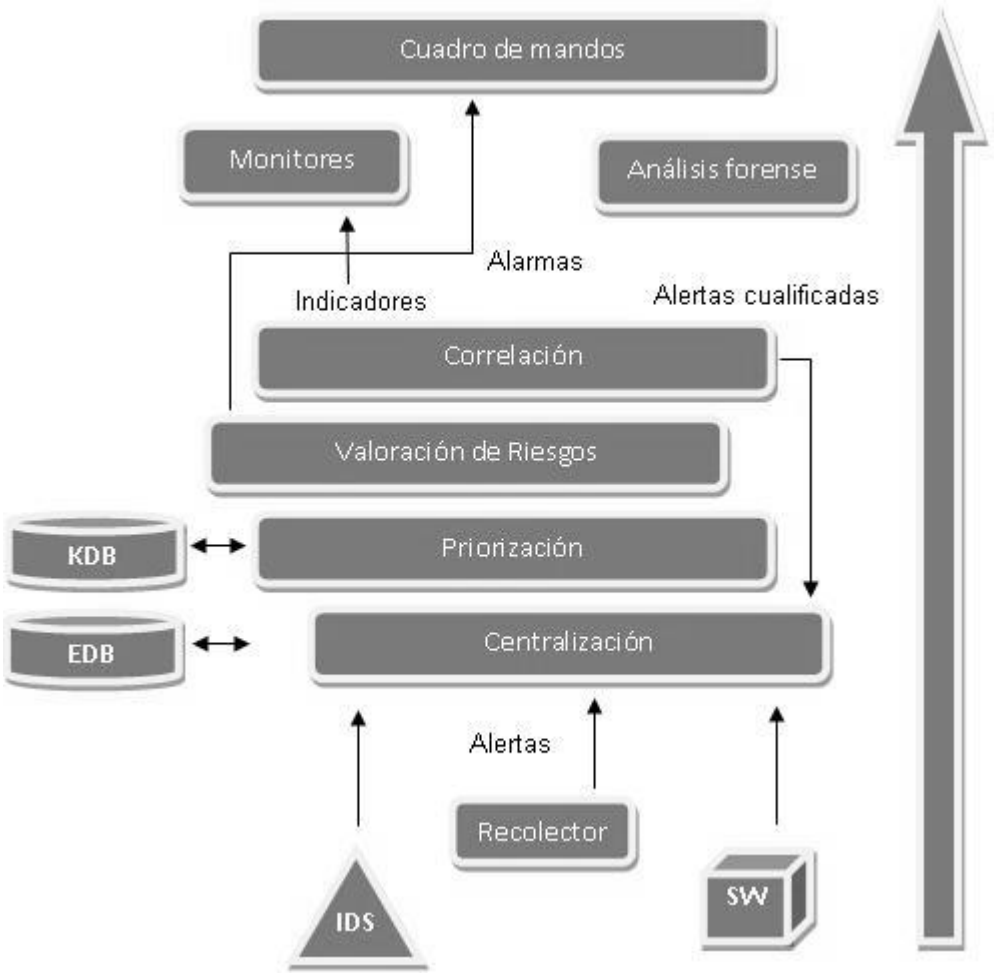

Figura # 5. Flujo de datos.

2.7 Descripción de los principales productos integrados.

Se ha planteado con anterioridad que *OSSIM* integra excelentes productos existentes que realizan tareas de detección de anomalías. Dichos productos presentan su propia estructura y funcionamiento por lo que es necesario hacer una breve descripción de los principales.

2.7.1 OSSEC

OSSEC HIDS es un sistema de detección de intrusión de código abierto basado en *HOST* (computadoras). Permite el análisis de log, chequeo de integridad de ficheros, monitoreo del registro

de sistemas *MS Windows*, detección de *rootkits,* alertas basadas en el tiempo y respuesta activa. Se puede utilizar tanto para el monitoreo de un solo active como para toda una red corporativa. Uno de los principales beneficios de *OSSEC* es su escalabilidad permitiendo monitorear múltiples sistemas desde un punto central.

2.7.1.1 Arquitectura y flujo de datos

La siguiente figura da una noción de cómo funciona *OSSEC* en una red:

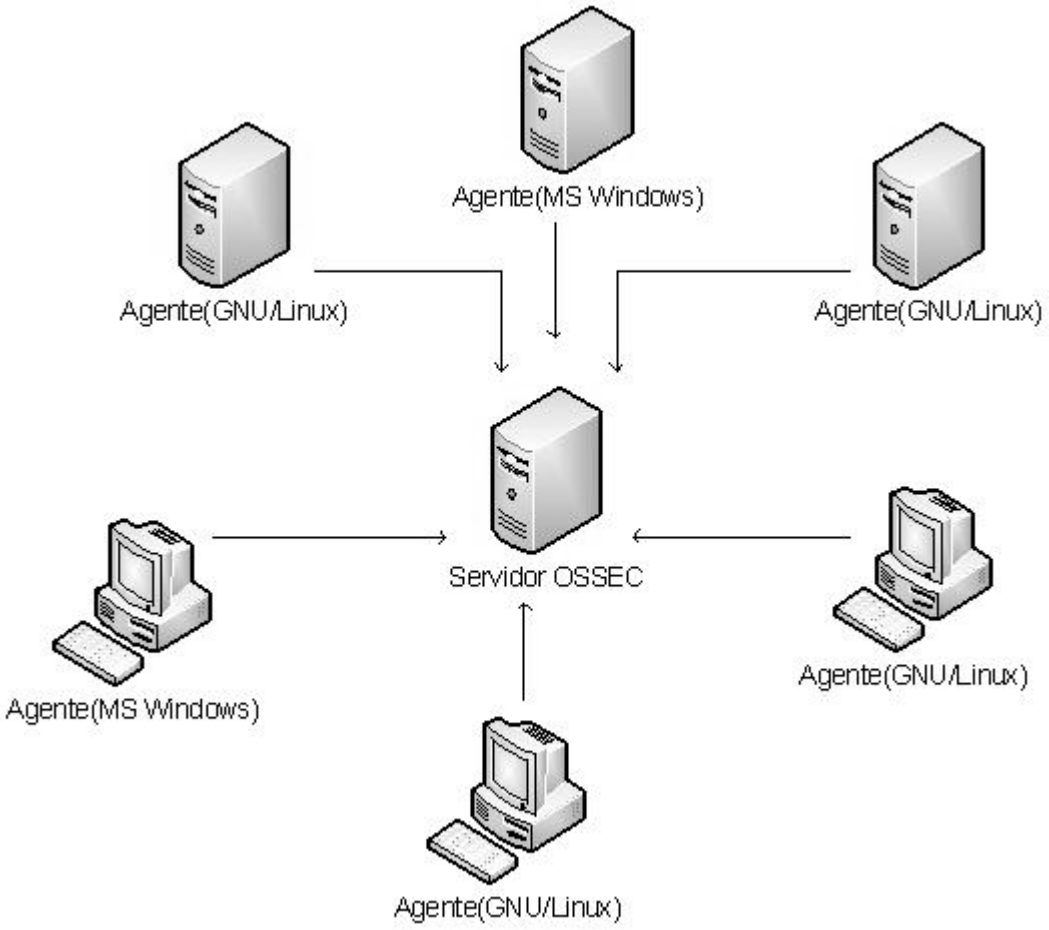

Figura # 6. Arquitectura de *OSSEC.*

Tanto los agentes *MS Windows* como los *GNU/Linux*, situados en servidores o estaciones de trabajo envían, hacia el servidor *OSSEC*, mediante una conexión segura y encriptada, los *log* de seguridad de todo lo que se realiza en cada activo para que sean insertados en una base de datos y luego ser procesados y mostrados al usuario mediante interfaz web; gráficamente:

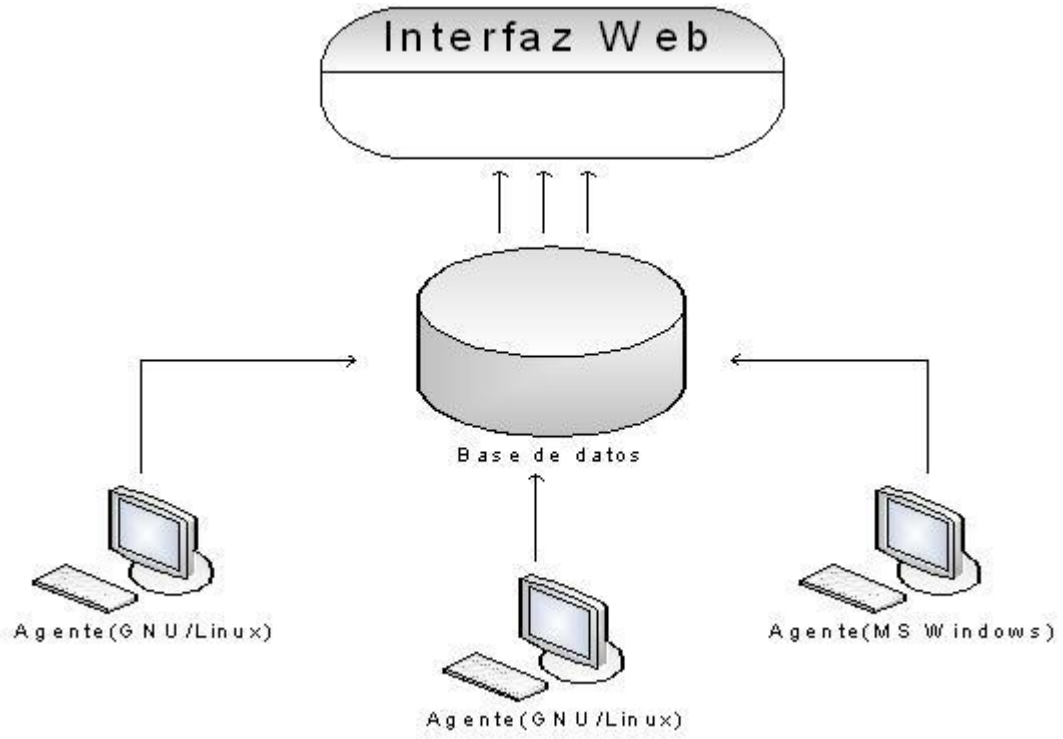

Figura # 7. Flujo de datos de *OSSEC*.

2.7.2 *Snort.*

Cariñosamente conocido por los diseñadores como “el cerdo” es un sistema de detección basado en la red que usa la detección de firmas; prácticamente indaga y examina los paquetes de datos de la red verificando si concuerda con ataques conocidos. Las principales características que lo distinguen del resto de los de su tipo son:

- Configurable
- Libre y gratuito
- Ampliamente usado
- Funciona en múltiple plataformas
- Constantemente actualizado

Básicamente tiene cinco componentes a través de los cuales fluye la información desde que es tomada de la red:

- Librería de captura de paquetes. Es quien se encarga de tomar los paquetes de la tarjeta de red. En sistemas *GNU/Linux* y *Unix*, dicha librería es *libpcap*; en sistemas *MS Windows*, la usada es *WinPcap*.
- Decodificador de paquetes. El paquete enviado por la librería de captura es decodificado para obtener toda la información de los protocolos contenida en dicho paquete la cual será procesada con posterioridad.
- Preprocesador. Contiene varios *plug-ins* los cuales pueden ser activados o no. El procesamiento ocurre sobre la información obtenida del paquete decodificado. Esta actividad determina si se alerta, clasifica o desecha el paquete antes de ser enviado a un proceso más intenso de detección.
- Mecanismo de detección. Es el centro de s*nort*. Toma la información llegada desde el preprocesador y mediantes las reglas predefinidas en los *plug-ins.* Dichas reglas contienen las "firmas" de los ataques conocidos.
- Salida. Una vez realizado todo lo anterior, *snort* da una información que puede ser vista en formato texto, binario o insertada en base de datos o enviadas a otros programas como *syslog*.

2.7.3 OCS *Inventory NG.*

Es una aplicación diseñada para ayudar al administrador de red en mantener un registro de la configuración de las computadoras y sistemas instalados en la red. Además es capaz de detectar todos los dispositivos activos en la red, tales como *switch, router*, impresoras de red y dispositivos desatendidos. Para cada uno guarda *MAC Address* y el *IP*, permitiendo clasificarlos luego. Cuando se está ejecutando la consola de administración, bajo sistemas *GNU/Linux*, también se puede escanear una *IP* o subred en busca de información detallada sobre activos no inventariados.

Esta aplicación utiliza un agente el cual funciona en las computadoras clientes, y un servidor de administración, quien centraliza los resultados de los inventarios permitiendo visualizarlos y realizar operaciones con ellos.

La comunicación entre los agentes y el servidor de administración se hace a través de los protocolos *HTTP/HTTPS*. Todos los datos son comprimidos para reducir el tráfico de red.

El servidor de administración cuenta con cuatro componentes principales:

- Servidor de base de datos: almacena la información de los inventarios.

- Servidor de comunicación: maneja las comunicaciones *HTTP* entre el servidor de base de datos y los agentes.
- Servidor de despliegue: almacena todos los paquetes de configuración de los agentes (requiere *HTTPS*)
- Consola de administración: permite a los administradores consultar la base de datos a través de cualquier navegador web.

Estos cuatro componentes pueden estar instalados en un solo servidor o en diferentes, con el objetivo de balancear la carga. Para más de diez mil computadoras inventariadas, se recomienda el uso de al menos dos servidores: uno para servidor de base de datos y servidor de comunicación, el otro para replica de la base de datos, servidor de administración y servidor de despliegue. Para establecer esta arquitectura distribuida solo se puede hacer en sistemas *GNU/Linux*, pues el paquete de instalación para *MS Windows* está todo integrado.

2.7.4 *Nessus*.

Es una gran herramienta diseñada para automatizar la prueba y descubrimiento de problemas de seguridad. Además ayuda a identificar y resolver dichos problemas. *Nessus* es un programa libre y gratis liberado bajo la licencia *GNU/GPL*. Una de las características más significativas es su tecnología cliente-servidor. Los servidores pueden ser situados en varios puntos estratégicos de la red permitiendo obtener pruebas desde diferentes puntos de vista. Un cliente central o varios distribuidos pueden controlar todos los servidores. El servidor funciona en sistemas *GNU/Linux*, *Mac OS X* e *IBM/AIX* pero es más fácil de instalar en el primero. Los clientes funcionan tanto en plataformas *MS Windows* como *GNU/Linux*.

2.7.5 *Ntop.*

Es un programa que a partir de tráfico de red muestra el uso de esta. *Ntop* está basado en la librería *libpcap* y está programado de forma que puede funcionar en plataformas *MS Windows*. La información que obtiene puede ser vista a través de cualquier navegador web. *Ntop* puede ser visto como un simple agente con una interfaz web embebida.

El uso de:

- Una interfaz web.
- Limitada configuración y administración vía interfaz web.
- Reducido uso del *CPU* y la memoria (esto varia de acuerdo al tamaño y el tráfico de la red)

hace de *Ntop* una herramienta de muy fácil de empleo y que se utilice para monitorear varios tipos de redes.

Además, *Ntop* brinda:

- Ordena el tráfico de red de acuerdo a varios protocolos.
- Muestra el tráfico de red ordenados de acuerdo a varios criterios.
- Muestra estadísticas (gráficos) del tráfico de red.
- Almacena en disco duro las estadísticas obtenidas.
- Identifica los sistemas operativos de las computadoras.
- Muestra la distribución del tráfico *IP* en rangos de tiempo.
- Analiza el tráfico *IP* y lo ordena de acuerdo a origen/destino.
- Muestra la matriz del tráfico *IP*.
- Actúa como colector de *NetFlow/sFlow* recibiendo el flujo de los *routers* o *switches.*

2.7.6 Nagios

Es un sistema *open source* popular para monitorizar una red. Monitoriza los ordenadores y servicios que se especifiquen, alertando cuando el comportamiento de la red no es el deseado y nuevamente cuando vuelve a su estado correcto. A través de los ficheros de configuración se puede definir lo que se quiere monitorear y cómo se desea que se haga. Todo esto es posible a través de los diferentes conceptos que maneja:

- Host (computador): puede ser una estación de trabajo o un servidor
- Servicios: constituyen los objetivos a chequear en cierto *host*. Estos servicios pueden ser todos los ofrecidos en una red corporativa.
- Comandos: instrucciones que se emplean para detectar el estado de un servicio determinado en un *host* o grupo de *host*.
- Grupo: es una forma de agrupar servicios, *host* o comandos.

Además, *Nagios* cuenta con una interfaz web, a través de la cual se puede visualizar toda la información de la configuración y estado de los activos y servicios de la red. Esta información puede ser vista en formatos de tablas, gráficos así como ser exportada a formato *PDF*. Es importante especificar que una vez configurado correctamente, *Nagios* muestra un mapa organizado y detallado de la red de datos.

2.7.7 Otros

Es importante aclarar que *OSSIM* no solo integra aplicaciones que de una forma u otra tienen relación directa con la detección de eventos y anomalías, o tráfico de red. Esas otras aplicaciones se emplean para obtener una mayor funcionalidad en la gestión de toda la información recolectada por las herramientas de detección de eventos y anomalías.

Ejemplo de ellas se tiene a la librería *FPDF* como utilitaria para la exportación de información a formato *PDF;* y *phpGACL* para gestionar las listas de menú personalizados que se mostrarán a cada usuario que accederá al sitio web de *OSSIM*.

2.8 Ventajas y limitaciones.

El uso de la herramienta *OSSIM* permite tener de manera centralizada y desde una interfaz única el control y gestión de los eventos y sucesos de la red de datos. Además, se puede afirmar que desde su *framework* posibilita que las herramientas de monitoreo y gestión empleadas hasta el momento de forma separada, se complementen unas a otras, brindando así una mayor fiabilidad de los resultados que muestra. Otras de las ventajas significativas son:

- Fácil instalación y configuración del sistema.
- Alta fiabilidad de los mensajes, alertas, resultados y reportes que muestra.
- Facilidad para crear una arquitectura jerárquica de distribución de servidores.
- Posibilidad de su instalación en único servidor.
- Capacidad de delegar tareas a los “agentes”.
- Todas las herramientas que engloba son de código abierto.
- Bajo costo de instalación e implantación del sistema.
- Escalabilidad mediante *plugins*.
- Administración y trabajo desde cualquier lugar “autorizado” debido a su interfaz *web* para la interacción con el usuario.
- Obtención de los reportes en diferentes formatos de presentación como *PDF*, gráficos, *flash*.
- Interacción y soporte con la mayoría de dispositivos y aplicaciones más usadas en la prestación de servicios en centros de datos.

Hasta el momento, toda aplicación posee varias limitaciones y *OSSIM* no es la excepción, aunque éstas son poco significativas se relacionan a continuación:

- Requiere de cierto nivel “intermedio” de conocimiento en temas de administración de redes.
- Requiere de cierto nivel “intermedio” de conocimiento sobre el sistema operativo *GNU/Linux*.
- Puede llegar a ser un poco “tedioso” por la gran cantidad de funcionalidades que posee.

- Poco soporte al uso de *OSSIM* en otras distribuciones de *GNU/Linux* que no sean las recomendadas por el grupo que lo mantiene.

2.9 Conclusiones.

La arquitectura de *OSSIM* permite flexibilidad de uso y configuración del sistema, posibilitando esto una fácil adaptación a las características reales de las redes de datos. Además el flujo de procesos y datos que implementa dicho sistema para la gestión de los eventos y sucesos garantiza la fiabilidad de los resultados que muestra.

Capítulo 3. Propuesta para la implantación de *OSSIM*.

3.1 Introducción.

En este capítulo se describen los requisitos de *software* y *hardware* necesarios para el adecuado funcionamiento de la plataforma *OSSIM*. Además se hace una propuesta de implantación del sistema que detalla las características del *hardware* a emplear, especifica los pasos para la instalación y configuración y refiere una breve panorámica de cómo utilizar el sistema. También se pormenorizan las condiciones bajo las cuales el sistema *OSSIM* fue sometido a prueba y se da un resumen de los resultados obtenidos durante el periodo de experimentación.

3.2 Requisitos generales.

El grupo de desarrollo de *OSSIM* recomienda que para instalar dicha plataforma se realice en una distribución *GNU/Linux*, preferentemente *Debian* o *Fedora*. Además, plantea que no se instale interfaz gráfica en el servidor y que la administración se realice desde otra estación de trabajo, ya sea mediante la *web* o por conexión *SSH.*

3.3 Propuesta de hardware.

Para un buen funcionamiento de la plataforma *OSSIM* se recomienda la utilización de un servidor con las siguientes características:

Dispositivo	Características
Microprocesador	Tecnología doble núcleo a 2,4 GHz (u otro micro de prestaciones superiores)
Memoria RAM	2 GB DDR 2 a 667MHz
Disco duro	2 disco duros de 400 GB o más cada uno (para 10 años)
Tarjeta de red	Ethernet a 100 Mb/s

Tabla # 2. Propuesta de hardware.

3.4 Propuesta de software.

La plataforma *OSSIM* se puede instalar de dos maneras diferentes: instalando una a una las herramientas que engloba para luego integrarlas manualmente con el *framework*, y procediendo dicha instalación desde un disco compacto, preparado para instalar todo y dejarlo con la configuración básica

necesaria para el funcionamiento adecuado de la plataforma. Lo más recomendable y la propuesta que se hace es de la segunda forma mencionada y con la versión 1.4 de dicho disco de instalación. Lo que quedará instalado será la variante para servidor (sin interfaz gráfica) de la distribución *GNU/Linux Debian 4(Etch),* el *framework OSSIM* y todas las herramientas que éste integra.

3.5 Propuesta de despliegue.

Se propone que el servidor donde esté instalada la plataforma *OSSIM* se sitúe en el nodo principal y que esté enfocado a la gestión de los eventos y sucesos de esa subred. Aunque desde esta ubicación, el sistema monitorea toda la red, si las subredes contienen más de ciento cincuenta estaciones de trabajo es conveniente que se instale un "agente *OSSIM*" en la subred (puede ser en cualquier estación de trabajo sin grandes prestaciones) con el objetivo de que se encargue del procesamiento de las vulnerabilidades y envíe los datos correspondiente hacia el servidor central. Además, es necesario instalar en cada servidor los agentes correspondientes a las herramientas *OSSEC* (recolección de *logs*) y *OCS Inventory* (inventariado detallado del *hardware*). También se pueden instalar estos agentes en aquellas estaciones de trabajo que sean de especial interés.

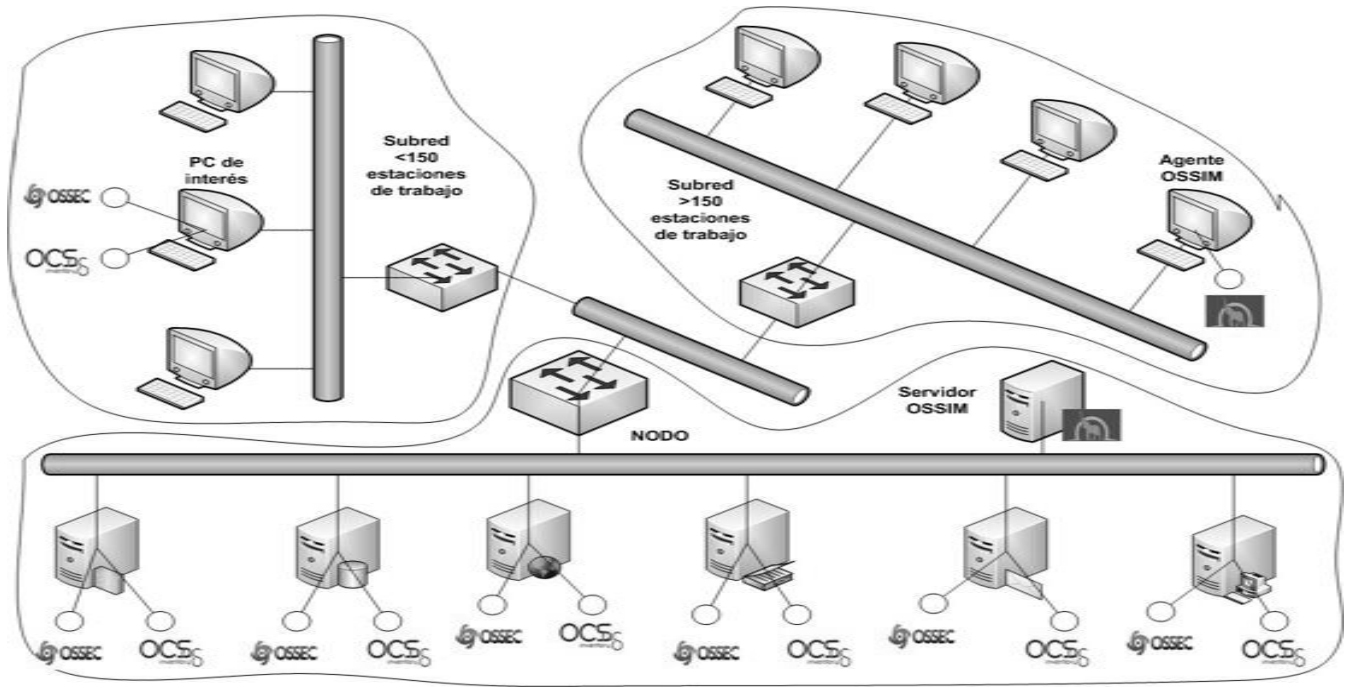

Figura # 8. Propuesta de despliegue.

3.6 Instalación.

La forma más común de instalar *OSSIM* es mediante el disco de instalación que se puede descargar desde el sitio oficial *http://www.ossim.net/download.php*

Una vez que se haya grabado en el disco compacto (*CD*) la instalación se procede a insertarlo en la torre de *CD* de la computadora y se reinicia el ordenador. *(Chequear que el orden de booteo de los dispositivos sea el adecuado para que inicie por la torre de CD, es decir, torre de CD-disco duro)*.

Si todo ha sido correctamente configurado se apreciará una pantalla con el logo de *OSSIM*, entonces se procede a realizar los pasos siguientes:

- Presionar tecla *INTRO* (*ENTER*).
- En la ventana que aparecerá, seleccionar idioma español (*Spanish*) y presionar *INTRO (ENTER)*.
- En la nueva ventana, seleccionar el país (*Sección Otro---Caribe---Cuba*) y presionar *INTRO (ENTER)*.
- Seleccionar la distribución del teclado de acuerdo al que se posea y presionar *INTRO (ENTER)*.
- Introducir el número *IP* que se desea que tenga la computadora (*Ej: 10.11.22.14*) y presionar *INTRO (ENTER)*.
- Introducir la máscara de subred (*Ej: 255.255.255.0*) y presionar *INTRO (ENTER)*.
- Introducir la puerta de enlace (*Ej: 10.11.22.254*) y presionar *INTRO (ENTER)*.
- Escribir el *IP* del *DNS* (*Ej: 10.0.0.10*) y presionar *INTRO (ENTER)*. Si es más de uno, se escriben separados por espacios. No se debe poner más de tres.
- Escribir el nombre de la computadora (*Ej: ossim_server*) y presionar *INTRO (ENTER)*.
- Escribir el dominio (*Ej: mi.dominio*) y presionar *INTRO (ENTER)*.
- Particionar el disco. Se recomienda que se haga el particionado de forma manual, creando tres particiones que deberían, preferentemente, tener las siguientes características básicas:

Partición	Sistema de ficheros	Capacidad	Punto de montaje
#1	ext2	60 MB	/boot
#2	ext3	10 GB	/
#3	ext3	resto	/var

Tabla # 3. Propuesta de particionado del disco duro.

Una vez creadas las particiones, seleccionar la opción de *"Escribir cambios en el disco duro"* y presionar *INTRO (ENTER)*.

- Escribir la contraseña para *root* (usuario administrador) y presionar *INTRO (ENTER)*.
- El sistema estará instalándose en el disco duro, aproximadamente unos diez minutos (se recomienda desconectar el cable de red).
- Sacar el *CD* de la torre, (conectar el cable de red si se desconectó en el paso anterior) y reiniciar.
- En el proceso de arranque el sistema automáticamente terminará de configurar todas las aplicaciones. Una vez que aparezca la opción de *loguearse* (escribir usuario y contraseña), se ha terminado la instalación del sistema.
- Para acceder y configurar la aplicación desde su interfaz web, desde otra estación de trabajo, abrir el navegador web preferido y escribir en la barra de direcciones: *http://IP_OSSIM_SERVER/ (Ej. http://10.11.22.14/)*

Los datos de usuario y contraseña para acceder son *admin*, *admin*; respectivamente. Es necesario cambiar dicha contraseña inmediatamente que se acceda al sitio. Para ello ir a: *Configuration---Users---[Change password].*

Es conveniente cambiar el idioma del sitio *web*, esto se puede realizar dirigiéndose a la sección: *Configuration---Main---[Language].*

3.7 Configuraciones necesarias.

Chequear que en la sección *Configuración---Principal* las configuraciones que están por defecto sean las correctas. *(En caso de tener dudas, no realizar ningún cambio).*

En caso de que más de una persona vaya acceder a la plataforma, es recomendable crear usuarios para cada una y darles los permisos adecuados a las tareas que realizarán, a través de la funcionalidad que ofrecida en *Configuración---Usuarios*. Además, es conveniente, configurar en *Configuración---Log de acción de usuario*, las tareas que se desean monitorear de las acciones de cada usuario sobre el sistema.

Es acertado que quede configurada la plantilla del correo que será enviado a los usuarios correspondientes cuando ocurran determinados eventos, para ello está la sección *Configuración---Plantilla de correo de incidencias*.

3.8 Panorámica de utilización de las funcionalidades del sistema.

Dashboard

OSSIM tiene una sección que se le puede catalogar como el termómetro del estado actual de la red de datos pues es una vista configurable que mediante gráficos estáticos y dinámicos muestra un resumen

de los eventos, incidencias, alarmas, tráfico de red y más que ha detectado la plataforma. Esta funcionalidad se encuentra en *Dashboard---Panel ejecutivo*.

Como complemento a la funcionalidad anterior, en *Dashboard---Alarmas*, se puede observar las alarmas generadas por el sistema. Esta vista permite hacer búsquedas avanzadas, así como seguir toda la información de donde procede la alarma. *(La funcionalidad de búsqueda avanza por varios criterios y aplicaciones de filtros, es común en cada una de las interfaces que muestran información recolectada o generada por el framework OSSIM; lo mismo sucede con la posibilidad de seguir hasta el origen todo lo referente a una información mostrada).*

Incidencias

Para conocer las incidencias de la red de datos, *OSSIM* presenta la sección *Incidencias---Incidencias*, donde aparecen agrupadas por estado. Además, se puede obtener un informe gráfico de dicha información con tan solo acceder a *Incidencias---Informe*.

Eventos

Un punto de partida importante para asegurar la red de datos es conocer las vulnerabilidades existentes las estaciones de trabajo, servidores y dispositivos de red; para esto se encuentra la sección *Eventos---Vulnerabilidades*. También es importante conocer las anomalías como cambios de sistemas operativos, cambio de puertos en los servicios o cambios de direcciones *MAC*; la localización de esta funcionalidad del sistema está en *Eventos---Anomalías*.

A través de *Eventos---RT Events*, *OSSIM* brinda la posibilidad de observar en tiempo real (*RT: real time*), los eventos que el *framework* va recolectando y procesando. Pero esto no es lo único que se puede realizar con los eventos pues si se desea ver el registro de todos los eventos se puede hacer mediante *Eventos---Event Viewer*.

Monitores

Siempre es muy útil poder observar como se ha comportado el tráfico de la red de datos así como pronosticar el comportamiento de dicho tráfico en el futuro y poder detectar alguna anomalía. Junto a lo anterior es de gran utilidad observar las sesiones de conexiones de red entre las distintas estaciones de trabajo y servidores existentes en la red de datos; mediante *Monitores---Red* y *Monitores---Sesion*, respectivamente, *OSSIM* provee las funcionalidades correspondientes para realizar lo antes expuesto.

La posibilidad de chequear en tiempo real el estado de los servicios que se brindan, así como el estado de funcionamiento de los diversos dispositivos y puestos de trabajo, y la variante de observar esto a través de una imagen dinámica denominada mapa de estado, la plataforma OSSIM la brinda en *Monitores---Disponibilidad*.

Informes

Todo sistema actual permite obtener informes mediante tablas, gráficos, paneles y más de las diferentes informaciones que gestiona. En el caso de *OSSIM*¸ muestra informes de las alarmas desglosadas por categorías y de la seguridad especificando detalles muy relevantes. Sin embargo el *framework* no se limita a mostrar dichos informes sino que es capaz de exportarlos a formato *PDF* dando la posibilidad al usuario de seleccionar que es lo que desea que se exporte hacia el documento *PDF*. Estas funcionalidades están en *Informes---Reporte de Alarmas*, *Informes---Reportes de seguridad* e *Informes---Reportes PDF*, respectivamente.

Además, se puede tener un informe de las computadoras analizadas y procesadas así como un inventario del *hardware* que poseen cada una de ellas. Las secciones *Informes---Equipos* e *Informes---OCS Inventory* son las que permiten acceder a lo descrito anteriormente.

Herramientas

Como estrategia de complementar y facilitar algunas tareas dentro de la plataforma *OSSIM*, las secciones *Herramientas---Análisis de red*, *Herramientas---Copia de seguridad* y *Herramientas---Log de usuario* brindan las posibilidades de escanear la red para obtener las estaciones de trabajo y poderlas gestionar luego, dígase agruparlas por subredes grupos de trabajo entre otros. También se puede realizar copias de seguridad así como observar un registro de todas las actividades llevadas a cabo por cada uno de los usuarios del sistema *OSSIM.*

Políticas

Para poder especificar las políticas de análisis de cada una de las estaciones de trabajo o grupo de estas, red o conjunto de subredes, mediante las secciones *Políticas---Equipos*, *Políticas---Grupo de equipos*, *Políticas---Red*, *Políticas---Grupo de redes*, el sistema permite realizar estas tareas con gran facilidad. *Si antes de configurar políticas en estas secciones, se ha utilizado la sección Herramientas—Análisis de red, el trabajo será más cómodo aún.*

El sistema permite, en la sección Políticas—Sensores, observar y establecer las políticas de trabajo de de cada uno de los sensores así como manipular el estado actual de los *plugins* de dichos sensores. Asimismo, en la sección *Políticas---Servidores* se llevan a cabo la misma labor que los sensores pero como su nombre lo indica, es con los servidores (se refiere a los servidores *OSSIM*).

Otra de las bondades del sistema *OSSIM* es la realización automática de ciertas acciones y respuestas ante la ocurrencia de determinados eventos, dicha funcionalidad se configura en las secciones *Políticas---Acciones* y *Políticas---Respuestas*.

Gracias a los *plugins*, el *framework* de *OSSIM* integra varias herramientas permitiendo una mejor gestión de los eventos y toda la información referente a los servicios, tráfico de red, estado de funcionamiento de estaciones de trabajo y dispositivos entre otros. El cómo funcionarán dichos *plugins*,

cuál será la información que recolectarán, cómo serán agrupados y más se puede especificar en la sección *Políticas---Grupos de plugins*.

Correlación

En esta sección se puede trabajar con las directivas de trabajo del *framework* así como ver las reglas de correlación cruzada entre las herramientas que están integradas a la plataforma *OSSIM*. Además, en la sección *Correlación---Backlog*, aparecen las incidencias, alarmas y eventos que han sido resultado del análisis de la información reunida por la mayoría de las herramientas agrupadas en el *framework*.

3.9 Resultados obtenidos tiempo de prueba.

Se hace necesario comentar sobre el rendimiento de desempeño de la plataforma *OSSIM*, para ello se hizo una prueba durante diez días bajo un entorno con las siguientes características:

- Una red *Ethernet* que trabaja bajo velocidades de hasta 100 Mb/s.
- Alrededor de trescientos estaciones de trabajos con sistemas operativos diferentes: *MS Windows XP* y *GNU/Linux* (*Gentoo, Ubuntu, Nova-FAR, FreeBSD*).
- Servidores brindando diferentes servicios como control de versiones, integración con *LDAP*, *firewall*, bases de datos, *proxy*, sitios *web*, *DNS*, entre otros.
- Características del ordenador utilizado como servidor *OSSIM*:
  - Microprocesador Intel *Pentium 4 Hyper-Threading* a 3GHz con cache L2 de 2MB.
  - Memoria *RAM* de 512MB.
  - Memoria *SWAP* de 1GB.
  - Disco duro *Serial ATA* de 160GB.
  - Tarjeta de red *Ethernet* a 1Gb/s.
- Se establecieron nueve subredes divididas en: ocho subredes con cantidades de estaciones de trabajo que oscilan entre treinta y cincuenta; y una pequeña subred con menos de diez estaciones.

Entre las principales funcionalidades que se configuraron para la realización de la prueba están:

- Análisis de vulnerabilidades.
- Detección de anomalías.
- Recolección de *logs*.
- Gestión de eventos y sucesos.
- Análisis del tráfico de red.
- Disponibilidad de los servicios.

- Generación de alarmas como resultado del procesamiento de la información y la aplicación automática de la correlación cruzada que presenta el sistema.

Los resultados obtenidos durante este periodo de prueba fueron muy convenientes de acuerdo a las prestaciones de la computadora utilizada como servidor de la plataforma *OSSIM*.

Se realizaron varios análisis de vulnerabilidades obteniendo los siguientes datos:

- Total de subredes analizadas: 9
- Total de estaciones analizadas: 2045
- Total de vulnerabilidades detectadas: 2180
- Promedio de vulnerabilidades por estación: 1.06
- Máxima cantidad de vulnerabilidades en una subred: 478
- Mínima cantidad de vulnerabilidades en una subred: 25
- Promedio de vulnerabilidades por subred: 242.22
- Máxima cantidad de vulnerabilidades en una estación: 28
- Mínima cantidad de vulnerabilidades en una estación: 0
- Promedio de vulnerabilidades en las veinte estaciones más vulnerables: 20.4

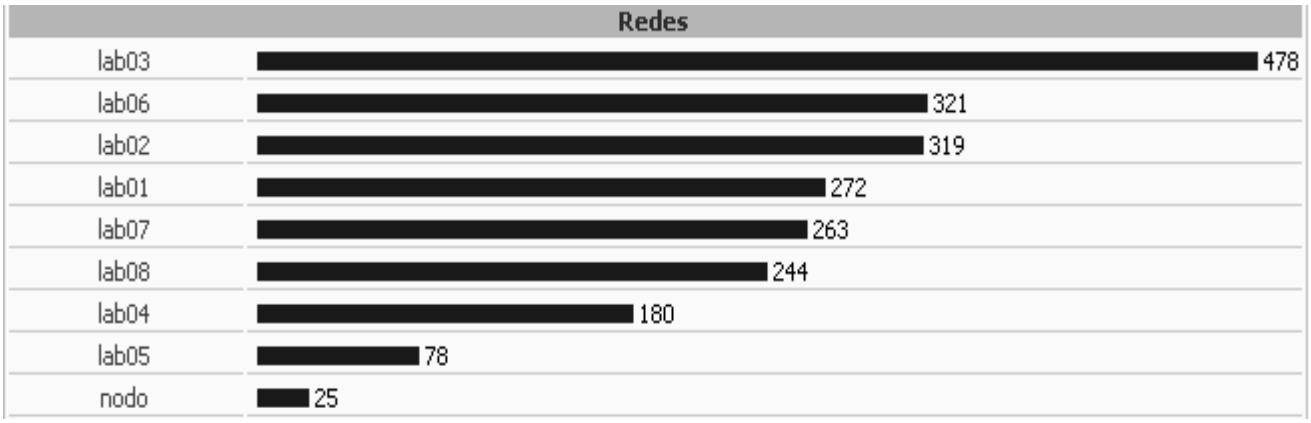

Figura # 9. Vulnerabilidades por subredes

Como parte de estos análisis se obtuvieron los siguientes gráficos como informe resumen:

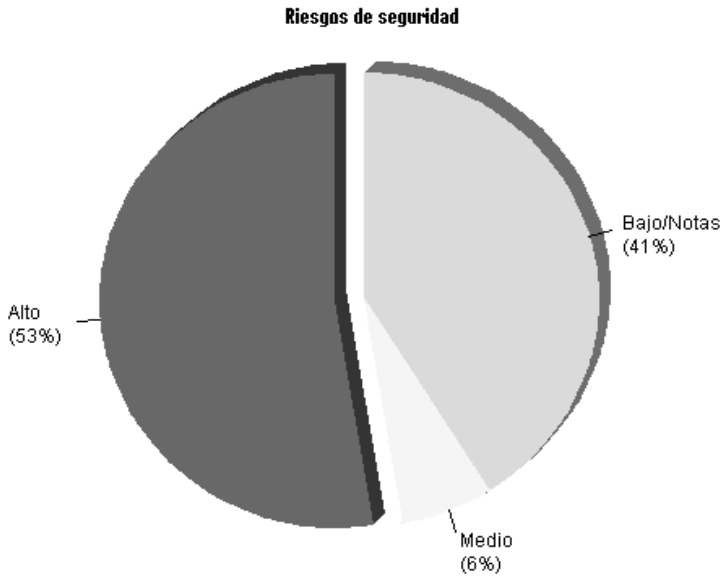

Figura # 10. Riesgos de seguridad.

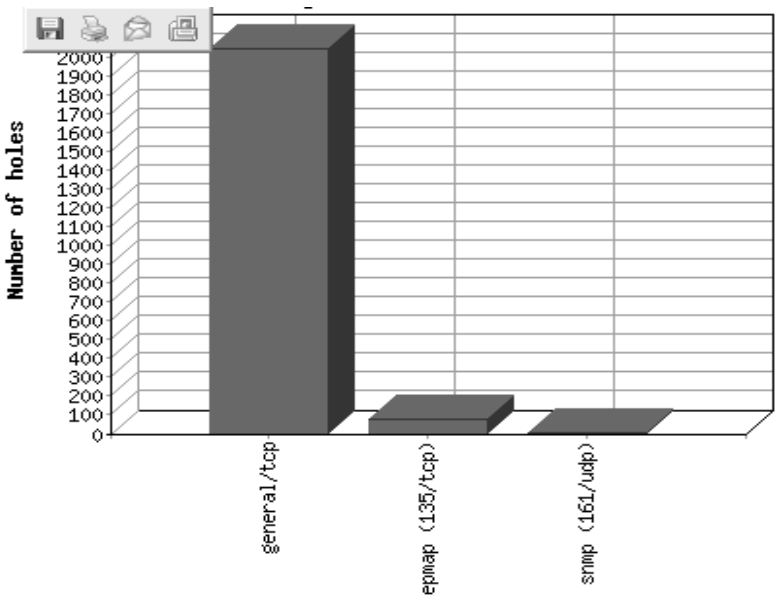

Figura # 11. Vulnerabilidades por protocolos.

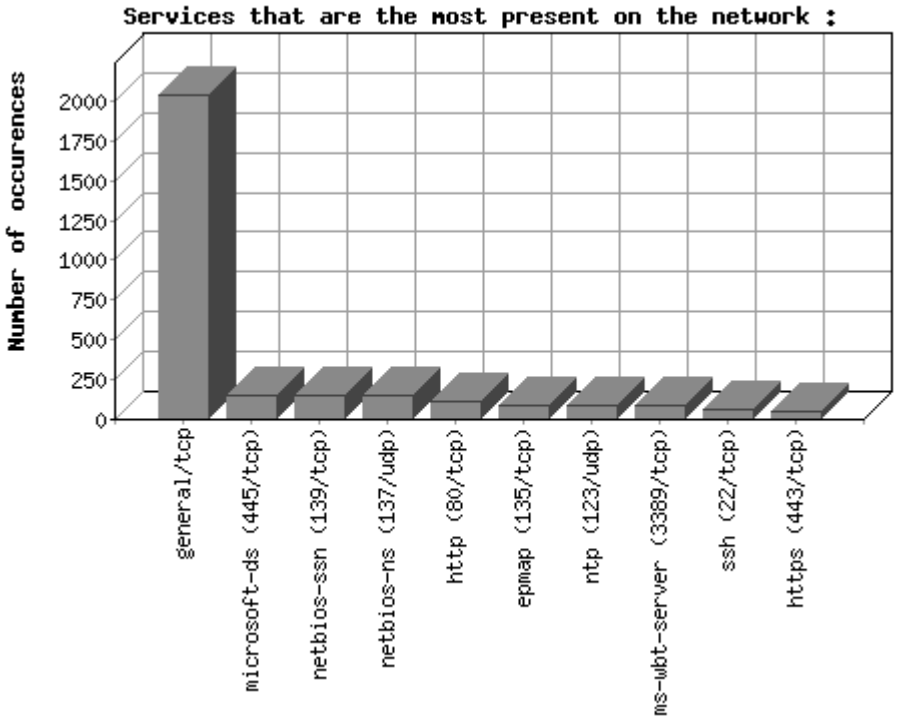

Figura # 12. Protocolos más utilizados en la red.

Los datos referentes a las anomalías, *logs*, eventos, incidencias y alarmas detectadas son:

- Total de anomalías: 13
  - Cambio de dirección *MAC*: 4
  - Cambio de sistema operativo: 7
  - Cambio de servicio: 2
- Total de eventos, sucesos y *logs*: 7712
  - Firewall: 139
  - Anomalías: 1077
  - Eventos Unix: 4434
  - Disponibilidad: 510
  - *Logs*: 1552
- Total de incidencias: 1866

    - Abiertas: 1605
    - Cerradas: 261
    - Vulnerabilidades: 1862
    - Genéricas: 4
- Total de alarmas: 4
- Total de *Backlog*: 12

Otros datos de interés son:

- Espacio ocupado en disco duro por los ficheros *logs*: 1.4GB
- Espacio ocupado en disco duro por la base de datos: 87MB
- Promedio porcentual de la utilización del microprocesador: 92%
- Promedio porcentual de la utilización memoria *RAM*: 98%
- Promedio porcentual de la utilización de la memoria *SWAP*: 87%
- Temperatura promedio del microprocesador: 69ºC
- Temperatura promedio del disco duro: 42ºC

Las pruebas realizadas demuestran que para un funcionamiento estable y confiable de la plataforma *OSSIM* durante un período de siete a diez años, así como la garantía del adecuado rendimiento de los dispositivos que conforman el ordenador que fungirá como servidor, es necesario los requerimientos de *hardware* descritos en el *epígrafe 3.3*.

3.10 Conclusiones.

A partir de las características del sistema y de los resultados obtenidos de la prueba realizada se hizo una propuesta factible para la implantación de *OSSIM* facilitando su rápido despliegue y empleo en los centros de datos.

Conclusiones

El empleo de una plataforma centralizada para la gestión de los eventos y sucesos de una red de datos proporciona simplicidad en la detección de anomalías y mal funcionamiento en los servicios que se brindan en los centros de datos.

La implantación del sistema *OSSIM* permite, a un bajo costo, poseer una infraestructura para el adecuado tratamiento de los eventos y sucesos relacionados con la red de datos. Además, el empleo de esta garantiza un alto grado de fiabilidad y calidad de los servicios que se ofertan.

Recomendaciones

- Implantar el sistema *OSSIM* en todo centro de datos que sea imprescindible la gestión de eventos y sucesos de la red de datos.
- Migrar la plataforma *OSSIM* a las distribuciones de *GNU/Linux Gentoo* o *Nova-FAR.*
- Crear un disco de instalación basado en las distribuciones de *GNU/Linux Gentoo* o *Nova-FAR* y que permita una mayor flexibilidad de configuración durante el proceso de instalación.

Bibliografía

1. **Lexico Publishing Group, LLC.** Dictionary.com. [En línea] [Citado el: 15 de enero de 2008.] http://dictionary.reference.com.

2. **Departamento de Sistemas Digitales. UCI.** Seguridad informática. *Conferencia: Introducción a la seguridad informática.* s.l. : UCI, 2007.

3. —. Seguridad Informática. *Conferencia: Sistemas de prevención de incidentes de seguridad.* s.l. : UCI, 2007.

4. *Oxford Advanced Learner´s Dictionary of current English.* s.l. : Oxford University Press, 2002.

5. **Stewart, Allen.** Importance of Understanding Logs from an Information Security StandPoint. [En línea] 2001. [Citado el: 15 de enero de 2008.] http://www.sans.org/rr/papers/33/200.pdf.

6. **Barret, B.** Home of Webalizer. [En línea] Iowa MR UNIX, 29 de noviembre de 2006. [Citado el: 15 de enero de 2008.] http://www.mrunix.net/webalizer/.

7. AWFFull | Dee and Steve´s Web. [En línea] 2005. [Citado el: 15 de enero de 2008.] http://www.stedee.id.au/awffull.

8. **Beermann, Cord.** Calamaris Home Page. [En línea] 2002. [Citado el: 15 de enero de 2008.] http://calamaris.cord.de/.

9. **Bauer, Kirk.** logwatch.org. [En línea] 23 de febrero de 2007. [Citado el: 15 de enero de 2008.] http://www2.logwatch.org/.

10. **Bali, Andras.** IPTQLOG Homepage. [En línea] 30 de 03 de 2002. [Citado el: 15 de enero de 2008.] http://drewie.host.sk/iptqlog/.

11. **InterSect Alliance.** InterSect Alliance - Open Source. [En línea] InterSect Alliance Pty Ltd, 1999-2007. [Citado el: 15 de enero de 2008.] http://www.intersectalliance.com/.

12. Sawmill- Universal Log File Analysis & Reporting. [En línea] Flowerfire, 2007. [Citado el: 15 de enero de 2008.] http://www.sawmill.net/.

13. **Free Software Foundation.** php-syslog-ng - Free Software Directory - Free Software Foundation. [En línea] Free Software Foundation, 7 de enero de 2008. [Citado el: 15 de enero de 2008.] http://directory.fsf.org/category/mon/.

14. *Análisis de ficheros log en GNU/Linux.* **Suárez, José Alberto.** 65,

15. loggrep - a tool for scanning logfiles. [En línea] sourceforge.net Inc. [Citado el: 15 de enero de 2008.] http://loggrep.sourceforge.net/.

16. **Dawson, Alex y Chadd, Adrian.** squid : Optimising Web Dilevery. [En línea] 21 de diciembre de 2007. [Citado el: 15 de enero de 2008.] http://www.squid-cache.org/.

Glosario de términos.

Término	Significado
DC	*Data Center* (Centro de datos).
24x7	24 horas al día por 7 día a la semana.
Apache	Servidor *web.*
AWK	Lenguaje de programación interpretado.
backdoor	Puerta trasera.
booteo	Secuencia de inicio en el arranque de una computadora.
BSD	*Berkeley Software Distribution* (Distribución de Software Berkeley)
byte	Unidad de medida relacionada con la capacidad de almacenamiento.
C	Lenguaje de programación.
C++	Lenguaje de programación.
CALM	Algoritmo de *OSSIM*.
CD	*Compact Disc* (Disco Compacto)
CGI	*Common Gateway Interface* (Interfaz de entrada común)
Corporation	Corporación.
CPU	*Central Procesor Unit* (Unidad Central de Procesamiento).
ddos	*Distributed Denial Of Service Attack* (Ataque de Denegación de Servicio Distribuido)
Debian	Distribución *GNU/Linux.*
DIDS	*Distributed IDS* (IDS distribuido)
DNS	*Domain Name Server* (Sevidor de nombres de dominio)
EDB	Base de datos de eventos (*OSSIM*).
exploit	Programa malicioso que se aprovecha de vulnerabilidades del sistema operativo.
FAR	Fuerzas Armadas Revolucionarias
Fedora	Distribución *GNU/Linux.*
finger	Obtención de información de usuarios en sistemas *Unix*
firewall	Cortafuegos.
flash	Conjunto de tecnologías multimedias.
fPDF	Librería para la creación de ficheros *PDF.*
framework	Marco de trabajo.
FreeBSD	Distribución *GNU/Linux* basada en Unix.
FSF	*Free Software Foundation*.
ftp	*File Transfer Protocol* (Protocolo para la transferencia de archivos)
GD Graphics	Librería para la creación de imágenes.
Gentoo	Distribución *GNU/Linux.*
GNU	Proyecto de desarrollo de *software* libre

Glosario de términos

GNU GPL	Licencia de *software* libre
GNU/Linux	Sistema operativo basado en Unix y resultado de la unión de los proyectos *GNU* y *Linux.*
HIDS	*Host IDS* (IDS basado en host)
host	Computadora.
html	*HyperText Markup Language* (Lenguaje de Marcado de Hipertexto)
http	*Hypertext Transfer Protocol* (Protocolo de transferencia de hipertexto)
https	*Hypertext Transfer Protocol Secure* (Protocolo seguro de transferencia de hipertexto)
IBM/AIX	Sistema operativo desarrollado por *IBM*
ICMP	*Internet Control Message Protocol* (Protocolo de mensajes de control de Internet)
IDS	*Intrusion Detection System* (Sistema de detección de intrusos)
IP	Dirección lógica de la tarjeta de red
IPTables	Herramienta empleada como cortafuegos.
ISO	*International Organization for Standardization* (Organización internacional para la estandarización)
IIS	*Internet Information Server* (Servidor de información de internet). Servidor de aplicaciones web desarrollado por *Microsoft*
KDB	Base de datos de framework (*OSSIM*)
kernel	Núcleo.
LDAP	*Lightweight Directory Access Protocol* (Protocolo de acceso a directorios)
libpcap	Librería para el trabajo con los paquetes de red (*GNU/Linux*)
libpng	Librería para el trabajo con imágenes en formato *png*
Linux	Kernel empleado en los sistemas operativos *GNU/Linux*
Lisp	Lenguaje de programación interpretado.
log	Fichero que contiene registro del comportamiento de programas o servicios.
logger	Programa para la obtención de *logs.*
mac addres	Dirección física de la tarjeta de red
Mac OS	Sistema operativo desarrollado por la compañía Apple Computer
Microsoft	Compañía que se dedica a la producción y venta de *software*
Microsoft SQL Server	Sistema Gestor de Bases de Datos desarrollado por *Microsoft*
Minix	Clon del sistema operativo *Unix*.
MS Windows	Sistema operativo desarrollado por la compañía *Microsoft*
Mysql	Sistema Gestor de Bases de Datos.
Nagios	Programa para el monitoreo de disponibilidad de servicios, dispositivos de red y computadoras.
nessus	Programa para el análisis de vulnerabilidades.
NIDS	*Network IDS* (IDS basado en red)

nmap	Programa para la exploración de la red
NovaLnx	Distribución *GNU/Linux* basada en *Gentoo.*
ntop	Herramienta que muerta el uso de la red.
open source	Código abierto
OpenBSD	Sistema operativo libre tipo *Unix.*
Oracle	Sistema Gestor de Bases de Datos.
OSSEC	Programa para la detección de intrusos basado en *host*
OSSIM	*Open Source Security Information Management* (Administración de información de seguridad de código abierto)
p0f	*Passive OS/network fingerprinting* (Obtención pasiva de información de usuarios)
P2P	*Peer To Peer* (Punto a Punto).
parser	Analizador sintáctico.
PCRE	*Perl Compatible Regular Expressions* (Expresiones regulares compatibles con Perl)
PDF	*Portable Document Format* (Formato de documento portátil)
Perl	Lenguaje Práctico para la Extracción e Informe. Lenguaje de programación.
PHP	Lenguaje de programación interpretado.
phpGACL	*PHP Generic Acces Control List* (Lista de control de acceso generico)
plug-ins	Aplicación informática que le da funcionalidad a otra.
PostgreSQL	Sistema Gestor de Bases de Datos.
proxy	Intermediario.
python	Lenguaje de programación.
ram	*Random Acces Memory* (Memoria de acceso aleatorio).
reference counting	Técnica para almacenar el número de referencias.
rootkit	Programa para ocultar otros programas.
router	Enrutador (dispositivo de red)
rrd	Herramienta para adquirir series de datos en el tiempo.
script	Conjunto de instrucciones.
Sed	Lenguaje de programación interpretado.
servidor	*Software* o computadora o la unión de ambos que brindan un servicio.
SGBD	Sistema Gestor de Bases de Datos.
Shell	Lenguaje de programación interpretado.
software	Programa, aplicación, sistema.
Software libre	Software libre de patentes privativas.
Solaris	Sistema operativo desarrollado por la Sun MicroSystems.
spamming	Efecto de enviar correos “basuras”
spoofing	Técnicas de suplantación de identidad.
SQL	*Structured Query Language* (Lenguaje estructurado de consultas)

squid	Programa que funge como servidor *proxy.*
ssh	*Secure Shell* (Terminal segura). Nombre del protocolo y programa que lo implementa.
swap	Sistema de ficheros
switch	Conmutador (dispositivo de red)
syslog-ng	Sistema para la recolección de *logs*
TCP	*Tansmission Control Protocol* (Protocolo Control de Trasmisión).
Transact-SQL	Lenguaje de programación de *SQL server*
Ubuntu	Distribución de *GNU/Linux.*
UCI	Universidad de las Ciencias Informáticas
UDB	Base de datos de perfiles (*OSSIM*).
UDP	*User Datagram Protocol* (Protocolo de Datagrama de Usuario)
Unix	Sistema operativo desarrollado por los laboratorios Bell de AT&T.
web	Documento de hipertexto y/o hipermedia.
winpcap	Librería para el acceso a los paquetes de red (*MS Windows*)
wu-ftp	Programa para la manipulación de ficheros log de un servicio *FTP*
xferlog ftp	Programa para la manipulación de ficheros log de un servicio *FTP*
zlib	Librería para la compresión de datos.

Tabla # 4. Glosario de términos.

Anexo #1. Fichero de configuración del *framework* (ossim.conf)

```
data_dir=/usr/share/ossim
base_dir=/usr/share/ossim/www
ossim_interface=eth0
ossim_link=/ossim/
adodb_path=/usr/share/php/adodb/

ossim_type=mysql
ossim_base=ossim
ossim_user=root
ossim_pass=nueva
ossim_host=localhost
ossim_port=3306

phpgacl_path=/usr/share/phpgacl/
phpgacl_type=mysql
phpgacl_host=localhost
phpgacl_base=ossim_acl
phpgacl_user=root
phpgacl_pass=nueva

email_alert=root@localhost
email_sender=ossim@localhost

  snort_type=mysql
  snort_base=snort
  snort_user=root
  snort_pass=nueva
  snort_host=localhost
  snort_port=3306
```

Anexo #2. Fichero de configuración del servidor (config.xml)

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

<config>
  <log filename="/var/log/ossim/server.log"/>
  <framework name="ossim" ip="127.0.0.1" port="40003"/>

  <datasources>
    <datasource name="ossimDS" provider="MySQL"
dsn="PORT=3306;USER=root;PASSWORD=nueva;DATABASE=ossim;HOST=localhost"/>
    <datasource name="snortDS" provider="MySQL"
dsn="PORT=3306;USER=root;PASSWORD=nueva;DATABASE=snort;HOST=localhost"/>
    <datasource name="osvdbDS" provider="MySQL"
dsn="PORT=3306;USER=root;PASSWORD=nueva;DATABASE=osvdb;HOST=localhost"/>
    <datasource name="ossecDS" provider="MySQL"
dsn="PORT=3306;USER=root;PASSWORD=nueva;DATABASE=ossec;HOST=localhost"/>
  </datasources>

  <directive filename="/etc/ossim/server/directives.xml"/>
  <scheduler interval="15"/>
  <server port="40001" name="ossim" ip="0.0.0.0"/>
</config>
```

Anexo #3. Fichero de configuración del agente *OSSIM* (config.xml)

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

<!DOCTYPE  config [

   <!-- Replace 127.0.0.1 with your sensor Ip -->
   <!ENTITY   sensor     "10.12.163.131" >

   <!-- Default network interface -->
   <!ENTITY    interface  "br0" >

   <!-- Default OSSIM database connection (db:host:dbname:user:pass) -->
   <!ENTITY    ossim_db   "mysql:localhost:ossim:root:tuxing" >

   <!-- Replace localhost with your server Ip -->
   <!ENTITY    serverip   "localhost" >

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

   <!-- plugins -->
   <!ENTITY    apache      SYSTEM '/etc/ossim/agent/plugins/apache.xml'>
   <!ENTITY    arpwatch    SYSTEM '/etc/ossim/agent/plugins/arpwatch.xml'>
   <!ENTITY    ca          SYSTEM '/etc/ossim/agent/plugins/ca.xml'>
   <!ENTITY    camonitor   SYSTEM '/etc/ossim/agent/plugins/camonitor.xml'>
   <!ENTITY    ciscoids    SYSTEM '/etc/ossim/agent/plugins/ciscoids.xml'>
   <!ENTITY    ciscopix    SYSTEM '/etc/ossim/agent/plugins/ciscopix.xml'>
   <!ENTITY    ciscorouter SYSTEM '/etc/ossim/agent/plugins/ciscorouter.xml'>
   <!ENTITY    fw1         SYSTEM '/etc/ossim/agent/plugins/fw1.xml'>
   <!ENTITY    heartbeat   SYSTEM '/etc/ossim/agent/plugins/heartbeat.xml'>
   <!ENTITY    iis         SYSTEM '/etc/ossim/agent/plugins/iis.xml'>
   <!ENTITY    iptables    SYSTEM '/etc/ossim/agent/plugins/iptables.xml'>
   <!ENTITY    juniperfw   SYSTEM '/etc/ossim/agent/plugins/juniperfw.xml'>
   <!ENTITY    nagios      SYSTEM '/etc/ossim/agent/plugins/nagios.xml'>
   <!ENTITY    nagios2     SYSTEM '/etc/ossim/agent/plugins/nagios2.xml'>
   <!ENTITY    netgear     SYSTEM '/etc/ossim/agent/plugins/netgear.xml'>
   <!ENTITY    netscreen   SYSTEM '/etc/ossim/agent/plugins/netscreen.xml'>
   <!ENTITY    ntop        SYSTEM '/etc/ossim/agent/plugins/ntop.xml'>
   <!ENTITY    ntsyslog    SYSTEM '/etc/ossim/agent/plugins/ntsyslog.xml'>
   <!ENTITY    opennms     SYSTEM '/etc/ossim/agent/plugins/opennms.xml'>
   <!ENTITY    osiris      SYSTEM '/etc/ossim/agent/plugins/osiris.xml'>
   <!ENTITY    p0f         SYSTEM '/etc/ossim/agent/plugins/p0f.xml'>
   <!ENTITY    pads        SYSTEM '/etc/ossim/agent/plugins/pads.xml'>
   <!ENTITY    postfix     SYSTEM '/etc/ossim/agent/plugins/postfix.xml'>
   <!ENTITY    prelude     SYSTEM '/etc/ossim/agent/plugins/prelude.xml'>
   <!ENTITY    realsecure  SYSTEM '/etc/ossim/agent/plugins/realsecure.xml'>
   <!ENTITY    rrd         SYSTEM '/etc/ossim/agent/plugins/rrd.xml'>
   <!ENTITY    snarewindows SYSTEM '/etc/ossim/agent/plugins/snarewindows.xml'>
```

```
    <!ENTITY    snort       SYSTEM '/etc/ossim/agent/plugins/snort.xml'>
    <!ENTITY    syslog      SYSTEM '/etc/ossim/agent/plugins/syslog.xml'>
    <!ENTITY    tcptrack    SYSTEM '/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="3600"/>

    <plugins>

<!-- &apache; -->
&arpwatch;
&ca;
&camonitor;
<!-- &ciscoids; -->
<!-- &ciscopix; -->
<!-- >&ciscorouter; -->
<!-- &fw1; -->
<!-- &heartbeat; -->
<!-- &iis; -->
<!-- &iptables; -->
<!-- &netgear; -->
<!-- &netscreen; -->
&ntop;
<!-- &ntsyslog; -->
<!-- &opennms; -->
<!-- &osiris; -->
&p0f;
&pads;
<!-- &postfix; -->
<!-- &prelude; -->
<!-- &realsecure; -->
&rrd;
<!-- &snarewindows; -->
<!-- &juniperfw; -->
&snort;
&syslog;
<!-- &tcptrack; -->
    </plugins>
</config>
```

Anexo #4. Fichero de configuración del agente *OSSIM* (config.cfg)

```
[daemon]
daemon=True
pid=/var/run/ossim-agent.pid

[event-consolidation]
by_plugin=1001-1150,1501-1550,4001-4010
enable=true
time=10

[log]
error=/var/log/ossim/agent_error.log
file=/var/log/ossim/agent.log
stats=/var/log/ossim/agent_stats.log
verbose=info

[output-csv]
enable=False
file="/var/log/ossim/agent-events.csv"

[output-db]
base=ossim_events
enable=true
host=localhost
pass=nueva
type=mysql
user=root

[output-plain]
enable=False
file=/var/log/ossim/agent-plain.log

[output-server]
enable=True
ip=10.12.163.131
port=40001

[plugin-defaults]
date_format=%Y-%m-%d %H:%M:%S
interface=eth0
ossim_dsn=mysql:localhost:ossim:root:nueva
sensor=10.12.163.131

[plugins]
arpwatch=/etc/ossim/agent/plugins/arpwatch.cfg
iptables=/etc/ossim/agent/plugins/iptables.cfg
nagios=/etc/ossim/agent/plugins/nagios.cfg
nmap=/etc/ossim/agent/plugins/nmap-monitor.cfg
```

```
ntop=/etc/ossim/agent/plugins/ntop-monitor.cfg
osiris=/etc/ossim/agent/plugins/osiris.cfg
ossec=/etc/ossim/agent/plugins/ossec.cfg
ossim-ca=/etc/ossim/agent/plugins/ossim-monitor.cfg
p0f=/etc/ossim/agent/plugins/p0f.cfg
pads=/etc/ossim/agent/plugins/pads.cfg
pam_unix=/etc/ossim/agent/plugins/pam_unix.cfg
rrd=/etc/ossim/agent/plugins/rrd.cfg
snare=/etc/ossim/agent/plugins/snare.cfg
snort=/etc/ossim/agent/plugins/snortunified.cfg
ssh=/etc/ossim/agent/plugins/ssh.cfg
sudo=/etc/ossim/agent/plugins/sudo.cfg

[watchdog]
enable=True
interval=30
restart_interval=3600
```

Anexo #5. Fichero de configuración de nagios (localhost.cfg)

```
define contact{
        contact_name                    nagios-admin
        alias                           Nagios Admin
        service_notification_period     24x7
        host_notification_period        24x7
        service_notification_options    w,u,c,r
        host_notification_options       d,r
        service_notification_commands   notify-by-email
        host_notification_commands      host-notify-by-email
        email                           nagios-admin@localhost
        }


define contactgroup{
        contactgroup_name       admins
        alias                   Nagios Administrators
        members                 nagios-admin
        }


define host{
        name                            generic-host
        notifications_enabled           1
        event_handler_enabled           1
          max_check_attempts            1
          notification_interval         0
        flap_detection_enabled          1
        failure_prediction_enabled      1
        process_perf_data               1
        retain_status_information       1
        retain_nonstatus_information    1
```

```
        notification_period             24x7
    register                    0
    }

define host{
        name                            linux-server
        use                             generic-host
        check_period                    24x7
        max_check_attempts              10
    check_command           check-host-alive
        notification_period             workhours
        notification_interval           120
        notification_options            d,u,r
        contact_groups                  admins
        register                        0
        }

define host{
    use                 linux-server
    host_name               ossim
    alias               ossim
    address                 10.12.163.131
        parents                         SwitchLab03
        check_command           check-host-alive
    max_check_attempts      10
    notification_interval   480
    notification_period     24x7
    notification_options    d,u,r
    }
define host{
    use                 linux-server
    host_name               SwitchNodo
    alias               SwitchNodo
```

```
        address                 10.12.170.254
        check_command           check-host-alive
        max_check_attempts      10
        notification_interval   480
        notification_period     24x7
        notification_options    d,u,r
        parents                 SwitchServer
}
define host{
            use                         linux-server
            host_name                   Saw
            alias                       Saw
            address                             10.12.170.1
            check_command                       check-host-alive
            max_check_attempts                  10
            notification_interval   480
            notification_period     24x7
            notification_options    d,u,r
            parents                             SwitchNodo
}

define host{
        use                     linux-server
        host_name               Firewall
        alias                   Firewall
        address                 192.168.8.2
        check_command           check-host-alive
        max_check_attempts      10
        notification_interval   480
        notification_period     24x7
        notification_options    d,u,r
}
```

```
define host{
        use                     linux-server
        host_name               Lab02
        alias                   Lab02
        address                 10.12.162.222
        check_command           check-host-alive
        max_check_attempts      10
        notification_interval   480
        notification_period     24x7
        notification_options    d,u,r
          parents                       SwitchLab02
}

define host{
        use                     linux-server
        host_name               NorgePC
        alias                   NorgePC
        address                 10.12.167.253
        check_command           check-host-alive
        max_check_attempts      10
        notification_interval   480
        notification_period     24x7
        notification_options    d,u,r
          parents                       SwitchLab07
}


define host{
        use                     linux-server
        host_name               Supanky
        alias                   Supanky
        address                 10.12.167.132
        check_command           check-host-alive
```

```
        max_check_attempts      10
        notification_interval   480
        notification_period     24x7
        notification_options    d,u,r
          parents                       SwitchLab07
}


define host{
        use                     linux-server
        host_name               SwitchLab01
        alias                   SwitchLab01
        address                 10.12.161.254
        check_command           check-host-alive
        max_check_attempts      10
        notification_interval   480
        notification_period     24x7
        notification_options    d,u,r
          parents               SwitchServer
}

define host{
        use                     linux-server
        host_name               SwitchLab02
        alias                   SwitchLab02
        address                 10.12.162.254
        check_command           check-host-alive
        max_check_attempts      10
        notification_interval   480
        notification_period     24x7
        notification_options    d,u,r
          parents               SwitchServer
}
```

```
define host{
        use                     linux-server
        host_name               SwitchLab03
        alias                   SwitchLab03
        address                 10.12.163.254
        check_command           check-host-alive
        max_check_attempts      10
        notification_interval   480
        notification_period     24x7
        notification_options    d,u,r
          parents                       SwitchServer
}

define host{
        use                     linux-server
        host_name               SwitchLab04
        alias                   SwitchLab04
        address                 10.12.164.254
        check_command           check-host-alive
        max_check_attempts      10
        notification_interval   480
        notification_period     24x7
        notification_options    d,u,r
        parents                 SwitchServer
}

define host{
        use                     linux-server
        host_name               SwitchLab05
        alias                   SwitchLab05
        address                 10.12.165.254
        check_command           check-host-alive
```

```
        max_check_attempts      10
        notification_interval   480
        notification_period     24x7
        notification_options    d,u,r
        parents                 SwitchServer
}

define host{
        use                     linux-server
        host_name               SwitchLab06
        alias                   SwitchLab06
        address                 10.12.166.254
        check_command           check-host-alive
        max_check_attempts      10
        notification_interval   480
        notification_period     24x7
        notification_options    d,u,r
        parents                 SwitchServer
}

define host{
        use                     linux-server
        host_name               SwitchLab07
        alias                   SwitchLab07
        address                 10.12.167.254
        check_command           check-host-alive
        max_check_attempts      10
        notification_interval   480
        notification_period     24x7
        notification_options    d,u,r
        parents                 SwitchServer
}
```

```
define host{
        use                     linux-server
        host_name               SwitchLab08
        alias                   SwitchLab08
        address                 10.12.168.254
        check_command           check-host-alive
        max_check_attempts      10
        notification_interval   480
        notification_period     24x7
        notification_options    d,u,r
        parents                 SwitchServer
}

define host{
        use                     linux-server
        host_name               SwitchServer
        alias                   SwitchServer
        address                 192.168.8.1
        check_command           check-host-alive
        max_check_attempts      10
        notification_interval   480
        notification_period     24x7
        notification_options    d,u,r
          parents                       Firewall
}

define hostextinfo{
          host_name             SwitchLab01
          icon_image            cook/network_switch.gif
          statusmap_image               cook/network_switch.gd2
}
```

```
define hostextinfo{
        host_name           SwitchLab02
        icon_image          cook/network_switch.gif
        statusmap_image         cook/network_switch.gd2
}

define hostextinfo{
        host_name           SwitchLab03
        icon_image          cook/network_switch.gif
        statusmap_image         cook/network_switch.gd2
}

define hostextinfo{
        host_name           SwitchLab04
        icon_image          cook/network_switch.gif
        statusmap_image         cook/network_switch.gd2
}

define hostextinfo{
        host_name           SwitchLab05
        icon_image          cook/network_switch.gif
        statusmap_image         cook/network_switch.gd2
}

define hostextinfo{
        host_name           SwitchLab06
        icon_image          cook/network_switch.gif
        statusmap_image         cook/network_switch.gd2
}

define hostextinfo{
        host_name           SwitchLab07
        icon_image          cook/network_switch.gif
```

```
        statusmap_image         cook/network_switch.gd2
}

define hostextinfo{
        host_name               SwitchLab08
        icon_image              cook/network_switch.gif
        statusmap_image         cook/network_switch.gd2
}

define hostextinfo{
        host_name               SwitchServer
        icon_image              cook/network_switch.gif
        statusmap_image         cook/network_switch.gd2
}

define hostextinfo{
        host_name               SwitchNodo
        icon_image              cook/network_switch.gif
        statusmap_image         cook/network_switch.gd2
}

define hostextinfo{
        host_name               Saw
        icon_image              ucid/linuxserver.gif
        statusmap_image         ucid/linuxserver.gd2
}

define hostextinfo{
        host_name               ossim
        icon_image              ucid/linuxserver.gif
        statusmap_image         ucid/linuxserver.gd2
}
```

```
define hostextinfo{
        host_name               Supanky
        icon_image              ucid/linuxpc.gif
        statusmap_image         ucid/linuxpc.gd2
}

define hostextinfo{
        host_name               NorgePC
        icon_image              ucid/linuxserver.gif
        statusmap_image         ucid/linuxserver.gd2
}



define hostextinfo{
        host_name               Firewall
        icon_image              ucid/firewall.gif
        statusmap_image         ucid/firewall.gd2
}

define hostgroup{
        hostgroup_name  test
        alias           Test Servers
        members         localhost
        }

define service{
        name                        generic-service
        active_checks_enabled           1
        passive_checks_enabled          1
        parallelize_check               1
        obsess_over_service             1
        check_freshness                 0
```

```
        notifications_enabled           1
        event_handler_enabled           1
        flap_detection_enabled          1
        failure_prediction_enabled      1
        process_perf_data               1
        retain_status_information       1
        retain_nonstatus_information    1
        is_volatile                     0
        register                        0
        }
define service{
          name                          local-service
          use                           generic-service
        check_period                    24x7
        max_check_attempts              4
        normal_check_interval           5
        retry_check_interval            1
        contact_groups                  admins
          notification_options          w,u,c,r
        notification_interval           60
        notification_period             24x7
        register                        0
          }

define service{
        use                     local-service
        host_name               localhost
        service_description     PING
          check_command                 check_ping!100.0,20%!500.0,60%
        }
define service{
          use                           local-service
          host_name                     Saw
```

```
        service_description             PING
        check_command                   check_ping!100.0,20%!500.0,60%
}

define service{
        use                     local-service
        host_name                   Saw
        service_description         BD
        check_command               check_pgsql!postgres
}

define service{
        use                     local-service
        host_name                   Saw
        service_description         HTTP
        check_command               check_http!3389
}

define service{
        use                             local-service
        host_name                       Supanky
        service_description             PING
        check_command                   check_ping!100.0,20%!500.0,60%
}
define service{
        use                     local-service
        host_name                   Supanky
        service_description         BD
        check_command               check_pgsql!postgres
}

define service{
        use                     local-service
```

```
        host_name               Supanky
        service_description     HTTP1
        check_command           check_http!3389
}

define service{
        use                     local-service
        host_name               Supanky
        service_description     HTTP2
        check_command           check_http!80
}

define service{
        use                     local-service
        host_name               NorgePC
        service_description     PING
        check_command           check_ping!100.0,20%!500.0,60%
}

define service{
        use                     local-service
        host_name               NorgePC
        service_description     BD
        check_command           check_pgsql!postgres
}

define service{
        use                     local-service
        host_name               NorgePC
        service_description     HTTP1
        check_command           check_http!3389
}
```

```
define service{
        use                     local-service
        host_name               NorgePC
        service_description     HTTP2
        check_command           check_http!80
}

define service{
        use                     local-service
        host_name               Firewall
        service_description     PING
        check_command           check_ping!100.0,20%!500.0,60%
}

define service{
        use                     local-service
        host_name               Lab02
        service_description     HTTP
        check_command           check_http!80
}

define service{
        use                     local-service
        host_name               ossim
        service_description     HTTP
        check_command           check_http!80
}

define service{
        use                     local-service
        host_name               SwitchLab01
        service_description     PING
        check_command           check_ping!100.0,20%!500.0,60%
```

```
}

define service{
        use                     local-service
        host_name               SwitchLab02
        service_description     PING
        check_command           check_ping!100.0,20%!500.0,60%
}

define service{
        use                     local-service
        host_name               SwitchLab03
        service_description     PING
        check_command           check_ping!100.0,20%!500.0,60%
}

define service{
        use                     local-service
        host_name               SwitchLab04
        service_description     PING
        check_command           check_ping!100.0,20%!500.0,60%
}

define service{
        use                     local-service
        host_name               SwitchLab05
        service_description     PING
        check_command           check_ping!100.0,20%!500.0,60%
}

define service{
        use                     local-service
        host_name               SwitchLab06
```

```
        service_description          PING
        check_command                check_ping!100.0,20%!500.0,60%
}

define service{
        use                     local-service
        host_name               SwitchLab07
        service_description          PING
        check_command                check_ping!100.0,20%!500.0,60%
}

define service{
        use                     local-service
        host_name               SwitchLab08
        service_description          PING
        check_command                check_ping!100.0,20%!500.0,60%
}

define service{
        use                     local-service
        host_name               SwitchNodo
        service_description          PING
        check_command                check_ping!100.0,20%!500.0,60%
}

define service{
        use                     local-service
        host_name               SwitchServer
        service_description          PING
        check_command                check_ping!100.0,20%!500.0,60%
}

define service{
```

```
        use                     generic-service
        host_name               SwitchLab03
          is_volatile                   0
          check_period                  24x7
          max_check_attempts                    4
          normal_check_interval                 8
          retry_check_interval          1
          notification_interval         120
          notification_period           24x7
          notification_options          c,r
        service_description         SNMP
        check_command                 snmp_tcpopen!.1.3.6.1.2.1.1.3.0
}

define service{
        use                     local-service
        host_name               localhost
        service_description         Root Partition
          check_command                         check_local_disk!20%!10%!/
        }

define service{
        use                     local-service
        host_name               localhost
        service_description         Current Users
          check_command                         check_local_users!20!50
        }

define service{
        use                     local-service
        host_name               localhost
        service_description         Total Processes
          check_command                         check_local_procs!250!400!RSZDT
```

```
        }

define service{
        use                     local-service
        host_name               localhost
        service_description     Current Load
          check_command                 check_local_load!5.0,4.0,3.0!10.0,6.0,4.0
        }
```